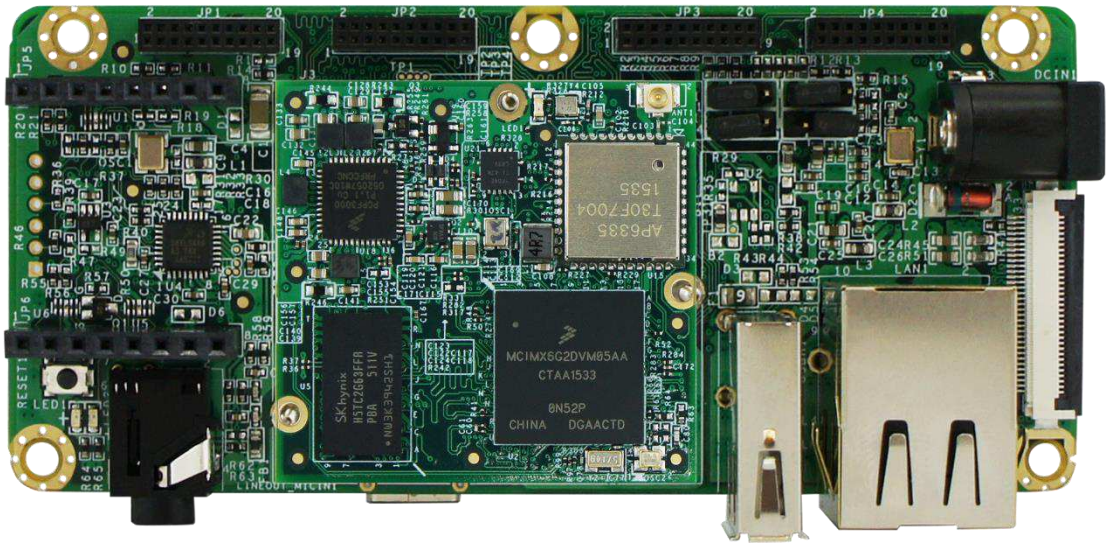
Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



HOBBITBOARD - NXP i.MX6Ultralite

(PART# HOBBIT6UL)

March 28, 2016

TABLE OF CONTENTS

1. Hobbitboard Product Overview

The Hobbitboard is a 2 board development board consisting of a System-on-Module and a carrier baseboard and optimized for the Internet-of-Things (IoT).

Figure 1 - Hobbitboard IC Identification and Overview

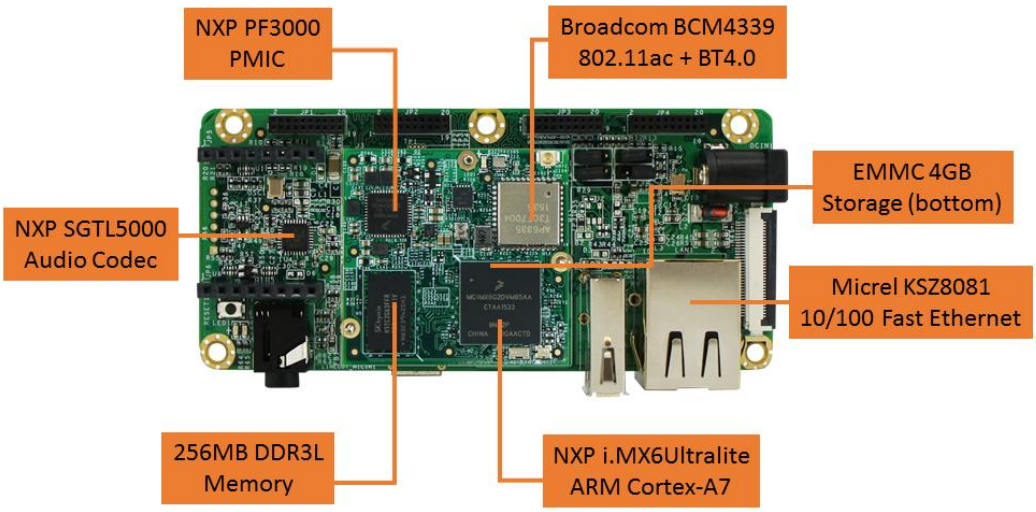
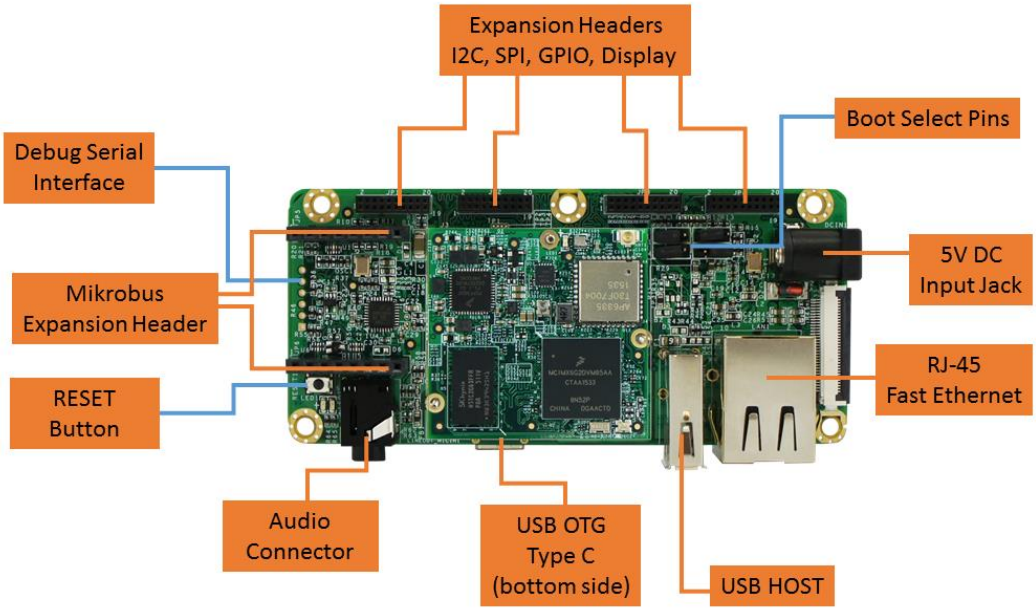


Figure 2 - Hobbitboard Connector Overview



1.1. Hobbitboard System-on-Module Overview

The Hobbitboard System-on-Module (PICO-IMX6UL-EMMC) has 3 Hirose high-speed 70 pin board-to-board connectors and integrates the NXP i.MX6Ultralite, Memory, eMMC, Power Management IC (PMIC) and WiFi / Bluetooth on the module.

Figure 3 - Hobbitboard System-on-Module

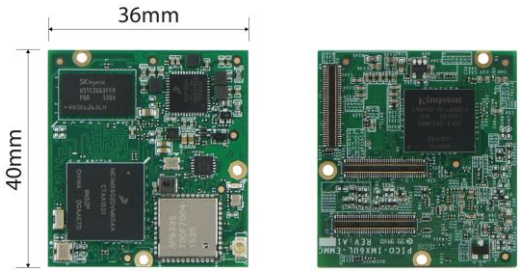
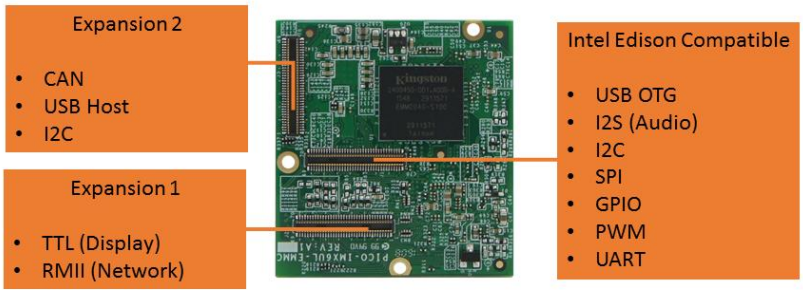


Figure 4 - Hobbitboard System-on-Module Signal Overview



1.2. Hobbitboard Carrier Baseboard Overview

The Hobbitboard Carrier Baseboard (PICO-HOBBIT-FL) has 3 Hirose high-speed 70 pin board-to-board connectors that connect to the System-on-Module and provides the real-world interfaces such as audio, network, USB and a large number of signals on the various pin headers.

Figure 5 - Hobbitboard Carrier Board

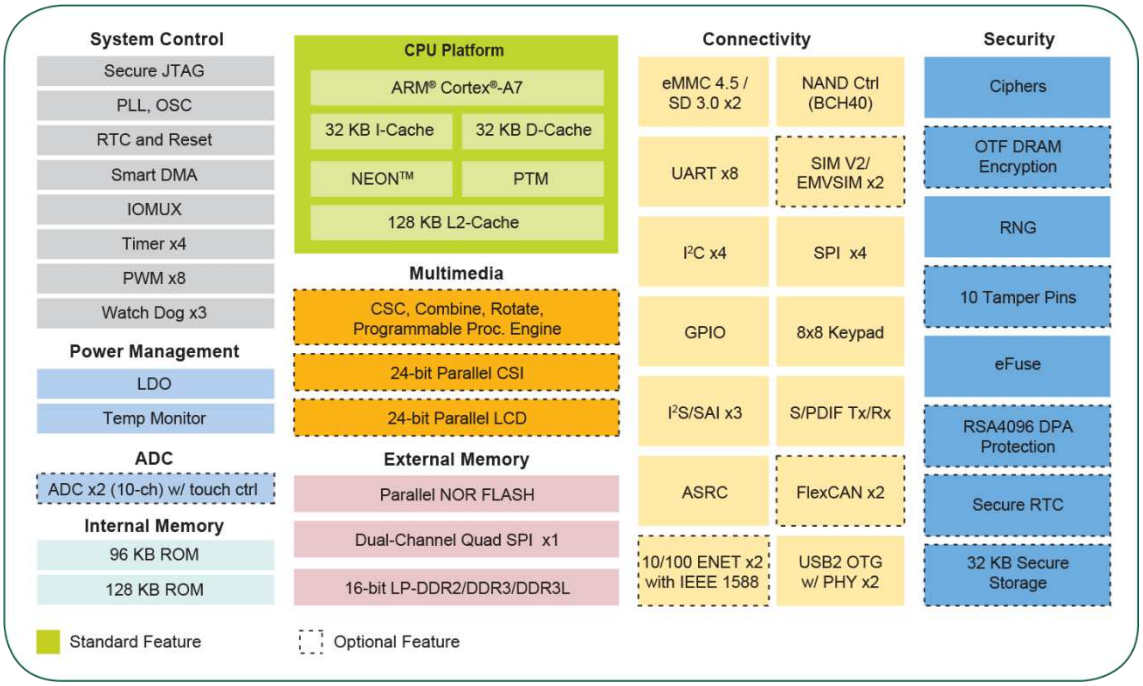
2. Core Components

2.1. NXP i.MX6Ultralite Cortex-A7 Processor

The i.MX 6UltraLite is an ultra-efficient processor family with featuring Freescale's advanced implementation of the single ARM Cortex®-A7 core, which operates at speeds of up to 528 MHz.

- The device is composed of the following major subsystems:
  - Single-core ARM Cortex-A7 MPCore™ Platform
  - 32 KBytes L1 Instruction Cache
  - 32 KBytes L1 Data Cache
  - Private Timer and Watchdog
  - TrustZone support
  - Cortex-A7 NEON MPE (Media Processing Engine) Co-processor
- PXP—PiXel Processing Pipeline for imagine resize, rotation, overlay and CSC. Offloading key pixel processing operations are required to support the LCD display applications.

Figure 6 - NXP i.MX6Ultralite Processor Blocks



2.2. NXP PF3000 Power Management IC (PMIC)

The Hobbitboard has on onboard NXP PF3000 power management integrated circuit (PMIC) that features a configurable architecture supporting the numerous outputs with various current ratings as well as programmable voltage and sequencing required by the components on the Hobbitboard Compute Module.

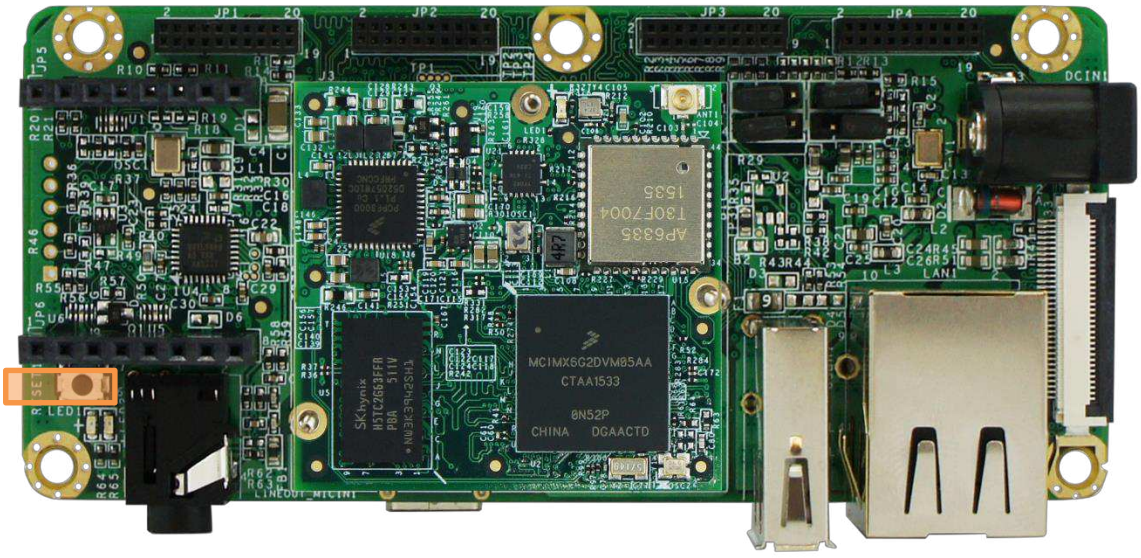
To perform a hard-reset of the Hobbitboard a software reset signal can be implemented.

CPU BALL	CPU PAD NAME	Pinmux (mode)	Signal	V	I/O	Description
E9	LCD_RESET	lcdif.RESET (mode0)	RESET	3V3	I	Connected to the PWRON signal of PMIC

The Hobbitboard Compute Module as well has an RESET signal routed on connector E1_36 this pin is connected to the RESET Button on the Hobbitboard. Simply pressing this button will RESET the Hobbitboard.

Connector	Signal	V	I/O	Description
E1_36	RESET	3V3	I	Connected to the PWRON signal of PMIC on the PICO Compute Module. Connected to the RESET Button on the Hobbitboard.

Figure 7 - Hobbitboard Reset Button Location



2.3. Memory (SKHynix)

The Hobbitboard integrates 256MB (2Gbit) Double Data Rate III (DDR3) Synchronous DRAM in a single (16 bit) channel configuration.

SK Hynix 2Gbit low power Double Data Rate III (DDR3L) Synchronous DRAM, ideally suited for the main memory applications which requires large memory density, high bandwidth and low power operation at 1.35V.

More information can be retrieved from SKHynix.

2.4. eMMC Storage (Kingston)

The Hobbitboard onboard 4GB eMMC device is connected on the SD1 pins of the i.MX6Ultralite processor in an 8 bit width configuration.

Kingston e•MMC™ products follow the JEDEC e•MMC™ 4.5 standard. It is an ideal universal storage solutions for many electronic devices, including smartphones, tablet PCs, PDAs, eBook readers, digital cameras, recorders, MP3, MP4 players, electronic learning products, digital TVs and set-top boxes. E•MMC™ encloses the MLC NAND and e•MMC™ controller inside as one JEDEC standard package, providing a standard interface to the host. The e•MMC™ controller directly manages NAND flash, including ECC, wear-leveling, IOPS optimization and read sensing.

The Kingston NAND Device is fully compatible with the JEDEC Standard Specification No.JESD84-B45.

More information can be retrieved from Kingston.

2.5. Broadcom BCM4339 WiFi/Bluetooth SiP Module

The Hobbitboard comes with an onboard WiFI/Bluetooth SIP module. The 802.11ac + BT SiP module is a small sized BGA mounted module that provides full function of 802.11ac and Bluetooth class 4.0 +HS

The small size & low profile physical design make it easier for system design to enable high performance wireless connectivity without space constrain. The low power consumption and excellent radio performance make it the best solution for OEM customers who require embedded 802.11ac Wi-Fi + Bluetooth features.

The SIP module is based on Broadcom BCM4339 chipset which is a WiFi + BT SOC. The Radio architecture & high integration MAC/BB chip provide excellent sensitivity with rich system performance.

In addition to WEP 64/128, WPA and TKIP, AES, CCX is supported to provide the latest security requirement on your network.

The SiP module is designed to operate with a single antenna for WiFi and Bluetooth to be connected to the u.FL connector (separate purchase, please purchase partnumber “WBANTENNAKIT”)

Figure 8 - Hobbitboard WiFi / Bluetooth Module and Antenna Connector Location

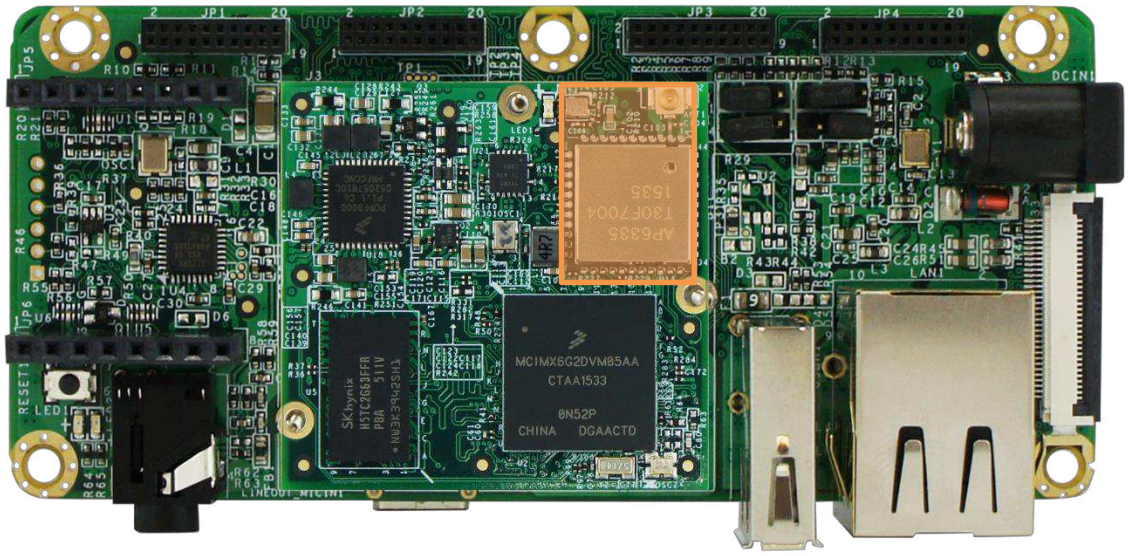


Table 1 - BCM4339 WiFi Signal Description

CPU BALL	PAD NAME	Pinmux (mode)	Signal	I/O	Description
D7	NAND_DATA00	usdhc2.DATA0 (mode1)	SDIO_D0	I/O	MMC/SDIO Data bit 0
B7	NAND_DATA01	usdhc2.DATA1 (mode1)	SDIO_D1	I/O	MMC/SDIO Data bit 1
A7	NAND_DATA02	usdhc2.DATA2 (mode1)	SDIO_D2	I/O	MMC/SDIO Data bit 2
D6	NAND_DATA03	usdhc2.DATA3 (mode1)	SDIO_D3	I/O	MMC/SDIO Data bit 3
C8	NAND_WE_B	usdhc2.CMD (mode1)	SDIO_CMD	I/O	MMC/SDIO Command
D8	NAND_RE_B	usdhc2.CLK (mode1)	SDIO_CLK	I/O	MMC/SDIO Clock
C6	NAND_DATA04	gpio4.IO[6] (mode5)	WL_HOST_WAKE	O	General purpose interface pin. This pin is high-impedance on power up and reset. Subsequently, it becomes an input or output through software control. This pin has a programmable weak pull-up/down.
A6	NAND_DATA06	gpio4.IO[8] (mode5)	WL_REG_ON	I	Used by PMU (OR-gated with BT_REG_ON) to power up or power down internal BCM4339 regulators used by the WLAN section. This pin is also a low-asserting reset for WLAN only (Bluetooth is not affected by this pin).

Table 2 - BCM4339 Bluetooth Signal Description

CPU BALL	PAD NAME	Pinmux (mode)	Signal	I/O	Description
M16	GPIO1_IO04	uart5.TX (mode8)	BT_UART_RXD	I	Bluetooth UART Serial Input. Serial data input for the HCI UART Interface
M17	GPIO1_IO05	uart5.RX (mode8)	BT_UART_TXD	O	Bluetooth UART Serial Output. Serial data output for the HCI UART Interface.
M15	GPIO1_IO09	uart5.CTS_B (mode8)	BT_UART_CTS	I/O	Bluetooth UART Clear to Send. Active-low clear-to-send signal for the HCI UART interface.
N17	GPIO1_IO08	uart5.RTS_B (mode8)	BT_UART_RTS	I/O	Bluetooth UART Request to Send. Active-low request-to-send signal for the HCI UART interface.
N14	JTAG_TRST_B	sai2.TX_DATA (mode2)	BT_PCM_IN	I	PCM data input
M14	JTAG_TCK	sai2.RX_DATA (mode2)	BT_PCM_OUT	O	PCM data output
N16	JTAG_TDI	sai2.TX_BCLK (mode2)	BT_PCM_CLK	I/O	PCM clock
N15	JTAG_TDO	sai2.TX_SYNC (mode2)	BT_PCB_SYNC	I/O	PCM sync signal
N9	SNVS_TAMPER8	gpio5.IO[8] (mode5)	BT_WAKE	I	Bluetooth device wake-up: Signal from the host to the module indicating that the host requires attention. • Asserted: Bluetooth device must wake-up or remain awake. • Deserted: Bluetooth device may sleep when sleep criteria are met. The polarity of this signal is software configurable and can be asserted high or low.
R6	SNVS_TAMPER9	gpio5.IO[9] (mode5)	BT_RST_N	I	Low asserting reset for BT core
B6	NAND_DATA05	gpio4.IO[7] (mode5)	BT_HOST_WAKE	O	Host UART wake up. Signal from the module to the host indicating that the module requires Attention. • Asserted: Host device must wake-up or remain awake. • Deserted: Host device may sleep when sleep criteria are met. The polarity of this signal is software configurable and can be asserted high or low.

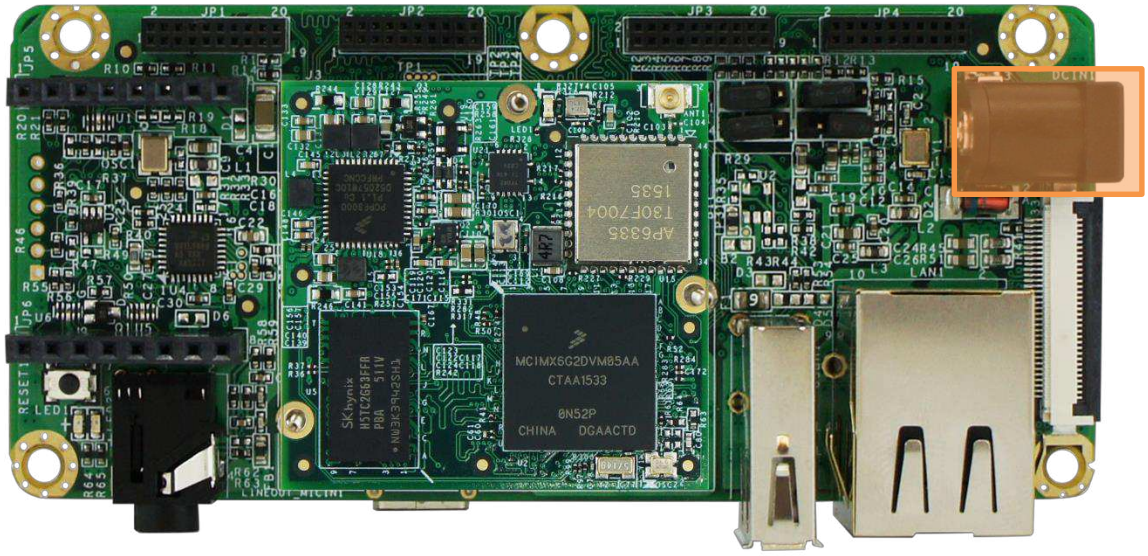
3. Hobbitboard Interfaces and Connectors

3.1. Power Input Connector

The Hobbitboard operates with a standard 5VDC power adaptor with at least 1.0 Ampere.

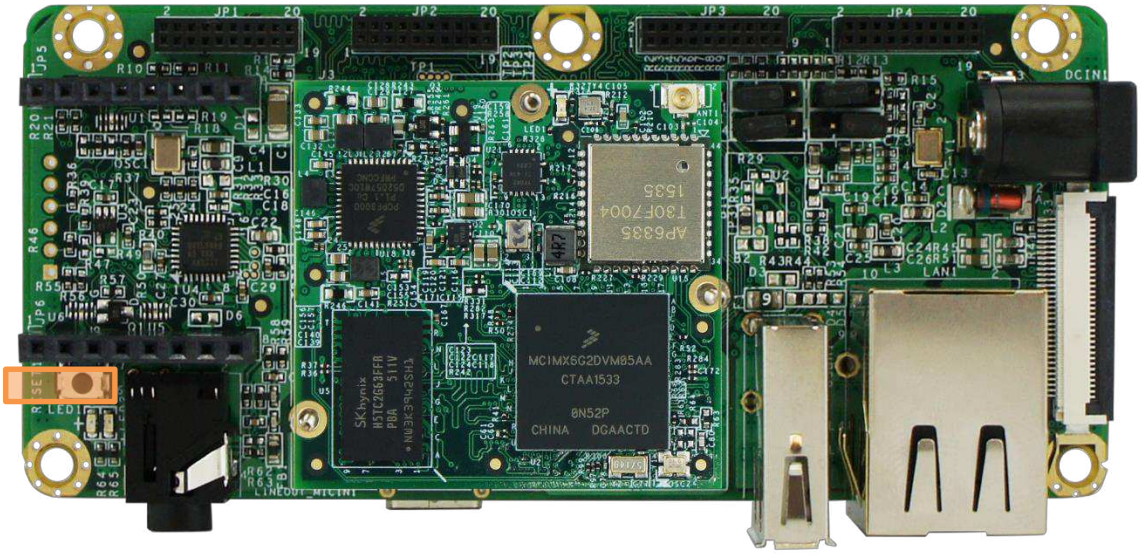
The connector dimensions are 5.5mm (barrel) / 2.1mm (tip).

Figure 9 - Hobbitboard Power Jack Location



3.2. System RESET Button

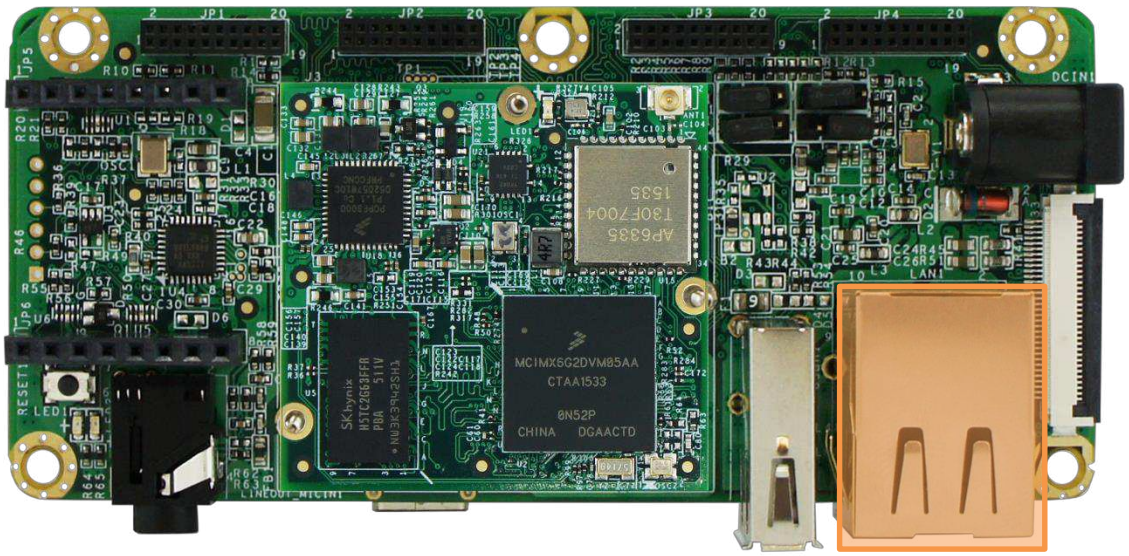
Figure 10 - Hobbitboard Reset Button Location



3.3. Fast Ethernet

The Hobbitboard features a 10/100 Mbit/s Fast Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with half- or full-duplex 10/100 Mbit/s Ethernet LANs.

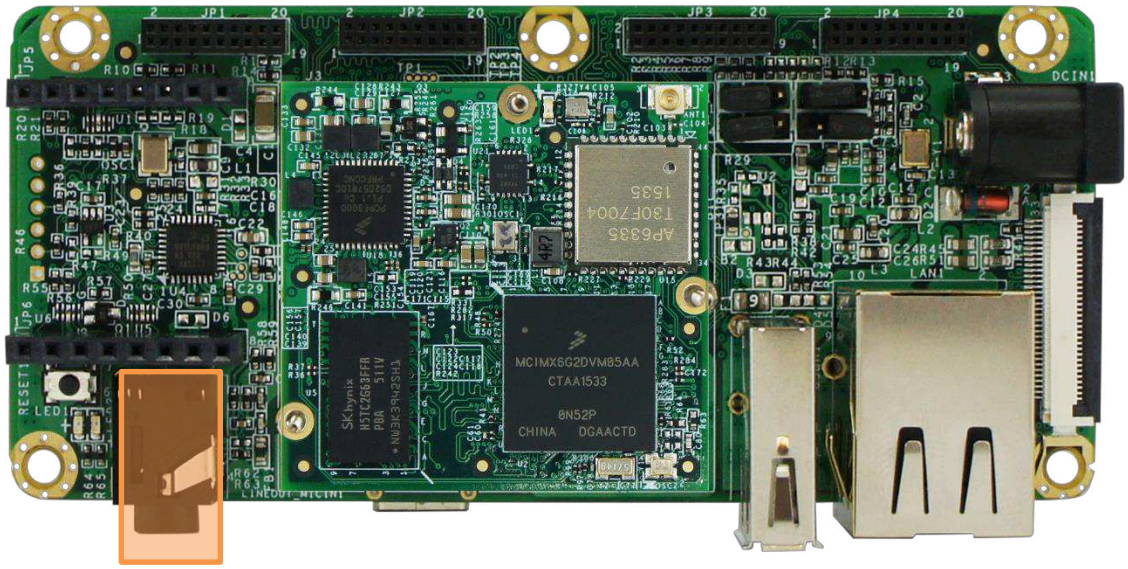
Figure 11 - Hobbitboard RJ-45 Network Connector Location



3.4. Audio Interface

The hobbitboards comes with an Audio jack which is compliant with the CTIA standard. A standard mobile phone headset will work.

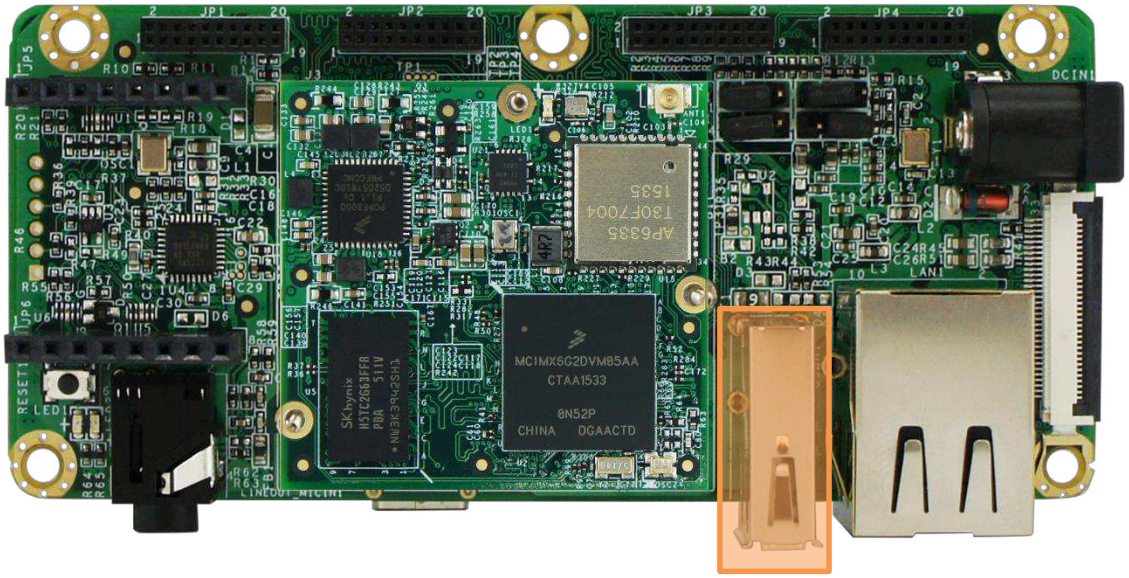
Figure 12 - Hobbitboard Audio Jack Location



3.5. Universal Serial Bus (USB) Host Interface

The Hobbitboard features a standard USB 2.0 Host Connector.

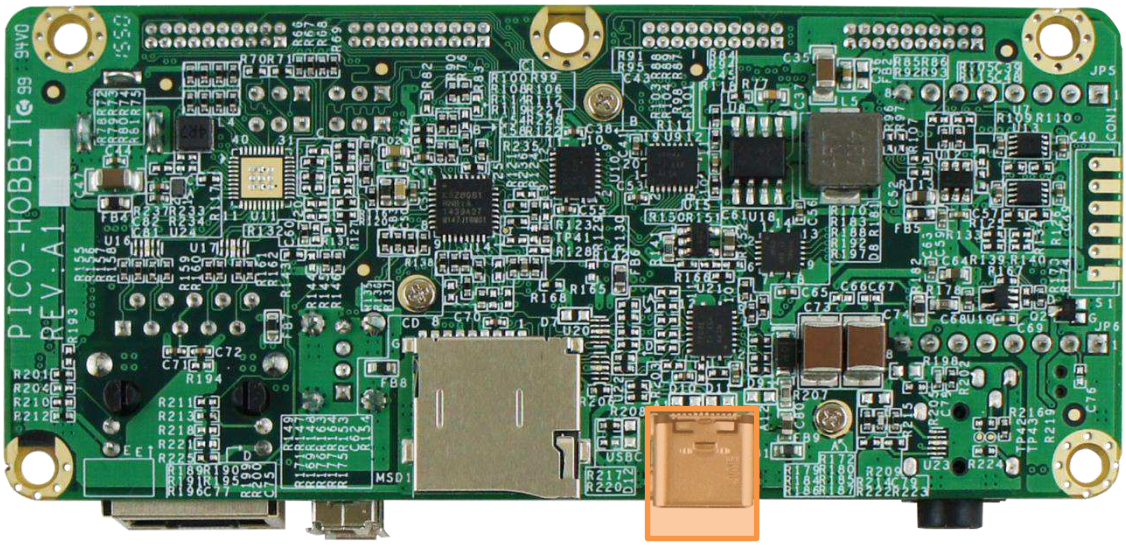
Figure 13 - Hobbitboard USB HOST Connector Location



3.6. Universal Serial Bus (USB) OTG Interface

The Hobbitboard incorporates a single USB Host/OTG controller.
The signals are routed to a USB Type-C connector.

Figure 14- Hobbitboard USB OTG Type-C Connector Location

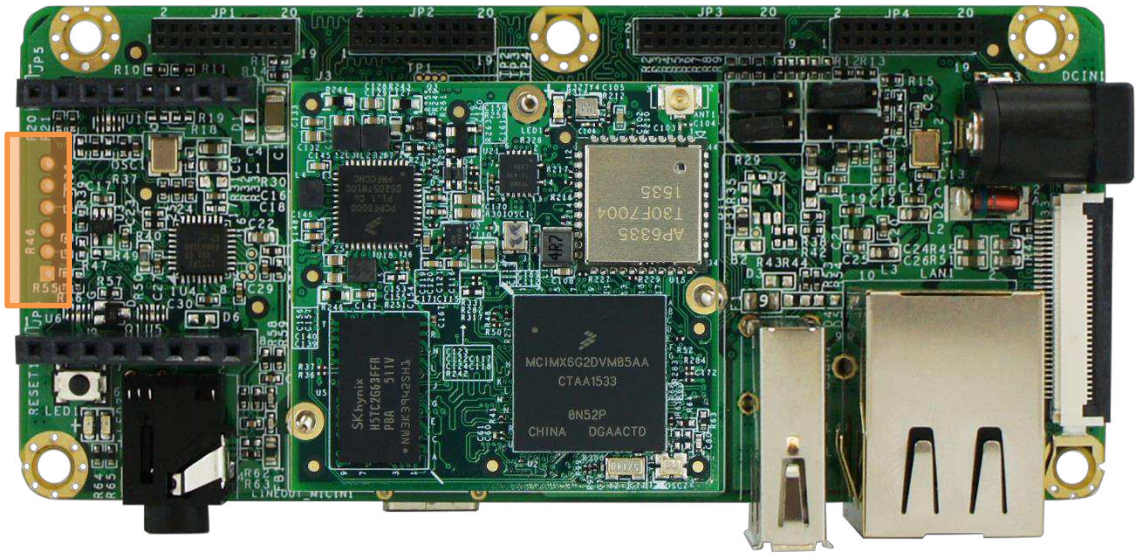


3.6. UART Debug Interface

The Hobbitboard Universal Asynchronous Receiver/Transmitter (UART) serial debug interface can be easily connected with an FTDI branded USB to Serial cable such as the FTDI **TTL-232R-3V3-WE**.

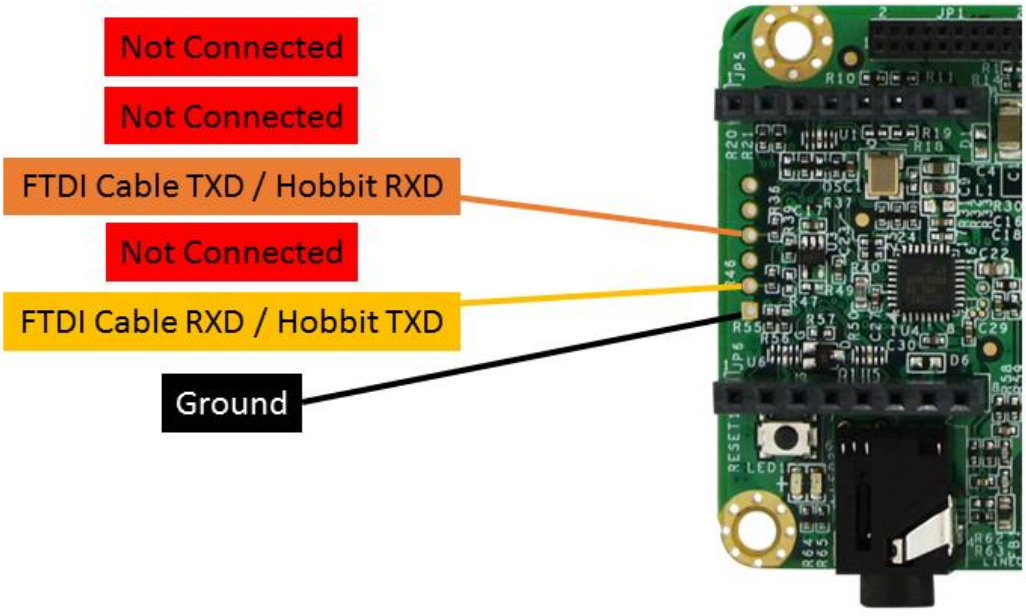
The debug interface can be found on the Hobbitboard at the following physical location and in software can be accessed over UART6.

Figure 15 - Hobbitboard Serial Debug Thru holes Location



While using the FTDI **TTL-232R-3V3-WE** only the brown/yellow/black cable should be connected to the following pins. All other wires of the cable can be left unconnected.

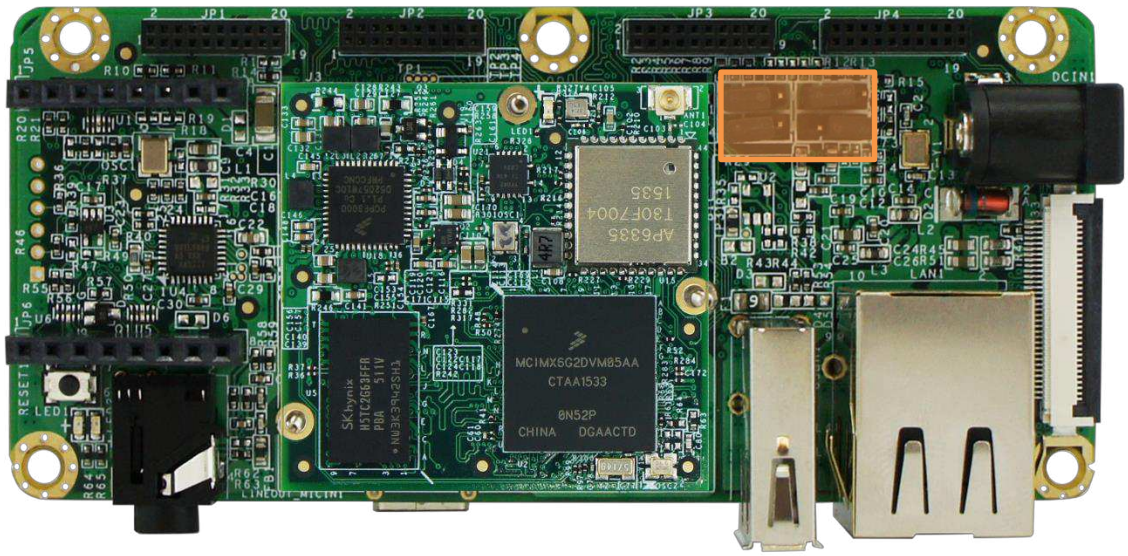
Figure 16 - Debug Cable Connection (FTDI TTL-232R-3V3-WE)

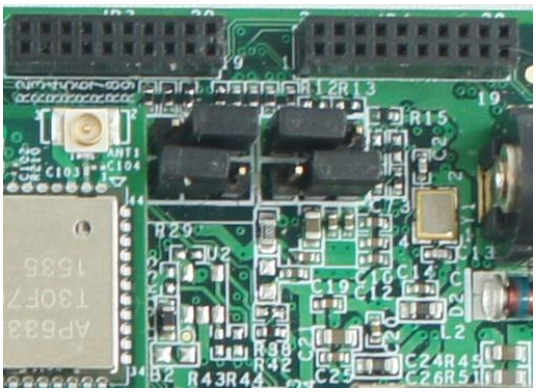
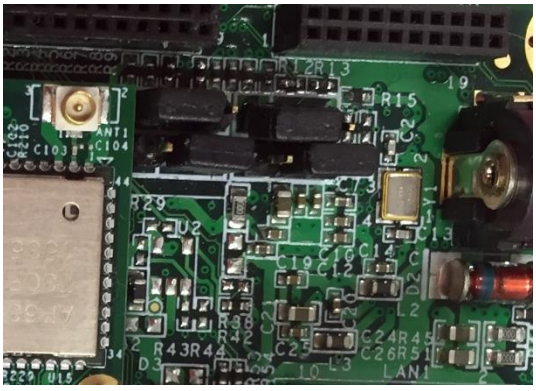


3.7. Serial Boot or eMMC Boot Control Pins

The Hobbitboard has a number of pins to override the default boot media (eMMC) and enter in Serial Boot Loader mode.

Figure 17 - Hobbitboard Boot Control Pins

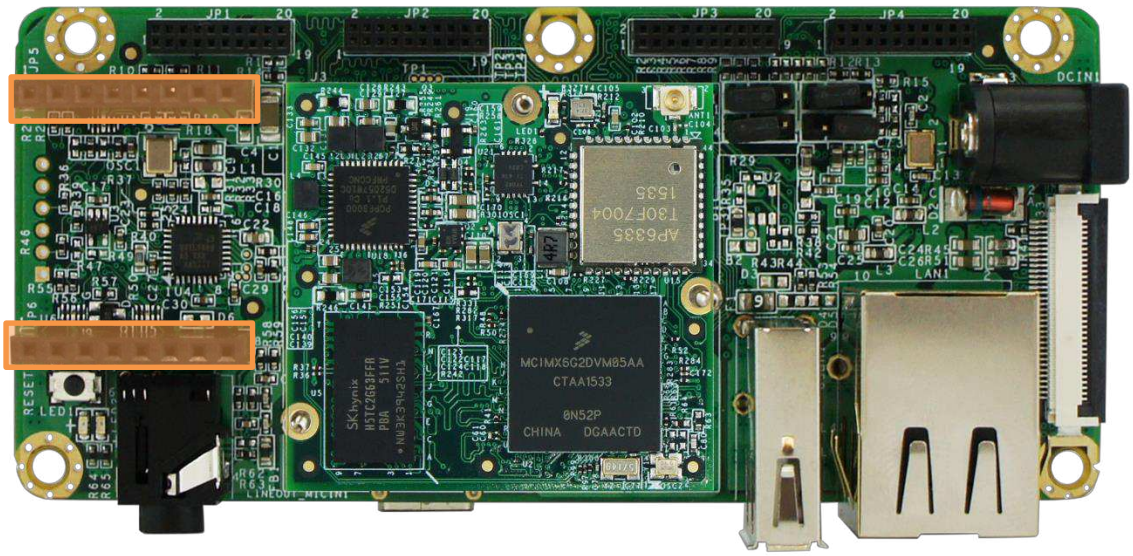


Boot from eMMC	Serial Boot Loader
	

3.8. Expansion Header Pins

The Hobbitboard has a number of expansion headers that can be used to connect displays, sensors, motors, and external devices.

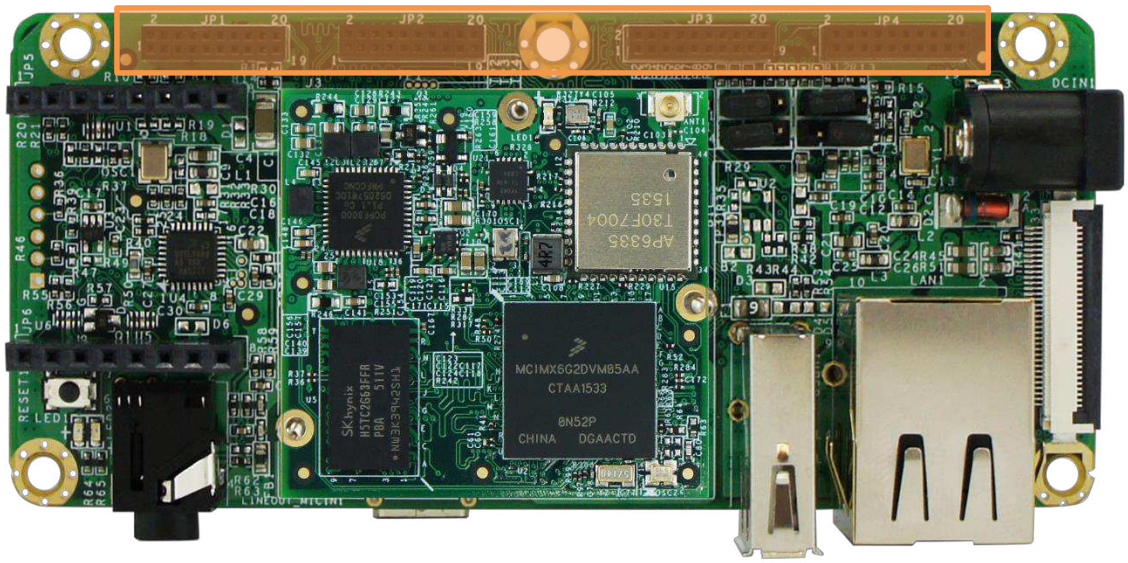
Figure 18 - Hobbitboard Mikrobus Header Location



PIN	i.MX6UL	CPU PAD NAME	Signal	V	I/O	Description
JP5_1	J14	UART1_RTS	USDHC1_CD_B	3V3	I/O	General Purpose Input Output with PWM control
JP5_2	F3	CSI_HSYNC	GPIO4_IO20	3V3	I/O	General Purpose Input Output
JP5_3	H16	UART3_RXD	UART3_RX	3V3	I	Universal Asynchronous Receive Transmit receive data signal
JP5_4	H17	UART3_TXD	UART3_TXD	3V3	O	Universal Asynchronous Receive Transmit transmit data signal
JP5_5	F17	UART5_TXD	I2C2_SCL	3V3	I/O	I²C bus clock line
JP5_6	G13	UART5_RXD	I2C2_SDA	3V3	I/O	I²C bus data line
JP5_7			5V Power	5V	P	5V Power
JP5_8			GND		P	Ground

PIN	i.MX6UL	CPU PAD NAME	Signal	V	I/O	Description
JP6_1			Analog Pin	3V3	ADC081C027 Analog signal accessible over I²C2	
JP6_2	PMIC	RESET	RESET	3V3	I	Reset power signal
JP6_3			NC			Not Connected
JP6_4	J16	UART2_RXD	ECSPI3_SCLK	3V3	O	Serial Peripheral Interface clock signal
JP6_5	J15	UART2_RTS	ECSPI3_MISO	3V3	I	Serial Peripheral Interface master input slave output signal
JP6_6	H14	UART2_CTS	ECSPI3_MOSI	3V3	O	Serial Peripheral Interface master output slave input signal
JP6_7			3V3 Power	3V3	P	3V3 Power
JP6_8			GND		P	Ground

Figure 19 - Hobbitboard Expansion Header Location



PIN	i.MX6UL	CPU PAD NAME	Signal	V	I/O	Description
JP1_1			3V3 Power	3V3	P	3V3 Power
JP1_2			3V3 Power	3V3	P	3V3 Power
JP1_3			NC			Not Connected
JP1_4			3V3 Power	3V3	P	3V3 Power
JP1_5			NC			Not Connected
JP1_6	B4	NAND_ALE	PWM3_OUT	3V3	O	LCD Backlight brightness Control
JP1_7			NC			Not Connected
JP1_8			NC			Not Connected
JP1_9			NC			Not Connected
JP1_10			NC			Not Connected
JP1_11			NC			Not Connected
JP1_12			NC			Not Connected
JP1_13			NC			Not Connected
JP1_14			NC			Not Connected
JP1_15			NC			Not Connected
JP1_16			3V3 Power	3V3	P	3V3 Power
JP1_17			NC			Not Connected
JP1_18			GND		P	Ground
JP1_19			GND		P	Ground
JP1_20			GND		P	Ground

PIN	i.MX6UL	CPU PAD NAME	Signal	V	I/O	Description
JP2_1	B9	LCD_DATA0	LCDIF_DATA0	3V3	O	LCD Pixel Data bit 0
JP2_2	E12	LCD_DATA10	LCDIF_DATA10	3V3	O	LCD Pixel Data bit 10
JP2_3	A9	LCD_DATA1	LCDIF_DATA1	3V3	O	LCD Pixel Data bit 1
JP2_4	D12	LCD_DATA11	LCDIF_DATA11	3V3	O	LCD Pixel Data bit 11
JP2_5	E10	LCD_DATA2	LCDIF_DATA2	3V3	O	LCD Pixel Data bit 2
JP2_6	C12	LCD_DATA12	LCDIF_DATA12	3V3	O	LCD Pixel Data bit 12
JP2_7	D10	LCD_DATA3	LCDIF_DATA3	3V3	O	LCD Pixel Data bit 3
JP2_8	B12	LCD_DATA13	LCDIF_DATA13	3V3	O	LCD Pixel Data bit 13
JP2_9	C10	LCD_DATA4	LCDIF_DATA4	3V3	O	LCD Pixel Data bit 4
JP2_10	A12	LCD_DATA14	LCDIF_DATA14	3V3	O	LCD Pixel Data bit 14
JP2_11	B10	LCD_DATA5	LCDIF_DATA5	3V3	O	LCD Pixel Data bit 5
JP2_12	D13	LCD_DATA15	LCDIF_DATA15	3V3	O	LCD Pixel Data bit 15
JP2_13	A10	LCD_DATA6	LCDIF_DATA6	3V3	O	LCD Pixel Data bit 6
JP2_14	C13	LCD_DATA16	LCDIF_DATA16	3V3	O	LCD Pixel Data bit 16
JP2_15	D11	LCD_DATA7	LCDIF_DATA7	3V3	O	LCD Pixel Data bit 7
JP2_16	B13	LCD_DATA17	LCDIF_DATA17	3V3	O	LCD Pixel Data bit 17
JP2_17	B11	LCD_DATA8	LCDIF_DATA8	3V3	O	LCD Pixel Data bit 8
JP2_18	A13	LCD_DATA18	LCDIF_DATA18	3V3	O	LCD Pixel Data bit 18
JP2_19	A11	LCD_DATA9	LCDIF_DATA9	3V3	O	LCD Pixel Data bit 9
JP2_20	D14	LCD_DATA19	LCDIF_DATA19	3V3	O	LCD Pixel Data bit 19

PIN	i.MX6UL	CPU PAD NAME	Signal	V	I/O	Description
JP3_1			3V3 Power	3V3	P	3V3 Power
JP3_2			3V3 Power	3V3	P	3V3 Power
JP3_3	N8	SNVS_TAMPER5	GPIO5_IO05	3V3	O	LCD backlight enable/disable
JP3_4	C14	LCD_DATA20	LCDIF_DATA20	3V3	O	LCD Pixel Data bit 20
JP3_5	P14	JTAG_TMS	GPIO1_IO11	3V3	O	LCD Voltage On
JP3_6	B14	LCD_DATA21	LCDIF_DATA21	3V3	O	LCD Pixel Data bit 21
JP3_7	G13	UART5_RXD	I2C2_SDA	3V3	I/O	I²C bus data line
JP3_8	A14	LCD_DATA22	LCDIF_DATA22	3V3	O	LCD Pixel Data bit 22
JP3_9	F17	UART5_TXD	I2C2_SCL	3V3	I/O	I²C bus clock line
JP3_10	B16	LCD_DATA23	LCDIF_DATA23	3V3	O	LCD Pixel Data bit 23
JP3_11	L17	GPIO1_IO03	I2C1_SDA	3V3	I/O	I²C bus data line
JP3_12	A8	LCD_CLK	LCDIF_CLK	3V3	O	LCD Pixel Clock
JP3_13	L14	GPIO1_IO02	I2C1_SCL	3V3	I/O	I²C bus clock line
JP3_14	C9	LCD_VSYNC	LCDIF_VSYNC	3V3	O	LCD Vertical Synchronization
JP3_15	K16	UART1_RXD	I2C3_SDA	1V8	I/O	I²C bus data line
JP3_16	D9	LCD_HSYNC	LCDIF_HSYNC	3V3	O	LCD Horizontal Synchronization
JP3_17	K14	UART1_TXD	I2C3_SCL	1V8	I/O	I²C bus clock line
JP3_18	B8	LCD_ENABLE	LCDIF_ENABLE	3V3	O	LCD dot enable pin signal
JP3_19			GND		P	Ground
JP3_20			GND		P	Ground

PIN	i.MX6UL	CPU PAD NAME	Signal	V	I/O	Description
JP4_1			5V Power	5V	P	5V Power
JP4_2			5V Power	5V	P	5V Power
JP4_3			5V Power	5V	P	5V Power
JP4_4	F2	CSI_VSYNC	GPIO4_IO19	3V3	I/O	General Purpose Input Output
JP4_5			5V Power	5V	P	5V Power
JP4_6	E4	CSI_DATA00	GPIO4_IO21	3V3	I/O	General Purpose Input Output
JP4_7	H14	UART2_CTS	ECSPI3_MOSI	3V3	O	Serial Peripheral Interface master output slave input signal
JP4_8	E3	CSI_DATA01	GPIO4_IO22	3V3	I/O	General Purpose Input Output
JP4_9	J15	UART2_RTS	ECSPI3_MISO	3V3	I	Serial Peripheral Interface master input slave output signal
JP4_10	E2	CSI_DATA02	GPIO4_IO23	3V3	I/O	General Purpose Input Output
JP4_11	J16	UART2_RXD	ECSPI3_SCLK	3V3	O	Serial Peripheral Interface clock signal
JP4_12	E1	CSI_DATA03	GPIO4_IO24	3V3	I/O	General Purpose Input Output
JP4_13			NC			Not Connected
JP4_14	K15	UART1_CTS	GPIO4_IO18	3V3	I/O	General Purpose Input Output
JP4_15	J17	UART2_TXD	ECSPI3_SS0	3V3		Serial Peripheral Interface Chip Select 1 signal
JP4_16	F3	CSI_HSYNC	GPIO4_IO20	3V3	I/O	General Purpose Input Output
JP4_17			GND		P	Ground
JP4_18	P11	SNVS_TAMPER2	GPIO5_IO02	3V3	I/O	General Purpose Input Output
JP4_19			GND		P	Ground
JP4_20			GND		P	Ground

4. Booting up the Hobbitboard

4.1.1. Overview

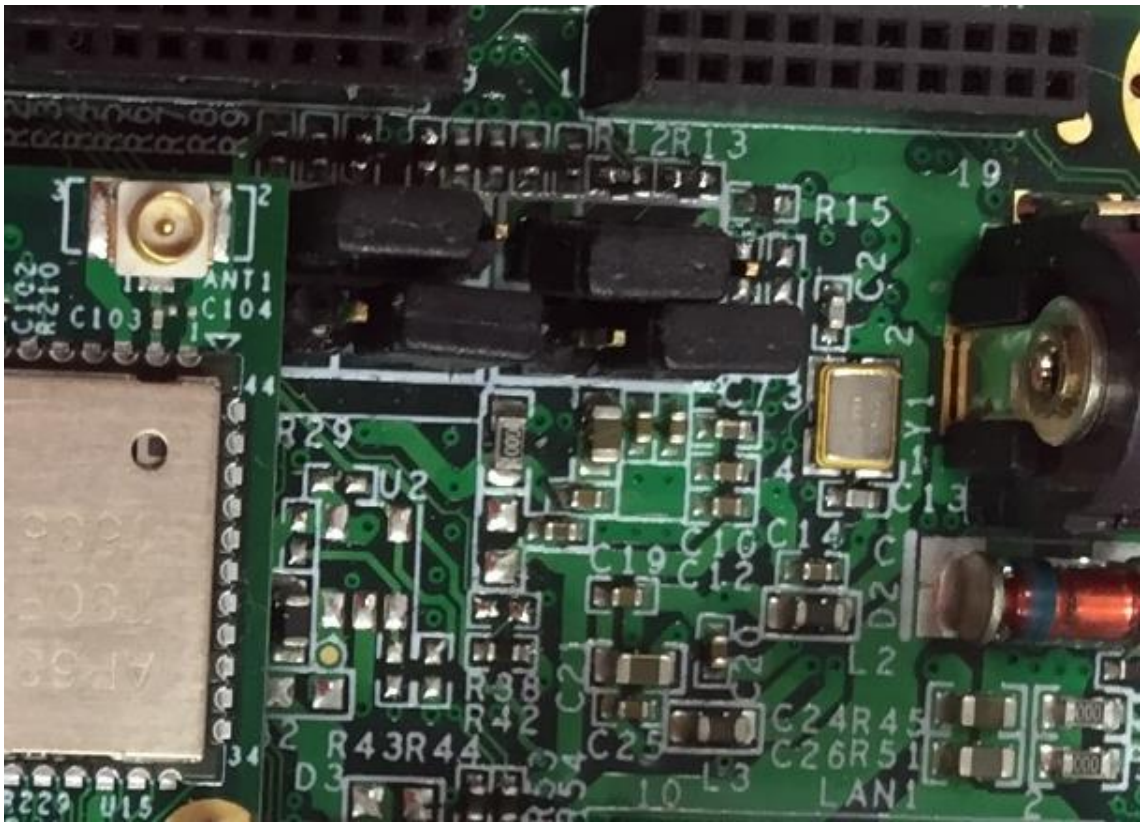
The boot mode for the Hobbitboard is controlled with jumpers on the baseboard. Normally, the board is intended to boot from the on-board eMMC flash, but sometimes the board needs to be booted from an external source. This can happen for example if the eMMC contains a faulty bootloader. This document guides how the on-board eMMC flash of a Hobbitboard can be flashed from a host PC.

4.1.2 i.MX6UL boot process details

When the boot jumpers are set to eMMC boot, the ROM code will attempt to boot from eMMC. If there is no bootable software present, the board will revert to "serial download mode". The name "serial download mode" is slighlty misleading, since the mode has grown past UART communication and nowadays is a way to access the board over an USB OTG port, or in the case of Hobbitboard, the USB type C port.

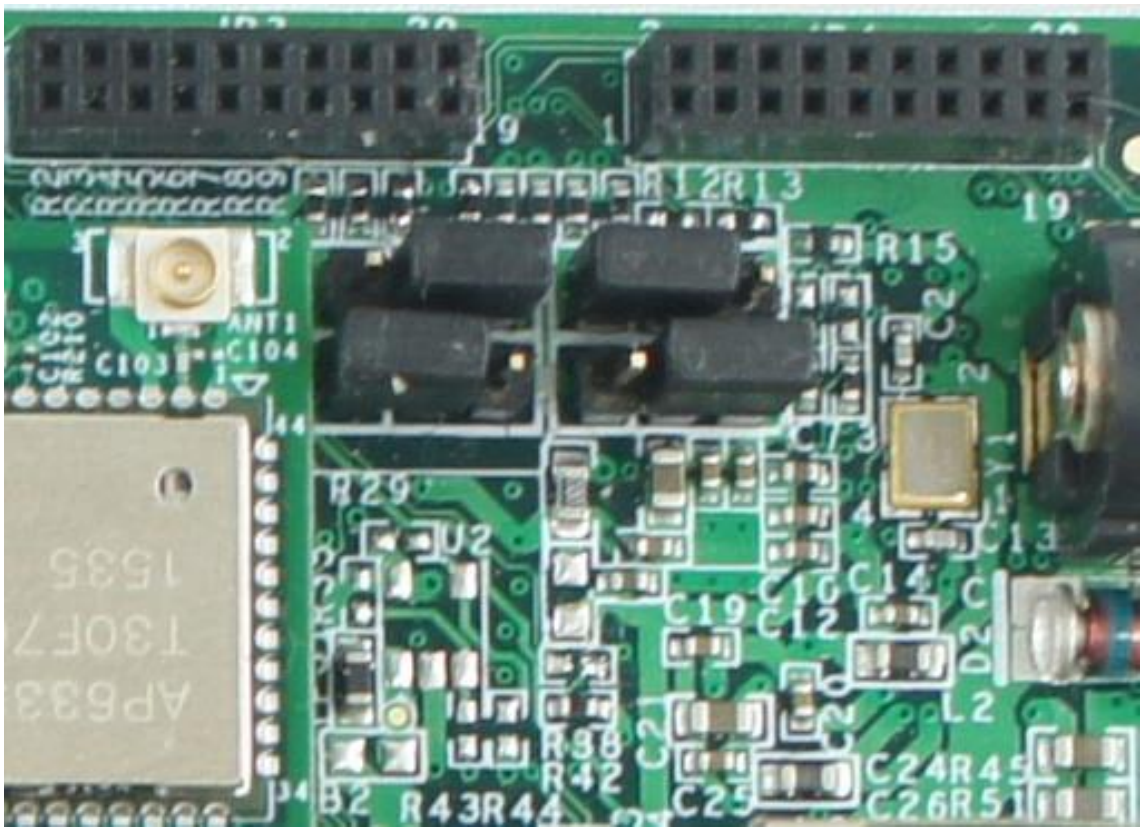
4.1.3 Changing Hobbitboard boot mode

To force the board into serial download mode using a Hobbitboard, change the boot mode jumpers J1 to 1-2 and jumper J4 to 2-3, as pictured below.



With this jumper setting, the board will not even attempt to boot from eMMC, but always expect it to be programmed over the USB type C connector.

To set the boot mode to eMMC, restore the jumpers J1 to 2-3 and jumper J4 to 1-2 (as pictured below).



In this mode the board will attempt to boot from eMMC, but can in some circumstances revert to serial download mode. This can happen for instance if the eMMC is not bootable. Note that "half-booting" software, like a u-boot that does not initialize memory timings for correct operation, will be interpreted as booting. More technically, if the ROM code finds the correct signature 1kB into the eMMC, it considers the board bootable from eMMC. To verify your Hobbitboard is in serial download mode, connect a USB cable between the host PC and the Hobbitboard. Power up the board. If a USB device with name "Freescale Semiconductor in Recovery Mode" appears, the board is in serial download mode.

4.1.4 Preparing a bootable software image

This section explains how software should be laid out inside the eMMC for a succesful boot. There are also steps how to prepare software to be programmed into the Hobbitboard eMMC.

4.1.4.1 Procedure overview

The procedure recommended here for programming the board is:

1. Prepare a file containing a bit-by-bit copy of what should be in eMMC (an "image" file). The layout of such files is described in sections 4.1.4.2 and 4.1.4.3.
2. Booting the board in serial download mode
3. Accessing the eMMC as a mass storage device (see chapters 4.2 and 4.3)
4. Copying the eMMC image file to the board
5. Booting the image

4.1.4.2 eMMC boot overview

This section describes how to use a Linux computer to prepare an image file of eMMC content. This step can be omitted, but is useful for creating image files which are easily distributable (since it is a all-in-one file).
A conventional Linux image consists of

- u-boot bootloader
- kernel
- devicetree
- root filesystem
- and sometimes, an initial ramdisk

The usual set up is that the kernel, device tree and the optional RAM disk is placed in a FAT partition as the first partition, and the OS filesystems occupies the remaining partitions. Linux systems usually occupy just the second partition, and in case a swap partition, also the third partition, but other operating systems, like Brillo, makes use of more advanced partition schemes.

Convenient ways to prepare an image using a Linux computer is using a block device (like a USB stick or an SD card) or a loopback device. It is also possible to access the eMMC as a mass storage device and manually place the data in the right place. See section 4.3.x.

4.1.4.3 Preparing an OS image

This section describes how to prepare the eMMC contents so the system can boot. This is intended for those wanting to prepare their own image, and not to use an image already provided by someone else.

The device is assumed to be a block device (like USB stick or SD card), but can beneficially be the eMMC itself (see chapter 4.2 and 4.3 on how to access the eMMC as a block device on a host PC). Here the block device is denoted /**dev**/**sdX;** care must be taken that the right device is used.

First, partition your device. Leave the first 1MB (or so) unpartitioned so that the first partition starts about 1MB into the device. The reason for this is that the u-boot bootloader needs to reside in unpartitioned space in the beginning of the block device.

The first partition is usually a FAT partition containing the Linux kernel as a zImage and the device tree blob (dtb) file. If an initial ramdisk is used, the initrd files can also reside in the FAT partition.

The second partition is usually the Linux root file system. Keep in mind when partitioning that the eMMC is limited in size to approximately 4GB, and not let your image become too large to fit.

Then copy your bootloader (u-boot) 1kB into the image. On a Linux host, the following command can be used. Remember to replace /dev/sdX with the appropriate device or image file.

 # dd if=u-boot.imx of=/dev/sdX bs=1k seek=1 conv=notrunc oflag=dsync

Thereafter format the additional partitions, with the expected file systems, copy the files there.

4.1.4.4 Creating the image file from a block device

As a last step create the image file from your block device. This can be done using the followin Linux commands: First list the partitions in the block device:

 # fdisk -lc /dev/sdX

The output looks something like:

Disk /dev/sdX: 3965 MB, 3965190144 bytes

122 heads, 62 sectors/track, 1023 cylinders

Units = cylinders of 7564 * 512 = 3872768 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x0005ffff

Device Boot	Start	End	Blocks	Id	System
/dev/sdX1	2	4	8192+	83	FAT12
/dev/sdX2	4	85	307200	83	Linux
/dev/sdX3	85	363	1048576	82	Linux swap

To extract the OS image from this issue the command

 # dd if=/dev/sdX of=image.img bs=3872768 count=85

The blocksize (bs= parameter) is taken from the line **"Units = cylinders of 7564 * 512 = 3872768 bytes"** and the count parameter is the end of the last non-swap partition, /dev/sdX2.

4.2. Programming PICO-IMX6UL-EMMC using a Windows host

This section guides on how to use a Windows 7 computer to access the eMMC on a Hobbitboard, and how to program an image file to the eMMC.
Tools needed:

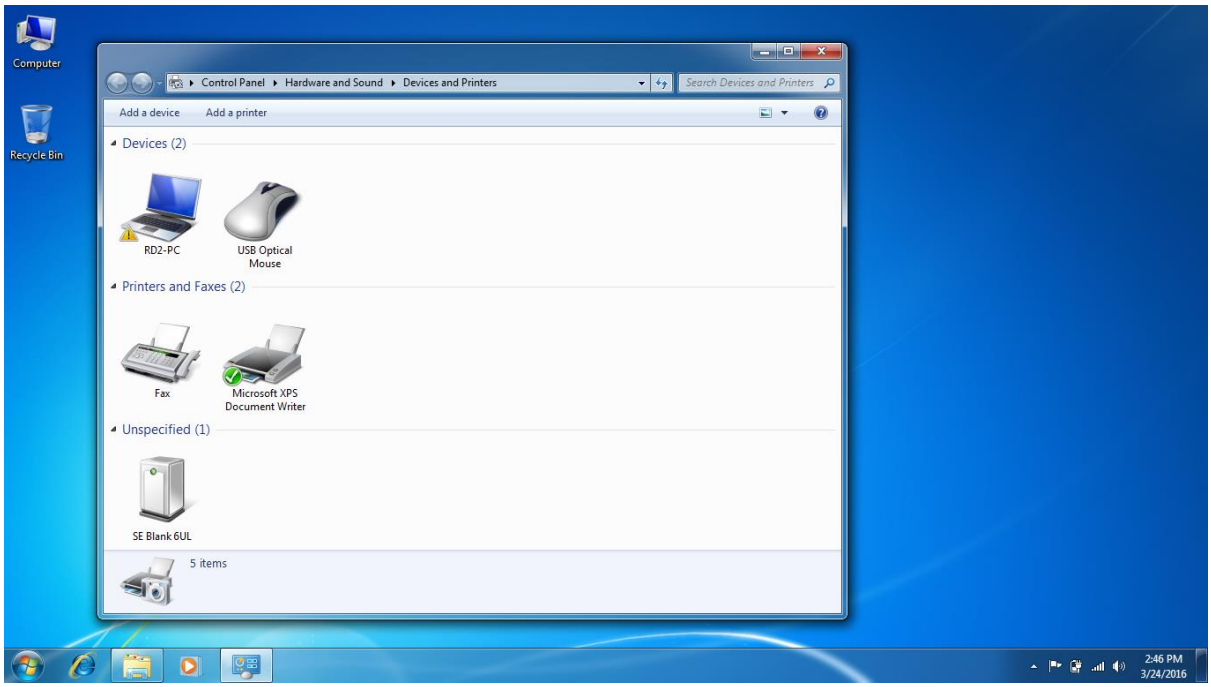
- sb_loader_imx6ul.exe (or similar)
- a "bootbomb" file that can be dropped on the board, enabling it to be accessible as a USB Mass Storage device
- Windiskimager or similar tool that allows raw writing of block devices.
- An eMMC image file to be programmed to the eMMC flash of the board.
- And optionally, a serial terminal emulator program

For convenience, there is a downloadable tool package containing the three first items at:

http://www.hobbitboard.org/downloads/hobbit/hobbitboard_tools-20160322.zip

4.2.1. Preparing the setup

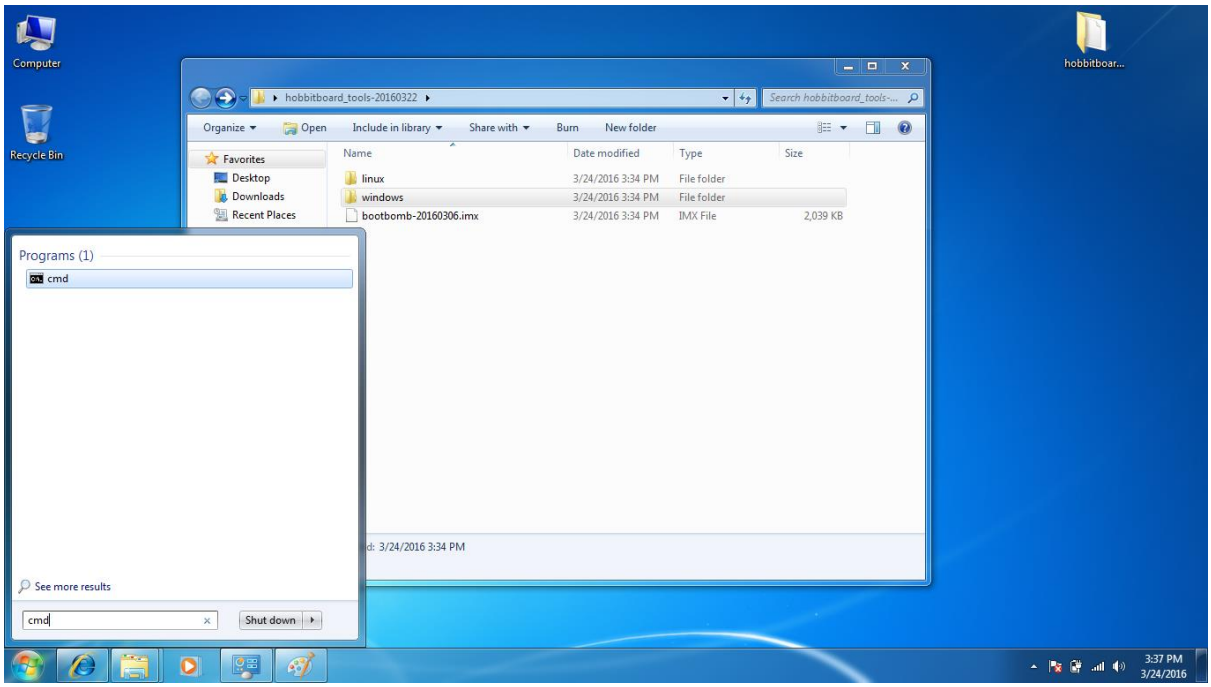
First attach a USB Type C peripheral cable to the board, and the other end to the host PC. Set the boot jumpers to serial download mode, power up the board, and verify that a "SE Blank 6UL" device appears (as below):



Next, download the tools package. Copy the folder inside the ZIP file to the Desktop.

4.2.2. Using sb_loader

Then start a command prompt by clicking on the start menu, and in the "Search programs and files" box enter "cmd" (see below):



In the command line interface, navigate to the Hobbitboard tools package and the sb_loader folder inside it by typing the following commands:

- cd Desktop
- cd hobbitboard_tools-20160322
- cd windows
- cd sb_loader

(see screenshot below)