



MAAXBOARD-OSM93

YOCTO LINUX BSP DEVELOPMENT GUIDE

Revision History

Revision	Date	Notes
LF6.6.3-1.0.0	12/01/2024	Updates for 6.6.3 release
LF6.6.3-1.0.0	12/17/2024	Update to Tria template. Minor improvements added.
LF6.12.3-1.0.0	07/11/2025	Updates for 6.12.3 release
LF6.6.52-2.2.0	08/01/2025	Reverted to 6.6.52 LTS version

Copyright and Compliance Statements

Copyright Statement

- The MaaXBoard OSM93 single board computer and its related intellectual property are owned by Tria Technologies.
- Tria Technologies has the copyright of this document and reserves all rights. Any part of the document should not be modified, distributed, or duplicated in any approach and form without written permission from Tria.

Disclaimer

- Tria does not make warranty of any kind, either expressed or implied, as to the program source code, software or documents provided with products, and including, but not limited to, warranties of fitness for a particular purpose. The entire risk as to the quality or performance of the program is with the user of products.

Regulatory Compliance

- MaaXBoard OSM93 single board computer has passed the CE, FCC & SRRC certifications.

TABLE OF CONTENTS

MaaXBoard-OSM93	1
Yocto Linux BSP Development Guide.....	1
Revision History.....	2
Copyright and Compliance Statements.....	3
Table of contents.....	4
Introduction.....	6
Prerequisites.....	6
Yocto Build process using docker	6
Docker.....	6
Install the required software packages	7
Fetch Source Code.....	8
Build the images using yocto.....	8
Set the Build Configuration.....	8
Build the image	9
Troubleshooting yocto builds	9
Fetch Errors.....	9
NXP Documentation.....	9
Enable bitbake debug	9
Yocto Project Documentation	9
Standalone Build of u-Boot and Kernel	10
Cross-compile tool chain.....	10
Build U-Boot in a standalone environment.....	11
Build Kernel in a standalone environment.....	14
System power on and boot up	15
Adding custom layers to the yocto build.....	15
Download the new meta-layer.....	15
Pull the correct meta-layer branch to match your build.....	15
Add the new meta-* layer to the build	16
Include the application into the image.....	16

Rebuild the image	16
Appendix	17
Hardware Documents	17
Software Documents.....	17
Contact Information.....	17

INTRODUCTION

This document provides the details required to build the yocto Linux and boot images for the MaaXBoard OSM93. Additional details are included to help users extend the yocto build by adding additional meta layers and including additional applications into the final deployment.

PREREQUISITES

Host PC OS:

- Ubuntu 64-bit OS, 20.04 LTS (recommended) or later LTS version

This document assumes use of Ubuntu 20.04, some packages may differ for other Ubuntu versions.

Hardware:

- Build Computer with at least 300GB of free disk space
- At least 4GB of RAM recommended

YOCTO BUILD PROCESS USING DOCKER

It's recommended to use docker and Ubuntu 20.04 LTS for MaaXBoard builds.

If you don't want to use docker, then skip to the [Install the Required Software Packages section](#).

Docker

Install docker on your Linux computer, if not already installed

```
$ sudo curl -sSL https://get.docker.com | sh
```

Add the current user to the docker group. This step is optional, the benefit is that if you add the current user into the docker group, then you can execute docker commands without sudo.

```
$ sudo usermod -aG docker <your current linux username>
```

Pull the OS image. If your region does not support this image then move on to the next section [Install the Required Software Packages](#) on your linux computer

```
$ docker pull ubuntu:20.04
```

Run the container

```
$ docker container run -it ubuntu:20.04 /bin/bash
```

Add a regular user account

Feel free to change the **tria** account to a username of your choice. Follow any prompts that are presented such as defining a password for the new user.

```
$ adduser tria
$ apt update
$ apt install sudo
$ sudo usermod -aG sudo tria
```

Switch to the new user

```
$ su tria
```

INSTALL THE REQUIRED SOFTWARE PACKAGES

Use the commands below to install the required packages to your docker container or build machine. You'll be prompted to define your locales settings during the install phase.

```
$ sudo apt-get update
$ sudo apt-get install -y wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python python3 python3-pip \
python3-pexpect xz-utils debianutils iputils-ping python3-git \
python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm rsync curl gawk \
zstd lz4 locales bash-completion
```

Install the repo tool

```
$ mkdir -p ~/bin
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
```

Set your Git configuration

```
$ git config --global user.name "Your Name"  
$ git config --global user.email "yourEmail@example.com"
```

Setup the locale

```
$ sudo sed -i '/en_US.UTF-8/s/^# //g' /etc/locale.gen  
$ sudo locale-gen
```

FETCH SOURCE CODE

Download NXP meta layers

```
$ mkdir ~/imx-yocto-bsp  
$ cd ~/imx-yocto-bsp  
$ repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-scarthgap  
-m imx-6.6.52-2.2.0.xml  
$ repo sync
```

Download meta-maaxboard layer from Github.

```
$ cd ~/imx-yocto-bsp/sources  
$ git clone https://github.com/Avnet/meta-maaxboard.git -b scarthgap meta-  
maaxboard
```

BUILD THE IMAGES USING YOCTO

Set the Build Configuration

If you want to create a new build folder or set the configuration for the first time, run the command:

```
$ cd ~/imx-yocto-bsp  
$ MACHINE=maaxboard-osm93 source sources/meta-maaxboard/tools/maaxboard-  
setup.sh -b maaxboard-osm93/build
```

If you want to build in an existing build folder, use the following command:

```
$ cd ~/imx-yocto-bsp  
$ source sources/poky/oe-init-build-env maaxboard-osm93/build
```

Build the image

Execute the following command to build a Weston Wayland image:

```
$ bitbake avnet-image-full
```

After the build has successfully completed, the output files are deployed in:

```
~/imx-yocto-bsp/maaxboard-osm93/build/tmp/deploy/images/maaxboard-osm93/
```

imx-boot-tagged	Bootloader Image
avnet-image-full-maaxboard-osm93-xxxx.rootfs.wic	System image, this includes: Linux kernel, DTB and root file system.
Image	Kernel image
maaxboard-osm93.dtb	MaaXBoard OSM93 device tree binary
overlays	MaaXBoard OSM93 device tree overlay binary
avnet-image-full-maaxboard-osm93-xxxx.rootfs.tar.bz2	System image compressed archive file

TROUBLESHOOTING YOCTO BUILDS

Fetch Errors

When bitbake pulls source code or fetches source code it may run into issue for any number of reasons including network issues, server issues, etc.. Typically, restarting the build and trying the fetch operation again will clear this error. If not, examine the error text to determine what the issue may be.

NXP Documentation

NXP has documented the overall yocto build process and potential issues. Please follow [this link](#) for details.

Enable bitbake debug

Sometimes turning on debug may help identify what's causing yocto builds to fail, add the -D option.

```
$ bitbake -D avnet-image-full
```

Yocto Project Documentation

The Yocto Project documents build debugging tools and techniques. Please follow [this link](#) for details.

STANDALONE BUILD OF U-BOOT AND KERNEL

This section describes how to build the U-boot image and the Kernel using either the SDK or the ARM GCC compiler in a standalone environment.

Cross-compile tool chain

The cross-compile tool chain can be either ARM GCC or the Yocto SDK.

ARM GCC

Download the tool chain for the A-profile architecture from the [arm Developer GNU-A Downloads](#) page. It is recommended to use the 10.3 version for this release. Look for the "**gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu.tar.xz**" download. *Make sure you download the GNU/Linux target file.* Once you have the file downloaded onto your Linux build machine or Docker container follow the instructions below.

```
$ mkdir ~/toolchain
$ tar -xJf gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu.tar.xz -C
~/toolchain
```

Execute the following command to verify that the toolchain is correctly installed

```
$ cd toolchain/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu/bin/
$ ./aarch64-none-linux-gnu-gcc -v
```

To compile a project with ARM GCC, first set the environment with the following commands before building :

```
$ TOOLCHAIN_PATH=$HOME/toolchain/gcc-arm-10.3-2021.07-x86_64-aarch64-none-
linux-gnu/bin
$ export PATH=$TOOLCHAIN_PATH:$PATH
$ export ARCH=arm64
$ export CROSS_COMPILE=aarch64-none-linux-gnu-
```

You're now ready to build U-Boot or the Linux Kernel, please skip down to the section titled [Build U-Boot in a Standalone Environment](#).

Yocto SDK

To generate the SDK from the Yocto Project build environment use the following commands . . .

Note: You must complete the full yocto build before generating the SDK.

```
$ cd ~/imx-yocto-bsp
$ source sources/poky/oe-init-build-env maaxboard-osm93/build
$ bitbake avnet-image-full -c populate_sdk
```

After the build completes, the generated script file is located at: `~/imx-yocto-bsp/maaxboard-osm93/build/tmp/deploy/sdk/fsl-imx-xwayland-glibc-x86_64-avnet-image-full-armv8a-maaxboard-osm93-toolchain-6.6-scarthgap.sh`

Execute this script to install the SDK. The default install location is `/opt` but the SDK files can be placed anywhere on the host machine.

```
./fsl-imx-xwayland-glibc-x86_64-avnet-image-full-armv8a-maaxboard-osm93-
toolchain-6.12-styhead.sh
NXP i.MX Release Distro SDK installer version 6.6-scarthgap
=====
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/6.6-
scarthgap):
You are about to install the SDK to "/opt/fsl-imx-xwayland/6.6-scarthgap".
Proceed [Y/n]? Y
Extracting SDK.....done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

When using the SDK to compile a project, first execute the following command to configure environment variables :

```
$ . /opt/fsl-imx-xwayland/6.6-scarthgap/environment-setup-armv8a-poky-linux
```

Build U-Boot in a standalone environment

Install required packages

The U-Boot build requires additional packages

```
$ sudo apt-get update
$ sudo apt-get install -y bison flex
```

Get the source code and firmware

Pull the u-boot source code, uboot-imx, imx-atf and imx-mkimage. Execute the following commands

```
$ mkdir tmp
$ cd tmp
$ git clone https://github.com/Avnet/uboot-imx.git -b maaxboard_lf-6.6.52-2.2.0
$ git clone https://github.com/Avnet/imx-atf.git -b maaxboard_lf-6.6.52-2.2.0
$ git clone https://github.com/Avnet/imx-mkimage.git -b maaxboard_lf-6.6.52-2.2.0
```

Download the firmware-imx source, decompress the image and accept NXP EULA when running

```
$ wget https://www.nxp.com.cn/lgfiles/NMG/MAD/YOCTO/firmware-imx-8.26-d4c33ab.bin
$ chmod +x firmware-imx-8.26-d4c33ab.bin
$ ./firmware-imx-8.26-d4c33ab.bin
```

Execute the 'ls' command to view and verify the contents of the tmp directory

```
$ ls tmp
firmware-imx-8.26-d4c33ab  firmware-imx-8.26-d4c33ab.bin  imx-atf  imx-
mkimage  uboot-imx
```

At this point the required source code and firmware have been prepared.

Compile script

Create a bash script in the tmp directory, then change the file mode

```
$ cd tmp
$ touch make_mxosm93_uboot.sh
$ chmod 766 make_mxosm93_uboot.sh
$ vi make_mxosm93_uboot.sh
```

Copy the following content into the make_mxosm93_uboot.sh script and save the file

```
#!/bin/bash
PRJ_PATH=`pwd`
export JOBS=`cat /proc/cpuinfo | grep processor | wc -l`
export CROSS_COMPILE=$HOME/toolchain/gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-
gnu/bin/aarch64-none-linux-gnu-
export SRCS="imx-atf uboot-imx imx-mkimage"
MKIMG_BIN_PATH=$PRJ_PATH/imx-mkimage/iMX93
export FMW_PATH=firmware
set -e
function fetch_firmware()
{
    cd $PRJ_PATH
    mkdir -p $FMW_PATH && cd $FMW_PATH
```

```

if [ ! -d firmware-imx-8.26-d4c33ab ] ; then
    wget https://www.nxp.com.cn/lgfiles/NMG/MAD/YOCTO/firmware-imx-8.26-d4c33ab.bin
    bash firmware-imx-8.26-d4c33ab.bin --auto-accept > /dev/null 2>&1
fi

if [ ! -d firmware-ele-imx-1.3.0-17945fc ] ; then
    wget https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/firmware-ele-imx-1.3.0-17945fc.bin
    bash firmware-ele-imx-1.3.0-17945fc.bin --auto-accept > /dev/null 2>&1
fi

if [ ! -d firmware-upower-1.3.1 ] ; then
    wget https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/firmware-upower-1.3.1.bin
    bash firmware-upower-1.3.1.bin --auto-accept > /dev/null 2>&1
fi

#if [ ! -d meta-maaxboard ] ; then
#    git clone https://github.com/Avnet/meta-maaxboard.git -b nanbielid
#fi
rm -f *.bin
}

function build_atf()
{
    cd $PRJ_PATH
    SRC=imx-atf
    if [ ! -d $SRC ] ; then
        git clone https://github.com/Avnet/$SRC.git -b maaxboard_lf-6.6.52-2.2.0
    fi
    cd $SRC
    make -j${JOBS} CROSS_COMPILE=${CROSS_COMPILE} PLAT=imx93 bl31
    set -x
    cp build/imx93/release/bl31.bin $MKIMG_BIN_PATH
    set +x
    cd $PRJ_PATH
}

function build_uboot()
{
    SRC=uboot-imx

    if [ ! -d $SRC ] ; then
        git clone https://github.com/Avnet/$SRC.git -b maaxboard_lf-6.6.52-2.2.0
    fi

    cd $PRJ_PATH/$SRC

    if [ ! -f .config ] ; then
        make ARCH=arm maaxboard-osm93_defconfig
    fi

    make -j${JOBS} CROSS_COMPILE=${CROSS_COMPILE} ARCH=arm

    set -x
    cp u-boot.bin $MKIMG_BIN_PATH
    cp u-boot-nodtb.bin $MKIMG_BIN_PATH
    cp spl/u-boot-spl.bin $MKIMG_BIN_PATH
    cp arch/arm/dts/maaxboard-osm93.dtb $MKIMG_BIN_PATH/$IMXBOOT_DTB
    cp tools/mkimage $MKIMG_BIN_PATH/mkimage_uboot
    set +x
}

function build_imxboot()

```

```

{
    SRC=imx-mkimage

    #if [ ! -d $SRC ] ; then
    #   git clone https://github.com/Avnet/$SRC.git -b maaxboard_lf-6.6.52-2.2.0
    #fi

    cd $PRJ_PATH
    # copy firmware
    cp $FMW_PATH/firmware-imx-*/firmware/ddr/synopsys/lpddr4_[id]mem_[12]d*.bin
$MKIMG_BIN_PATH
    cp $FMW_PATH/firmware-ele-imx-*/mx93a1-ahab-container.img $MKIMG_BIN_PATH

    cd $PRJ_PATH/${SRC}
    # generate bootloader image
    make SOC=iMX93 REV=A1 flash_singleboot

    cp $MKIMG_BIN_PATH/flash.bin u-boot-maaxboard-osm93.imx
    chmod a+x u-boot-maaxboard-osm93.imx
    # copy bootloader image out
    cp u-boot-maaxboard-osm93.imx $PRJ_PATH
}
fetch_firmware
build_atf
build_uboot
build_imxboot

```

Execute the script to build the u-boot image

```

$ ./make_mxosm93_uboot.sh
$ ls -t
u-boot-maaxboard-osm93.imx  imx-mkimage  uboot-imx  imx-atf  firmware
make_mxosm93_uboot.sh  firmware-imx-8.26-d4c33ab  firmware-imx-8.26-
d4c33ab.bin

```

The boot image for Maaxboard OSM93 will be written to the current directory

Build Kernel in a standalone environment

Clone the Linux source code

```

$ git clone https://github.com/Avnet/linux-imx.git -b maaxboard_lf-6.6.52-
2.2.0

```

Check that the environment variables are correctly set

```

$ echo $CROSS_COMPILE $ARCH

```

Build the kernel sources

```

$ cd linux-imx
$ make distclean
$ make maaxboard-osm93_defconfig
$ make -j4

```

Execute the 'ls' command to view the Image and dtb files after compilation

```
$ ls arch/arm64/boot/Image
$ ls arch/arm64/boot/dts/freescale/maaxboard*.dtb
arch/arm64/boot/dts/freescale/maaxboard-osm93.dtb
```

Execute the following command to compile the kernel modules, and install the modules to rootfs in the current directory.

```
$ make modules
$ make modules_install INSTALL_MOD_PATH=./rootfs
```

SYSTEM POWER ON AND BOOT UP

To program the generated new Bootloader and System image files into MaaXBoard OSM93's eMMC memory, or for guidance on power-up MaaXBoard OSM93, the boot-up process, and how to exercise the supported BSP features of MaaXBoard OSM93, please refer to <http://avnet.me/maaxboard-osm93-yocto-user-manual>.

ADDING CUSTOM LAYERS TO THE YOCTO BUILD

To add new yocto recipes to the build . . .

1. Add the new meta-* layer to the sources directory
2. Add the new meta layer path into the `~/imx-yocto-bsp/sources/meta-maaxboard/bblayers.conf.sample.osm93` file
3. Add the application binary into the `~/imx-yocto-bsp/sources/meta-maaxboard/images/avnet-image-full.bb` file

As an example, we'll add the meta-aws layer to the build and include the aws-iot-device-client into the filesystem.

Download the new meta-layer

You can view a list of meta layers hosted by Open Embedded [here](#). For this exercise we'll clone the <https://github.com/aws4embeddedlinux/meta-aws> meta layer.

```
$ cd ~/imx-yocto-bsp/sources
$ git clone https://github.com/aws4embeddedlinux/meta-aws.git
```

Pull the correct meta-layer branch to match your build

```
$ cd meta-aws
$ git checkout styhead
```

Add the new meta-* layer to the build

- Edit the `~/imx-yocto-bsp/sources/meta-maaxboard/bblayers.conf.sample.osm93` file
- Locate the **i.MX Yocto Project Release layers** section, then add a line at end of the section to include your new meta-layer into the build

```
# i.MX Yocto Project Release layers
BBLAYERS += "${BSPDIR}/sources/meta-imx/meta-imx-bsp"
BBLAYERS += "${BSPDIR}/sources/meta-imx/meta-imx-sdk"
BBLAYERS += "${BSPDIR}/sources/meta-imx/meta-imx-m1"
BBLAYERS += "${BSPDIR}/sources/meta-imx/meta-imx-v2x"
BBLAYERS += "${BSPDIR}/sources/meta-nxp-demo-experience"

BBLAYERS += "${BSPDIR}/sources/meta-arm/meta-arm"
BBLAYERS += "${BSPDIR}/sources/meta-arm/meta-arm-toolchain"
BBLAYERS += "${BSPDIR}/sources/meta-browser/meta-chromium"
BBLAYERS += "${BSPDIR}/sources/meta-clang"
BBLAYERS += "${BSPDIR}/sources/meta-openembedded/meta-gnome"
BBLAYERS += "${BSPDIR}/sources/meta-openembedded/meta-networking"
BBLAYERS += "${BSPDIR}/sources/meta-openembedded/meta-fileystems"
BBLAYERS += "${BSPDIR}/sources/meta-qt6"
BBLAYERS += "${BSPDIR}/sources/meta-security/meta-openssl"
BBLAYERS += "${BSPDIR}/sources/meta-security/meta-tpm"
BBLAYERS += "${BSPDIR}/sources/meta-virtualization"
BBLAYERS += "${BSPDIR}/sources/meta-aws"
```

Include the application into the image

- Edit the `~/imx-yocto-bsp/sources/meta-maaxboard/images/avnet-image-full.bb` file
- Add an `aws-iot-device-client` line to the `CORE_IMAGE_EXTRA_INSTALL:append` statement

```
CORE_IMAGE_EXTRA_INSTALL:append = " \
    ${EXTRA_GCC_TOOL} \
    tzdata vim tree \
    gnupg \
    parted \
    i1 \
    "

    freerap \
    nodejs \
    nodejs-npm \
    python3 \
    python3-pip \
    aws-iot-device-client \
    "
```

Rebuild the image

Note: You must run the full `MACHINE=maaxboard-osm93 source sources/meta-maaxboard/tools/maaxboard-setup.sh -b maaxboard-osm93/build` command before the edited configuration files will be used

```
$ cd ~/imx-yocto-bsp
$ MACHINE=maaxboard-osm93 source sources/meta-maaxboard/tools/maaxboard-setup.sh -b maaxboard-osm93/build
$bitbake avnet-image-fill
```

APPENDIX

Hardware Documents

For the detail hardware introduction, please refer to *MaaXBoard OSM93 Hardware user manual*.

Software Documents

MaaXBoard-OSM93 supports Yocto Linux, for additional info, refer to the following documents:

- ***MaaXBoard OSM93 Linux Yocto User Manual***
Describes how to boot up MaaXBoard OSM93 and aspects of the BSP functionality
- ***MaaXBoard OSM93 Linux Yocto Development Guide***
Detailed guidance on how to rebuild the Linux system image (This document)

Contact Information

Product Webpage: <https://avnet.me/maaxboard-osm93>