This document covers how to run the NXP Demo Experience while also covering the demos that are included, and how to operate them.
1 Introduction

The NXP Demo Experience is a user-friendly interface that allows you to launch preselected demonstrations included in the Linux Board Support Package (BSP) that NXP provides.

The NXP Demo Experience is for the ones who are interested in showcasing various features and capabilities of the SoCs provided by NXP. The demos included in this application are meant to be easy to run for users of all skill levels, making complex use cases accessible to anyone. Users need some knowledge when it comes to setting up equipment on Evaluation Kits (EVKs), such as changing Device Tree Blob (DTB) files.

This user guide is intended for end users of the NXP Demo Experience. This document covers how to run the NXP Demo Experience while also covering the demos that are included and how to operate them.

To use this software, users need at a minimum:

- A supported NXP Evaluation Kit (EVK)
- A display output (MIPI DSI or HDMI)
- A connected mouse

Note: Some demos require more than the minimum required equipment. To find the required materials for each demo, refer to Included demos chapter.

1.1 Installing the NXP Demo Experience

The NXP Demo Experience comes preinstalled on NXP-provided demo Linux images. These images can be found Embedded Linux for i.MX Applications Processors. Alternatively, a user can build the demo images, which include the NXP Demo Experience, by following the i.MX Yocto Project User’s Guide (document IMXLXYOCTOUG). In both cases, the imx-image-full image must be used.

2 Demo launcher

2.1 Graphical User Interface (GUI)

On boards where the NXP Demo Experience is available, an NXP Logo is displayed on the top left-hand corner of the screen. Users can start the demo launcher by clicking this logo.
After opening the program, users can launch demos using the following options shown in Figure 2 below:

1. To filter the list, select the icon on the left to expand the filter menu. From this menu, users can select a category or subcategory that filters the demos displayed in the launcher.

2. A scrollable list of all the demos supported on that EVK appears in this area with any filters applied. Clicking a demo in the launcher brings up information about the demo.

3. This area displays the name, categories, and description of the demos.

4. Clicking Launch Demo launches the currently selected demo. A demo then can be force-quit by clicking the Stop current demo button in the launcher (appears once a demo is started).

   **Note:** Only one demo can be launched at a time.
2.2 Text User Interface (TUI)

Demos can also be launched from the command line through log-in into the board remotely or using the onboard serial debug console. Keep in mind that most demos still require a display to run successfully.

*Note: If prompted for a login, the default user name is “root” and no password is required.*

To start the text user interface, type the following command into the command line.

```
# demoexperience tui
```
The interface can be navigated using the following keyboard inputs:

- **Up and down arrow keys**: Select a demo from the list on the left
- **Enter key**: Runs the selected demo
- **Q key or Ctrl+C keys**: Quit the interface
- **H key**: Opens the help menu

Demos can be closed by closing the demo onscreen or pressing the "Ctrl" and "C" keys at the same time.

## 3 Included demos

This chapter describes the available demos that can be launched from the NXP Demo Experience’s demo launcher. To see the available demos for a specific Linux version and board, refer to the [Demo Experience’s Release Notes](#).

**Note:** If the demo covers up the demo launcher window or another window that is required, drag the necessary windows, including video output windows, with the mouse.

### 3.1 Machine learning demos

The following demos show machine learning use cases that are possible with the Neural Processing Unit (NPU) included on-chip.

#### 3.1.1 NNStreamer demos

NNStreamer is a set of GStreamer components that enable machine learning in video pipelines. The included demos use NNStreamer to create video outputs that overlay inference information onto the video.
All NNStreamer demos have identical user interfaces. When these demos launch, users are presented with a control panel, as shown in Figure 4.

1. This button can be used to quit the demo and stop video playback.
2. Various options can be set before a demo is run:
   - **Source**: Select the camera to use or to use the example video provided (requires an Internet connection, not available in "Brand Detection" demo).
   - **Backend**: Select whether to use the NPU (if available) or CPU for inferences.
   - **Height**: Select the input height of the video if using a camera.
   - **Width**: Select the input width of the video if using a camera.
   - **Label Color**: Select the color of the overlay labels.
3. Clicking the **Run** button locks the current settings and starts the camera feed and inferencing.
4. While the video plays, statistics about the current speed of the video feed and inferencing are displayed here. Video refresh represents the speed of the video inferencing pipeline. The inference time represents the time that it takes to complete one inference.

**Note**: If the NPU is selected, the first time a demo is run, the NPU does a process called "warming up". Here, the NPU must convert the model file into something it can read. While this result is cached to speed up future runs, this process can take a while.

### 3.1.1.1 Object classification

**Required materials**: Mouse, display, camera (if tested with camera), and Internet connection (if using example video)
This demo launches a GStreamer pipeline that gets a video feed from a camera or video file and runs a classification inference on the video frames as they come in. This demo uses a pretrained quantized MobileNet V1 TFLite model that is trained on objects included in the ImageNet Large-Scale Visual Recognition Challenge 2012 (ILSVRC2012) object set. The result of the inference is then displayed within the video frame.

When the demo starts, a video overlay of the following information is shown:

1. The label with the highest probability of being in the image.
2. Frames Per Second (FPS) of the video inferencing pipeline and Inferences Per Second (IPS) based on the time it takes to complete one inference.

### 3.1.1.2 Object detection

**Required materials:** Mouse, display, camera (if tested with camera), and Internet connection (if using example video)

This demo launches a GStreamer pipeline that gets a video feed from a camera or video file and runs a detection inference on the video frames as they come in. This demo uses a pretrained quantized MobileNet Single Shot Detection (SSD) V2 TFLite model that is trained on objects included in the Common Objects in Context (COCO) object dataset. The result of the inference is then displayed within the video frame.
When the demo starts, a video overlay of the following information is shown:

1. Detected objects have boxes drawn around them, and their labels are displayed near the top left corner.
2. FPS of the video inferencing pipeline and IPS based on the time it takes to complete one inference.

3.1.1.3 Pose detection

**Required materials:** Mouse, display, camera (if tested with camera), and Internet connection (if using example video)

This demo launches a GStreamer pipeline that gets a video feed from a camera or video file and runs a pose detection inference on the video frames as they come in. This demo uses a pretrained quantized PoseNet ResNet-50 TFLite model that is trained on 17 body points. When the video starts, the detected pose is overlaid onto the incoming video feed.
3.1.1.4 Brand detection

**Required materials:** Mouse, display, camera (if tested with camera), and Internet connection (if using example video)

This demo launches a GStreamer pipeline that gets a video feed from a camera and runs a brand detection inference on the video frames as they come in. This demo uses a pretrained quantized Popular Products V1 TFLite model that is trained on 100,000 US products. The result of the inference is then displayed within the video frame.
When the demo starts, a video overlay of the following information is shown:

1. The label with the highest probability of being in the image.
2. FPS of the video inferencing pipeline and IPS based on the time it takes to complete one inference.

### 3.1.1.5 ML gateway

**Required materials:** Mouse, display, and camera for each device. Two devices that are connected to the same network are required.

Machine Learning (ML) gateway allows devices with limited ML processing power to use the resources of another much more powerful device to accelerate ML inferences. This demo allows users to create an ML gateway on the i.MX 8M Plus EVK and have other devices connected to it, such as the i.MX 8M Mini EVK. Additionally, the server broadcasts its IP address so clients can connect to the gateway effortlessly. An NNStreamer pipeline is used to collect the data from the client, process it on the ML gateway (inference), and return the results back to the client. The client interprets the results from the ML gateway (decoding stage of bounding boxes for this demo) and shows them on the display. This demo requires two devices, both connected over a strong Internet connection.
When the demo starts, users must pick whether the device is a server or a client.

• If the device is acting as a server, select **Set up a server**…
• If the device is acting as a client, select **Set up a client**…

### 3.1.1.5.1 Setting up a server

Before the server starts up, it displays the current IP address next to server IP. By selecting the appropriate option from the drop-down, the server enables users to choose whether to perform inferences on the CPU or the NPU (if available). When ready, click the **Start Server** button. If the server is set up successfully, the button changes to Server is running.
Figure 10. ML gateway server setup
3.1.1.5.2 Setting up a client

When setting up a client, the device looks for a server IP and, when found, displays that as an option to pick in the next window. Users can also type in their own IP address. When ready, click the **Connect to Server** button. The device connects to the server and displays a video output with the detected objects marked.

![ML Client Setup](image_url)

*Figure 11. ML gateway client setup*
3.1.2 PyelQ demos

PyelQ is a suite of demos that demonstrate machine learning use cases in various applications. Most applications are written in Python, which is why the suite is named PyelQ. Instead of analyzing a video stream, these demos run inferences on photographs.

When starting the demo from the demo experience launcher, all demos run a launcher before running the demos. It ensures that PyelQ is installed and sets up the demo.
Note: These demos require PyelQ version 3.0.1. If another version is installed, uninstall PyelQ before running the demos below.

3.1.2.1 Object classification

**Required materials:** Mouse, display output, and an Internet connection

This demo installs PyelQ if not installed and runs an object classification inference on an image. This demo uses a quantized MobileNet V1 model trained on objects included in the ILSVRC2012 object set.
3.1.2.2 Object detection

**Required materials:** Mouse, display output, and an Internet connection

This demo installs PyeIQ if not installed and runs an object detection inference on an image. This demo uses a quantized MobileNet Single Shot Detection (SSD) V1 TFLite model trained on the COCO object set.
3.1.2.3 Mask detection

**Required materials:** Mouse, display output, and an Internet connection.

This demo installs PyeIQ if not installed and runs a mask detection inference on an image. This demo uses a model trained to detect if a user is wearing a mask or not.

![Figure 15. PyeIQ object detection example](image-url)
3.1.3 OpenCV demos

The following demo uses the OpenCV library to display video frames with inference data displayed on them.

3.1.3.1 Face recognition

**Required materials:** Mouse, display output, camera, and an Internet connection.

In this demo, users can register and recognize faces in a video feed. The demo first uses a quantized TFLite SSD MobileNet V2 model to detect the faces that are in the video frame. With those faces, it then crops the faces and uses a quantized TFLite FaceNet model to create face embeddings. These face embeddings can be used to compare against faces saved previously.
When the demo starts, a new window is displayed that allows the user to set face recognition options:

1. This button can be used to quit the demo.
2. These options allow certain aspects of demos to be changed:
   • **Source:** Select the camera to use.
   • **Backend:** Select whether to use the NPU (if available) or CPU for inferences.
3. This button confirms the settings and starts the demo.
4. This area displays the status updates while the demo is starting.

When the Run button is pressed, the demo first downloads the required model from the Internet and then "warm-up" the NPU. This warm-up time can take a couple of minutes the first time the demo is run. In future runs, the warm-up time takes less time.
When the demo is ready to begin, the following window opens.

1. A camera view with inference information layered over the frame.
2. A control panel.

The camera view window has the following information:
1. Any faces that are found are outlined with a box. A red box means that no face is saved with a similarity higher than the recognition cutoff (%) when compared to the face in the box. A green box means that a face is saved with a similarity higher than the recognition cutoff (%) when compared to the face in the box. If the face detected is similar to any of the registered faces, the name of this registered face is displayed above the bounding box, together with the percentage similarity. Otherwise, the “Not found” label is displayed instead, meaning there is no match.

2. The following statistics are shown:
   - **Recognition time**: The IPS and inference time in milliseconds it took to generate a face embedding for the faces in the picture. Since there can be multiple faces in the frame, the total time for all faces varies based on the number of faces in the frame.
   - **Detection time**: The IPS and inference time in milliseconds that it took to find all the faces in the picture.
   - **Overall time**: The IPS and inference time in milliseconds that it took to prepare a video frame from when the application received it.

![Figure 20. Face recognition control panel](image)

The control panel has the following controls to change the behavior of the demo:

- **Detection Cutoff (%):** It sets the percentage confidence required to detect a face in a video frame. Slide this option up if other objects are being detected as faces or down if faces are not being detected.
- **Recognition Cutoff (%):** It sets the percentage similarity required to recognize a face in a video frame. Slide this option up if the demo is falsely identifying faces or down if the demo does not identify the face.
- **Register Face:** It registers new faces to the database to be recognized. This is done locally and is removed when the application closes.

**How to register a face:**
1. Click the **Register Face** button on the control panel.
2. A countdown from 5 seconds starts. At the end of the countdown, the application saves the current frame and detects any faces in the frame.
3. For all faces in the frame, a window prompt appears, as shown in Figure 21. Type the name of the face that appears in the blue box in the camera view. When done, click the Register Face button. If a face should not be registered, click the Skip Face button.

Note: Not always able to identify faces? Registering the same face multiple times helps increase the data pool the database has to pull from.

Figure 21. Face recognition register face

3.1.4 TFLite demos

TensorFlow Lite (TFLite) demos are basic demos that show the basic functionality of the TensorFlow library.

3.1.4.1 ML benchmark

Required materials: Mouse, display, and Internet connection.

When running the ML benchmarking demo, two machine learning model files representing the same model are run on both the CPU and NPU. Before running, the demo attempts to download these models from the Internet. After a successful download, the models are run, and the times are compared on the screen. Currently, the only model that can be run is a pretrained quantized MobileNet V1 TFLite model.

3.1.4.2 Driver Monitoring System (DMS) demo

Required materials: Mouse, display, camera, and Internet connection.

Newer cars today have advanced navigation systems that can even drive the car for the user. While these systems are tested, no system is perfect and may require the driver to intervene. To ensure safe operation of these advanced systems, car manufacturers may implement smart safety systems to keep everyone on the road safe. One of these systems is a Driver Monitoring System (DMS). DMS watches the driver to ensure that they have their eyes on the road and are not displaying symptoms of fatigue. This demo shows an example implementation of a DMS demo. A camera and Internet connection are required for this demo.

When starting the demo, users can select a video device to use for the demo. Select the correct device from the dropdown menu and click “Start” to start the demo. The options window downloads any required files and starts the camera feed. On the left bar, the user’s attention, doziness, and yawn status can be monitored. At the top, there is a large colorful bar that increases if the user does unsafe activities (leaves the frame, looks away, falls
asleep) and decreases if the driver is safe. On the right, the driver detected has a box around their face. The penalties for these unsafe actions can be adjusted by clicking the button on the top left of the window.

### 3.1.4.3 Selfie segmentation

**Required materials:** Mouse, display, camera, and Internet connection.

This demo uses a machine learning model for semantic segmentation to key out the background of a video feed and replace it with a picture of a conference room. This demo is intended to work with people as the foreground object. People too far away from the camera may result in bad segmentation. A camera and Internet connection is required for this demo.

![Segmentation options window](image)

**Figure 22. Segmentation options window**

First, select the source that should be used for the demo. When done, click run to start the demo.
When the demo starts, a video feed appears with the background looking like a conference room. People in the video frame should be the only objects shown in the frame. At the top left part of the screen, users can see runtime preference numbers, like the frames per second (FPS) and inference time (Inf).

3.2 Multimedia demos

This demo package has multimedia demos related to GStreamer, Image Signal Processing (ISP), and audio.

3.2.1 GStreamer demos

GStreamer is a flexible tool that allows users to build and deploy video pipelines quickly. Below are demos that show examples of how to use GStreamer.

3.2.1.1 Video test demo

Required materials: Mouse, display output, and a camera (optional)

This demo allows the user to test display outputs and cameras connected to the camera. When launched, users see the GUI shown in Figure 24, where they can launch GStreamer pipelines.

Note: The GStreamer windows do not have toolbars that allow users to close and move the window. To move windows, click anywhere in the video output area and drag the window to the correct area on the screen. The GStreamer windows are closed when the demo quits.

The video test demo launches a GStreamer pipeline that plays a test source on the screen.
The following options are available to users while running the demo:

1. This button quits the demo and closes all GStreamer windows that are running at the current time.
2. These options allow users to control what the video output looks like:
   - **Source**: Allows users to pick either a test source or a camera source to be shown in the output.
   - **Resolution**: Allows users to change the video resolution of the output video.
   - **Scale to output**: Allows users to scale the video up to fill the entire screen. For example, if 720x480 is selected as the resolution and the display is 1920x1080, the video output first is captured at or scaled down to keep the same format for consistency: "720x480" and then scaled up to keep the same format for consistency: "1920x1080".
3. This button starts a video pipeline with the settings that the user has selected.

*Note: The current application only allows cameras to be used for a single video stream at a time. Also, make sure to select a supported resolution for the camera being used.*

### 3.2.1.2 Camera using the VPU

**Required materials**: Mouse, display output, and camera

This demo launches a GSTreamer pipeline that encodes a video feed from a camera, then decodes and displays the decoded video on the screen. The encode and decode are done using the on-chip Video Processing Unit (VPU).
3.2.1.3  Multi-camera preview

Required materials: Mouse, display output, OV5640 camera, and Basler camera

Supported cameras:
• Basler camera (<EVKNAME>-evk-basler-ov5640.dtb)

This demo launches a GStreamer pipeline that displays two camera feeds simultaneously on the screen.

3.2.2  Image Signal Processor (ISP) demos

Some SoCs have built-in ISPs that allow on-chip image control similar to the one in a DSLR camera. The demos below show ways to use this powerful tool.

3.2.2.1  ISP control demo

Required materials: Mouse, display output, and a supported camera

This demo launches a GStreamer pipeline that displays the current video output and a window that allows users to change and manipulate the video feed using API calls to the ISP.

Supported cameras:
• OS08A20 (<EVKNAME>-evk-os08A20)
• Basler camera (<EVKNAME>-evk-basler.dtb)

![Figure 25. ISP control demo example](image)

The following options can currently be changed through the UI:
• Black level subtraction (red, green.r, green.b, blue)
• Dewarp
  – Dewarp on/off
  – Change dewarp mode
– Vertical and horizontal flip
• FPS limiting
• White balance
  – Auto white balance on/off
  – White balance control (red, green.r, green.b, blue)
• Color processing
  – Color processing on/off
  – Color processing control (brightness, contrast, saturation, and hue)
• Demosaicing
  – Demosaicing on/off
  – Threshold control
• Gamma Control
  – Gamma on/off
  – Gamma mode (logarithmic or equidistant)
• Filtering
  – Filter on/off
  – Filter control (denoise and sharpness)

Figure 26. ISP control demo control panel

The ISP can be controlled using the control panel that appears after the video is started.

1. This button can be used to quit the demo and stop video playback.
2. This button shows a log of the API calls used when changes are made from the control panel.
3. This drop-down allows the user to swap between different control panel sections.
4. This area includes the settings and options for the current section. Changes to these settings change the video feed playing in the background and then revert when the demo is exited.

3.2.2.2 Video dump demo

Required materials: Mouse, display output, external drive, and supported camera

Supported cameras:
- Basler camera (<EVKNAME>-evk-basler.dtb)

Figure 27. Video dump example

This demo allows users to dump raw camera frame data onto a connected drive. It also allows you to set some settings before starting the frame dump process:

1. This button can be used to quit the demo.
2. This area selects the camera to use and loads it into the application. To load the camera, select the camera from the Camera drop-down and click Load Camera. It gets the supported modes and tests whether the

/dev/video2 loaded!
camera can be run with this demo. Once the camera is loaded in, the other options in this window unlock. If the camera cannot be loaded, an error pop-up occurs.

3. Here users can set the settings about the frames that get saved onto the drive:
   - **Mode**: Select the mode the camera is in. It typically changes the height, width, FPS, or other camera features like HDR.
   - **Format**: Select the format the raw frame data is saved in. The options are RAW12, RAW10, RAW8, NV16, NV12, and YUYV.
     
     **Note**: All formats may not be supported for all cameras.
   - **Postprocessing**: Allows the user to set modifications to the image if supported. "Crop" crops a frame while "scale" scales the frame to the selected size.
   - **Width**: Shows the width of the selected mode. If postprocessing is enabled, it allows users to change the width of an image.
   - **Height**: Shows the height of the selected mode. If postprocessing is enabled, it allows users to change the height of an image.
   - **FPS**: The frames per second of a selected mode. It is not how quickly frames are dumped into storage.
   - **Save to... (it missed one)**: Allows the user to pick a save location for the raw frame data.

4. This button starts the process of getting raw frame data and dumping it to the selected save location. Clicking this button again after the process is started causes the program to stop dumping frames and saves the data.

5. The status bar indicator shows updates on what is happening currently with the demo.
   
   **Note**: Raw image data are large files that most image viewers cannot interpret. It is not meant to be a way to save processed images onto a drive.
   
   **Warning**: Do not disconnect the drive early! Disconnecting the drive early may result in the data saved onto the drive becoming corrupted.

### 3.2.3 Audio

A few audio demos have also been included to test audio equipment. These tests have no graphical output.

#### 3.2.3.1 Audio record

**Required materials**: Mouse, display, and microphone

This test records an audio file from the headphone input with a 10 second duration.

#### 3.2.3.2 Audio play

**Required materials**: Mouse, display, and headphones

This test plays the audio file recorded on the audio record test.

**Note**: This demo only works if audio record is run first.

### 3.2.4 Voice

Just like a mouse and keyboard, voice is a way for us to interact with edge devices. These demos show off NXP’s voice technologies such as VoiceSeeker, VoiceSpot, and VIT.

**Note**: A valid 8-mic array board is required to be attached and set-up for these demos to be used correctly. To acquire one, see [https://www.nxp.com/part/8MIC-RPI-MX8#](https://www.nxp.com/part/8MIC-RPI-MX8#).
3.2.4.1 i.MX voice control

You can control the NXP Demo Experience with the power of your voice. This demo uses the trial version of VoiceSpot (locked to "Hey NXP!" as the wake word) and VoiceSeekerLite (no echo cancelation) to open and close some of the demos in the NXP Demo Experience. The voice commands are all processed locally, meaning no cloud service is required.

**Note:** This demo changes the file at `/etc/asound.conf`. The demo attempts to undo the changes if it closes normally. If the demo crashes or is stopped through another method other than the cross in the top right-hand side of the window, the original file can be found at `/etc/asound_org.conf` as of the last time the demo was run.

Figure 28. i.MX voice control

The first window asks if the user would like to use the Cortex-M core for low-power voice operations. To set this up, follow the instructions in the i.MX Linux User’s Guide section 8.3.4 so that low-power voice is running on the Cortex-M core.

**Note:** These settings are not enabled by default and the application cannot check if the Cortex-M core is running correctly. Enabling this setting without the proper setup may require you to manually power cycle the EVK.
The demo first attempts some setup steps before it starts listening for commands. If there are any errors, the setup stops, and the error displays. Once the demo is successfully set up, the wake word and the list of commands get listed on the left-hand side of the window.

To wake up the board, say "Hey NXP!". When the status indicator switches to "Listening…", say the command that you wish to run. The demo then launches (might take a couple of seconds). To stop the demo, say "Hey NXP!" and say another command or "Stop Demo".

When the Cortex-M core setting is enabled, saying “Suspend” causes the EVK to put the Cortex-A core into suspend mode. This causes the screen to go black and the mouse and keyboards to become unresponsive. To take the Cortex-A core out of suspend mode, simply say "Hey NXP!" and the screen and controls should reappear.

### 3.2.4.2 i.MX Multimedia Player

The i.MX Multimedia Player is a demo application based on NXP Voice Intelligent Technology (VIT). VIT is a free library that provides voice recognition and integrates a complete audio front end, wake-word engine, and voice-command solution to control IoT devices.

The Multimedia Player application uses Bluetooth to play back audio and controls Bluetooth commands using VIT voice commands. The application requires the 8-Microphone Array Board (8MIC-RPI-MX8) installed on the i.MX hardware for voice enablement. The 8MIC-RPI-MX8 requires to set the FDTFILE to a proper 8-microphone board revision DTB in the uBoot environment. Visit [https://www.nxp.com/part/8MIC-RPI-MX8#](https://www.nxp.com/part/8MIC-RPI-MX8#) for details.
To run the demo, launch the i.MX Multimedia Player from the NXP Demo Experience GUI. It takes a few seconds to load the application and dependencies. Once the i.MX Multimedia Player is launched, use a smartphone or a tablet to search for the "i.MX-MultimediaPlayer" Bluetooth device and pair it.
Once a Bluetooth device is connected, you can start controlling audio playback using voice commands. Use the "Hey NXP" wake word to let VIT wake up the device and then a voice command to control the audio playback. When a Bluetooth connection is established you can see the audio metadata displayed in the demo GUI. You can control the audio using the "Pause, Next Song, Previous Song, Mute, Volume Up, Volume Down, and Stop" voice commands. Use the "Stop Player" command to stop the demo.

The following voice commands are supported:

- Play Music
- Pause
- Previous Song
- Next Song
- Volume Up
- Volume Down
- Mute
- Stop
- Stop Player
3.3 TSN 802.1 Qbv Demo

In this demo, we showcase the TSN 802.1Qbv Time Aware Scheduling (TAS) standard with two traffic classes, namely iperf to emulate best-effort traffic and camera streaming to emulate high priority time-sensitive traffic.
Traffic is streamed between i.MX 8MP and i.MX 8MM boards to demonstrate configurable length and repeating Qbv gate schedules.

**Test Steps:**

- After booting the i.MX 8MM board, you will see the UI, as shown in **Figure 35**.
As shown in Figure 36, click on the three lines in the upper left corner to access the list of TSN supported demos.

As shown in Figure 37, clicking the TSN option followed by the "TSN 802.1Qbv" icon will show the "Launch Demo" button.
• As shown in Figure 38, click on the “Launch Demo” button to start the TSN 802.1 Qbv Demo. After clicking the button, a window with “loading gif” opens, as shown in Figure 39.

• If the connections are not successful, a pop-up window shows up, as shown in Figure 40. After confirming the connections and system setup, click the "OK" button to launch the demo.
• If Ethernet connections are successful and the camera is not connected, the window shown in Figure 41 displays. Click the “OK” button to get back to the “Launch Demo” window.
Figure 41. Click on "Ok" button followed by "Launch Demo"

- If the connections are successful, the TSN Qbv demo login window shows up, as shown in Figure 42.

Figure 42. TSN Qbv demo login window

- The "Video source" and "Start Demo" buttons are in the login window. After selecting the "Video Source" (as shown in Figure 43), the "Start Demo" button is enabled.
If a camera is connected, the available video sources are displayed in a dropdown menu. Select the desired video source to access the camera feed.

After selecting the video source, click the “Start Demo” button to start the demo. After clicking the “Start Demo” button, the “loading gif” window shows up.
• “Start Demo” button will turn into “Stop Demo” and behavior should change accordingly.
• After a successful start of the demo, the default graph window and camera window show up, as shown in Figure 45.

Figure 44. “Start Demo” button and "loading gif" window
• By default, the demo starts with no qbv configuration. The iperf throughput ranges between 900-950 Mbit/s and the camera streams without glitches.

• Select the drop down by clicking the “Configurations drop-down” button to change the TSN Qbv configurations. After clicking the button, you can see the UI (Figure 46).

• Click the “Qbv1 (Video Prioritization)” configuration button.
After clicking the "Qbv1 (Video Prioritization)" button, as shown in Figure 47, the pop-up window shown in Figure 48 appears.

Click the "Ok" button to continue the demo.

Expected result:

- The iperf limits with throughput of 450-500 Mbit/s and camera FPS values should be 25-30 FPS, as shown in Figure 49.
- The iperf traffic is limited to a half of the line rate and the other half of line rate is available for camera traffic. The camera is therefore able to stream at its maximum rate, which, in this example, is much less than 500 Mbit/s.
• Click the “Qbv2 (Iperf Prioritization)” configuration button to select the Qbv2 configuration.

After clicking the “Qbv2 (Iperf Prioritization)” button, a pop-up with a message appears, as shown in Figure 51.

Figure 49. “Qbv1 Graph” window and “Camera” window

Figure 50. “Qbv2 (Iperf Prioritization)” configuration button

Figure 51.
Click the “OK” button to continue the demo.

**Expected results:**

- With the Qbv2 configuration, camera streams with glitches. In graph iperf, the throughput ranges between 900-950 Mbit/s and the camera FPS ranges between 0-10 FPS.
- The camera is limited to a half line rate (0-15 FPS) and iperf with a full line rate (900-950 Mbit/s).
- The FPS values alter based on the lighting conditions. The values are limited from the current FPS when the configurations are applied.

You can select any of the TSN Qbv standards randomly.
- Click the “No qbv (No Prioritization)” configuration button.
After clicking the "No qbv (No Prioritization)" button, as shown in Figure 53, a pop-up with a message appears, as shown in Figure 54.

Click the "OK" button to continue the demo.

Expected results:

- With No qbv (No Prioritization), the iperf throughput should come between 900-950 Mbit/s and the camera should stream fine with camera FPS values of 25-30 FPS.
- The iperf and the camera stream with full line rates.
Select the “Stop demo” button in the TSN Qbv login window to stop the demo.

You can switch across any of these configurations to understand how the demo works.

3.3.1 Application note link

TSN 802.1Qbv Demonstration using i.MX 8M Plus (document AN13995)

3.3.2 Limitations

Keep the following limitations in mind while working on the TSN Qbv demo:
• During verification, the TSN Qbv demo setup should not be disturbed.
• For a smooth demo experience, Ethernet cables and camera should not be disturbed.
• Mostly Logitech and Papalook cameras are used to verify the TSN Qbv demo.
• Camera frame rate (number of frames captured per second, FPS) depends on lighting conditions of the area where the camera is placed. For example, in a dark, low-light, or moderate-light area, the camera frame rate is 10-15 FPS.
• For the TSN Qbv demo, the recommended frame rate is 25-30 FPS. To achieve this frame rate, the camera should be placed in an area with bright light.
• The TSN Qbv demo windows open on the monitor in a random order. The windows must be re-arranged manually.
• During verification, the TSN Qbv demo windows should not be closed.
• Initially, the iPerf drops to 800 Mbit/s when selecting Qbv2 configuration and streams with a full line rate. However, this behavior is achieved in very rare cases.
• The TSN Qbv demo should be stopped before switching to any other demo.

3.4 GPU

Several demos that show off the power of an on-chip GPU are included in this section.

3.4.1 OpenVG 2D

OpenVG 2D is a cross-platform API that provides low-level hardware acceleration for vector graphics. The following demo uses this library to perform their various actions.

3.4.1.1 Tiger G2D

Required materials: Mouse and display output

This demo shows a vector image being rotated and scaled using OpenVG.

3.4.2 GLES2

OpenGL Embedded Systems is a set of APIs for rendering 2D and 3D graphics using an onboard GPU. The following demos use this library to display various graphics.

3.4.2.1 Vivante launcher

Required materials: Mouse and display output

This demo shows an example of a launcher menu that is used in an infotainment system.
Figure 57. Vivante launcher example

3.4.2.2 Cover flow

**Required materials:** Mouse and display output

This demo shows an example of a movie selection screen that would be used in an infotainment system.

3.4.2.3 Vivante tutorial

**Required materials:** Mouse and display output

This demo shows an example of a complex shape generated using the GPU.
3.4.2.4 Bloom

**Required materials:** Mouse and display output

This demo shows an example of how to create a bloom effect. The idea is not to create the most accurate bloom, but something that is fairly fast to render. Instead of increasing the kernel size to get a good blur, we do a fairly fast approximation by downscaling the original image to multiple smaller render targets. We then blur them using a relatively small kernel, and then finally rescale the result to the original size.
3.4.2.5 Blur

**Required materials:** Mouse and display output

This demo uses the two-pass linear technique and further reduces the bandwidth requirement by downscaling the source image to 1/4 its size (1/2w x 1/2h) before applying the blur and then upscaling the blurred image to provide the final image. It works well for large kernel sizes and relatively high sigma's, but the downscaling produces visible artifacts with low sigma's.
3.4.2.6 Eight layer blend

**Required materials:** Mouse and display output

This demo creates a simple parallax scrolling effect by blending eight 32-bit per pixel 1080 p layers on top of each other. It is not the most optimal way to do it as it uses eight passes. But it does provide a good example of the worst-case bandwidth use for the operation. The demo is created to compare GLES to the G2D eight blend functionality.
3.4.2.7 Fractal shader

**Required materials:** Mouse and display output

This demo can render both the Julia and Mandelbrot set using a fragment shader. This demo is used to demonstrate GPU shader performance by using up roughly 515 instructions to render each fragment while generating the Julia set. It uses no textures, has no overdraw, and has a minimal bandwidth requirement.
3.4.2.8  Line Builder 101

**Required materials:** Mouse and display output

This demo shows a simple example of dynamic line rendering using the Line Builder 101 helper class. The Line Builder 101 has "Add" methods for most FslBase Math classes, such as BoundingBox, BoundingSphere, BoundingFrustrum, and Ray.
3.4.2.9 Model loader

**Required materials:** Mouse and display output

This demo demonstrates how to use the FslSceneImporter and Assimp to load a scene and render it using OpenGLES2. The model is rendered using a simple per-pixel directional light shader.
3.4.2.10  S03 transform

**Required materials:** Mouse and display output

This demo renders an animated vertex-colored triangle. It shows how to modify the model matrix to rotate a triangle and how to utilize demoTime and deltaTime to do frame rate independent animation.

3.4.2.11  S04 projection

**Required materials:** Mouse and display output
This demo shows how to build a perspective projection matrix and render two simple 3D models using frame rate independent animation.

![Figure 66. S04 projection example](image)

### 3.4.2.12 S06 texturing

**Required materials:** Mouse and display output

This demo shows how to use the Texture class to use a texture in a cube. This demo also shows how to use the ContentManager service to load a *.png file from the Content directory into a bitmap utility class which is then used to create an OpenGL ES texture.

![Figure 67. S06 texturing example](image)
3.4.2.13 Mapping

**Required materials**: Mouse and display output

This demo shows how to use a cubemap texture to simulate a reflective material. This demo also shows you how to use the ContentManager service to load a *.dds file from the Content directory into a Texture utility class which is then used to create an OpenGL ES cubemap texture.

![Mapping example](image)

3.4.2.14 Mapping refraction

**Required materials**: Mouse and display output

This demo is a variation from "Mapping". As in the previous example, a cubemap texture is used, but this time instead of simulating a reflective material, a refractive material is simulated. This demo also shows how to use the ContentManager service to load a *.dds file from the Content directory into a Texture utility class which is then used to create an OpenGL ES cubemap texture.
Figure 69. Mapping refraction example

4 References

- TSN 802.1Qbv Demonstration using i.MX 8M Plus (document AN13995).

5 Revision history

Table 1 summarizes the changes done to this document since the initial release.

<table>
<thead>
<tr>
<th>Revision number</th>
<th>Release date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>24 June 2022</td>
<td>Initial release</td>
</tr>
<tr>
<td>1</td>
<td>16 September 2022</td>
<td>Updated for 5.15.52 release</td>
</tr>
<tr>
<td>2</td>
<td>07 December 2022</td>
<td>Updated for 5.15.71 release</td>
</tr>
<tr>
<td>3</td>
<td>28 June 2023</td>
<td>Added Section 3.3.</td>
</tr>
<tr>
<td>4</td>
<td>22 August 2023</td>
<td>Added Section 3.2.4.2.</td>
</tr>
</tbody>
</table>
6 Legal information

6.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors’ aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer’s own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer’s sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer’s applications and products planned, as well as for the planned application and use of customer’s third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer’s applications or products, or the application or use by customer’s third party customer(s). Customer is responsible for doing all necessary testing for the customer’s applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer’s third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer’s general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors’ warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors’ specifications such use shall be solely at customer’s own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors’ standard warranty and NXP Semiconductors’ product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

6.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

i.MX — is a trademark of NXP B.V.

TensorFlow, the TensorFlow logo and any related marks — are trademarks of Google Inc.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction ......................................................... 2</td>
</tr>
<tr>
<td>1.1</td>
<td>Installing the NXP Demo Experience ............................... 2</td>
</tr>
<tr>
<td>2</td>
<td>Demo launcher .................................................... 2</td>
</tr>
<tr>
<td>2.1</td>
<td>Graphical User Interface (GUI) ................................. 2</td>
</tr>
<tr>
<td>2.2</td>
<td>Text User Interface (TUI) ..................................... 4</td>
</tr>
<tr>
<td>3</td>
<td>Included demos ................................................... 5</td>
</tr>
<tr>
<td>3.1</td>
<td>Machine learning demos ........................................ 5</td>
</tr>
<tr>
<td>3.1.1</td>
<td>NNStreamer demos ................................................... 5</td>
</tr>
<tr>
<td>3.1.1.1</td>
<td>Object classification ........................................... 6</td>
</tr>
<tr>
<td>3.1.1.2</td>
<td>Pose detection .................................................... 8</td>
</tr>
<tr>
<td>3.1.1.4</td>
<td>Brand detection ................................................... 9</td>
</tr>
<tr>
<td>3.1.1.5</td>
<td>ML gateway .......................................................... 10</td>
</tr>
<tr>
<td>3.1.2</td>
<td>PyeiQ demos ........................................................... 14</td>
</tr>
<tr>
<td>3.1.2.1</td>
<td>Object classification ............................................... 15</td>
</tr>
<tr>
<td>3.1.2.2</td>
<td>Object detection .................................................... 16</td>
</tr>
<tr>
<td>3.1.2.3</td>
<td>Mask detection ..................................................... 17</td>
</tr>
<tr>
<td>3.1.3</td>
<td>OpenCV demos ..................................................... 18</td>
</tr>
<tr>
<td>3.1.3.1</td>
<td>Face recognition ................................................... 18</td>
</tr>
<tr>
<td>3.1.4</td>
<td>TF Lite demos ....................................................... 22</td>
</tr>
<tr>
<td>3.1.4.1</td>
<td>ML benchmark ........................................................ 22</td>
</tr>
<tr>
<td>3.1.4.2</td>
<td>Driver Monitoring System (DMS) demo .......................... 22</td>
</tr>
<tr>
<td>3.1.4.3</td>
<td>Selfie segmentation ................................................ 23</td>
</tr>
<tr>
<td>3.2</td>
<td>Multimedia demos .................................................. 24</td>
</tr>
<tr>
<td>3.2.1</td>
<td>GStreamer demos ..................................................... 24</td>
</tr>
<tr>
<td>3.2.1.1</td>
<td>Video test demo ..................................................... 24</td>
</tr>
<tr>
<td>3.2.1.2</td>
<td>Camera using the VPU .......................................... 25</td>
</tr>
<tr>
<td>3.2.1.3</td>
<td>Multi-camera preview ............................................. 26</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Image Signal Processor (ISP) demos ............................ 26</td>
</tr>
<tr>
<td>3.2.2.1</td>
<td>ISP control demo ................................................... 26</td>
</tr>
<tr>
<td>3.2.2.2</td>
<td>Video dump demo ................................................... 28</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Audio ................................................................. 29</td>
</tr>
<tr>
<td>3.2.3.1</td>
<td>Audio record ........................................................ 29</td>
</tr>
<tr>
<td>3.2.3.2</td>
<td>Audio play .......................................................... 29</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Voice ................................................................. 29</td>
</tr>
<tr>
<td>3.2.4.1</td>
<td>i.MX voice control ................................................... 30</td>
</tr>
<tr>
<td>3.2.4.2</td>
<td>i.MX Multimedia Player .......................................... 31</td>
</tr>
<tr>
<td>3.3</td>
<td>TSN 802.1 Qbv Demo .............................................. 34</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Application note link ............................................. 47</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Limitations .......................................................... 47</td>
</tr>
<tr>
<td>3.4</td>
<td>GPU ................................................................. 48</td>
</tr>
<tr>
<td>3.4.1</td>
<td>OpenVG 2D ............................................................ 48</td>
</tr>
<tr>
<td>3.4.1.1</td>
<td>Tiger G2D ........................................................... 48</td>
</tr>
<tr>
<td>3.4.2</td>
<td>GLES2 ................................................................. 48</td>
</tr>
<tr>
<td>3.4.2.1</td>
<td>Vivante launcher ................................................... 48</td>
</tr>
<tr>
<td>3.4.2.2</td>
<td>Cover flow ............................................................ 49</td>
</tr>
<tr>
<td>3.4.2.3</td>
<td>Vivante tutorial .................................................... 49</td>
</tr>
<tr>
<td>3.4.2.4</td>
<td>Bloom ................................................................. 50</td>
</tr>
<tr>
<td>3.4.2.5</td>
<td>Blur ................................................................. 51</td>
</tr>
<tr>
<td>3.4.2.6</td>
<td>Eight layer blend ................................................... 52</td>
</tr>
<tr>
<td>3.4.2.7</td>
<td>Fractal shader ....................................................... 53</td>
</tr>
<tr>
<td>3.4.2.8</td>
<td>Line Builder 101 ................................................... 54</td>
</tr>
<tr>
<td>3.4.2.9</td>
<td>Model loader ........................................................ 55</td>
</tr>
<tr>
<td>3.4.2.10</td>
<td>S03 transform ....................................................... 56</td>
</tr>
<tr>
<td>3.4.2.11</td>
<td>S04 projection ........................................................ 56</td>
</tr>
<tr>
<td>3.4.2.12</td>
<td>S06 texturing ........................................................ 57</td>
</tr>
<tr>
<td>3.4.2.13</td>
<td>Mapping ............................................................... 58</td>
</tr>
<tr>
<td>3.4.2.14</td>
<td>Mapping refraction .................................................. 58</td>
</tr>
<tr>
<td>4</td>
<td>References ......................................................... 59</td>
</tr>
<tr>
<td>5</td>
<td>Revision history .................................................... 59</td>
</tr>
<tr>
<td>6</td>
<td>Legal information ................................................... 60</td>
</tr>
</tbody>
</table>

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2023 NXP B.V. All rights reserved.

For more information, please visit: http://www.nxp.com

Date of release: 22 August 2023
Document identifier: DEXPUG