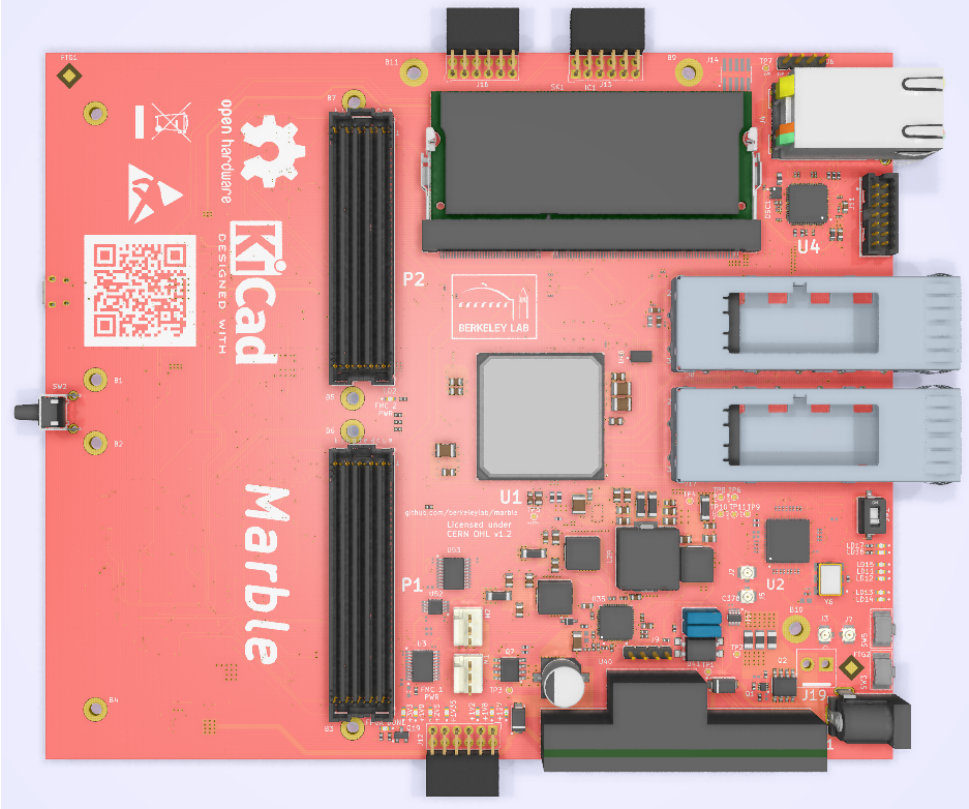
Marble – Factory Acceptance Tests

v1.0 2020

May 14, 2021



Introduction

This document presents a user–friendly guide on how to perform Factory Acceptance Tests (FAT) for the Marble module. Tests begin from a short description of where Marble's useful connector, LED indicators, and test points are placed on the board. It is highly recommended to perform the tests in the order presented in this document.

Contents

1 Overview to Marble module

> Design files are open source and can be downolad from Gilhub:
> https://github.com/BerkeleyLab/Marble

> Each board is marked with a revision number. Make sure that the downloaded files correspond to your board.

1.1 Hardware requirements

To perform all of the tests following hardware are required:

1. Lab bench power supply.
2. Micro USB calbe.
3. QSFP loopback module.
4. FMC Tester module.
5. Multimeter.
6. MMC JTAG (example: SEGGER J-LINK mini).
7. FPGA JTAG (example: Digilent JTAG HS3).

1.2 Software requirements

To perform all of the tests following software are required:

1. Vivado 19.1.
2. Serial port terminal.

2 Power connection

Before connecting the power supply for the first time, check that the main bus is not shorted. Using a multimeter set in resistance measurement mode, measure the resistance between metal pads of the J19 connector (Figure 1).

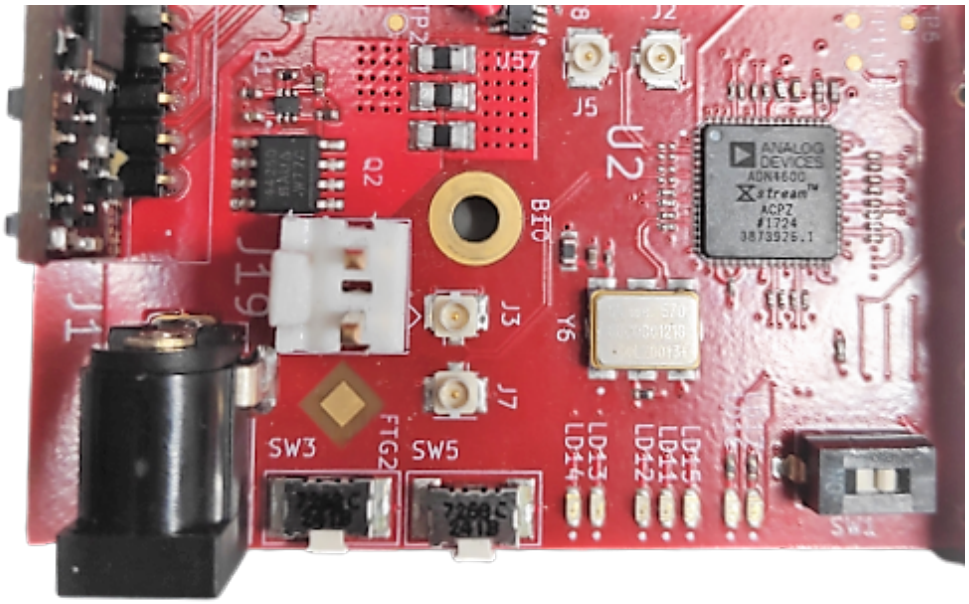


Figure 1: Connector's terminals.

- Measured resistance should be around **200 kOhm**

If the resistance is correct, connect the main power. For this purpose, the current limitation on the power supply should be set to 100mA and the voltage to 12V. **Make sure that the current limit of the laboratory power supply is on.** Now the power cable can be connected to the board and the used laboratory power supply channel can be switched on. This should result in the *12V* LED lighting up as shown on Figure 2. Now it is recommended to go to section Microcontroller programing.

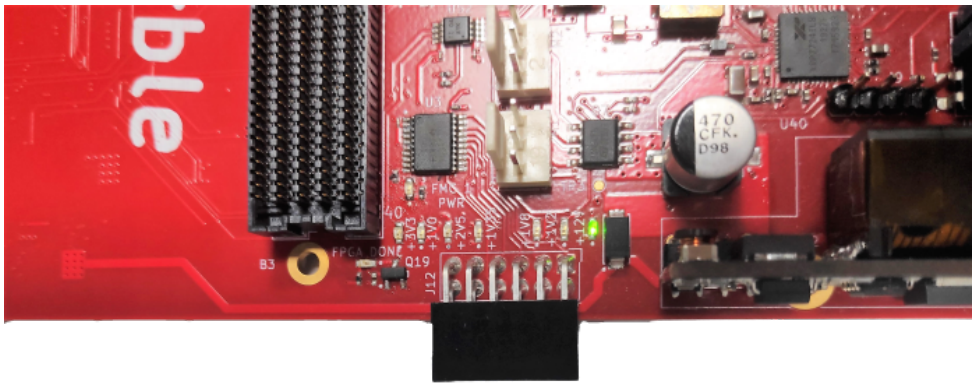


Figure 2: 12V indicator LED is on.

3 Microcontroller programing

Download the latest version of the microcontroller testing code from github:

https://https://gitlab.lbl.gov/spaiagua/marble_mmc/-/tree/i2c_rework

A recent version of OpenOCD (v0.10.0 or later) is required.

1. Connect JTAG module to **J14** like it is shown on Figure 3.
2. Connect the micro USB cable and using the serial terminal, connect to the last serial port for the new listed in the operating system. Use 115200 boudrate.
3. Power up the board.
4. Program the microcontroller using the following commands:
   (a) Go to the main folder of the downloaded repository.
   (b) Open command terminal and run command:

```
$ make download
```

5. After the successful programming, a menu in the serial terminal should appeared and LEDs (LD15, LD11, LD12) should blink in the "snake" pattern.

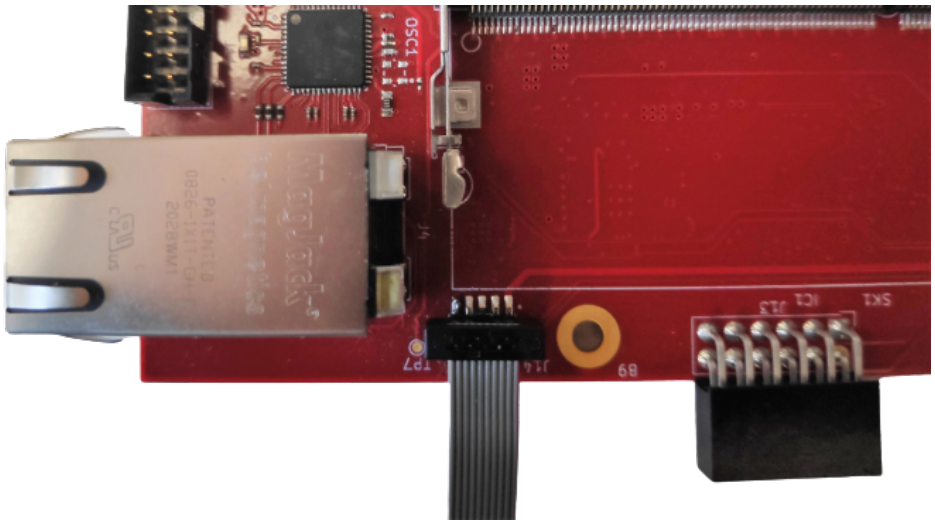


Figure 3: J14 connector.

4 Power Supply programing

Before doing the steps below, it is highly recommended to measure if there are no shorts on power rails. Measure resistance between the test points:

1. **TP12 (GND)** and **TP7 (+2V0)**.
2. **TP12 (GND)** and **TP9 (+1V0)**.

3. **TP12 (GND)** and **TP10 (+2V5)**.
4. **TP12 (GND)** and **TP8 (+1V8)**.
5. **TP12 (GND)** and **TP6 (+1V5)**.
6. **TP12 (GND)** and **TP4 (+1V2)**.
7. **TP12 (GND)** and **TP11 (+3V3)**.
8. **TP12 (GND)** and **TP11 (+3V3USB)**.
9. **TP12 (GND)** and **TP11 (+3V3P)**.
10. **TP12 (GND)** and **TP5 (+1V05)**.

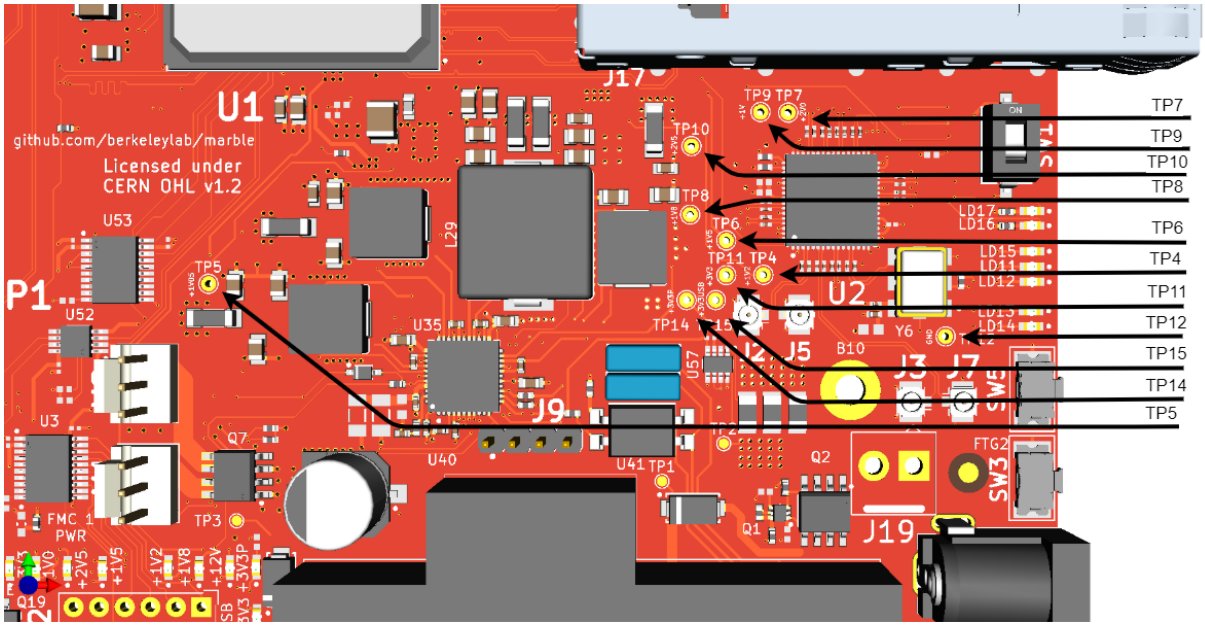


Figure 4: Test points.

Perform the following steps only and exclusively when there are no shorts on the power rails!

1. Connect the micro USB cable and using the serial terminal, connect to the last serial port for the new listed in the operating system. Use 115200 boudrate.
2. Set up votage to 12V and the current limit to 1A on lab power supply.
3. Power up the board.
4. From the menu displayed in the serial terminal, select the option `g) XRP7724 go`.
5. Make a power cycle by turning off and on the lab power supply.
6. All power LED indicators should be on (Figure 5).

7. Using multimeter measure voltage between points:
   (a) **TP12 (GND)** and **TP7 (+2V0)** - expected voltage: +2.0V.
   (b) **TP12 (GND)** and **TP9 (+1V0)** - expected voltage: +1.0V.
   (c) **TP12 (GND)** and **TP10 (+2V5)** - expected voltage: +2.5V.
   (d) **TP12 (GND)** and **TP8 (+1V8)** - expected voltage: +1.8V.
   (e) **TP12 (GND)** and **TP6 (+1V5)** - expected voltage: +1.5V.
   (f) **TP12 (GND)** and **TP4 (+1V2)** - expected voltage: +1.2V.
   (g) **TP12 (GND)** and **TP11 (+3V3)** - expected voltage: +3.3V.
   (h) **TP12 (GND)** and **TP11 (+3V3USB)** - expected voltage: +3.3V.
   (i) **TP12 (GND)** and **TP11 (+3V3P)** - expected voltage: +3.3V.
   (j) **TP12 (GND)** and **TP5 (+1V05)** - expected voltage: +1.05V.

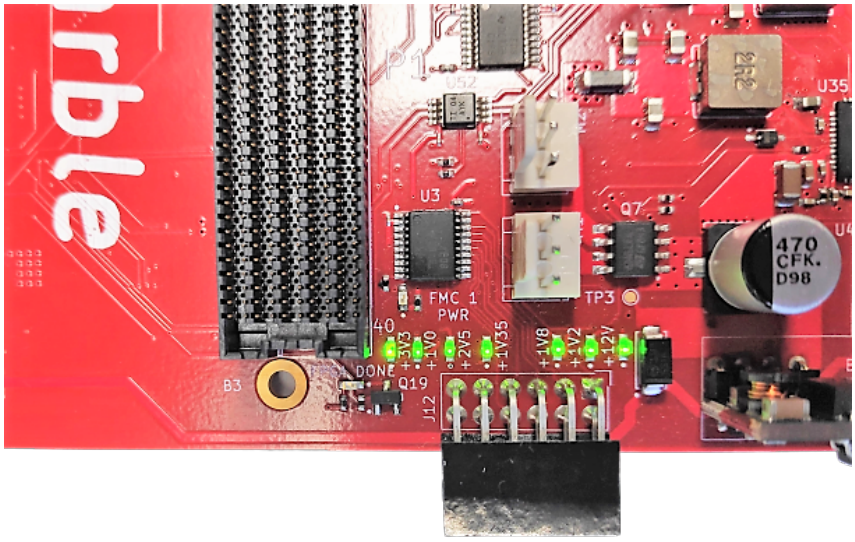


Figure 5: Power LED indicators after a successful power cycle.

5 FPGA programing

Download the latest version of the FPGA testing code from github:

> https://github.com/BerkeleyLab/Bedrock/tree/marblev2

> Before testing the FPGA it is recommended to set up the current limit to 2A on the lab power supply.

5.1 FMC test

Board power should be turned off when inserting and removing the FMC module.

1. Plug FMC Tester module to one of the FMC connector as it is shown on Figure ??.
2. Connect the micro USB cable and using the serial terminal, connect to the last serial port for the new listed in the operating system. Use 115200 boudrate.
3. Change the network adapter settings to connect with static **192.168.9.10** IP address.
4. Connect Marble to the computer using an ethernet cable.
5. Power up the board.
6. In the serial terminal menu choose 4) GPIO control > a) FMC power to turn on power for FMCs.
7. Program the FPGA using the following steps:
   (a) Go to the folder **Bedrock/projects/test_marble_family/**
   (b) Open command termianal and run command:

```
$ mutil usb
```

8. To make a test do the following steps:
   (a) Go to the folder **Bedrock/projects/test_marble_family/**
   (b) To test P1 FMC connector run the following command:

```
$ PYTHONPATH=../../badger:../../peripheral_drivers
/i2cbridge python fmc_test_c.py --ip $IP --fmc 0
```

   (c) To test P2 FMC connector run the following command:

```
$ PYTHONPATH=../../badger:../../peripheral_drivers
/i2cbridge python fmc_test_c.py --ip $IP --fmc 1
```

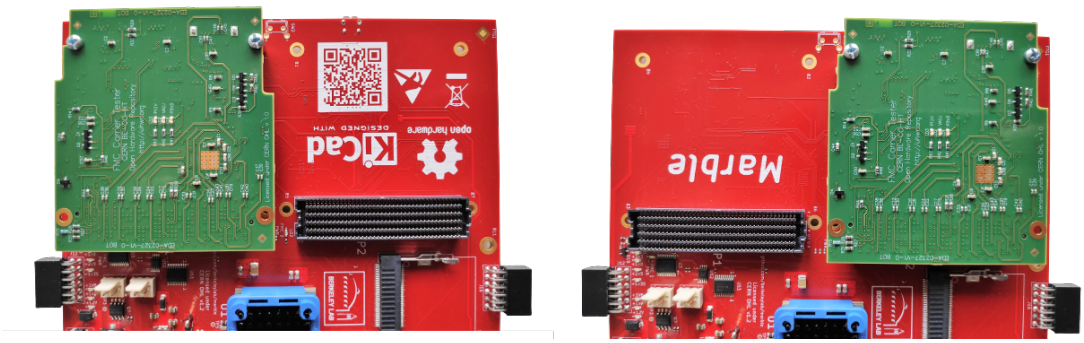


Figure 6: FMC connectors with plugged FMC Tester board.

5.2 QSFP test

Board power should be turned off when inserting and removing the QSFP module.

1. Plug QSFP loopback module to one of the ports.
2. Plug FPGA JTAG module.
3. Configure FPGA using `marble_ibert.bit` bit file.
   (a) Run Vivado
   (b) Go to Flow > Open Hardware Manager and then Tools > Auto Connect
   (c) Click Tools > Program Device > xc7k160t_0 to open the programming window.
   (d) Choose the *bitstream file* and click Program
4. After the successful programming, the Dashboard should start automatically.
5. Detect links by clicking Serial I/O Links > Auto-detect links
6. Correct detected and working links should look like in figure 7.
7. Connect the QSFP loopback module to the other QSFP connector and repeat the above steps.

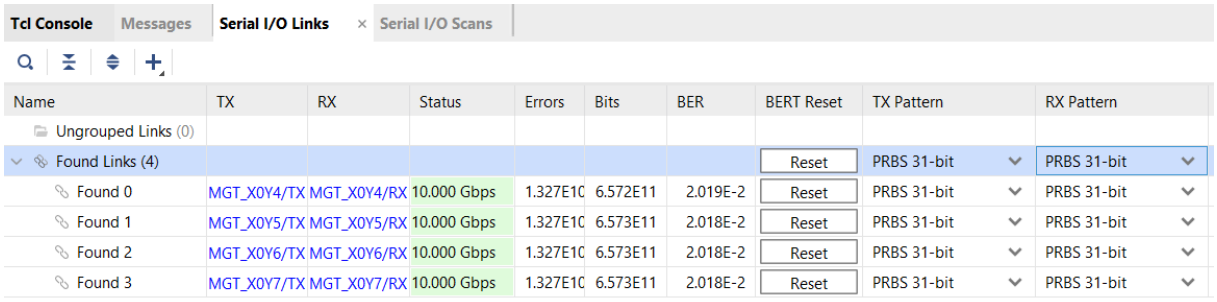


Figure 7: Working links.

5.3 Ethernet test

The following steps should be performed to test the Ethernet:

1. Program the FPGA using the following steps:
   (a) Go to the folder **Bedrock/projects/test_marble_family/**
   (b) Open command termianal and run command:

```
$ mutil usb
```

2. Change the network adapter settings to connect with static **192.168.9.10** IP address.
3. Connect Marble to the computer using an ethernet cable.
4. In Linux command terminal run:

```
$ ping 192.168.9.10
```

5. Expected result:

```
Pinging [192.168.9.10] with 32 bytes of data:
Reply from 192.168.9.10: bytes=32 time=17ms TTL=55
Reply from 192.168.9.10: bytes=32 time=152ms TTL=55
Reply from 192.168.9.10: bytes=32 time=12ms TTL=55
Reply from 192.168.9.10: bytes=32 time=14ms TTL=55

Ping statistics for 192.168.9.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 12ms, Maximum = 152ms, Average = 48ms
```

6. For data traffic test use *iperf*. Run command:

```
$ iperf -c 192.168.9.10 -u -b 1000M -p 802
```

6 Appendix

6.1 An alternative way to program FPGA

Programming FPGA using Vivado and Digilent JTAG HS3:

1. Run Vivado
2. Go to Flow » Open Hardware Manager and then Tools » Auto Connect
3. Click Tools » Program Device » xc7k160t_0 to open the programming window.
4. Choose the *bitstream file* and click Program
5. After the successful programming, the Dashboard should start automatically.

References