

Vision Toolbox for MATLAB

Quick-Start

**Embedded Target for the S32V234 Family of
Automotive Vision Processors**

Version 2018.R1.EAR1

Target Based Automatic Code Generation Tools
For MATLAB™ working with Mathworks Image Processing and Computer Vision Toolboxes



Summary

1	Introduction	1-3
1.1	Purpose	1-3
1.2	Audience.....	1-3
1.3	Definitions, Acronyms and Abbreviations	1-3
2	Installation	2-4
2.1	System Requirements	2-4
2.2	MATLAB Required and Recommended Products.....	2-4
2.3	Mandatory Software	2-5
2.3.1	NXP Support Package for S32V234	2-5
2.3.2	NXP Vision Toolbox for S32V234.....	2-10
2.3.3	License Generation and Activation	2-15
2.3.4	Vision SDK and Build Tools.....	2-20
2.3.5	Setting up the Environment	2-22
2.4	Optional Software.....	2-23
2.4.1	SD Card Bootable Linux Image	2-23
3	Vision Application	3-24
3.1	Examples Library & Help	3-24
3.2	Face Detection in Simulation Mode	3-25
3.2.1	Running the Algorithm for Pictures	3-25
3.2.2	Running the Algorithm using Video Frames from Webcam	3-27
3.2.3	Running the Algorithm for Videos.....	3-29
3.3	Face Detection on S32V234 Vision Processor.....	3-31
3.3.1	Configure the microSD Card.....	3-31
3.3.2	S32V234 Evaluation Board Configuration	3-34
3.3.3	USB to UART connection.....	3-36
3.3.4	Compile and Run on S32V234.....	3-39

1 Introduction

This Quick Start Guide is designed to get you up and running in a matter of minutes with the concepts used by the NXP Vision Toolbox for S32V234 automotive vision processors. This toolbox is designed to be used from MATLAB in conjunction with the NXP S32V234 Vision SDK that support the Linux OS runtime environment.

The first part of this document covers the toolbox installation and setup of required prerequisites.

The second part then shows how to run a simple vision application in simulation and then on the real hardware evaluation board.

1.1 Purpose

The purpose of this document is to demonstrate how to install all the required software and run a vision application on NXP S32V234 automotive vision processors.

1.2 Audience

This document is intended to:

- MATLAB Computer Vision System users that wish to evaluate the NXP HW&SW solutions;
- NXP S32V234 users that need to have a quick start-up into vision applications and ready to run examples;

1.3 Definitions, Acronyms and Abbreviations

Acronym	Description
ACF	APEX Core Framework
APEX	A parallel image processing accelerator HW block part of NXP S32V234 SoC.
APEX COMPILER	Set of tools (NXP APU compiler) that allow compilation of code for APEX subsystem
ARM	Family of RISC architectures
SDK	Software Development Kit
ISP	Image Signal Processor

2 Installation

Installing the NXP Vision Toolbox for S32V234 is the first step in setting up and running automatic code generation from MATLAB for NXP S32V234 automotive vision processors and development boards.

The next sections present all the steps required to have the toolbox installed successful and ready for running the first application.

2.1 System Requirements

For a flowless development experience the minimum recommended PC platform is:

- Windows® 7/10 64bit Operating System
- At least 2 GHz CPU Speed
- At least 4 GB of RAM
- At least 20 GB of free disk space.
- Internet connectivity for web downloads.

2.2 MATLAB Required and Recommended Products

The NXP Vision Toolbox for S32V234 requires the following Mathwork's products to be installed. Make sure you have a valid license for the products marked as "Required"

Product	Version Compatibility	Required or Recommended
MATLAB	R2018a	Required
MATLAB Coder	R2018a	Required
Embedded Coder	R2018a	Required
Image Processing Toolbox	R2018a	Required
Computer Vision System Toolbox	R2018a	Required
Embedded Coder Support Package for ARM Cortex-A Processors	R2018a	Required
MATLAB Support Package for USB Webcams	R2018a	Recommended
Image Acquisition Toolbox Support Package for OS Generic Video Interface	R2018a	Recommended

Due to code generation performance issues the NXP Vision Toolbox uses a special feature `row-major` that has been introduced in [MATLAB Coder 2018a](#).

2.3 Mandatory Software

NXP Vision Toolbox is delivered as MATLAB Toolbox Package (MLTBX) that can be installed:

- Online from Mathwork's File Exchange [website](#). For convenience, a NXP Support Package for S32V234 is available to assist throughout the installation process of the NXP Vision Toolbox and supplementary software;
- Offline from NXP [website](#) as a MATLAB Add-on;

This section shows how to install the NXP Vision Toolbox using online approach directly from Mathwork's Add-ons File Exchange website. In case you have already downloaded the NXP Vision Toolbox for S32V234 MLTB file from NXP's official web page then jump directly to section 2.3.2 NXP Vision Toolbox for S32V234

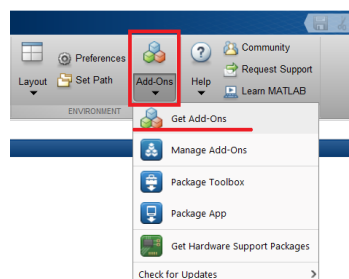
To have the NXP Vision Toolbox installed and configured properly the following actions should be executed:

- Use Get Add-ons menu from MATLAB to search for "S32V Support Package" online and install the toolbox;
- Start the NXP Support Package for S32V234 and follow the steps indicated in the user interface;
- Download and install the NXP Vision Toolbox for S32V234 from NXP [website](#)
- Register and activate the NXP Vision Toolbox license
- Download and install the NXP Vision SDK package, including the cross-compilation tools for ARM and APEX cores
- Set the APU Compiler and Vision SDK Environment Variables

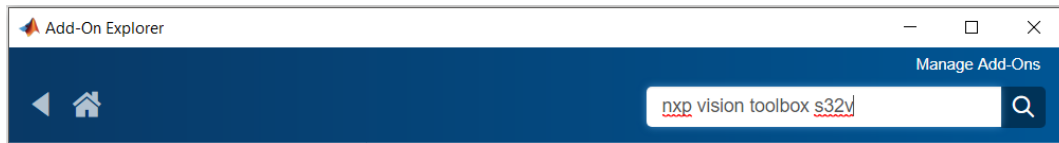
Each of these actions are explained in the following sub-chapters.

2.3.1 NXP Support Package for S32V234

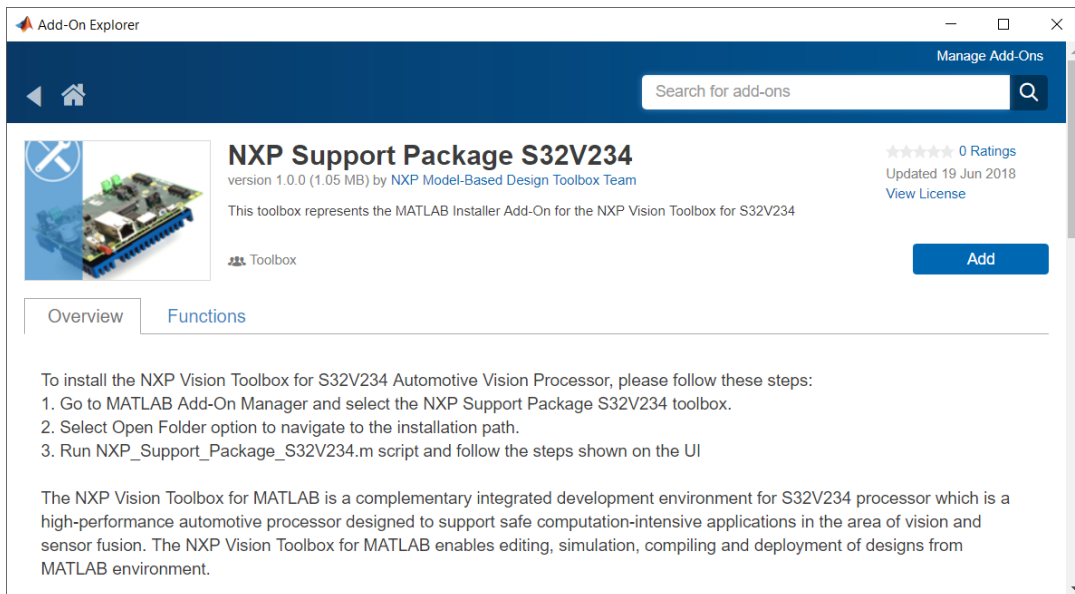
For convenience a step-by-step installer guide is available on Mathwork's File Exchange [website](#). Open MATLAB 2018a and select Get Add-Ons:



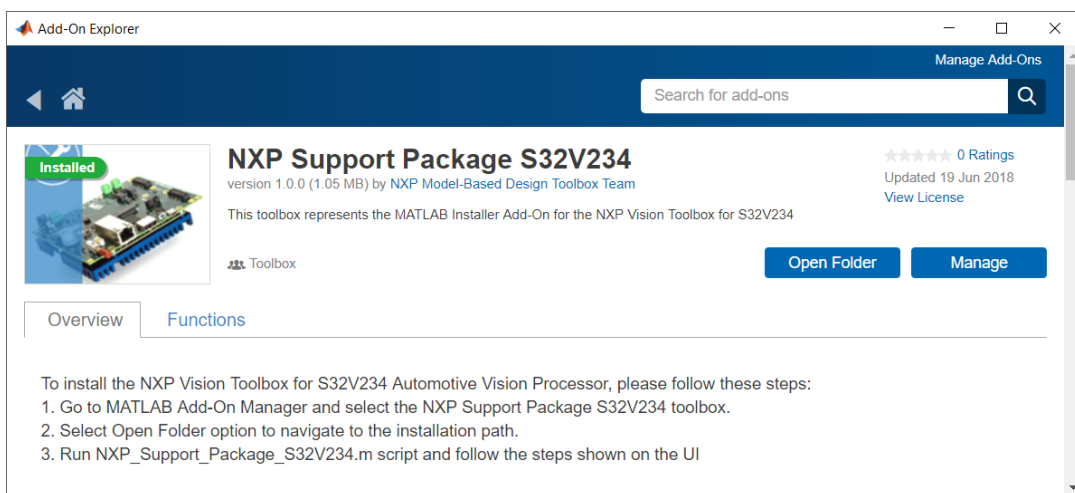
1. Once the Add-On Explorer window opens, search for “nxp vision toolbox s32v”



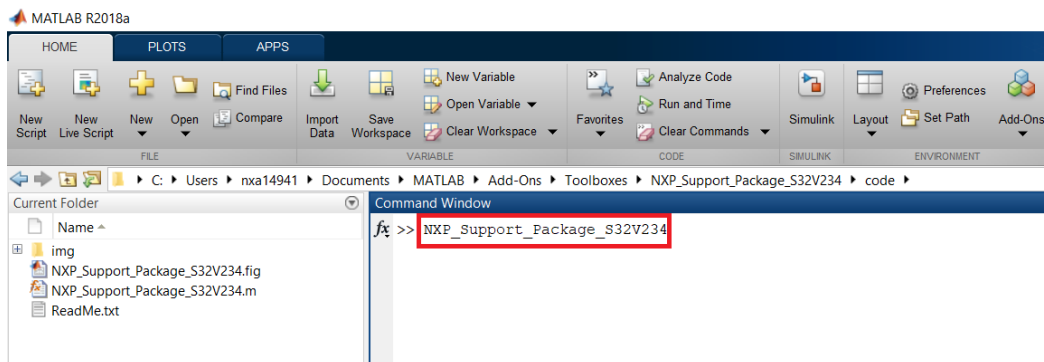
2. Select the NXP Support Package for S32V234 and click on Add button to start the installation of the installer guide into your MATLAB instance.



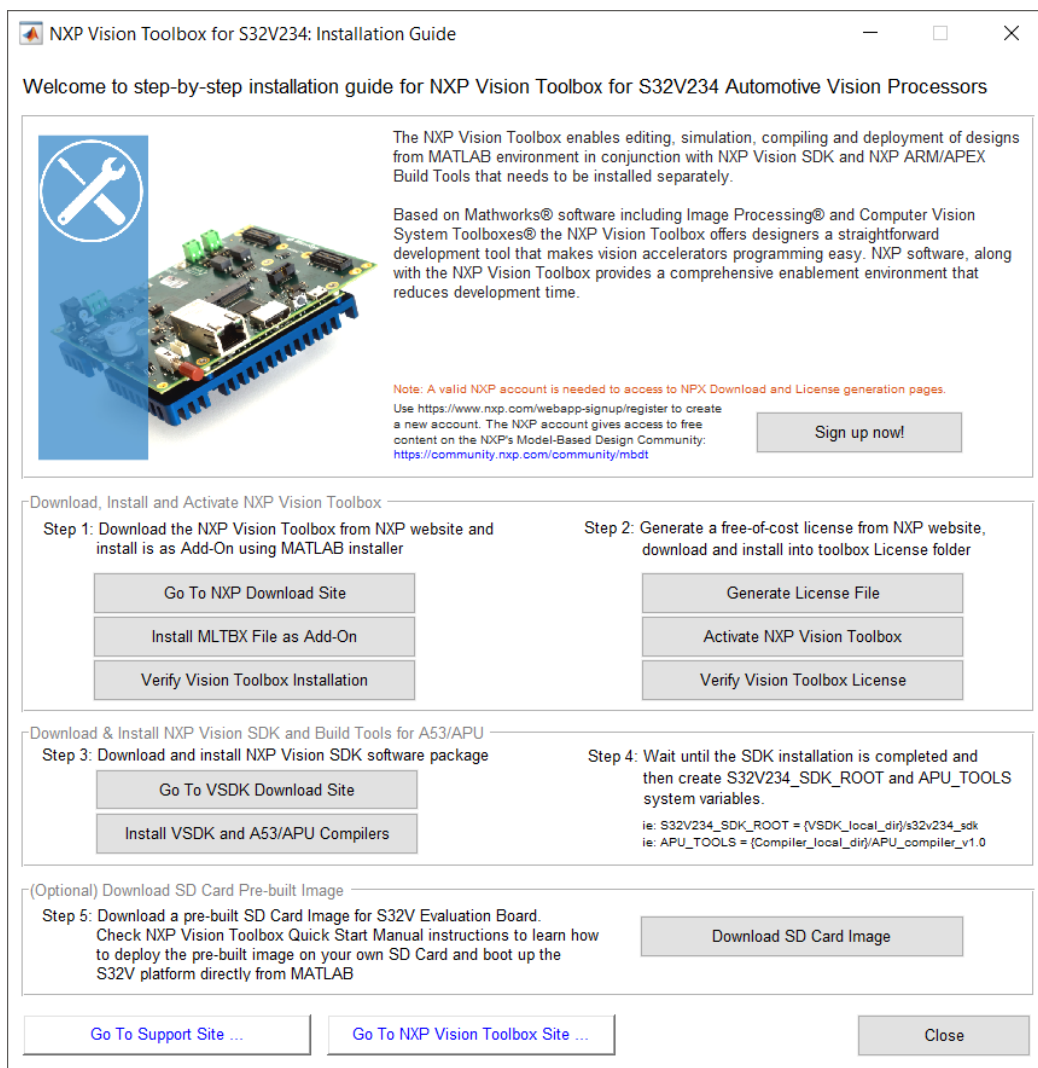
3. Wait until the toolbox is installed and then click on Open Folder button.



4. Run the `NXP_Support_Package_S32V234` command in your MATLAB console to start the Installer Guide.



5. The NXP Support Package for S32V234 - Installer Guide User Interface is started



The Installer Guide contains instructions for downloading, installing and verification of all software components required for being able to develop vision application with MATLAB for NXP S32V234 automotive vision processors:

- Steps to download, install and verification of the NXP Vision Toolbox for S32V234
- Steps to generate, activate and verification of the license for NXP Vision Toolbox for S32V234
- Steps to download and install NXP Vision SDK package
- Steps to configure the software environment for code generation
- Steps to download additional software

There are 2 main advantages of using this Installer Guide:

- Each step completion is automatically checked by the tool. If the action is completed successfully, then the tool is going to mark it as green. If a particular step cannot be verified then the tool will issue an warning or error and is going to highlight in red that particular step that needs more attention for user side.
- Future updates will be made available via this online toolbox. In case you wish to keep your software up to date, then please install this into your MATLAB Add-ons and once a new update will be available your MATLAB instance will notify you.

The next screen capture shows how the Installer Guide notify user of successful or failed actions. At the end of installation all push buttons should be green.


NXP Vision Toolbox for S32V234: Installation Guide

Welcome to step-by-step installation guide for NXP Vision Toolbox for S32V234 Automotive Vision Processors



The NXP Vision Toolbox enables editing, simulation, compiling and deployment of designs from MATLAB environment in conjunction with NXP Vision SDK and NXP ARM/APEX Build Tools that needs to be installed separately.

Based on Mathworks® software including Image Processing® and Computer Vision System Toolboxes® the NXP Vision Toolbox offers designers a straightforward development tool that makes vision accelerators programming easy. NXP software, along with the NXP Vision Toolbox provides a comprehensive enablement environment that reduces development time.




Warning

NXP Vision Toolbox for S32V234 was not found!
Please check Add-Ons/Manage Add-Ons menu to verify if this toolbox exists

OK

License generation pages.

Sign up now!

Download, Install and Activate NXP Vision Toolbox

Step 1: Download the NXP Vision Toolbox from NXP website and install is as Add-On using MATLAB installer

Go To NXP Download Site

Install MLTBX File as Add-On

Verify Vision Toolbox Installation

Step 2: Generate a free-of-cost license from NXP website, download and install into toolbox License folder

Generate License File

Activate NXP Vision Toolbox

Verify Vision Toolbox License

Download & Install NXP Vision SDK and Build Tools for A53/APU

Step 3: Download and install NXP Vision SDK software package

Go To VSDK Download Site

Install VSDK and A53/APU Compilers

Step 4: Wait until the SDK installation is completed and then create S32V234_SDK_ROOT and APU_TOOLS system variables.

ie: S32V234_SDK_ROOT = {VSDK_local_dir}/s32v234_sdk
ie: APU_TOOLS = {Compiler_local_dir}/APU_compiler_v1.0

(Optional) Download SD Card Pre-built Image

Step 5: Download a pre-built SD Card Image for S32V Evaluation Board. Check NXP Vision Toolbox Quick Start Manual instructions to learn how to deploy the pre-built image on your own SD Card and boot up the S32V platform directly from MATLAB

Download SD Card Image

Go To Support Site ...

Go To NXP Vision Toolbox Site ...

Close

2.3.2 NXP Vision Toolbox for S32V234

You can obtain the NXP Vision Toolbox for S32V234 by:

- Using the Installer guide “Go To NXP Download Site” button

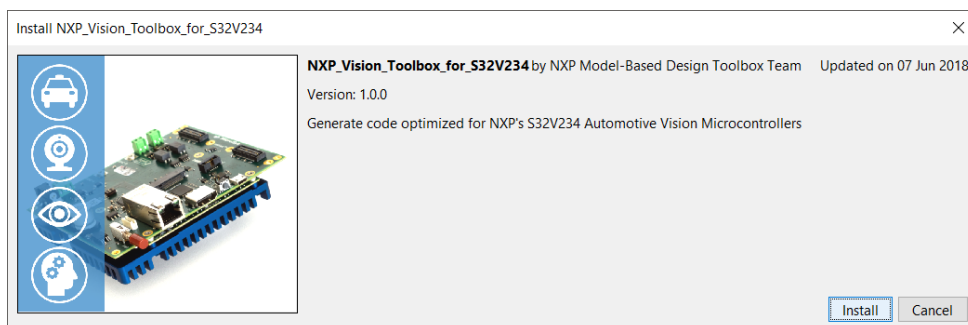


- Go directly into your NXP Software Account and download the toolbox using this [link](#)

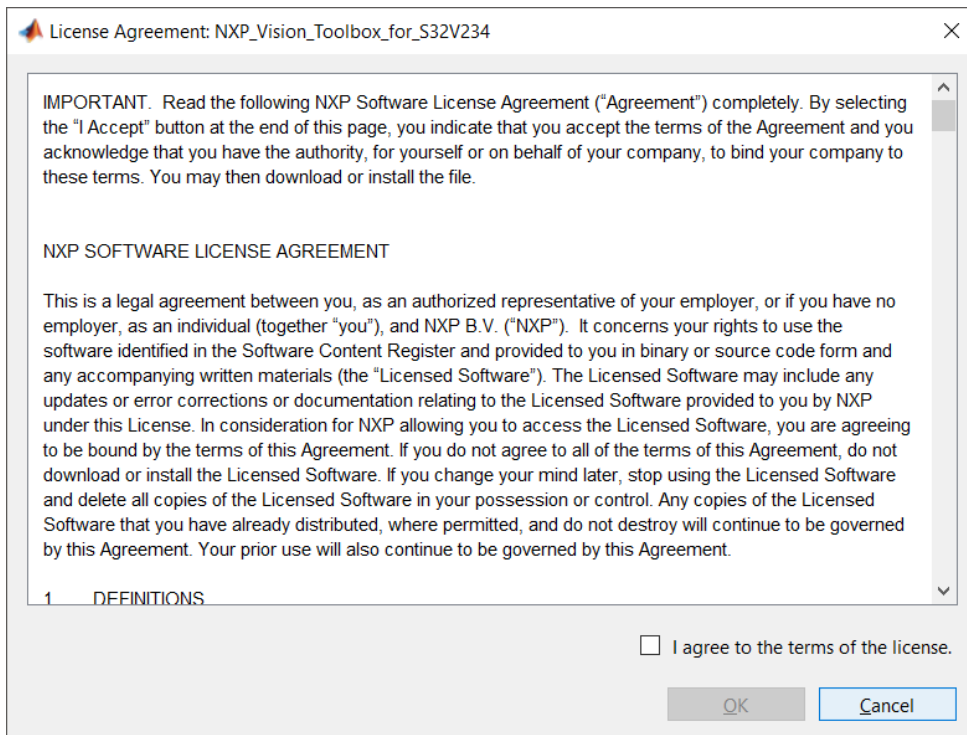
No matter which option is used, the NXP Vision Toolbox for S32V234 installation steps are similar: once you have the toolbox on your PC, double click on the *.mltbx file to start the MATLAB Add-ons installer that will automatically start the installation process.

You will be prompted with the following options:

1. The NXP’s Vision Toolbox Installation Wizard dialog will appear. Click “Install” to proceed.

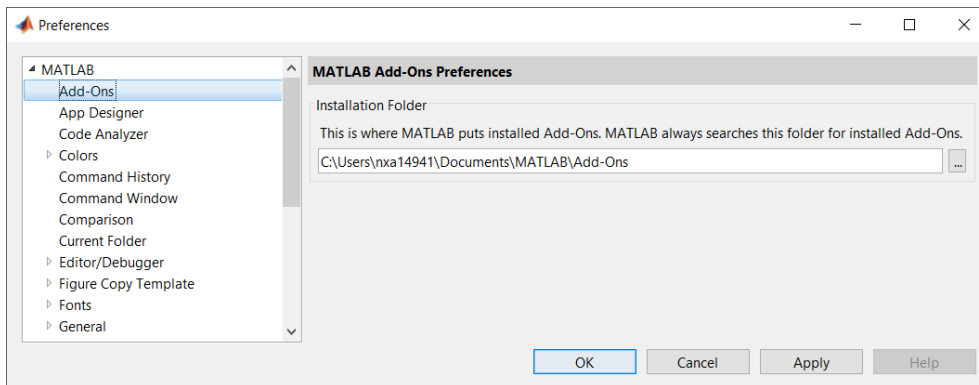


2. Indicate acceptance of the NXP Software License Agreement by selecting “I agree to the terms of the license” to proceed.

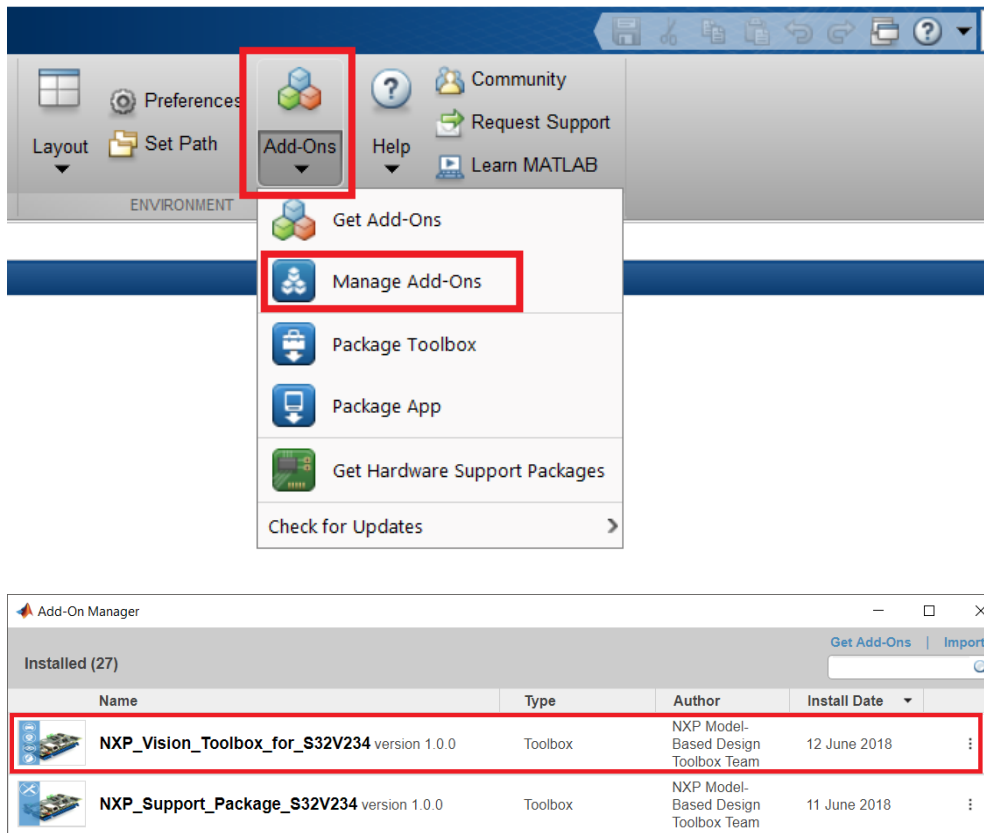


3. Click "OK" to start the MATLAB installation process. The rest of the process is silent and under MATLAB control. All the files will be automatically copied into default Add-Ons folder within the MATLAB

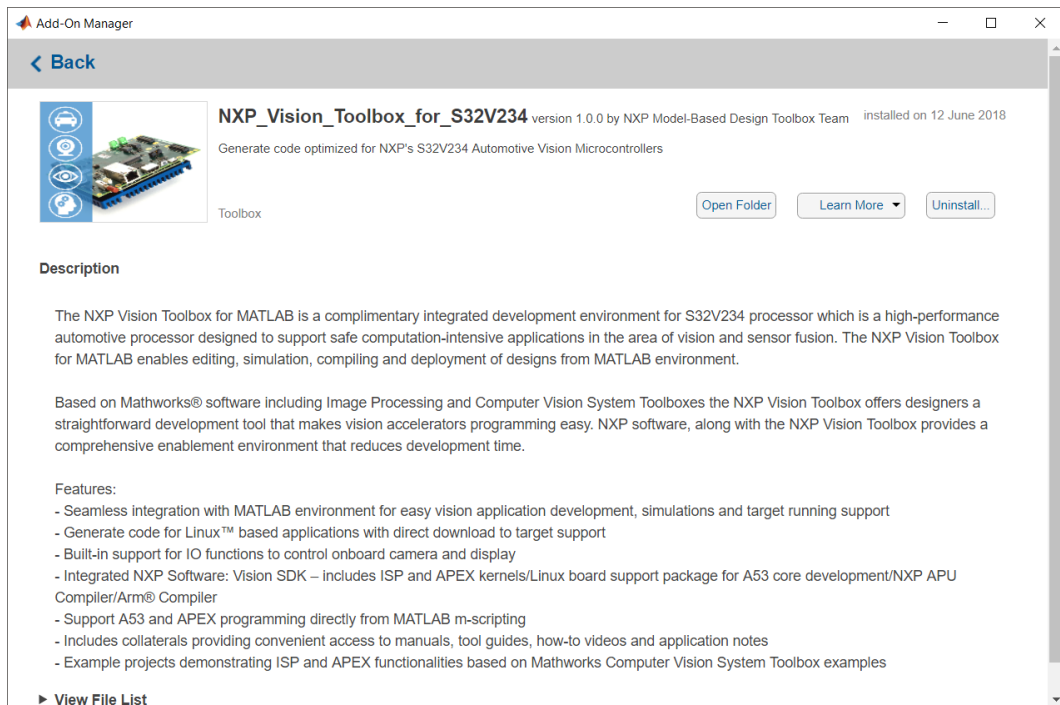
The default location can be changed prior to installation by changing the Add-Ons path from MATLAB Preferences



4. After a couple of seconds, the NXP's Vision Toolbox should be visible as a new Add-ons.

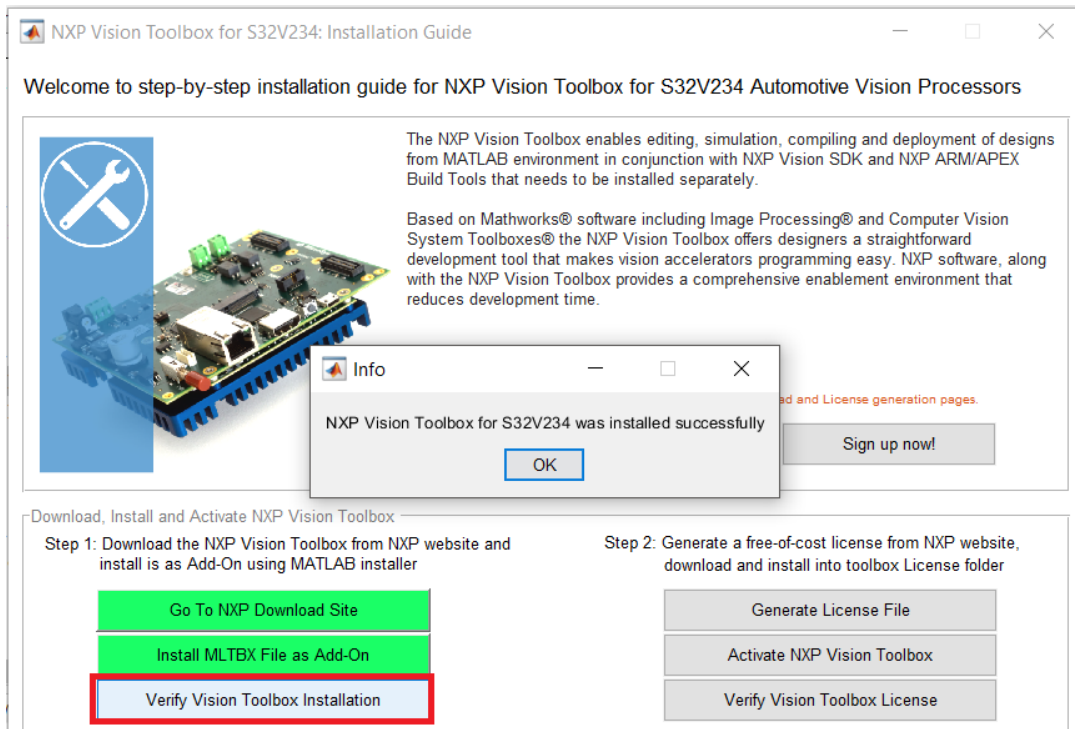


5. More details about the NXP's Vision Toolbox can be found by clicking on View Details

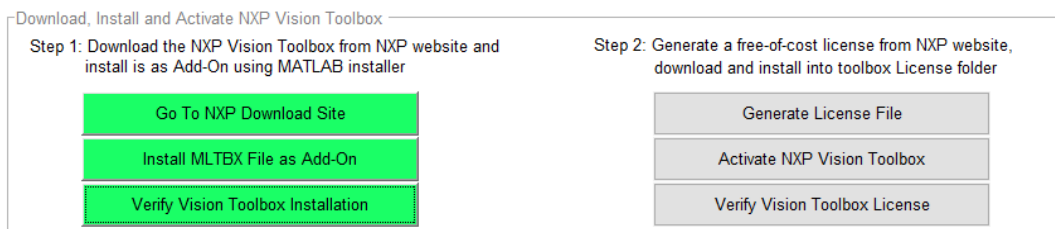


6. NXP Vision Toolbox documentation, help and examples are fully integrated with MATLAB development environment. Get more details by accessing the standard Help and Supplemental Software section

7. In case you are using the Installer Guide, then you have the option to check if the NXP Vision Toolbox is installed correctly on your MATLAB environment by simply clicking on “Verify Vision Toolbox Installation” button



After this step you should see all button related with Vision Toolbox Step 1, green

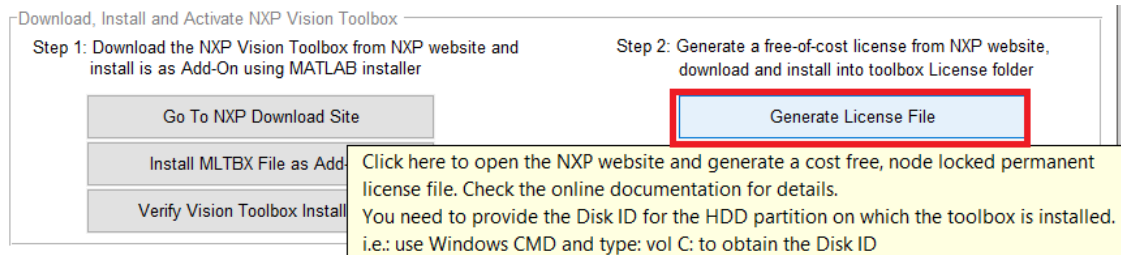


2.3.3 License Generation and Activation

The NXP Vision Toolbox for S32V234 is available free of charge, however, a valid license is required.

You can obtain the NXP Vision Toolbox for S32V234 license free of charge by:

- Using the Installer guide “Generate License File” button



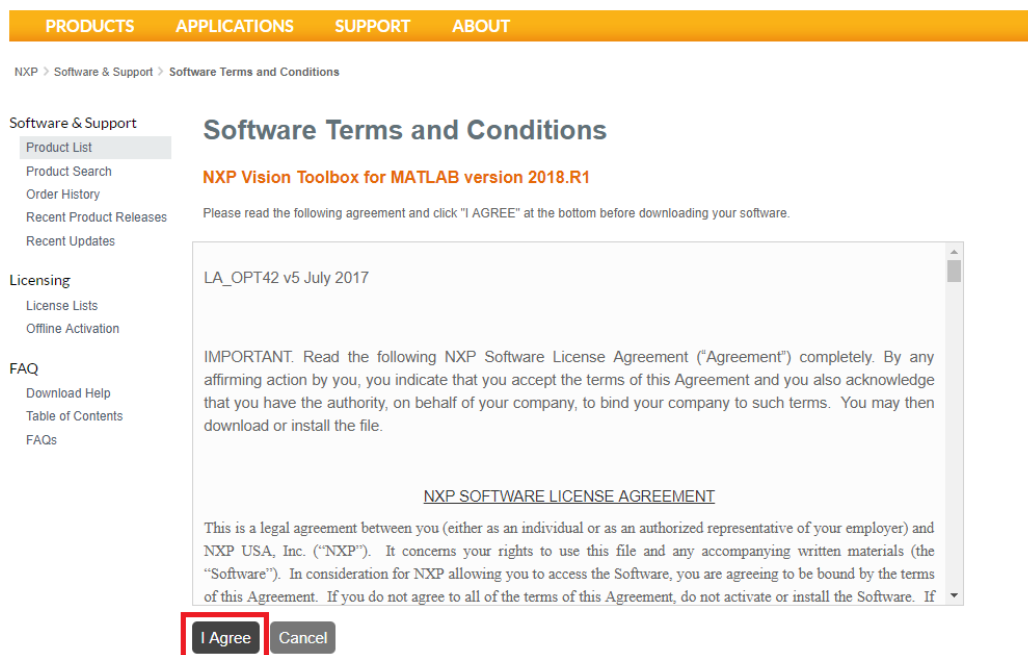
- Go directly into your NXP Software Account and Generate the license using this [link](#)

For more details about license generation please refer to online or offline manual:
[Vision_Toolbox_License_Activation.pdf](#)

Perform the following steps to obtain the NXP Vision Toolbox for S32V234 license:

1. For the first-time log-in, the “Software Terms and Conditions” page will be displayed. Click on “I agree” button to consent to the software license agreement.

NOTE In this section we presume, you already logged into your NXP account to download the toolbox prior to license generation step.



2. Click on “License Keys” tab

PRODUCTS APPLICATIONS SUPPORT ABOUT

NXP > Software & Support > Vision Toolbox > NXP Vision Toolbox for MATLAB version 2018.R1 : Files

Software & Support

- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing

- License Lists
- Offline Activation

FAQ

- Download Help
- Table of Contents
- FAQs

Product Download

NXP Vision Toolbox for MATLAB version 2018.R1

Files License Keys Notes [Download Help](#)

The Vision Toolbox is delivered as a MATLAB MLTBX file. To avoid any kind of file corruption during download process, make sure you select the file you wish to download using the checkbox and then click on "Download Selected Files" button

Show All Files 5 Files

☐	File Description	File Size	File Name
☐	+ NXP Support Package for S32V234 version 2018.R1.EAR1	1.1 MB	NXP_Support_Package_S32V234_20180620.mltbx
☐	+ NXP Vision Toolbox for S32V234 Release Notes	943.9 KB	NXP_Vision_Toolbox_Release_Notes.pdf
☐	+ NXP Vision Toolbox for S32V234 version 2018.R1.EAR1	81.2 MB	NXP_Vision_Toolbox_S32V234_20180620.mltbx
☐	+ S32V234-EVB SD-CARD Image	138.1 MB	S32V234-EVB_29288_image.gz
☐	+ Software Content Register for NXP Vision Toolbox	1.2 KB	Software_Content_Register_NXP_Vision_Toolbox.txt

Download Selected Files

3. Verify if the correct tool and version are identified and then check the box and click on “Generate” button.

PRODUCTS APPLICATIONS SUPPORT ABOUT

NXP > Software & Support > License Information

Software & Support

- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing

- License Lists
- Offline Activation

FAQ

- Download Help
- Table of Contents
- FAQs

License Information

NXP Vision Toolbox for MATLAB version 2018.R1

Generate

Item Description	NXP Vision Toolbox for MATLAB version 2018.R1
Order Number	VISION-MATLAB_v2018.R1_64522011
Purchase Order Number	
Total Number of Licenses:	101

☐ License Applicable to Product(s):

Version	Description
2018	NXP Vision Toolbox for MATLAB version 2018.R1 (View EULA)
R1	

101 Available

Generate

4. Select Disk Serial Number or Ethernet address as the “Node Host ID”. If you do not know your Disk Serial Number nor the Ethernet address then check the link available on this page with details about License Generation. Enter a name for license to help managing them in case you need to use the Vision Toolbox on multiple computers. (Optional)

PRODUCTS APPLICATIONS SUPPORT ABOUT

NXP > Software & Support > Generate Licenses

Software & Support

- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing

- License Lists
- Offline Activation

FAQ

- Download Help
- Table of Contents
- FAQs

Generate Licenses

Instructions for finding your host ID details are available [here](#).

Please do not use spaces in the **Name** field (for node-locked licenses) or **Host Description** field (for floating licenses). These fields are available to add brief text notes to your license.

License Applicable to Product(s):		Number of Licenses Available
Version	Description	
2018 R1	NXP Vision Toolbox for MATLAB version 2018.R1	101

Node Host ID

Disk Serial Number ▼ 66B72EBD

Name

dp-laptop

Node Host ID

▼

Name

Node Host ID

▼

Name

Node Host ID

▼

Name

Node Host ID

▼

Name

Node Host ID

▼

Name

Generate

5. Click on “Generate” button to get the license. Verify if the information is correct: Toolbox version, expiration date, Node Host ID

PRODUCTS APPLICATIONS SUPPORT ABOUT

NXP > Software & Support > View Licenses

Software & Support

- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing

- License Lists
- Offline Activation

FAQ

- Download Help
- Table of Contents
- FAQs

View Licenses

Below are the licenses you just generated.

[License Overview](#) [Print Friendly](#) [Save All](#)

License Applicable to Product(s):

Version	Description
2018 R1	NXP Vision Toolbox for MATLAB version 2018.R1

License Quantity: 1

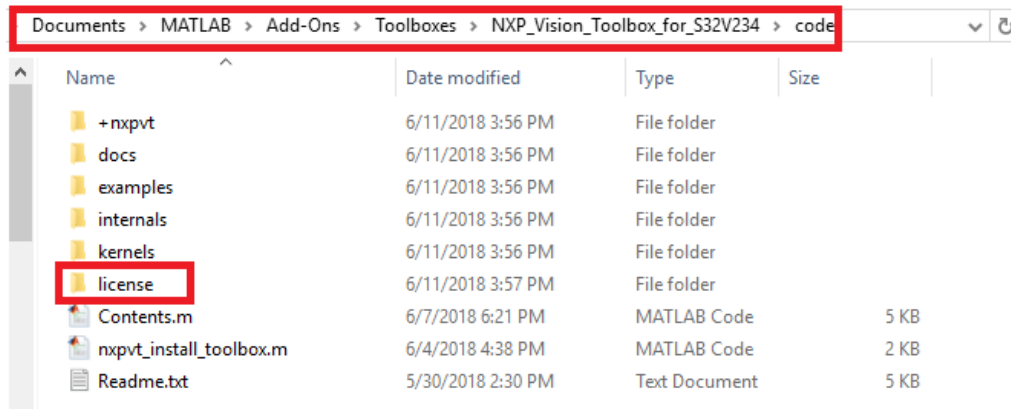
Expiration Date: May 28, 2023

Disk Serial Number: 66B72EBD (dp-laptop)
Generated By: Dumitru-Daniel Popa on Jun 11, 2018

```
#NXP Vision Toolbox for MATLAB version 2018.R1 - NXP Vision Toolbox for MATLAB
#version 2018.R1 for Dumitru-Daniel Popa Software Account
# License for DISK_SERIAL_NUM=66B72EBD dp-laptop
INCREMENT Vision_Toolbox_freescale 1.0 28-may-2023 uncounted \
  VENDOR_STRING="NXP Vision Toolbox for MATLAB version 2018.R1" \
  HOSTID=DISK_SERIAL_NUM=66b72ebd ISSUER="Freescale \
  Semiconductor" ISSUED=11-jun-2018 ckt=209 SN="FSL - 23623107" \
  TS_OK_SIGN="19DD 45FC 3172 59EE FE0C BF47 E125 F403 2F74 5A63 \
  65D6 AC28 EFCD 98C5 067F 1369 5E18 A4F3 995E 0B60 E9B8 8766 \
  10ED F53C CAC2 2DFE 9222 E4E3 CC9A B960"
```

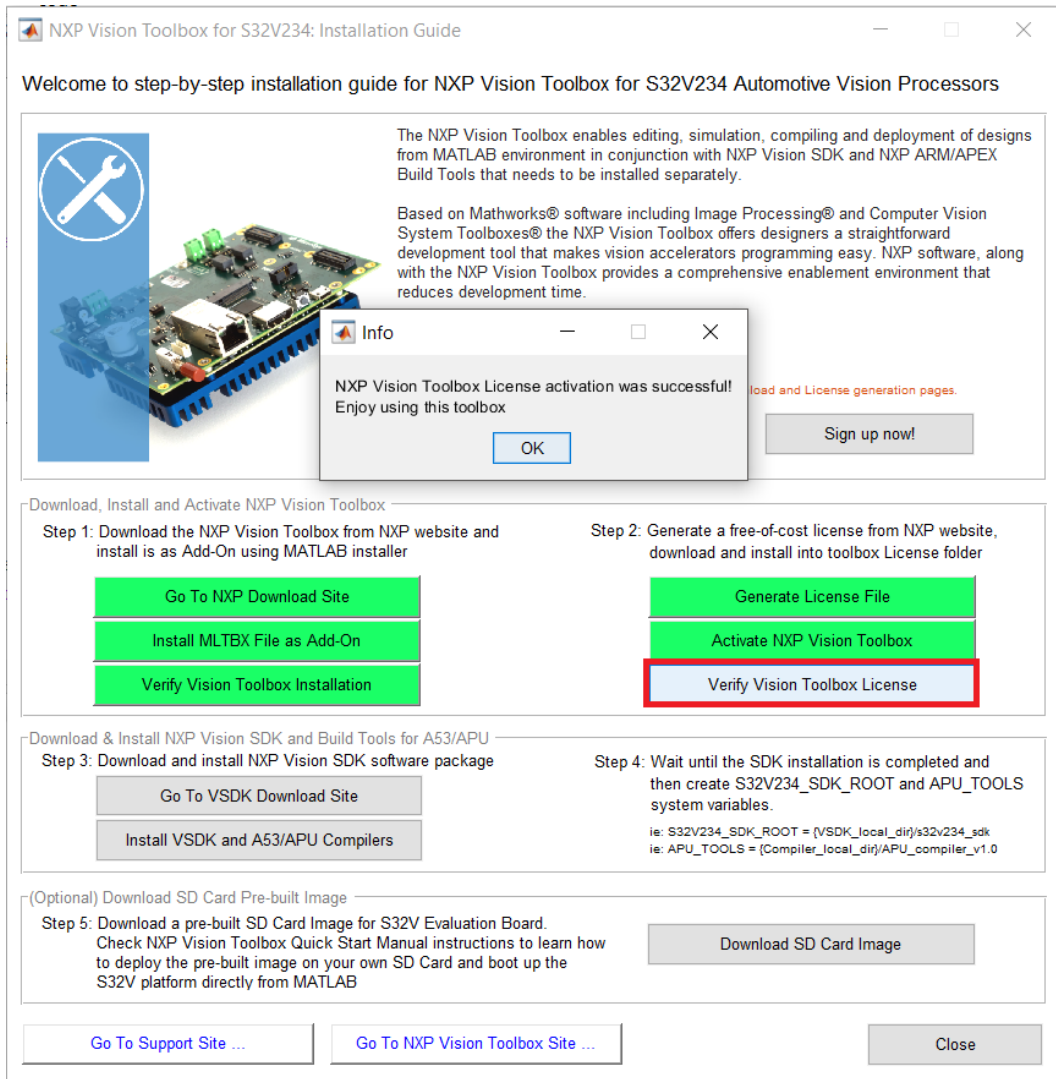
[License Overview](#) [Print Friendly](#) [Save All](#)

6. Either click on “Save All” or copy and paste the file into a text editor, and save the file as “license.dat” into the “Vision Toolbox installed directory\license” folder.



In case you are using the Installer Guide, then you can save the license file anywhere and use the “Activate NXP Vision Toolbox” option to make sure the license is copied correctly in the appropriate toolbox location

7. Check if the license file is installed correctly by using the “Verify Vision Toolbox License” button. If everything is ok, then the Installer Guide will confirm the action



Alternatively, you can check from command line is the license for NXP Vision Toolbox is activated. Run the command `nxpvt_license_check`. If there are issues with the license, this command will return the root-cause.

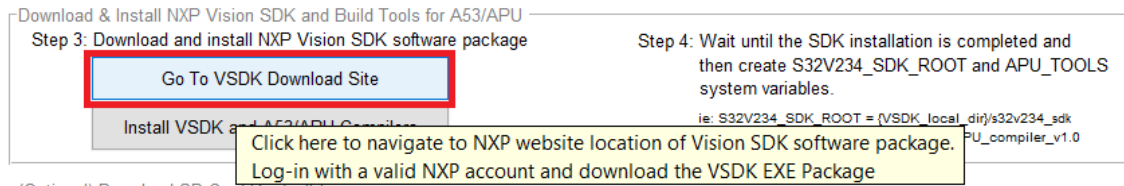
```
Command Window
>> nxpvt_license_check
Error using nxpvt_license_check
License Error: -9, Invalid host. The hostid of this system does not match
the hostid specified in the license file.
In case you do not have a license, please go to The NXP Vision Toolbox Web
Site to get a free license or request a demo. Provide the following HostID:
66B7-2EBD
fx >> |
```

2.3.4 Vision SDK and Build Tools

All the code generated by NXP Vision Toolbox is based on S32V234 Vision SDK package. This software package is also free of charge and apart of optimized kernels and libraries for the S32V automotive vision processors, it also contains the build tools to cross-compile the MATLAB generated code to ARM A53 and APEX cores.

You can obtain the S32V234 Vision SDK free of charge by:

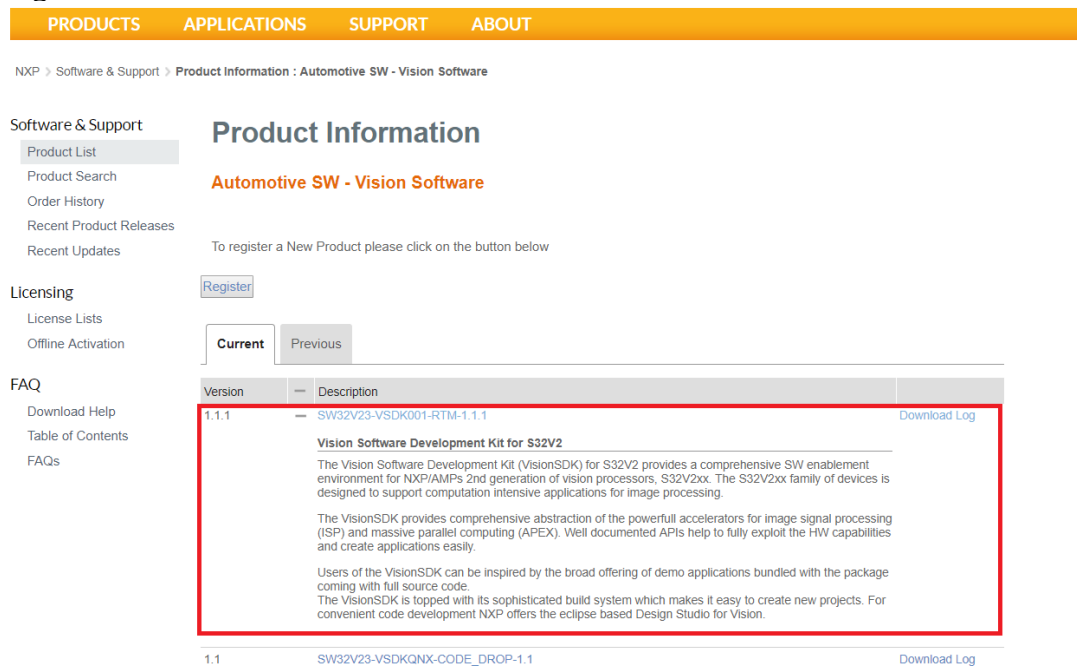
- Using the Installer guide “Go To VSDK Download Site” button



- Go directly to NXP [website](#)

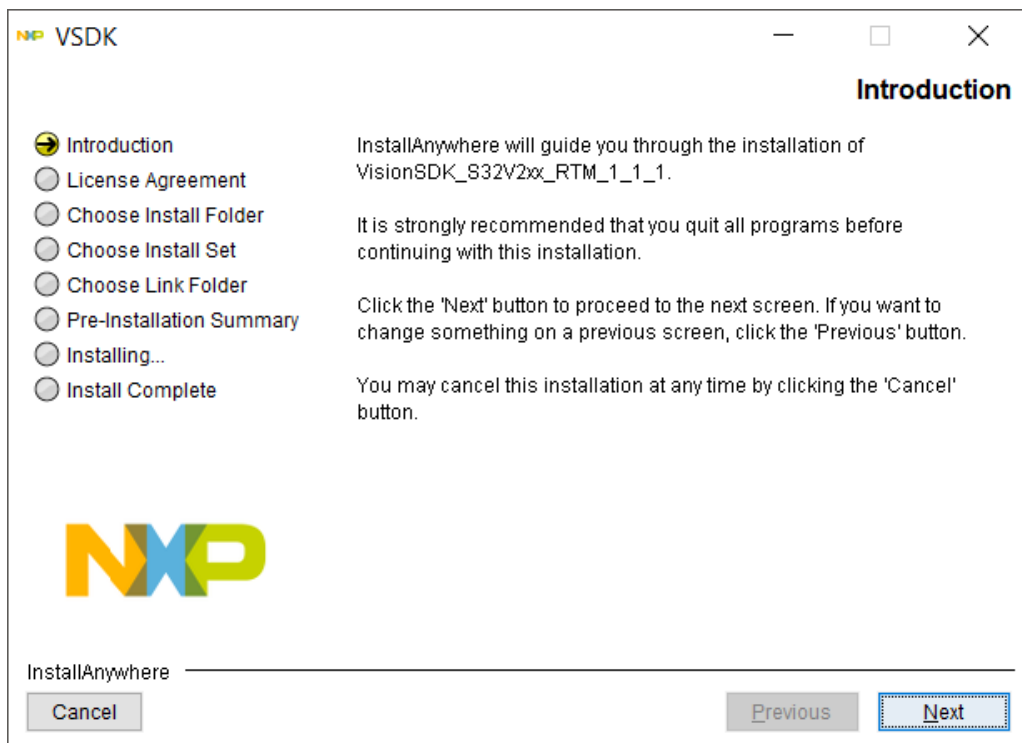
Perform the following steps to obtain and install the S32V234 Vision SDK and NXP Build Tools:

1. Download the Vision SDK RTM v1.1.1 on your PC. Due the size of the package this might take a while.



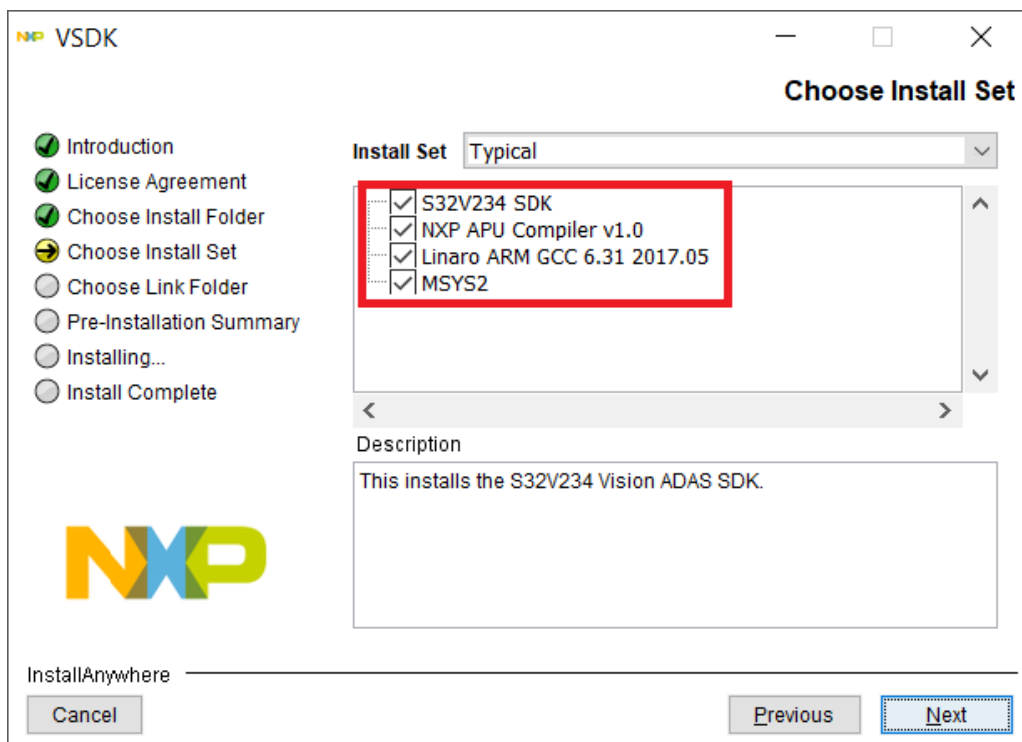
2. Once the VisionSDK_S32V2_RTM_1_1_1.exe download is finished, select “Install VSDK and A53/APU Compilers” option in the Installer Guide UI.

3. Select the exe file and wait for the Vision SDK Install Anywhere to start.



4. Make sure you follow all the steps and install the:

- NXP APU Compiler v1.0 – used to compile the generated code for APEX Vision Accelerator
- Linaro ARM GCC 6.31 – used to compile the generated code for ARM A53
- MSYS2 – used to configure the bootable Linux image and to download the actual vision application to the S32V234 Evaluation Board

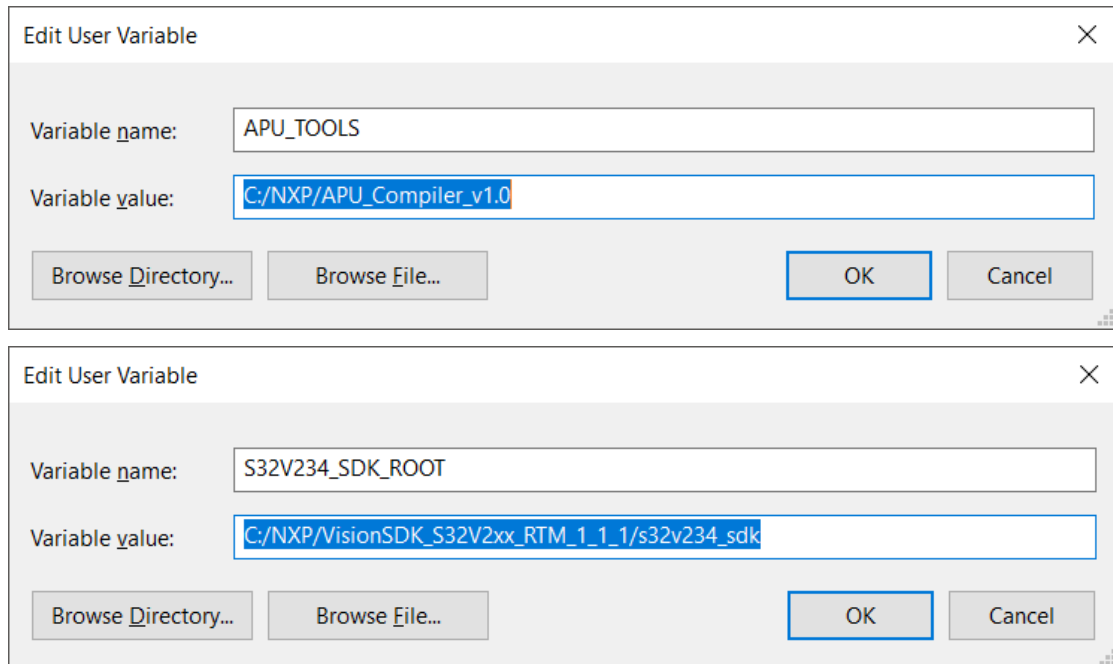


2.3.5 Setting up the Environment

The last step required for software configuration is to set two system or user environmental variables `APU_TOOLS` and `S32V234_SDK_ROOT` that points to:

```
APU_TOOLS= C:/NXP/APU_Compiler_v1.0  
S32V234_SDK_ROOT = C:/NXP/VisionSDK_S32V2xx_RTM_1_1_1/s32v234_sdk
```

Ensure system or user environment variables, corresponding to the compiler(s) you have installed, are defined to compiler path value as shown below:



Note: Paths shown are for illustration, your installation path may be different. Once environmental variables are setup you will need to restart MATLAB to use these variables.

2.4 Optional Software

This section describes the additional software that may be needed to have the full setup working and to be able to download and run vision application directly from MATLAB on S32V234 Evaluation Boards.

2.4.1 SD Card Bootable Linux Image

The S32V234 Vision SDK is delivered with pre-built images that can be used to configure the S32V234 evaluation board to have is up and running for vision application. If case you are familiar with Linux OS, then please follow the procedures shown in the Vision SDK Manuals for building and configuration of the bootable SD Card.

For users that are not familiar with Linux OS or simply do not have a Linux Machine available to configure the SD Card for the S32V234 EVB there is a simple alternative.

Using the Installer Guide you can download a pre-built Linux bootable image for S32V234 EVB that can be configured by MATLAB directly from Windows OS.

The screenshot shows the NXP Vision Toolbox for MATLAB website. At the top, a yellow callout box contains the following text: "Click here to navigate to NXP website location of Pre-built SD-Card Image. The SD-Card Image can be copied directly from Windows OS to an empty SD-CARD to create a bootable Linux card needed for running vision applications". Below this, a red box highlights the "Download SD Card Image" button. The main content area is titled "Product Download" and "NXP Vision Toolbox for MATLAB version 2018.R1". It features a "Files" tab and a table of available files. The table has columns for "File Description", "File Size", and "File Name". The file "S32V234-EVB SD-CARD Image" is highlighted with a red box, and its download link "S32V234-EVB_29288_image.gz" is also highlighted. A red box at the bottom highlights the "Download Selected Files" button.

(Optional) Download SD Card Pre-built Image

Step 5: Download a pre-built SD Card Image for S32V Evaluation Board. Check NXP Vision Toolbox Quick Start Manual instructions to learn how to deploy the pre-built image on your own SD Card and boot up the S32V platform directly from

Download SD Card Image

Click here to navigate to NXP website location of Pre-built SD-Card Image. The SD-Card Image can be copied directly from Windows OS to an empty SD-CARD to create a bootable Linux card needed for running vision applications

Go To Support Site ...

PRODUCTS APPLICATIONS SUPPORT ABOUT

NXP > Software & Support > Vision Toolbox > NXP Vision Toolbox for MATLAB version 2018.R1 : Files

Software & Support

- Product List
- Product Search
- Order History
- Recent Product Releases
- Recent Updates

Licensing

- License Lists
- Offline Activation

FAQ

- Download Help
- Table of Contents
- FAQs

Product Download

NXP Vision Toolbox for MATLAB version 2018.R1

Files License Keys Notes [Download Help](#)

The Vision Toolbox is delivered as a MATLAB MLTBX file. To avoid any kind of file corruption during download process, make sure you select the file you wish to download using the checkbox and then click on "Download Selected Files" button

Show All Files 5 Files

☐	File Description	File Size	File Name
☐	+ NXP Support Package for S32V234 version 2018.R1.EAR1	1.1 MB	NXP_Support_Package_S32V234_20180620.mltbx
☐	+ NXP Vision Toolbox for S32V234 Release Notes	943.9 KB	NXP_Vision_Toolbox_Release_Notes.pdf
☐	+ NXP Vision Toolbox for S32V234 version 2018.R1.EAR1	81.2 MB	NXP_Vision_Toolbox_S32V234_20180620.mltbx
☒	+ S32V234-EVB SD-CARD Image	138.1 MB	S32V234-EVB_29288_image.gz
☐	+ Software Content Register for NXP Vision Toolbox	1.2 KB	Software_Content_Register_NXP_Vision_Toolbox.txt

Download Selected Files

In this case you need to provide a 16GB microSD card, preferably SDHC or SDXC class 10.

3 Vision Application

This section shows how to simulate, generate the code, configure the S32V234 evaluation and run a face detection application in real time on the NXP Hardware.

3.1 Examples Library & Help

NXP's Vision Toolbox comes with an Examples Library that let you test and run multiple applications. To open the library, go to `MATLAB Help` (or simply press F1) and select the `NXP Vision Toolbox for S32V234 Supplemental Software`

There are four groups of examples that highlights four different types of functionalities supported by NXP Vision Toolbox for S32V234:

- Vision Applications – contains complex application like face, pedestrian and lane detection demos that can be run in both simulation and hardware;
- APEX Kernels – contains examples like Sobel and Gauss filters;
- APEX Computer Vision – show how to use the APEXCV classes to build optimized examples on ARM and APEX cores ;
- S32V234 EVB IO Examples;

For the Quick Start we are going to choose the Face Detection application demo and go thru all steps to simulate and run on the target.

3.2 Face Detection in Simulation Mode

3.2.1 Running the Algorithm for Pictures

Go to ...examples/apps/face detection folder and open the m-script file face_detection_image_main.m

Alternatively, you can open the example from the MATLAB Help. The script should look like the one shown below. This script is using as input an image and will run the face detection algorithm that is implemented using Local Binary Patterns and Cascading Classifiers. At the end, if any faces are found, the script will display a red rectangle on top of the original picture.

```
function face_detection_image_main() %#codegen
    inImgPath = 'data/face_detection.png';
    width = uint32(640);
    height = uint32(480);

    inImgUMat = nxpvt.imread(inImgPath);
    if isempty(inImgUMat)
        fprintf('Failed to open input image: %s.', inImgPath);
        return;
    end

    fdetector = nxpvt.FaceDetector(width, height);

    fNum = int32(0);
    fps = int32(0);

    nxpvt.tic;

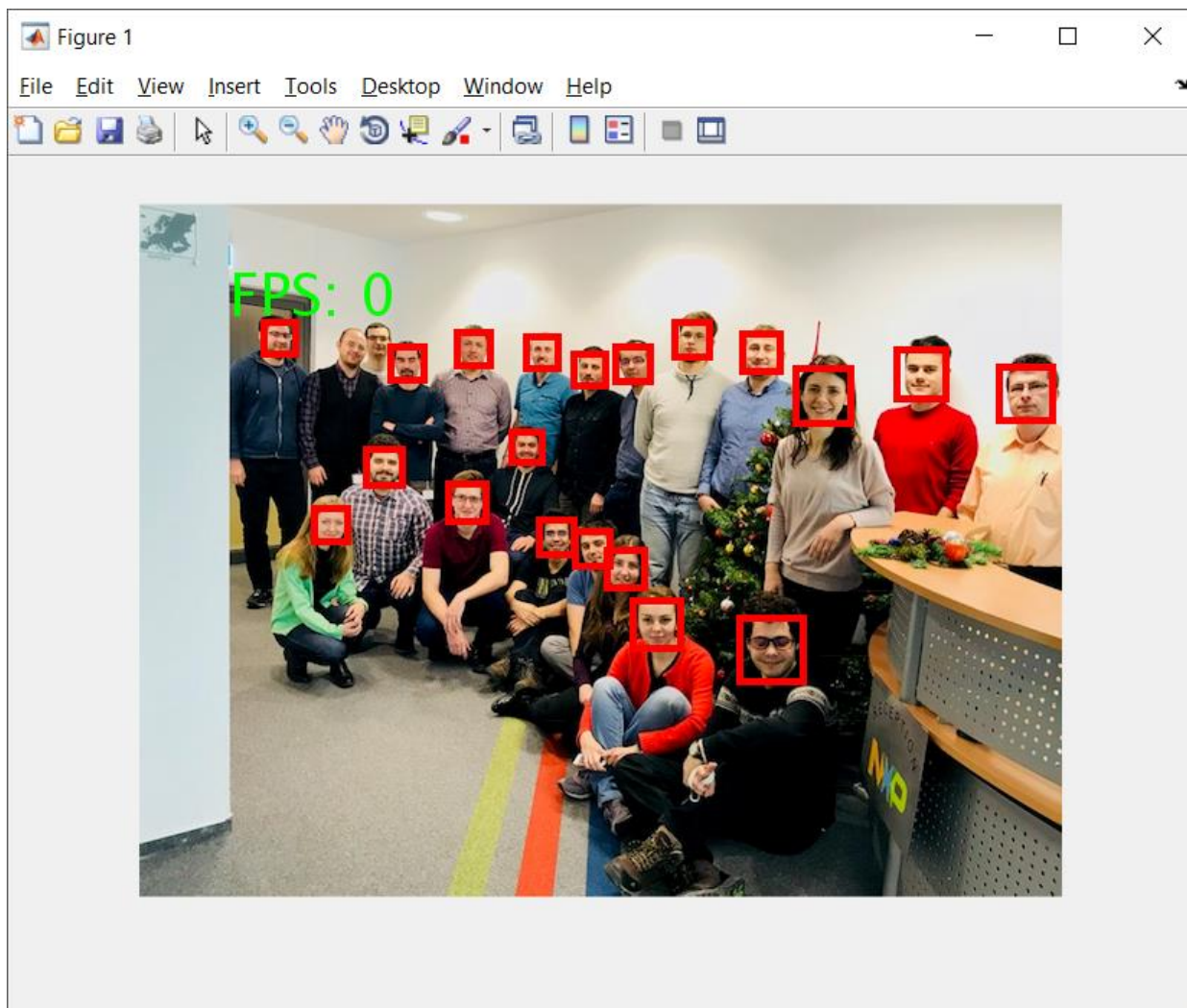
    % Get faces.
    [bbox, l] = fdetector.GetFaces(inImgUMat);

    outImgUMat = nxpvt.cv.rectangle(inImgUMat, bbox, [255, 0, 0], 5);
    outImgUMat = nxpvt.cv.putText(outImgUMat, sprintf('FPS: %d', fps), [50, 100],
        nxpvt.cv.HersheyFonts.FONT_HERSHEY_SIMPLEX, 2, [0, 255, 0], 2);

    nxpvt.imshow(outImgUMat);

    fps = int32(fix(1/nxpvt.toc));
    fprintf('%d] FPS: %d, Faces detected: %d, \n', fNum, fps, int32(l));
end
```

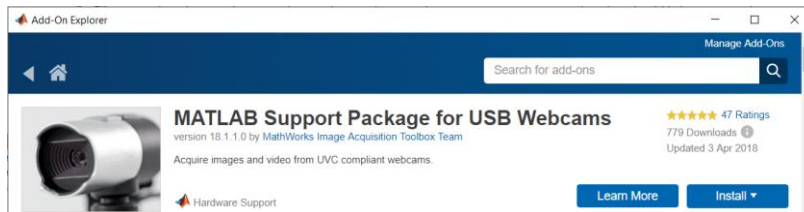
All you should do is to run this script from MATLAB to produce the results. Press F5 to start. The result should be identical with the one shown in the next figure.



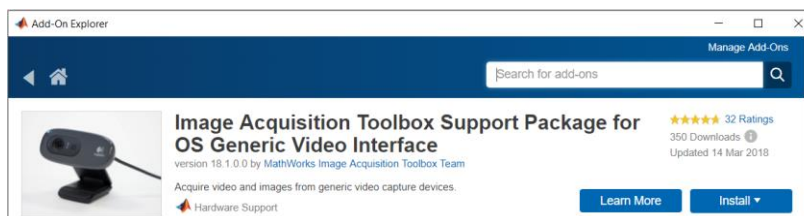
3.2.2 Running the Algorithm using Video Frames from Webcam

The next step is to try the algorithm on some real footage capture from a webcam. Before running this test, you need to install two additional toolboxes that allows you to capture frames from the webcam. Use the MATLAB Get Add-On menu to find and install the following toolboxes:

- [MATLAB Support Package for USB Webcams](#)



- [Image Acquisition Toolbox Support Package for OS Generic Video Interface](#)



Once both these toolboxes are installed, go to ...examples/apps/face detection folder and open the m-script file `face_detection_camera_main.m`

```
function face_detection_camera_main() %#codegen
    width = uint32(1280);
    height = uint32(720);

    if coder.target('MATLAB')
        input = nxpvt.videoinput('winvideo', 1, width, height);
    else
        input = nxpvt.videoinput('sony', 1, width, height, true, false);
    end

    fdetector = nxpvt.FaceDetector(width, height);

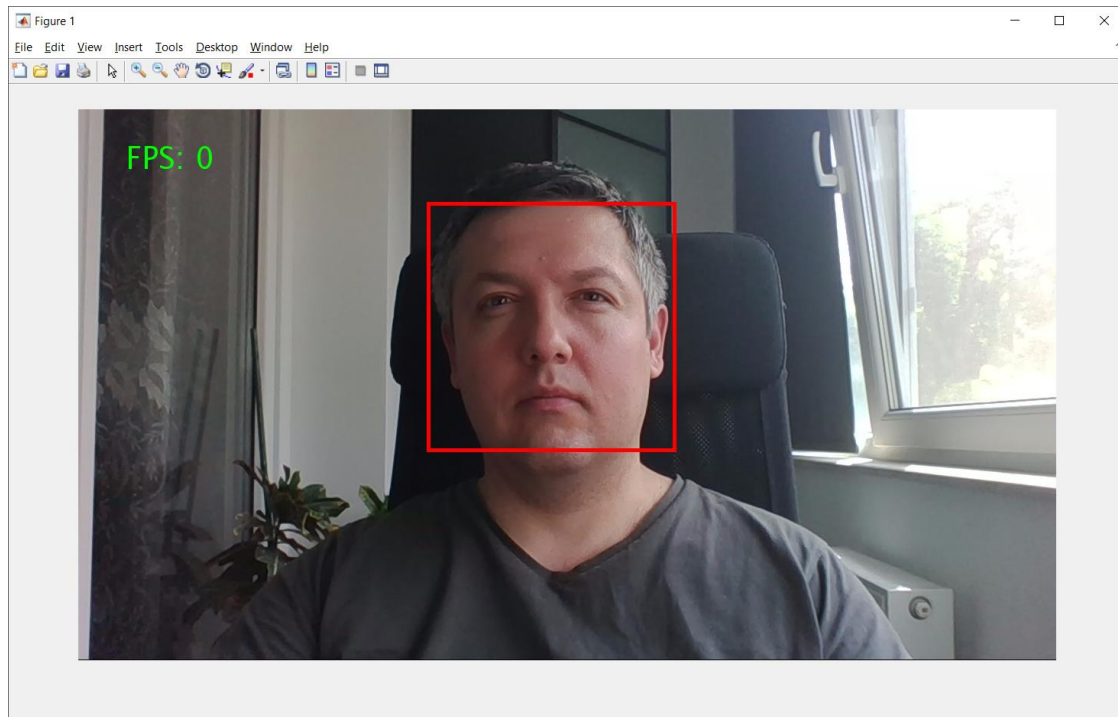
    fNum = int32(0);
    fps = int32(0);
    while true
        nxpvt.tic;
        fNum = fNum + 1;
        frame = input.getsnapshot();

        % Get faces.
        [bbox, l] = fdetector.GetFaces(frame);
        frame = nxpvt.cv.rectangle(frame, bbox, [255, 0, 0], 5);
        frame = nxpvt.cv.putText(frame, sprintf('FPS: %d', fps), [50, 100],
            nxpvt.cv.HersheyFonts.FONT_HERSHEY_SIMPLEX, 2, [0, 255, 0], 2);
        nxpvt.imshow(frame);

        fps = int32(fix(1/nxpvt.toc));
        fprintf(['%d] FPS: %d, Faces detected: %d, \n', fNum, fps, int32(1));
    end
end
```

In this example the source of the video frame is different between MATLAB Simulation and S32V234 Hardware Test. Notice, how is the application is running on the target the source for input is the Sony camera. Using `coder.target()` we can choose during code generation the source the inputs.

If the setup was performed correctly, the algorithm should detect your face.



3.2.3 Running the Algorithm for Videos

The vision algorithm can be then tested on pre-recorded video to emulate and test the real use-case on the hardware. Go to ...examples/apps/face_detection folder and open the m-script file face_detection_video_main.m

```
function face_detection_video_main() %#codegen
    if coder.target('MATLAB')
        videoFile = fullfile(matlabroot, 'toolbox', 'vision', 'visiondata',
                               'visionface.avi');
    else
        videoFile = 'data/visionface.avi';
    end
    width = uint32(640);
    height = uint32(480);

    obj.videoReader = nxpvt.VideoReader(videoFile);

    if coder.target('MATLAB')
        % Generate a video player, only for simulation.
        obj.videoPlayer = vision.VideoPlayer();
    end

    fdetector = nxpvt.FaceDetector(width, height);

    fNum = int32(0);
    fps = int32(0);

    while hasFrame(obj.videoReader)
        nxpvt.tic;
        fNum = fNum + 1;

        frame = readFrame(obj.videoReader);

        % Get faces.
        [bbox, 1] = fdetector.GetFaces(frame);

        frame = nxpvt.cv.rectangle(frame, bbox, [255, 0, 0], 5);
        frame = nxpvt.cv.putText(frame, sprintf('FPS: %d', fps), [50, 100],
                                   nxpvt.cv.HersheyFonts.FONT_HERSHEY_SIMPLEX, 2, [0, 255, 0], 2);

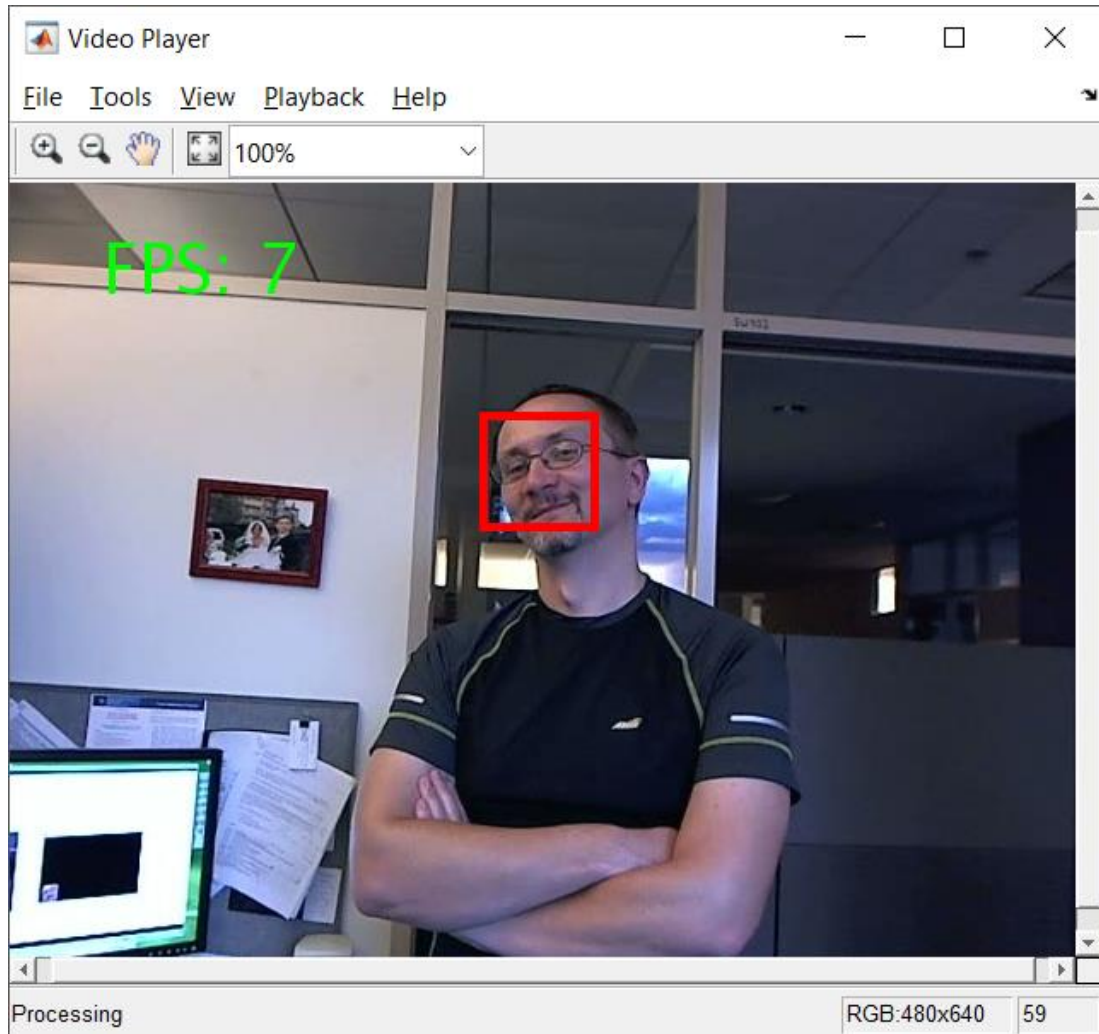
        if coder.target('MATLAB')
            step(obj.videoPlayer, frame.data);

            %Exit the loop if the video player figure is closed.
            if ~isOpen(obj.videoPlayer)
                break;
            end
        else
            nxpvt.imshow(frame);
        end

        fps = int32(fix(1/nxpvt.toc));
        fprintf('%d] FPS: %d, Faces detected: %d, \n', fNum, fps, int32(1));
    end
end
```

The same m-script can be executed in both Simulation or on the S32V234 EVB. Depending on the type of target used to run the application (MATLAB vs. S32V234) there are slightly different actions executed. Still the Face Detection algorithm is the same for both cases.

In this case we are using a video source from one of the standard MATLAB [examples](#). If the test is successful you should see this:



3.3 Face Detection on S32V234 Vision Processor

In this section the focus is on the steps required to generate the code and running the application on the S32V234.

3.3.1 Configure the microSD Card

As it was mentioned in the paragraph 2.4.1, you need to have a Linux OS configured to boot up the S32V234 platform to enable the Eth, ISP, ACF and other various drivers and modules needed for running embedded vision algorithms.

The entire procedure for configuration and booting up the platform is described in the Vision SDK manuals. Unfortunately, not everyone has access to a Host PC with Linux OS to configure a SD card (*formatting, uboot, filesystem, linux image copy*). For this reason, a complete microSD card bootable image for S32V234EVB that can be configured from Windows OS is distributed alongside NXP Vision Toolbox.

Follow the next steps to create a bootable SD card for S32V234 Evaluation Board:

1. Begin by inserting a microSD card with at least 16GB capacity in your Host PC running Windows OS. The Windows OS should be able to recognize the SD card and assign a drive letter (e.g.: "D:")



2. From MATLAB command window run the command:

```
nxpvt_create_target('..path to SD card image..\S32V234-  
EVB_29288_image.gz', 'D:');
```

This command will format the card and then is going to copy all the required files from the *.gz image to the SD Card for booting up the Linux on S32V234EVB.

3. The copying process might take a while depending on the SD Card class type. During the process the following message will be shown on the screen. Wait until the copying process is finalized.

```

C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create\create...
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>SET MSYS2_BASH_PATH=C:/NXP/VisionSDK_S32V2xx_RTM_1_1_1/s32v234_sdk\..\msys32\usr\bin
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>set sourceImg=C:/ADAST_GIT/adast_dp/sdcard_img/S32V234-EVB_29288_image.gz
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>set dstImgWin=D:
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>C:/NXP/VisionSDK_S32V2xx_RTM_1_1_1/s32v234_sdk\..\msys32\usr\bin\bash.exe -i -c "source getLinuxStyleDrivePath.sh D:" 1>tmp
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>set /p dstImg= 0<tmp
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>IF "sdb" == "" (ECHO "Drive letter D: not found." ) ELSE (ECHO "Using drive sdb." )
"Using drive sdb."
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>C:/NXP/VisionSDK_S32V2xx_RTM_1_1_1/s32v234_sdk\..\msys32\usr\bin\bash.exe -i -c "gunzip -c C:/ADAST_GIT/adast_dp/sdcard_img/S32V234-EVB_29288_image.gz | dd of=/dev/sdb bs=16M status=progress"
163250176 bytes (163 MB, 156 MiB) copied, 14.0048 s, 11.7 MB/s

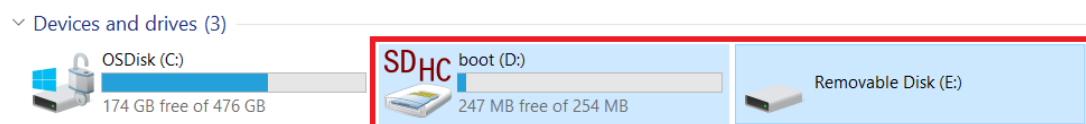
```

```

C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create\create...
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>SET MSYS2_BASH_PATH=C:/NXP/VisionSDK_S32V2xx_RTM_1_1_1/s32v234_sdk\..\msys32\usr\bin
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>set sourceImg=C:/ADAST_GIT/adast_dp/sdcard_img/S32V234-EVB_29288_image.gz
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>set dstImgWin=D:
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>C:/NXP/VisionSDK_S32V2xx_RTM_1_1_1/s32v234_sdk\..\msys32\usr\bin\bash.exe -i -c "source getLinuxStyleDrivePath.sh D:" 1>tmp
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>set /p dstImg= 0<tmp
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>IF "sdb" == "" (ECHO "Drive letter D: not found." ) ELSE (ECHO "Using drive sdb." )
"Using drive sdb."
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>C:/NXP/VisionSDK_S32V2xx_RTM_1_1_1/s32v234_sdk\..\msys32\usr\bin\bash.exe -i -c "gunzip -c C:/ADAST_GIT/adast_dp/sdcard_img/S32V234-EVB_29288_image.gz | dd of=/dev/sdb bs=16M status=progress"
15494971392 bytes (15 GB, 14 GiB) copied, 1596 s, 9.7 MB/s
dd: error writing '/dev/sdb': Input/output error
0+157699 records in
0+157698 records out
15502147584 bytes (16 GB, 14 GiB) copied, 1596.69 s, 9.7 MB/s
C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\internals\target\create>rm tmp

```

4. After the copying process is completed, you should be able to see an additional drive mapped on your system (e.g. E) that cannot be accessed since is a ext3 partition.



5. Check that the initial mapped drive (e.g. D) contains: Image and s32v234-evb files

SD > NXL71469 > boot (D:)				
Name	Date modified	Type	Size	
Image	5/16/2018 11:07 A...	File	7,650 KB	
s32v234-evb.dtb	5/16/2018 11:03 A...	DTB File	23 KB	

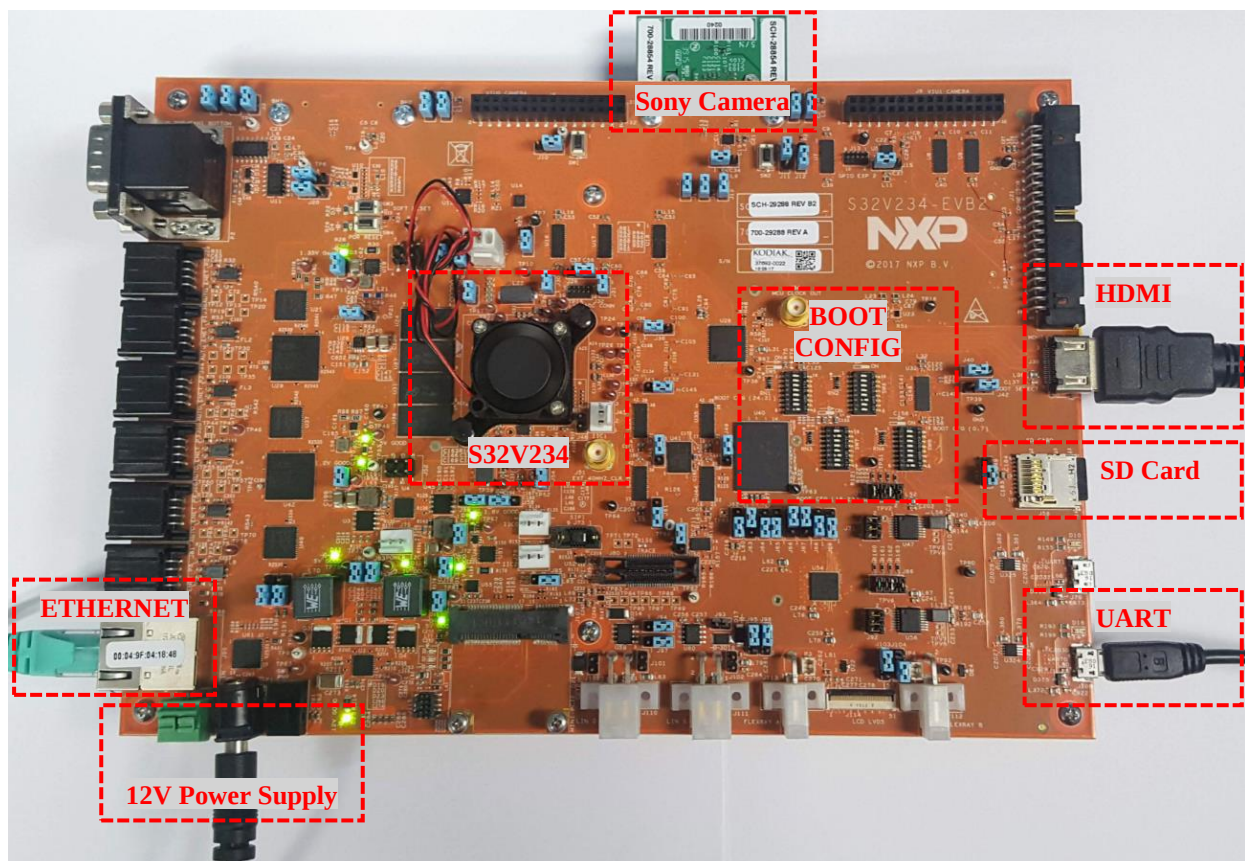
6. Remove the SD card from the Host PC and check the next section for details on how to bootup the S32V234 Evaluation Board

3.3.2 S32V234 Evaluation Board Configuration

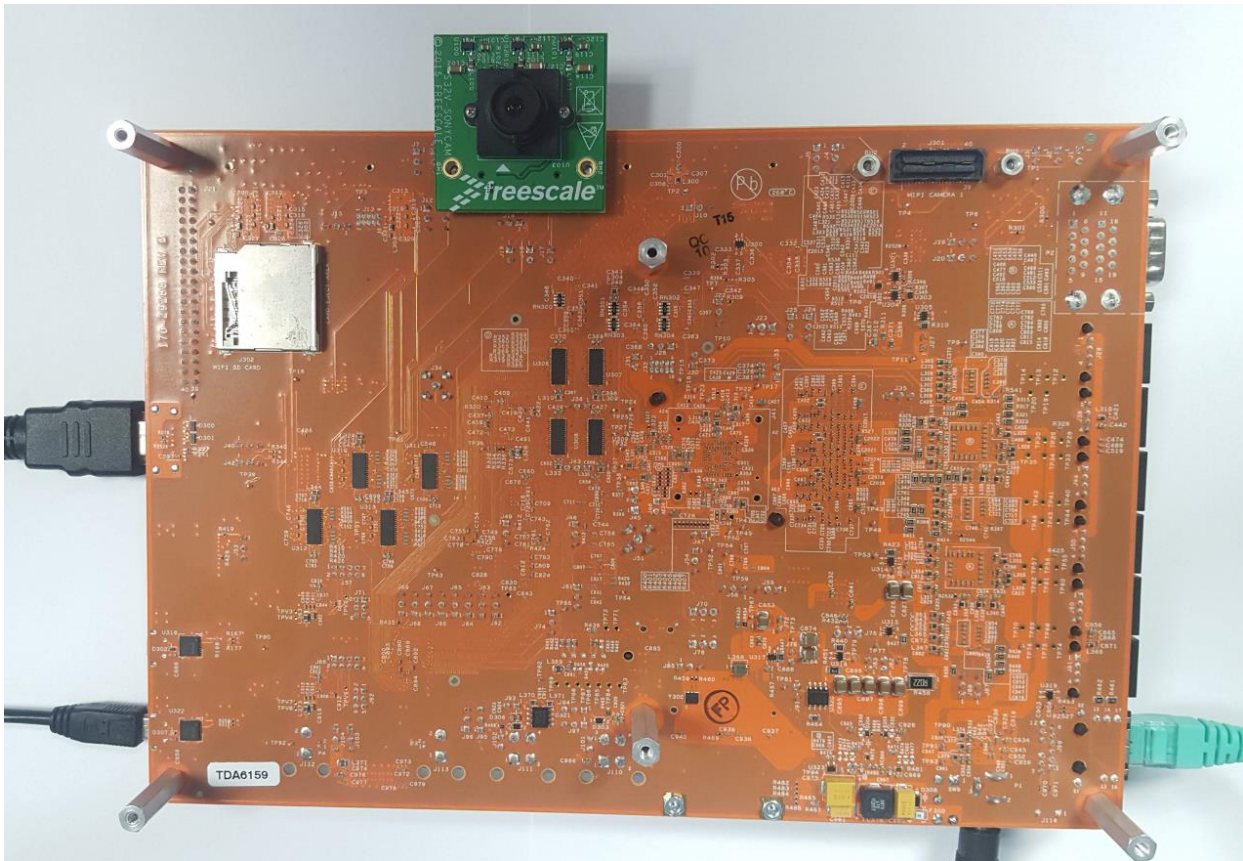
All the examples provided with the Vision Toolbox were developed on S32V234EVB. Additional information about this development kit can be found on NXP official web page [here](#).

Before running any example on the S32V234EVB you need to perform the following steps:

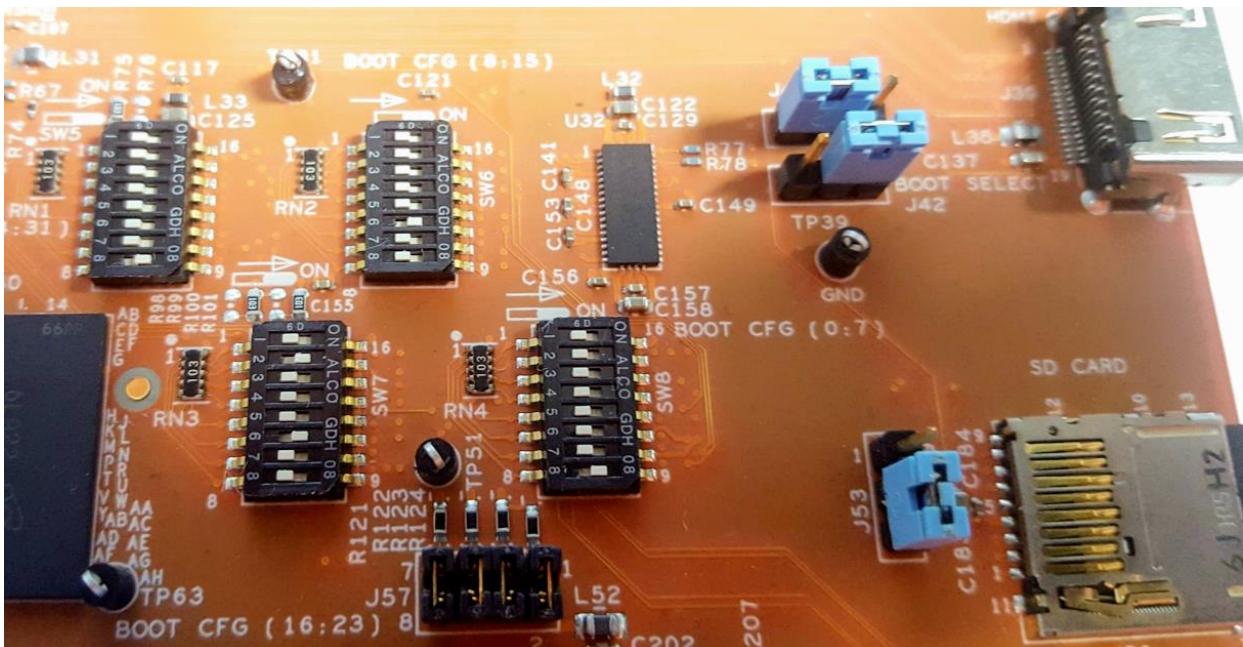
1. Insert the micro SD-card that has been configured in the previous section into the micro SD card slot
2. Insert the Sony camera into the MIPI Camera 0 port. The Sony camera is used for capturing the video frames used for computer vision processing
3. Insert an Ethernet cable in the ETH port. This will be used for downloading the application via TCP/IP
4. Connect the S32V234EVB via a microUSB cable with your Host PC. This is used for finding the IP of the board.
5. Set the Jumper J36 into position 1-2 to allow data to be displayed on a LCD monitor via HDMI
6. Connect a LCD monitor via HDMI cable with S32V234EVB
7. Configure the S32V234EVB Boot Configuration switches as shown below
8. Power on the board



The back side of the S32V234 Evaluation board is shown below:



The boot configuration switches position are shown below:

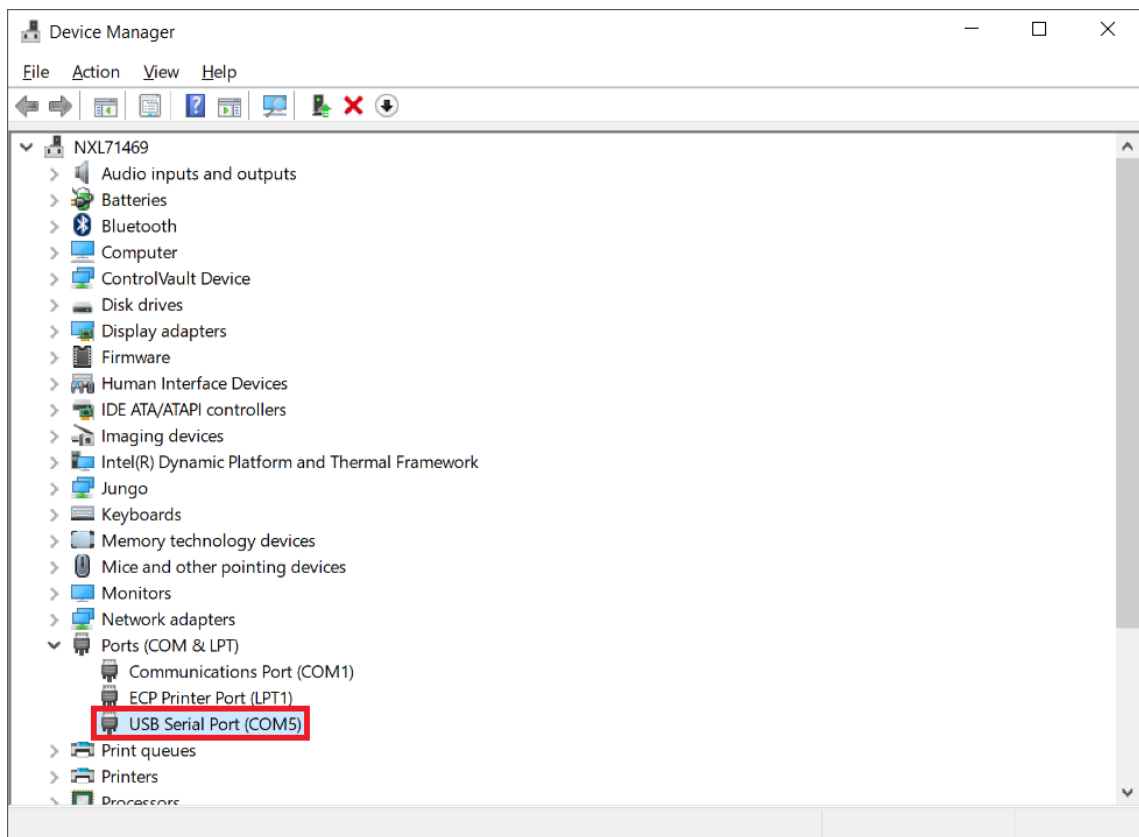


3.3.3 USB to UART connection

The UART enables the users to configure the boot up process and to find out the board IP address after reset.

To be able to connect the S32V234 board to the host PC running on Windows OS, the USB to UART bridge FDT Driver needs to be installed. Follow the next steps to configure the UART communication:

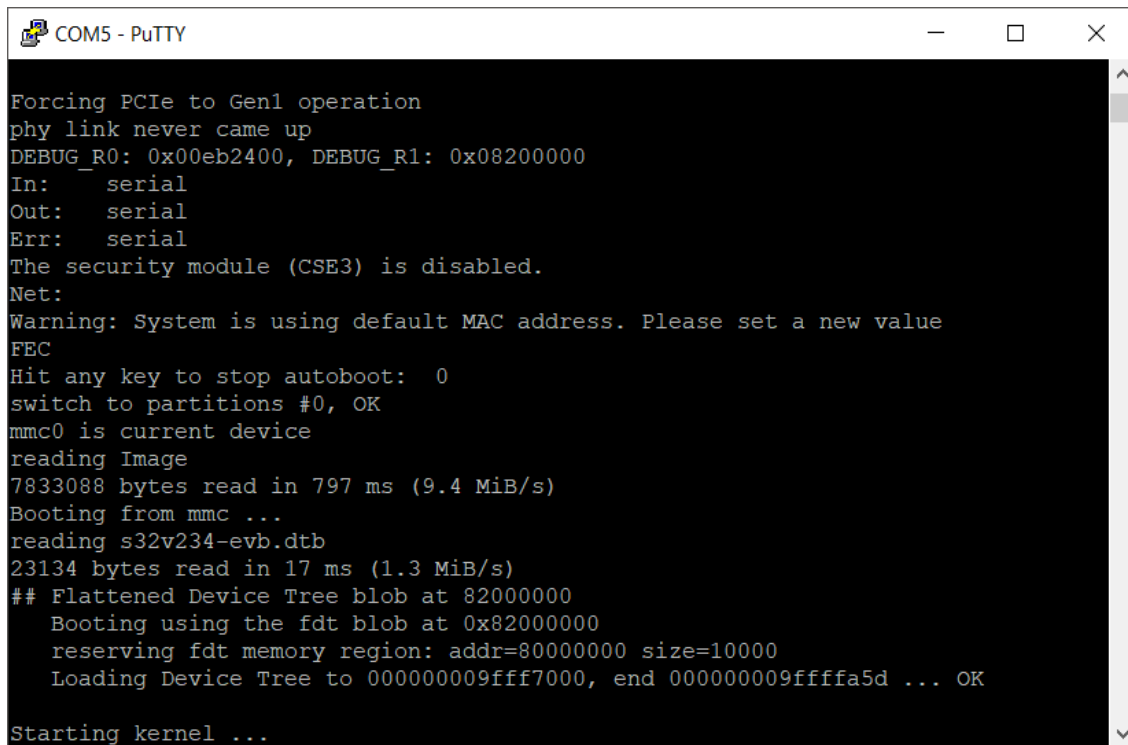
1. Download the USB to UART driver from <http://www.ftdichip.com/Drivers/VCP.htm>
2. Turn on the boards.
3. Install the Driver
4. Open Device Manager and find the COM port assigned



5. In Port Settings in the Driver's properties, set following settings: bits per second: 115200, data bits: 8, parity: None, stop bits: 1, flow control: None

After successful setup of the driver, it is possible to connect turned-on boards with console client application (e.g. Putty) on COM<number> and 115200 bps.

After resetting the S32V234 evaluation board you should be able to see the bootup process.

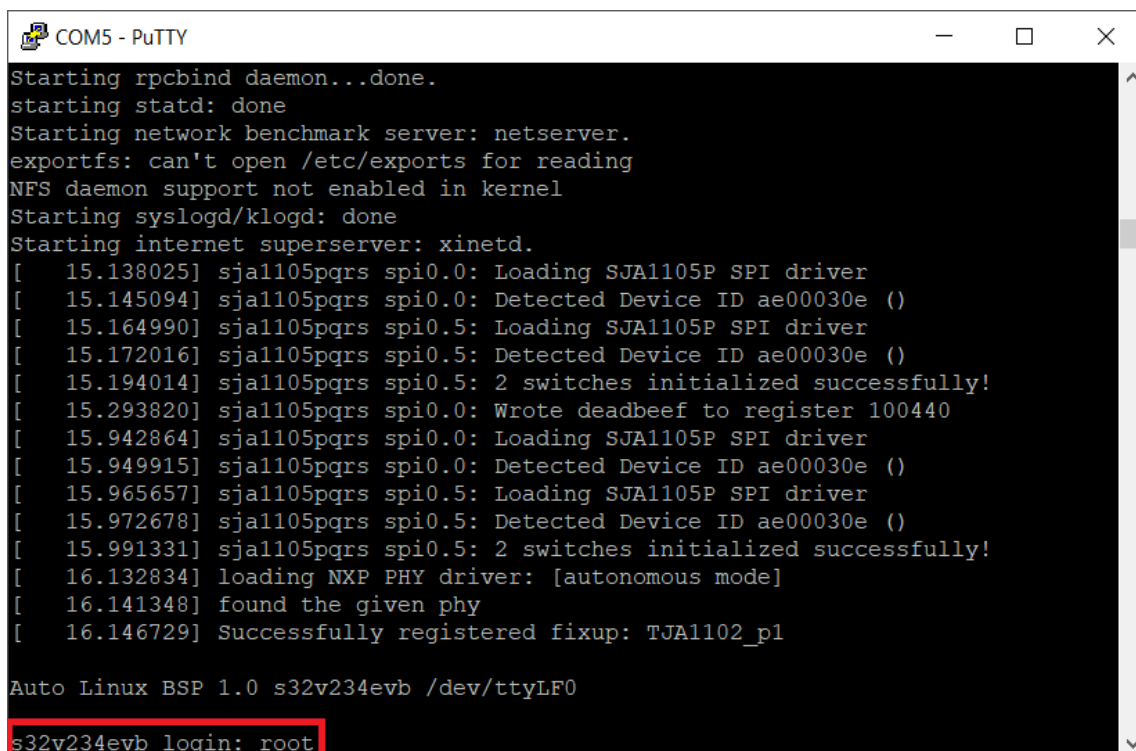


```
COM5 - PuTTY

Forcing PCIe to Gen1 operation
phy link never came up
DEBUG_R0: 0x00eb2400, DEBUG_R1: 0x08200000
In:    serial
Out:   serial
Err:   serial
The security module (CSE3) is disabled.
Net:
Warning: System is using default MAC address. Please set a new value
FEC
Hit any key to stop autoboot: 0
switch to partitions #0, OK
mmc0 is current device
reading Image
7833088 bytes read in 797 ms (9.4 MiB/s)
Booting from mmc ...
reading s32v234-evb.dtb
23134 bytes read in 17 ms (1.3 MiB/s)
## Flattened Device Tree blob at 82000000
Booting using the fdt blob at 0x82000000
reserving fdt memory region: addr=80000000 size=10000
Loading Device Tree to 000000009fff7000, end 000000009fffa5d ... OK

Starting kernel ...
```

Wait until the Linux is loaded and type `root` to log in.

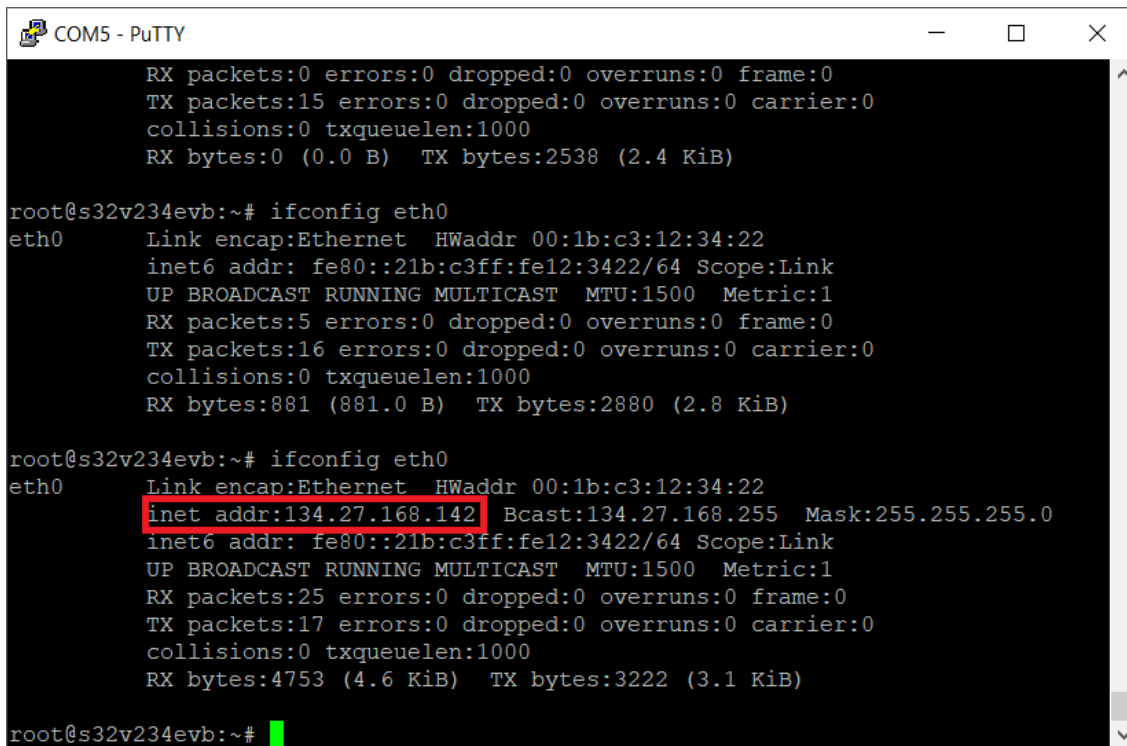


```
COM5 - PuTTY

Starting rpcbind daemon...done.
starting statd: done
Starting network benchmark server: netserver.
exportfs: can't open /etc/exports for reading
NFS daemon support not enabled in kernel
Starting syslogd/klogd: done
Starting internet superserver: xinetd.
[ 15.138025] sjal105pqrs spi0.0: Loading SJA1105P SPI driver
[ 15.145094] sjal105pqrs spi0.0: Detected Device ID ae00030e ()
[ 15.164990] sjal105pqrs spi0.5: Loading SJA1105P SPI driver
[ 15.172016] sjal105pqrs spi0.5: Detected Device ID ae00030e ()
[ 15.194014] sjal105pqrs spi0.5: 2 switches initialized successfully!
[ 15.293820] sjal105pqrs spi0.0: Wrote deadbeef to register 100440
[ 15.942864] sjal105pqrs spi0.0: Loading SJA1105P SPI driver
[ 15.949915] sjal105pqrs spi0.0: Detected Device ID ae00030e ()
[ 15.965657] sjal105pqrs spi0.5: Loading SJA1105P SPI driver
[ 15.972678] sjal105pqrs spi0.5: Detected Device ID ae00030e ()
[ 15.991331] sjal105pqrs spi0.5: 2 switches initialized successfully!
[ 16.132834] loading NXP PHY driver: [autonomous mode]
[ 16.141348] found the given phy
[ 16.146729] Successfully registered fixup: TJA1102_p1

Auto Linux BSP 1.0 s32v234evb /dev/ttyLF0
s32v234evb login: root
```

Type `ifconfig eth0` to find the IP address of the board.



```
COM5 - PuTTY
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B)  TX bytes:2538 (2.4 KiB)

root@s32v234evb:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1b:c3:12:34:22
          inet6 addr: fe80::21b:c3ff:fe12:3422/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:881 (881.0 B)  TX bytes:2880 (2.8 KiB)

root@s32v234evb:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1b:c3:12:34:22
          inet addr:134.27.168.142  Bcast:134.27.168.255  Mask:255.255.255.0
          inet6 addr: fe80::21b:c3ff:fe12:3422/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:25 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4753 (4.6 KiB)  TX bytes:3222 (3.1 KiB)

root@s32v234evb:~#
```

This is the IP address needed to download the application from MATLAB to the S32V234 after code generation and build is completed.

3.3.4 Compile and Run on S32V234

To generate the code and a face detection application on the S32V234 EVB begin by opening the `face_detection_camera_main.m` MATLAB script file from the `...examples/apps/face detection` folder:

```
function face_detection_camera_main() %#codegen
    width = uint32(1280);
    height = uint32(720);

    if coder.target('MATLAB')
        input = nxpvt.videoinput('winvideo', 1, width, height);
    else
        input = nxpvt.videoinput('sony', 1, width, height, true, false);
    end

    fdetector = nxpvt.FaceDetector(width, height);

    fNum = int32(0);
    fps = int32(0);
    while true
        nxpvt.tic;
        fNum = fNum + 1;
        frame = input.getsnapshot();

        % Get faces.
        [bbox, l] = fdetector.GetFaces(frame);
        frame = nxpvt.cv.rectangle(frame, bbox, [255, 0, 0], 5);
        frame = nxpvt.cv.putText(frame, sprintf('FPS: %d', fps), [50, 100],
            nxpvt.cv.HersheyFonts.FONT_HERSHEY_SIMPLEX, 2, [0, 255, 0], 2);
        nxpvt.imshow(frame);

        fps = int32(fix(1/nxpvt.toc));
        fprintf('%d FPS: %d, Faces detected: %d, \n', fNum, fps, int32(1));
    end
end
```

Define a structure `config` that controls the behavior of the code generation and target configuration options. You can find all configuration fields explained in the next table.

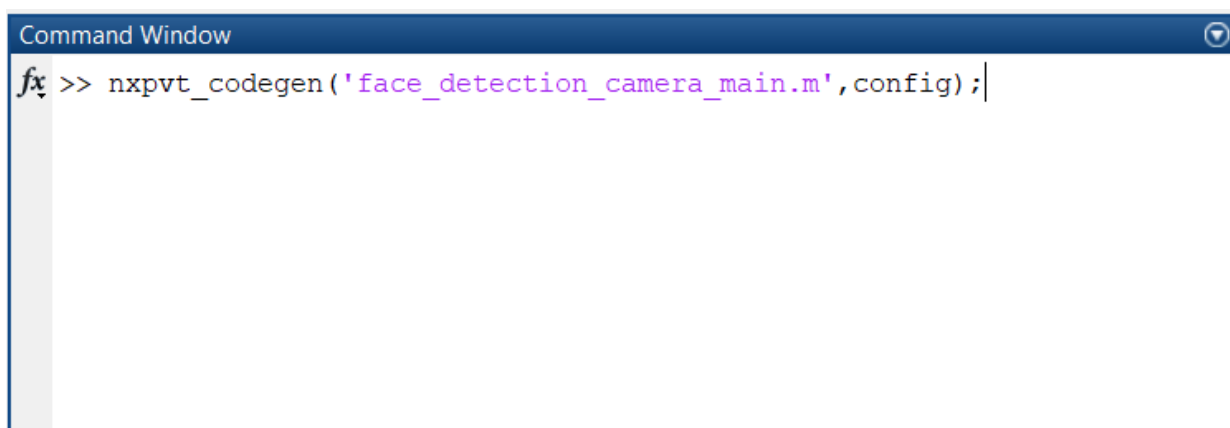
Config fields	Description
MakeJobs	Number of CPU jobs used for parallel compilation
Optimize	When set to true, the cross compilers will use O3 optimization level. This ensure the best performance
Deploy	When set to true, the NXP Vision Toolbox will deploy the application on to the target after code generation stage. For the deployment to work the user needs to configure <code>config.TargetIpAddress</code> as well. If this is not set explicitly to true it will default to false
DeployPath	The path where the executable will be copied on the target. If left empty the <code>'/examples/'</code> folders will be used. This path should be an absolute path

TargetIpAddress	The IP address for the target
RemoteFilename	The name of the executable on the target. If left empty it will default to the entryFunc name with the elf extension instead of the .m extension
ExtraFiles	Files that are used by the elf (e.g.: videos, images). If left empty no extra files will be copied on the target. The paths for this file should be relative to the DeployPath

Here is an example for config structure configuration:

```
config.MakeJobs      = 8;
config.Optimize      = true;
config.Deploy        = true;
config.TargetIpAddress = '134.27.168.142';
config.DeployPath     = '/examples/face_detection_camera_main/';
config.ExtraFiles     = {'../data/lbpcascade_frontalface.xml',
                        'lbpcascade_frontalface.xml'};
```

To start the code generation process, invoke nxpvt_codegen function in MATLAB console.



The image shows a MATLAB Command Window with the following text: `fx >> nxpvt_codegen('face_detection_camera_main.m',config);`

After the application code generation and build is completed you should see this in your MATLAB Command Window.

The screenshot shows the MATLAB IDE with the 'face_detection_camera_main.m' file open. The Command Window displays the following output:

```

arm64/gnu/optimized C:\nxpvt_imread_c-cpp.o
arm64/gnu/optimized C:\nxpvt_imshow_c-cpp.o
arm64/gnu/optimized C:\nxpvt_imwrite_c-cpp.o
arm64/gnu/optimized C:\nxpvt_umat_wrapper-cpp.o
arm64/gnu/optimized C:\nxpvt_utils-cpp.o
arm64/gnu/optimized C:\nxpvt_videoinput-cpp.o
arm64/gnu/optimized C:\nxpvt_videoinput_wrapper-cpp.o
arm64/gnu/optimized C:\nxpvt_videoreader-cpp.o
arm64/gnu/optimized C:\nxpvt_videoreader_wrapper-cpp.o
arm64/gnu/optimized C:\tinyxml2-cpp.o
arm64/gnu/optimized LD face_detection_camera_main.elf <= nxpvt_FaceDetector-cpp.o nxpvt_FaceDetector_wrapper-cpp.o CascadeClas
make[3]: Leaving directory 'C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps\face_detection'
make[2]: Leaving directory 'C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps'
make[1]: Leaving directory 'C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps'

```

The Vision Toolbox will automatically download the application to the target:

The screenshot shows a terminal window with the following output:

```

C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps\face_detection>SE
T params=C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps\face_detection>SE
T params=134.27.168.130 codegen//exe//face_detection_camera_main/build-v234ce-gnu-linux-o//face_detection_camera_main.e
lf face_detection_camera_main.elf 'examples/' '../data/lbpcascade_frontalface.xml' 'lbpcascade_frontalface.xml'

C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps\face_detection>SE
T params=134.27.168.130 codegen//exe//face_detection_camera_main/build-v234ce-gnu-linux-o//face_detection_camera_main.e
lf face_detection_camera_main.elf 'examples/' '../data/lbpcascade_frontalface.xml' 'lbpcascade_frontalface.xml'

C:\Users\nxa14941\Documents\MATLAB\Add-Ons\Toolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps\face_detection>C:
/NXP/VisionSDK_S32V2xx_RT1_1_1/s32v234_sdk\..\msys32\usr\bin\bash.exe -c "C:\Users\nxa14941\Documents\MATLAB\Add-Ons\T
oolboxes\NXP_Vision_Toolbox_for_S32V234\code\examples\apps\face_detection\deploy\deployOnTarget.sh 134.27.168.130 codegen//exe//fac
e_detection_camera_main/build-v234ce-gnu-linux-o//face_detection_camera_main.elf face_detection_camera_main.elf 'exampl
es/' '../data/lbpcascade_frontalface.xml' 'lbpcascade_frontalface.xml'"
OpenSSH already installed.
mkdir -p 'examples/'
=====
Calling copyExtraFiles.sh with 134.27.168.130 codegen//exe//face_detection_camera_main/build-v234ce-gnu-linux-o//face_d
etection_camera_main.elf face_detection_camera_main.elf /examples/ ../data/lbpcascade_frontalface.xml lbpcascade_fron
talface.xml
Copying extra files...
../data/lbpcascade_frontalface.xml =====> lbpcascade_frontalface.xml
ssh root@134.27.168.130 mkdir -p /examples/.
The authenticity of host '134.27.168.130 (134.27.168.130)' can't be established.
ECDSA key fingerprint is SHA256:JqsKwbjBh18oGBXvhFXD+4uiNFHv1UF7HwPkY0WPszw.
Are you sure you want to continue connecting (yes/no)? yes

```

And you should be able to detect people's faces with NXP S32V234 on board camera



Congratulations! You succeeded with running your first example created with Vision Toolbox for S32V234

How to Reach Us:**Home Page:**

www.nxp.com

Web Support:

www.nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

NXP Semiconductor reserves the right to make changes without further notice to any products herein. NXP Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. NXP Semiconductor does not convey any license under its patent rights nor the rights of others. NXP Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the NXP Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use NXP Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold NXP Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that NXP Semiconductor was negligent regarding the design or manufacture of the part.

MATLAB, Simulink, Stateflow, Handle Graphics, and Real-Time Workshop are registered trademarks, and TargetBox is a trademark of The MathWorks, Inc. Microsoft and .NET Framework are trademarks of Microsoft Corporation. Flexera Software, Flexlm, and FlexNet Publisher are registered trademarks or trademarks of Flexera Software, Inc. and/or InstallShield Co. Inc. in the United States of America and/or other countries.

NXP, the NXP logo, CodeWarrior and ColdFire are trademarks of NXP Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Flexis and Processor Expert are trademarks of NXP Semiconductor, Inc. All other product or service names are the property of their respective owners

©2018 NXP Semiconductors. All rights reserved.

