1. System Requirements

BG Network's Embedded Security Software Architecture (ESSA) is a Yocto-based solution. The build has been tested with Ubuntu 20.04 (LTS). For other supported Linux distributions, please refer to the Yocto Project Reference Manual.

The ESSA is only tested with the WINSYSTEM ITX-P-C444 board (imx8mq-itx-p-c444).

Essentials: The packages that are needed to build an image on a headless system:

```
$ sudo apt install gawk wget git diffstat unzip texinfo gcc build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm python3-subunit mesa-common-dev zstd liblz4-tool snapd minicom
```

```
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/repo
```

```
$ chmod a+x  ~/repo
```

```
$ sudo cp ~/repo /usr/bin
```

2. Core Image build

(1) Create a directory for the ESSA BSP for the ITX-P-C444 board:

```
$ mkdir ~/bgn-essa-c444
```

```
$ cd ~/bgn-essa-c444
```

(2) repository initialization

```
$repo init -u https://github.com/bgnetworks/c444-manifest.git -b master -m itx-p-c444_5.4.47.xml
```

```
$wget --directory-prefix .repo/manifests https://raw.githubusercontent.com/bgnetworks/meta-bgn-essa/zeus-w-caam/meta-mender-c444/scripts/c444_5.4.47-essa-demo.xml
```

```
$repo init -m c444_5.4.47-essa-demo.xml
```

(3) pull repositories

```
$repo sync
```

The files and directories that are in the bgn-essa-c444

```
README-IMXBSP  setup-environment  sources  winsys-setup-release.sh
```

The Yocto meta directories that are in the sources

```
base            meta-freescale-3rdparty  meta-openembedded  meta-security
meta-bgn-essa   meta-freescale-distro    meta-python2       meta-timesys
meta-browser    meta-imx                 meta-qt5           meta-winsys
meta-freescale  meta-mender              meta-rust          poky
```

(4) setup the build environment at first time

```
$ MACHINE=imx8mq-itx-p-c444 DISTRO=c444-xwayland source c444-setup-essa.sh -b build
```

(5) build core image

```
$ bitbake core-image-base
```

The following build configurations should show up in the terminal:

```
Build Configuration:
BB_VERSION           = "1.44.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "ubuntu-20.04"
TARGET_SYS           = "aarch64-poky-linux"
MACHINE              = "imx8mq-itx-p-c444"
DISTRO               = "c444-xwayland"
DISTRO_VERSION       = "5.4-zeus"
TUNE_FEATURES        = "aarch64 cortexa53 crc crypto"
TARGET_FPU           = ""
meta
meta-poky            = "HEAD:cba967414370e195d109353e51510bd829aa86c3"
meta-oe
meta-multimedia
meta-python          = "HEAD:9e60d30669a2ad0598e9abf0cd15ee06b523986b"
meta-freescale       = "HEAD:c2a0d924f6200eea1fb1d1b61e7420ea5da2f8fb"
meta-freescale-3rdparty = "HEAD:dbcc686f52c3c84db8cb86aa8973a4e373651b98"
meta-freescale-distro = "HEAD:ca27d12e4964d1336e662bcc60184bbff526c857"
meta-bsp
meta-sdk
meta-ml              = "HEAD:a083ffbbc3f4d1c02b1542746784d7f641a75b60"
meta-browser         = "HEAD:5f365ef0f842ba4651efe88787cf9c63bc8b6cb3"
meta-rust            = "HEAD:a012a1027defe28495f06ed522a7a82bdd59a610"
meta-gnome
meta-perl
meta-networking
meta-filesystems     = "HEAD:9e60d30669a2ad0598e9abf0cd15ee06b523986b"
meta-qt5             = "HEAD:b8bcf8cb576d072f435a0177375e54424f67d1c9"
meta-python2         = "HEAD:231c3d74cfcf734c3415e86ea8dd97f73ddced9d"
meta-tpm             = "HEAD:440c37f0b623ccc0aa0328613908608d6362adda"
meta-winsys          = "HEAD:4e8679406d5c7fdd22c4e7dd126bb1db5a4fc2d3"
meta-mender-core     = "HEAD:f7e9e8bb985b4df62b703ed94559b0c23ec683b2"
meta-mender-c444
meta-essa-c444       = "HEAD:5dfcf4532191999b02ff857ae3d0c4ef44d4cbf5"
```

Note: the image build may take a few hours at the first time

3. Program the images into the board EMMC

(1) install uuu tool

```
$sudo snap install universal-update-utility
```

(2) change to the image directory

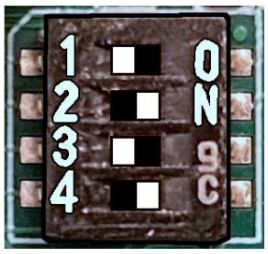
```
$ cd ~/bgn-essa-c444/build/tmp/deploy/images/imx8mq-itx-p-c444
```

```
compile.log                                                  imx-boot
core-image-base.env                                          imx-boot-imx8mq-itx-p-c444-sd.bin-flash_dp_evk
core-image-base-imx8mq-itx-p-c444-20211031234755.dataimg     imx-boot-imx8mq-itx-p-c444-sd.bin-flash_evk
core-image-base-imx8mq-itx-p-c444-20211031234755.ext4        imx-boot-imx8mq-itx-p-c444-sd.bin-flash_evk_no_hdmi
core-image-base-imx8mq-itx-p-c444-20211031234755.manifest    imx-boot-imx8mq-itx-p-c444-sd.bin-print_fit_hab
core-image-base-imx8mq-itx-p-c444-20211031234755.mender      imx-boot-tools
core-image-base-imx8mq-itx-p-c444-20211031234755.sdcard      lpddr4_pmu_train_1d_dmem.bin
core-image-base-imx8mq-itx-p-c444-20211031234755.sdimg       lpddr4_pmu_train_1d_imem.bin
core-image-base-imx8mq-itx-p-c444-20211031234755.sdimg.bz2   lpddr4_pmu_train_2d_dmem.bin
core-image-base-imx8mq-itx-p-c444-20211031234755.testdata.json lpddr4_pmu_train_2d_imem.bin
core-image-base-imx8mq-itx-p-c444-20211031234755.wic         modules--5.4-r0-imx8mq-itx-p-c444-20211029001315.tgz
core-image-base-imx8mq-itx-p-c444.dataimg                    modules-imx8mq-itx-p-c444.tgz
core-image-base-imx8mq-itx-p-c444.ext4                       signed_dp_imx8m.bin
core-image-base-imx8mq-itx-p-c444.manifest                   signed_hdmi_imx8m.bin
core-image-base-imx8mq-itx-p-c444.mender                     tee.bin
core-image-base-imx8mq-itx-p-c444.sdcard                     tee.mx8mqevk.bin
core-image-base-imx8mq-itx-p-c444.sdimg                      u-boot.bin
core-image-base-imx8mq-itx-p-c444.sdimg.bz2                  u-boot.bin-sd
core-image-base-imx8mq-itx-p-c444.testdata.json              uboot.env
core-image-base-imx8mq-itx-p-c444.wic                        u-boot-imx8mq-itx-p-c444.bin
core-image-base-imx-imx-boot-bootpart.wks                    u-boot-imx8mq-itx-p-c444.bin-sd
Image                                                        u-boot-sd-2020.04-r0.bin
Image--5.4-r0-imx8mq-itx-p-c444-20211029001315.bin           u-boot-spl.bin
Image-imx8mq-itx-p-c444.bin                                  u-boot-spl.bin-imx8mq-itx-p-c444
imx8mq-itx-p-c444--5.4-r0-imx8mq-itx-p-c444-20211029001315.dtb u-boot-spl.bin-imx8mq-itx-p-c444-2020.04-r0-sd-2020.04-r0
imx8mq-itx-p-c444.dtb                                        u-boot-spl.bin-imx8mq-itx-p-c444-sd
imx8mq-itx-p-c444-imx8mq-itx-p-c444.dtb                      u-boot-spl.bin-sd
```

(3) set the itx-p-c444 board to serial download mode

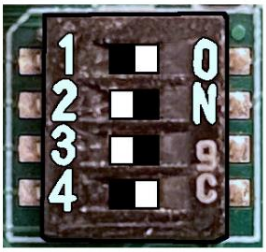
Serial Downloader Mode

1 = 0
2 = 1
3 = Do not care
4 = Do not care

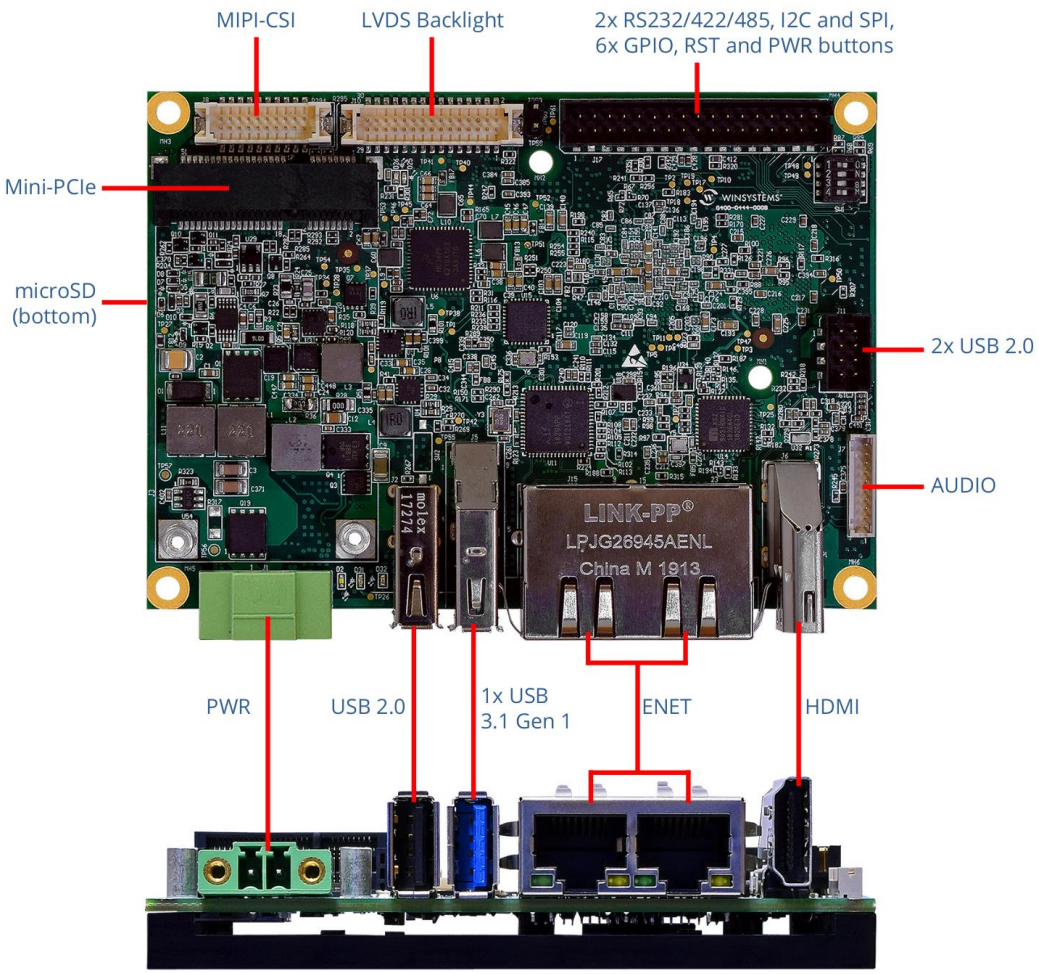


Boot from eMMC

1 = 1
2 = 0
3 = 0
4 = 1



(4) Connect the board (USB 3.1 Gen 1 connector) to the build machine with a USB type A to type A cable



(5) Power up the board

(6) Confirm the board has been set up correctly

```
$uuu -lsusb
```

```
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.4.138-0-g051a8fe

Connected Known USB Devices
        Path     Chip     Pro      Vid      Pid      BcdVersion
        ==================================================
        1:5      MX8MQ    SDP:     0x1FC9 0x012B   0x0001
```

(7) burn the uboot and image

```
$ sudo uuu -b emmc_all imx-boot core-image-base-imx8mq-itx-p-c444.sdimg
```

```
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.4.138-0-g051a8fe

Success 1    Failure 0


1:5      8/ 8 [Done                                  ] FB: done
```

(8) power down the board

(9) set the itx-p-c444 board to the emmc boot mode

4. Block encryption example

(1) set up minicom

```
$ sudo minicom -s
```

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
```

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

(2) open minicom

```
$ sudo minicom
```

(3) connect to the itx-p-c444 board J3 COM PORT 1 with a USB-UART cable

(4) power up the itx-p-c444 board and log in as root after booting is completed

(5) create a random key

```
$ caam-keygen create mykey ecb -s 16
```

```
$ cd /data/caam
```

```
root@imx8mq-itx-p-c444:/data/caam# ls
mykey  mykey.bb
```

(6) add the key into the Linux keyring

```
$ cat mykey | keyctl padd logon mykey1: @s
```

```
root@imx8mq-itx-p-c444:/data/caam# keyctl list @s
1 key in keyring:
361817030: --alsw-v     0     0 logon: mykey1:
```

(7) Create a file and link to loop device

```
$ dd if=/dev/zero of=encrypted.img bs=1M count=32
```

```
$ losetup /dev/loop0 encrypted.img
```

(8) use the generated random key for block encryption

```
$ dmsetup -v create myEncryptedBlock --table "0 $(blockdev --getsz /dev/loop0) crypt capi:tk(cbc(aes))-plain
:36:logon:mykey1: 0 /dev/loop0 0 1 sector_size:512"
```

```
root@imx8mq-itx-p-c444:/data/caam# dmsetup table --showkey myEncryptedBlock
0 52336 crypt capi:tk(cbc(aes))-plain :36:logon:mykey1: 0 7:0 0
```

(9) build file system

```
$ mkfs.ext4 /dev/mapper/myEncryptedBlock
```

(10) mount the encrypted block

```
$ mkdir -p /mnt/myBlock
```

```
$ mount /dev/mapper/myEncryptedBlock /mnt/myBlock
```

(11) create a new file in the encrypted block

```
$ echo "This is a test of disk encryption on i.MX" > /mnt/myBlock/readme.txt
```

(12) umount and remove the encrypted block

```
$ umount /mnt/myBlock
```

```
$ dmsetup remove myEncryptedBlock
```

(13) power cycle the itx-p-c444 board and log in as root after the booting is completed

(14) import the generated random key

```
$ cd /data/caam
```

```
$ caam-keygen import mykey.bb importKey
```

```
root@imx8mq-itx-p-c444:/data/caam# ls
encrypted.img  importKey  mykey.bb
```

(15) add the key into the Linux keyring

```
$ cat importKey | keyctl padd logon mykey2: @s
```

```
root@imx8mq-itx-p-c444:/data/caam# keyctl list @s
2 keys in keyring:
786093632: ----s-rv     0     0 user: invocation_id
910674659: --alsw-v     0     0 logon: mykey2:
```

(16) link the file to loop device

```
$ losetup /dev/loop0 encrypted.img
```

(17) use the imported key for block encryption

```
$ dmsetup -v create myEncryptedBlock --table "0 $(blockdev --getsz /dev/loop0) crypt capi:tk(cbc(aes))-plain
:36:logon:mykey2: 0 /dev/loop0 0 1 sector_size:512"
```

(18) mount the encrypted block

```
$ mount /dev/mapper/myEncryptedBlock /mnt/myBlock
```

(19) check the readme

```
root@imx8mq-itx-p-c444:/data/caam# cat /mnt/myBlock/readme.txt
This is a test of disk encryption on i.MX
```