
[View Gallery](#)

1.5k 2 8 11

## TEAM (1)

 Manuel Tosone

Join this project's team

 [GitHub - Linux Capable Business Card](#)

 **HARDWARE**

 **ONGOING PROJECT**

**BUSINESS CARD**

**ARM**

**LINUX**

**PCB**

**BUILDROOT**

This project was created on 08/18/2021 and last updated a year ago.

## DESCRIPTION

- Microchip AT91SAM9N12 microprocessor, ARM926 core running at 400MHz
- 256MB RAM DDR2-533
- 128MB of NAND FLASH
- micro SD-Card slot
- 8x LEDs

PCB design, assembling, and testing.

## DETAILS

### Block Diagram

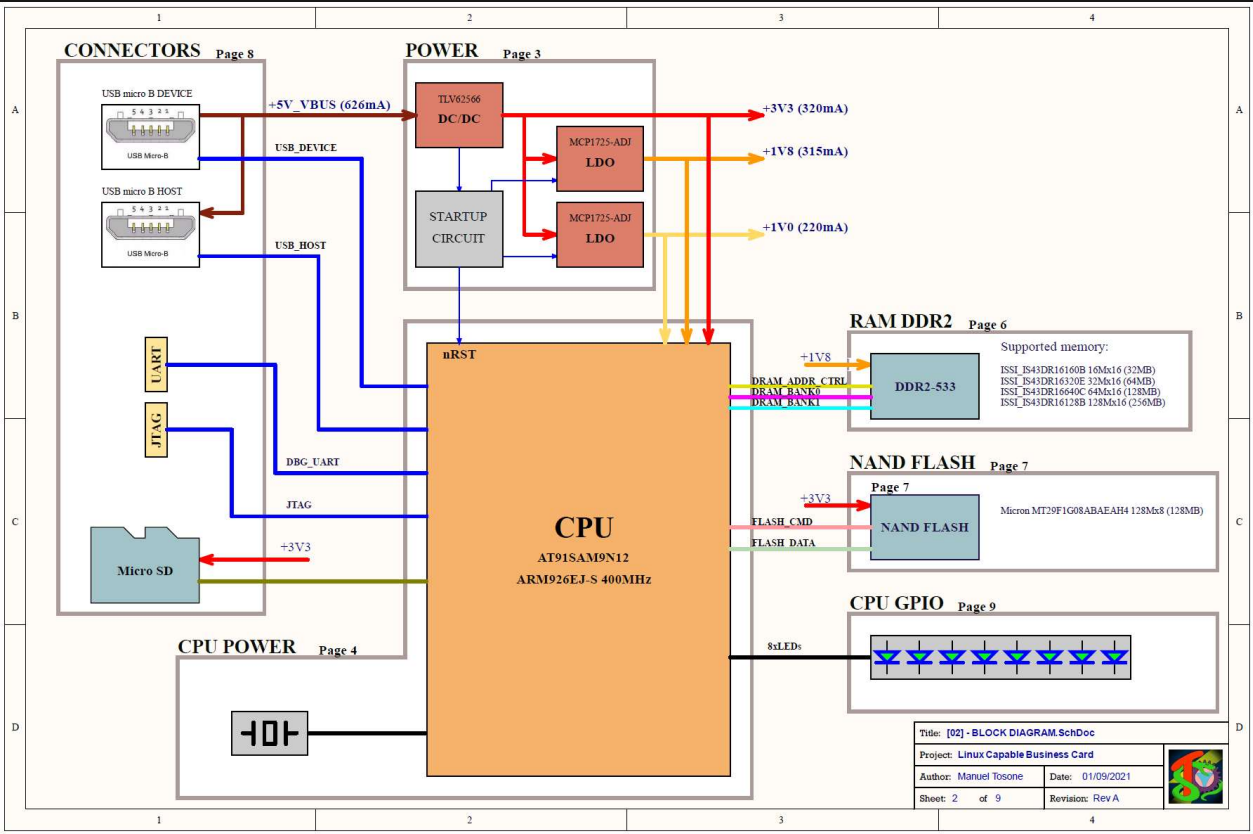
I wanted to design a more complex board, something with DDR memory and a CPU that can run Linux. This board includes a 400MHz ARM processor with DDR2 RAM and NAND FLASH. I don't care about the usefulness of this board, it only serves as practice with high-speed design.

The circuit is very simple, it is built around the **AT91SAM9N12** MPU. The board supports up to 256MB of DDR2 RAM and 128MB NAND FLASH. There are two USB ports one HOST and one DEVICE. The DEVICE port is used to load the firmware and connect to a Linux terminal through a virtual com port. The HOST port can be used to connect external peripherals to the board (wifi dongle for example). For debugging there are UART and

JTAG connections. The micro SD card can be used as external storage or as a boot device.

The power supply is a little more complicated. The USB 5V is lowered to 3.3V with a buck converter. The 3.3V is then lowered to 1.8V and 1V with linear LDO regulators. To ensure reliable operation the power rails need to come up in a specific order. The enable and power-good signals from the regulators are used to generate the proper sequence (first 3.3V, then 1.8V, last 1V) and to keep the CPU in reset until the power is stable.

The schematic document can be found [here](#).



## PCB Considerations

To keep the cost down I want to make a 4 layer board. To manufacture the boards I'll use the JLCPCB service since it is the cheapest that I know of that can manufacture 3.5mil (0.09mm) minimum track width and spacing. To route the DDR interface properly, all the tracks from the two data banks must be routed on the same layer with a solid ground plane underneath. To achieve that on a 4 layer board it is necessary to route two tracks in between BGA pads. BGA packages on this board have a 0.8 mm pin pitch. Making the pads 0.35mm in diameter leaves just enough space for two tracks with the minimum width and spacing.

Copper weight	Min. Trace width	Min. Spacing	Patterns
H/HOZ (Inner layer)	5mil (0.127mm)	5mil (0.127mm)	
1oz (Outer layer)	1/2 layers: 5mil (0.127mm) 4/6 layers: 3.5mil(0.09mm)	1/2 layers: 5mil (0.127mm) 4/6 layers: 3.5mil(0.09mm)	
2oz (Outer layer)	8mil (0.2mm)	8mil (0.2mm)	

## Stackup

For controlled impedance stackups there are two options, they differ in the prepreg and core thickness. For this project, I'll go with the second option (JLC2313 Stackup). Having

the reference planes closer to the signal layers reduces crosstalk and, for a given impedance, the traces are smaller.

The 4 layers are utilized in the following manner:

- Layer 1: Almost all signals, RAM data banks are routed only on the top layer with a solid ground plane directly below, on layer 2.
- Layer 2: Solid ground plane.
- Layer 3: Power plane. The 1.8V power plane is the reference for the RAM address command and control signals.
- Layer 4: RAM address command and control signals referencing the power plane above. Ground pour to balance construction.

#### 4-Layer Impedance Control Stackup

Thickness

0.8mm

1.0mm

1.2mm

1.6mm

2.0mm

Current layer: 4-layer

a) JLC7628 Stackup:

Layer	Material Type	Thickness	
Top Layer1	Copper	0.035 mm	
Prepreg	7628*1	0.2 mm	
Inner Layer2	Copper	0.0175 mm	0.5 mm (with copper core)
Core	Core	0.465 mm	
Inner Layer3	Copper	0.0175 mm	
Prepreg	7628*1	0.2 mm	
Bottom Layer4	Copper	0.035 mm	

b) JLC2313 Stackup:

Layer	Material Type	Thickness	
Top Layer1	Copper	0.035 mm	
Prepreg	2313*1	0.1 mm	
Inner Layer2	Copper	0.0175 mm	0.7 mm (with copper core)
Core	Core	0.665 mm	
Inner Layer3	Copper	0.0175 mm	
Prepreg	2313*1	0.1 mm	
Bottom Layer4	Copper	0.035 mm	

For signal integrity, it is required to control the impedance on high-speed interfaces. In other words, the tracks must be made a specific width that's based upon the stackup. For differential pairs, the distance between traces also depends on the stackup.

The impedances are as follows:

- Most of the tracks are routed as 55 Ohm, including DDR2 data and address signals.
- DDR2 clock differential pair 100 Ohm differential.
- USB differential pair 90 Ohm differential.

Using the JLCPCB impedance calculator, the required trace width and spacing have been calculated.

Layer	Impedance	trace width	space
TOP/BOTTOM	55 Ohm SE	4.73 mil (0.120 mm)	-
TOP/BOTTOM	90 Ohm DIFF	5.55 mil (0.141 mm)	6 mil (0.152 mm)
TOP/BOTTOM	100 Ohm DIFF	4.35 mil (0.110 mm)	6 mil (0.152 mm)

# PCB Layout

I made a 15-minute video where I go through the layout of the board. The actual layout took roughly 20 hours.

## PROJECT LOGS

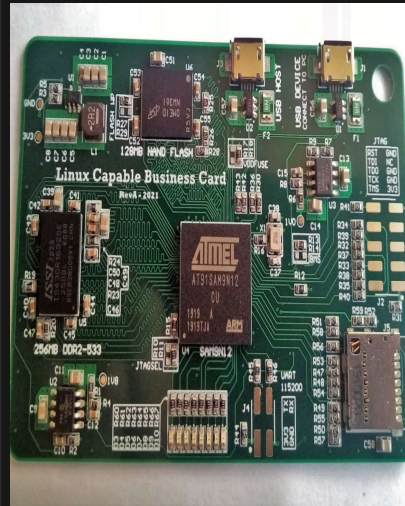
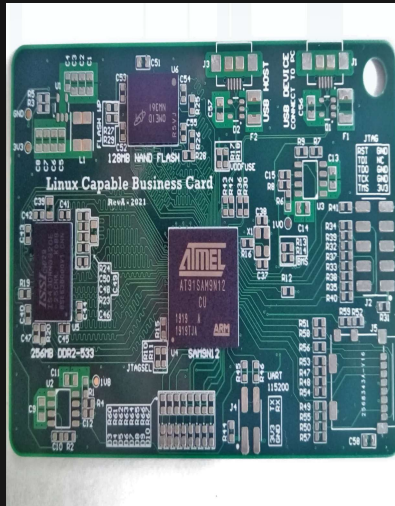
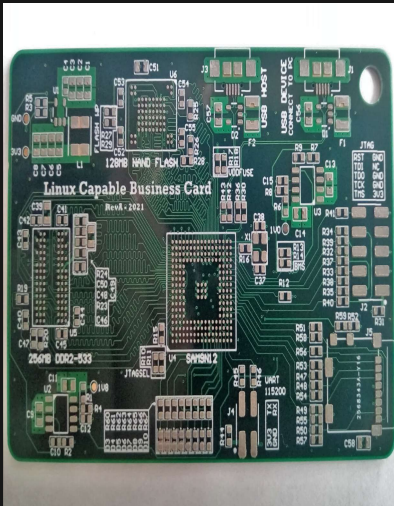


### Booting Linux from SD Card

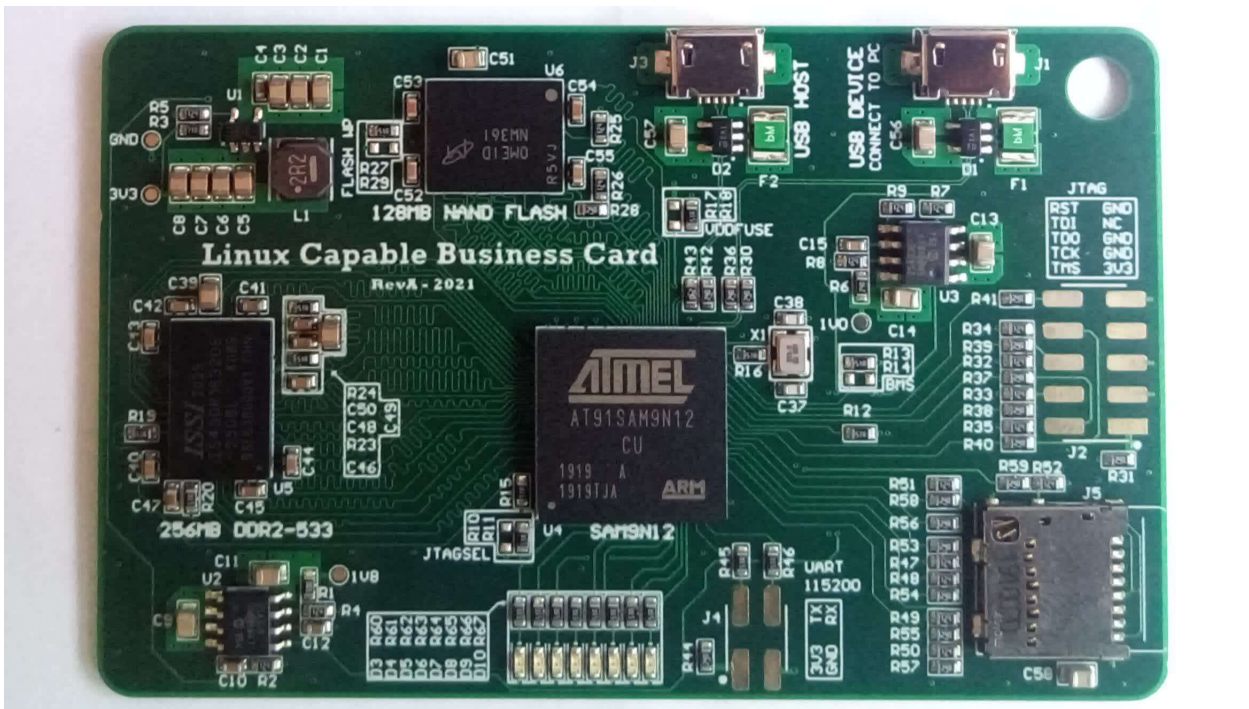
Manuel Tosone • 09/17/2021 at 09:57 • 0 comments

## Assembling

The board is hand-assembled. I use a stencil to spread the solder paste and then the components are placed with tweezers. When working with BGA packages I always place them first. To check the alignment, I look under the package from its side. If I can see light in between every row the component is aligned. Using a thicker paste it is possible to tweak the position of the package by bumping it from the side without smearing the paste. When the component is in place I lightly push down on it to make sure every ball is in contact with the paste. At this stage, if I screw up the placement, I can always clean the board and start over with minimal drama. When all the BGAs are in place, I position the other components starting with the hardest. Soldering is done with hot air. Managing the heat is a little tricky, mostly because it's hard to tell when the BGAs are soldered properly. What works for me, is to set the temperature to 100 °C and preheat the entire board for about two minutes, then I increase the temperature and start reflowing component by component.







## First Power Up

Applying power for the first time is always a little scary.

First, with a multimeter, I checked every power rail for shorts. Next, I connected the board to a current limited power supply and measured the voltages on the three test points. The voltages were good. There is a sequence in which the power rails must come up to ensure proper operation. The correct order is 3v3, 1v8, 1v0, with the CPU reset releasing last. Looking at the scope capture below we see that it is correct.



The last check is to see if the oscillator is running, and it is. That's promising, we might have a working board!



## Linux kernel and Buildroot

I'm no expert when it comes to embedded Linux. Since this is my first time using buildroot to compile Linux from scratch, I won't try to explain how it works (I barely understand it myself ;) ) but I will link to some good resources in case you want to learn more.

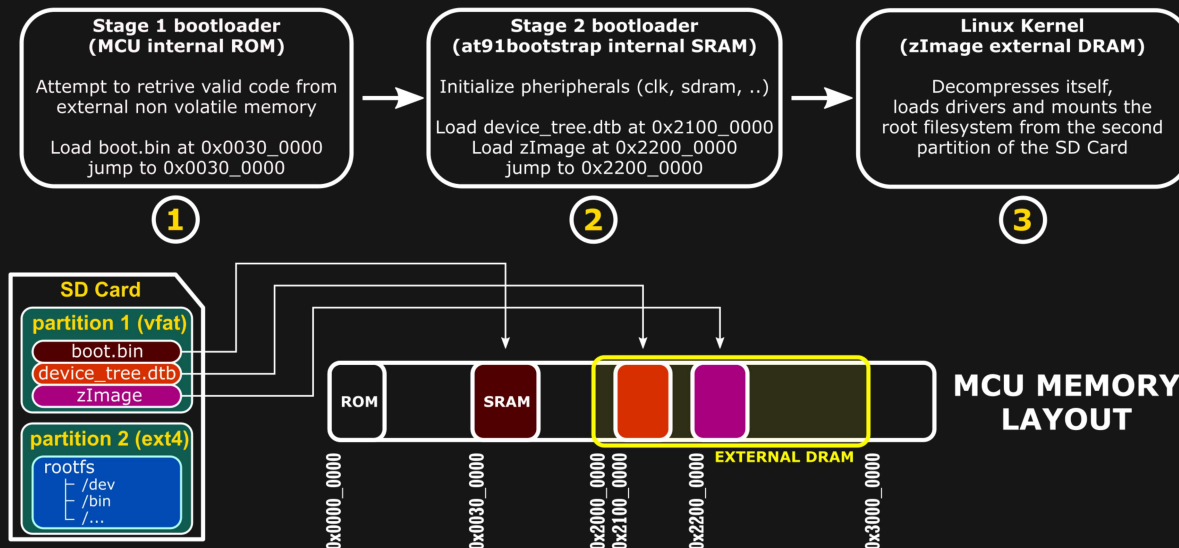
- Buildroot - Create a Custom Project  
- <https://microchipdeveloper.com/32mpu:buildroot-custom-project>
- AT91Bootstrap - <https://www.linux4sam.org/bin/view/Linux4SAM/AT91Bootstrap4>
- Making embedded Linux computer - <https://hforsten.com/making-embedded-linux-computer.html>
- Mastering Embedded Linux  
- <https://www.thirtythirtyfour.net/posts/2020/01/mastering-embedded-linux-part-3-buildroot/>
- Embedded Linux System - <https://brainyv2.hak8or.com/>

To keep things simple I've decided to have the second stage bootloader loading Linux directly instead of using a third stage like u-boot.

The boot process starts with the first stage bootloader that resides in the internal ROM. It tries to load valid code from external non-volatile memories. When booting from the SD Card it looks for a "**boot.bin**" file (second stage bootloader) in the root directory of a FAT12/16/32 partition. That file is copied to the internal SRAM and executed.

The second stage bootloader initializes the hardware, loads the device tree and kernel image to the external DRAM. After that jumps to the kernel image that is self-extracting (it decompresses itself inside the memory). Then the kernel loads the root file system from the second partition of the SD Card.

## AT91SAM9N12 - SD Card Boot Sequence



## RAM Troubleshooting

After putting the image on the SD Card and connecting to the debug serial port, I applied power through the USB. The board Initially seemed to work fine, the SD card was detected, the second stage bootloader was able to run and load the kernel. It was all fine until the MPU started executing code from the external DRAM, there was some text output from the kernel but then it just kernel panic and gave up trying. On subsequent attempts it didn't behave consistently, the system did freeze or kernel panic at different times. After enough tries, it managed to boot and I was able to log in, from the terminal I tried to run **memtester** (userspace tester for stress-testing the...