

e-CAM130_CURB

Developer Guide



Version 1.1

e-con Systems

11/29/2021





Disclaimer

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



Contents

INTRODUCTION TO E-CAM130_CURB	4
PREREQUISITES FOR RASPBERRY PI 4	5
INSTALLATION PROCEDURE FOR RASPBERRY PI 4	6
BUILDING FROM SOURCE	9
FAQ	11
GLOSSARY	12
SUPPORT	<u>13</u>



INTRODUCTION TO e-CAM130_CURB

e-con Systems is a leading Embedded Product Design Services Company which specializes in advanced camera solutions. e-CAM130_CURB is a new MIPI camera which uses the AR1335 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-CAM130_CURB has a 13 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-CAM130_CURB camera contains 1/3.2" AR1335 CMOS image sensor from On Semiconductor® and is interfaced to the J3 camera connector of the Raspberry Pi 4 development kit using the ACC_NANO_ADP board.

This document explains how to setup the Raspberry Pi 4 development kit for using e-CAM130_CURB.

Software Requirements

The software requirements are as follows:

- Cross compiler toolchain
- Pre-build raspian OS
- Yocto Build setup



Prerequisites for Raspberry Pi 4

This section describes the requirements to use e-CAM130_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 8GB size
- SD card reader
- UART to USB converter
- Raspberry Pi 4 development kit
- Power adapter compatible with Raspberry Pi 4 with output 5V 3A
- Micro HDMI to HDMI cable
- e-CAM130_CURB camera module (e-CAM137A_CUMI1335_MOD + ACC_NANO_ADP + FIFO 03028B8M Lens + 686715152001 15CM FFC Cable)



Installation Procedure for Raspberry PI 4

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

Setting Up the Host PC Environment for Raspian OS:

we have tested our product with Raspian Buster kernel versions 5.4.x and 5.10.x

The steps to setup the environment are as follows:

1. 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

2. 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

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-5.4.y https://github.com/raspberrypi/linux
```



Building from Source (RaspbianOS):

1. Apply patch provided by econ

Change the directory

```
Cd linux
```

Copy the patch to current directory

```
Cp <patch location>/e-CAM130_CURB_RaspbianOS_kernel_<ver>.patch ./
```

Apply the patch

```
Patch -p1 -i e-CAM130_CURB_RaspbianOS_kernel_<ver>.patch
```

Note:

Please ensure you are applying the correct version of patch with your kernel version . Currently we tested kernel 5.4 and 5.10 with RaspbianOS

2. 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 Raspberrypi , we need to use config name appropriately. Please refer https://www.raspberrypi.com/documentation/computers/linux_kernel.html#kernel for more details .



3. 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 RaspbianOS

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 Raspberrypi , 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/ar1335_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-gnueabi-hf-  
INSTALL_MOD_PATH=mnt/rootfs/ modules_install
```

Note:

Econ module driver will be generated under drivers/media/i2c/ar1335_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=ar1335_mcu
```

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
```

3. apply econ patch provided in the release package for ecam130_curb support

```
Cd poky-dunfell  
patch -p1 -i ./e-CAM130_CURB_Release_1.0.patch
```



4. build the images for raspberrypi 3 and raspberrypi 4 as required

```
source oe-init-build-env build_rpi4 (or) source oe-init-build-env  
build_rpi3
```

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

5. flash the images to sd card

```
cd tmp/deploy/images/raspberrypi4 (or) cd  
tmp/deploy/images/raspberrypi3
```

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

(or)

```
sudo dd if=core-image-sato-raspberrypi3.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
```

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>



Building and Installing module:

1. Driver files can be added or modified at meta-econsys layer

```
meta-econsys/recipes-modules_development/<module_name>
```

Note:

A) Make sure Makefile and bb files are modified properly and also required files are copied to the "files" directory (*meta-econsys/recipes-modules_development/<module_name>/files/*)

B) Make sure module is added in **CORE_IMAGE_EXTRA_INSTALL** variable in *meta/recipes-sato/images/core-image-sato.bb* like below

```
CORE_IMAGE_EXTRA_INSTALL += "v4l-utils i2c-tools  
<module_name>"
```

2. Build command to compile only module

```
bitbake <module_name>
```

Note:

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

```
/lib/modules/5.4.72-v7l/extra/<module_name>.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

- A) Including kernel and DTB patches

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



raspberrypi_<version>.bb

1 .Creating DTB or kernel patch

- i. build kernel first using below command. yocto will automatically fetch source, untar it , apply patches , and finally produce output images
- ii. **bitbake linux-raspberrypi**
- iii. copy the kernel source from **build_rpi4/tmp/work-shared/raspberrypi4/kernel-source/** and maintain a separate development source outside yocto for creating DTB and kernel level patches alone
- iv. initialize a local git where you copied kernel source
- v. modify DTB or kernel level code
- vi. create patch
- vii. copy the patches to **linux-raspberrypi_<version>.bb** file directory as shown above
- viii. patches will be automatically applied to yocto kernel during build.
- ix. Output files will be generated at **tmp/deploy/images/raspberrypi4/**

2.Include DTB in Yocto BUILD

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

**RPI_KERNEL_DEVICETREE_OVERLAYS ?= " **

**overlays/<dtb_name>.dtbo **

3. Add our DTB to config.txt file of SD card boot partition

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



FILE:

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

Example:

```
echo "dtoverlay=<DTB_name>" >>  
{DEPLOYDIR}/{BOOTFILES_DIR_NAME}/config.txt
```

Building and Installing Application:

1. Modify gstcapture application provided by econ under
`meta-econsys/recipes-multimedia/gstcapture/`
2. Building the application
`bitbake gstcapture`

Note:

make sure application is added in CORE_IMAGE_EXTRA_INSTALL variable in core-image-sato.bb file to get installed in filesystem

3. Run the complete build

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



Troubleshooting

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

After flashing, the Raspberry Pi 4 board is not booting, or the display is blank. How to solve this issue?

To solve this issue, please follow these steps:

- Use the correct command with sudo permission whenever needed.
- Use the PC with Ubuntu 18.04/16.04 64-bit for flashing.
- Maintain enough free space in hard disk before flashing.



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-CAM130_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 *CAM130_CURB_Getting_Started_Manual_<REV>.pdf*.



After understanding how to setup the Raspberry Pi 4 development kit using e-CAM130_CURB MIPI camera, you can refer to the following documents to understand more about e-CAM130_CURB.

- *e-CAM130_CURB Release Notes*
- *e-CAM130_CURB Release Package Manifest*



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.



Contact Us

If you need any support on e-CAM130_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	11-April-2020	Initial draft	Product Marketing Team
1.1	24-MAR-2021	Yocto support added	Product Marketing Team
1.2	29-NOV-2021	Building from source added	Product Marketing Team
1.3	3-DEC-2021	RaspbianOS support with kernel 5.10 added	Product Marketing Team

