

Getting started with Boot2Qt for i.MX Developer's Kits

Embedded Artists AB

Jörgen Ankersgatan 12
SE-211 45 Malmö
Sweden

<https://www.EmbeddedArtists.com>

Copyright 2019 © Embedded Artists AB. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Embedded Artists AB.

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: www.embeddedartists.com/contact.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1	Document Revision History	4
2	Introduction.....	5
2.1	Conventions.....	5
3	Build Boot to Qt Linux image.....	6
3.1	Requirements	6
3.2	Git	6
3.3	Meta-boot2qt repository	6
3.4	Build image.....	7
3.5	Final result	7
3.6	Other images	7
4	Deploying the image.....	9
4.1	OTG boot mode – J2 jumper	9
4.2	UUU	9
4.3	Download the bundle	10
4.4	Run the tool in Ubuntu.....	10
4.5	Run the tool in Windows.....	10
4.6	Troubleshoot	11
4.7	Use your own image	12
5	Run the demo	13
5.1	iMX6 Quad COM	13
5.2	iMX7 Dual uCOM / COM	13
5.3	iMX8M Mini uCOM	14
5.4	iMX8M Quad COM	15
6	Next step.....	16

1 Document Revision History

<i>Revision</i>	<i>Date</i>	<i>Description</i>
A	2019-11-22	First release

2 Introduction

This document provides you with step-by-step instructions for how to get started with Boot2Qt for Embedded Artists iMX6, iMX7, and iMX8 based COM boards. Boot2Qt is part of *Qt for Device Creation* and is a light-weight, Qt-optimized, full software stack for embedded Linux systems.

The Boot2Qt demo is currently available for:

- iMX8M Mini uCOM
- iMX8M Quad COM
- iMX6 Quad COM
- iMX7 Dual COM
- iMX7 Dual uCOM

Note: It is possible to test and evaluate Boot2Qt by following these instructions, but if you would like to develop your own application and sell a product you must have a **commercial license** from Qt.

<https://www.qt.io/licensing/>

Additional documentation you might need is.

- *Boot to Qt Software stack* - <https://doc.qt.io/QtForDeviceCreation/qt2-index.html>
- *Qt for Device Creation* - <https://doc.qt.io/QtForDeviceCreation/>
- *Working with Yocto to build Linux* – found on Embedded Artists website.
- Getting started guide - <https://www.embeddedartists.com/getting-started/>

2.1 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

Constant width text – is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the  
development workstation, i.e., on the workstation where you edit,  
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,  
input given to the terminal attached to the COM Board
```

```
This field is used to illustrate example code or excerpt from a  
document.
```

3 Build Boot2Qt Linux image

3.1 Requirements

If you plan to build a Boot2Qt image you need a Linux host machine that is setup in the same way as when working with Yocto. The instructions in this document has been tested on a machine running **Ubuntu 18.04**.

Install required packages:

```
$ sudo apt-get install gawk curl git-core git-lfs diffstat unzip
p7zip-full texinfo gcc-multilib build-essential chrpath libsdl1.2-
dev xterm gperf bison g++-multilib git-lfs install
```

If you get build errors that could be related to missing packages please look at the required packages listed in the documentation from Qt and also Yocto.

- Qt: <https://doc.qt.io/QtForDeviceCreation/qtee-custom-embedded-linux-image.html#requirements>
- Yocto: <https://www.yoctoproject.org/docs/current/ref-manual/ref-manual.html#ubuntu-packages>

3.2 Git

Git is being used to checkout repositories. If you haven't already configured Git you might at least have to specify name and e-mail address as shown below. Change "Your name" to your actual name and "Your e-mail" to your e-mail address.

```
$ git config --global user.name "Your name"
$ git config --global user.email "Your e-mail"
```

3.3 Meta-boot2qt repository

Embedded Artists has made a fork of the official meta-boot2qt repository from Qt where support has been added for the Embedded Artists COM boards.

Create a working directory (replace `myworkdir` with your own directory name).

```
$ mkdir ~/myworkdir
$ cd ~/myworkdir
```

Clone the meta-boot2qt repository into the sources directory:

```
$ git clone https://github.com/embeddedartists/meta-boot2qt.git sources/meta-
boot2qt
```

Checkout the branch you want to work with. The table below lists the branches that are available.

Branch name	Versions
ea-thud	Qt: 5.13.2 u-boot: 2018.03 Linux: 4.14.78

```
$ cd sources/meta-boot2qt
$ git checkout <branch>
```

3.4 Build image

Initialize the build for the board / device you are using. The supported devices are listed in the table below.

Device name	Description
imx6qea-com	iMX6 Quad COM board / kit
imx7dea-com	iMX7 Dual COM board / kit
imx7dea-ucom	iMX7 Dual uCOM board / kit
imx8mmea-ucom	iMX8M Mini uCOM board / kit
imx8mqea-com	iMX8M Quad COM board / kit

First make sure you are in your working directory. If you just checked out a branch as described in the previous section you must first move up two directories.

```
$ cd ~/myworkdir
$ ./sources/meta-boot2qt/b2qt-init-build-env init --device <device>
```

Setup the environment. Use the same device name as listed in the table above.

```
$ export MACHINE=<device>
$ source ./setup-environment.sh
```

Build the image.

Note: Depending on the capabilities of your computer this can take several hours.

```
$ bitbake b2qt-embedded-qt5-image
```

3.5 Final result

When the build has completed the result will be located in the following directory.

```
~/myworkdir/<build dir>/tmp/deploy/images/<device>/
```

There are several files in this directory. The one that contains the complete file system and the one used in the deployment scripts is the wic-file. It will typically be named `b2qt-embedded-qt5-image-<device>.wic`.

3.6 Other images

It is also possible to build an SDK.

```
$ bitbake meta-toolchain-b2qt-embedded-qt5-sdk
```

The result will be located in.

```
~/myworkdir/<build dir>/tmp/deploy/sdk/
```

If you want to build the toolchain for Windows.

```
$SDKMACHINE=i686-mingw32 bitbake meta-toolchain-b2qt-embedded-qt5-  
sdk
```


4 Deploying the image

NXP's Manufacturing Tool called UUU is used to deploy an image to the target. You can download a bundle containing the UUU tool, scripts and pre-built Boot2Qt images from <http://imx.embeddedartists.com/>.

4.1 USB OTG boot mode – J2 jumper

Before deploying an image, the board must be put into USB OTG boot mode.

This is accomplished by closing the J2 jumper on the COM Carrier board; see Figure 1 or Figure 2 to locate the jumper. Please note that in the figure the jumper is in open state which means that the COM board will boot from eMMC. Also note that a power cycle is needed to place the COM board in USB OTG boot mode.

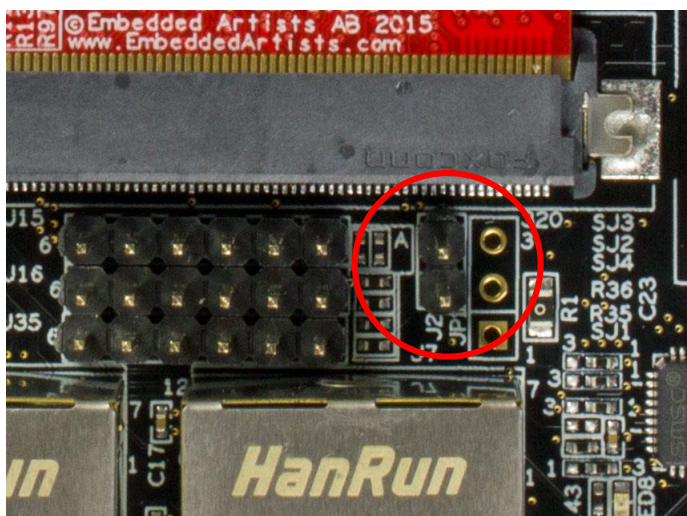


Figure 1 - J2 jumper (opened state) on COM Carrier Board V1



Figure 2 - J2 jumper (opened state) on a COM Carrier Board V2

4.2 UUU

UUU (Universal Update Utility) is version 3 of MFGTool but it has been rewritten, is publicly available on GitHub (<https://github.com/NXPmicro/mfgtools>) and it can be run on both Windows and Linux while the older versions of MFGTool were limited to Windows only. UUU is the preferred utility to use when downloading images.

UUU can be used to write images to the board. This tool is sending files and instructions over USB and the board must be set in OTG boot mode for it to work.

Useful links:

- UUU on GitHub: <https://github.com/NXPmicro/mfgtools>
- UUU release page: <https://github.com/NXPmicro/mfgtools/releases>

4.3 Download the bundle

Download the bundle for the board you are using from <http://imx.embeddedartists.com/>

Unpack the file somewhere on your computer. Below is a description of some of the content in the file.

- uuu (root): Contains a README file.
- uuu/* .uuu: The different download configurations.
- uuu/files/: Contains pre-compiled versions of images. The tool will look in this directory when selecting images to download to the board.

4.4 Run the tool in Ubuntu

In Linux, open a terminal and navigate to the folder where the uuu file was unpacked. Make sure that the tool is executable and then execute the tool:

```
$ cd ~/uuu_imx8mm_ucom_4.14.78_boot2qt
$ chmod +x ./uuu
$ sudo ./uuu boot2qt.uuu
```

The terminal will show a progress bar like illustrated below while it is running:

```
andli@lenovo:~/uuu_imx8mq_com_4.14.78$ sudo ./uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 0   Failure 0

2:24  20/23  [=====47%                ] FBK: ucp files/ea-image-base-imx8mqea-com.tar.bz2 t:-
```

After a successful run it will look like this:

```
andli@lenovo:~/uuu_imx8mq_com_4.14.78$ sudo ./uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 1   Failure 0

2:24  23/23  [Done                ] FBK: DONE
```

If a problem occurs then the program will terminate and print an error message like this

```
andli@lenovo:~/uuu_imx8mq_com_4.14.78$ sudo ./uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 0   Failure 1

2:24  1/ 1  [HID(W):LIBUSB_ERROR_IO      ] SDP: boot -f files/u-boot-imx8mqea-com.bin
andli@lenovo:~/uuu_imx8mq_com_4.14.78$
```

4.5 Run the tool in Windows

In Windows, open a Command Prompt, navigate to the folder where the uuu file was unpacked and then run the tool:

```
C:\> cd c:\temp\uuu_imx8mm_ucom_4.14.78_boot2qt
C:\temp\uuu_imx8mm_ucom_4.14.78_boot2qt> uuu.exe boot2qt.uuu
```

The terminal will show a progress bar similar to below while it is running:

```
c:\temp\uuu_imx8mq_com_4.14.78>uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 0    Failure 0

1:23  20/23  [=====> 34%                ] FBK: ucp files/ea-image-base-imx8mqea-com.tar.bz2 t:-
```

After a successful run it will look like this:

```
c:\temp\uuu_imx8mq_com_4.14.78>uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 1    Failure 0

1:23  23/23  [Done                            ] FBK: DONE

c:\temp\uuu_imx8mq_com_4.14.78>
```

If a problem occurs then the program will terminate and print an error message like this

```
c:\temp\uuu_imx8mq_com_4.14.78>uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 0    Failure 1

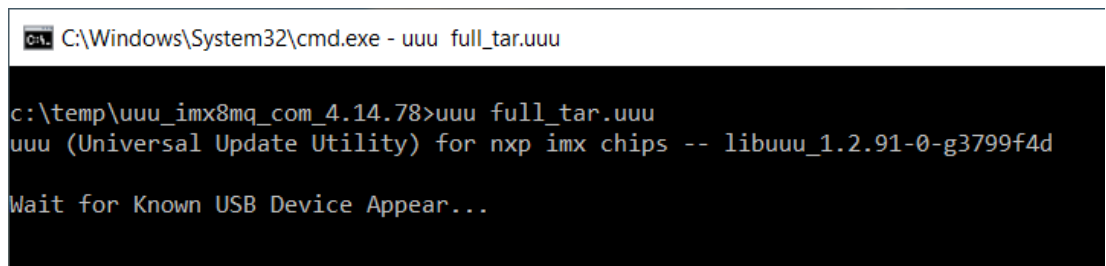
1:23  20/23  [BuIk(R):LIBUSB_ERROR_PIPE     ] FBK: ucp files/ea-image-base-imx8mqea-com.tar.bz2 t:-

c:\temp\uuu_imx8mq_com_4.14.78>
```

4.6 Troubleshoot

Some common problems and solutions:

- **The first time you run uuu on your computer it fails.**
This is likely because of USB driver installation. Let the driver install, reset the hardware and then run the uuu command again. In Windows it is three different drivers that are needed so this procedure might have to be repeated three times - each time the procedure gets a little bit further.
- **UUU appears to hang with a "Wait for Known USB Device Appear..." message like this:**



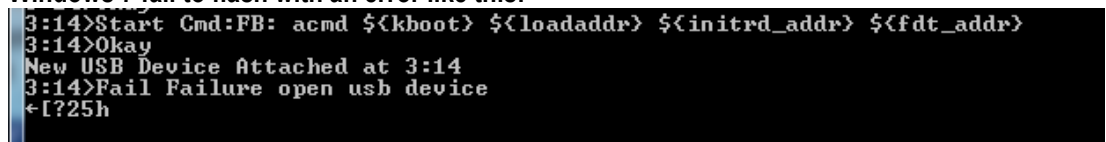
```
cmd C:\Windows\System32\cmd.exe - uuu full_tar.uuu

c:\temp\uuu_imx8mq_com_4.14.78>uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Wait for Known USB Device Appear...
```

This means that the hardware is either not connected to the computer with the USB cable or it is not in the USB OTG boot mode. Check and follow instructions in section 4.1 and then run the uuu command again.

- **Windows 7 fail to flash with an error like this:**



```
3:14>Start Cmd:FB: acmd ${kboot} ${loadaddr} ${initrd_addr} ${fdt_addr}
3:14>Okay
New USB Device Attached at 3:14
3:14>Fail Failure open usb device
←[?25h
```

It could be due to a driver problem. Follow instructions here:

<https://github.com/NXPmicro/mfgtools/wiki/WIN7-User-Guide>

- **Windows 7 terminal does not appear as in the screenshots**
This is because Windows 7 does not support what the UUU tool calls "VT mode" so it defaults to verbose mode which has a lot more printouts and no progress bar.
- **UUU in Ubuntu reports failure to open usb device:**

```
andli@lenovo:~/uuu_imx8mq_com_4.14.78$ ./uuu full_tar.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.2.91-0-g3799f4d

Success 0      Failure 1

2:24  1/ 0  [Failure open usb device, Try ]
andli@lenovo:~/uuu_imx8mq_com_4.14.78$
```

This happens if the uuu program is not executed with the correct rights. Either use "sudo uuu" or setup udev rules so that sudo rights are not needed. The instructions for how to create the udev rules are built into the tool so run "uuu -udev" and then follow the steps:

```
andli@lenovo:~/uuu_imx8mq_com_4.14.78$ ./uuu -udev
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="012f", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0129", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0076", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0054", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0061", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0063", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0071", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="007d", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="15a2", ATTRS{idProduct}=="0080", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0128", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0126", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0135", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="0134", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1fc9", ATTRS{idProduct}=="012b", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="0525", ATTRS{idProduct}=="b4a4", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="0525", ATTRS{idProduct}=="b4a4", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="066f", ATTRS{idProduct}=="9afe", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="066f", ATTRS{idProduct}=="9bff", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="0525", ATTRS{idProduct}=="a4a5", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="18d1", ATTRS{idProduct}=="0d02", MODE="0666"

1: put above udev run into /etc/udev/rules.d/99-uuu.rules
   sudo sh -c "uuu -udev >> /etc/udev/rules.d/99-uuu.rules"
2: update udev rule
   sudo udevadm control --reload-rules
andli@lenovo:~/uuu_imx8mq_com_4.14.78$
```

4.7 Use your own image

If you have built your own image as described in chapter 3 then you can deploy this image by copying the wic file from the build and replace the wic file in the `uuu/files` directory. When you have replaced the file follow the instructions above to deploy the image.

5 Run the demo

To be able to run Boot2Qt all you need to do is make sure jumper J2 is in open state (see section 4.1 for more information) and then reset (or power cycle) the board. However, for the demo to be useful you need to have a display connected to the board. The sections below describe how to use the default display interface for a specific board. If you want to use a different display please visit the "Display solutions page": <https://www.embeddedartists.com/display-solutions-for-com-boards/>.

You need to be able to interact with the demo in some way. If your display has a **touch** panel you can use this. If not, you can connect a computer **mouse** with USB interface to the board.

5.1 iMX6 Quad COM

The default display interface is HDMI. Follow the steps below to enable the HDMI interface.

1. Reset the board and stop u-boot from autobooting.
2. Enable HDMI using the `eadisp` command

```
=> eadisp enable hdmi
```

3. For Qt to work properly the display interface must be setup with 32 bits per pixel (bpp)

```
=> setenv cmd_custom fdt set fb_hdmi default_bpp <0x20>
```

4. Save the changes

```
=> saveenv
```

5. Now you can reboot the board

By default, a resolution of 1280x720 is used when enabling HDMI. You can change this using `eadisp conf`. To see which interfaces and resolutions that have been preconfigured in u-boot just enter `eadisp`.

```
=> eadisp
Available display configurations:

   0) lvds0  hannstar:18:64998375,1024,768,220,40,21,7,60,10,0,0,0,0
   1) lvds0  nhd-10.1-1024600af:24:51208521,1024,600,160,160,23,12,0,0,0,0,1,0
   ...
  11) hdmi   1280x720M@60:m24:74161969,1280,720,220,110,20,5,40,5,0,0,0,0
  12) hdmi   1920x1080M@60:m24:148500148,1920,1080,148,88,36,4,44,5,0,0,0,0
  13) hdmi   640x480M@60:m24:25200342,640,480,48,16,33,10,96,2,0,0,0,0
  14) hdmi   720x480M@60:m24:27027027,720,480,60,16,30,9,62,6,0,0,0,0
```

To change to another existing resolution, for example 1920x1080 use `eadisp conf`.

```
=> eadisp conf hdmi 12
=> saveenv
```

For more information about `eadisp` and how to use displays with Embedded Artists Developer's Kits read the document *Adding Displays to iMX Developer's Kits* available on Embedded Artists website.

5.2 iMX7 Dual uCOM / COM

The iMX7 Dual uCOM / COM board supports the RGB display interface. In this example we are using a 7 inch display with resolution 800x480 from Innolux.

1. Reset the board and stop u-boot from autobooting.
2. Enable the rgb interface by using the `eadisp` command.

```
=> eadisp enable rgb
=> saveenv
```

3. By default, the 7-inch display from Innolux is selected as the display to use. There are more displays preconfigured. To see which are available just enter `eadisp`.

```
=> eadisp
Available display configurations:

0)  rgb  Innolux-AT070TN:24:33336667,800,480,89,164,75,75,10,10,0,0,1,0
1)  rgb  nhd-4.3-480272ef:24:9009009,480,272,2,2,2,2,41,10,0,0,1,0
2)  rgb  nhd-5.0-800480tf:24:29232073,800,480,40,40,29,13,48,3,0,0,1,0
3)  rgb  nhd-7.0-800480ef:24:29232073,800,480,40,40,29,13,48,3,0,0,1,0
4)  rgb  umsh-8864:24:9061007,480,272,20,20,20,20,3,3,0,0,1,0
5)  rgb  umsh-8596-30t:24:33264586,800,480,128,120,20,20,8,5,0,0,1,1
6)  rgb  umsh-8596-33t:24:32917475,800,480,200,200,45,45,1,1,0,0,1,1
7)  rgb  rogin-rx050a:24:32917475,800,480,200,200,45,45,1,1,0,0,1,1
```

To change to another display use `eadisp conf`.

```
=> eadisp conf rgb 3
=> saveenv
```

The display from Innolux has a resistive touch panel (AR1021 controller). You can enable it by using the `eatouch` command.

```
=> eatouch
Available Touch Controllers:

1)  ar1021
2)  ilitek
3)  sitronix
4)  egalax
5)  ft5x06
6)  mxt1664
```

Use `eatouch enable` to enable the ar1021 touch controller.

```
=> eatouch enable rgb 1
=> saveenv
```

For more information about `eadisp` and how to use displays with Embedded Artists Developer's Kits read the document *Adding Displays to iMX Developer's Kits* available on Embedded Artists website.

5.3 iMX8M Mini uCOM

The iMX8M Mini application processor only supports MIPI-DSI as display interface, but a MIPI-DSI to HDMI bridge has been added to the uCOM adapter board so you can use an HDMI display in this case.

Set the resolution to use by following the instructions below.

1. Reset the board and stop u-boot from autobooting (that is, boot into u-boot console).

2. Set the video argument in the `extra_bootargs` variable.

```
=> setenv extra_bootargs video=HDMI-A-1:720x480@60
=> saveenv
```

3. Now you can reboot the board.

5.4 iMX8M Quad COM

The default display interface is HDMI.

Set the resolution to use by following the instructions below.

1. Reset the board and stop u-boot from autobooting (that is, boot into u-boot console).
2. Set the video argument in the `extra_bootargs` variable.

```
=> setenv extra_bootargs video=HDMI-A-1:720x480@60
=> saveenv
```

3. Now you can reboot the board.

6 Next step

Following the instructions in this document will get you up-and-running with the Boot2Qt demo. If you want to develop your own application you have to consult the documentation from Qt. You will also need a **commercial license** from Qt.

Below are some useful links.

Customization

<https://doc.qt.io/QtForDeviceCreation/qtee-customization.html>

Deploy your first project with Boot2Qt

<https://doc.qt.io/QtForDeviceCreation/b2qt-tutorial-deploying.html>

Qt for Device Creation

<https://doc.qt.io/QtForDeviceCreation/qtdc-index.html>

Licensing

<https://www.qt.io/licensing/>