BRTSYS_AN_093

PanL Hub Developers' Guide

Version 1.0

Issue Date: 18-12-2025

This document provides comprehensive instructions for understanding, configuring, and extending the capabilities of the PanL Hub44/80.

Use of BRTSys devices in life support and/or safety applications is entirely at the user’s risk, and the user agrees to defend, indemnify, and hold BRTSys harmless from any and all damages, claims, suits, or expense resulting from such use.

BRT Systems Pte Ltd (BRTSys)
1 Tai Seng Avenue, Tower A, #03-01, Singapore 536464
Tel: +65 6547 4827
Web Site: http://www.brtsys.com
Copyright © BRT Systems Pte Ltd

Table of Contents

Copyright © BRT Systems Pte Ltd

Copyright © BRT Systems Pte Ltd

1 Introduction

The PanLHub44/80 is a multi-port networked hub based on a Linux platform and can house various wireless connectivity modules such as Wi-Fi, ZigBee or RF433 Transmitter to connect to a vast array of wireless smart devices. The hub also allows wired RS-485 based devices to be connected to its RJ45 and RJ11 ports for data and power transmission via Ethernet cables. The Hub can be connected to a local network either through the 10/100T Ethernet Port or through the in-built Wi-Fi module if it's present in the hub.

Following are the main features of PanL Hub44/80:

- Quad-core 1.2GHz 64-bit quad-core ARM Cortex A53
- Built-in 1GB LPDDR2 RAM memory
- Built-in 4GB eMMC flash primary storage
- Linux Operating System
- 10/100BASE TX Ethernet Port
- RJ45 RS485 Ports for PanL Hub44/80
- RJ11 RS485 Ports only for PanL Hub44
- Real Time Clock (RTC)
- Built-in Temperature sensor
- Built-in Crypto Authentication
- Wi-Fi 802.11 b/g/n (optional)
- Zigbee (optional)
- RF433 Transmitter (optional)
- Power switch
- DC power Input: +24V / 2.5A (PanL Hub44), +24V / 5.0A (PanL Hub80)
- Operating temperature range: 0°C to +55°C
- FCC ID: 2ATZF-PH4404XXA (PanL Hub44),
  2ATZF-PH8004XXA (PanL Hub80)

1.1 Intended Audience

The PanL Hub Developer's guide is designed to assist developers in understanding, configuring, and extending the capabilities of the PanL Hub44/80. This document provides technical information on hardware components, software interfaces, and communication protocols that enable seamless integration with a wide range of connected devices.

1.2 FCC Compliance Statement

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) These devices may not cause harmful interference, and

(2) These devices must accept any interference received, including interference that may cause undesired operation.

NOTE: The equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If the equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Re-orient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

To maintain compliance with FCC's RF exposure guidelines, at least 20cm of separation distance between the device and the user's body must be always maintained.

FCC Radiation Exposure Statement

This device complies with FCC radiation exposure limits set forth for an uncontrolled environment and it also complies with Part 15 of the FCC RF Rules. This equipment must be installed and operated in accordance with the instructions provided and the antenna(s) used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter. End-users and installers must be provided with antenna installation instructions and consider removing the no-collocation statement.

Caution

Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Contains FCC ID:
2ATZF-PH4404XXA (PanL Hub44)
2ATZF-PH8004XXA (PanL Hub80)

2 Specifications

Platform	OS	Linux
	CPU	Quad Core 1.2GHz 64bit ARM Processor
	Chipset	Broadcom BCM2837
Memory	Internal Memory	1GB RAM, 4GB eMMC Flash
Wireless Connectivity	Wi-Fi	802.11 b/g/n (2.4GHz), up to 150 Mbps (Optional)
	ZigBee	802.15.4 (2.4GHz) (Optional)
	RF433	SRD 433.92MHz (Optional)
I/O Interface	Ethernet	1 x 10/100BASE TX
	RJ45	4 x Ports to support RS485 **(for PanL Hub44)** 8 x Ports to support RS485 **(for PanL Hub80)**
	RJ11	4 x Ports to support RS485 **(for PanL Hub44)**
Features	Temperature Sensor	Built-in
	Crypto Authentication	Built-in
	RTC	CR1225 Battery (Lasts ~ 3 Years)
	Reset Button	Push Switch
	Power ON/OFF Switch	Rocker Switch
	Power Indicator LED	Red LED
	Status Indicator LED	Health Status (A/B) - 2 x RGB LEDs
		RJ45/RJ11 Ports - 8 x RGB LEDs **(for PanL Hub44)** RJ45 Ports - 8 x RGB LEDs **(for PanL Hub80)**
Power	Input Voltage	24V DC (60W-AC-DC-Adapter 100~240 VAC to 24V DC **(for PanL Hub44)** 24V DC (120W-AC-DC-Adapter 100~240 VAC to 24V DC **(for PanL Hub80)**
	Power Connector	24V DC / 2.5A Jack **(for PanL Hub44)** 24V DC / 5A Jack **(for PanL Hub80)**
	Output Voltage	RJ45 - 4 x 24V DC @500mA; RJ11 - 4 x 5V DC 100mA **(for PanL Hub44)**
		8 x 24V DC @500mA **(for PanL Hub80)**
Physical Characteristics	Housing	Polycarbonate ABS
	Dimensions	129.44mm x 256.3mm x 35.0mm
	Weight	450 grams
Environmental Limits	Operating Temperature	0 to +55°C
	Storage Temperature	0 to +70°C
	Ambient Relative Humidity	20 to 85% (non-condensing)
Standards& Certifications	EMC (FCC/CE)	EN 55032:2015+AC:2016 Class B; CISPR 32:2015+C1: 2016 Class B; EN 55035:2017; FCC PART 15, Subpart B
	Radio Equipment Directive (RED)	EN 301 489-1 V2.2.0; EN 301 489-3 V2.1.1; EN 301 489-17 V3.2.0 **(for PanL Hub44)**
	Safety (LVD)	IEC 62368-1: 2014; IEC 62368-1: 2014 +A11:2017 **(for PanL Hub44)**
	RFID (FCC/CE)	EN 300 328 V2.1.1; EN 300 220-1 V3.1.1; EN 300 220-2 V3.1.1; EN 62311:2008; FCC PART 15, Subpart C (15.225) **(for PanL Hub44)**
Package Contents	Hardware Components	1 x PanL Hub44; 1 x 60W Power Adapter; 4 x RS485 PanL Terminators 1 x PanL Hub80; 1 x 120W Power Adapter; 8 x RS485 PanL Terminators
	Documentation	1 x Quick Start Guide

Table 1 - PanL Hub44/80 Specifications

Copyright © BRT Systems Pte Ltd

3 Hardware Setup and Description

3.1 PanL Hub44/80 Hardware Features

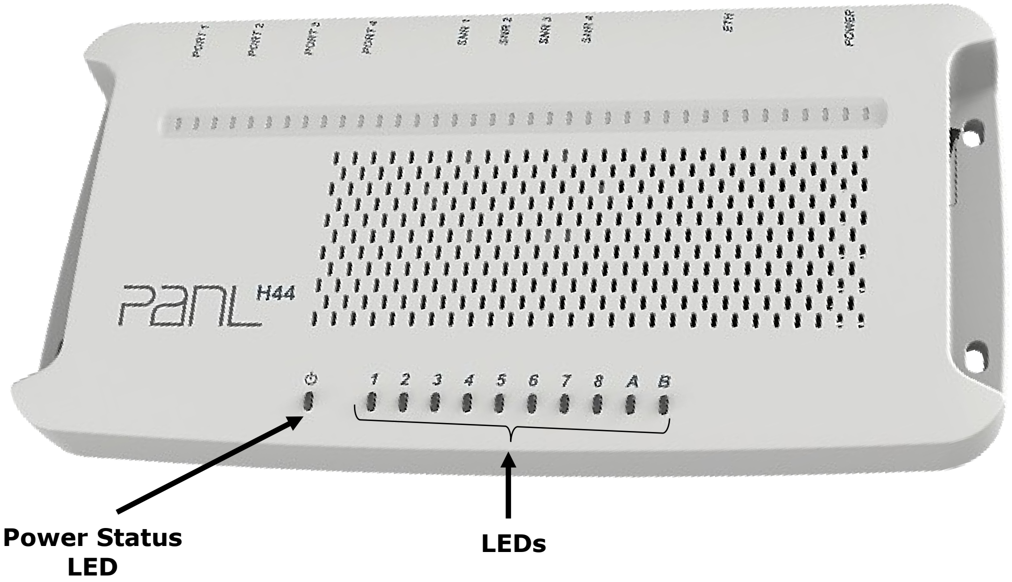


Figure 1 - PanL Hub44 Top View

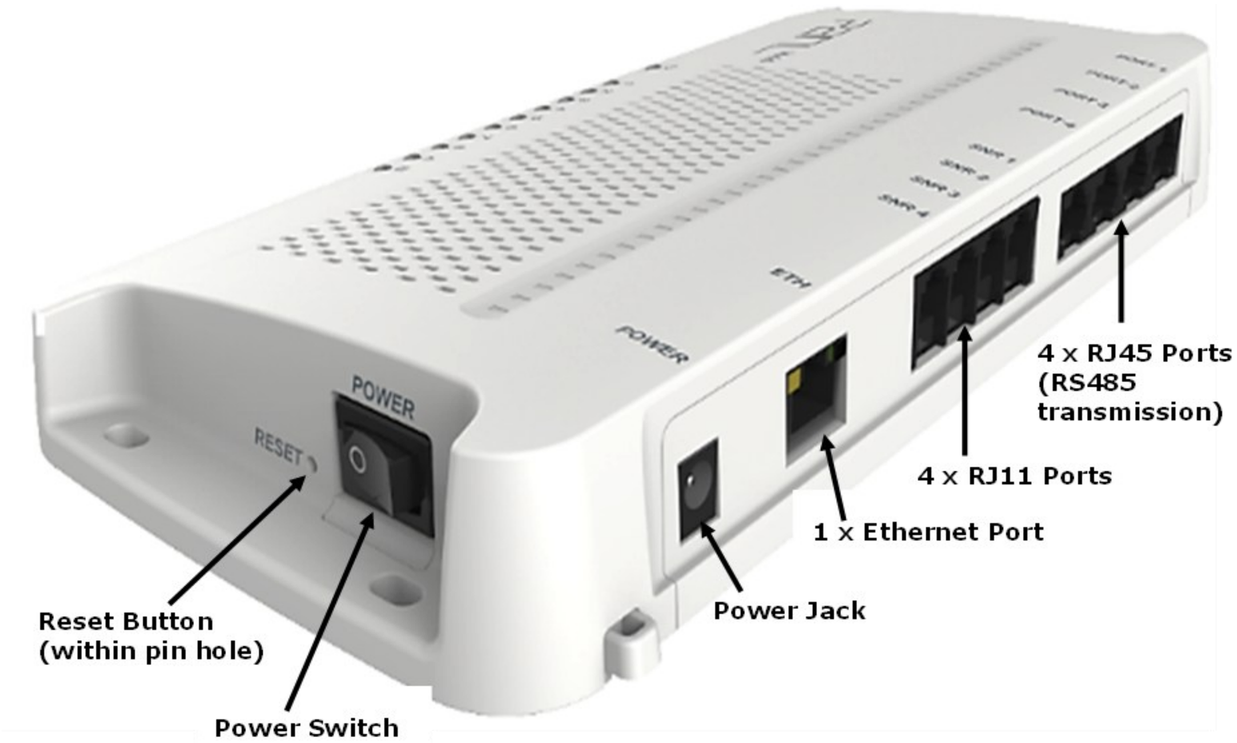


Figure 2 - PanL Hub44 Side View

Copyright © BRT Systems Pte Ltd

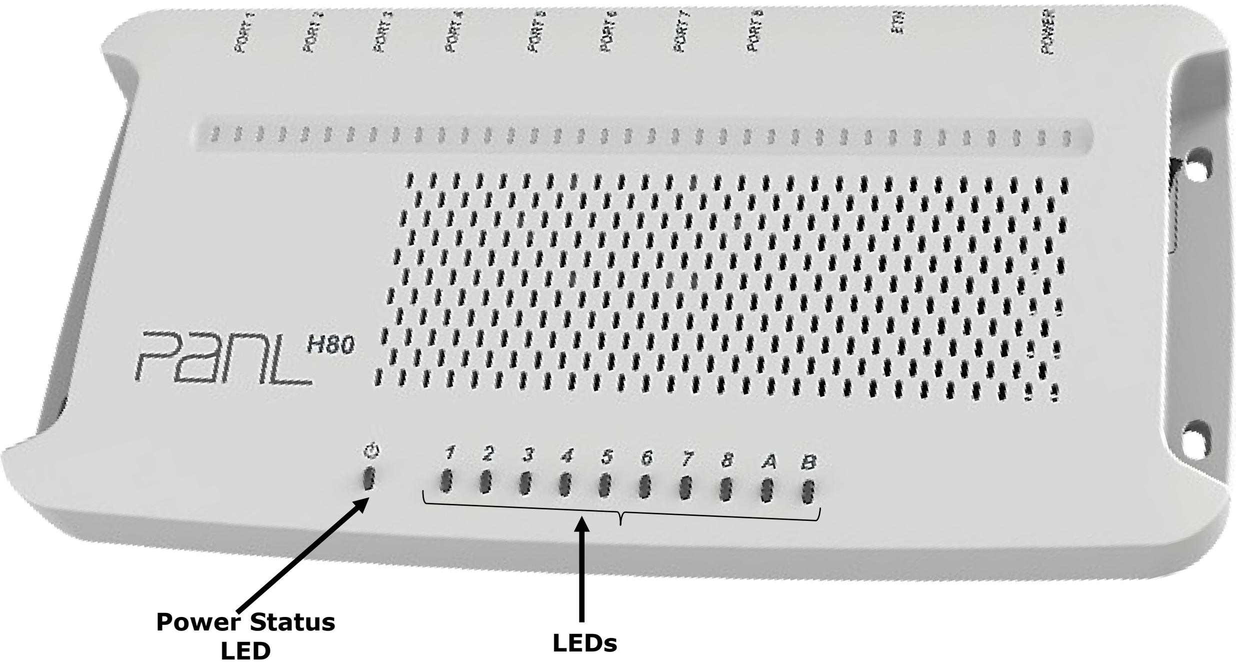


Figure 3 - PanL Hub80 Top View

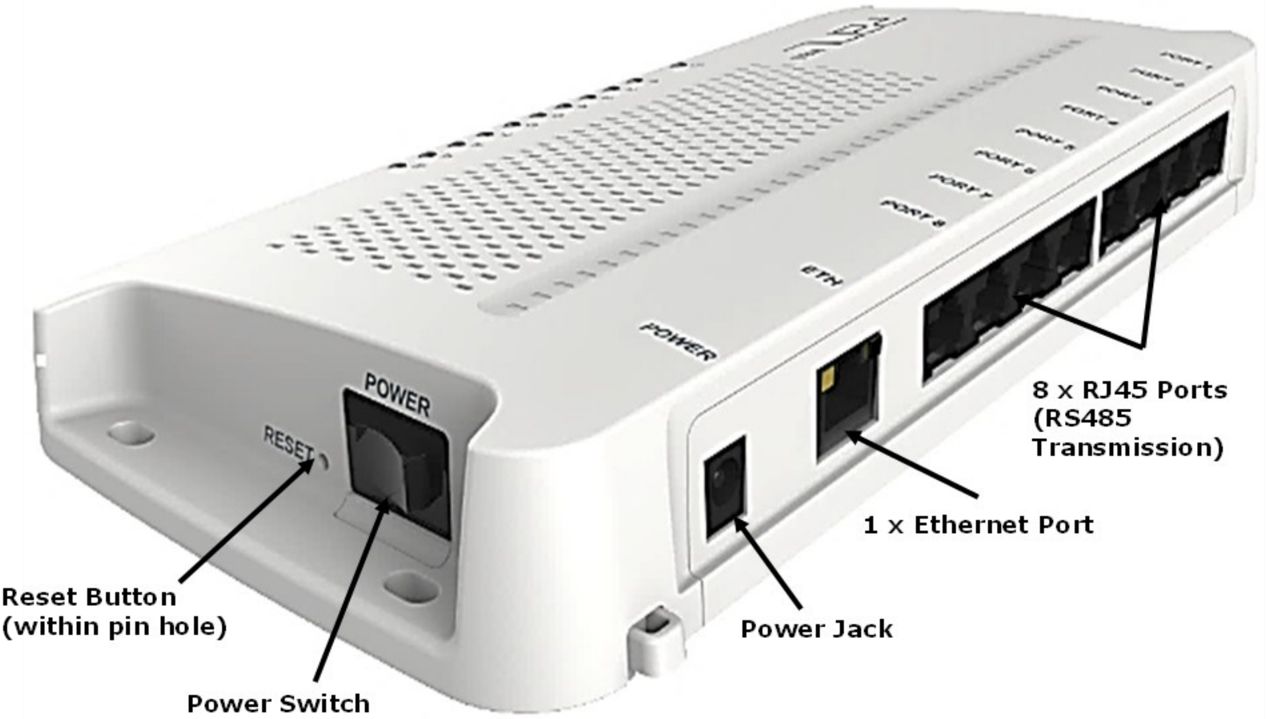


Figure 4 - PanL Hub80 Side View

3.1.1 Power Supply

3.1.1.1 PanL Hub44

PanL Hub44 is powered by a 24VDC/2.5A adaptor which is provided along with the package. It is highly recommended to use a powered adapter with the recommended power rating (such as the one included with the package) to ensure the hub operates within its specified parameters. Power input from the Power Jack and switch ON/OFF by using the Power Switch.

3.1.1.2 PanL Hub80

PanL Hub80 is powered by a 24VDC/5A adaptor which is provided along with the package. It is highly recommended to use a power adapter with the recommended power rating (such as the one included with the package) to ensure the hub operates within its specified parameters. Power input from the Power Jack and switch ON/OFF by using the Power Switch.

3.1.2 CPU System

Both PanL Hub44 and PanL Hub80 runs on Linux platforms and is powered by a Raspberry Pi Compute Module 3, which features a Quad Core 1.2GHz 64bit ARM Processor. A designated heat sink and DC5V fan provide CPU cooling system.

3.1.3 Wireless Connectivity

PanL Hub44 and PanL Hub80 comes with three types of connectivity modules: Wi-Fi, Zigbee and RF433.

3.1.3.1 Wi-Fi

The Wi-Fi module provides an alternative option to connect to a local network to communicate with other Wi-Fi enabled devices if Ethernet network cable wiring is not feasible.

Standard: IEEE 802.11 b/g/n Wi-Fi Module
Data Rate: Up to 150Mbps
Antenna: Integrated 2.4 GHz PCB Antenna

3.1.3.2 ZigBee

The ZigBee module is powered by the latest System-On-Chip (SOC) from Texas Instruments™, the CC2538 and it uses an on-board antenna. The CC2538 is an integrated platform for IEEE 802.15.4 ZigBee® applications.

The device integrates a low power 2.4 GHz transceiver, an MCU based on an ARM Cortex-M3 core (32 bit) and a hardware accelerator for the IEEE 802.15.4 MAC layer.

3.1.3.3 RF433 Transmitter

The RF433 module operates on 433.926MHz frequency, ASK/OOK modulation and up to 4.8kbps speed transmission rate.

3.1.4 Power Indicator

A red colour LED indicator to indicate the power status (ON/OFF) of the PanL Hub44 and PanL Hub80.

3.1.5 RGB LED Indicator

There is a total of 10 user-definable RGB LED indicators on the PanL Hub44 and PanL Hub80. Refer to Section 5.1.3 for instructions on configuring the LEDs.

3.1.6 Ethernet Port

Both PanL Hub44 and PanL Hub80 have an Ethernet port 10/100 BASE TX which can be used to connect the hub to a local network through a standard network CAT cable.

3.1.7 RJ45 Ports (RS485 Transmission and Power)

There are four RJ45 ports (PORT1 to PORT4) available on PanL Hub44 (Refer to Figure 2) which provides a communication and power channel to any devices with an RS485 interface and powered at +24V DC.

There are eight RJ45 ports (PORT1 to PORT8) available on PanL Hub80 (Refer to Figure 4) which provides a communication and power channel to any devices with an RS485 interface and powered at +24V DC.

Pin 3 connected to MCU IO with pull up resistor, the feature can be defined by user. A standard network CAT 8P8C cable can be used. The length of the RJ45 8P8C cables from the port should account for power drop and remain within the RS485 specification.

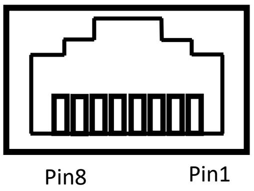


Figure 5 - RJ45 Port

Pin Number	1	2	3	4,5	6,7,8
Function	RS485 B/Z	RS485 A/Y	I/O	DC24V OUT	GND

Table 2 - RJ45 Port Pin Function

3.1.8 RJ11 Ports for PanL Hub44 (RS485 Transmission and Power)

There are four RJ11 ports (SNR1 to SNR4) available on the PanL Hub44 (Refer to Figure 6), providing both communication and power to any device with an RS485 interface and powered by 5V DC. A standard RJ11 4P4C cable can be used. The length of the cable from the sensor port should account for power drop and remain within the RS485 specification.

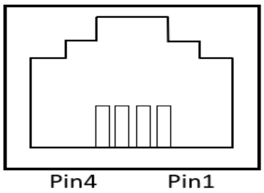


Figure 6 - RJ11 Port

Pin Number	1	2	3	4
Function	GND	DC5V OUT	RS485 A/Y	RS485 B/Z

Table 3 - RJ11 Port Pin Function

3.1.9 Reset Button

Both the PanL Hub44 and PanL Hub80 are equipped with a reset button. Note that this button is not limited to the reset function; it can be configured according to the user’s design. Refer to the Section 5.1.2 for more details.

3.1.10 Micro-SD Card Slot

Both the PanL Hub44 and PanL Hub80 include a Micro-SD card slot for user-defined functionality.

3.1.11 RTC Battery and Battery Switch

Both the PanL Hub44 and PanL Hub80 comes with a CR1225 battery for the RTC circuitry. A slide switch located on the bottom side allows the user to turn the battery on or off.

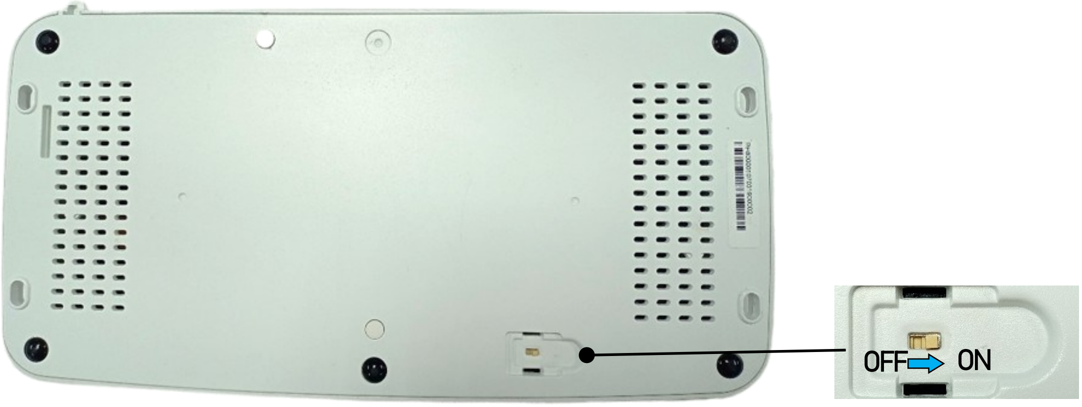


Figure 7 - Battery Switch Location- Enclosure Assembled

3.2 PanL Hub44 - PCB Profile

The circuit board sits beneath the enclosure. The following images show the board and highlight key components.

Dimension of main board: 217.0mm (L) X 115.0mm (W) X 1.6mm (T) with tallest component height of approximately 16.5mm.

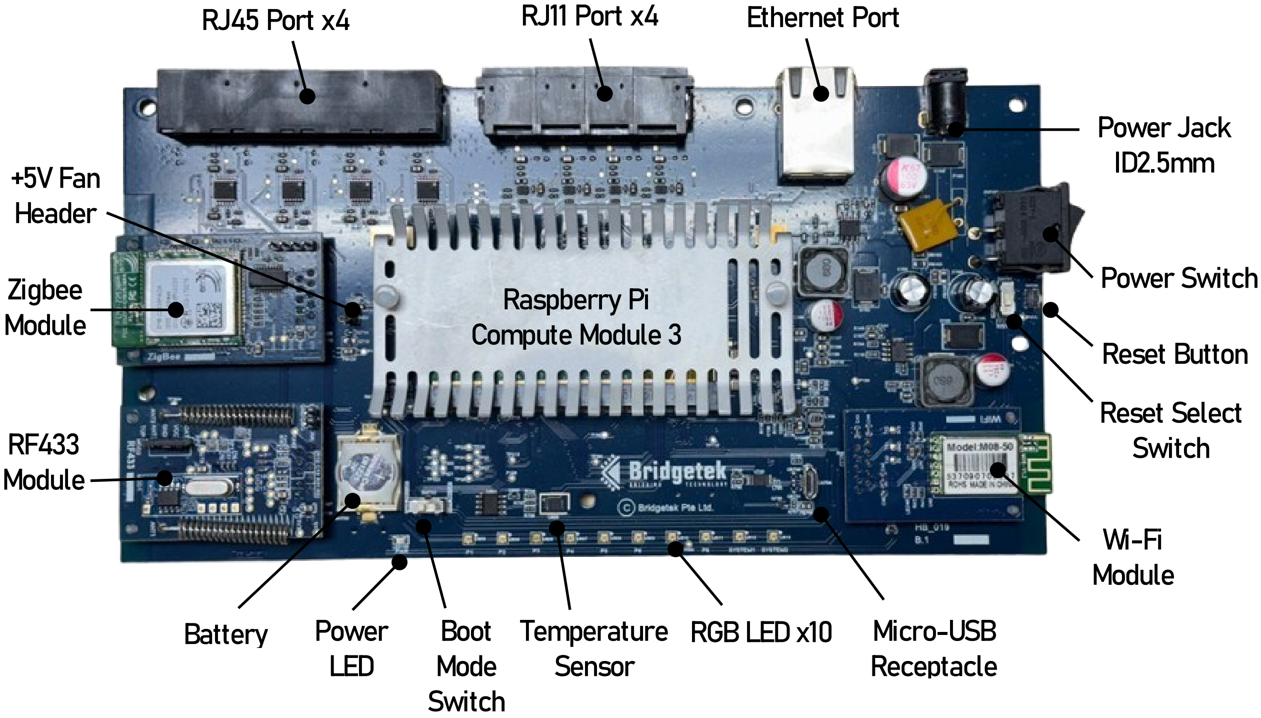


Figure 8 - PanL Hub44 PCB Top View

Copyright © BRT Systems Pte Ltd

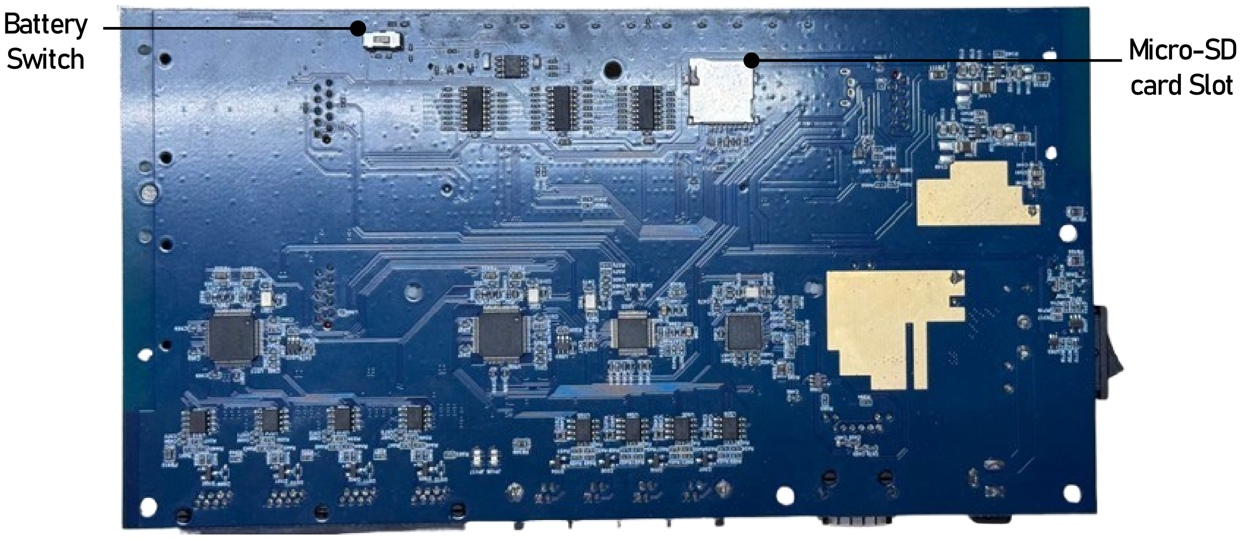


Figure 9 - PanL Hub44 PCB Bottom View

Refer to the board schematics for detailed PanL Hub44 diagrams.

3.3 PanL Hub80 - PCB Profile

Dimension of main board: 217.0mm (L) X 115.0mm (W) X 1.6mm (T) with tallest component height of approximately 16.5mm.

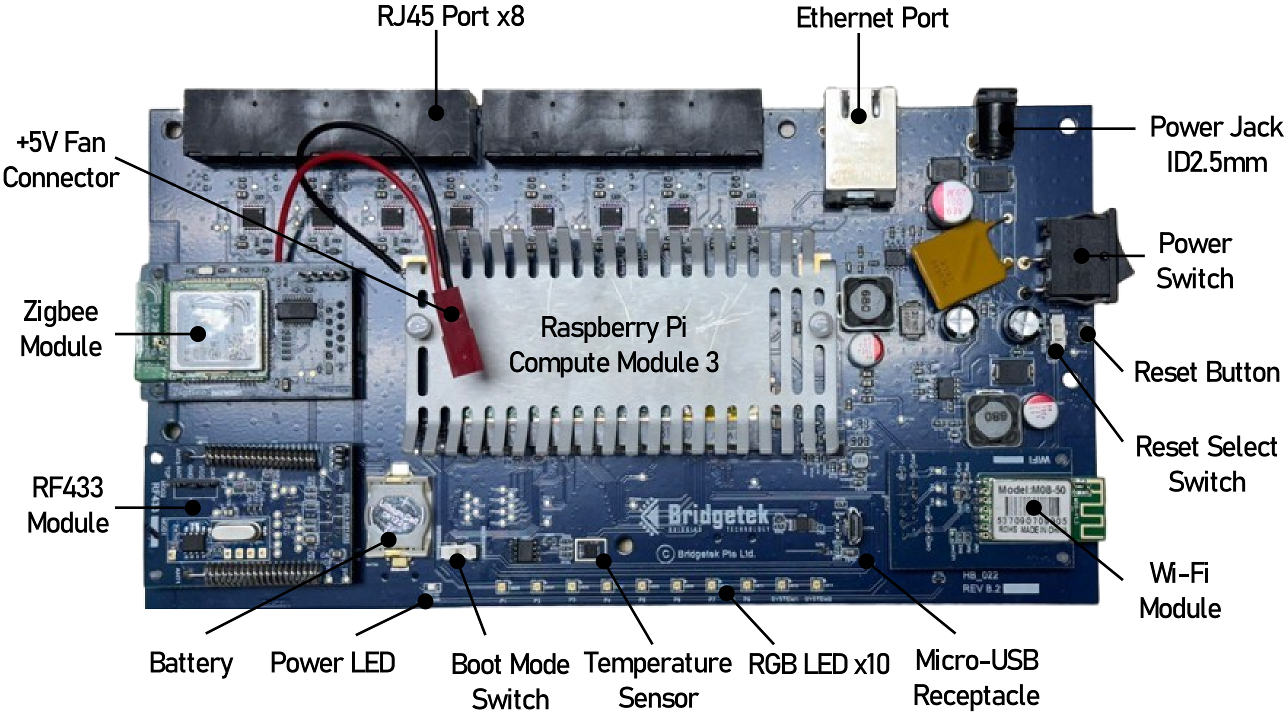


Figure 10 - PanL Hub80 PCB Top View

Copyright © BRT Systems Pte Ltd

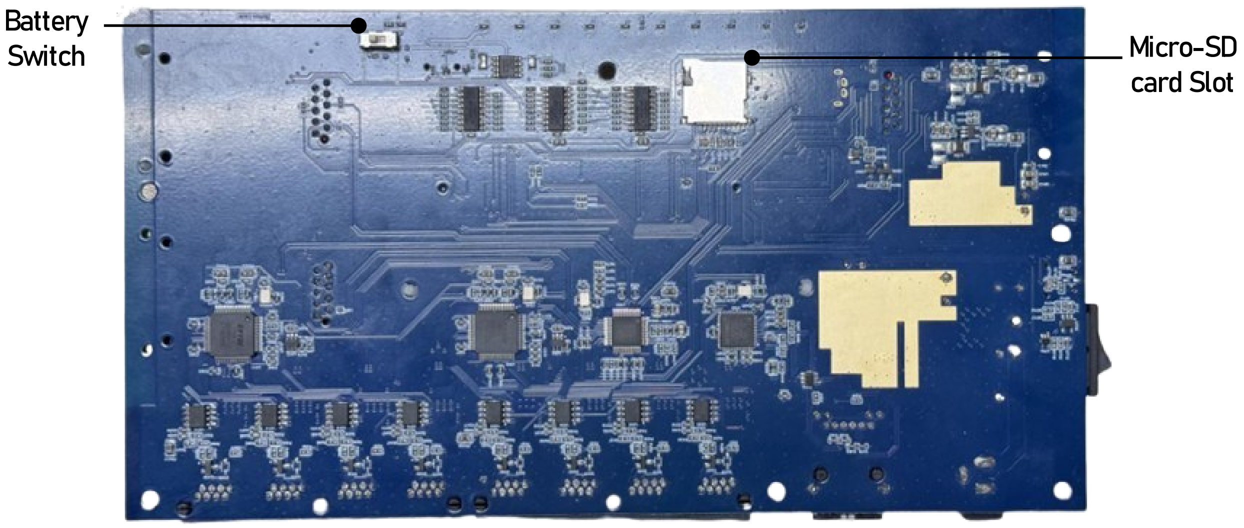


Figure 11 - PanL Hub80 PCB Bottom View

Refer to the board schematics for detailed PanL Hub80 diagrams.

Copyright © BRT Systems Pte Ltd

4 Getting Started with PanL Hub

4.1 Setup Instructions

1. Take note of the following recommendations when setting up PanL Hub:
   - Ensure the availability of an AC outlet nearby.
   - Place the hub on a flat surface area.
   - Best to place the hub in an open area with good ventilation.
   - Best to position the hub central to all wired or wireless devices (if any).
   - Place the hub near to Ethernet network point if using wired Ethernet connection.
   - Apply screws to the sides of the hub to secure the hub.

2. Connect an RJ45 Ethernet cable (not included in PanL Hub package) to the Ethernet port of the hub. Alternatively, the hub can be connected to a wireless network if the Wi-Fi module is built-in the hub.

3. Connect any wired devices to the hub through the RJ45 ports or RJ11 ports. Ensure that the length of the RJ45 8P8C cables do not exceed 100 Meters from the port to the wired device. In the event that a single port is used to power up a few devices in a daisy chain configuration, the first connection to a device and the subsequent connections between the devices must not exceed 50 meters in cable length. The total combined cable length must not exceed 100 meters. For RJ11 port connections, the RJ11 6P4C cable length must not exceed 50 meters from the port to the device.

Note: Ensure that the power cable connection and all device connections are in place before powering on the PanL Hub.

It is not advisable to add or remove any connected devices from the hub while the hub is in ON condition. Instead, switch OFF the power and plug in or out the PanL devices first if required.

4. Attach an AC power plug cable to the power adaptor included and connect to the hub’s power jack. Switch on the power button located at the side of the hub.

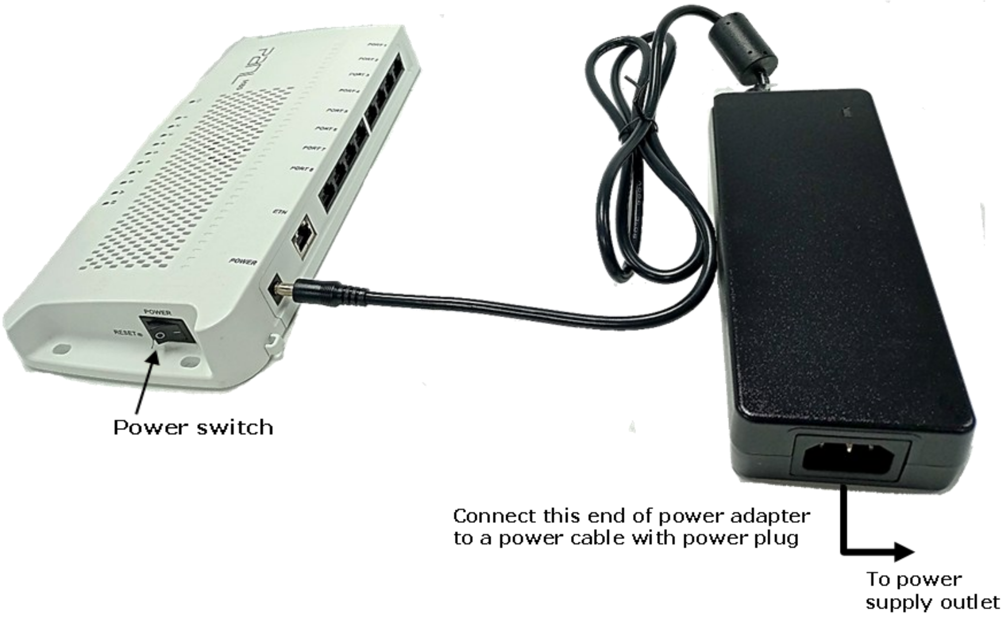

Power switch
Connect this end of power adapter to a power cable with power plug
To power supply outlet

Figure 12 - Powering on PanL Hub

Copyright © BRT Systems Pte Ltd

4.2 Running Sample Applications on PanL Hub

4.2.1 Required Switch Configuration

Before running any application or SDK example, ensure the following switches on the PCB are set:

Switch	Function	Required Setting	Description
SW701	Reset Switch	**SW_RST**	Enables software detection of the Reset button.
SW702	Boot Mode Switch	**BOOT_DISABLED** (default)	Leave disabled during normal operation. Enable only when flashing OS via USB.
SW703	Battery Switch	**BATT_ENABLED**	Powers the real-time clock.

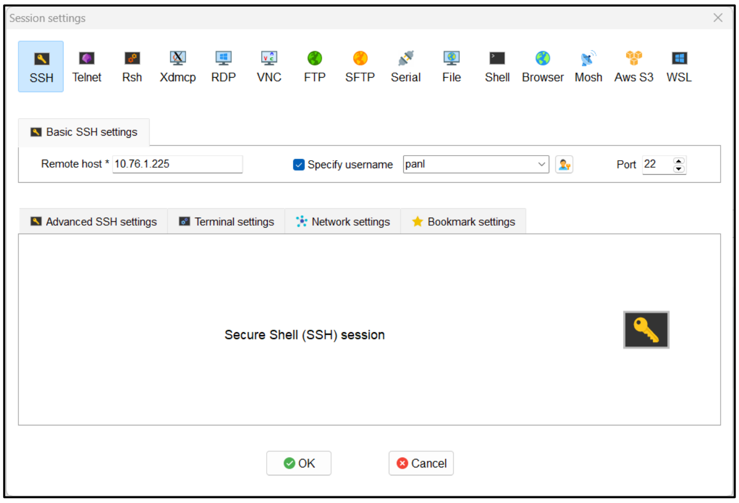
Table 4 - Switch Configuration

4.2.2 How to Connect to PanL Hub

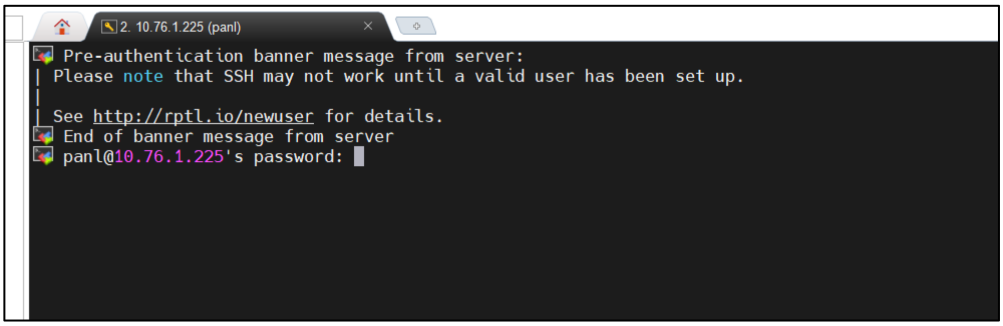
1. Connect the device to your local network via an Ethernet cable.

2. Determine the PanL Hub's IP address by checking your router configuration or using a network scanning tool. (e.g., Advanced IP Scanner).

3. **Launch MobaXterm** (or any other SSH connection tool). Select *SSH.* Enter *Remote Host* IP (PanL Hub) address and following default credentials:
   - ○ *Username:* panl
   - ○ *Password:* panlhub



Remote host * 10.76.1.225 ☑ Specify username panl Port 22



```
Pre-authentication banner message from server:
| Please note that SSH may not work until a valid user has been set up.
|
| See http://rptl.io/newuser for details.
End of banner message from server
panl@10.76.1.225's password: 
```

Copyright © BRT Systems Pte Ltd

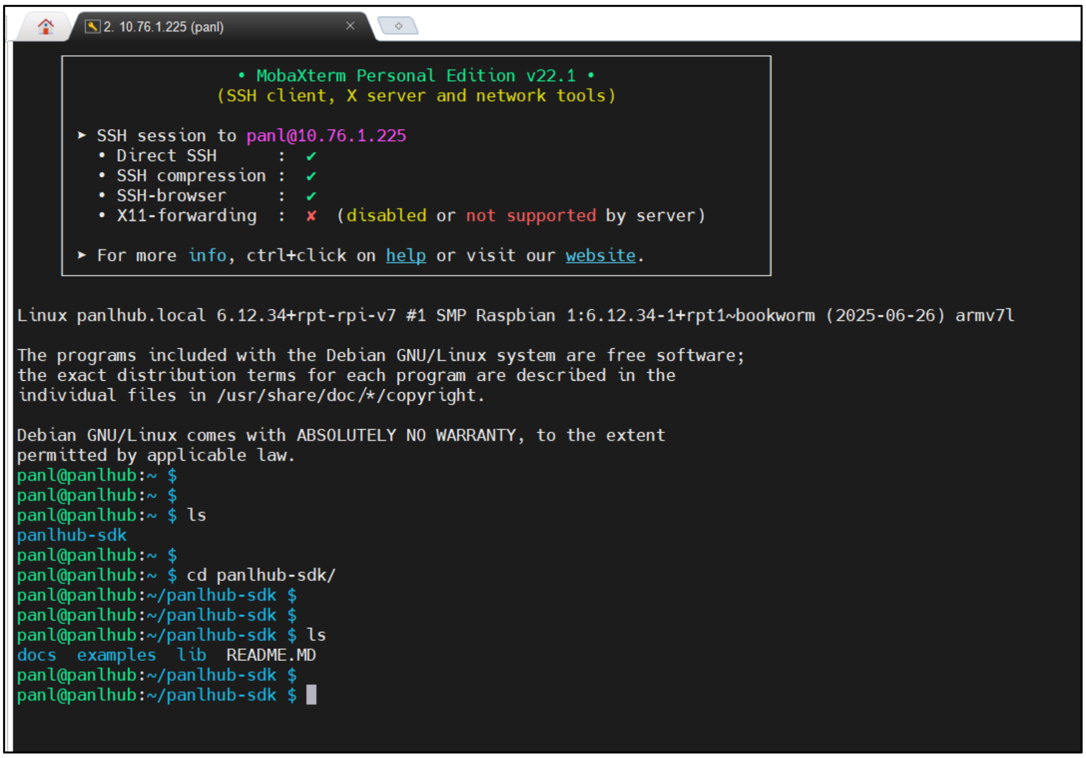
4.2.3 SDK Installation and Directory Structure

The PanL Hub SDK comes pre-installed at the following location: **/home/panlhub/panlhub-sdk**

Directory layout:

```
panlhub-sdk/
├── docs/          # API references and technical docs
├── examples/      # Sample applications (Zigbee, Wi-Fi, RF433, board control)
├── lib/           # Core libraries for PanL Hub APIs
└── README.txt     # Summary of SDK usage
```

Refer to figure below for the SDK directory structure.



```
                    • MobaXterm Personal Edition v22.1 •
                  (SSH client, X server and network tools)

      ➤ SSH session to panl@10.76.1.225
        • Direct SSH      :  ✔
        • SSH compression :  ✔
        • SSH-browser     :  ✔
        • X11-forwarding  :  ✘  (disabled or not supported by server)

      ➤ For more info, ctrl+click on help or visit our website.

Linux panlhub.local 6.12.34+rpt-rpi-v7 #1 SMP Raspbian 1:6.12.34-1+rpt1~bookworm (2025-06-26) armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
panl@panlhub:~ $
panl@panlhub:~ $
panl@panlhub:~ $ ls
panlhub-sdk
panl@panlhub:~ $
panl@panlhub:~ $ cd panlhub-sdk/
panl@panlhub:~/panlhub-sdk $
panl@panlhub:~/panlhub-sdk $
panl@panlhub:~/panlhub-sdk $ ls
docs  examples  lib  README.MD
panl@panlhub:~/panlhub-sdk $
panl@panlhub:~/panlhub-sdk $ █
```

Copyright © BRT Systems Pte Ltd

4.2.4 Run a sample test (Zigbee)

Here is an example demonstrating how to control Zigbee devices using the PanL Hub. Ensure that your PanL Hub has a Zigbee daughterboard installed before proceeding.

The steps below demonstrate how to discover Zigbee devices, select a Zigbee light device, and control it by turning it ON or OFF.

1. **Run Zigbee Gateway -** The Zigbee gateway, running on the PanL Hub, communicates with the Zigbee module via a serial connection, allowing clients to connect to and control devices within the Zigbee network.

```
# Install the external libraries (if you haven't done so).
sudo cp examples/zigbee/zbgw/lib/libprotobuf.so.8 /usr/lib
sudo cp examples/zigbee/zbgw/lib/libprotoc.so.8 /usr/lib
sudo cp examples/zigbee/zbgw/lib/libsocket++.so /usr/lib
sudo ldconfig

# Run server:
cd examples/zigbee/zbgw/servers

chmod +x GATEWAY_SRVR_arm NPI_lnx_arm_server NWKMGR_SRVR_arm OTA_SRVR_arm ZLSZNP_arm logger
chmod +x start_application track_servers zigbeeHAgw

sudo ./zigbeeHAgw beaglebone

# After it's up, let it run and in another terminal, run the Zigbee client.
```

2. **Run Zigbee Client -** The Zigbee client is an application running on the PanL Hub that communicates with the Zigbee gateway to discover and control Zigbee devices.

```
# Run client:
cd examples/zigbee/zbapp

chmod +x main

sudo ./main
# Install the external libraries (if you haven't done so).
sudo cp examples/zigbee/zbgw/lib/libprotobuf.so.8 /usr/lib
sudo cp examples/zigbee/zbgw/lib/libprotoc.so.8 /usr/lib
sudo cp examples/zigbee/zbgw/lib/libsocket++.so /usr/lib
```

3. Zigbee client interactive commands

- Discovery

```
FTDI> discovery 300
[10:51:32.626] [16893] [info] HandleDiscoveryDuration() PROCESSING(18)...
[10:51:33.626] [16893] [info] HandleDiscoveryDuration() PROCESSING(19)...
...
[10:51:36.627] [16893] [info] HandleDiscoveryDuration() PROCESSING(300)...
sudo ldconfig
```

- Show list of devices

```
FTDI> device list
[10:53:54.687] [16893] [info] [158D00067CEFE1-1] name=ZbLightGeneric
```

- Turn off the light

```
FTDI> device off 158D00067CEFE1 1
```

- Turn on the light

```
FTDI> device on 158D00067CEFE1 1
```

Copyright © BRT Systems Pte Ltd

- Remove the light

```
FTDI> device remove 158D00067CEFE1 1
```

- Exit

```
FTDI> exit
```

5 Supported APIs

This section describes all the PanL Hub SDK API modules located in*:*
~/panlhub-sdk/lib/board_control

These APIs allow developers to control hardware components such as LEDs, buttons, RF modules, and RS-485 devices.

5.1 API Classes

5.1.1 Class Board

This class defines the information necessary for board configuration and control. It serves as a dependency for other classes that implement the control logic.

The following is an example script demonstrating how to use the class.

```
from lib.board_control import Board, Button
from time import sleep

try:
    board = Board()
    button = Button(board)
    print('Press and hold the side button (Reset button).')
    print('Or press CTRL+C to halt.')

    while True:
        if button.is_being_pressed():
            print(f"Button is being pressed.")

        sleep(1)
finally:
    board.cleanup()
```

In this script, the board object is initialized and passed to the button object during initialization. Before the script exits, `board.cleanup()` is called to finalize internal resources and restore the board to its default state, preventing warnings in subsequent uses.

5.1.2 Class Button

The Button class provides a simple API for checking the state of the Reset button. It is initialized with a Board instance, and the `button.is_being_pressed()` method returns the current press state.

The example script checks the button state every second, which is not efficient for real-time applications. See the example script provided in Section 5.1.3.

For event-driven detection of press and release actions, refer to the `test_event.py` example, which uses asynchronous file descriptor monitoring.

5.1.3 Class LedStrip

The LedStrip class provides APIs for controlling the board's LED strip. Here are some useful methods provided by the class.

- The `configure_led()` method modifies the configuration of a specific LED.
- The `configure_all_leds()` method modifies the configuration of all LEDs in the strip.
- The `apply_changes()` method commits the configuration changes to the LED hardware. After this method is called, all of the configuration that has been modified earlier will be reflected on the LED strip.

Copyright © BRT Systems Pte Ltd

The following script sets the first three LEDs of the strip to red, green and blue colors respectively.

```
from lib.board_control import Board, LedStrip
from time import sleep

try:
    board = Board()
    led_strip = LedStrip(board)
    led_strip.configure_led(0, True, 0xFF0000, 0.1)
    led_strip.configure_led(1, True, 0x00FF00, 0.1)
    led_strip.configure_led(2, True, 0x0000FF, 0.1)
    led_strip.apply_changes()
    print('Press CTRL+C to exit.')

    while True:
        sleep(1)
finally:
    # Reset LED strip's state:
    led_strip.configure_all_leds(False, 0, 0)
    led_strip.apply_changes()
    board.cleanup()
```

5.1.4 Class LoadSense

The LoadSense class provides APIs for querying load states of the RS485 physical ports. The following methods are available.

- The `has_load()` method returns True if a port is connected to external hardware; otherwise, it returns False. It is possible to pass a port index to check load state of a specific port.
- The `read_load()` method returns a list of load states of all RS485 ports.

The following script regularly checks the load states of RS485 ports. Try changing port usage to see the output.

```
from lib.board_control import Board, LoadSense
from time import sleep

try:
    board = Board()
    sense = LoadSense(board)
    print('Try using a port.')

    while True:
        if sense.has_load():
            print('Load states:', sense.read_load())

        sleep(1)
finally:
    board.cleanup()
```

The example script polls the button state every second, which is inefficient for real-time applications. For event-driven detection, see the `test_event.py` example, which uses asynchronous file descriptor monitoring.

5.1.5 Class PortPowerController

The PortPowerController class provides API functionality to regulate power delivery to RS485 physical ports.

The `set_power()` method configures the power distribution to the ports. It accepts a list of power states, one for each port. Possible values of a state include:

- `True`: Power enabled.
- `False`: Power disabled.

Copyright © BRT Systems Pte Ltd

The following example enables the power supply for the first two ports. Connect devices to these ports to verify functionality.

```
from lib.board_control import Board, PortPowerController
from time import sleep

try:
    board = Board()
    power_controller = PortPowerController(board)
    power_controller.set_power([True, True, False, False, False, False, False, False])
    print('Power will be supplied on port indices 0 and 1.')

    while True:
        sleep(1)
finally:
    board.cleanup()
```

5.1.6 Class ThermalReader

The ThermalReader class provides APIs for retrieving temperature data from the board and the core processing unit. The following methods are supported.

- The `get_board_temp()` method returns the temperature of the board.
- The `get_core_temp()` method returns the temperature of the core processing unit.

Here is an example usage.

```
from lib.board_control import Board, ThermalReader

try:
    board = Board()
    reader = ThermalReader(board)
    print(f"Core temperature is {reader.get_core_temp()} C degrees")
    print(f"Board temperature is {reader.get_board_temp()} C degrees")
finally:
    # Although ThermalReader does not use GPIO, there is no harm to call a board.cleanup()
here.
    board.cleanup()
```

5.1.7 Class PanlRf433Device

The PanlRf433Device class provides APIs for using the RF433 extension module. Here are some available methods.

- The `connect()` method establishes a logical connection to the RF433 module.
- The `check_status()` method tests if the connection is made and the RF433 module is reachable.
- The `fetch_version()` method retrieves version information of the RF433 module.
- The learn method puts the RF433 module into learning mode where it listens for and returns a received RF433 radio message.
- The `play()` method asks the RF433 module to emit a RF433 radio message.
- The `reset()` method resets the RF433 module.

Use the following example with an RF433 device to observe the output. After displaying version information and connection status, the script puts the RF433 module into learning mode. Any captured messages will be replayed after 5 seconds.

```
from lib.board_control import Board, PanlRf433Device
import time
try:
    board = Board()
    device = PanlRf433Device(board)
    device.connect()
    print('RF module status:', device.check_status())
    print('RF module version (HW, FW):', device.fetch_version())
    key = device. Learn()
    print('Learned key:', key)

    print('The key will be replayed after 5 seconds.')
    time.sleep(5)
    print('Playing...')
    result = device.play(key)
    print('Played:', result)
except Exception as ex:
    print('Exception:', ex)
finally:
    board.cleanup()
```

5.1.8 Class Rtc

The Rtc class provides APIs for getting and setting RTC time. Although the RTC is perfectly integrated to the OS, users can use this class to handle the RTC time.

- The `get time()` method returns time data of the RTC clock.
- The `set_time()` method sets time data to the RTC clock.

Here is the example usage of Rtc class.

```
from lib.board_control import Board, Rtc

try:
    board = Board()
    rtc = Rtc(board)
    print('sec, min, hour, day, mon, year, week day (unused), year day (unused), is DST
(unused)')
    print(rtc.get_time())
finally:
    board.cleanup()
```

It is recommended to use the standard Linux `hwclock` commands for RTC interaction. Use `sudo hwclock -r` for reading and `sudo hwclock -w` for writing.

5.2 Advanced Usage

5.2.1 Monitor Board Events

Polling component states is inefficient for real-time applications. This example demonstrates an event-driven approach using `gpiod` and `select.epoll`. It monitors the GPIO pin of the Reset button and prints messages on both press and release events.

```
import gpiod
import select

# GPIO settings
GPIO_CHIP = "/dev/gpiochip0"
GPIO_PIN = 45 # Reset button. Change the pin number to the GPIO pin that you want to
monitor.

# Open GPIO chip and request the line
chip = gpiod.Chip(GPIO_CHIP)
line = chip.get_line(GPIO_PIN)

# Configure GPIO as input with edge detection
line.request(consumer = "gpio_epoll", type=gpiod.LINE_REQ_EV_BOTH_EDGES)

# Get the file descriptor for epoll
fd = line.event_get_fd()
epoll = select.epoll()
epoll.register(fd, select.EPOLLIN)
```

Copyright © BRT Systems Pte Ltd

```
print(f"Monitoring GPIO {GPIO_PIN} for events using epoll...")
print(f"<Press and release the Reset button>")
try:
    while True:
        events = epoll.poll()

        for fileno, event in events:
            if fileno == fd:
                gpio_event = line.event_read()
                edge_type = "RISING" if gpio_event.type == gpiod.LineEvent.RISING_EDGE else
"FALLING"
                print(f"GPIO {GPIO_PIN} event detected! Type: {edge_type}")
finally:
    epoll.unregister(fd)
    epoll.close()
    line.release()
```

5.2.2 Control a Modbus Device

This example demonstrates using the PanL Hub to control and read data from a 4-in-1 ModBus sensor connected via an RS-485 port. Note that the `baudrate`, `parity`, `bytesize`, `stopbits`, and register addresses may vary in your setup. The example assumes the specified values are correct. Below is the register map:

Name	**Starting Address**	**Quantity of Registers**	**Supported Function Codes**	**Parameter Range and Description**
Address	0x0000	1	0x03, 0x10	1 to 126.
Device UUID	0x0026	8	0x03	MSxxxxxxxxxxxxyy where x is ASCII char and yy is 16bit running number.
Identify	0x0152	1	0x06	Write 1 to start blinking the device @1Hz.

Table 5 - Register Map

Note: Remember to use `PortPowerController` to supply power to the active port.

This example uses `pymodbus` with a serial client to communicate with the sensor. Enable `handle_local_echo` in the configuration, as the hardware loops back sent data.

```
from lib.board_control import Board, PortPowerController
from pymodbus.client import AsyncModbusSerialClient
from pymodbus.framer import FramerType
from pymodbus import ModbusException
from time import sleep
import asyncio

def to_uuid(registers: list[int]):
    r = ''

    for reg in registers[:-1]:
        r += chr((reg >> 8) & 0xFF) + chr(reg & 0xFF)

    return f"{r}{registers[-1]:05d}"
```

```
async def test_modbus_4in1_sensor(if_name: str):
    try:
        client = AsyncModbusSerialClient(
            port= if_name,
            framer= FramerType.RTU,
            baudrate= 9600,
            bytesize= 8,
            parity= 'E',
            stopbits= 1,
            name= 'testmodbus',
            timeout= 0.1,
            handle_local_echo= True,
        )


        if await client.connect():
            res = await client.read_holding_registers(address= 0x0000, count= 1, slave=
126)
            print(f"Device address: {res.registers}")

            res = await client.read_holding_registers(address= 0x0026, count= 8, slave=
126)
            print(f"Device UUID: {to_uuid(res.registers)}")

            res = await client.write_register(address= 0x0152, value= 1, slave= 126)
            print("The sensor's LED should be blinking now.")
            sleep(5) # Wait a bit to observe the blinking
            client.close()
    except ModbusException as ex:
        print(ex)

try:
    board = Board()
    power_controller = PortPowerController(board)
    power_controller.set_power([True, False, False, False, False, False, False, False])
    print('Please connect the 4-in-1 sensor to the first RS485 port.')
    asyncio.run(test_modbus_4in1_sensor(board.RS485_IF0))
finally:
    power_controller.set_power()
    board.cleanup()
```

The first `read_holding_registers()` function call should return the sensor address (which is 126, in the example).

The second `read_holding_registers()` function call should return the sensor UUID (in this format: MS01010101272105632).

The `write_register()` function call will make the sensor LED blink.

6 Contact Information

Refer to https://brtsys.com/contact-us/ for contact information.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any BRT Systems Pte Ltd (BRTSys) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested BRTSys devices and other materials) is provided for reference only. While BRTSys has taken care to assure it is accurate, this information is subject to customer confirmation, and BRTSys disclaims all liability for system designs and for any applications assistance provided by BRTSys. Use of BRTSys devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify, and hold harmless BRTSys from any and all damages, claims, suits, or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted, or reproduced in any material or electronic form without the prior written consent of the copyright holder. BRT Systems Pte Ltd, 1 Tai Seng Avenue, Tower A, #03-01, Singapore 536464. Singapore Registered Company Number: 202220043R.

Appendix A – FAQ

How to open the PanL Hub case?

When performing tasks such as flashing the operating system image or modifying system configurations, it may be necessary to access the main circuit board. The board is housed beneath the device enclosure. Refer to Sections 3.2 and 3.3 for the PCB profiles for PanL Hub44 and PanL Hub80, respectively.

How to flash OS image to PanL Hub?

When there is an error with the PanL Hub, resolving it may require flashing an OS image to the hub, which involves re-writing the operating system into the hub's memory.

Follow the steps below to flash an OS image to a PanL Hub.

1. **Prepare the Image**

    - Obtain the OS image (*2025-09-22-raspbian-12-cm3-lite.img*) by downloading it from the BRTSys website, contacting BRTSystems directly, or creating a custom OS image.

2. **Power Off and Open the Hub**

    - **Note: Turn off the PanL Hub.**
    - Open the enclosure to access the internal circuit board.

3. **Configure Switch**

    - Locate **SW702** on the board, next to the battery. By default, it is set to *BOOT_DISABLED*. Switch it to *BOOT_ENABLED* (slide right) to enable USB boot.
    - Locate switch SW701 on the board and flip it upward.

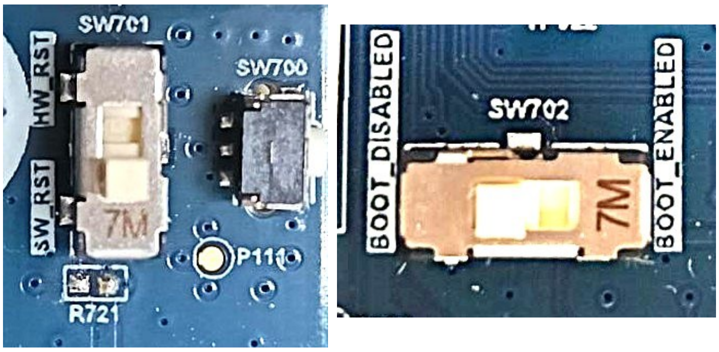


Figure 13 - SW701 and SW702

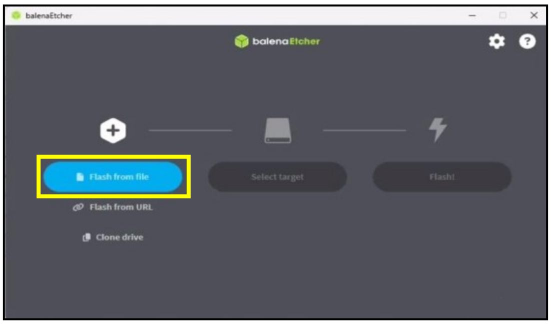
4. **Connect to Computer**

    - Connect the PanL Hub to your computer using a micro-USB cable.
    - Power on the board. It is now ready for USB boot.

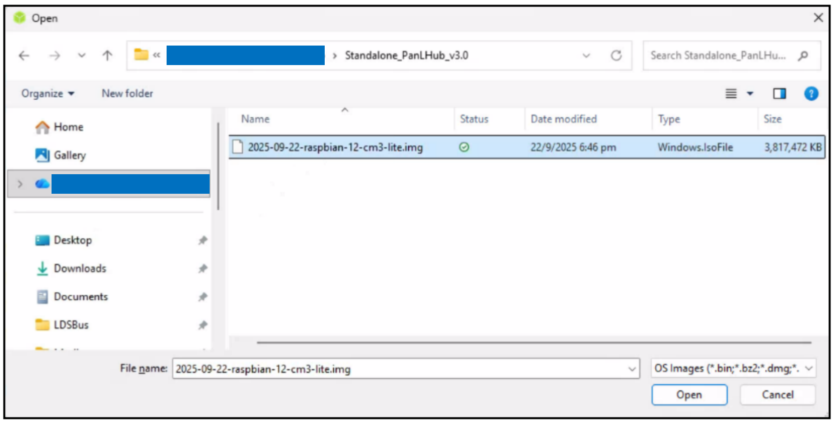
Copyright © BRT Systems Pte Ltd

5. Flash the Image

- Download balenaEtcher tool.
- Open balenaEtcher and Select *Flash From file*



- Navigate to the location of the OS image file and select it.



[redacted] › Standalone_PanLHub_v3.0

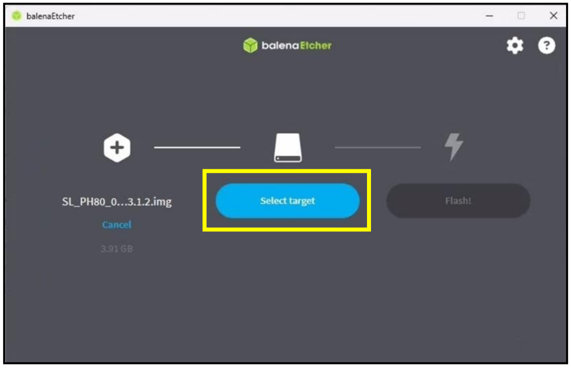
Search Standalone_PanLHu...

[redacted]

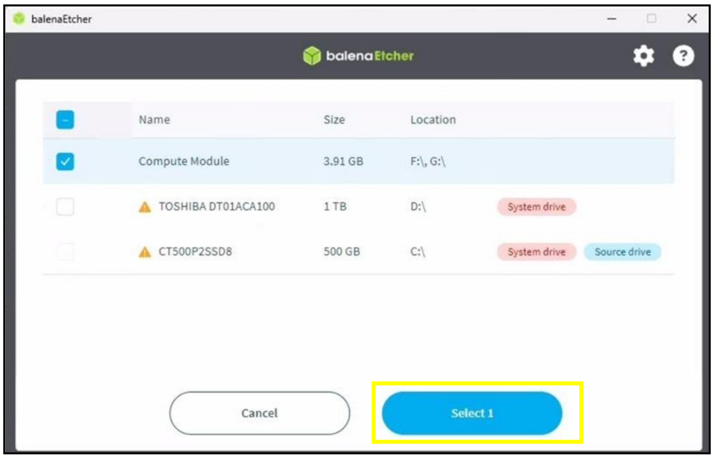
Name	Status	Date modified	Type	Size
2025-09-22-raspbian-12-cm3-lite.img	✓	22/9/2025 6:46 pm	Windows.IsoFile	3,817,472 KB

File name: 2025-09-22-raspbian-12-cm3-lite.img

OS Images (*.bin;*.bz2;*.dmg;*.



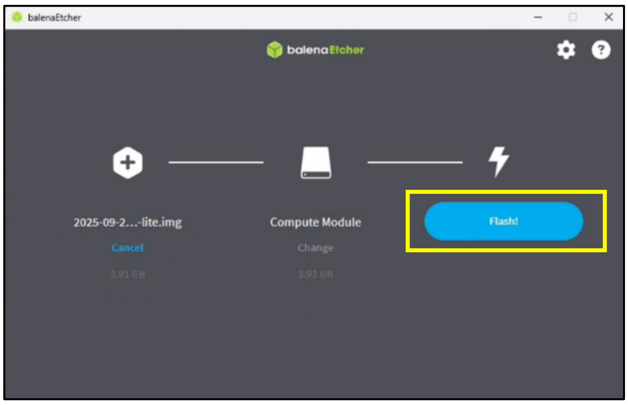
- Click *Select target* and choose **USB port driver** as Compute Module.

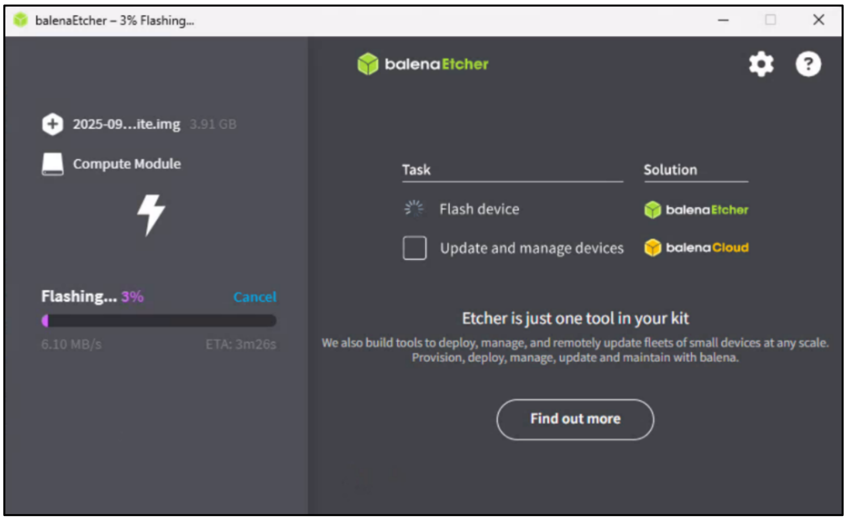


	Name	Size	Location	
☑	Compute Module	3.91 GB	F:\, G:\	
□	⚠ TOSHIBA DT01ACA100	1 TB	D:\	System drive
□	⚠ CT500P2SSD8	500 GB	C:\	System drive Source drive

Copyright © BRT Systems Pte Ltd

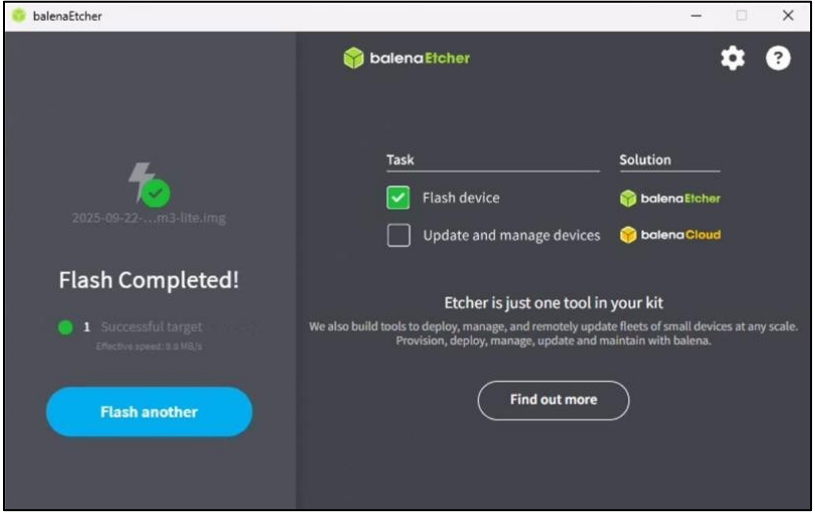
- Click *Flash* and wait for the process to complete.





balenaEtcher – 3% Flashing...
balenaEtcher
2025-09...ite.img 3.91 GB
Compute Module
Flashing... 3%
Cancel
6.10 MB/s
ETA: 3m26s
Task
Solution
Flash device
balenaEtcher
□ Update and manage devices
balenaCloud
Etcher is just one tool in your kit
We also build tools to deploy, manage, and remotely update fleets of small devices at any scale.
Provision, deploy, manage, update and maintain with balena.
Find out more

- The flashing process typically takes approximately 10 minutes.



balenaEtcher
balenaEtcher
2025-09-22-....m3-lite.img
Flash Completed!
1 Successful target
Effective speed: 3.8 MB/s
Flash another
Task
Solution
☑ Flash device
balenaEtcher
□ Update and manage devices
balenaCloud
Etcher is just one tool in your kit
We also build tools to deploy, manage, and remotely update fleets of small devices at any scale.
Provision, deploy, manage, update and maintain with balena.
Find out more

6. Restore Normal Boot

Note: After the flashing process is complete, power off the PanL Hub and disconnect the micro-USB cable.

- Switch **SW702** back to *BOOT_DISABLED*.
- Power on the PanL Hub. The device is now ready for normal operation.

Copyright © BRT Systems Pte Ltd

Why does the Reset button not respond?

Ensure that the **SW701** switch, located in front of the Reset button on the bare circuit board, is set to **SW_RST** (slide downward).

Why is RTC not working?

Ensure that the **SW703** switch, located at the back of the bare circuit board, is set to **BATT_ENABLED** (slide right). Replace the battery if it is drained.

Can external storage be used?

The PanL Hub has a micro-SD card slot located at the back of the bare circuit board (**CN701**). Insert your micro SD card and restart the PanL Hub. Once it is powered on, verify the detection by running the command:

```
sudo lsblk
```

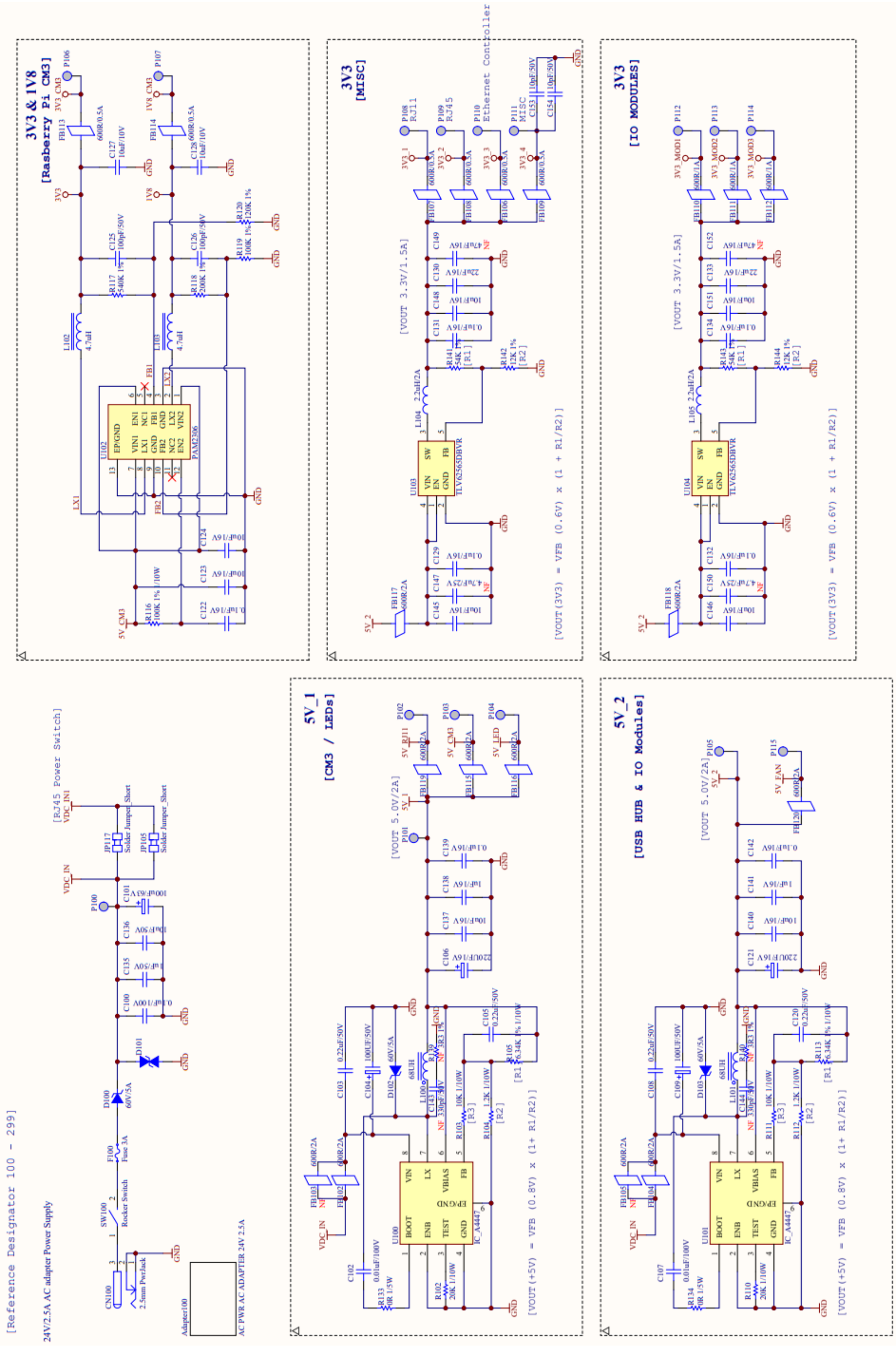
You should see an additional MMC device corresponding to the inserted card.

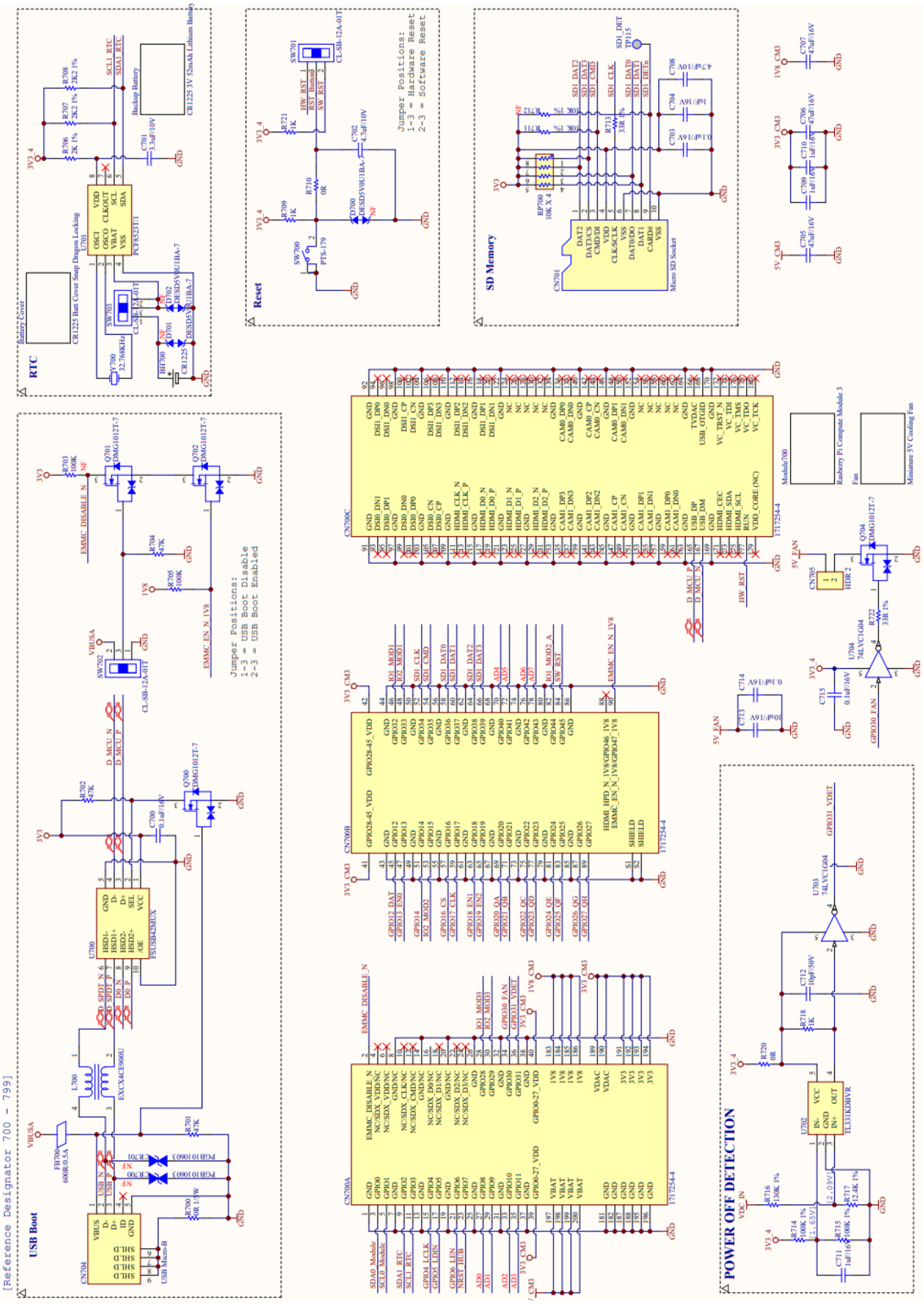
Note: In rare cases, the PanL Hub may fail to boot if the micro-SD card is not supported. If this occurs, remove the card and restart the device; it should boot normally.

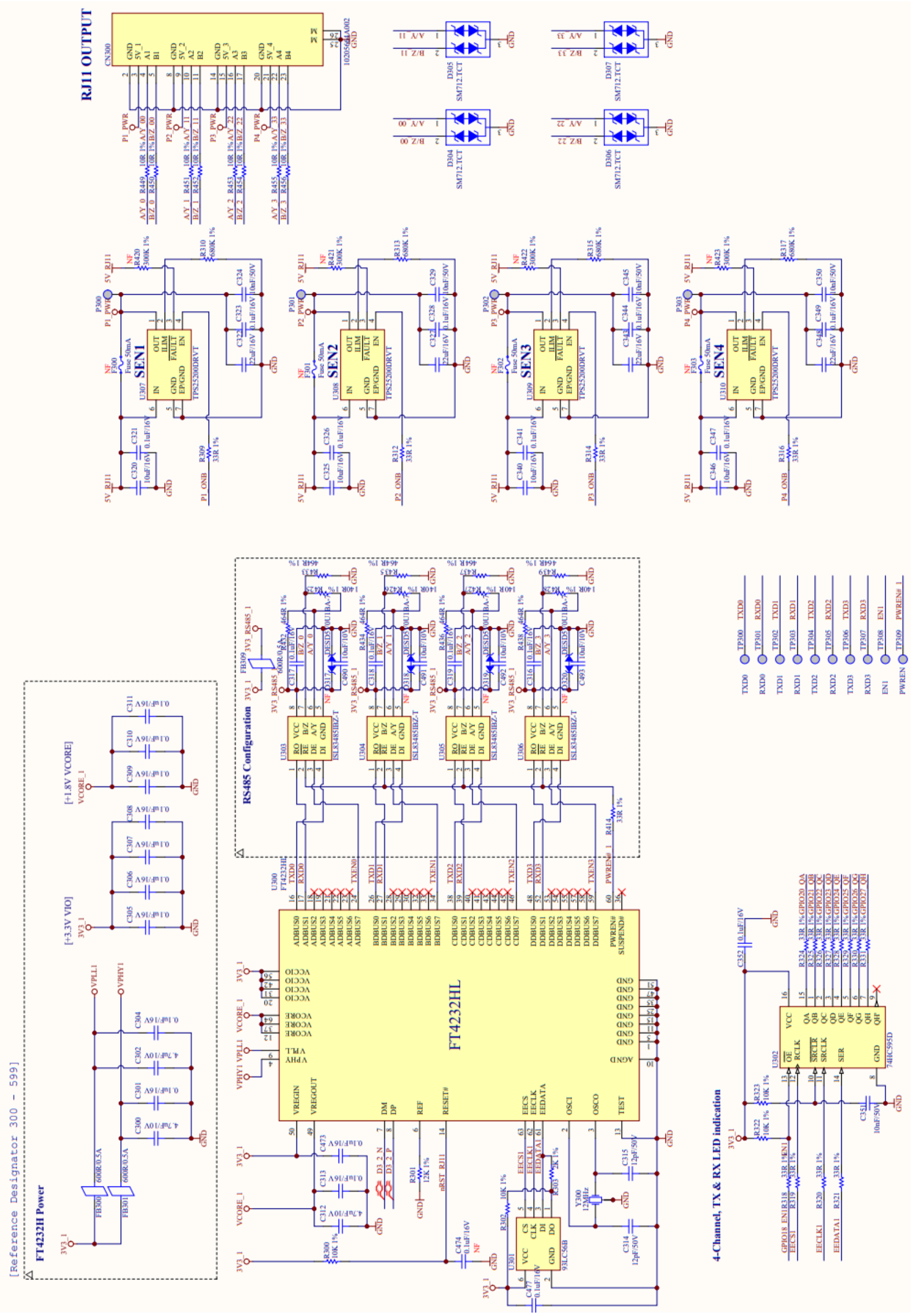
Appendix B – Board Schematics

PanL Hub44 Board Schematics

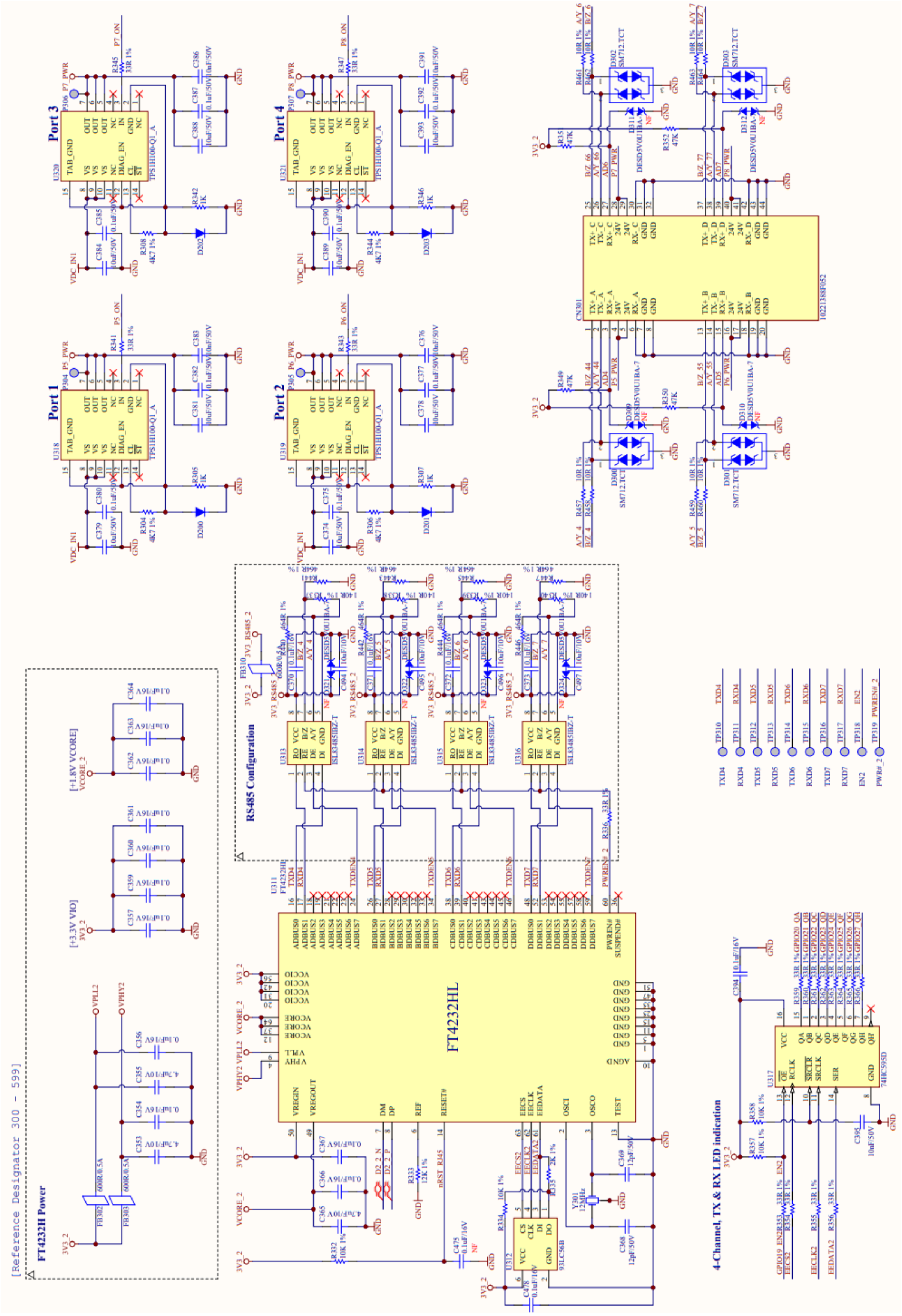
Mainboard Schematics



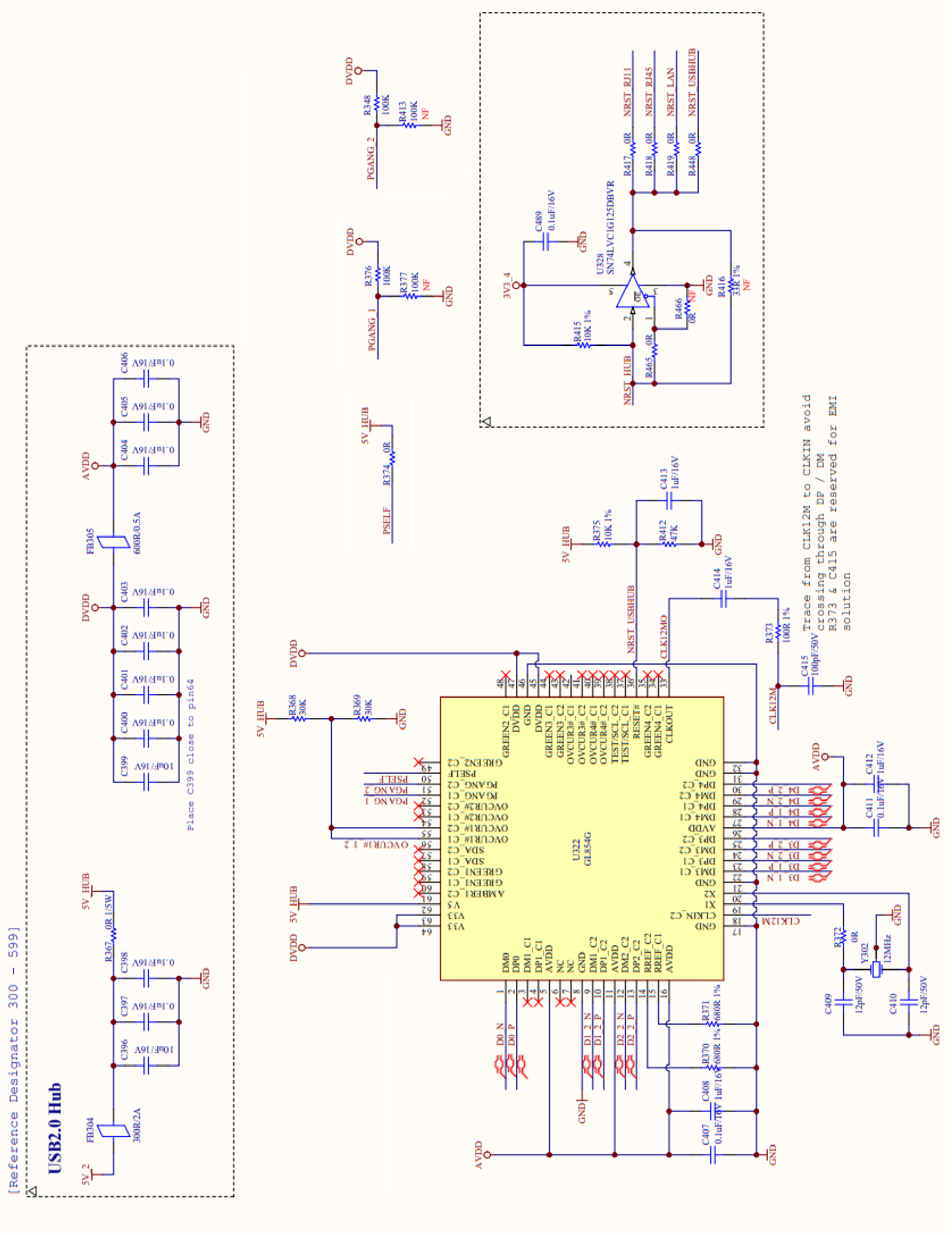

[Reference Designator 700 - 799]
USB Boot
CN704
USB Micro-B
VBUSA
FB700 600R/0.5A
L700 EXC24CE900U
U700 FSUSB42MUX
Q700 DMG1012T-7
R702 47K
C700 0.1uF/16V
D MCU N
D MCU P
SW702 CL-SB-12A-01T
Jumper Positions:
1-3 = USB Boot Disable
2-3 = USB Boot Enabled
EMMC DISABLE N
Q701 DMG1012T-7
Q702 DMG1012T-7
R703 100K
R704 47K
R705 100K
EMMC EN N 1V8
RTC
Battery Cover
CR1225 Batt Cover Snap Dragon Locking
U701 PCF8523T/1
Y700 32.768KHz
BH700 CR1225
Backup Battery
CR1225 3V 52mAh Lithium Battery
C701 3.3uF/10V
R706 2K 1%
R707 2K2 1%
R708 2K2 1%
SCL1 RTC
SDA1 RTC
Reset
SW700 PTS-179
SW701 CL-SB-12A-01T
HW RST
RST Button
SW RST
R709 1K
R710 0R
R721 1K
C702 4.7uF/10V
D700 DESD5V0U1BA-7
Jumper Positions:
1-3 = Hardware Reset
2-3 = Software Reset
SD Memory
CN701
Micro SD Socket
RP700 10K X 4
R711
R712
R713 33R 1%
C703 0.1uF/16V
C704 1uF/16V
C708 4.7uF/10V
SD1 DET
TP115
CN700A
CN700B
CN700C
1717254-4
Raspberry Pi Compute Module 3
Module700
Fan
Miniature 5V Cooling Fan
CN705 HDR 2
5V FAN
Q704 DMG1012T-7
U704 74LVC1G04
R722 33R 1%
C714 0.1uF/16V
C713 10uF/16V
C715 0.1uF/16V
GPIO30 FAN
C705 47uF/16V
C706 47uF/16V
C707 47uF/16V
C709 1uF/16V
C710 1uF/16V
5V CM3
3V3 CM3
1V8 CM3
POWER OFF DETECTION
VDC IN
R714 100K 1%
R715 100K 1%
R716 130K 1%
R717 12.4K 1%
C711 1uF/16V
U702 TL331KDBVR
R718 1K
R720 0R
C712 10pF/50V
U703 74LVC1G04
GPIO31 VDET
3V3 4
3V3
GND



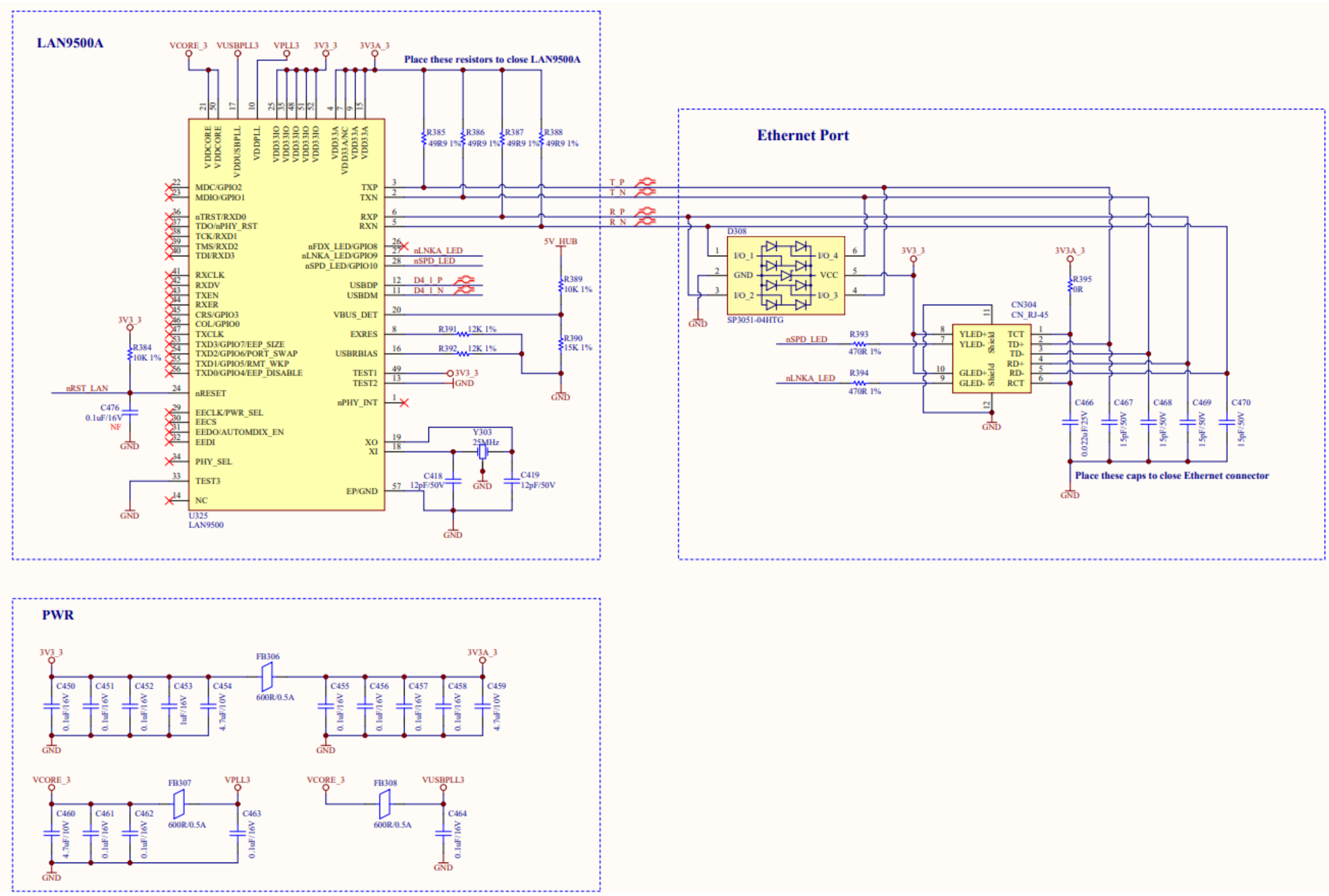
Copyright © BRT Systems Pte Ltd

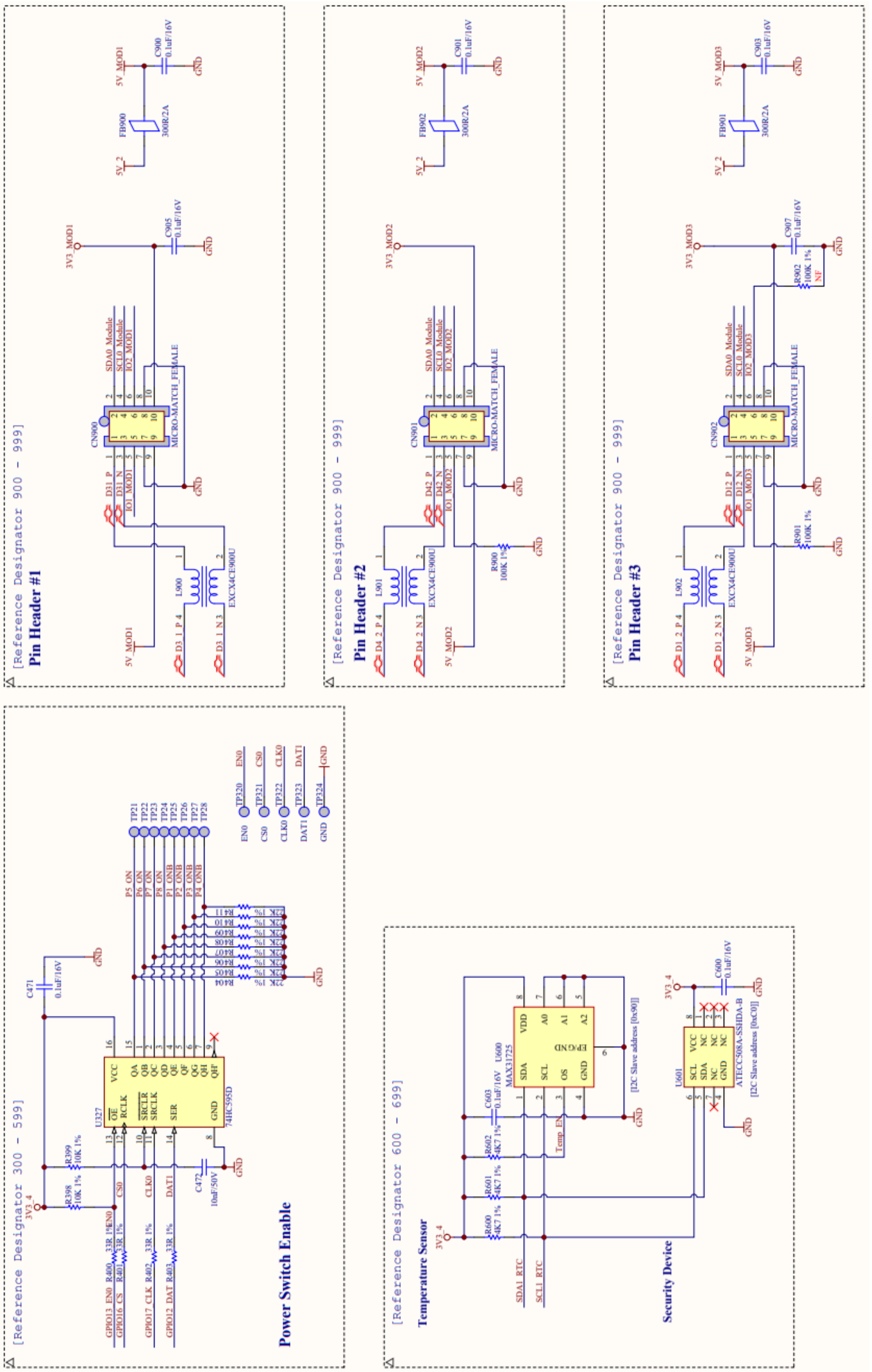


Copyright © BRT Systems Pte Ltd

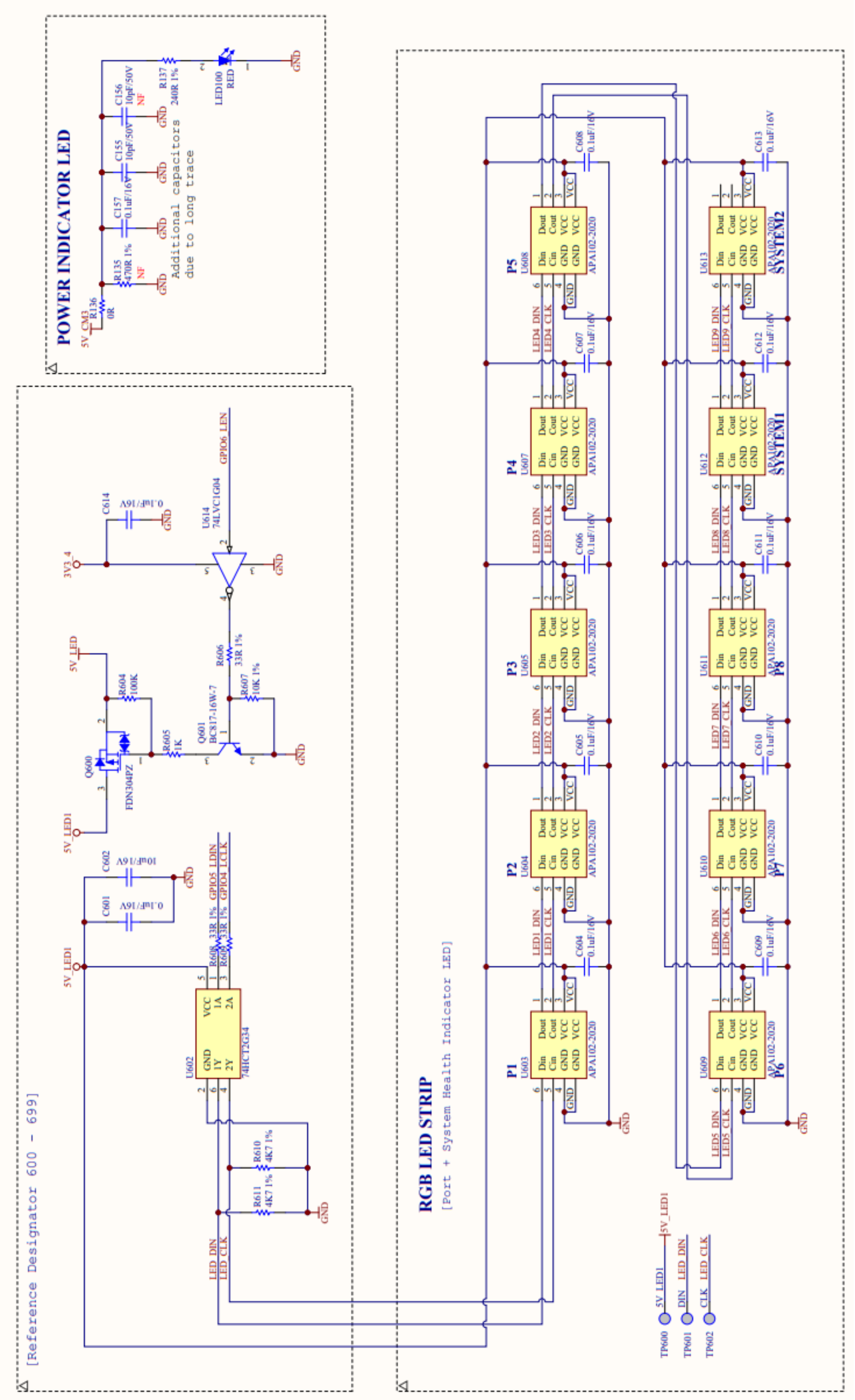


Copyright © BRT Systems Pte Ltd


LAN9500A
VCORE_3
VUSBPLL3
VPLL3
3V3_3
3V3A_3
Place these resistors to close LAN9500A
VDDCORE
VDDCORE
VDDUSBPLL
VDDPLL
VDD33IO
VDD33IO
VDD33IO
VDD33IO
VDD33IO
VDD33A
VDD33A/NC
VDD33A
VDD33A
R385 49R9 1%
R386 49R9 1%
R387 49R9 1%
R388 49R9 1%
MDC/GPIO2
MDIO/GPIO1
nTRST/RXD0
TDO/nPHY_RST
TCK/RXD1
TMS/RXD2
TDI/RXD3
RXCLK
RXDV
TXEN
RXER
CRS/GPIO3
COL/GPIO0
TXCLK
TXD3/GPIO7/EEP_SIZE
TXD2/GPIO6/PORT_SWAP
TXD1/GPIO5/RMT_WKP
TXD0/GPIO4/EEP_DISABLE
nRESET
EECLK/PWR_SEL
EECS
EEDO/AUTOMDIX_EN
EEDI
PHY_SEL
TEST3
NC
TXP
TXN
RXP
RXN
nFDX_LED/GPIO8
nLNKA_LED/GPIO9
nSPD_LED/GPIO10
USBDP
USBDM
VBUS_DET
EXRES
USBRBIAS
TEST1
TEST2
nPHY_INT
XO
XI
EP/GND
U325
LAN9500
nLNKA_LED
nSPD_LED
D4_1_P
D4_1_N
5V_HUB
R389 10K 1%
R390 15K 1%
R391 12K 1%
R392 12K 1%
3V3_3
GND
Y303 25MHz
C418 12pF/50V
C419 12pF/50V
3V3_3
R384 10K 1%
nRST_LAN
C476 0.1uF/16V NF
GND
T_P
T_N
R_P
R_N
Ethernet Port
D308
I/O_1
GND
I/O_2
I/O_4
VCC
I/O_3
SP3051-04HTG
3V3_3
3V3A_3
R395 0R
CN304
CN_RJ-45
YLED+
YLED-
Shield
GLED+
GLED-
Shield
TCT
TD+
TD-
RD+
RD-
RCT
nSPD_LED
R393 470R 1%
nLNKA_LED
R394 470R 1%
C466 0.022uF/25V
C467 15pF/50V
C468 15pF/50V
C469 15pF/50V
C470 15pF/50V
Place these caps to close Ethernet connector
PWR
3V3_3
C450 0.1uF/16V
C451 0.1uF/16V
C452 0.1uF/16V
C453 1uF/16V
C454 4.7uF/10V
FB306 600R/0.5A
3V3A_3
C455 0.1uF/16V
C456 0.1uF/16V
C457 0.1uF/16V
C458 0.1uF/16V
C459 4.7uF/10V
VCORE_3
C460 4.7uF/10V
C461 0.1uF/16V
C462 0.1uF/16V
FB307 600R/0.5A
VPLL3
C463 0.1uF/16V
VCORE_3
FB308 600R/0.5A
VUSBPLL3
C464 0.1uF/16V
GND



Copyright © BRT Systems Pte Ltd



Copyright © BRT Systems Pte Ltd

Wi-Fi Module Schematics

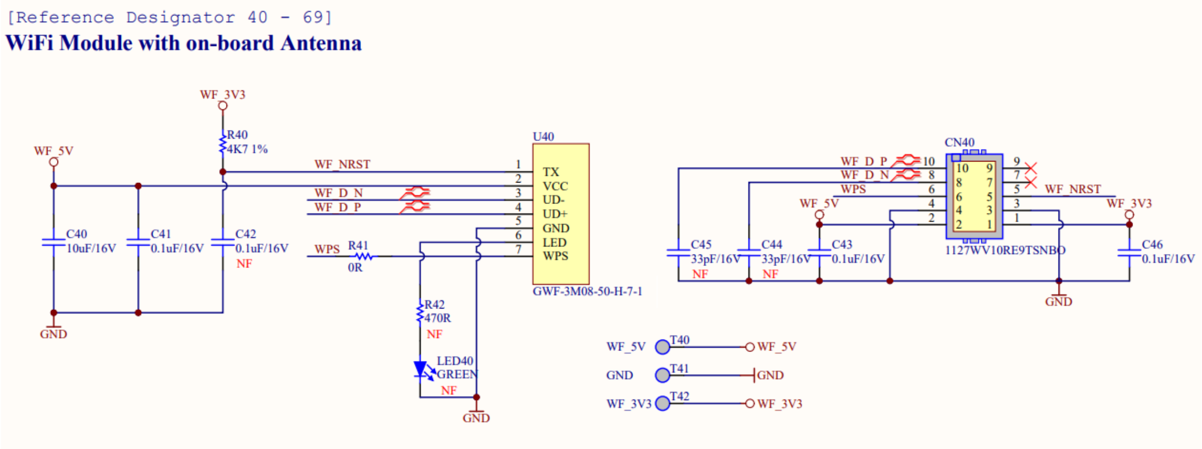


Figure 14 - Wi-Fi Module Schematics

ZigBee Module Schematics

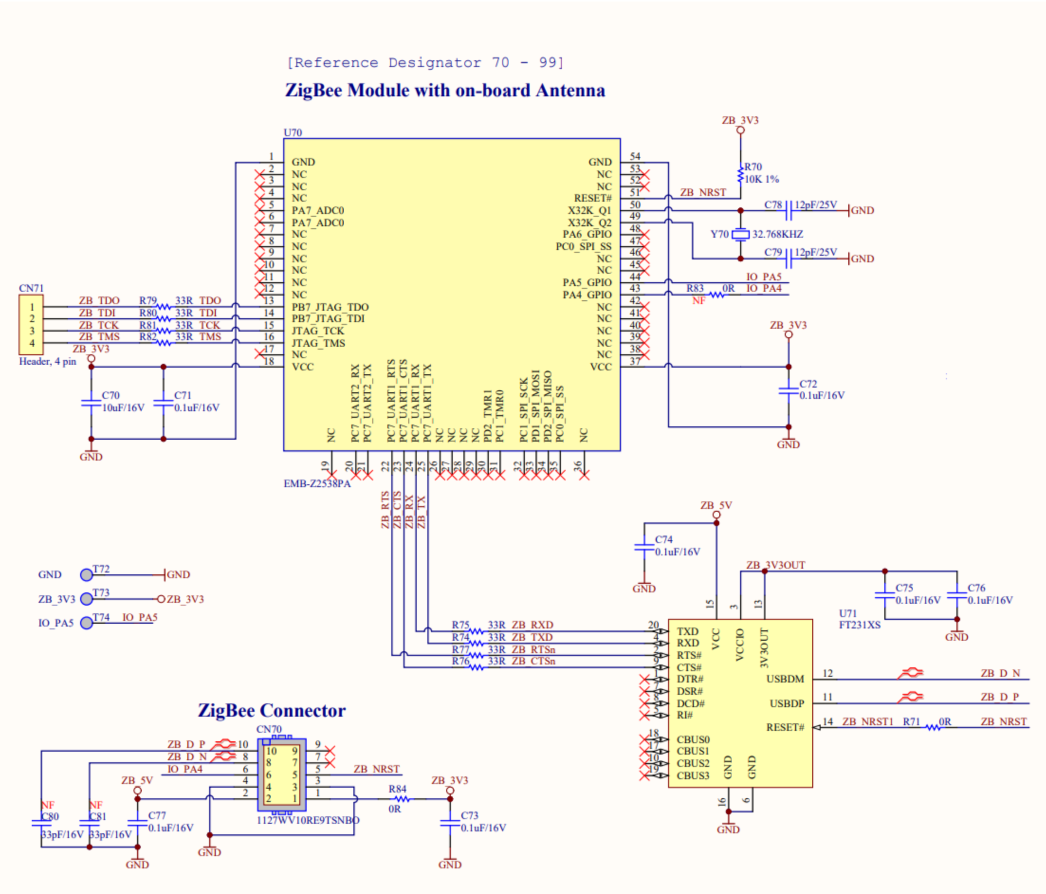


Figure 15 - Zigbee Module Schematics

Copyright © BRT Systems Pte Ltd

RF433 Module Schematics

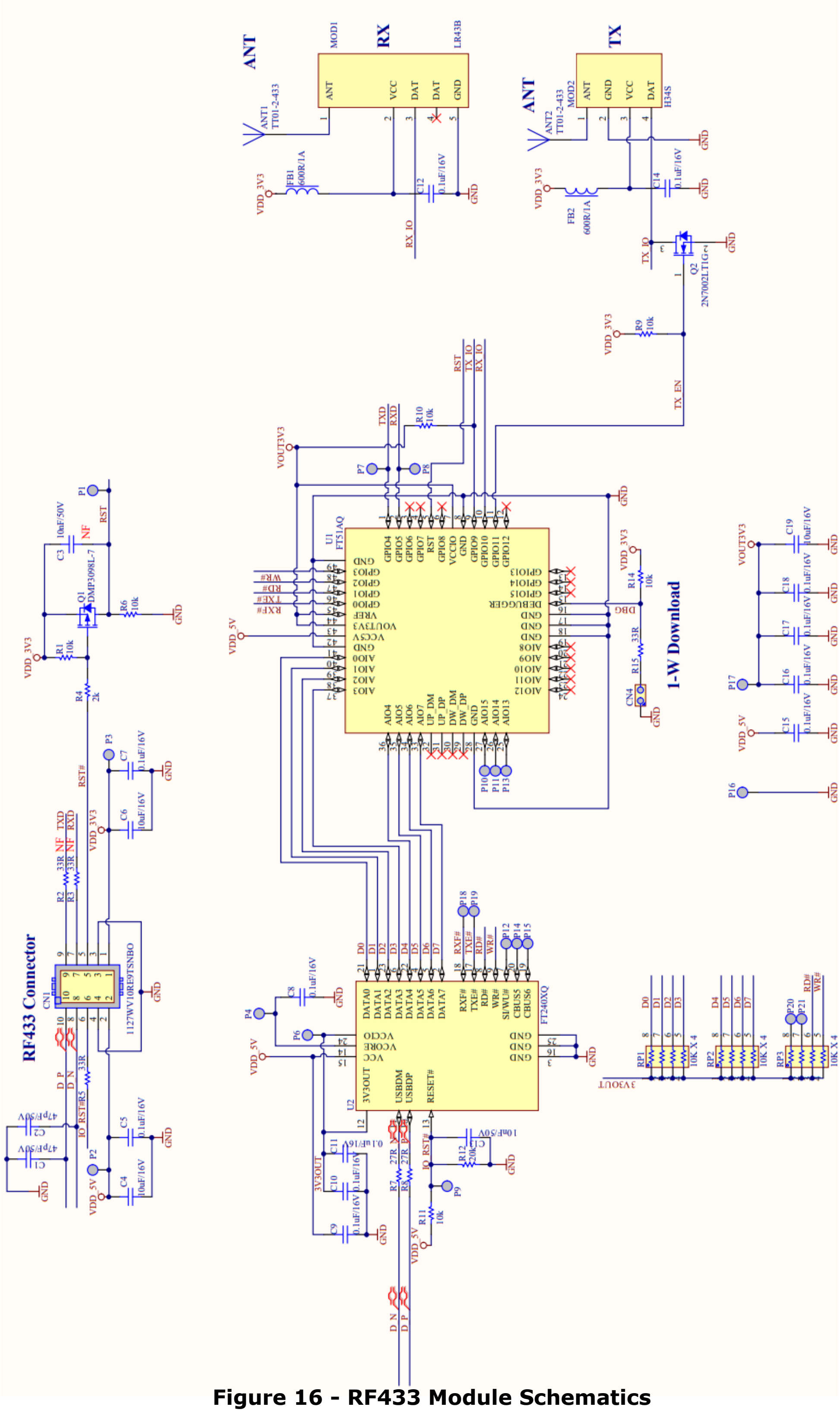
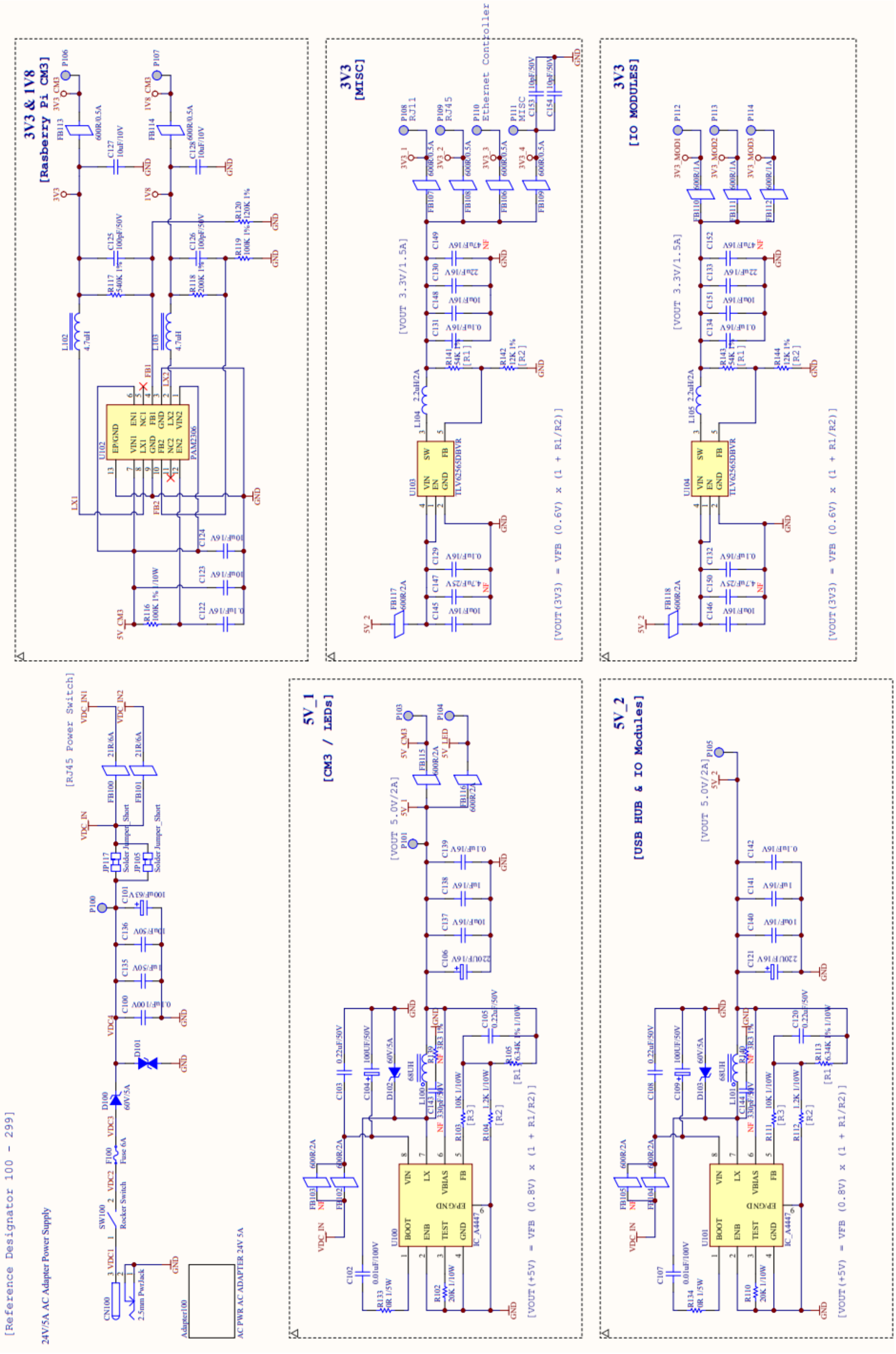
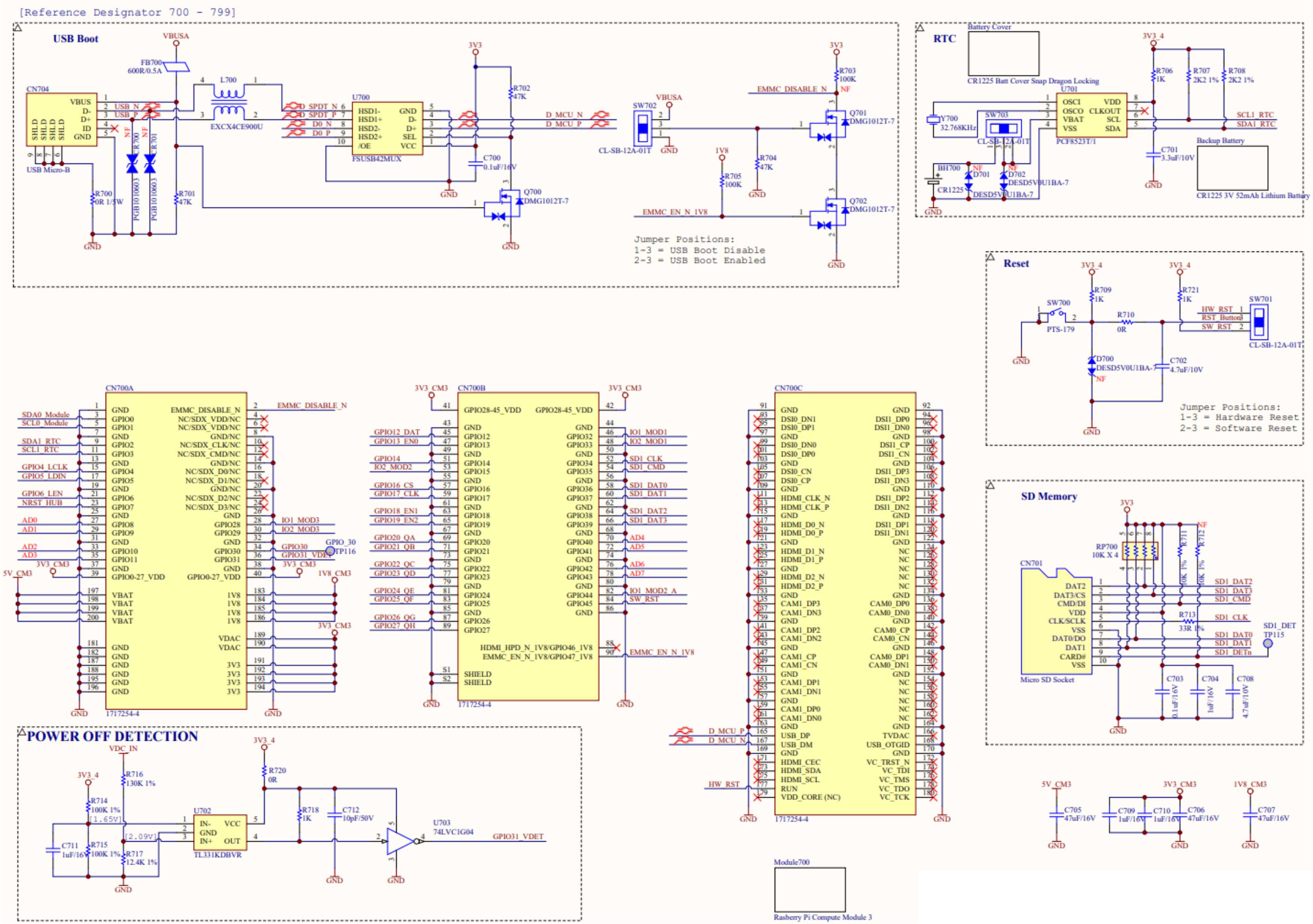


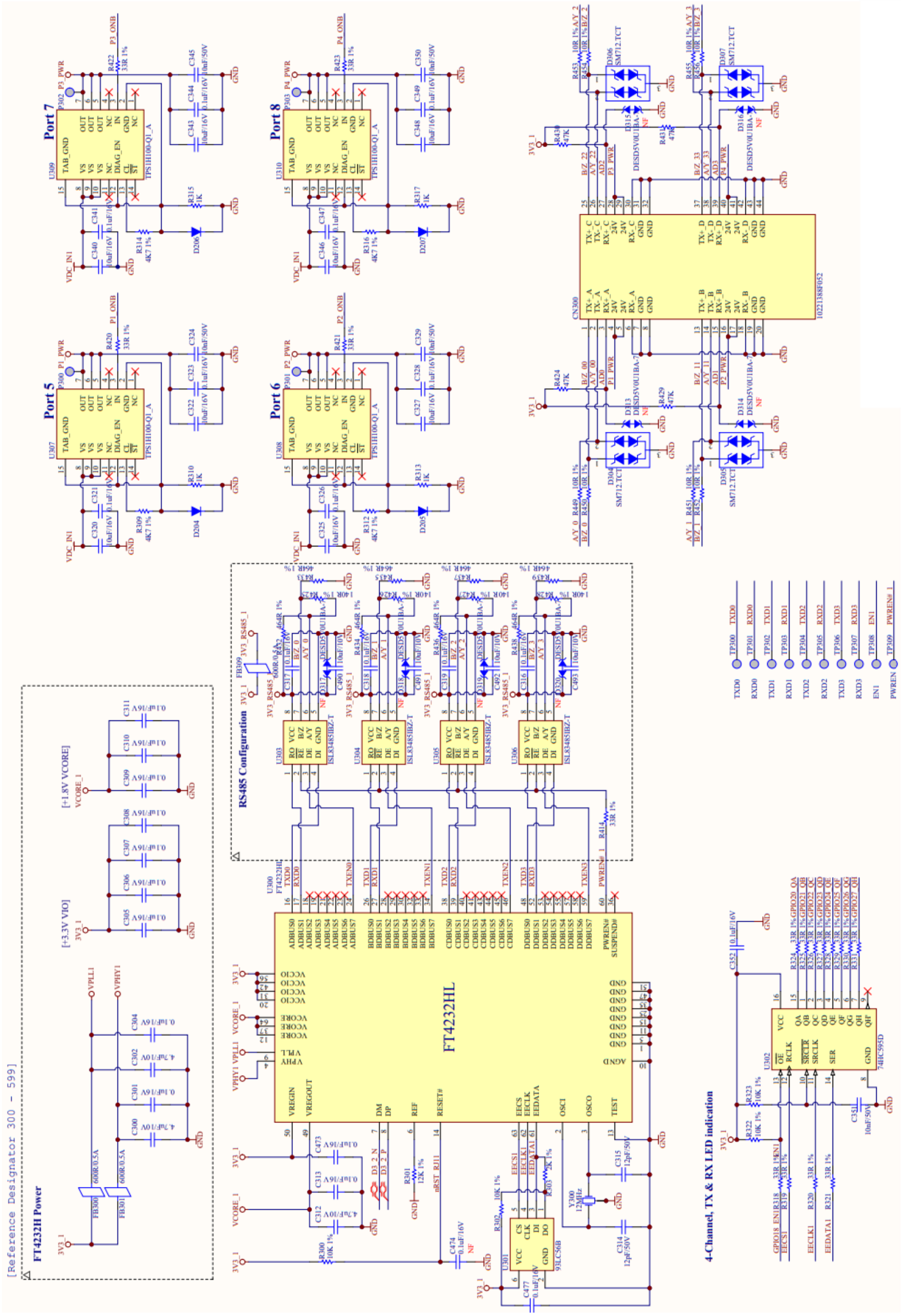
Figure 16 - RF433 Module Schematics

PanL Hub80 Board Schematics

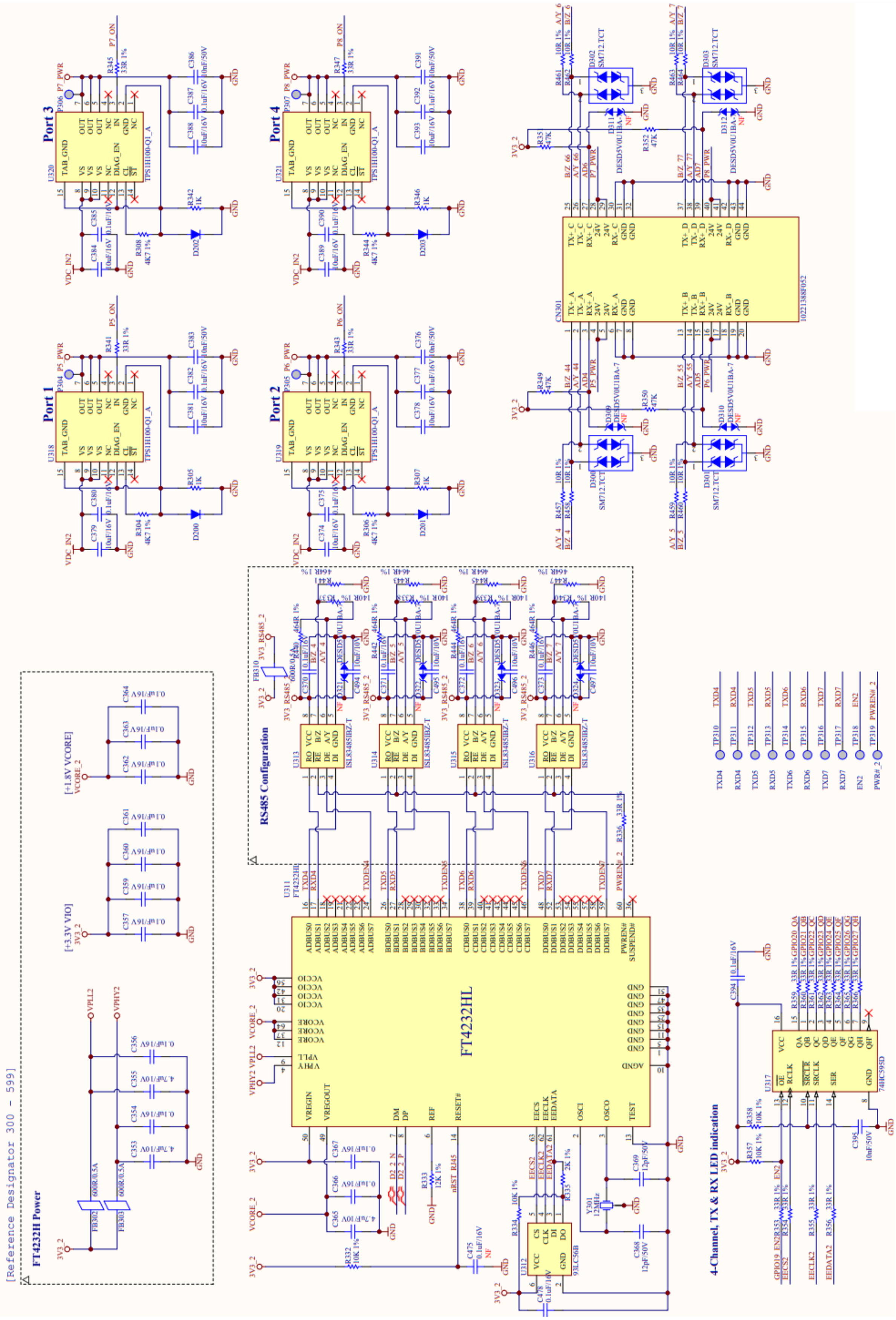
Mainboard Schematics


3V3 & 1V8 [Rasberry Pi CM3]
3V3 [MISC]
3V3 [IO MODULES]
5V_1 [CM3 / LEDs]
5V_2 [USB HUB & IO Modules]
[Reference Designator 100 - 299]
24V/5A AC Adapter Power Supply
[RJ45 Power Switch]
[VOUT 3.3V/1.5A]
[VOUT 5.0V/2A]
[VOUT (3V3) = VFB (0.6V) x (1 + R1/R2)]
[VOUT (+5V) = VFB (0.8V) x (1 + R1/R2)]
PAM2306
TLV62565DBVR
IC_A4447
Ethernet Controller

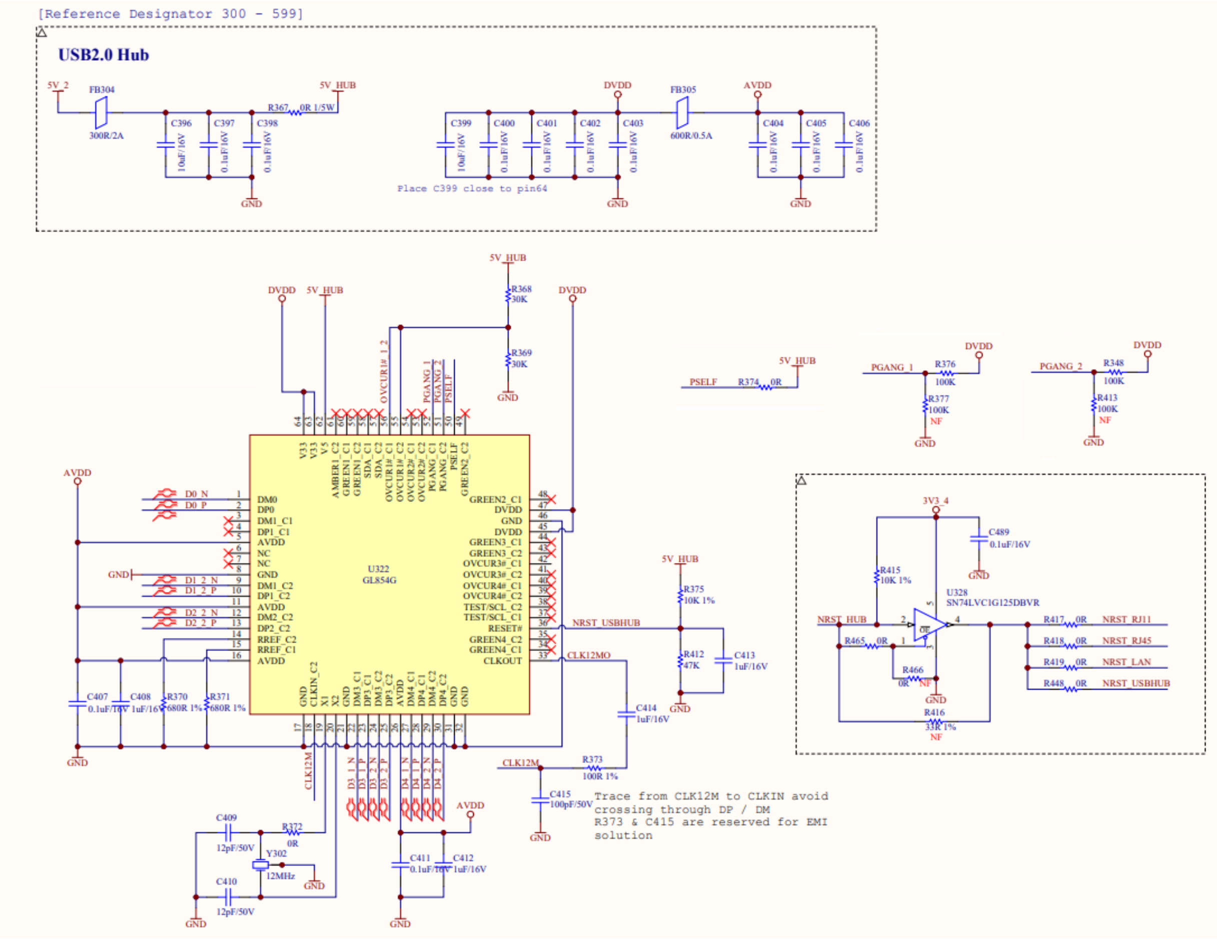

[Reference Designator 700 - 799]
USB Boot
VBUSA
FB700
600R/0.5A
CN704
VBUS
D-
D+
ID
GND
SHLD
USB Micro-B
USB_N
USB_P
NF
R700
0R 1/5W
PGB1010603
R701
47K
L700
EXCX4CE900U
U700
HSD1-
HSD1+
HSD2-
HSD2+
/OE
GND
D-
D+
SEL
VCC
FSUSB42MUX
D_SPDT_N
D_SPDT_P
D0_N
D0_P
3V3
R702
47K
C700
0.1uF/16V
Q700
DMG1012T-7
D_MCU_N
D_MCU_P
SW702
CL-SB-12A-01T
1V8
R705
100K
R704
47K
EMMC_DISABLE_N
R703
100K
Q701
DMG1012T-7
EMMC_EN_N_1V8
Q702
DMG1012T-7
Jumper Positions:
1-3 = USB Boot Disable
2-3 = USB Boot Enabled
GND
RTC
Battery Cover
CR1225 Batt Cover Snap Dragon Locking
U701
OSCI
OSCO
VBAT
VSS
VDD
CLKOUT
SCL
SDA
PCF8523T/1
Y700
32.768KHz
SW703
CL-SB-12A-01T
BH700
CR1225
D701
D702
DESD5V0U1BA-7
3V3_4
R706
1K
R707
2K2 1%
R708
2K2 1%
SCL1_RTC
SDA1_RTC
C701
3.3uF/10V
Backup Battery
CR1225 3V 52mAh Lithium Battery
Reset
SW700
PTS-179
R709
1K
R710
0R
D700
DESD5V0U1BA-7
C702
4.7uF/10V
R721
1K
HW_RST
RST_Button
SW_RST
SW701
CL-SB-12A-01T
Jumper Positions:
1-3 = Hardware Reset
2-3 = Software Reset
SD Memory
CN701
DAT2
DAT3/CS
CMD/DI
VDD
CLK/SCLK
VSS
DAT0/DO
DAT1
CARD#
Micro SD Socket
RP700
10K X 4
R711
R712
10K 1%
R713
33R 1%
SD1_DAT2
SD1_DAT3
SD1_CMD
SD1_CLK
SD1_DAT0
SD1_DAT1
SD1_DETn
SD1_DET
TP115
C703
0.1uF/16V
C704
1uF/16V
C708
4.7uF/10V
CN700A
SDA0_Module
SCL0_Module
SDA1_RTC
SCL1_RTC
GPIO4_LCLK
GPIO5_LDIN
GPIO6_LEN
NRST_HUB
AD0
AD1
AD2
AD3
3V3_CM3
5V_CM3
EMMC_DISABLE_N
IO1_MOD3
IO2_MOD3
GPIO_30
TP116
GPIO30
GPIO31_VDET
1V8_CM3
VBAT
VDAC
1717254-4
CN700B
GPIO28-45_VDD
GPIO12_DAT
GPIO13_EN0
GPIO14
IO2_MOD2
GPIO16_CS
GPIO17_CLK
GPIO18_EN1
GPIO19_EN2
GPIO20_QA
GPIO21_QB
GPIO22_QC
GPIO23_QD
GPIO24_QE
GPIO25_QF
GPIO26_QG
GPIO27_QH
IO1_MOD1
IO2_MOD1
SD1_CLK
SD1_CMD
SD1_DAT0
SD1_DAT1
SD1_DAT2
SD1_DAT3
AD4
AD5
AD6
AD7
IO1_MOD2_A
SW_RST
HDMI_HPD_N_1V8/GPIO46_1V8
EMMC_EN_N_1V8/GPIO47_1V8
SHIELD
EMMC_EN_N_1V8
CN700C
D_MCU_P
D_MCU_N
HW_RST
USB_DP
USB_DM
HDMI_CEC
HDMI_SDA
HDMI_SCL
RUN
VDD_CORE (NC)
TVDAC
USB_OTGID
VC_TRST_N
VC_TDI
VC_TMS
VC_TDO
VC_TCK
POWER OFF DETECTION
VDC_IN
R716
130K 1%
R714
100K 1%
[1.65V]
C711
1uF/16V
R715
100K 1%
[2.09V]
R717
12.4K 1%
U702
IN-
GND
IN+
VCC
OUT
TL331KDBVR
R720
0R
R718
1K
C712
10pF/50V
U703
74LVC1G04
GPIO31_VDET
C705
47uF/16V
C709
1uF/16V
C710
1uF/16V
C706
47uF/16V
C707
47uF/16V
Module700
Rasberry Pi Compute Module 3

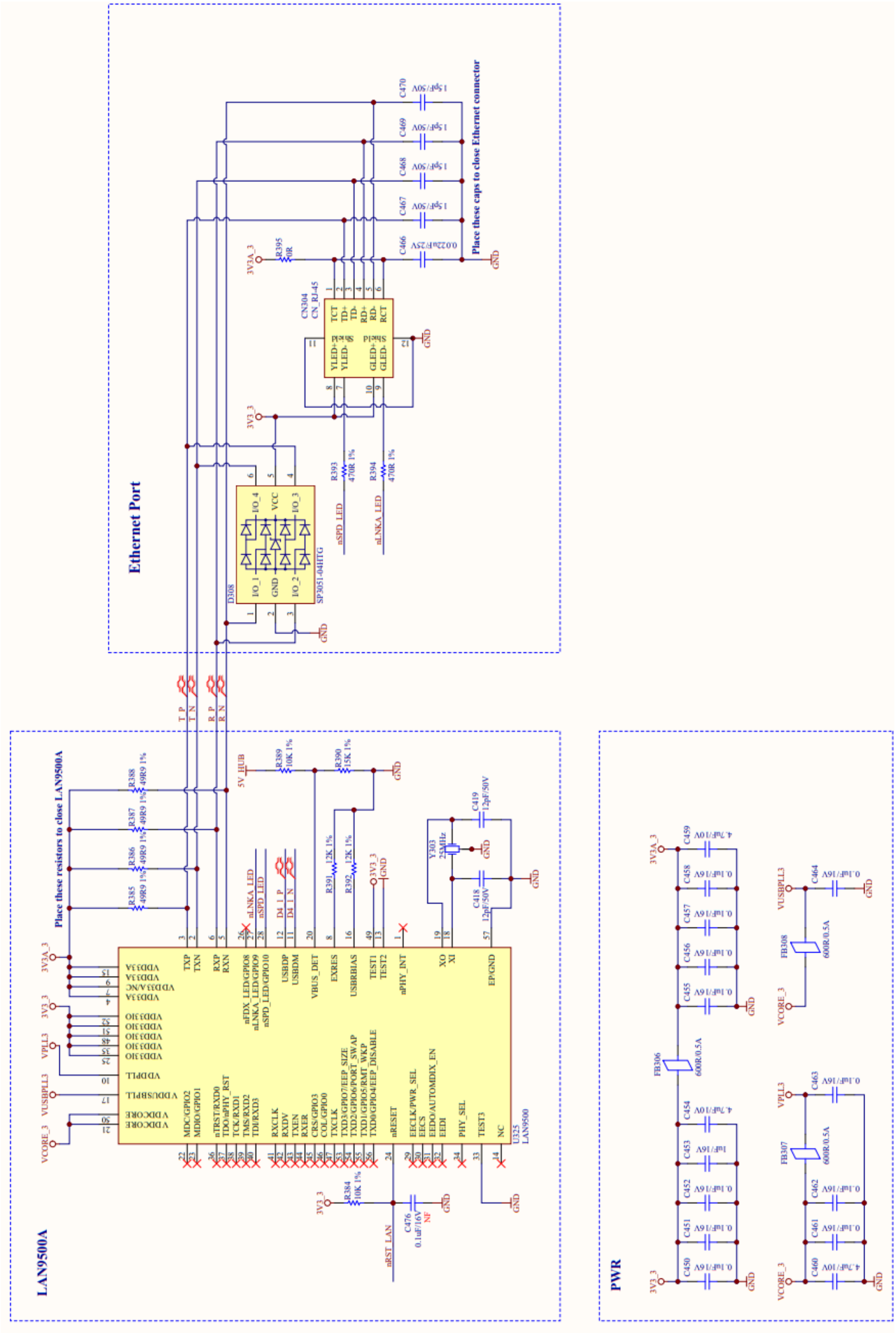


Copyright © BRT Systems Pte Ltd

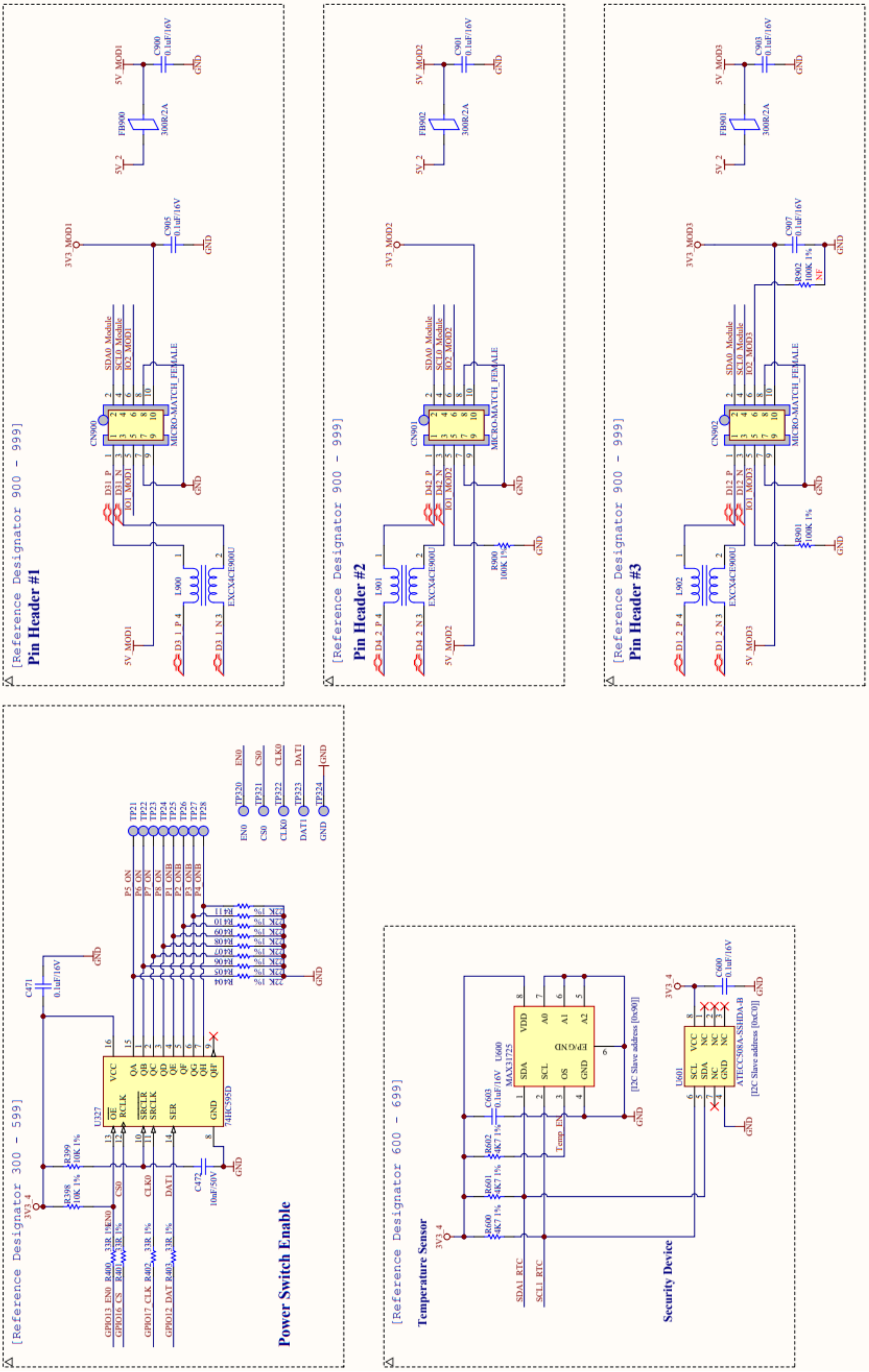


Copyright © BRT Systems Pte Ltd

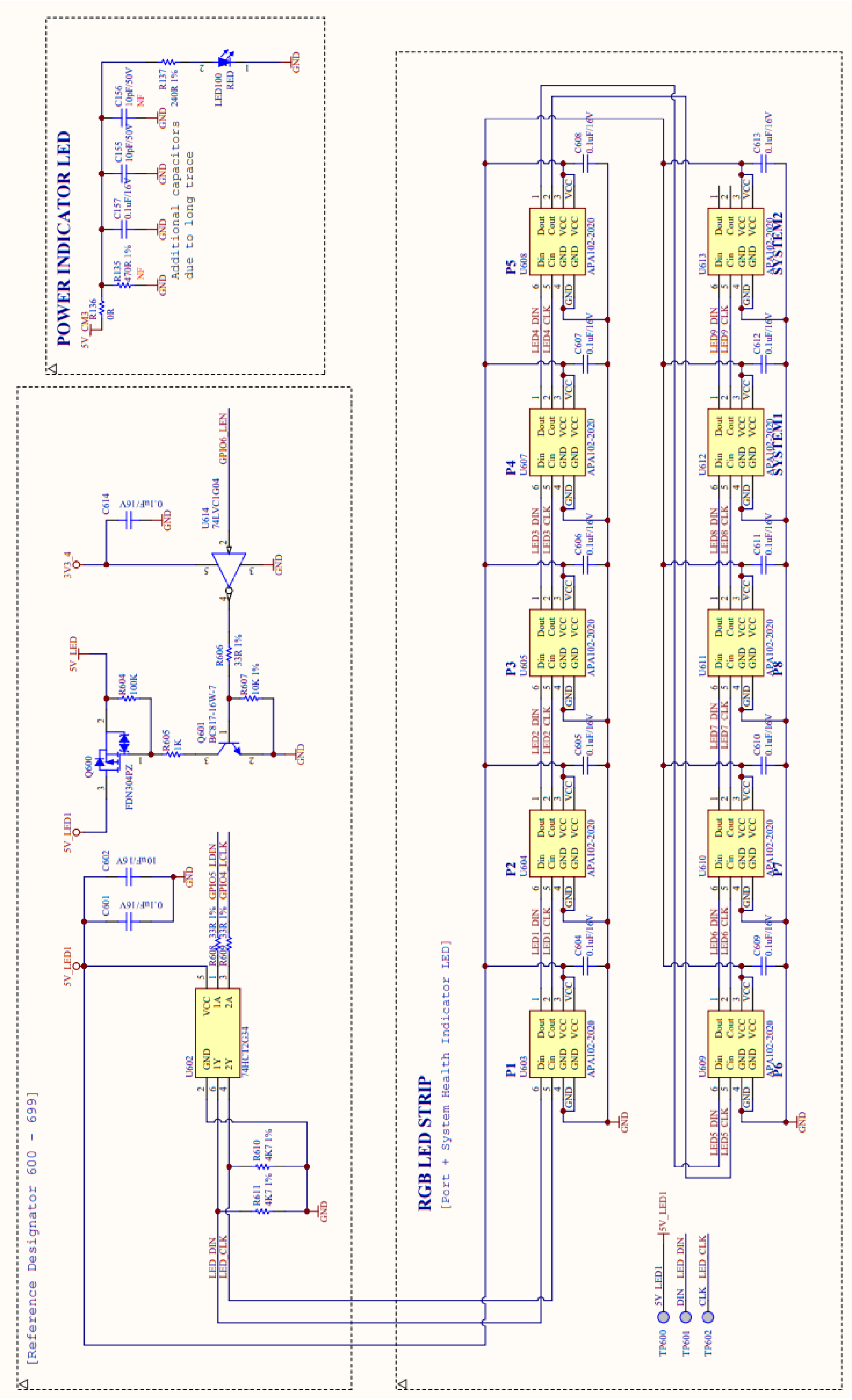

[Reference Designator 300 - 599]
USB2.0 Hub
5V_2
FB304
300R/2A
C396
10uF/16V
C397
0.1uF/16V
C398
0.1uF/16V
R367 0R 1/5W
5V_HUB
GND
DVDD
C399
10uF/16V
C400
0.1uF/16V
C401
0.1uF/16V
C402
0.1uF/16V
C403
0.1uF/16V
FB305
600R/0.5A
AVDD
C404
0.1uF/16V
C405
0.1uF/16V
C406
0.1uF/16V
Place C399 close to pin64
R368 30K
R369 30K
OVCUR1# 1 2
PGANG 1
PGANG 2
PSELF
U322
GL854G
V33
V33
V5
AMBER1_C2
GREEN1_C1
GREEN1_C2
SDA_C1
SDA_C2
OVCUR1#_C1
OVCUR1#_C2
OVCUR2#_C1
OVCUR2#_C2
PGANG_C1
PGANG_C2
PSELF
GREEN2_C2
DM0
DP0
DM1_C1
DP1_C1
AVDD
NC
NC
GND
DM1_C2
DP1_C2
AVDD
DM2_C2
DP2_C2
RREF_C2
RREF_C1
AVDD
GREEN2_C1
DVDD
GND
DVDD
GREEN3_C1
GREEN3_C2
OVCUR3#_C1
OVCUR3#_C2
OVCUR4#_C1
OVCUR4#_C2
TEST/SCL_C2
TEST/SCL_C1
RESET#
GREEN4_C2
GREEN4_C1
CLKOUT
GND
CLKIN_C2
X1
X2
GND
DM3_C1
DP3_C1
DM3_C2
DP3_C2
AVDD
DM4_C1
DP4_C1
DM4_C2
DP4_C2
GND
GND
D0_N
D0_P
D1_2_N
D1_2_P
D2_2_N
D2_2_P
D3_1_N
D3_1_P
D3_2_N
D3_2_P
D4_1_N
D4_1_P
D4_2_N
D4_2_P
C407 0.1uF/16V
C408 1uF/16V
R370 680R 1%
R371 680R 1%
CLK12M
C409 12pF/50V
R372 0R
Y302 12MHz
C410 12pF/50V
C411 0.1uF/16V
C412 1uF/16V
NRST_USBHUB
CLK12MO
C414 1uF/16V
R373 100R 1%
C415 100pF/50V
R375 10K 1%
R412 47K
C413 1uF/16V
Trace from CLK12M to CLKIN avoid crossing through DP / DM
R373 & C415 are reserved for EMI solution
PSELF
R374 0R
PGANG_1
R376 100K
R377 100K NF
PGANG_2
R348 100K
R413 100K NF
3V3_4
C489 0.1uF/16V
R415 10K 1%
U328
SN74LVC1G125DBVR
OE
NRST_HUB
R465 0R
R466 0R NF
R416 33R 1% NF
R417 0R NRST_RJ11
R418 0R NRST_RJ45
R419 0R NRST_LAN
R448 0R NRST_USBHUB



Copyright © BRT Systems Pte Ltd



Copyright © BRT Systems Pte Ltd


POWER INDICATOR LED
5V_CM3
R136
0R
R135
470R 1%
NF
C157
0.1uF/16V
C155
10pF/50V
C156
10pF/50V
NF
R137
240R 1%
LED100
RED
GND
Additional capacitors
due to long trace
[Reference Designator 600 - 699]
3V3_4
C614
0.1uF/16V
U614
74LVC1G04
GPIO6_LEN
5V_LED
R604
100K
Q600
FDN304PZ
R605
1K
Q601
BC817-16W-7
R606
33R 1%
R607
10K 1%
5V_LED1
C602
10uF/16V
C601
0.1uF/16V
U602
74HCT2G34
VCC
1A
2A
GND
1Y
2Y
R608 33R 1%
R609 33R 1%
GPIO5_LDIN
GPIO4_LCLK
R610
4K7 1%
R611
4K7 1%
LED_DIN
LED_CLK
RGB LED STRIP
[Port + System Health Indicator LED]
P1
U603
P2
U604
P3
U605
P4
U607
P5
U608
P6
U609
P7
U610
P8
U611
SYSTEM1
U612
SYSTEM2
U613
Din
Cin
GND
GND
Dout
Cout
VCC
VCC
APA102-2020
LED1_DIN
LED1_CLK
LED2_DIN
LED2_CLK
LED3_DIN
LED3_CLK
LED4_DIN
LED4_CLK
LED5_DIN
LED5_CLK
LED6_DIN
LED6_CLK
LED7_DIN
LED7_CLK
LED8_DIN
LED8_CLK
LED9_DIN
LED9_CLK
C604
C605
C606
C607
C608
C609
C610
C611
C612
C613
0.1uF/16V
5V_LED1
TP600
5V_LED1
TP601
DIN
LED_DIN
TP602
CLK
LED_CLK

Wi-Fi Module Schematics

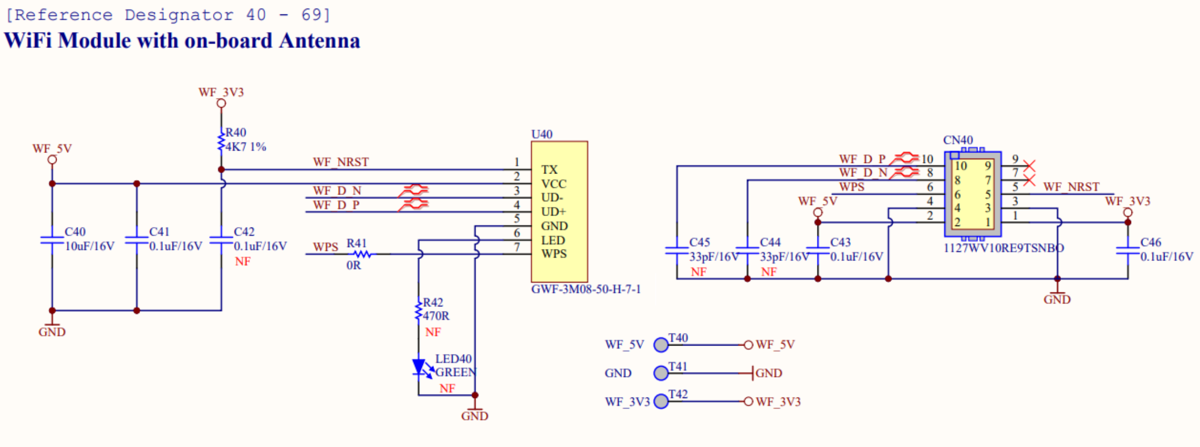


Figure 17 - Wi-Fi Module Schematics

ZigBee Module Schematics

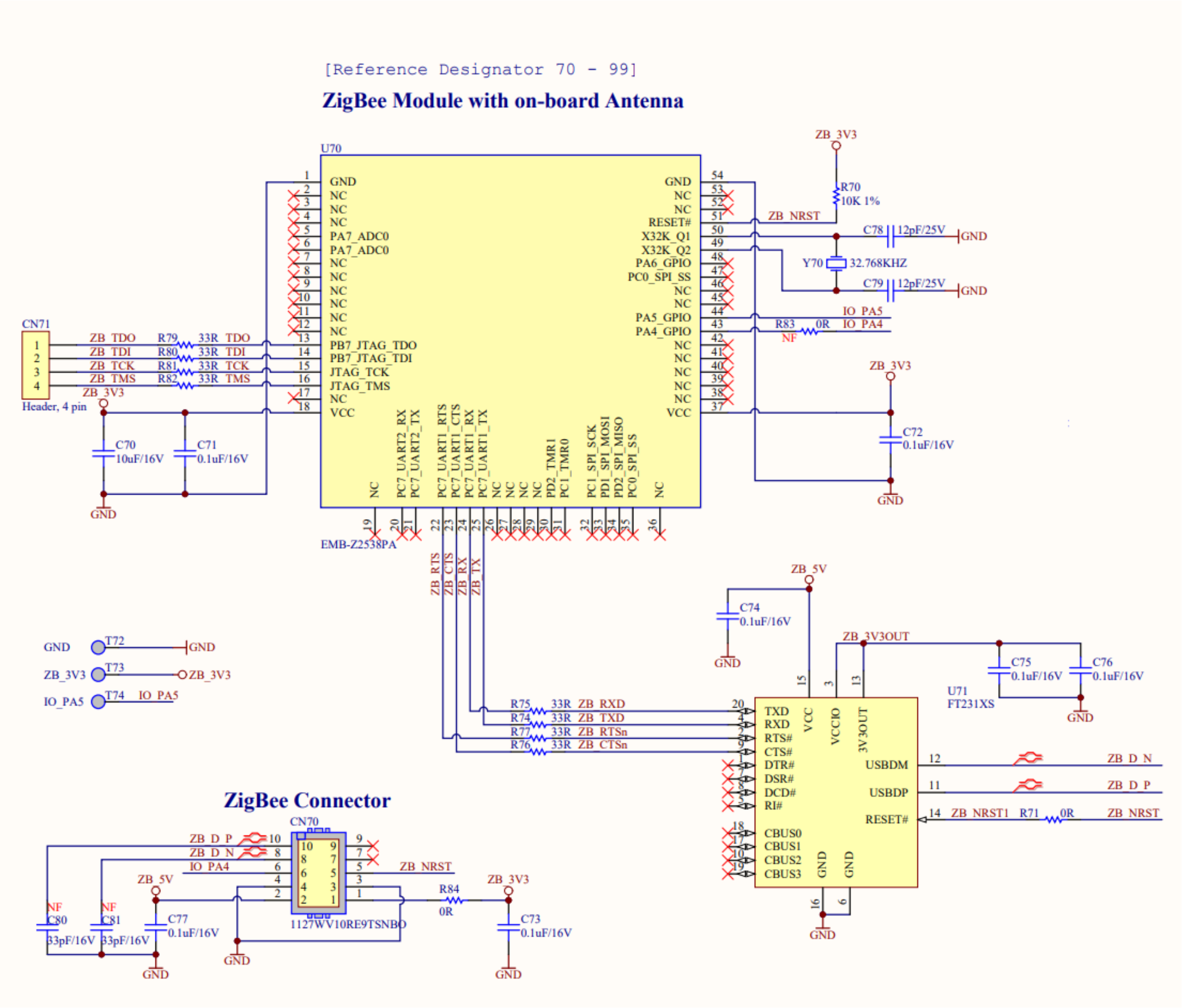


Figure 18 - Zigbee Module Schematics

Copyright © BRT Systems Pte Ltd

RF433 Module Schematics

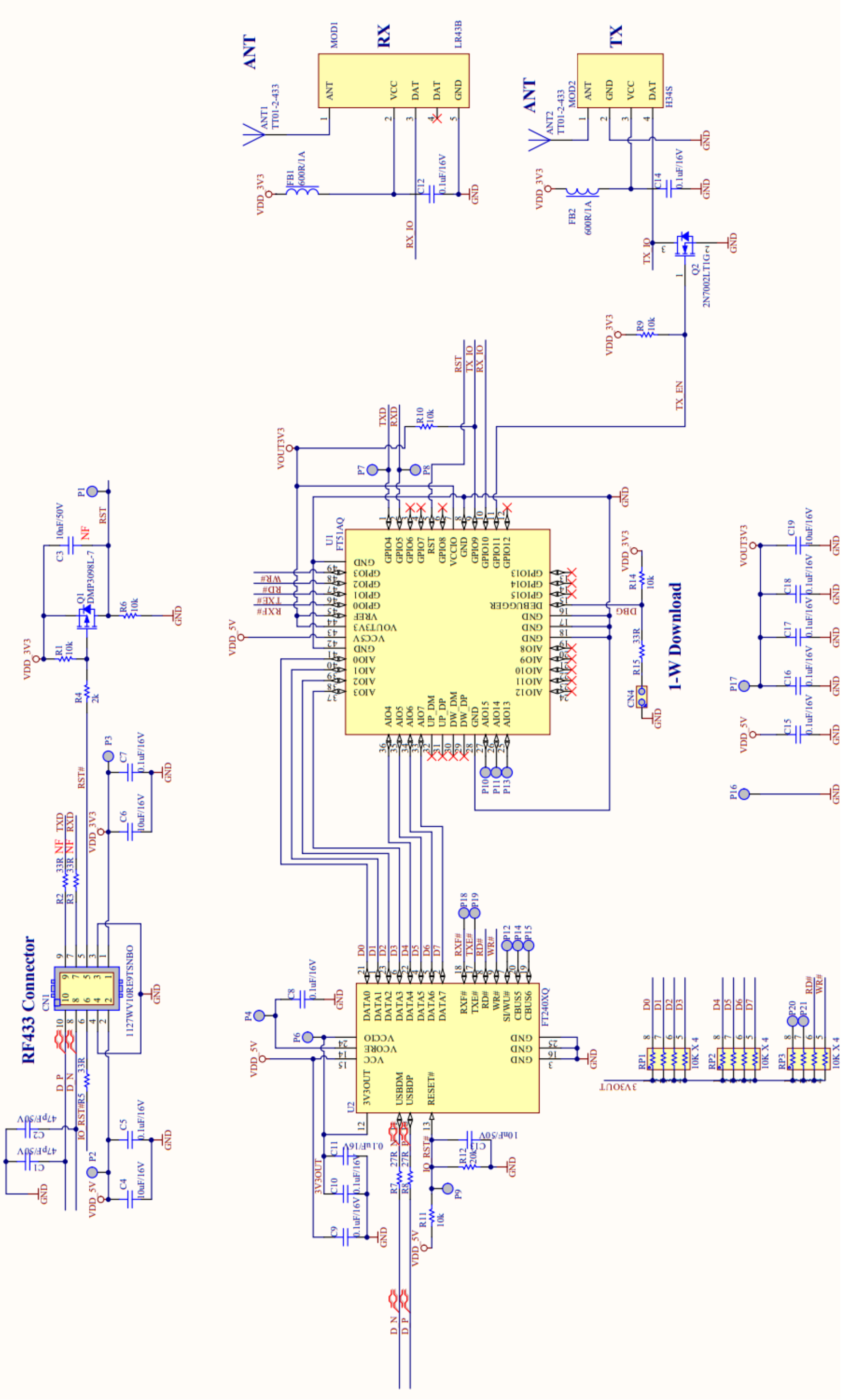


Figure 19 - RF433 Module Schematics

Copyright © BRT Systems Pte Ltd

Appendix C – References

Document References

NA

Acronyms and Abbreviations

Terms	Description
API	Application Programming Interface
ARM	Advanced RISC Machine
eMMC	Embedded MultiMediaCard
GPIO	General Purpose Input/Output
IEEE	Institute of Electrical and Electronics Engineers
LED	Light Emitting Diode
LPDDR	Low Power Double Data Rate
Mbps	Million bits per second
MCU	Microcontroller Unit
OS	Operating System
PCB	Printed Circuit Board
RAM	Random Access Memory
RGB	Red Green Blue
RTC	Real-Time Clock
SDK	Software Development Kit
SOC	System On Chip
SSH	Secure Shell

Copyright © BRT Systems Pte Ltd

Appendix D – List of Figures & Tables

List of Figures

List of Tables

Copyright © BRT Systems Pte Ltd

Appendix E – Revision History

Document Title:	BRTSYS_AN_093 PanL Hub Developers' Guide
Document Reference No.:	BRTSYS_000221
Clearance No.:	BRTSYS#143
Product Page:	PanL Hub - BRT Systems Pte Ltd
Document Feedback:	Send Feedback

Revision	Changes	Date
Version 1.0	Initial Release	18-12-2025

Copyright © BRT Systems Pte Ltd