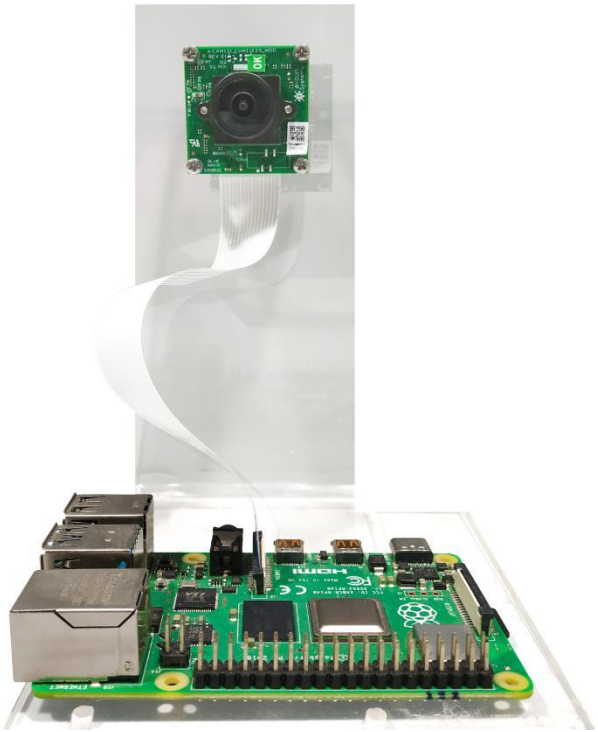
e-CAM20_CURB

Developer Guide



Version 1.4

e-con Systems

2/8/2022

Disclaimer

e-con Systems reserves the right to edit/modify this document without any prior intimation of whatsoever.

Contents

INTRODUCTION TO e-CAM20_CURB:

e-con Systems is a leading Embedded Product Design Services Company which specializes in advanced camera solutions. e-CAM20_CURB is a new MIPI camera which uses the AR0234 camera module. It is a 2-Lane module connected to the Raspberry Pi 4 development kit launched by e-con Systems. The prebuilt driver for this camera along with the camera board is provided by e-con Systems.

The Raspberry Pi 4 development kit is a small, powerful computer for embedded applications, Artificial Intelligence (AI) and Internet of Things (IoT).

e-CAM20_CURB has a 2.3 MP color camera with S-mount (also known as M12 board lens) lens holder. The S-mount is one of the most used small form factor lens mounts for board cameras. e-CAM20_CURB camera contains 1/2.6" AR0234 CMOS image sensor from On Semiconductor® and is interfaced to the J3 camera connector of the Raspberry Pi 4 development kit using the ACC-XVRNX-MIPICAMERA-ADP board.

This document explains how to setup the Raspberry Pi 4 development kit for using e-CAM20_CURB

Prerequisites:

This section describes the requirements to use e-CAM20_CURB on the Raspberry Pi 4 development kit.

The prerequisites are as follows:

- Host PC which runs Ubuntu 16.04/18.04 (64-bit).
- SD card of minimum 16GB size
- SD card reader
- UART to USB convertor
- Raspberry Pi 4 development kit
- Power adapter compatible with Raspberry Pi 4 with output 5V 3A
- Micro HDMI to HDMI cable
- e-CAM20_CURB camera module (e-CAM217_CUMI0234_MOD + ACC-XVRNX-MIPICAMERA-ADP + FIFO 03028B8M Lens + 686615152001 15CM FFC Cable)

This section describes the steps for building and installing the kernel.

Setting Up the Environment for Raspian OS:

we have tested our product with Raspian Buster kernel versions 5.10.x

The steps to setup the environment are as follows:

1. Download the Raspian OS with desktop and recommended software (32 bit) from the following link,

   https://www.raspberrypi.com/software/operating-systems/

2. Run the following command to install cross compiler toolchain and dependencies, for building the kernel.

   ```
   sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev
   ```

   Note: Assuming SD card is ready with pre-flashed RaspianOS buster

3. Run the following command to install 32 bit toolchain.
   ```
   sudo apt install crossbuild-essential-armhf
   ```

Downloading Source code and Applying the Patch

1. You can download either full kernel source code with all branch history or download only required branch for development.

2. To download Full kernel source (Will consume more space and time):

```
git clone  https://github.com/raspberrypi/linux
```

To download only particular branch:

```
git clone --depth=1 --branch rpi-5.10.y https://github.com/raspberrypi/linux
```

Building from Source (RaspianOS):

1. Apply patch provided by econ
Change the directory

```
cd linux
```

2. Copy the patch to current directory

```
cp  <patch location>/e-CAM20_CURB_RaspianOS_kernel_<ver>.patch   ./
```

3. Apply the patch

```
Patch -p1 -i  e-CAM20_CURB_RaspianOS_kernel_<ver>.patch
```

Note:

Please ensure you are applying the correct version of patch with your kernel version .

Currently we tested kernel 5.10 with RaspianOS

4. Run the following commands to setup the required environment variables from kernel top directory.

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
```

```
make  bcm2711_defconfig
```

Note:

For RaspberryPi4 config name is bcm2711_defconfig . for other variants of Raspbarrypi , we need to use config name appropriately. Please refer https://www.raspberrypi.com/documentation/computers/linux_kernel.html#kernel for more details .

5. Run the following command to build kernel image, dtb and driver.

```
make zImage modules dtbs -j<n>
```

Note: Use -j<n> depending upon the number of parallel executions possible in your build PC. It will fasten the build process.

Flashing Raspberry Pi 4 Development Kit

The steps to flash the Raspberry Pi 4 development kit are as follows:

1. Connect the SD card pre-flashed with RaspianOS

Note: /*dev*/sdx1 (boot) and /dev/sdx2 (rootfs) will be detected.

2. Mount the SD card partitions

```
mkdir -p mnt/boot
```

```
mkdir -p mnt/rootfs
```

```
sudo mount /dev/sdx1 mnt/boot/
```

```
sudo mount /dev/sdx2 mnt/rootfs/
```

3. Run the following commands to copy the images to SD card locations.

Copy the kernel image file

```
sudo cp arch/arm/boot/zImage  mnt/boot/kernel7l.img
```

Note:

For RaspberryPi4, kernel name is kernel7l.img . for other variants of Raspbarrypi , we need to use kernel name appropriately. Please refer

https://www.raspberrypi.com/documentation/computers/linux_kernel.html#kernel for more details.

Copy the device tree files

```
cp arch/arm/boot/dts/*.dtb   mnt/boot/
```

```
cp arch/arm/boot/dts/overlays/*.dtb*  mnt/boot/overlays/
```

Note:

Econ module overlay will be generated under arch/arm/boot/dts/overlays/ar0234_mcu.dtbo, which will be copied automatically to sd card with above command

4. Install the modules:

```
sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
 INSTALL_MOD_PATH=mnt/rootfs/ modules_install
```

Note:

Econ module driver will be generated under drivers/media/i2c/ar0234_mcu.ko , which will be copied automatically to sd card with above command

Modifying Configuration Files in Boot Partition

1. Open with editor

```
vim mnt/boot/config.txt
```

2. Add the following line in the config file

```
dtoverlay=ar0234_mcu
```

Testing the camera:

After the board booted , test the following commands to check whether camera detected successfully . If not , please recheck the above steps carefully . To ensure hardware connections and functionality, we strongly recommend to test the camera module first with prebuild binary provided.

```
ls /dev/video0
```

Once above node is detected , check the stream using gstreamer pipeline provided in getting started manual.

Building from Source (Yocto):

All the below mentioned source files will be part of patch provided in the release package. Below steps are provided in case user needs to modify existing driver,dtb,kernel,application (or) if user wants to port the files to other branches of yocto.

It is mandatory that user needs some basic knowledge of yocto. If not please go through below open source contents from yocto project.

https://www.yoctoproject.org/docs/1.6.1/dev-manual/dev-manual.html

Setting Up the Environment for Yocto:

1. Install basic packages for yocto build environment

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev xterm
```

2. clone dunfell version of yocto build system

```
git clone -b dunfell git://git.yoctoproject.org/poky.git poky-dunfell
cd poky-dunfell
git clone -b dunfell https://github.com/meta-qt5/meta-qt5
git clone -b dunfell https://git.yoctoproject.org/git/meta-raspberrypi
git clone -b dunfell https://git.openembedded.org/meta-openembedded
```

3. source the environmental variables ,setup build directory and build

```
source oe-init-build-env build_rpi4   (or)  source oe-init-build-env build_rpi3
bitbake core-image-sato
```

4. copy econ camera supported meta-econsys provided in the release package to the top directory of Yocto build

```
cp  <release package directory>/yocto_files/meta-econsys  ../
```

5. All the econ modified files in yocto build system and patches are provided for reference in the following directory of release package

```
ls   <release package directory>/yocto_files/econ_patches_and_modified_files_ref
```

6. Add the meta-econsys and other cloned layers in conf/bblayers.conf file like below

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"
BBPATH = "${TOPDIR}"
BBFILES ?= ""
BBLAYERS ?= " \
 ${TOPDIR}/../meta \
 ${TOPDIR}/../meta-poky \
 ${TOPDIR}/../meta-yocto-bsp \
 ${TOPDIR}/../meta-qt5 \
 ${TOPDIR}/../meta-raspberrypi \
 ${TOPDIR}/../meta-openembedded/meta-oe \
 ${TOPDIR}/../meta-econsys \
 "
```

7. Add/modify following contents in conf/local.conf file

```
MACHINE ??= "raspberrypi4"

CORE_IMAGE_EXTRA_INSTALL += "kernel-modules v4l-utils i2c-tools userland ffmpeg
gstreamer1.0 gstcapture gstreamer1.0-omx omxplayer lrzsz copytofilesystem ar0234"

KERNEL_IMAGETYPE = "uImage"

RPI_USE_U_BOOT = "1"

IMAGE_FSTYPES = "tar.bz2 ext4 rpi-sdimg"

ENABLE_UART = "1"

LICENSE_FLAGS_WHITELIST = "commercial license"

IMAGE_ROOTFS_SIZE ="4194304"

#4GB in Kbytes

VIDEO_CAMERA = "1"

RASPBERRYPI_CAMERA_V2 = "1"

HDMI_FORCE_HOTPLUG= "1"

#DISABLE_VC4GRAPHICS= "1"

BB_NUMBER_THREADS ?= "8"

PARALLEL_MAKE ?= "-j 8"
```

Building and Installing module:

1. Driver files are added and recipes written under meta-econsys layer

```
meta-econsys/recipes-modules_development/ar0234
```

2. Build command to compile only module

```
bitbake ar0234
```

Note:

After complete sato image build , module will be automatically copied to the following directory in target root file system

```
/lib/modules/5.4.72-v7l/extra/ar0234.ko
```

Building , Installing DTB and kernel changes:

Kernel level modifications needs to be included as patch . Otherwise any modifications done directly in kernel source under **build_rpi4/tmp/work-shared/raspberrypi4/kernel-source** directories will be lost during rebuilds

1. Include the kernel and device tree patch files from in release package to the following location.

```
meta-raspberrypi/recipes-kernel/linux/files/
```

2. Add the patch file names in the file

```
meta-raspberrypi/recipes-kernel/linux/linux-raspberrypi_5.4.bb
```

Example:

```
SRC_URI += "
        < other default patches in the bb file  > \
         file://0001-DTB-Support-for-econ-camera.patch \
         file://kernel_frame_sync_support.patch \
         file://econ_config.cfg \
        "
```

3. Include DTB in Yocto BUILD

DTB needs to be included in ./meta-raspberrypi/conf/machine/include/rpi-base.inc

```
RPI_KERNEL_DEVICETREE_OVERLAYS ?= " \
overlays/ar0234_mcu.dtbo \
```

4. Modify following file to add DTB name to config.txt file in boot partition of SD card , otherwise module wont get loaded during boot

FILE:

./meta-raspberrypi/recipes-bsp/bootfiles/rpi-config_git.bb

Example:

```
echo "dtoverlay=ar0234_mcu" >> {DEPLOYDIR}/${BOOTFILES_DIR_NAME}/config.txt
```

Note: Add this line as a last line of do_deploy() function.

5. Command to Build kernel and device tree only.

```
bitbake linux-raspberrypi
```

6. Output files will be generated at **tmp/deploy/images/raspberrypi4/**

Building complete yocto:

1. Command to run the complete build after modification

```
bitbake core-image-sato
```

2. The command to enter output directory of the above build.

```
cd tmp/deploy/images/raspberrypi4/
```

3. Flash SD card using the following command

```
sudo dd if=core-image-sato-raspberrypi4.rpi-sdimg of=/dev/sdX bs=4M status=progress
```

Note:

Above step should be done carefully . SD card detected partition should be replaced properly . for example: of=/dev/sdc . if wrong partition used, you may corrupt your development PC.

6. Boot the board from SD card (access through UART interface)

username: root

In this section, you can view the commonly occurring issue and their troubleshooting step.

1. **I have flashed Raspberry Pi 4 development kit already. What are the steps to install the binaries?**

   Refer to the *e-CAM20_CURB_Getting_Started_Manual_<REV>.pdf* to upgrade the latest binaries.

2. **Why my camera is not detected?**

   Please make sure you connect the FPC cables as mentioned in *CAM20_CURB_Getting_Started_Manual_<REV>.pdf*.

DTB: Device Tree Blob.

Micro SD: micro Secure Digital.

GNU: GNU's Not Unix.

MIPI: Mobile Industry Processor Interface.

OS: Operating Systems.

Rootfs: Root Filesystems.

UART: Universal Asynchronous Receiver/Transmitter**.**

USB: Universal Serial Bus.

YOCTO: Linux kernel based OS build system

Contact Us

If you need any support on e-CAM20_CURB product, please contact us using the Live Chat option available on our website - https://www.e-consystems.com/

Creating a Ticket

If you need to create a ticket for any type of issue, please visit the ticketing page on our website - https://www.e-consystems.com/create-ticket.asp

RMA

To know about our Return Material Authorization (RMA) policy, please visit the RMA Policy page on our website - https://www.e-consystems.com/RMA-Policy.asp

General Product Warranty Terms

To know about our General Product Warranty Terms, please visit the General Warranty Terms page on our website - https://www.e-consystems.com/warranty.asp

Revision History

Rev	Date	Description	Author
1.0	15-April-2021	Initial draft	Product marketing team
1.1	06-May-2021	RaspberryPi 3 support added	Product marketing team
1.2	29-NOV-2021	Building from source added	Product marketing team
1.4	10-FEB-2022	Raspian Support added	Product marketing team