USER MANUAL

RK3399-Q7 System-on-Module

Hexa-Core ARM Cortex-A72/A53

featuring the Rockchip RK3399 application processor

Document revision: Release v1.4
Issue date: Apr 11, 2018

Contents

1 Introduction

Congratulations for acquiring our new flagship product, combining best-in-class performance with a rich set of peripherals.

Note: The latest version of this manual and related resources can always be found on our website at the following address:

https://www.theobroma-systems.com/rk3399-q7/

1.1 Device Overview

The RK3399 is a low power, high performance processor for computing, personal mobile internet devices and other smart device applications. Based on a big.LITTLE architecture, it integrates a dual-core Cortex-A72 and a quad-core Cortex-A53. These 64bit-capable ARMv8 processors support both the ARM Crypographic Extension (e.g. for wire-rate AES encryption) and AdvSIMD vector processing. A dual-channel memory interface sustains the memory bandwidths required by even the most demanding embedded applications.

2 First Steps

This chapter provides instructions for getting the RK3399-Q7 EVK running after opening the box.

2.1 Required Tools

- PZ1 (Pozidriv) screwdriver

2.2 Insert the Module

Insert the RK3399-Q7 module at a 30-degree angle into the connector in the base board. Once fully inserted , push it down until it rests on the standoffs and check alignment of the mounting holes.

Note: The module springs back into the 30-degree angle once released. This is expected, and alignment will be kept. The module will be secured into place together with the heatsink.

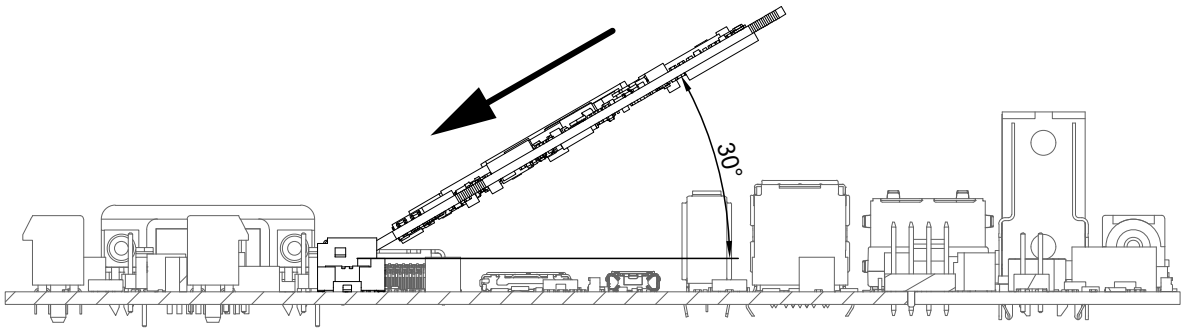


Fig. 1: Module mounting

2.3 Mount the Heatsink

The heatsink has the thermal pad attached on the bottom. Peel off the red protective foil.

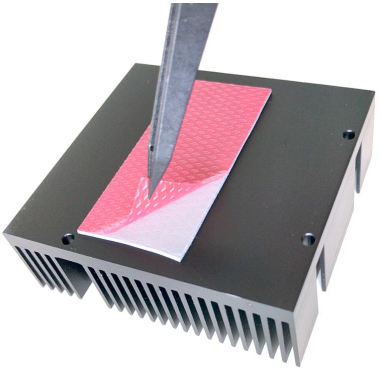


Fig. 2: Thermal pad protective foil

Push the module down flat and place the heatsink spacer on the module with the smooth side facing up. Make sure the orientation is correct by checking alignment of the mounting holes. Place the heatsink on the spacer and screw it down gently using the four included M2.5 screws.

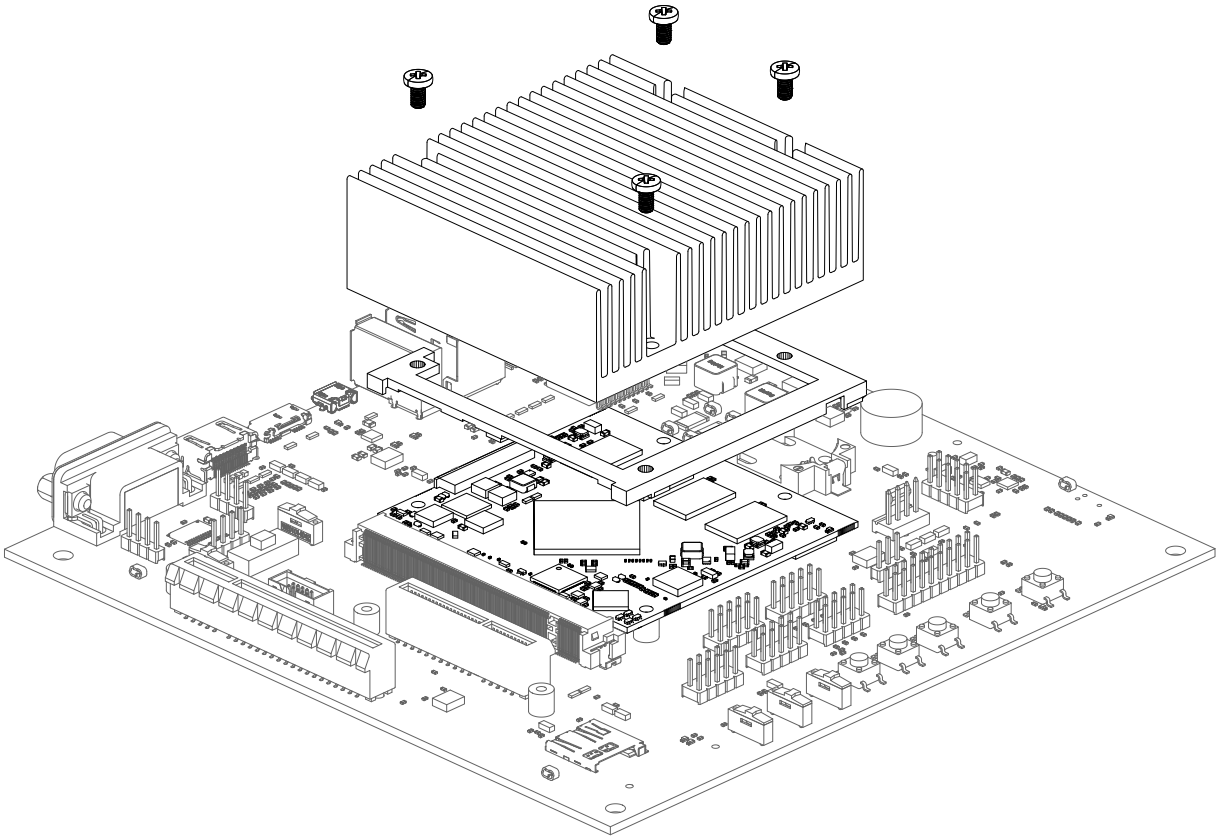


Fig. 3: Heatsink mounting

2.4 Mount the Fan (optional)

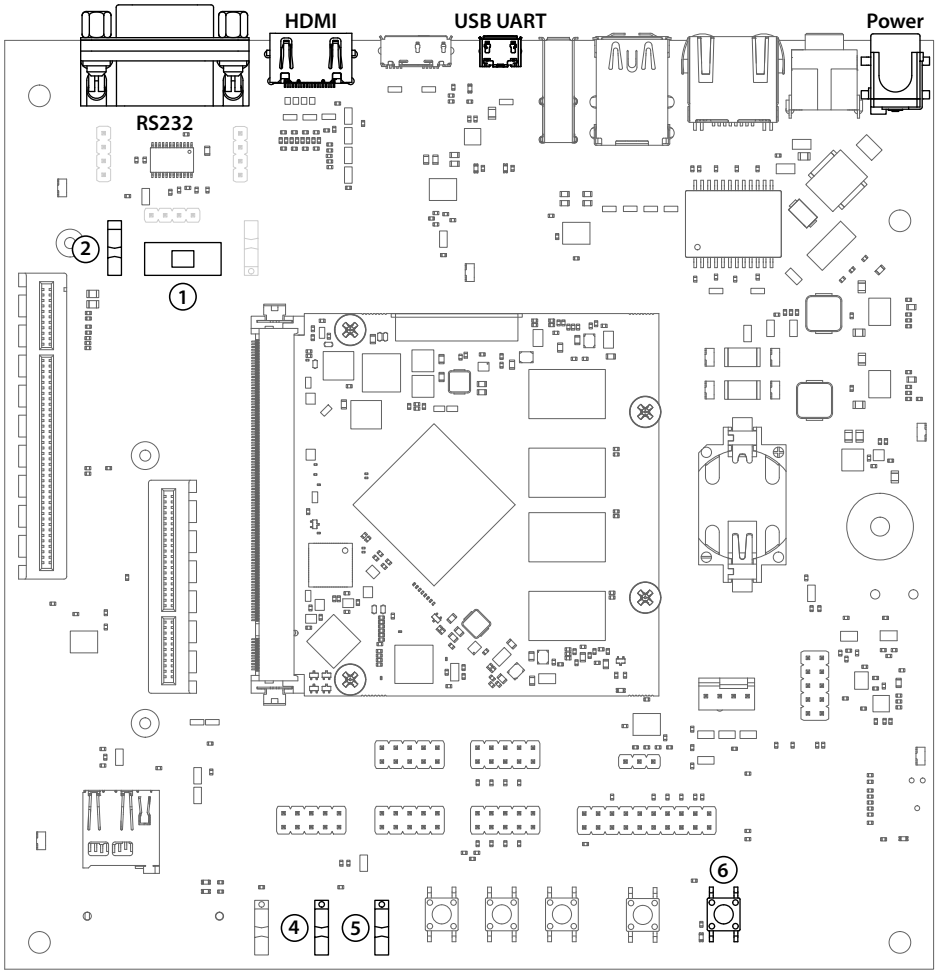
The fan is only necessary in exceptionally high ambient temperatures. Under normal conditions, the RK3399-Q7 operates passively cooled.

Place the fan on the heatsink and clip it into place using the provided holder. Connect the cable to the header labeled `FAN` on the baseboard.

2.5 Power Up

The EVK modules comes preloaded with software that boots into a full desktop environment on the HDMI output. For bootloader configuration and Linux console the serial interface can be used. Connect either a Micro-USB or RS-232 cable to the corresponding port. Select the correct UART with UART selector slider (1). For Micro-USB the slider has to be in the right position to route the default console UART0 to the USB UART bridge. For RS-232 the slider has to be in the left position and the protocol slider (2) has to be in the RS-232 position.

Connect the power supply and verify the sliders are in the position `Normal Boot` (4) and `Normally Off` (5). Press the `Power Button` (6) to power the board. You will see the boot progress and later on a login prompt on the serial interface. If a HDMI display is connected video output will follow shortly after.


HDMI
USB UART
Power
RS232
2
1
4
5
6

3 Using the EVK

This chapter provides instructions for using the EVK, such as booting and how to configure and use I/O peripherals (e.g. serial console, Ethernet).

3.1 Evaluation Board Overview

An overview of the available connectors and devices on the EVK is shown below.

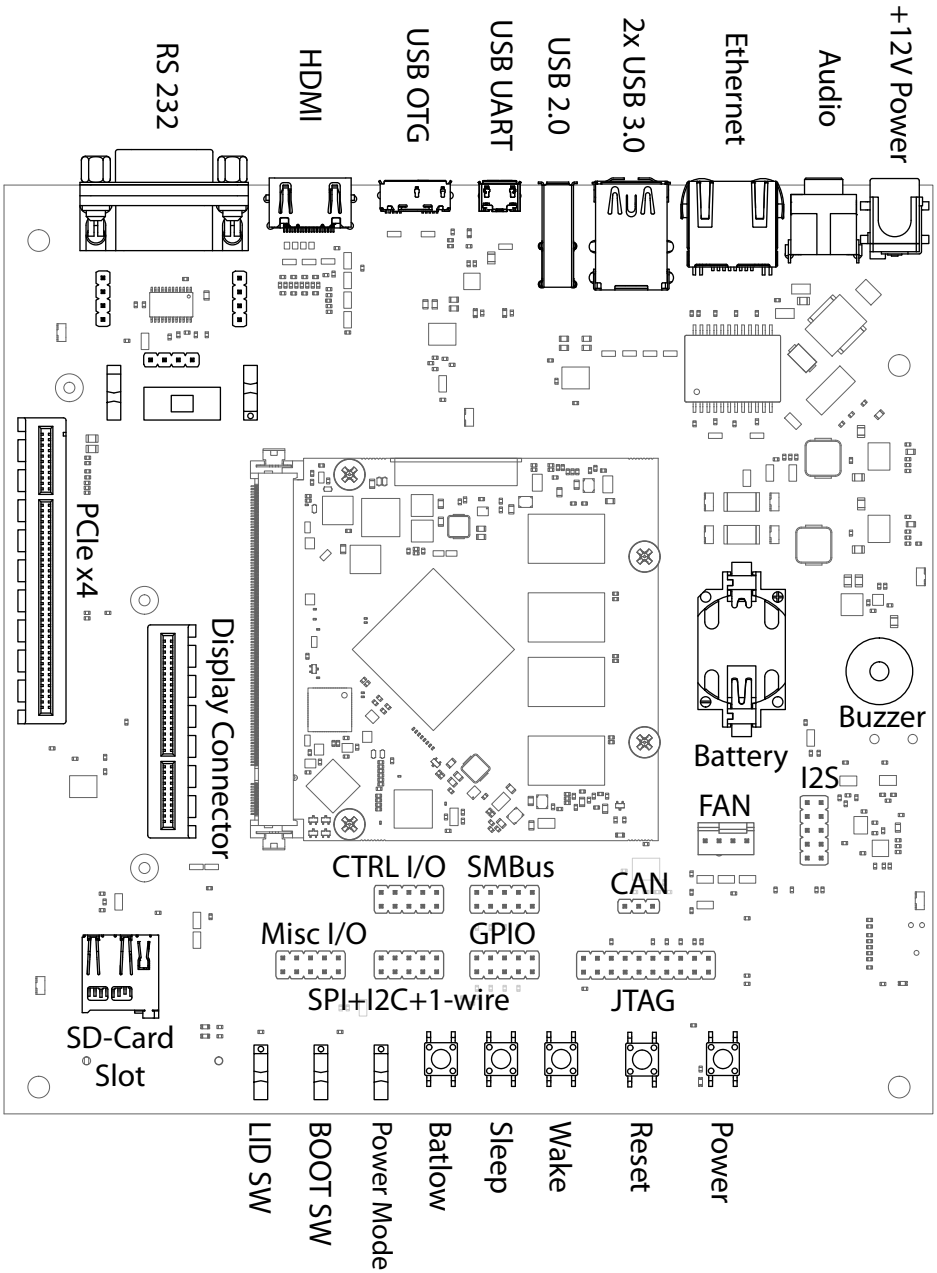


Fig. 1: The base board for to RK3399-Q7 module.

3.2 Power Supply

The baseboard can operate with a single 12V DC power supply.

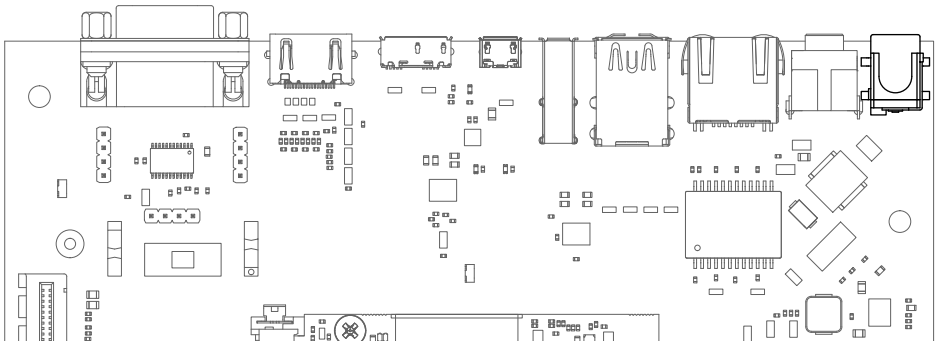


Fig. 2: 12V Power connector

Power can be controlled manually from the board using the `Power` control buttons and switches, located on the lower right side of the board (see *3.1 Evaluation Board Overview*).

Depending on the setting of `Normally On / Normally Off` switch the board will boot as soon as it receives power.

3.3 Control Buttons and Switches

The control buttons provide the following functionality:

- `Power` toggles the module power supply
- `Reset` triggers a module reset
- `Batlow`, `Sleep` and `Wake` are routed to GPIOs on the Q7 module

Several slide switches are located on the lower left:

- `LID` is routed to a GPIO on the module, simulates lid open/close.
- `Normally On / Normally Off`, as described above, sets the state after power loss.
- `BIOS Disable / Normal Boot` forces SD card boot or the normal boot order, respectively.

3.4 CPU Fan

Operation in high environmental temperatures may require a CPU fan. The fan connector is located next to the bottom right corner of the Q7 expansion area (see board overview).

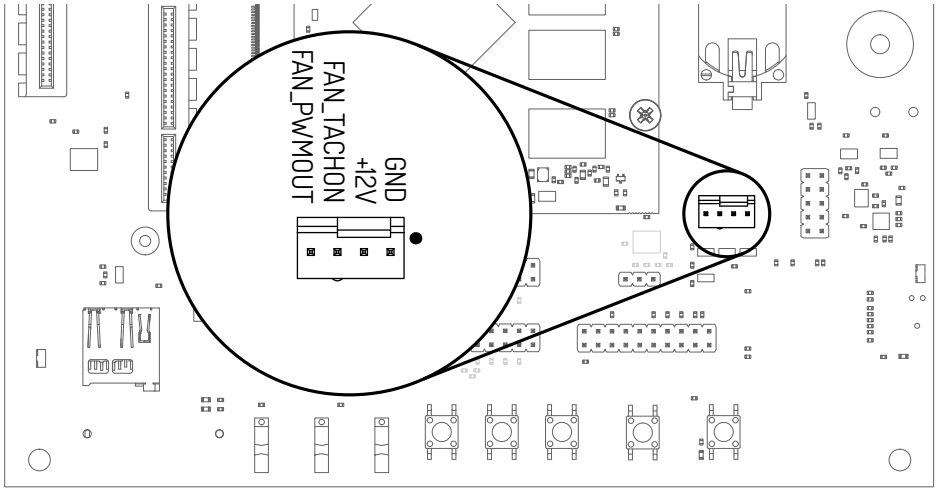


Fig. 3: Fan connector

Note: The fan is only necessary in high ambient temperatures. Under normal conditions, the RK3399-Q7 operates passively cooled.

3.5 Boot Order

The used boot order of the RK3399-Q7 module depends on the value of the `BIOS_DISABLE#` signal. On the Haikou baseboard, this signal can be set using a slider switch, with the two positions labeled *Normal Boot*, and *BIOS Disable*.

As shown in the table below, the *BIOS Disable* position disables both on-module storage devices:

	Normal Boot	*BIOS Disable*
1	SPI NOR flash	SD card
2	eMMC storage	USB loader
3	SD card	
4	USB loader	

If no bootloader is found on any storage device, the RK3399-Q7 module will go into USB loader mode where it shows up as a USB device on the USB-OTG port.

The electrical state of the `BIOS_DISABLE#` signal for both slider positions is shown below:

Slider Position	`BIOS_DISABLE#` signal
Normal Boot	Floating (on-module pull-up to 1.8V)
BIOS Disable	GND

3.6 USB Serial Console

The evaluation board contains an on-board Silicon Labs CP2102N USB-serial converter. Connect the included Micro-USB cable to the Micro-USB jack labeled `USB-UART Bridge`:

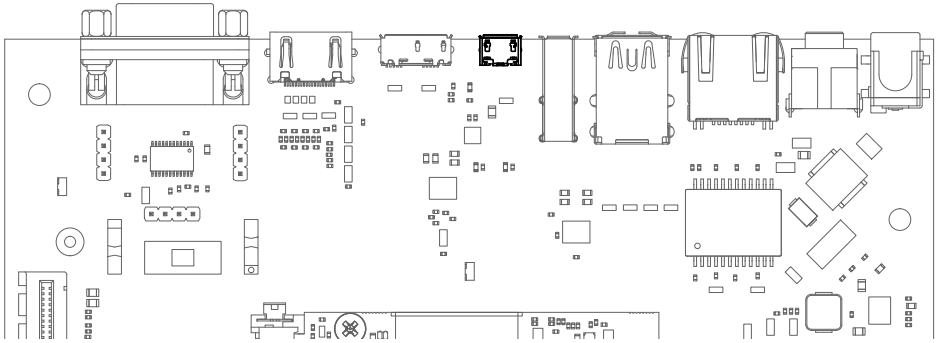


Fig. 4: USB UART

The serial converter does not require additional drivers on Windows and Linux.

For Mac OS, drivers are available from Silicon Labs: http://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

The Q7 modules has two external UARTs:

- UART0 is, by default, used for the serial console for interactive login.
- UART1 is unused by default and can be freely used for machine-to-machine communications or other purposes.

The switch `UART0 / UART1` cross-switches UART0 and UART1 between the `RS232 / RS485` jack and the onboard USB-serial converter:

Switch Position	`RS232 / RS485` jack connected to:	USB-serial converter connected to:
`UART0`	UART0 (interactive console)	UART1
`UART1`	UART1	UART0 (interactive console)

For interactive login through the USB-serial converter, make sure the switch is on the `UART1` position

Picocom can be used to connect via the serial line (assuming the USB-serial converter is USB0):

```
picocom -b 115200 /dev/ttyUSB0
```

Note: Make sure to disable software flow-control (XON/XOFF). Otherwise serial input may not be recognized.

After system bootup, the login console appears on the terminal:

```
rk3399-q7 login:
```

You can log in as `root` with password `root` or as user `user` with password `user`.

3.7 RS-232 and RS-485

To connect via RS-232 or RS-485, connect to the `RS232 / RS485` jack on the base board.

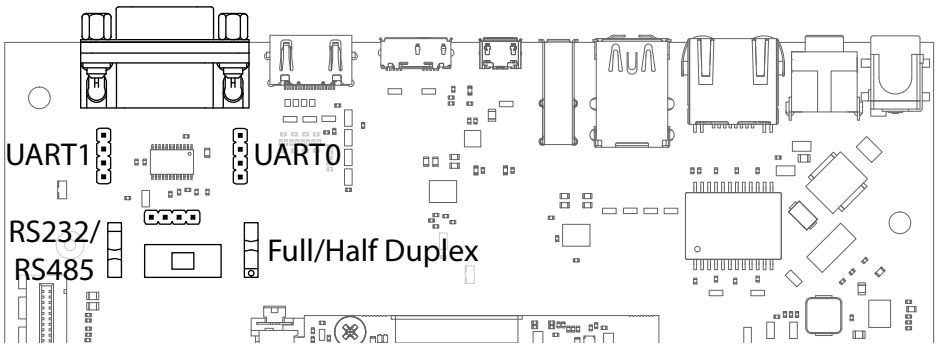


Fig. 5: RS-232 connector

The switch labeled `RS-232 / RS-485` selects between RS-232 and RS-485 mode on the jack.

In RS-485 mode, the switch labeled `Full Duplex / Half Duplex` selects full- or half-duplex mode, respectively. It has no effect in RS-232 mode, which is always full-duplex.

3.8 TTL UART

UART0 and UART1 are also available through the pin headers `P12 UART0` and `P30 UART1` next to the `RS232 / RS485` jack. The signal level is 3.3V.

3.9 Ethernet

The RK3399-Q7 has built-in Gigabit Ethernet routed to a standard jack on the evaluation board.

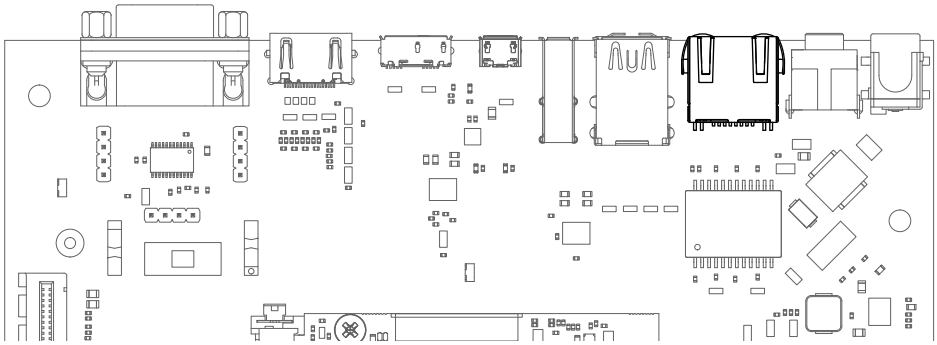


Fig. 6: Ethernet jack

The SD card that is shipped with the EVK is configures to automatically retrieve an IP via DHCP and provides SSH login on port 22.

3.10 USB Interfaces

The RK3399-Q7 provides four USB ports:

- 1x USB 3.0 OTG
- 2x USB 3.0 Host
- 1x USB 2.0 Host

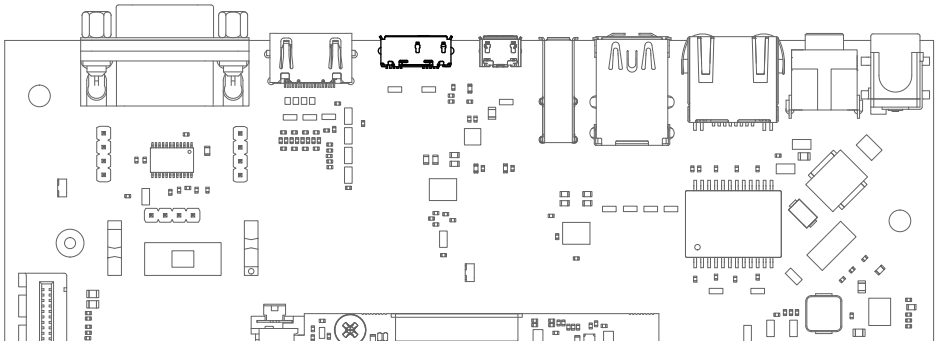


Fig. 7: USB 3.0 OTG port (dual-role port: can be used as a host or device interface)

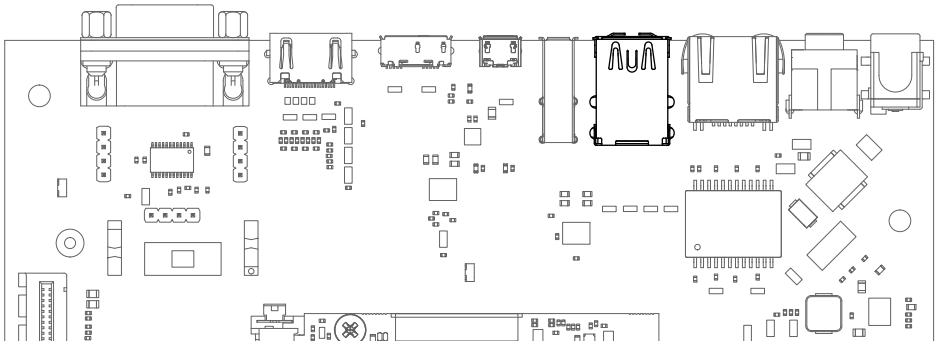


Fig. 8: USB 3.0 host ports

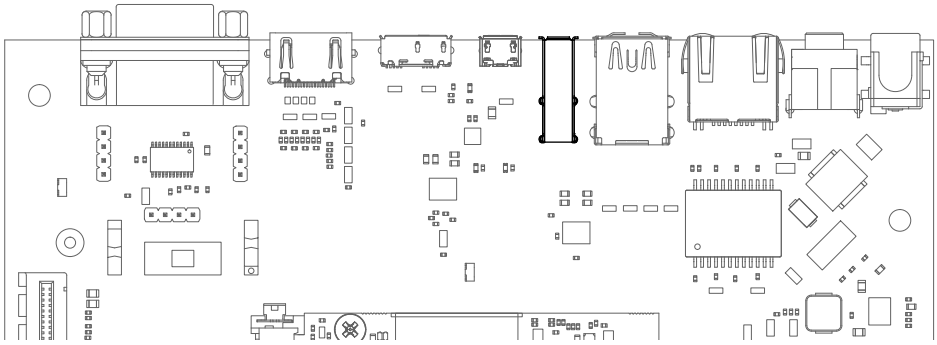


Fig. 9: USB 2.0 host port

3.10.1 Connecting an External USB Drive

To connect a USB drive, plug it into one of the USB ports. The system should recognize the drive immediately. Check the kernel log to find the device name:

```
journalctl -k -10
```

You will be able to mount its partitions (assuming mapping to `/dev/sdb1`):

```
mkdir /mnt/usb1
mount /dev/sdb1 /mnt/usb1
ls /mnt/usb1
```

3.11 Video

The RK3399-Q7 supports video output on HDMI, eDP and MIPI-DSI. Some of these interfaces are muxed on the module.

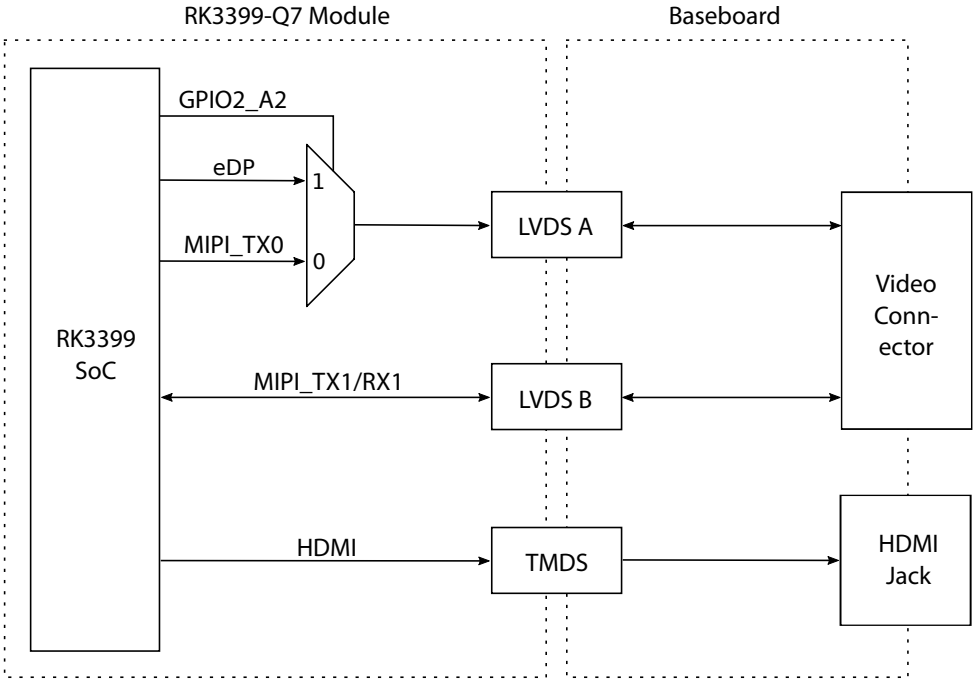


Fig. 10: RK3399-Q7 video muxing

The EVK software will use HDMI as the default video output. Connect a display to the HDMI port and a desktop environment will be shown once booting has finished. Screen resolution will be set based on the display EDID information. Resolutions up to 3840x2160 are supported.

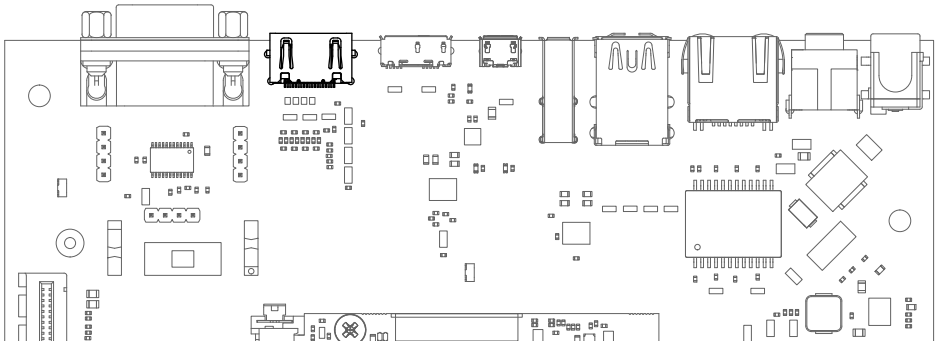


Fig. 11: HDMI port

For eDP and MIPI-DSI the Qseven LVDS pins are used which are routed to the `Display connect`. This expansion slot uses a PCIe connector as mechanical connection, which allows easy development of adapter boards for various different display types.

Qseven Port	Function	Alternate Function
LVDS A	MIPI-DSI	eDP
LVDS B	MIPI-DSI	MIPI-CSI

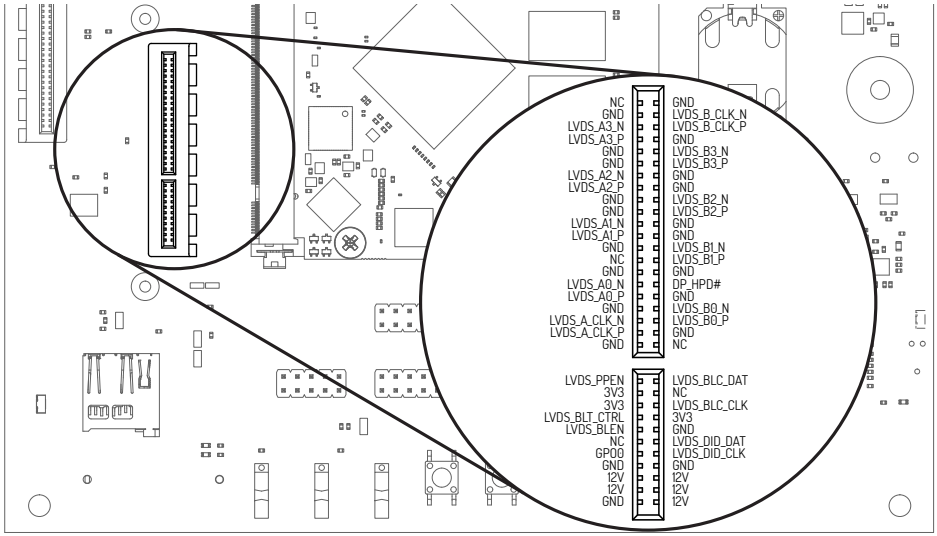


Fig. 12: Display connector pinout

The kernel devicetree defines the used display configuration. Example device trees for various output configurations are provided with the EVK software package.

To configure the bootloader which devicetree to load, edit the configuration variable `fdtfile` in the file `/boot/puma_rk3399/userEnv.txt`. For example to enable DisplayPort write:

```
fdtfile=rk3399-puma-edp.dtb
```

Filename	Display 1	Display 2
rk3399-puma.dtb	HDMI	
rk3399-puma-edp.dtb	Display Port on LVDS A	
rk3399-puma-mipidsi.dtb	MIPI-DSI on LVDS A	
rk3399-puma-hdmi+edp.dtb	HDMI	Display Port on LVDS A
rk3399-puma-hdmi+mipidsi.dtb	HDMI	MIPI-DSI on LVDS A

See https://git.theobroma-systems.com/som-hardware.git/ for video adapter reference designs.

3.12 RTC

the RK3399-Q7 contain a real-time clock (RTC) on-module. The RTC is read by the kernel on bootup and used to set the system clock.

To check the RTC value, use `hwclock`:

```
hwclock
Thu 30 Apr 2015 03:51:20 PM CEST  -0.826662 seconds
```

The RTC will be set automatically to the system clock on shutdown, so you can set the system clock using the date command and reboot to update the RTC:

```
date --set 2015-04-20
date --set 03:51:30
```

You can also update the RTC immediately, again with `hwclock`:

```
hwclock -v
```

You can set up an NTP client so the time will always be updated from the Internet. Install the client first:

```
apt-get install ntp
```

Feel free to change the `/etc/ntp.conf` file to use more local time sources (change servers from `pool.ntp.org` to use a server from your country, such as `at.pool.ntp.org`).

3.13 SPI, I2C and 1-Wire

I2C, 1-wire-bus and SPI interfaces are both available on the connector labeled `SPI+I2C+1-Wire`.

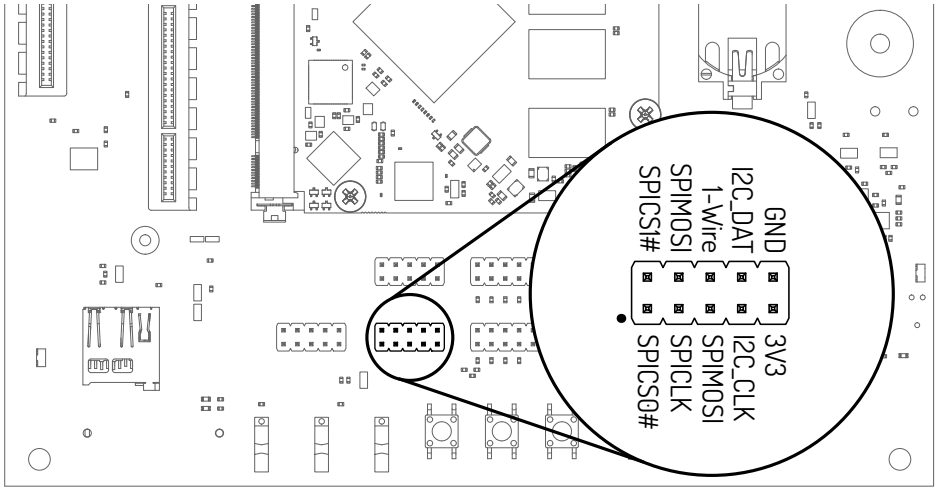


Fig. 13: I2C and SPI header

For I2C, there is the package i2c-tools available in Debian:

```
apt-get install i2c-tools
```

3.13.1 I2C Example - Using a Touch Keyboard

This example uses the Atmel AT42QT2160 touch keyboard (see datasheet).

Make sure the Linux kernel driver is enable via *menuconfig*:

```
make menuconfig
```

Navigate to *Device Drivers -> Input device support -> Keyboards* and check the *ATMEL AT42QT2160 Touch Sensor Chip*. You must recompile the kernel and deploy it to the SD card (see Software Guide).

3.13.2 SMBus

The board provides communication through SMBus. It is basically like I2C with an additional line for interrupt and is used for connecting sensors and power peripherals.

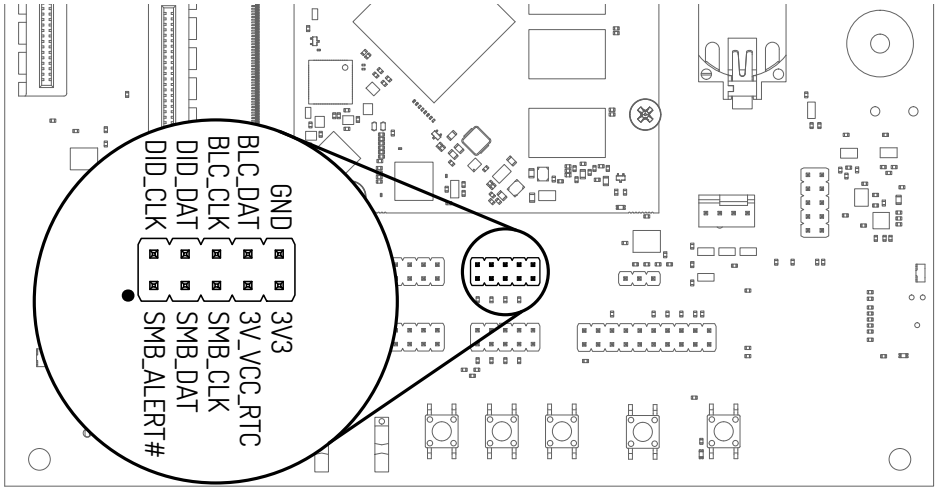


Fig. 14: SMBUS header

3.13.3 Linux Bus Numbering

Linux identifies each I2C bus using a bus number. The table below shows the mapping between Q7 names, Linux bus number and EVK header.

Q7 name	Linux bus #	EVK Header
GP0_I2C	4	SPI+I2C+1-Wire
SMB / GP1_I2C	2	SMBus
GP2_I2C / LVDS_DID	1	Display connector
LVDS_BLC	7	Display connector

The other I2C buses (as reported by *i2cdetect -l*) are internal to the module and not routed to the Q7 connector.

3.14 GPIOs

Eight GPIOs are provided on the pin header labeled `GPIO`.

The location on the board is displayed below:

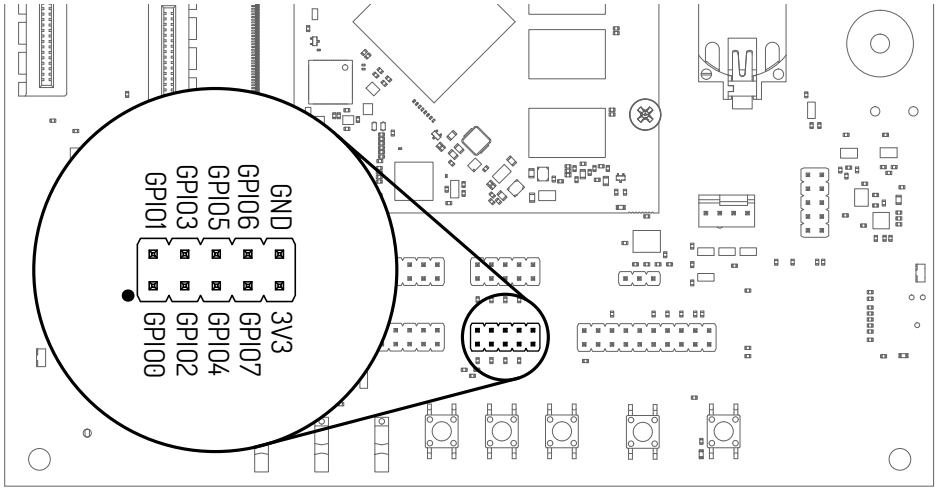


Fig. 15: GPIO header

The GPIO numbers printed on the board refer to numbers used in the Qseven specification. They are different than the ones used in Linux via */sys/class/gpio*.

The mapping is shown in the following table:

Qseven signal	CPU pin	Linux GPIO #
GPIO0	GPIO4_D4	156
GPIO1	GPIO4_D1	153
GPIO2	GPIO4_D0	152
GPIO3	GPIO4_D5	157
GPIO4	GPIO4_D2	154
GPIO5	GPIO4_C4	148
GPIO6	GPIO4_C3	147
GPIO7	GPIO4_D3	155

To calculate the Linux GPIO # for CPU pins that are not listed in this table, use the following formula:

```
n = (block_number * 32) + (sub_block_number * 8) + index
```

Where:

- block_number: index of the block number

- sub_block_number: the alphabetical index of the block name, minus 1
- index: the pin number within the block

Example:

```
GPIO4_D4 -> (4 * 32) + (3 * 8) + 4 = 156
```

To enable a GPIO, write the Linux GPIO # to the special *export* file:

```
echo 156 > /sys/class/gpio/export
ls /sys/class/gpio/gpio156
cat /sys/class/gpio/gpio156/direction
in
cat /sys/class/gpio/gpio156/value
0
```

To set the direction to output, write *out* in the GPIO's direction file:

```
echo out > /sys/class/gpio/gpio156/direction
echo 1 > /sys/class/gpio/gpio156/value
```

The GPIO will be set to a value of *1* (high at 3.3V).

3.15 Audio

The board provides two audio connectors for input and output. `Line-in` is on top and `Headphones` is on bottom of the audio connector.

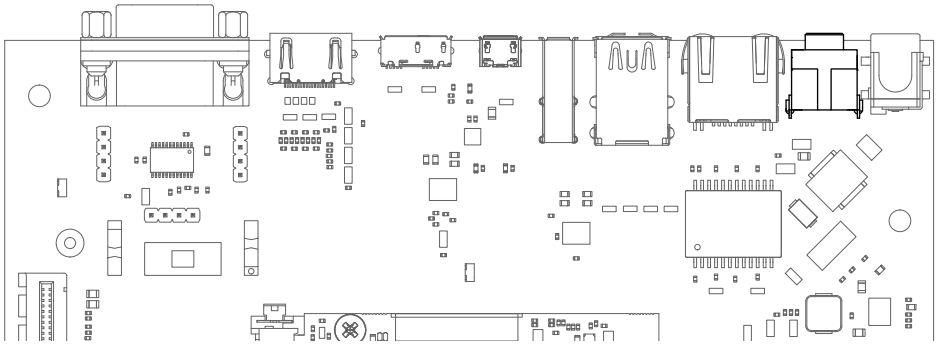


Fig. 16: Audio input/output port

Additionally, an expansion connector for I2S audio is available on the bottom row of the board:

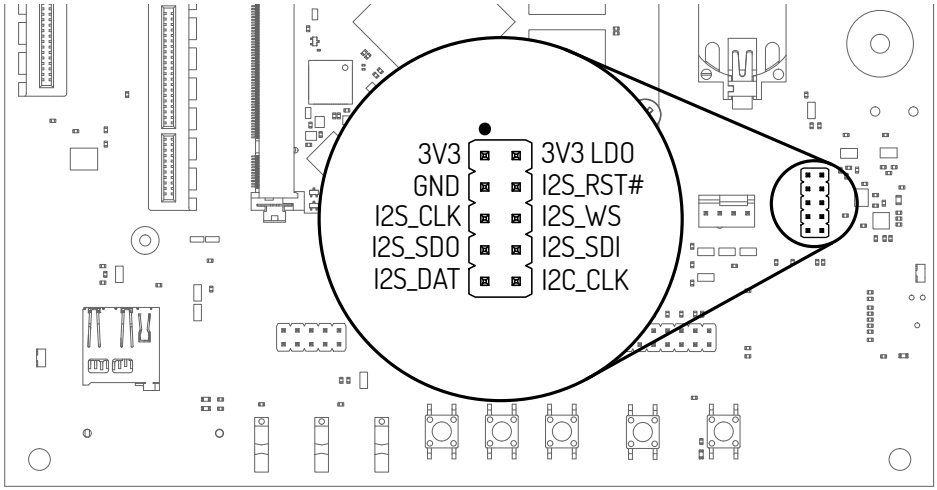


Fig. 17: Connecting to the audio expansion connector

3.16 CAN Bus

The board provides a CAN connector on the bottom row.

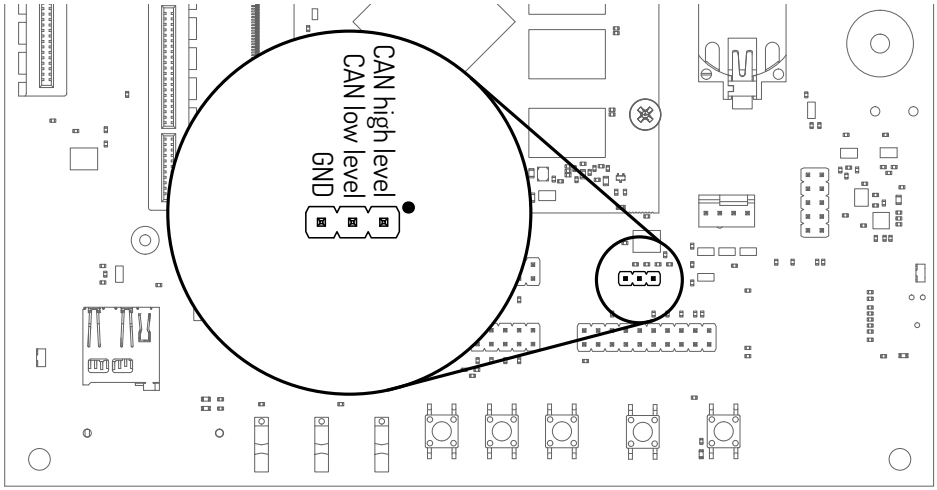


Fig. 18: CAN header

3.17 MISC Connector

The board provides signals for thermal overheat of external hardware and the processor, utility signals for SD and GPIO0.

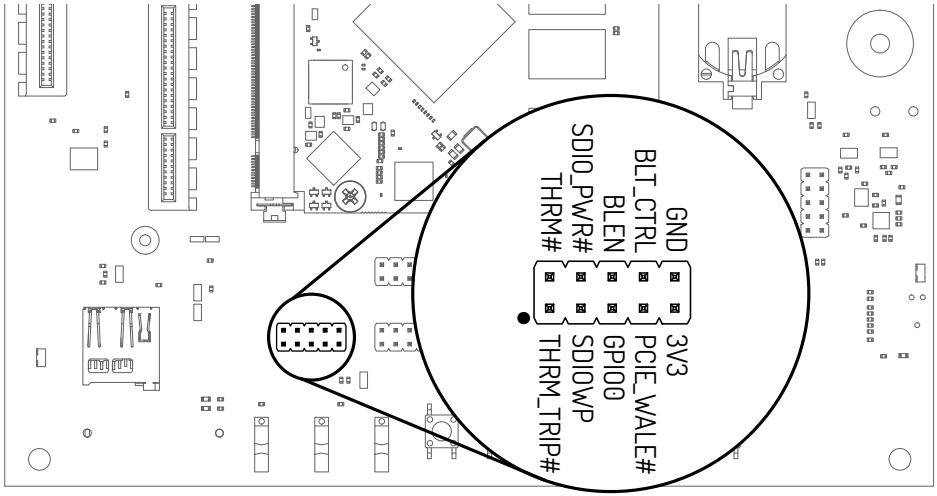


Fig. 19: MISC header

3.18 RF-Module

An additional RF-Module for wireless communication can be soldered on the bottom right of the baseboard.

For more information visit: https://www.theobroma-systems.com/rf-modules-3815

4 Software Guide

This chapter provides instructions for compiling and deploying the BSP (Board Support Package) software to the Q7 module.

4.1 Architecture Overview

The BSP consists of several parts. They run on different parts of the CPU and each play their role in the boot process. Because the CPU contains cores running different instruction sets (ARMv6-M and ARMv8-A), two different compilers are needed. The table below list the parts and their instruction set.

BSP Part	Architecture
Cortex-M0 power management firmware	ARMv6-M
ATF (ARM Trusted Firmware)	ARMv8-A
U-Boot bootloader	ARMv8-A
The Linux kernel	ARMv8-A
Debian user-space	ARMv8-A

The next section explains how to install suitable cross-compilers for both instruction sets.

The section “Compiling Linux Applications” provides guidance for compiling user-space applications for the RK3399.

4.2 Prerequisites

You need a recent x86_64 Linux installation to run the cross-compiler on and at least 10GB of disk space. The cross-compiler requires libc version 2.2.5. Distributions shipping this version are, among others:

- Ubuntu 16.04 “Xenial”
- Debian 8 “Jessie”
- Debian 9 “Stretch”

We recommend Debian 9 “Stretch” or Ubuntu 16.04 “Xenial”. Please install the following packages to set up the common build infrastructure:

```
sudo apt install device-tree-compiler u-boot-tools build-essential git bc debootstrap qemu-user-
↪static libssl-dev
```

4.2.1 ARMv6-M Compiler

The "GNU Embedded Toolchain for ARM" is suitable for compiling the Cortex-M0 power management firmware. It can be downloaded from https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads .

For this manual, it is assumed that version `6-2017-q1` is used. Direct link to the file:

https://developer.arm.com/-/media/Files/downloads/gnu-rm/6_1-2017q1/gcc-arm-none-eabi-6-2017-q1-update-linux.tar.bz2

Extract the tar.bz2 archive to /opt:

```
sudo tar -xf gcc-arm-none-eabi-6-2017-q1-update-linux.tar.bz2 -C /opt
```

4.2.2 ARMv8-A Compiler

The Linaro aarch64-linux-gnu toolchain is suitable for compiling all other parts of the BSP. It is also suitable for compiling user-space applications. You can download ready-to-use binaries from Linaro: https://releases.linaro.org/components/toolchain/binaries/6.3-2017.02/aarch64-linux-gnu/ .

Direct link to the file:

https://releases.linaro.org/components/toolchain/binaries/6.3-2017.02/aarch64-linux-gnu/gcc-linaro-6.3.1-2017.02-x86_64_aarch64-linux-gnu.tar.xz

Extract the tar.xz archive to /opt:

```
sudo tar -xf gcc-linaro-6.3.1-2017.02-x86_64_aarch64-linux-gnu.tar.xz -C /opt
```

4.3 Compile the Cortex-M0 power management firmware

The Cortex-M0 firmware runs inside a microcontroller embedded in the CPU IC. It implements power-management functionality and helpers (e.g. DRAM frequency switching support).

Set up environment variables to make use of the ARMv6-M compiler, then download the source code and compile:

```
export ARCH=arm64
export CROSS_COMPILE=/opt/gcc-arm-none-eabi-6-2017-q1-update/bin/arm-none-eabi-
git clone https://git.theobroma-systems.com/rk3399-cortex-m0.git
cd rk3399-cortex-m0
make
cd ..
```

4.3.1 Optional: Compile the cross-compiler

As an alternative to using a ready-made compiler, the firmware repository has a mechanism to compile the ARMv6-M-compiler as a part of the build process. This is called "internal toolchain".

If you want to use the internal toolchain instead you will also need the following packages:

```
sudo apt install libssl-dev autoconf gperf bison flex texinfo help2man gawk libncurses5-dev
```

Then to use the internal toolchain, specify "USE_INTERNAL_TOOLCHAIN=1" as part of your invocation to make.:

```
make USE_INTERNAL_TOOLCHAIN=1
```

4.4 Compile the ATF

Download the source code and compile using:

```
export CROSS_COMPILE=/opt/gcc-linaro-6.3.1-2017.02-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
git clone https://git.theobroma-systems.com/arm-trusted-firmware.git
cd arm-trusted-firmware
make PLAT=rk3399 bl31
cd ..
```

4.5 Compile U-Boot

U-Boot is used as the bootloader on the RK3399-Q7 module.

Download the source code using:

```
git clone https://git.theobroma-systems.com/puma-u-boot.git
```

The U-Boot build process uses the files generated in the previous steps Copy the previously generated files `rk3399m0.bin` from the Cortex-M0 firmware and `bl31.bin` from the ATF to the puma-u-boot directory. Recent U-Boot releases expect the `bl31.bin` file under the name `bl31-rk3399.bin`. To support all variants, the file is copied twice:

```
cp rk3399-cortex-m0/rk3399m0.bin puma-u-boot
cp arm-trusted-firmware/build/rk3399/release/bl31.bin puma-u-boot/bl31.bin
cp arm-trusted-firmware/build/rk3399/release/bl31.bin puma-u-boot/bl31-rk3399.bin
```

Then you are ready to compile U-Boot:

```
cd puma-u-boot
export ARCH=arm64
export CROSS_COMPILE=/opt/gcc-linaro-6.3.1-2017.02-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
make puma-rk3399_defconfig
make -j4
tools/mkimage -n rk3399 -T rksd  -d spl/u-boot-spl.bin spl_sd.img
tools/mkimage -n rk3399 -T rkspi -d spl/u-boot-spl.bin spl_spi.img
make u-boot.itb
cd ..
```

The resulting bootloader consists of three files: `spl_sd.img`, `spl_spi.img` and `u-boot.itb`, but only one of the `spl` files is used at a time. The file `spl_sd.img` is only used when booting from SD-card or eMMC, while `spl_spi.img` is only used when booting from SPI NOR flash.

4.6 Compile the Boot Script

The U-Boot boot sequence is controlled by a file called `boot.scr`. This file is generated from a plain-text file called `boot.cmd`.

Download the repository and generate `boot.scr` using:

```
git clone https://git.theobroma-systems.com/som-tools.git
cd som-tools
make -C boot-script
cd ..
```

4.7 Compile the Linux Kernel

The kernel source code can be cloned with:

```
git clone https://git.theobroma-systems.com/puma-linux.git
```

Compile using:

```
cd puma-linux
export ARCH=arm64
export CROSS_COMPILE=/opt/gcc-linaro-6.3.1-2017.02-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
make puma-rk3399_defconfig
make -j4 rockchip/rk3399-puma.dtb Image
```

This will create the two files needed for booting with U-Boot (paths are relative to the `puma-linux` directory):

- The device tree: `arch/arm64/boot/dts/rockchip/rk3399-puma.dtb`
- The kernel image: `arch/arm64/boot/Image`

4.8 Building the root filesystem

A filesystem can be created using Debootstrap, specifying arm64 as architecture in the command line.

Supposing the target dir is called `rk3399-rootfs` and the chosen distribution is Debian 9 "Stretch" (recommended):

```
export targetdir=/opt/rk3399-rootfs
sudo mkdir -p $targetdir
sudo debootstrap --arch=arm64 --foreign stretch $targetdir http://deb.debian.org/debian/
```

Next, copy the qemu-arm-static binary into the right place for the binfmt packages to find it and copy the resolv.conf file from the host system:

```
sudo cp /usr/bin/qemu-aarch64-static $targetdir/usr/bin/
sudo cp /etc/resolv.conf $targetdir/etc
```

This will provide a very basic arm64 rootfs in the targetdir. For the next stages, we `chroot` to the target dir:

```
sudo chroot $targetdir
```

Second stage of debootstrap inside the new root dir:

```
/debootstrap/debootstrap --second-stage
```

Set up the apt package sources:

```
cat << EOT > /etc/apt/sources.list
deb http://deb.debian.org/debian stretch main contrib non-free
deb http://deb.debian.org/debian stretch-updates main contrib non-free
deb http://security.debian.org/ stretch/updates main contrib non-free
EOT
```

We can now pull the latest apt database from the Debian mirrors and install locales:

```
apt update
apt install locales
echo "en_US.UTF-8 UTF-8" > /etc/locale.gen
locale-gen
```

Install any additional packages inside the chroot. An ssh server and sudo are recommended:

```
apt install openssh-server sudo
```

Set the root password for logging in via serial console:

```
passwd
```

To log in over ssh, create another user besides root (root login over ssh is not permitted by default):

```
adduser user
```

Add the new user to the `sudo` group so you can switch to root privileges if needed:

```
adduser user sudo
```

Set up a basic network configuration file with DHCP via eth0 and enable automatic DNS configuration through systemd-resolved:

```
cat << EOT > /etc/systemd/network/eth0.network
[Match]
Name=eth0
[Network]
DHCP=yes
EOT
```

(continues on next page)

(continued from previous page)

```
systemctl enable systemd-networkd
systemctl enable systemd-resolved
ln -f -s /lib/systemd/resolv.conf /etc/resolv.conf
```

When executing `systemctl enable` you may get the message `qemu: Unsupported syscall: 278`. The operation still succeeds and the message can be safely ignored.

Set the hostname and exit from the chroot:

```
echo rk3399-q7 > /etc/hostname
exit
```

We now have a root filesystem which can be deployed to the SD card.

4.9 Deploy on SD Card

4.9.1 Partition Setup

Both U-Boot and Linux will be located on the same SD card. The layout of the card after setup is as follows:

Offset	Contents	Files
0	Partition table	
32kiB	U-Boot SPL	spl_sd.img
240kiB	U-Boot environment	
256kiB	U-Boot + ATF + Cortex-M0 firmware	u-boot.itb
2MiB	Partition 1 (ext4 - Linux root fs)	boot.scr, Image, rk3399-puma.dtb, defaultEnv.txt and rootfs

To setup a SD card for booting you first need to create partitions. Partitions can be created using fdisk (assuming the SD card is mapped to /dev/sd“X“ , where `X` should be replaced with your corresponding device-letter) and has no partitions (this can be checked using the `p` command):

```
sudo fdisk /dev/sdX
> p
```

This should show an empty partition table, for example:

```
Disk /dev/sdX: 3980 MB, 3980394496 bytes
123 heads, 62 sectors/track, 1019 cylinders, total 7774208 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xdbbd45c7

Device Boot      Start         End      Blocks   Id  System
```

If there are partitions on the sdcard, they can be deleted with `o`.

The required partition can be created with the command `n`, then accepting the defaults, except for `First sector`, where we use 4096:

```
> n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): <ENTER>
Partition number (1-4, default 1): <ENTER>
First sector (2048-7774207, default 2048): 4096
Last sector, +sectors or +size{K,M,G} (...): <ENTER>
```

This will create a primary partition at offset 2MiB. Enter `w` to write the new partition table to the disk:

```
> w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Now we format the partition as ext4:

```
sudo /sbin/mkfs.ext4 -E lazy_itable_init=0 /dev/sdX1
```

The option `lazy_itable_init=0` speeds up the first boot because it initializes the inode tables in advance.

The SD card is now ready to have the U-Boot bootloader and Linux deployed.

4.9.2 Deploy U-Boot

The U-Boot images `spl_sd.img` and `u-boot.itb` are written to the SD card. Assuming the SD card is mapped to /dev/sdX:

```
sudo dd if=puma-u-boot/spl_sd.img of=/dev/sdX bs=1k seek=32  conv=nocreat
sudo dd if=puma-u-boot/u-boot.itb of=/dev/sdX bs=1k seek=256 conv=nocreat
```

4.9.3 Deploy the Linux Kernel and the Root Filesystem

Mount the SD card partition and copy the rootfs (assuming that the rootfs is located at `/opt/rk3399-rootfs` and the sd card at `/dev/sdX1`):

```
sudo mkdir -p /mnt/sdcard
sudo mount /dev/sdX1 /mnt/sdcard
sudo cp -av /opt/rk3399-rootfs/* /mnt/sdcard
```

Copy kernel image, device tree and boot script into the boot directory:

```
sudo cp -r som-tools/boot-script/boot/{boot.scr,puma_rk3399} /mnt/sdcard/boot
sudo cp puma-linux/arch/arm64/boot/dts/rockchip/rk3399-puma.dtb /mnt/sdcard/boot/puma_rk3399
sudo cp puma-linux/arch/arm64/boot/Image /mnt/sdcard/boot/puma_rk3399
```

Finally, unmount the SD card:

```
sudo umount /mnt/sdcard
```

The SD card is ready for booting.

4.9.4 U-Boot Customization

The boot script `/boot/boot.scr` handles the boot sequence. Unless you want to customize the sequence, no further action is required.

If you want to step through the sequence manually or customize it, you can execute the following commands on the U-Boot prompt:

```
setenv bootargs root=/dev/mmcblk0p1 rw rootwait
ext4load mmc 1:1 $kernel_addr_r boot/puma_rk3399/Image
ext4load mmc 1:1 $fdt_addr_r boot/puma_rk3399/rk3399-puma.dtb
booti $kernel_addr_r - $fdt_addr_r
```

Optionally, these commands can be compiled together in a single command and saved so it is performed on every subsequent boot:

```
setenv bootargs root=/dev/mmcblk0p1 rw rootwait
setenv boot_sd "ext4load mmc 1:1 $kernel_addr_r boot/puma_rk3399/Image && \
ext4load mmc 1:1 $fdt_addr_r boot/puma_rk3399/rk3399-puma.dtb && \
booti $kernel_addr_r - $fdt_addr_r"
setenv bootcmd run boot_sd
saveenv
```

To reset the U-Boot settings to default, execute:

```
env default -f -a
saveenv
```

Note: `root=/dev/mmcblk0p1` and `ext4load mmc 1:1` refer to the SD Card. Use `root=/dev/mmcblk1p1` and `ext4load mmc 0:1` instead to boot from eMMC.

4.10 Deploy on SPI NOR-flash

To have a reliable boot sequence even if eMMC and/or SD-card fail, U-Boot can be written in to the onboard NOR-flash. While `u-boot.itb` is used for both SD-card and SPI-flash boot, `spl_spi.img` is used instead of `spl_sd.img`.

From U-boot:

```
sf probe
load mmc 1 $kernel_addr_r root/spl_spi.img
sf erase 0 +$filesize
sf write $kernel_addr_r 0 $filesize

load mmc 1 $kernel_addr_r root/u-boot.itb
sf erase 0x40000 +$filesize
sf write $kernel_addr_r 0x40000 $filesize
```

From Linux:

```
dd if=spl_spi.img of=/dev/mtdblock0 bs=256k seek=0 conv=nocreat
dd if=u-boot.itb  of=/dev/mtdblock0 bs=256k seek=1 conv=nocreat
```

4.11 Deploy on On-Board eMMC storage

As the eMMC storage is only accessible from the module itself, you must first boot the RK3399-Q7 from SD card.

Partition and format the eMMC storage as described in *4.9.1 Partition Setup*, but using the device `/dev/mmcblk1`.

Mount the eMMC partition and copy the contents of the SD card to the eMMC storage. The copy process will take about 30 seconds:

```
sudo mkdir -p /mnt/emmc
sudo mount /dev/mmcblk1p1 /mnt/emmc
sudo cp -ax / /mnt/emmc
sudo umount /mnt/emmc
```

The final step is copying the bootloader to the eMMC:

```
dd if=/boot/puma_rk3399/spl_sd.img of=/dev/mmcblk1 bs=1k seek=32  conv=nocreat
dd if=/boot/puma_rk3399/u-boot.itb of=/dev/mmcblk1 bs=1k seek=256 conv=nocreat
```

Shut down the board (`poweroff` command) and remove the SD card. Make sure the boot selector switch is set to "`Normal Boot`". The next boot will run U-Boot off the internal eMMC storage.

4.12 Compiling Linux Applications

The easiest option is to compile your applications directly on a running RK3399-Q7 module. Just install the gcc package and related utilities and you are good to go:

```
sudo apt-get install build-essential
```

The second option is to cross-compile your applications. The ARMv8-A compiler that was installed earlier is suitable to compile applications for the RK3399-Q7.

4.13 Serial Number

Each RK3399-Q7 module has a unique serial number that can be read by software.

In U-Boot, the serial number is contained in the environment variable `serial#`. You can print it using the command:

```
printenv serial#
```

Under Linux, it is represented by a simple text file in `/sys`:

```
cat /sys/firmware/devicetree/base/serial-number
```

The serial number is fixed in hardware (derived from the SoC *CPU ID*) and cannot be modified.

4.14 MAC Address

By default, the MAC address of each RK3399-Q7 module is a random value derived from the serial number. The properties of this default MAC address are:

- It is a *Locally Administered Address*: The U/L bit of the MAC address is set to 1
- It is not guaranteed to be globally unique
- The address is fixed for each RK3399-Q7 module. It stays constant across reboots as it is deterministically derived from the serial number

To set your own *Universally Administered Address*, you overwrite the U-Boot environment variable `ethaddr`. On the U-Boot prompt, with *XX:XX:XX:XX:XX:XX* replaced by your MAC address:

```
setenv ethaddr XX:XX:XX:XX:XX:XX
saveenv
```

The MAC address can be queried from the U-Boot prompt using:

```
printenv ethaddr
```

To reset the MAC address to the default value, run:

```
env delete ethaddr
saveenv
```

5 Hardware Guide

This Hardware Guide provides information about the features, connectors and signals available on the RK3399-Q7 module.

5.1 Qseven Implementation

Qseven has mandatory and optional features. Following table shows the feature set of the RK3399-Q7 module compared to the minimum ARM/RISC based and maximum configuration according to the Q7 standard.

System I/O Interface	Q7 Minimum	RK3399-Q7	Q7 Maximum
PCI Express lanes	0	4	4
Serial ATA channels	0	0	2
USB 2.0 ports	1	1	8
USB 3.0 ports	0	3	3
LVDS channels	0	0	2
Embedded DisplayPort	0	1	1
MIPI_CSI	0	2	2
HDMI	0	1	1
High Definition Audio / AC‘97 / I2S	0	1	1
Ethernet 10/100/Gigabit	0	1x Gigabit	1x Gigabit
UART	0	1	1
GPIO	0	8	8
Secure Digital I/O	0	1	1
System Management Bus	0	1	1
I^2C Bus	1	4	4
SPI Bus	0	1	1
CAN Bus	0	1	1
Watchdog Trigger	1	1	1
Power Button	1	1	1
Power Good	1	1	1
Reset Button	1	1	1
LID Button	0	1	1
Sleep Button	0	1	1
Suspend to RAM (S3 mode)	0	1	1
Wake	0	1	1
Battery low alarm	0	1	1
Thermal control	0	1	1
FAN control	0	1	1

Note: The RK3399-Q7 module is available in different variants. This document describes the maximum configuration. For details about orderable variants please refer to the order-code document.

5.2 Q7 Connector Pinout

The following table shows the signals on the edge connector of the RK3399-Q7 module.

Empty cells are not connected (NC) pins.

Pin	Signal	Pin	Signal
1	GND	2	GND
3	GBE_MDI3-	4	GBE_MDI2-
5	GBE_MDI3+	6	GBE_MDI2+
7	GBE_LINK100#	8	GBE_LINK1000#
9	GBE_MDI1-	10	GBE_MDIO0-
11	GBE_MDI1+	12	GBE_MDIO0+
13	GBE_LINK#	14	GBE_ACT#
15		16	SUS_S5#
17	WAKE#	18	SUS_S3#
19	GP0	20	PWRBTN#
21	SLP_BTN#	22	LID_BTN#
23	GND	24	GND
25	GND	26	PWGIN
27	BATLOW#	28	RSTBTN#
29		30	
31		32	
33		34	GND
35		36	
37		38	
39	GND	40	GND
41	BIOS_DISABLE# / BOOT_ALT#	42	SDIO_CLK#
43	SDIO_CD#	44	SDIO_LED
45	SDIO_CMD	46	SDIO_WP
47	SDIO_PWR#	48	SDIO_DAT1
49	SDIO_DAT0	50	SDIO_DAT3
51	SDIO_DAT2	52	
53		54	
55		56	USB_OTG_PEN
57	GND	58	GND
59	I2S_WS	60	SMB_CLK / GP1_I2C_CLK

Continued on next page

Table 1 – continued from previous page

Pin	Signal	Pin	Signal
61	I2S_RST#	62	SMB_DAT / GP1_I2C_DAT
63	I2S_CLK	64	SMB_ALERT#
65	I2S_SDI	66	GP0_I2C_CLK
67	I2S_SDO	68	GP0_I2C_DAT
69	THRM#	70	WDTRIG#
71	THRMTRIP#	72	WDOUT
73	GND	74	GND
75	USB_SSTX0-	76	USB_SSRX0-
77	USB_SSTX0+	78	USB_SSRX0+
79		80	
81	USB_SSTX2-	82	USB_SSRX2-
83	USB_SSTX2+	84	USB_SSRX2+
85	USB_2_3_OC#	86	USB_0_1_OC#
87	USB_P3-	88	USB_P2-
89	USB_P3+	90	USB_P2+
91	USB_CC	92	USB_ID
93	USB_P1-	94	USB_P0-
95	USB_P1+	96	USB_P0+
97	GND	98	GND
99	LVDS_A0+	100	LVDS_B0+
101	LVDS_A0-	102	LVDS_B0-
103	LVDS_A1+	104	LVDS_B1+
105	LVDS_A1-	106	LVDS_B1-
107	LVDS_A2+	108	LVDS_B2+
109	LVDS_A2-	110	LVDS_B2-
111	LVDS_PPEN	112	LVDS_BLEN
113	LVDS_A3+	114	LVDS_B3+
115	LVDS_A3-	116	LVDS_B3-
117	GND	118	GND
119	LVDS_A_CLK+	120	LVDS_B_CLK+
121	LVDS_A_CLK-	122	LVDS_B_CLK-
123	LVDS_BLT_CTRL / GP_PWM_OUT0	124	GP_1-Wire_Bus
125	GP2_I2C_DAT / LVDS_DID_DAT	126	LVDS_BLC_DAT / eDP0_HPD#
127	GP2_I2C_CLK / LVDS_DID_CLK	128	LVDS_BLC_CLK
129	CAN0_TX	130	CAN0_RX
131	TMDS_CLK+	132	USB_SSTX1-

Continued on next page

Table 1 – continued from previous page

Pin	Signal	Pin	Signal
133	TMDS_CLK-	134	USB_SSTX1+
135	GND	136	GND
137	TMDS_LANE1+	138	
139	TMDS_LANE1-	140	
141	GND	142	GND
143	TMDS_LANE0+	144	USB_SSRX1-
145	TMDS_LANE0-	146	USB_SSRX1+
147	GND	148	GND
149	TMDS_LANE2+	150	HDMI_CTRL_DAT
151	TMDS_LANE2-	152	HDMI_CTRL_CLK
153	DP_HDMI_HPD#	154	
155	PCIE_CLK_REF+	156	PCIE_WAKE#
157	PCIE_CLK_REF-	158	PCIE_RST#
159	GND	160	GND
161	PCIE3_TX+	162	PCIE3_RX+
163	PCIE3_TX-	164	PCIE3_RX-
165	GND	166	GND
167	PCIE2_TX+	168	PCIE2_RX+
169	PCIE2_TX-	170	PCIE2_RX-
171	UART0_TX	172	UART0_RTS#
173	PCIE1_TX+	174	PCIE1_RX+
175	PCIE1_TX-	176	PCIE1_RX-
177	UART0_RX	178	UART0_CTS#
179	PCIE0_TX+	180	PCIE0_RX+
181	PCIE0_TX-	182	PCIE0_RX-
183	GND	184	GND
185	GPIO0	186	GPIO1
187	GPIO2	188	GPIO3
189	GPIO4	190	GPIO5
191	GPIO6	192	GPIO7
193	VCC_BAT	194	SPKR / GP_PWM_OUT2
195	FAN_TACHOIN / GP_TIMER_IN	196	FAN_PWMOUT / GP_PWM_OUT1
197	GND	198	GND
199	SPI_MOSI	200	SPI_CS0#
201	SPI_MISO	202	SPI_CS1#
203	SPI_SCK	204	

Continued on next page

Table 1 – continued from previous page

Pin	Signal	Pin	Signal
205		206	
207		208	
209		210	
211		212	
213		214	
215		216	
217		218	
219	VCC	220	VCC
221	VCC	222	VCC
223	VCC	224	VCC
225	VCC	226	VCC
227	VCC	228	VCC
229	VCC	230	VCC

5.3 Signal Details

5.3.1 Ethernet

Signal	Type	Signal Level	Description
GBE_MDI[0:3]+ GBE_MDI[0:3]-	I/O	Analog	Gigabit Ethernet Controller: Media Dependent Interface Differential Pairs 0,1,2,3. The MDI can operate in 1000, 100 and 10 Mbit/sec modes
GBE_ACT#	OC	3.3V	Gigabit Ethernet Controller activity indicator, active low
GBE_LINK#	OC	3.3V	Gigabit Ethernet Controller link indicator, active low
GBE_LINK100#	OC	3.3V	Internally connected to GBE_LINK#
GBE_LINK1000#	OC	3.3V	Internally connected to GBE_LINK#
GBE_CTREF	REF	Analog	Center Tap Voltage

5.3.2 USB

Signal	Type	Signal Level	Description
USB_P[0:2]+ USB_P[0:2]-	I/O	USB	High speed universal Serial Bus Port 0, 1, 2 differential pairs
USB_SSTX[0:2]+ USB_SSTX[0:2]-	I/O	USB	Super speed universal Serial Bus Port 0, 1, 2 transmit differential pairs
USB_SSRX[0:2]+ USB_SSRX[0:2]-	I/O	USB	Super speed universal Serial Bus Port 0, 1, 2 receive differential pairs
USB_0_1_OC#	I	3.3V	Over current detect input 1. This pin is used to monitor the USB power over current of the USB Ports 0 and 1
USB_2_3_OC#	I	3.3V	Over current detect input 1. This pin is used to monitor the USB power over current of the USB Ports 2 and 3
USB_ID	I	3.3V	Configures the mode of the USB Port 1. If the signal is `active high` the Port will be configured as USB Client
USB_VBUS	I	5.0V	USB VBUS pin, 5V tolerant
USB_OTG_PEN	O	3.3V	USB Power enable for OTG port USB 1

5.3.3 SDIO

Signal	Type	Signal Level	Description
SDIO_CD#	I	3.3V	SDIO Card Detect. This signal indicates when a SDIO/MMC card is present
SDIO_CLK	O	3.3V	SDIO Clock
SDIO_CMD	I/O	3.3V	SDIO Command/Response
SDIO_LED	O	3.3V	SDIO LED. Used to drive an external LED to indicate transfers on the bus
SDIO_WP	I	3.3V	SDIO Write Protect
SDIO_PWR#	O	3.3V	SDIO Power Enable. This signal is used to enable the power being supplied to a SD/MMC card device
SDIO_DAT0-4	I/O	3.3V	SDIO Data lines

5.3.4 I2C

Signal	Type	Signal Level	Description
Q7_I2C_CLK	O	3.3V	I2C bus clock line connected to RK3399
Q7_I2C_DAT	I/O	3.3V	I2C bus data line connected to RK3399
LVDS_DID_CLK /GP2_I2C_CLK	O	3.3V	I2C bus clock line connected to RK3399
LVDS_DID_DAT /GP2_I2C_DAT	I/O	3.3V	I2C bus data line connected to RK3399
SMB_CLK GP1_I2C_CLK	O	3.3V	Clock line of System Management Bus. Alternate function I2C Bus clock line
SMB_DAT GP1_I2C_DAT	I/O	3.3V	Data line of System Management Bus. Alternate function I2C Bus data line
LVDS_BLC_DAT	O	3.3V	I2C bus clock line connected to RK3399, Kerkey and baseboard EEPROM
LVDS_BLC_CLK	I/O	3.3V	I2C bus data line connected to RK3399, Kerkey and baseboard EEPROM

5.3.5 I2S

Signal	Type	Signal Level	Description
I2S_RST#	O	3.3V	I2S Codec Reset
I2S_WS	O	3.3V	I2S Word Select
I2S_CLK	O	3.3V	I2S Serial Data Clock
I2S_SDO	O	3.3V	I2S Serial Data Output
I2S_SDI	I	3.3V	I2S Serial Data Input

5.3.6 HDMI

Signal	Type	Signal Level	Description
TMDS_CLK+ TMDS_CLK-	O	TMDS	TMDS differential pair clock lines
TMDS_LANE[0:2]+ TMDS_LANE[0:2]-	O	TMDS	TMDS differential pair lanes 0, 1, 2
HDMI_CTRL_CLK	O	3.3V	DDC based control signal (clock) for HDMI device
HDMI_CTRL_DAT	I/O	3.3V	DDC based control signal (data) for HDMI device
HDMI_HPD#	I	3.3V	Hot plug detection signal

5.3.7 Video

The RK3399-Q7 does not feature LVDS as the CPU lacks this interface. Instead the Qseven LVDS pins are used for MIDI-DSI. These signals are electrical compatible but are not defined in the Qseven standard.

Q7 Pin	Function	Alternate Function
LVDS_A0_P	MIPI_TX0_D0P	EDP_TX0_P
LVDS_A0_N	MIPI_TX0_D0N	EDP_TX0_N
LVDS_A1_P	MIPI_TX0_D1P	EDP_TX1_P
LVDS_A1_N	MIPI_TX0_D1N	EDP_TX1_N
LVDS_A2_P	MIPI_TX0_D2P	EDP_TX2_P
LVDS_A2_N	MIPI_TX0_D2N	EDP_TX2_N
LVDS_A3_P	MIPI_TX0_D3P	EDP_TX3_P
LVDS_A3_N	MIPI_TX0_D3N	EDP_TX3_N
LVDS_A_CLK_P	MIPI_TX0_CLKP	EDP_AUX_P
LVDS_A_CLK_N	MIPI_TX0_CLKN	EDP_AUX_N
LVDS_B0_P	MIPI_TX1/RX1_D0P	
LVDS_B0_N	MIPI_TX1/RX1_D0N	
LVDS_B1_P	MIPI_TX1/RX1_D1P	
LVDS_B1_N	MIPI_TX1/RX1_D1N	
LVDS_B2_P	MIPI_TX1/RX1_D2P	
LVDS_B2_N	MIPI_TX1/RX1_D2N	
LVDS_B3_P	MIPI_TX1/RX1_D3P	
LVDS_B3_N	MIPI_TX1/RX1_D3N	
LVDS_B_CLK_P	MIPI_TX1/RX1_CLKP	
LVDS_B_CLK_N	MIPI_TX1/RX1_CLKN	

The LVDS A pins are muxed between MIPI_TX0 and eDP on the Module. The active function is selected with a GPIO pin.

Function	CPU Pin	Linux GPIO #
LVDS A Mux	GPIO2_A2	34

LVDS A Mux	Function
0	MIPI
1	eDP

5.3.8 GPIO

Signal	Type	Signal Level	Description
GPIO[0-7]	I/O	3.3V	General purpose inputs/outputs 0 to 7

5.3.9 CAN

Signal	Type	Signal Level	Description
CAN0_TX	O	3.3V	CAN (Controller Area Network) TX output for CAN Bus channel 0
CAN0_RX	I	3.3V	CAN (Controller Area Network) RX input for CAN Bus channel 0

5.3.10 SPI

Signal	Type	Signal Level	Description
SPI_MOSI	O	3.3V	Master serial output/Slave serial input signal
SPI_MISO	I	3.3V	Master serial input/Slave serial output signal
SPI_SCK	O	3.3V	SPI clock output
SPI_CS0#	O	3.3V	SPI chip select 0 output
SPI_CS1#	O	3.3V	SPI chip select 1 output (used when two devices are connected)

5.3.11 UART

Signal	Type	Signal Level	Description
UART0_TX	O	3.3V	Serial data transmit
UART0_RX	I	3.3V	Serial data receive
UART0_CTS#	I	3.3V	Handshake signal: ready to send data
UART0_RTS#	O	3.3V	Handshake signal: ready to receive data

5.3.12 Misc

Signal	Type	Signal Level	Description
WDTRIG#	I	3.3V	Watchdog trigger signal
WDOUT	O	3.3V	Watchdog event indicator
SMB_CLK GP1_I2C_CLK	O	3.3V	Clock line of System Management Bus. Alternate function I2C Bus clock line
SMB_DAT GP1_I2C_DAT	I/O	3.3V	Data line of System Management Bus. Alternate function I2C Bus data line
SMB_ALERT#	I	3.3V	System Management Bus Alert input
SPKR GP_PWM_OUT2	O	3.3V	PC speaker (buzzer) output. Alternate function general purpose PWM output
BIOS_DISABLE# /BOOT_ALT#	I	3.3V	Disables the onboard bootloader and uses the one the SD card instead. If no bootloader is available on the SD card it falls back to USB recovery mode
GP_1-Wire_Bus	I/O	3.3V	General Purpose 1-Wire bus interface
THRM#	I	3.3V	Thermal Alarm active low signal generated by the external hardware to indicate an over temperature situation. This signal can be used to initiate thermal throttling
THRMTRIP#	O	3.3V	Thermal Trip indicates an overheating condition of the processor. If 'THRMTRIP#' goes active the system immediately transitions to the S5 State (Soft Off)
FAN_PWMOUT /GP_PWM_OUT1	O	3.3V	PWM output for fan speed control. Alternate function general purpose PWM output. Function based on microcontroller firmware
FAN_TACHOIN /GP_TIMER_IN	I	3.3V	Fan tachometer input. Alternate function general purpose timer input. Function based on microcontroller firmware

5.3.13 Power Management

Signal	Type	Signal Level	Description
RSTBTN#	I	3.3V	Reset button input. An active low signal resets the module
BATLOW#	I	3.3V	Battery low input
WAKE#	I	3.3V	External system wake event. An active low signal wakes the module from a sleep state
SUS_S3#	O	3.3V	Indicated that the system is in suspend to ram (S3)
SUS_S5#	O	3.3V	Indicated that the system is in soft-off state (S5)
SLP_BTN#	I	3.3V	Sleep button. Signals the system with an falling edge to transition into sleep or wake from a sleep state
LID_BTN#	I	3.3V	LID button. Low active signal to detect a LID switch to transition into sleep or wake from a sleep state

5.3.14 Power

Signal	Nominal Input	Description
VCC	5V	Main supply for the module
VCC_RTC	3V	Backup supply for the RTC. If not used it can be left unconnected

5.4 On-board Devices

5.4.1 Power-Manager

The Rockchip RK808 is connected to the CPU via RSB and an interrupt line:

RK808 Pin	Function	CPU Pin
19	SCL	I2C0_SCL_u (ball N30)
18	SDA	I2C0_SDA_u (ball M26)
49	IRQ	GPIO1_C6 (ball L25)

5.4.2 DDR3

- 4GB RAM of DDR3-1600 (2 independent channels, each 32-bit wide)

5.4.3 eMMC

- eMMC connected through the 8-bit wide SDIO interface EMMC_D on the CPU.

Signal	CPU Pin	Linux GPIO #
RESET	GPIO0_A5	5

5.4.4 NOR Flash

- 32 MiB serial NOR flash
- Connected to the CPU via SPI1:

Signal	CPU Pin
CLK	GPIO_B1 (ball P28)
MOSI	GPIO_B0 (ball R31)
MISO	GPI1_A7 (ball P27)
CS	GPIO_B2 (ball P29)

5.4.5 Companion Controller

The on-board microcontroller provides additional features to the CPU, exposed via I2C and USB. It emulates standard ICs and does not need custom drivers in Linux.

Feature	CPU Connection	Emulated IC	Qseven Pins
RTC	I2C	ISL1208	none
Temperature sensor and fan controller	I2C	AMC6821	FAN_TACHOIN, FAN_PWMOUT
CAN	USB	UCAN	CAN0_TX, CAN0_RX

The STM32-microcontroller can be flashed from the CPU by taking it into DFU mode (USB recovery). Pull BOOT0 low and cycle reset (GPIOs listed below). The microcontroller will appear as a new USB device in Linux.

Function	CPU Pin	Linux GPIO #
NRST	GPIO1_D0	56
BOOT0	GPIO2_B4	76

5.4.6 Ethernet PHY

The Micrel KSZ9031RNX is connected to the CPU via RGMII and MDIO. Further connections are shown below.

PHY signal	Connected to	Linux GPIO #
RESET	CPU pin GPIO3_C0	112
MDIO	CPU pin GPIO3_B5	107
MDC	CPU pin GPIO3_B0	102
LED1	Qseven GBE_LINK1000 and GBE_LINK100 and GBE_LINK (tied together)	
LED2	Qseven GBE_ACT	

5.4.7 Test points

Test point	Connected to
TP1	STM32 USART2 TX
TP2	STM32 USART2 RX
TP3	GND

5.5 USB

The RK3399 CPU has two USB 3.0 SuperSpeed controllers. The Genesys Logic, Inc. GL3523 USB hub provides two additional USB 3.0 super-speed ports.

The routing of Qseven signals to CPU and/or hub port is shown below. The *Linux Port #* column shows the identifier that is used in Linux `dmesg` output. The format is: "usb BUS#-1.HUBPORT#"

Qseven Port #	Speed	Connected to	Linux Port #	Notes
USB_P0	USB 2.0 Hi-Speed	Hub	usb 7-1.1	
USB_P1	USB 2.0 Hi-Speed	CPU	usb 5-1	OTG Port
USB_P2	USB 2.0 Hi-Speed	Hub	usb 7-1.2	
USB_P3	USB 2.0 Hi-Speed	Hub	usb 7-1.3	
USB_SSTX0 / USB_SSRX0	USB 3.0 SuperSpeed	Hub	usb 8-1.1	Use together with USB_P0
USB_SSTX1 / USB_SSRX1	USB 3.0 SuperSpeed	CPU	usb 6-1	Use together with USB_P1
USB_SSTX2 / USB_SSRX2	USB 3.0 SuperSpeed	Hub	usb 8-1.2	Use together with USB_P2

The `lsusb -t` command shows the USB topology in a tree view and is highly recommended. It's output is discussed below, for a RK3399-Q7 module without additional devices connected:

`Bus 07` and `Bus 08` are connected to the GL3523 hub. The CAN controller is connected to `Port 4` on the hub:

```
lsusb -t
/:  Bus 08.Port 1: Dev 1, Class=root_hub, Driver=xhci-hcd/1p, 5000M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 5000M
/:  Bus 07.Port 1: Dev 1, Class=root_hub, Driver=xhci-hcd/1p, 480M
    |__ Port 1: Dev 2, If 0, Class=Hub, Driver=hub/4p, 480M
        |__ Port 4: Dev 3, If 0, Class=, Driver=uCAN, 12M
```

Linux `Bus 05` and `Bus 06` are routed directly to the Qseven ports `USB_P1` and `USB_SSTX1 / USB_SSRX1`:

```
/:  Bus 06.Port 1: Dev 1, Class=root_hub, Driver=xhci-hcd/1p, 5000M
/:  Bus 05.Port 1: Dev 1, Class=root_hub, Driver=xhci-hcd/1p, 480M
```

Additional internal USB 2.0 busses, currently unused:

```
/:  Bus 04.Port 1: Dev 1, Class=root_hub, Driver=ohci-platform/1p, 12M
/:  Bus 03.Port 1: Dev 1, Class=root_hub, Driver=ohci-platform/1p, 12M
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=ehci-platform/1p, 480M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-platform/1p, 480M
```

The USB hub can be held in reset, if required. This disables all USB ports connected to the hub. The reset signal routing is shown below:

Hub signal	CPU Pin	Linux GPIO #
USBHUB_RESETn	GPIO4_A3	131

5.6 Using Qseven Signals as GPIO

Most Qseven signals can be reused as general purpose pin. Following table shows the mapping and the possible direction between the edge connector and the SoC.

Qseven Pin	Signal	CPU Pin	Linux GPIO #	Direction
16	SUS_S5#	GPIO1_A1	33	Output
17	WAKE#	GPIO0_B1	9	Input
18	SUS_S3#	GPIO1_A0	32	Output
19	GPO0	GPIO2_B3	75	Output
21	SLP_BTN#	GPIO0_B3	11	Input
22	LID_BTN#	GPIO0_A4	4	Input
27	BATLOW#	GPIO0_B2	10	Input
42	SDIO_CLK#	GPIO4_B4	140	Bidirectional
43	SDIO_CD#	GPIO0_A7	7	Input
44	SDIO_LED	GPIO1_A2	34	Output
45	SDIO_CMD	GPIO4_B5	141	Bidirectional
46	SDIO_WP	GPIO0_B5	13	Input
47	SDIO_PWR#	GPIO1_C1	49	Output
48	SDIO_DAT1	GPIO4_B1	137	Bidirectional
49	SDIO_DAT0	GPIO4_B0	136	Bidirectional
50	SDIO_DAT3	GPIO4_B3	139	Bidirectional
51	SDIO_DAT2	GPIO4_B2	138	Bidirectional
56	USB_OTG_PEN	GPIO0_A2	2	Output
59	I2S_WS	GPIO3_D2	122	Output
60	SMB_CLK / GP1_I2C_CLK	GPIO2_A1	65	Bidirectional
61	I2S_RST#	GPIO4_A5	133	Output
62	SMB_DAT / GP1_I2C_DAT	GPIO2_A0	64	Bidirectional
63	I2S_CLK	GPIO3_D0	120	Output
64	SMB_ALERT#	GPIO0_B4	12	Input
65	I2S_SDI	GPIO3_D3	123	Input
66	GP0_I2C_CLK	GPIO1_B4	44	Bidirectional
67	I2S_SDO	GPIO3_D7	127	Output
68	GP0_I2C_DAT	GPIO1_B3	43	Bidirectional
69	THRM#	GPIO0_A3	3	Input
71	THRMTRIP#	GPIO1_A3	35	Output

Continued on next page

Table 2 – continued from previous page

Qseven Pin	Signal	CPU Pin	Linux GPIO #	Direction
111	LVDS_PPEN	GPIO4_D6	158	Output
112	LVDS_BLEN	GPIO1_C7	55	Bidirectional
123	LVDS_BLT_CTRL / GP_PWM_OUT0	GPIO4_C2	146	Output
124	GP_1-Wire_Bus	GPIO4_C7	151	Bidirectional
125	GP2_I2C_DAT / LVDS_DID_DAT	GPIO4_A1	129	Bidirectional
127	GP2_I2C_CLK / LVDS_DID_CLK	GPIO4_A2	130	Bidirectional
150	HDMI_CTRL_DAT	GPIO4_C0	144	Bidirectional
152	HDMI_CTRL_CLK	GPIO4_C1	145	Bidirectional
156	PCIE_WAKE#	GPIO2_D2	90	Input
158	PCIE_RST#	GPIO4_C6	150	Output
171	UART0_TX	GPIO2_C1	81	Output
172	UART0_RTS#	GPIO2_C3	83	Output
177	UART0_RX	GPIO2_C0	80	Input
178	UART0_CTS#	GPIO2_C2	82	Input
185	GPIO0	GPIO4_D4	156	Bidirectional
186	GPIO1	GPIO4_D1	153	Bidirectional
187	GPIO2	GPIO4_D0	152	Bidirectional
188	GPIO3	GPIO4_D5	157	Bidirectional
189	GPIO4	GPIO4_D2	154	Bidirectional
190	GPIO5	GPIO4_C4	156	Bidirectional
191	GPIO6	GPIO4_C3	147	Bidirectional
192	GPIO7	GPIO4_D3	155	Bidirectional
199	SPI_MOSI	GPIO2_C5	85	Output
200	SPI_CS0#	GPIO2_C7	87	Output
201	SPI_MISO	GPIO2_C4	84	Input
202	SPI_CS1#	GPIO2_D0	88	Output
203	SPI_SCK	GPIO2_C6	86	Output

5.7 Electrical Specification

5.7.1 Power Supply

The power supply requirements are listed in the table below and are identical to the Qseven specification.

Rail	Description	Nominal voltage	Tolerance
VCC	Main power supply	5V	4.75 … 5.25V
VCC_RTC	Backup battery	3V	2.4 … 3.3V

5.8 Mechanical Specification

5.8.1 Module Dimensions

The mechanical dimensions of the module are shown below.

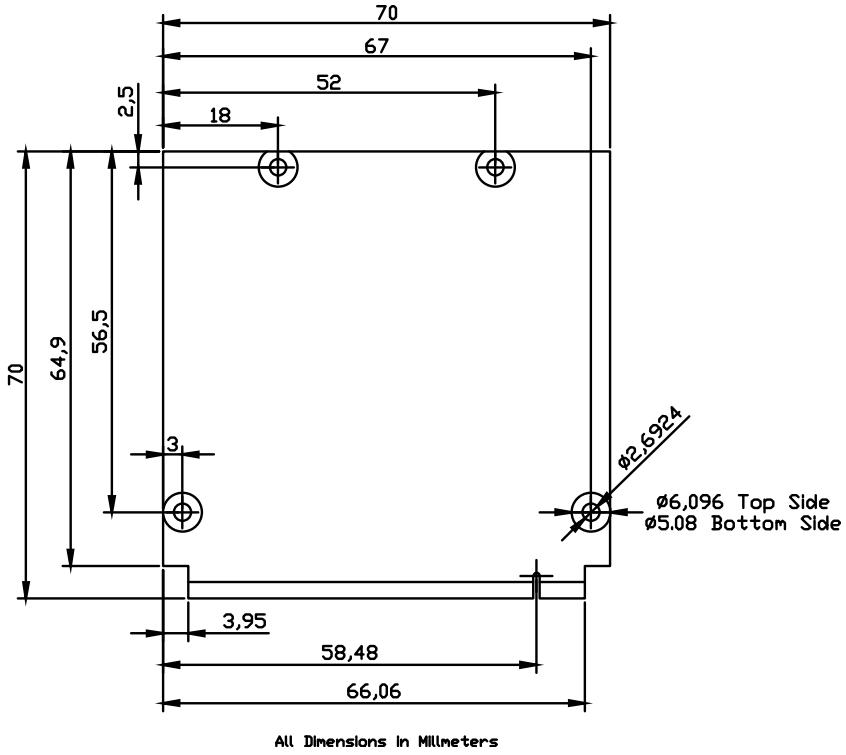


Fig. 1: Module dimensions (all values in mm)

5.8.2 Baseboard Dimensions

The mechanical dimensions of the baseboard are conform with the form factor for Mini-ITX and it can be mounted in a standard Mini-ITX PC Case.

6 Revision History

Date	Revision	Changes
Apr 11, 2018	v1.4	Add heatsink spacer instructions Clarify U-Boot deployment on SD card
Jan 22, 2018	v1.3	Improve image readability, add Linux GPIO# column
Oct 31, 2017	v1.2	Describe Linux USB bus number mapping
Oct 2, 2017	v1.1	Add I2C number mapping table Update SPL and BL31 file names
Jul 13, 2017	v1.0	Adapt debootstrap instructions for Debian 9 “Stretch” Add GPIO Pin Muxing section
Jul 10, 2017	v0.3	Complete software guide chapter Add video mux information
Jun 2, 2017	v0.2	Preliminary public release
May 30, 2017	v0.1	First internal release