Internship course summer 1403

PoE Injector

Mentor: Mrs. Shaterzadeh

Members: Arezoo savlani ,Mehrad Afshari Nazari

Table of Contents

Introduction

This project is for the Parman Company. We chose it for our summer internship course.

Goals and objectives

This project includes the design of an electronic controller plug-in board for the POE injector that is the main board which Parman has designed before.
Plugin board is a microcontroller responsible for generating and controlling I2C, UART and RMII signals for interaction and controlling POE controller chip on the main board.

Essential Information

POE

Electronic devices need a small amount of electricity to be able to do their work.
In our daily life and in home telephones, we have seen that some devices do not have a plug for electricity
They only have phone port input. It seems that their energy is supplied with information. They say Power Over Ethernet (POE) to this technology.
In this technology, DC power enters the devices through a copper wire. These devices can be CCTVs cameras, VoIP phones, systems with wireless access points (WAPs).

Pros and Cons:

Advantages:

1- Reducing the cost of wiring for the building by reducing the number of devices that require outlets
2- Reducing the electrical wires of the devices and occupying less space
3- Less restrictions on the decoration of electrical appliances because they do not have to be plugged to sockets.

Disadvantages:

1- POE has low voltage and because of that it is less efficient compared to normal electricity also Ethernet conductors make it worse.
2- It often requires additional hardware

History:

This technology was developed first by Cisco in 2000. They wanted to test this technology in IP Phones.
In 2003, they designed the first standard which was named IEEE802.3af and it is also known as Type 1. This standard makes 15.4 Watts power that was sufficient for that time's technology.
After six years in 2009, a new version came out and they named it PoE+ or Type 2. Its standard was named IEEE 802.3at and it supplies 30 watts.
In 2011, UPOE came with IEEE802.3bt. It is named Type 3 and supplies 60 watts.
Finally in 2018, Type4 came that can make 90 watts.

Phases

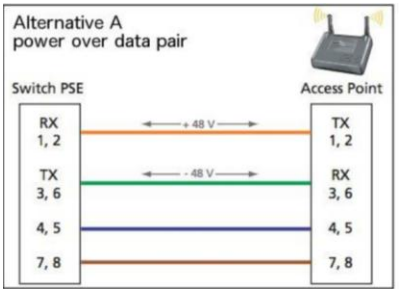
Phase 1: Theoretical search and information

1.1 Standards:

10BASE-T, 100Base-TX, 1000Base-T, 4PPoE:

Four twisted pairs of wires are used for data, from the standard BASE10-T and BASE100-TX
They use 5 Cat cabling for data. It gives data using pin 1 and 2 and with Using pins 3 and 6 takes data. (In some models it is the opposite)
The BASE10-TX transmitter sends two differential voltages +2.5 or -2.5, and the BASE100-TX transmitter sends three differential voltages, +1, 0, or -1. Positive voltage on pins 4 and 5 and pins 7 and 8 Negative voltage flows.

The BASE100-TX follows the same wiring patterns as the BASE10-TX, but due to the higher bit rate, It is more sensitive to the quality and length of the wire.



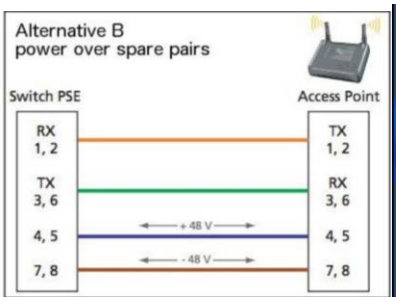


Figure 1: Two type of POE standard

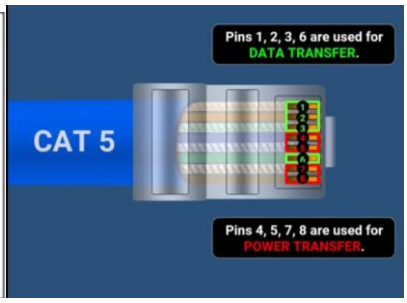


Figure2: Standard of wires for data and power

4PPoE uses all four twisted pairs in parallel and it is the most optimal mode.

CAT 5 cable

Cables are divided into several categories in terms of attenuation: Category 3, Category 4 and Category 5.

Category 5 cables (abbreviated as CAT5) have less attenuation compared to Category 3 cables, which only have a speed of 16 megabits/seconds. It can be used in high-speed Ethernets.

An upgraded version of this type of cable called CAT 5e is available in the market. They have 4 pairs of wires. However, most of the time only two pairs of them are used to reduce the noise.

Also, we can see the difference between UTP and STP cables that is about having or not having a protective shield.

Figure 3: CAT5 wires with color range

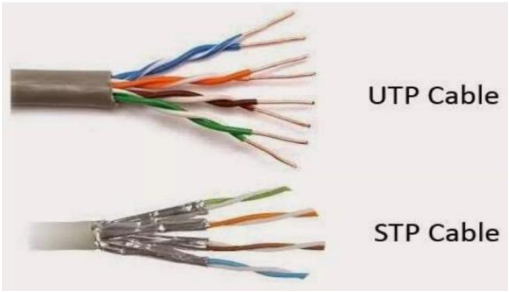


Figure 4: UTP & STP cables

RJ45

The most used Port that has 8 wires. Wires are twisted paired and they are arranged in color order. It is mostly used to connect computers to devices such as switch, modem, or a router.

Figure 5: RJ45 connector

1.2 Devices:

PD:

The receiver device which uses PoE to work.

PSE:

To receive data and energy, we must use the PSE device. One of them is a POE switch that receives data and DC power by connecting to the device. Therefore, it is necessary for our switch to be POE so that it can transmit power.

POE Injector

If our switch is not POE, we can use POE-injector to unify power and data and send it to the device. In this way, it takes the power from the socket with the plug and receives the data from the POE-Non switch with the Lan port, and by using the POE port, we can receive the data and power output and send it to the device.

Figure 6: Types of POE Injector

Types of POEs based on the type of power application:

Active: First, it measures to find the required power for the device, because if it gives too much power, the device will burn.

Passive: Transfers full power to the device without measuring the device's needs. So, it is necessary when we use this type of POE to pay attention to the voltage of the device because the device may suffer.

1.3 Control board:

Our plug-in port will be connected to the main board by port and it named as micro plugin in this block diagram. It uses I2C protocol to get data from switch board. We have regulators to convert power to range that we use in our board and IC. We have transformer to send POE signal to switch board. Our controller-board will use NPI and UART protocol to send or receive data to LEDs and RJ45 ports.

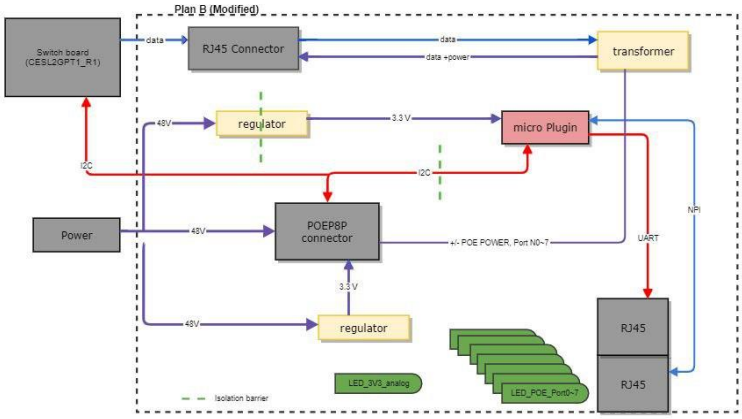


Figure 7: Block diagram of main board

It takes power from isolation barrier regulator that convert 48V to 3.3V. It takes data with I2C protocol from switch board and use UART and NPI protocol to send data to RJ45 ports.

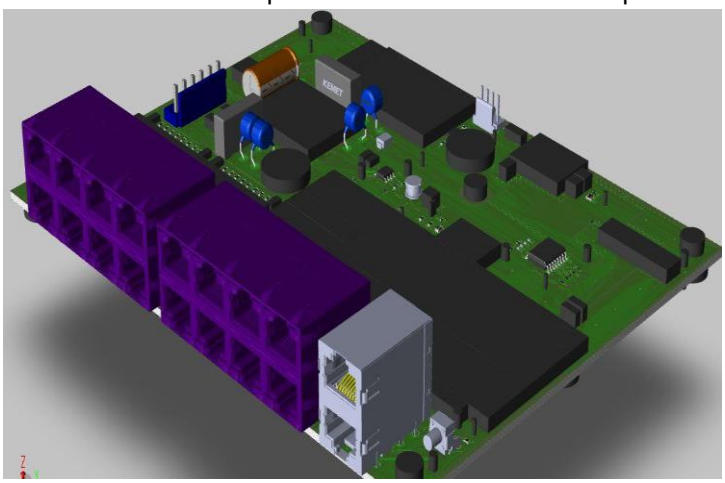


Figure 8: Main board 3D image

1.4 Protocols:

a) UART

Universal **A**synchronous **R**eceiver **T**ransmission is one of the oldest and most reliable serial transmission protocols. In the serial transmission method, only one bit is sent at any time, but in the parallel transmission method, several bits of information are transmitted through several lines. In comparison between these two methods, it can be said that in the parallel transmission method, due to the transmission of several bits of information at a time, its speed is higher.

On the other hand, for long distances, using the parallel transmission method has a high cost due to the increase of communication wires. Therefore, it is not suitable for long distance transmission using the parallel method and the UART protocol.This protocol is being used in COM serial ports, RS-232 and modems.It is used for low speeds and convenient applications.

Many developers use UART for wireless connectivity and computer processing; because it is easy to set up and more favorable for the user.

How does it work?

The special feature of UART is that the sending and receiving of data will be available only when the transmitter and receiver are coordinated with the following 5 settings:

1. Speed:

Since UART is an asynchronous communication protocol, no special clock is used to set the data transfer speed. Instead of clock, baud rate (mostly 9600 bits per second) is used to set the timing of sending data and bits.

The following table lists some of the standard baud rates:

Common UART baud rates
4800
9600
19200
57600
115200

For example, the transfer speed of one data bit: 1/9600 = 104 microseconds
In fact, the device needs to count 104 microseconds to send the next bit.

Figure 9: Standard baud rates

2. Data Length:
The specified number of bits that the receiver stores in its registers
which are arranged from the least significant bit (LSB) to the most significant bit (MSB). It is Usually 8 bits (1 byte)
3. Start bit:
Using a low signal to show the time to start sending data to the receiver (zeroing should take time as one bit period)
4. End bit:
Using a high signal to show the time of the last bit that is informed to the receiver.
We consider this bit to be the most important bit. (It should take one bit period to become one)
5. Parity bit (or balance):
To know the correctness of sent data with help of a zero or one bit (odd and even error)
A total of 11 bits are used for data transmission. When there is no sending or receiving the transmission line is in the "idle" state, the voltage level corresponding to a logical one is on the transmission line. Changing the status from logical one to zero means the start of transmission (start bit) and the receiver is ready to get information. After that, a data byte is sent.
After sending data, the receiver takes the balance bit which is actually the XOR of the individual data bits. If the number of 1 is odd, this value will be 1 and otherwise it will be zero.
When the receiver gets the data it calculates the XOR once again and detects an error if it is different from the received balance bit value and the received packet is completely ignored. Finally, after confirming or rejecting the data, the status changes from zero to logical one and means the end of sending (end bit).

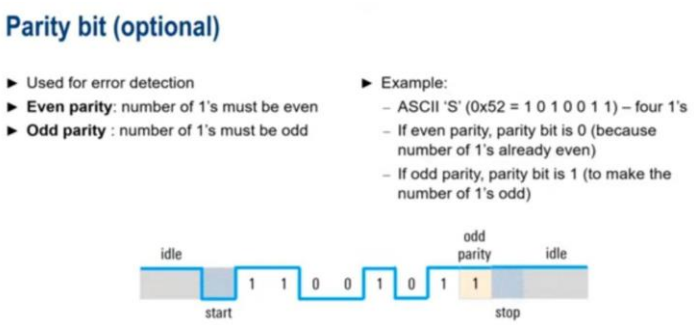


Figure 10: Example of UART protocol

Type of the Errors we can have:
1- Overrun and Underrun:
When the device receives more information than the memory capacity in a certain period of time (for example, 104 microseconds) or does not receive anything.
2- Framing:
When the receiver does not receive the end bit after 8 data bits.
3- Parity (It was explained in previous part)
4- Break condition:
When the input of the receiver is zero for a long time, this is almost the same as Framing error.
Communication Process and Hardware:
Data is sent and received using two pins Tx and Rx pin. Data in parallel from the bus Data is sent to the UART, the sender device creates a data packet by adding the start, balance and end bits of one. Next, send this packet serially and bit by bit through the TX pin and receiver device receives serial data through the RX pin and converts it to parallel mode and removes the start, balance and end bits. At the end, this data is sent to the data bus in the receiver.

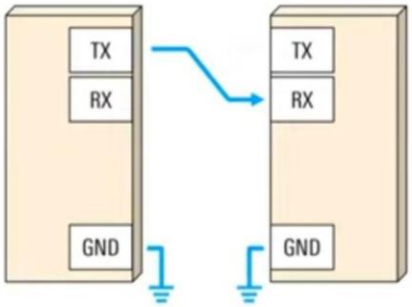


Figure 11: Communication Process

In general, there are three methods for communication between sender and receiver:
1- One-way method (Simplex): In this method, information is transmitted in only one direction.
2- Half duplex method (Duplex Half): In this method, information can be transmitted in two directions. But at any moment only one direction is possible.
3-Two-way method (Duplex Full): In this method, information is transmitted in both directions at the same time

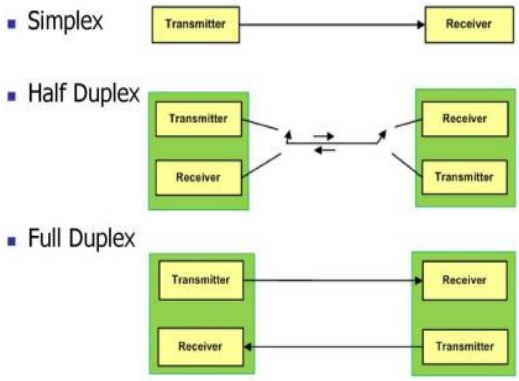


Figure 11: Methods for communication

Advantages of using UART:
Use of only two wires (Clock cable not required)
Presence of Parity bit for error checking

Disadvantages of using UART:
The size of the data sent is limited.
We have only one master (controller) and one slave (controlled).
The transfer rate of both sides should be close or equal (10% range).

b) I2C

I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave serial communication protocol developed by Philips Semiconductor (now NXP Semiconductors).
It is widely used for connecting peripherals to microcontrollers in embedded systems. I2C is favored for its simplicity and efficiency in short-distance communication within a single device or on a single board.

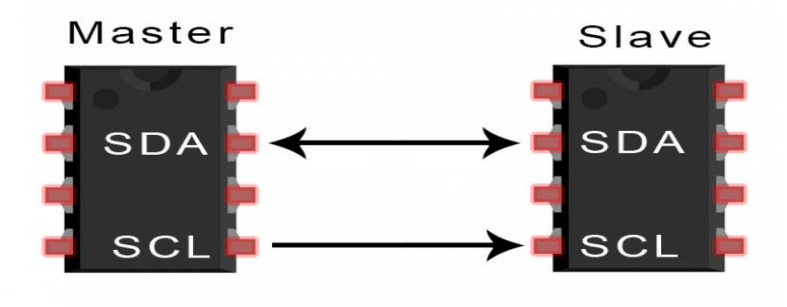


Figure 11: I2C Protocol Hardware

Key Features of I2C

1. **Two-Wire Interface:**
   - SDA (Serial Data): Carries data between devices.
   - SCL (Serial Clock): Carries the clock signal, which synchronizes data transfer.
2. **Multi-Master and Multi-Slave:**
   - Multiple master devices can control the bus, and multiple slave devices can share the bus.
   - Only one master can control the bus at any given time to avoid conflicts.
3. **Addressing:**
   - Each device on the I2C bus has a unique address, which can be either 7-bit or 10-bit.
4. **Speed Modes:**
   - Standard Mode: Up to 100 kbps.
   - Fast Mode: Up to 400 kbps.
   - Fast Mode Plus: Up to 1 Mbps.
   - High-Speed Mode: Up to 3.4 Mbps.

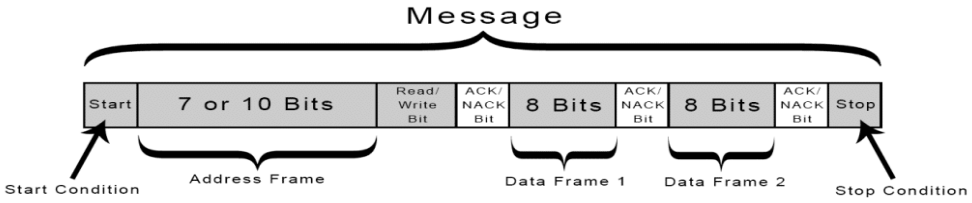


Figure 12: I2C Protocol

How I2C Works?

Data Transmission

1. **Start Condition:**
    - Initiated by the master, this involves a high-to-low transition on the SDA line while SCL is high. It signals all connected devices that a communication session is starting.

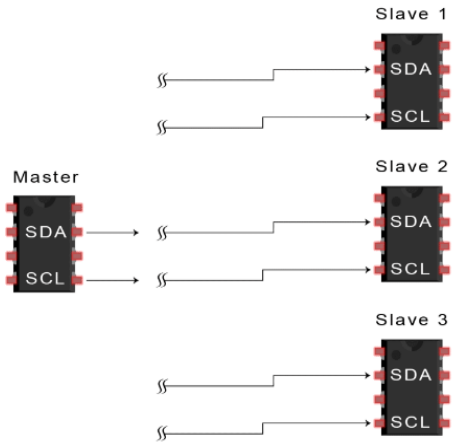


Figure 13: Start Condition

2. **Address Frame:**
    - The master sends a 7-bit (or 10-bit) address along with a read/write bit. The read/write bit indicates whether the master wants to read from (1) or write to (0) the slave.

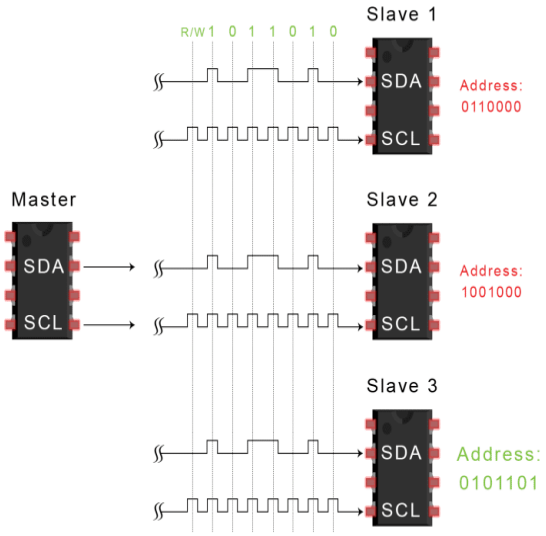


Figure 14: Address Frame

3. **Acknowledgment (ACK/NACK):**
    - After receiving the address, the addressed slave device sends an ACK (low signal) if it recognizes the address, or a NACK (high signal) if it does not.

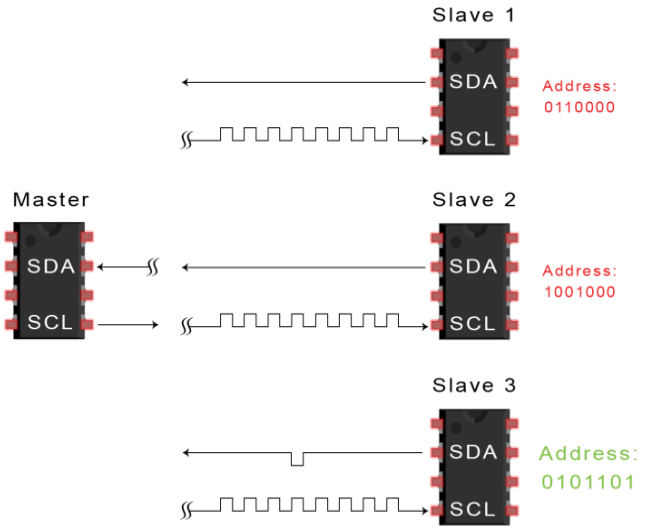


Figure 15: ACK/NACK

4. **Data Frames:**
    - Data is transferred in 8-bit bytes. Each byte is followed by an acknowledgment bit.
    - Data transfer can be from master to slave (write operation) or from slave to master (read operation).

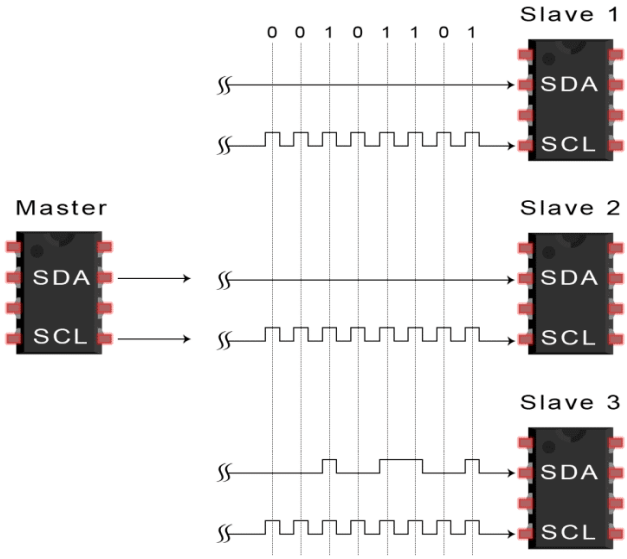


Figure 16: Data Frame

5. After each data frame has been transferred, the receiving device returns another ACK bit to the sender to acknowledge successful receipt of the frame

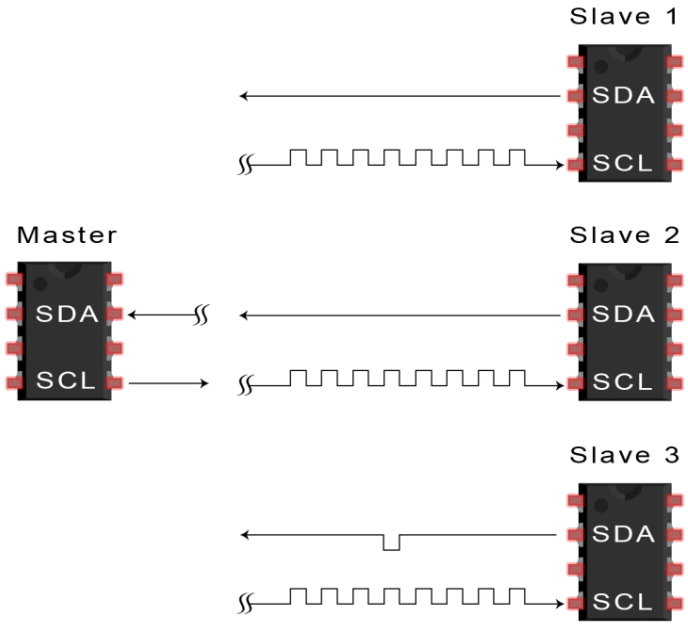


Figure 16: Receiption configuration

6. **Stop Condition:**
   - Indicates the end of a data transfer session. It involves a low-to-high transition on the SDA line while SCL is high.

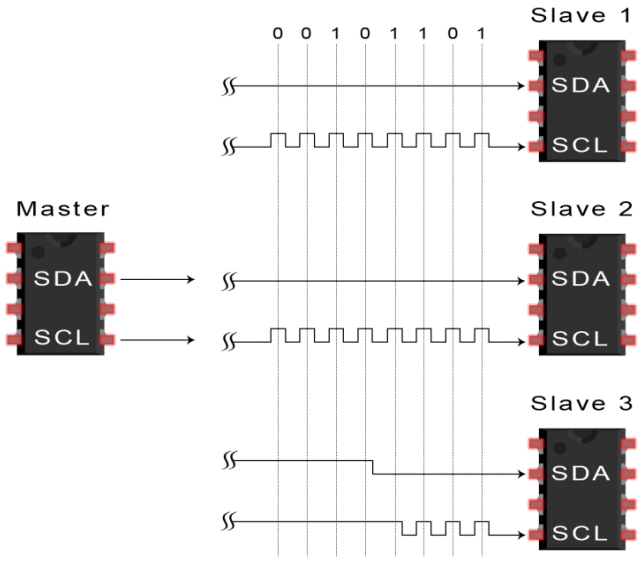


Figure 16: Stop Condition

Advantages

- Only uses two wires
- Supports multiple masters and multiple slaves
- ACK/NACK bit gives confirmation that each frame is transferred successfully
- Hardware is less complicated than with UARTs
- Well known and widely used protocol

Disadvantages

- Speed: Slower compared to protocols like SPI.
- Distance: Limited to short-distance communication due to capacitance and signal integrity issues.
- Bus Contention: Potential for conflicts if multiple masters attempt to control the bus simultaneously.
- The size of the data frame is limited to 8 bits

---

c) NPI

A common system configuration is to use the Texas Instruments CC13xx/CC26xx as a network processor (NP) where a host microcontroller (MCU) is able to control the CC13xx/CC26xx platform by sending serial commands. This requires certain power domains of the NP to be on in order to receive and handle these commands. It also requires translating serial packets into commands for the NP as well as translating responses into serial packets to be sent to the Host. The handling of these messages along with managing the power domains in an efficient manner is the responsibility of the network process interface (NPI) framework of the CC13xx/CC26xx platform. The NPI framework enables the CC13xx/CC26xx to go into low power modes when not being used and to be asynchronously woken up to respond to commands from the host MCU.

1.1 Introduction

The NPI framework is shared among the following (insert marketing name) products:

- Bluetooth Smart
- Z-Stack ZigBee Network Processor (ZNP)
- TI-15.4-Stack MAC Co-Processor (MAC-CoP)

Between these different products, there are some differences in how data framed for serial communication, as well as how messages are handled within the NP. Despite these minor differences, shown in red, all products share the same basic NPI framework shown below:

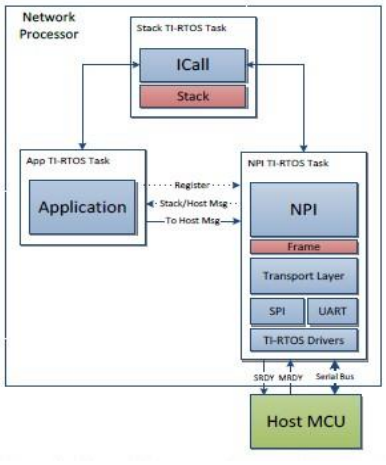


Figure 17: NPI Network

NPI's functionality can be broken down into three main components: managing the serial interface between the NP and Host, managing command and responses to and from the stack, and optionally routing stack or host messages to a registered application task.

1.2 Serial Interface

In order to properly manage the power domains of CC13xx/CC26xx platforms, NPI must include a way for an external host to wake the network processor from low power modes. This is done through inclusion of "master ready", MRDY, and "slave ready", SRDY, signals. These two signals along with the respective SPI or UART serial bus signals comprise the serial interface of NPI.

MRDY is an input pin on the network processor and when set by the Host will wake the device enabling it to receive data over the serial bus.

SRDY is an output pin on the network processor, and, when set by the NP, signals to the host that it may begin data transfer. This sequence of setting MRDY and SRDY pins comprises a "handshake" process to guarantee both devices are awake and ready to send and/or receive data.

1.3 Command and Response Message Routing

The common use case of the NPI framework is a basic network processor. This means that the framework is strictly acting as a sending and receiving mechanism between the host MCU and the stack task. However, there also exist use cases that include some amount of "custom" application-dependent processing.

For these cases not all messages from the host should be delivered to the stack and not all messages from the stack should be transferred directly to the host. To enable these types of applications, a mechanism to easily change the paths of inbound and outbound messages is built into NPI.

In Figure 1, the possible message rerouting with the application task can be seen. An application can easily interject messages to the host within the command and response traffic of the stack using the NPI APIs. An application can also register with NPI to either "Intercept" or "Echo" all messages originating from the host and/or all messages originating from the stack. This allows the application to modify messages as needed or to even process certain application specific messages without sending them to either the host or stack.

2 Interfacing with a Host Processor

2.1 Physical Interface

The NPI framework currently supports two serial interfaces, SPI and UART. The CC13xx/CC26xx has two SPI modules and only one UART module. When configured to use SPI, NPI will by default use the second SPI module (SSI1) (Note: This is due to the SmartRF06 board using SSI0 for driving LCD). For each serial interface, power management can be enabled and regardless of interface type MRDY and SRDY pins remain consistent. There are three possible configurations:

- UART with Power Management
- UART without Power Management
- SPI with Power Management

2.1.1 UART

NPI supports two different configurations of UART. One configuration requires only two pins, UART TX and UART RX. This is an always-on configuration that does not permit the CC13xx/CC26xx to enter into low power modes. The second configuration does allow low power modes because of the inclusion of the MRDY and SRDY pins for power management. Some UART interfaces include hardware flow control pins, CTS and RTS, however we do not currently support hardware flow control. Instead the MRDY and SRDY pins can be used for both power management and software flow control.

UART Signal Name	7x7: IOID	5x5: IOID	4x4: IOID	EM Header ID	SmartRF06 Port ID
RX	2	1	1	RF1.7	P408.14
TX	3	0	2	RF1.9	P408.12
MRDY	19	6	4	RF1.10	P403.12
SRDY	12	4	3	RF1.12	P403.16

Table1: UART in NPI Protocol

2.1.2 SPI

There is only one SPI configuration which requires the use of MRDY and SRDY. The NPI framework requires the ability to asynchronously send data from NP to Host. Because of this requirement it is not possible to have a configuration with the traditional SPI signals alone (SCLK, MOSI, MISO, CS). The SRDY signal is used to signal the SPI master that a slave to master transmission is pending so the master should toggle the SPI SCLK signal to begin the transfer. Without this signal, an asynchronous NP to Host transfer is not possible. The MRDY signal performs the chip select, CS, function among other functions so the chip select signal is not needed with the SPI configuration.

NOTE: The signals highlighted in the table below suggest an alternative pin assignment for SPI using SSI1 to avoid a conflict with the LCD control signals on the SmartRF06.

SPI Signal Name	7x7:IOID	5x5:IOID	4x4:IOID	EM Header ID	SmartRF06 Port ID
MOSI	9	11	9	RF1.18	P404.8
MISO	8	12	0	RF1.20	P404.10
SCLK	10	10	8	RF1.16	P404.4
MOSI	*23*	*n/a*	*n/a*	*RF2.5*	*P404.12*
MISO	*24*	*n/a*	*n/a*	*RF2.10*	*P404.18*
SCLK	*30*	*n/a*	*n/a*	*RF2.12*	*P405.2*
MRDY	19	6	4	RF1.10	P403.12
SRDY	12	4	3	RF1.12	P403.16

Table1: SPI in NPI Protocol

2.1.3 Configuration Management

With any network processor project that uses the NPI framework, switching between configurations can be easily managed. There are only two preprocessor defines that are needed.

- **POWER_SAVING** – If defined, MRDY/SRDY as well as low power modes are used. If not defined, the device will be always on and NPI must use UART.

- **NPI_USE_SPI/NPI_USE_UART** – Only one of these can be defined at a time. Each define corresponds to the underlying serial interface that will be used by NPI

- **NPI_SREQRSP** – If defined, support for synchronous REQ/RSP messaging is included. This is currently only relevant for MT-based NPI functionality.

2.2 NPI Messages

NPI messages describe an ordering of events on the interface pin, not the content of the data that is sent or received. NPI frames define data content that is sent or received. In this section, the focus will first be to describe messaging and the behavior of the physical interface and then to describe how data content, or frames, are then sent over these messages.

When using power management pins, MRDY and SRDY, there are only two events whose order determines the message: assertion of MRDY and the assertion of SRDY (Note: MRDY and SRDY are active low, thus “assertion” refers to a logic value of 0). Before any data can be sent, a “handshake” must occur and this handshake can be initiated by either Host or NP. There for two messages exist: Asynchronous Request – transfer initiated by Host, and Asynchronous Indication – transfer initiated by NP. When not using power management pins, the two events that define an Asynchronous Request or Asynchronous Indication is the beginning of a data transfer on the UART Master TX pin or the UART Slave TX pin respectively.

2.2.1 Asynchronous Request (AREQ)

An asynchronous request is a message that is sent from Host to NP. The host must initiate the handshake by asserting MRDY, once SRDY has been asserted, then data can be transferred Host to NP. MRDY is de-asserted by the Host once all data has been transferred and then NP de-asserts SRDY notifying the Host that it may be in a low power mode. The order of events is shown below:

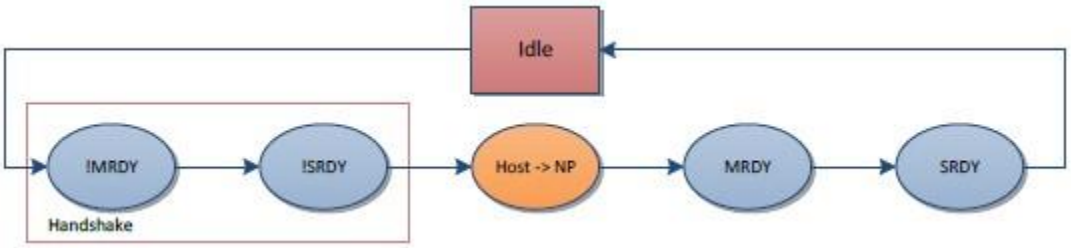


Figure 18: Block diagram of AREQ

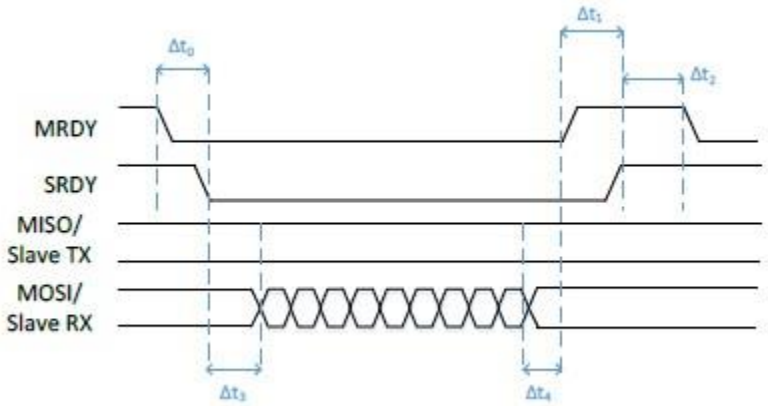


Figure 19: Signals of AREQ

2.2.2 Asynchronous Indication (AIND)

An asynchronous indication is a message that is sent from NP to Host. The only difference between this message and AREQ is that SRDY is asserted first and the direction of the data transfer is inverted. The order of events can be seen below:

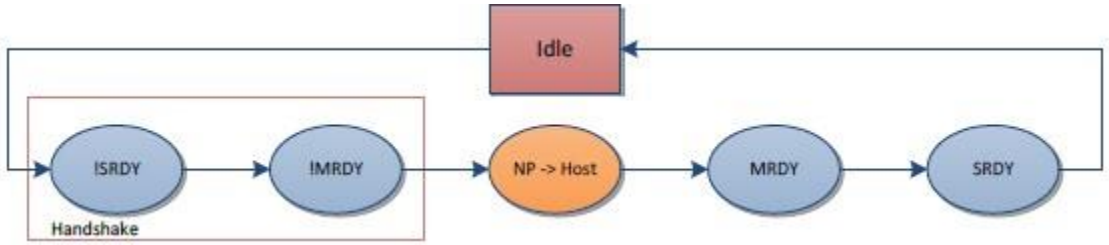


Figure 20: Block diagram of AIND

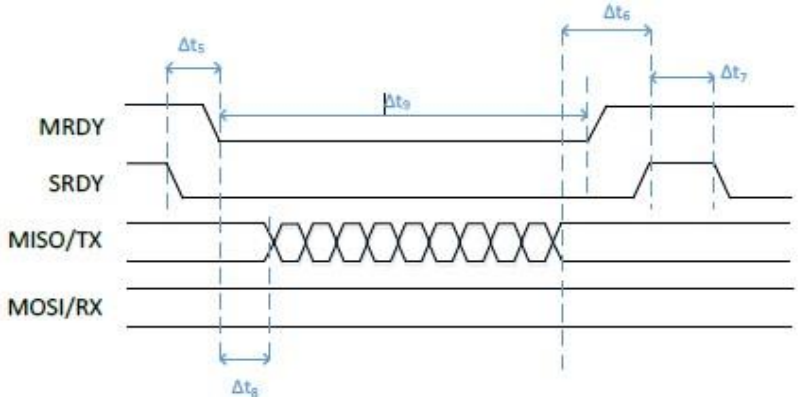


Figure 21: Signals of AIND

2.2.3 Bidirectional Messaging

With any protocol that allows asynchronous messaging, where a signal transition denotes the start of a message as with MRDY and SRDY, there are inherent collisions between messages. Instead of requiring intricate collision handling, the NPI framework allows for bidirectional messaging to occur. This means that data can be sent from the Host to NP and from NP to the Host in the same message window regardless of the handshake order.

While reducing collision handling, bidirectional messaging adds some complexity to what operations must be performed or initiated by each device. For every AIND the NP initiates, it must prepare to read as well as write when MRDY is asserted. For every AREQ, the Host must prepare to read as well as write once SRDY is asserted. Each device will also need to handle any FIFOs that could potentially be overrun during a message and check at the end of every message to see what, if anything, has been received.

The scenario where bidirectional messaging will occur is when a device's input pin (SRDY wrt Host) has been asserted, and prior to asserting its output pin (MRDY wrt Host) the device gets a pending message to write. Below is the modified ordering of events where there exists now a partial order on the MRDY and SRDY assertions.

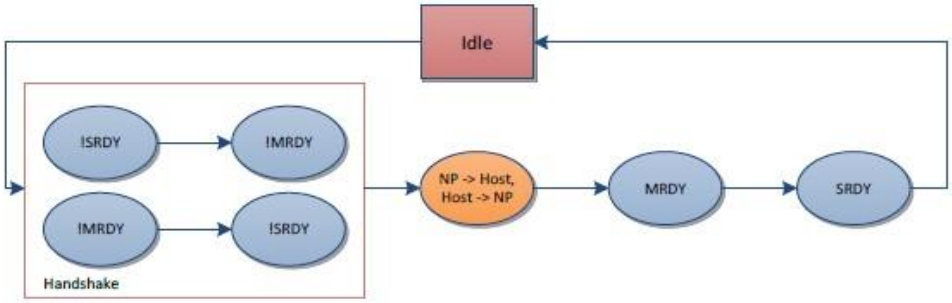


Figure 22: Block diagram of bidirectional massaging

Steps to Implement NPI

1. Hardware Setup

1. **Select Appropriate Hardware**:
    - Choose the appropriate Texas Instruments CC13xx or CC26xx evaluation module (EM) based on your application requirements.
    - Ensure you have a compatible host MCU for communication with the CC13xx/CC26xx platform.
2. **Connect the Host MCU to CC13xx/CC26xx**:
    - Connect the host MCU to the CC13xx/CC26xx module using a serial communication interface such as UART, SPI, or I2C.
    - Make sure the power supply, ground, and other necessary signals (such as reset and interrupt lines) are correctly connected.

2. Software Configuration

1. **Install Development Tools**:
    - Download and install the TI Code Composer Studio (CCS) or any other preferred integrated development environment (IDE).
    - Install the SimpleLink CC13xx/CC26xx software development kit (SDK) from Texas Instruments.

2. **Configure the SDK for NPI**:
    - Open the project in the SDK related to the CC13xx/CC26xx platform.
    - Include the NPI libraries and dependencies in your project. The NPI libraries are typically provided within the SDK.
3. **Initialize NPI in the Application**:
    - In your application code, initialize the NPI framework. This involves setting up the serial interface, power management, and command processing.
    - Example initialization code snippet:

```
// Include necessary headers

#include "npi.h"
#include "npi_tl_uart.h"
#include "npi_task.h"

void npi_init(void) {
    // Initialize NPI Transport Layer (UART in this case)
    NPI_InitTransportLayer();

    // Initialize NPI Task
    NPI_Task_createTask();
}
```

3. Develop Communication Protocols

1. **Define Serial Commands and Responses**:
    - Define the set of serial commands that the host MCU will send to the network processor. This includes specifying command IDs, parameters, and response formats.
    - Implement the command handling functions on the CC13xx/CC26xx side to process these commands.
2. **Command Processing Example**:

```
    void processCommand(uint8_t commandId, uint8_t *params) {
    switch (commandId) {
        case CMD_ID_1:
            // Handle command 1
            break;
        case CMD_ID_2:
            // Handle command 2
            break;
        default:
            // Handle unknown command
            break;
    }
}
```

3. **Translate Serial Packets**:
    - Implement functions to translate received serial packets into internal commands for the network processor.
    - Implement functions to translate responses from the network processor into serial packets to be sent back to the host MCUZ

4. Power Management

1. **Manage Power Domains**:
    - Use the NPI framework functions to manage power domains. Ensure that necessary power domains are enabled when receiving and processing commands.
    - Implement mechanisms to enter low-power modes when the network processor is idle and wake up asynchronously when commands are received.
2. **Power Management Example**:

```
void managePower() {
  // Enter low-power mode
  Power_enterLowPowerMode();

  // Wake-up on receiving a command
  if (commandReceived) {
    Power_exitLowPowerMode();
    processCommand(receivedCommandId, receivedParams);
  }
}
```

5. Testing and Debugging

1. **Debugging Tools**:
    - Use debugging tools provided by the IDE to step through the code, set breakpoints, and inspect variables.
    - Utilize serial communication analyzers to monitor the data exchange between the host MCU and the network processor.
2. **Testing**:
    - Test the entire setup by sending commands from the host MCU and verifying the responses from the network processor.
    - Ensure the power management functions are working correctly by measuring the current consumption in different power modes.

1.5 OSI Model

<table>
<tr><th colspan="2">OSI Model Layer</th><th colspan="2">TCP/IP Model</th></tr>
<tr><td rowspan="4">Host Layers</td><td>7 | Application</td><td colspan="2" rowspan="3">Application Layer</td></tr>
<tr><td>6 | Presentation</td></tr>
<tr><td>5 | Session</td></tr>
<tr><td>4 | Transport</td><td colspan="2">TCP/UDP</td></tr>
<tr><td rowspan="3">Media Layers</td><td>3 | Network</td><td colspan="2">IP</td></tr>
<tr><td>2 | Data Link</td><td rowspan="2">Ethernet</td><td>MAC & LLC</td></tr>
<tr><td>1 | Physical</td><td>PHY</td></tr>
</table>

Figure 23: OSI Model layers

The OSI (Open Systems Interconnection) model is a conceptual framework used to understand and implement networking protocols in seven layers. Each layer serves specific functions and interacts with the layers directly above and below it. Here's a brief overview of the OSI model:

1. Physical Layer (Layer 1)
2. Data Link Layer (Layer 2)
3. Network Layer (Layer 3)
4. Transport Layer (Layer 4)
5. Session Layer (Layer 5)
6. Presentation Layer (Layer 6)
7. Application Layer (Layer 7)

Layer 1: Physical Layer

The Physical Layer is the lowest layer of the OSI model. It deals with the physical connection between devices and the transmission and reception of raw bit streams over a physical medium. Here are the key functions and characteristics:

- Transmission Media: Includes cables (e.g., twisted pair, coaxial, fiber optic) and wireless methods (e.g., radio, infrared).
- Signal Encoding: Converts digital data into electrical, optical, or radio signals.
- Bit Rate Control: Manages the rate at which data is transmitted (e.g., bits per second).
- Physical Topology: Defines the physical layout of the network devices and cabling.
- Hardware Specifications: Includes components like connectors, transceivers, and network interface cards (NICs).

The Physical Layer is responsible for ensuring that when one device sends a bit, it is received correctly by another device.

Layer 2: Data Link Layer

The Data Link Layer is the second layer in the OSI model and is responsible for node-to-node data transfer and error detection and correction. It is divided into two sublayers: Logical Link Control (LLC) and Media Access Control (MAC).

Sublayers of Data Link Layer:

1. **Logical Link Control (LLC) Sublayer:**
    - Flow Control: Manages data flow to prevent the sender from overwhelming the receiver.
    - Error Detection and Correction: Uses mechanisms to detect and correct errors that may occur in the Physical Layer.
    - Interface with Network Layer: Provides a way for the Network Layer to access the services of the Data Link Layer.
2. **Media Access Control (MAC) Sublayer:**
    - MAC Addressing: Provides a unique identifier for each device on a network (e.g., Ethernet MAC addresses).
    - Access Control: Determines how devices share the medium (e.g., CSMA/CD for Ethernet).
    - Frame Delimiting: Defines the start and end of frames, and formats frames to include MAC addresses and error-checking information.

Key Functions of the Data Link Layer:

- **Framing:** Encapsulates packets from the Network Layer into frames.
- **Addressing:** Adds physical addresses (MAC addresses) to frames for proper delivery on the same network.
- **Error Detection and Correction:** Uses techniques like checksums and CRC to detect and correct errors in transmitted frames.
- **Flow Control:** Ensures that data frames are sent at a rate that matches the receiver's capacity.

The Data Link Layer provides reliable data transfer by detecting and possibly correcting errors that may occur in the Physical Layer, and by managing access to the physical medium.

Layer 3: Network Layer

The Network Layer is responsible for data routing, packet forwarding, and logical addressing (IP addresses). It determines the best path to send data from source to destination across multiple networks.

- **Key Functions:**
    - Logical Addressing: Assigns IP addresses to devices.
    - Routing: Determines the best path for data to travel across a network.
    - Packet Forwarding: Moves packets from one network to another.
    - Fragmentation: Breaks down large packets into smaller ones for transmission.

Layer 4: Transport Layer

The Transport Layer ensures reliable data transfer between devices. It provides error detection and correction, data flow control, and data segmentation and reassembly.

- **Key Functions:**
    - Segmentation and Reassembly: Splits data into segments and reassembles them at the destination.
    - Flow Control: Manages the rate of data transmission.
    - Error Detection and Correction: Ensures data is delivered without errors.
    - Connection Management: Establishes, maintains, and terminates connections (e.g., TCP, UDP).

Layer 5: Session Layer

The Session Layer manages sessions or connections between applications. It establishes, maintains, and terminates communication sessions.

- **Key Functions:**
    - Session Management: Establishes and terminates sessions between devices.
    - Synchronization: Adds checkpoints during data transfer for recovery.
    - Dialog Control: Manages two-way communication sessions (full-duplex or half-duplex).

Layer 6: Presentation Layer
The Presentation Layer translates data between the application layer and the network format. It is responsible for data encryption, compression, and translation.

- **Key Functions:**
  - Data Translation: Converts data between different formats (e.g., ASCII to EBCDIC).
  - Data Encryption: Encrypts and decrypts data for security.
  - Data Compression: Compresses data to reduce bandwidth usage.

Layer 7: Application Layer
The Application Layer is the topmost layer and provides network services directly to end-user applications. It facilitates interaction between software applications and the network.

- **Key Functions:**
  - Network Services: Provides services like email (SMTP), file transfer (FTP), and web browsing (HTTP).
  - Application Interfaces: Interfaces with software applications to implement network services.
  - Resource Sharing: Allows applications to access network resources.

Phase 2: Schematic Design

Description:

This board is designed to enhance the performance of the PoE main board by utilizing a microcontroller. Additionally, it provides a way to monitor the status of each PoE injector port on the main board.
The board consists of four sections:

1. **MCU (Microcontroller Unit)**
2. **Ethernet Path**
3. **Communication Path**
4. **Port**

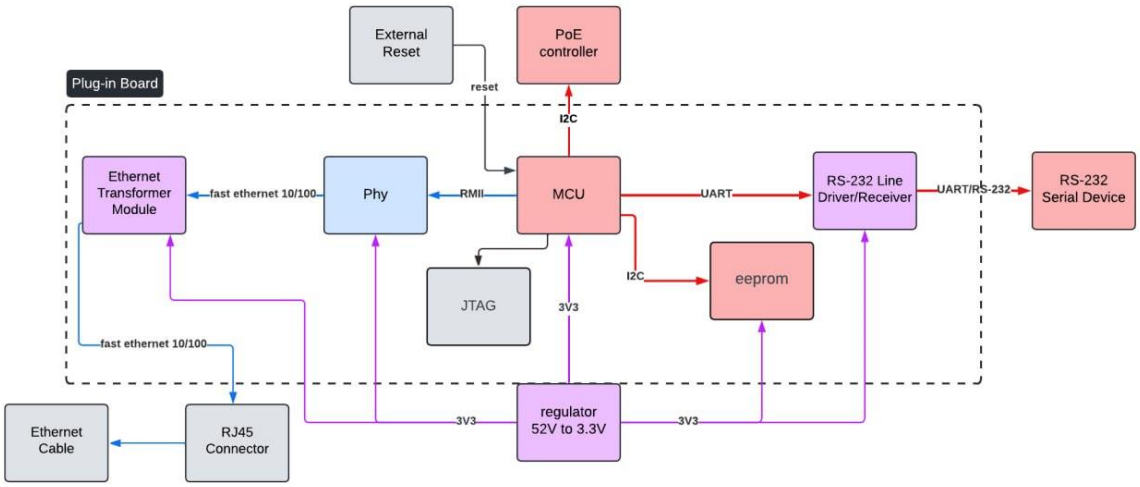


Figure 24: Block diagram of control board

Sections:

1. **MCU**

   This section is responsible for managing the plug-in board. For this purpose, a microcontroller has been used. The microcontroller needed to support I2C to communicate with the PoE's main board controller and the UART protocol for communicating with the PC for programming and monitoring. Additionally, it should support an Ethernet interface for network connectivity and allow for an external reset in special cases.

   Based on these requirements, we chose the STM32F765 microcontroller.

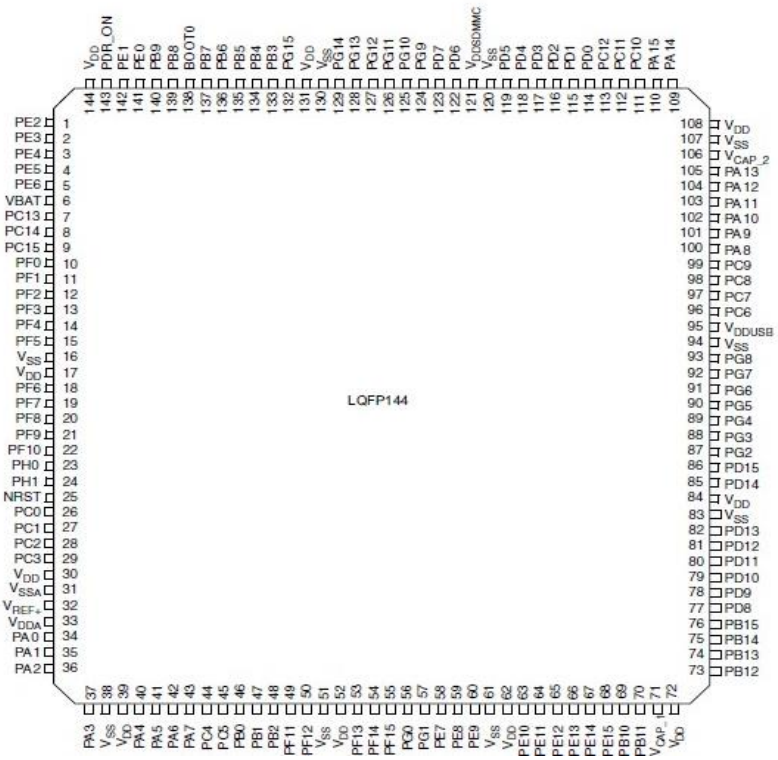


Figure 24: STM32F765 pins

AT24C04D_SSHM_T(eeprom):

There is an I2C section for the communication protocol between the micro and different chips and there is an addressing section that outputs three address pins that can be set to zero or one by hardware (by using header) and it has a fixed address section that is specified in the datasheet for the first four bits (1010). Address of this part can be specified so that the micro can call the EEPROM with this address. The point is that on a common bus, another part must not have a same address with the same part.

Figure 25: eeprom IC

JTAG:

Figure 26: JTAG IC

JTAG (Joint Test Action Group) is a standard (IEEE 1149.1) used for testing, programming, and debugging electronic devices, especially microcontrollers, processors, and PCBs.

Key Uses:

1. Boundary-Scan Testing: Checks interconnections between ICs on a board without physical probes.
2. Programming: Loads firmware or bitstreams onto devices like microcontrollers and FPGAs.
3. Debugging: Allows control of a device (e.g., stepping through code, inspecting memory/registers).

JTAG Signals:

- TDI (Test Data In): Serial data input.
- TDO (Test Data Out): Serial data output.
- TCK (Test Clock): Clock signal.
- TMS (Test Mode Select): State control.
- TRST (optional): Test reset.

Applications:

- Embedded system debugging
- Firmware programming
- Board-level testing

JTAG enables efficient test, programming, and debugging of electronics through a serial interface.

2. Ethernet Path

This section details the components required for establishing a network connection. Given that the MCU includes a MAC controller, a PHY component is necessary to manage the physical layer of data transmission.

The PHY component, specifically the **DP83848KSQ_NOPB**, converts digital data to signals for transmission over physical media and vice versa, ensuring synchronization, error detection, and optimal network performance.

Additionally, since the Ethernet section extends outside of the board, electrical protection and isolation are crucial. To address this, the **LP82453NL** Ethernet transformer module has been selected to provide the necessary protection and isolation.

Figure 27: LP82453NL IC

3. Communication Path

This section ensures a reliable and secure connection between the MCU and external devices using the UART protocol. Since the MCU will communicate with devices that adhere to the RS-232 standard, a voltage level converter is required to bridge the gap between different logic levels. To fulfill this need, we have selected the **MAX3232EIPWO** driver, which effectively converts the voltage levels to enable seamless communication between the MCU and RS-232 compatible devices.

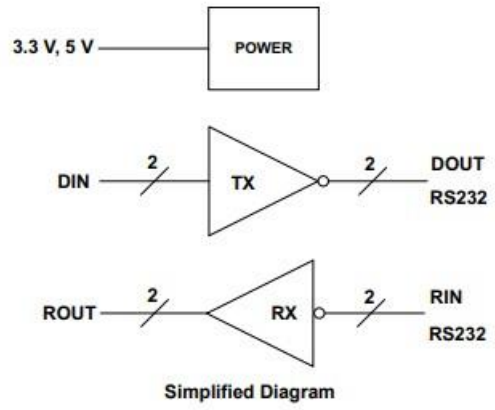


Figure 28: Block diagram of MAX3232EIPWO

Table 8-1. Each Driver(1)

INPUT DIN	OUTPUT DOUT
L	H
H	L

(1) H = high level, L = low level

Table 8-2. Each Receiver(1)

INPUT RIN	OUTPUT ROUT
L	H
H	L
Open	H

(1) H = high level, L = low ievel,
Open = input disconnected or connected driver off

4. Port

To facilitate the connection of our plug-in board to the main board, we have used a male header on the plug-in board and a female header on the main board. We opted for a single male connector with 2x10 pins, which meets our requirements. This configuration includes two pins dedicated to GND, three ports for GND_EARTH, and two pins for a 3.3V power supply. This arrangement ensures adequate power conductivity and reliable connection between the main board and the plug-in board.

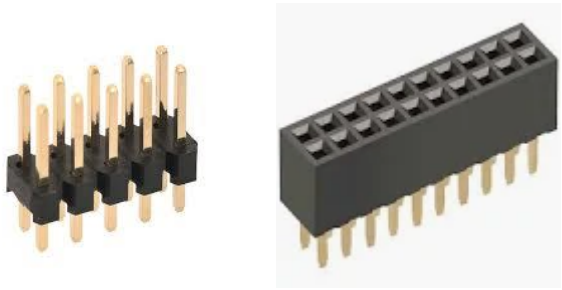


Figure 29: Port

Design:

1. Learning Cadence:

Generally, we learned about making part in cadence but mentor gave us a LIB file that has a part we need and they were used in company and were made in the past. So we just need to put them in the pages and connect them. At the first we create our pages with a default size and then we understand that some IC s are very big and also they have part types. So if we had added all of them to one page it would have been complicated. So we planned to have 5 pages look like our sections in previous part. Then we put related parts in each page. (More details in First Design part)

There are some important features in cadence that is very useful we will tell just some of them:

i. **Place Net Alias:**

This feature is for naming parts. It has some rules that will be mentioned in standards part.

ii. **Off-Page connector:**

When you are going to have more than one page for one board you need to make connection between them. Defining a net just with name will not make connection in cadence you need to use Off-Page connector to connect pins from two different pages. But for power and ground you don't need to define off-page connector because they are global nets. There is three type of off-page connector and it depends on pins typo: bidirectional – input – output

(**Note:** Using one off-page connector in one page is enough. If you want to use it more than once you don't need to add extra off-page just add it with net name)

iii. **Place Bus:**

This is for defining bus in schematic. At the first we used it for our 4-bit data but then we understand that we need just two bit so we removed. (More details in First Design part)

iv. **Netlist report:**

This one is for checking connections and is useful when you want to go to PCB design and making a layout.

2. Schematic design:

After putting parts in pages, we need to connect the pins. So we asked mentor to give us standard datasheets. We read datasheets to know which pins we need to use from our parts and type of them (I ,O ,I/O). And also we need to know which pins coupling capacitors or any resistor between GND or VCC.

We got pictures of our part's connections from previous products as a help.

After that we connected our pins and go through checking it more. (More details in Milestones part)

3. Standards:

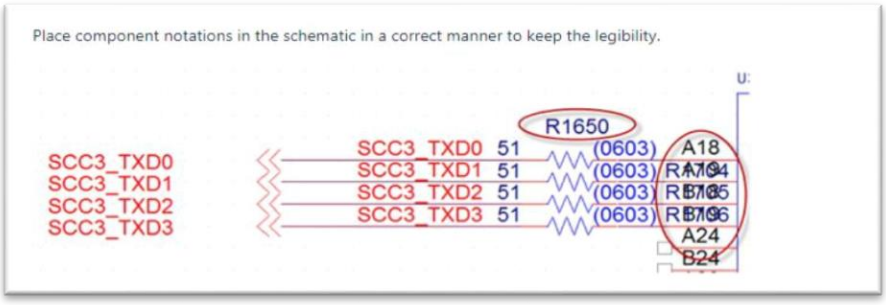
Mentor sent us some naming rules that related to companies rules to make it standard. Because it should be understandable for team and also another person out of the project team who maybe wants to make changes in design in the future.

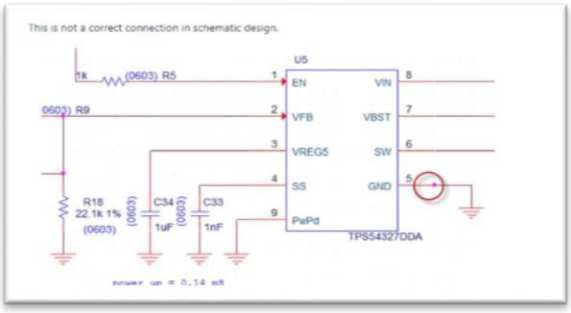
i. Some important points of naming:
ii. Try as much as you can to not use specific letter (/ or + or …) space in your names and also in directories. (Use underline instead of space)
iii. Put numbers for your pages and parts.
iv. Try not to use global nets with same voltage and different names.
v. Put numbers on elements like capacitors or resistors in the way that is more understandable. (We numbered them in this template: (Page number + number or letter like R313: resistor 13 of page 3)
vi. You should name your design file in standard way with date of edit. (Like: POEICPT1_R1_2024_8_13_3)

We have other rules that mentioned in this image:

Part status: (starting with double underline)

a. **nothing**: Placed: must be placed in all situations
b. **__NP**: Not Placed: must not be placed in all situations
c. **__NPS**: Not Placed in Sample: must not be placed in sample boards, but must be placed in production plan. e.g. series resistor in JTAG chain.
d. **__PS**: Placed in Sample: must be placed in sample boards only. e.g. JTAG bypass chip.





We Decided to design our schematic together and we get attached with date of the files.

First Design:

Our first design is for 23 of Mordad:

1. EEPROM

NC:

This pin can be connected to GND or left floating.

A1, A2:

The A1 and A2 pins are used to select the hardware device address and correspond to the sixth and fifth bit of the I 2 C seven-bit slave address. These pins can be directly connected to VCC or GND, allowing up to four devices on the same bus.

EEPROM_WP:

Connecting the WP pin to GND will ensure normal write operations. When the WP pin is connected to VCC, all write operations to the memory are inhibited.

EEPROM_SDA:

Open-drain bidirectional input/output pin used to serially transfer data to and from the device.

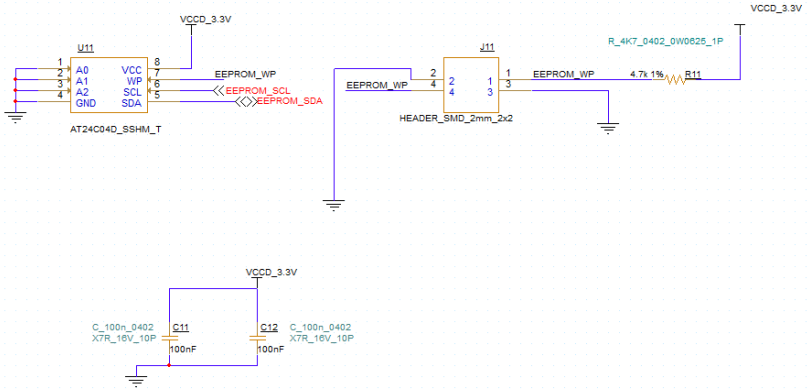
The SDA pin must be pulled-high using an external pull-up resistor (not to exceed 10KΩ in value) and may be wire-ORed with any number of other open-drain or open-collector pins from other devices on the same bus.

EEPROM_SCL:

Serial Clock. Is used to provide a clock to the device and to control the flow of data to and from the device. Command and input data present on the SDA pin is always latched in on the rising edge of SCL, while output data on the SDA pin is clocked out on the falling edge of SCL.

The SCL pin must either be forced high when the serial bus is idle or pulled-high using an external pull-up resistor

As far as our application is concerned, we only work with I2C and we don't want a high rate, the following part is sufficient. The pins that are supposed to be used in common are related to I2C4.
WP should be grounded only during reading and writing, and it can be disabled during the rest of the time...
Our EEPROM connected to the MCU with two off-pages. As datasheet said we connected unused pins to GND and we get EEPROM_WP from header because it can be changed by hardware according to the datasheet.



2. Ethernet

Mac Data Interface:

TXD:

Transmit data input pins

MII: TXD[3:0], that accept data synchronous to the TX_CLK (2.5 MHz in 10 Mb/s mode or 25 MHz in 100 Mb/s mode).

RMII: TXD[1:0], that accept data synchronous to the 50-MHz reference clock.

RXD:

Received data pins

MII: Nibble wide receive data signals driven synchronously to the RX_CLK, 25 RXD_0 36 MHz for 100 Mb/s mode, 2.5 MHz for 10 Mb/s mode). RXD[3:0] signals contain valid data when RXD_1 37 RX_DV is asserted.

RMII: 2-bits receive data signals, RXD[1:0], driven synchronously to the X1 clock, 50 MHz.

TX_CLK , RX_CLK::

25 MHz Transmit/Receive clock in 100Mb/s mode or 2.5 MHz in 10 Mb/s mode derived from the 25-MHz reference clock.

Unused in RMII mode. The device uses the X1 reference clock input as the 50-MHz reference for both transmit and receive.

RX_DV:

MII: Asserted high to indicate that valid data is present on the corresponding RXD[3:0].

RMII: Receive Data Valid indication independent of Carrier Sense.

RX_ER:

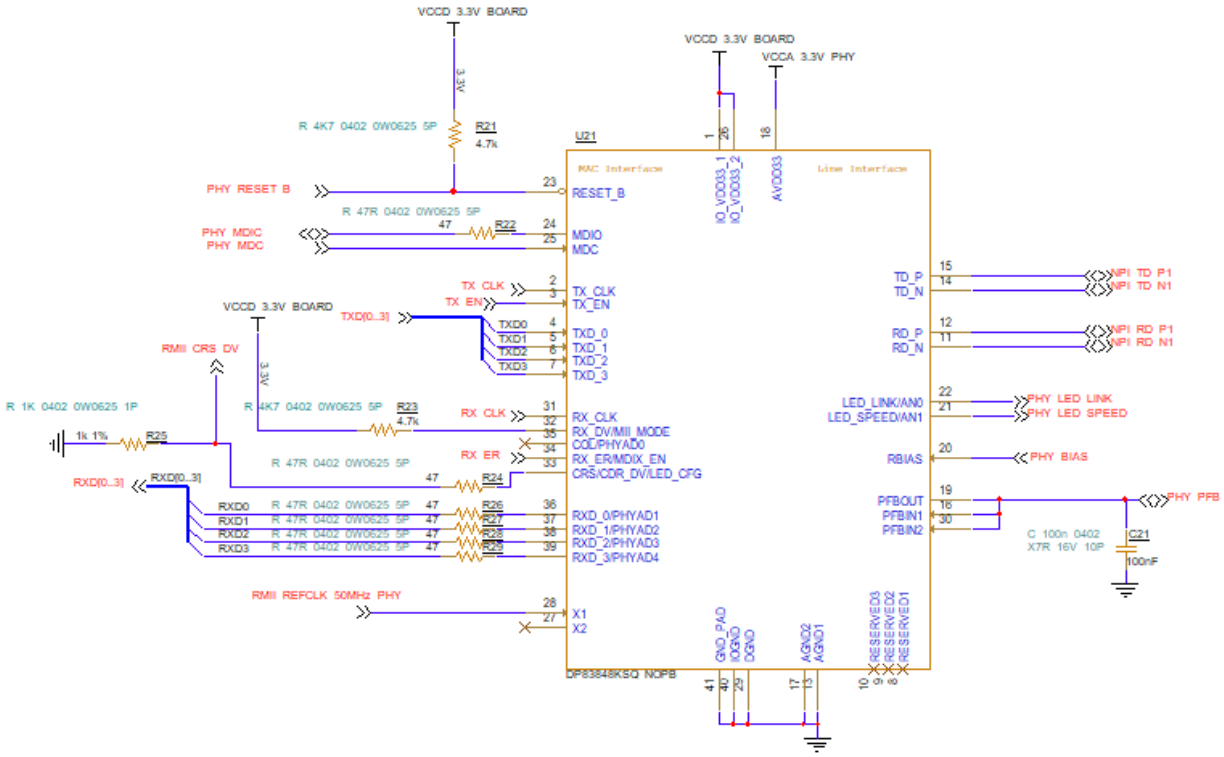
MII: Asserted high synchronously to RX_CLK to indicate that an invalid symbol has been detected within a received packet in 100 Mb/s mode.

RMII: Assert high synchronously to X1 whenever it detects a media error and RX_DV is asserted in 100 Mb/s mode.

This pin is not required to be used by a MAC, in either MII or RMII mode, because the PHY is required to corrupt data on a receive error.

TX_EN: Active high input indicates the presence of valid data inputs.
MII: TXD[3:0].
RMII: TXD[1:0].
RMII_CRS_DV:
MII: Asserted high to indicate the receive medium is non-idle.
RMII: This signal combines the RMII Carrier and V Receive Data Valid indications.
COL:
For RMII Specification, no COL signal is required.
Serial Management Interface:
PHY_MDIO:
Synchronous clock to the MDIO management data input/output serial interface which may be asynchronous to transmit and receive clocks. The maximum clock rate is 25 MHz with no minimum clock rate.
PHY_MDC:
Synchronous clock to the MDIO management data input/output signal that may be sourced by the station management entity or the PHY. This pin requires a 1.5-kΩ pull-up resistor
Reset:
PHY_RESET_B:
Active Low input that initializes or re-initializes the DP83848x.
Clock Interface:
X1(RMII_REFCLK):
This pin is the primary clock reference input for the RMII mode and must be connected to a 50-MHz 0.005% (+50 ppm) CMOS-level oscillator source.
X2:
This pin must be left unconnected if an external CMOS oscillator clock source is used.
10 Mb/s and 100 Mb/s PMD Interface:
TD-, TD+ , RD- , RD+:
Differential common driver transmit output/receive input (PMD Input/Output Pair). These differential pins are automatically configured to either 10BASE-T or 100BASE-TX signaling.
These pins require 3.3-V bias for operation.
Special Connections
RBIAS: Bias Resistor Connection. A 4.87-kΩ 1% resistor should be connected from RBIAS to GND
FPBOUT: Power Feedback Output. Connect this pin to PFBIN1 Power Feedback Input.
FPBIN1, FPBIN2: These pins are fed with power from PFBOUT pin. be connected close to each pin. I PFBIN2 30 Note: Do not supply power to these pins other than from PFBOUT.
RESERVED1,2,3: These pins must be left unconnected.

We used bus for RXD and TXD because at the first we don't know to use RMII or MII. We used resistors for protection and they should be in small ranges like 47 Ω. We have 2 global voltages, VCCD_3.3V_BOARD is our general VCC. As datasheet said, we have VCCA_3.3V_PHY for other usage. We have NPI off-pages at the right and PHY off-pages at the left. At the first we had a different datasheet and somehow we not covered it completely, so our coupling capacitor and resistors are not correct. Also we have LED pins, that we had thought we will use it in our board at the first.



3. MCU

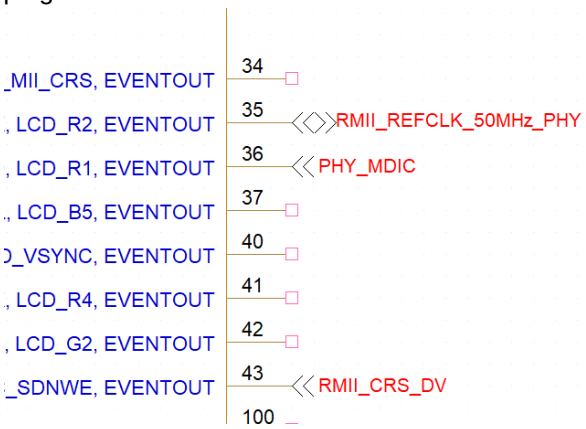
We have RXD and TXD data bus, clock and enable. Also pins that related to PHY and URT and I2C.

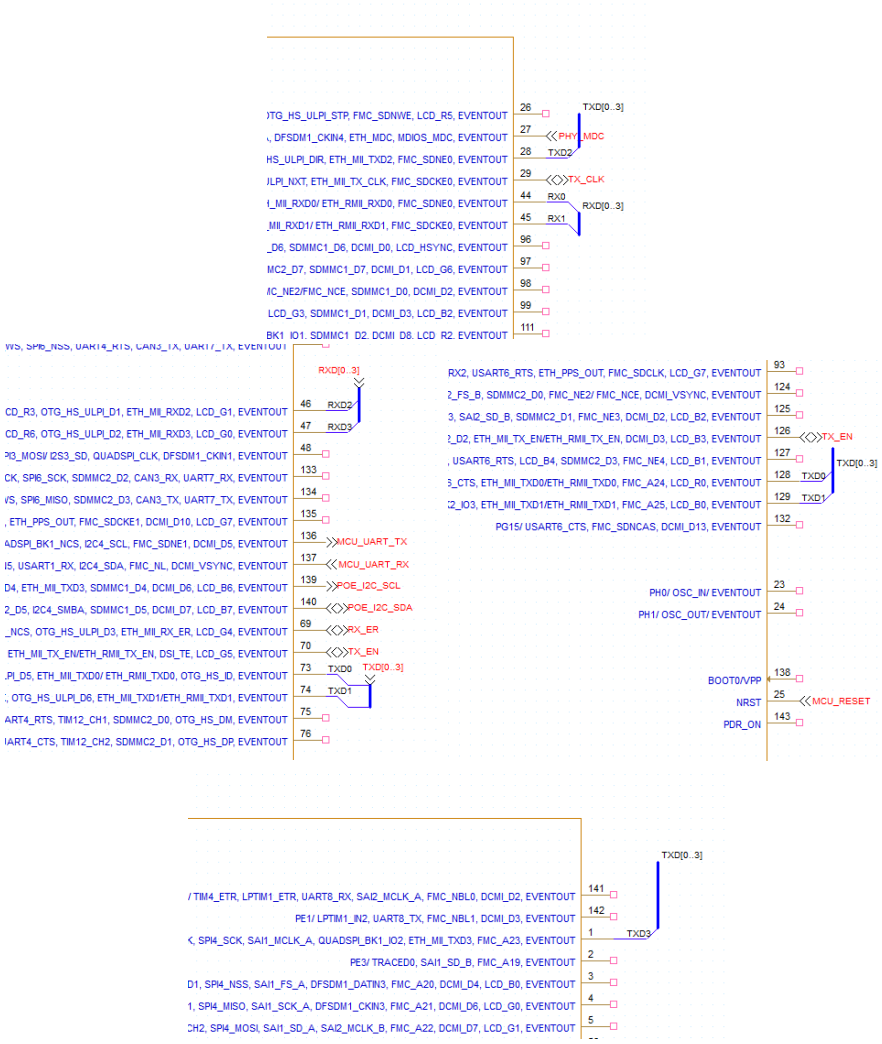
MCU_URT_TX, MCU_URT_RX:

UART protocol pins between plug-in board microcontroller and RS-232 driver

POE_I2C_SCL,POE_I2C_SDA:

I2C protocol pins between plug-in board microcontroller and Main board micro chip.

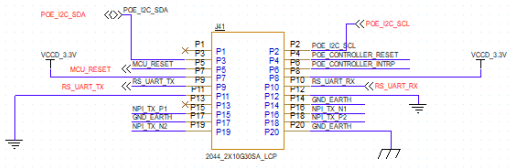




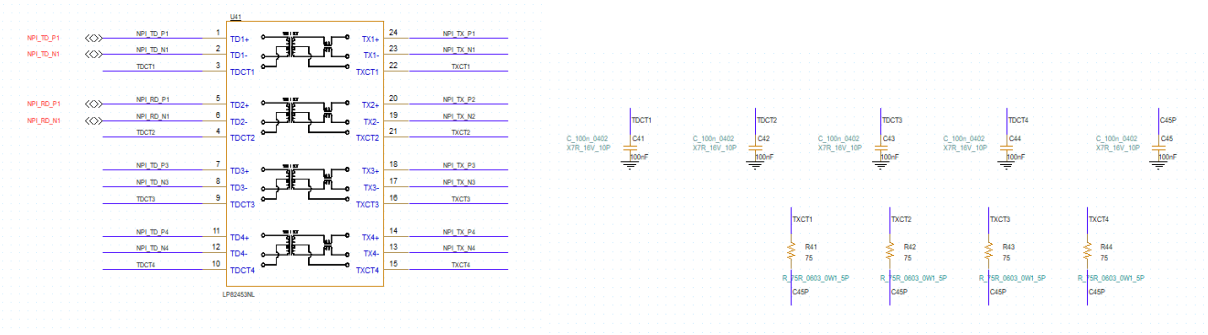
This time power pins of MCU were not connected in this state.

4. Ports

We labeled our pins based on our main board's pin. We have NPI,I2C,URT to communicate with main board and also we got GND_EARTH and GND and VCC.

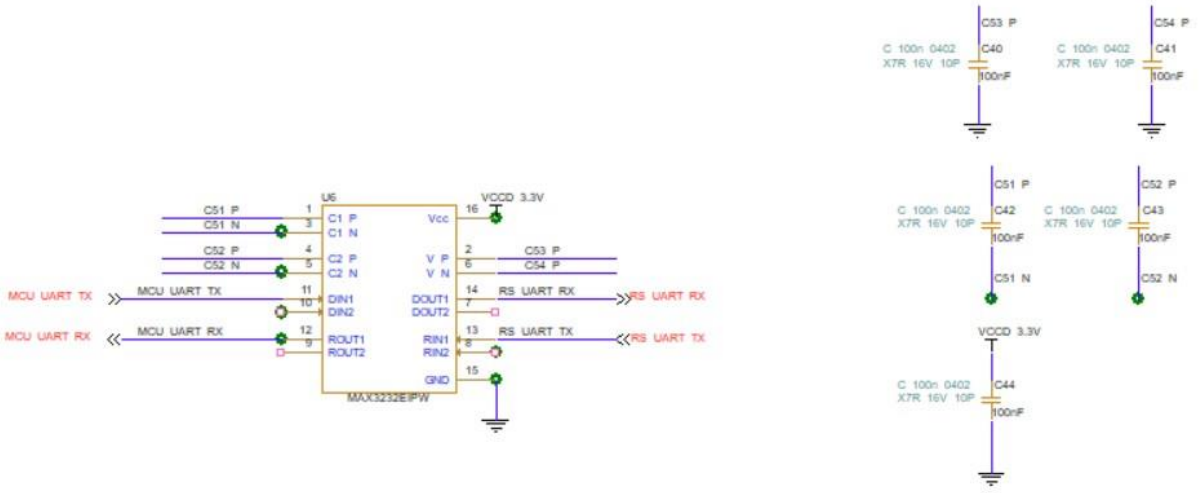


And we have our transformer in the last port with peripheral circuit , including capacitors and resistors that they act as a low pass filter .



5. URT

We have URT off-pages to connect to MCU and coupling capacitors. It takes URT from port and sends it to MCU.

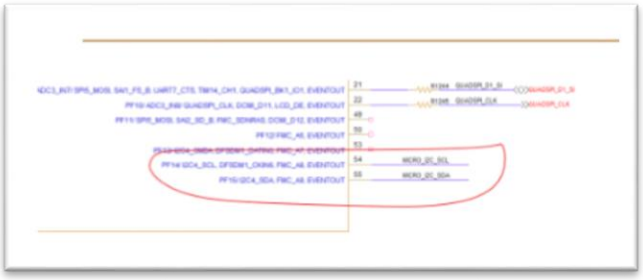


Milestones:

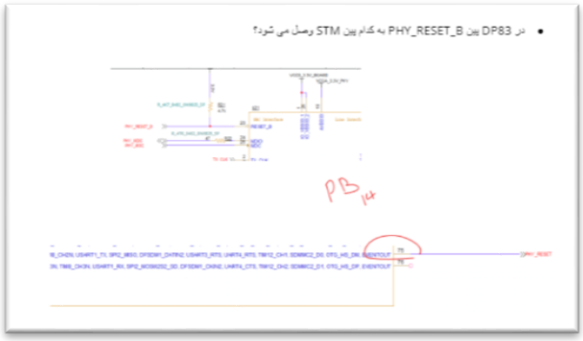
For this part we used a creative method. We made a google doc with our mentor and we asked our questions there with picture and she answered us with red color. Again we asked new questions with blue color and mentor answered. It was very efficient way. We mentioned some important parts for full doc check the link below:

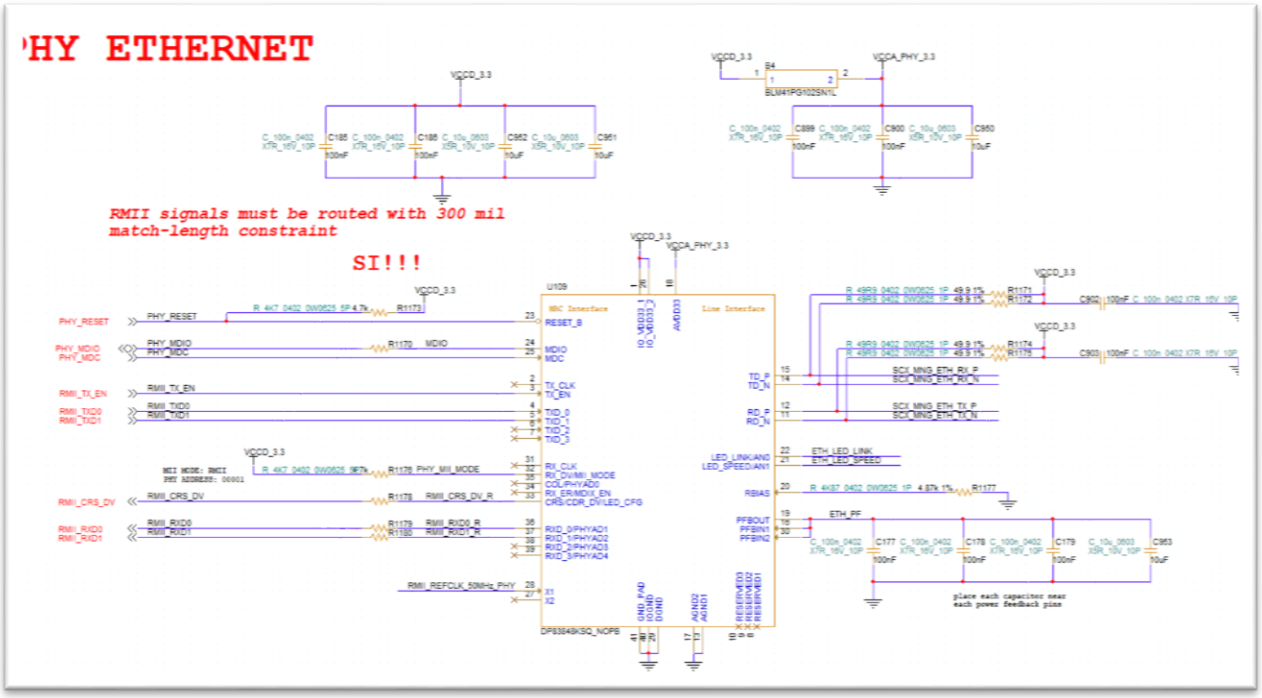
[Question doc]

EEPROM:



Ethernet:


در DP83 پین PHY_RESET_B به کدام پین STM وصل می شود؟


HY ETHERNET
RMII signals must be routed with 300 mil match-length constraint
SI!!!
PHY_RESET
PHY_MDIO
PHY_MDC
RMII_TX_EN
RMII_TXD0
RMII_TXD1
RMII_CRS_DV
RMII_RXD0
RMII_RXD1
RMII_REFCLK_50MHz_PHY
DP83849IKSQ_NOPB
place each capacitor near each power feedback pins

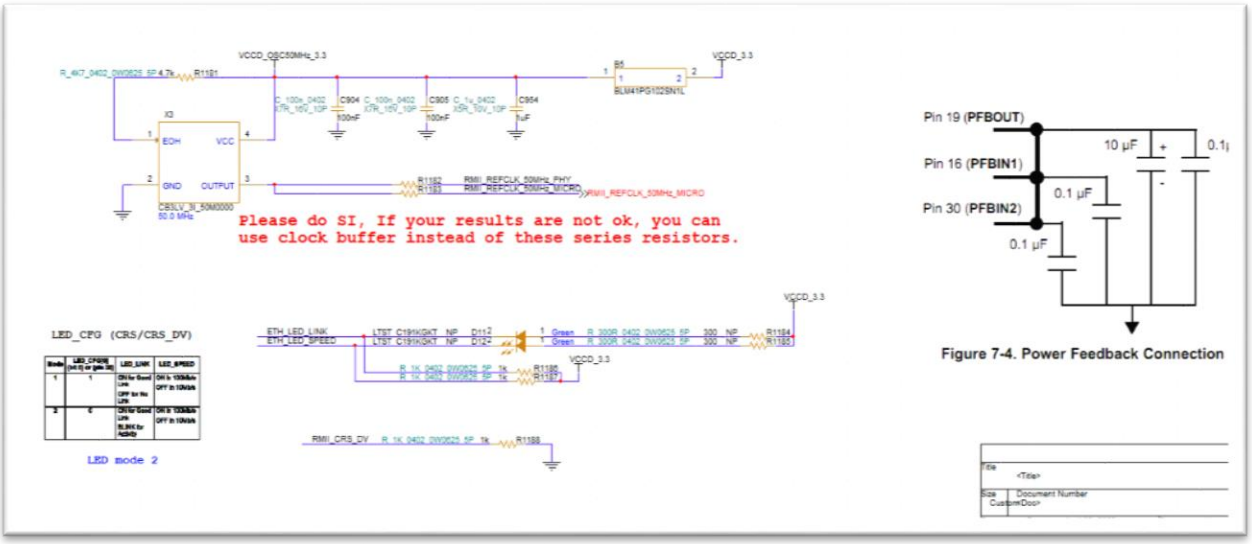
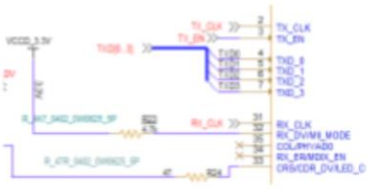


Figure 7-4. Power Feedback Connection

- در تصویر اول tx_clk و rx_clk می بینیم که NC ولی در مدار ما نیست کدام درست است؟



آیا طبق دیتاشیت در تصویر زیر، برای RMII اینا مورد استفاده ست؟ نمی خواستیم RMII باشیم؟

TEXAS INSTRUMENTS

www.ti.com

DP83848H, DP83848J, DP83848K, DP83848M, DP83848T

SNLS250E – MAY 2008 – REVISED APRIL 2015

SIGNAL NAME	TYPE	PIN #	DESCRIPTION
RX_DV	O, PD	32	**MII RECEIVE DATA VALID:** Asserted high to indicate that valid data is present on the corresponding RXD[3:0]. **RMII Synchronous Receive Data Valid:** This signal provides the RMII Receive Data Valid indication independent of Carrier Sense.
RX_ER	S, O, PU	34	**MII RECEIVE ERROR:** Asserted high synchronously to RX_CLK to indicate that an invalid symbol has been detected within a received packet in 100 Mb/s mode. **RMII RECEIVE ERROR:** Assert high synchronously to X1 whenever it detects a media e RX_DV is asserted in 100 Mb/s mode. This pin is not required to be used by a MAC, in either MII or RMII mode, because the Phy is required to corrupt data on a receive error.
RXD_0 RXD_1 RXD_2 RXD_3	S, O, PD	36 37 38 39	**MII RECEIVE DATA:** Nibble wide receive data signals driven synchronously to the RX_CLK, 25 MHz for 100 Mb/s mode, 2.5 MHz for 10 Mb/s mode). RXD[3:0] signals contain valid data when RX_DV is asserted. **RMII RECEIVE DATA:** 2-bits receive data signals, RXD[1:0], driven synchronously to the X1 clock, 50 MHz.
TX_CLK	O	2	**MII TRANSMIT CLOCK:** 25 MHz Transmit clock output in 100Mb/s mode or 2.5 MHz in 10 Mb/s mode derived from the 25-MHz reference clock. Unused in RMII mode. The device uses the X1 reference clock input as the 50-MHz reference for both transmit and receive.

EN English (Unite

4.4 MAC Data Interface

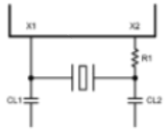
SIGNAL NAME	TYPE	PIN #	DESCRIPTION
COL	S, O, PU	35	**MII COLLISION DETECT:** Asserted high to indicate detection of a collision condition (simultaneous transmit and receive activity) in 10 Mb/s and 100 Mb/s Half-Duplex Modes. While in 10BASE-T Half-Duplex mode with heartbeat enabled this pin is also asserted for a duration of approximately 1 µs at the end of transmission to indicate heartbeat (SQE test). In Full Duplex Mode, for 10 Mb/s or 100 Mb/s operation, this signal is always logic 0. There is no heartbeat function during 10Mb/s full duplex operation. **RMII COLLISION DETECT:** Per the RMII Specification, no COL signal is required. The MAC will recover CRS from the CRS_DV signal and use that along with its TX_EN signal to determine collision.
CRS/CRS_DV	S, O, PU	33	**MII CARRIER SENSE:** Asserted high to indicate the receive medium is non-idle. **RMII CARRIER SENSE/RECEIVE DATA VALID:** This signal combines the RMII Carrier and Receive Data Valid indications. For a detailed description of this signal, see the RMII Specification.
RX_CLK	O	31	**MII RECEIVE CLOCK:** Provides the 25 MHz recovered receive clocks for 100 Mb/s mode and 2.5 MHz for 10 Mb/s mode. Unused in RMII mode. The device uses the X1 reference clock input as the 50 MHz reference for both transmit and receive.

- بعضی از مقاومت ها تلورانس یک درصد ندارند و باید اصلاح شود
- فعلا دستی اصلاح کنین در شماتیک که مقدار درست و نیازمندی درستش جلو چشم باشه بعدا لایبری درست جایگزین بشه
- ما در سمت mcu دوبار txd0 و txd1 گذاشته ایم (126و127و73و74)و قطعا یکی باید پاک شود ولی در شماتیک معیاری نداریم شاید در pcb و تاثیری که جایگاه پین ها در پیچیدگی فیزیک مدار می گذارد معیار خوبی باشد
- نمی دونم درست مشکل رو فهمیدم یا نه. از نظر روت بله شاید PCB معیار قرار بگیرد برای راحتی ... ولی آیا طبق عکس هایی که از سمت میکرو فرستادم نمیشه مشابه همونا برای یکدست بودن از همان پین ها استفاده کنیم ؟ عکس هایی که جدیدا 5 6 سوال پایین تر فرستاده بودم

- برای txd ها در سمت mcu مقاومت 4.7 کا گذاشتیم ایا درست است؟ یا 47 اهمی بهتر است؟

کیلو اهم مقدار زیادی برای سورس ترمینیشن هست. از اردر اهم باید باشد

- در ص 72 دیتاشیت گفته است برای RX_CLK,TX_CLK , RXD در MII مقاومت 50 اهم استفاده می کنیم ولی راجب RMII چیزی نگفته است .(نکته ی دیگر اینکه پکیج ما 47 اهمی است آیا ایرادی دارد همان را بگذاریم؟) 47 خوبه
- این قسمت ETHERNET ما طبق تصویر ارسالی طراحی کرده ایم ولی در دیتاشیت با این کریستال مواجه شدیم از ص 70 و در ص 7 دیتا شیت گفته که برای RMII نمی خواهد (این را چک کنید)



من بررسی کردم برای اینکه کلاک اترنت مون دقت کافی رو داشته باشه بهتره به اسیلاتور داخلی میکرو اکتفا نکنیم و به اسیلاتور در مدار بذاریم که کلاک رفرنس میکرو و فای را تامین کند

بنابراین در صورت استفاده از اسیلاتور نیاز به کریستال نیست

برای بخش فای سیگنال های اسیلاتور رو در عکس بالاتر گذاشتم

- تصویر دوم برای وقتی است که برد فضای led داشته باشد و مدار جانبی است یا اجباری است؟
- همان طور هم که قبلا گفتم در این برد نیازی به LED نداریم پس استفاده نمی کنیم و میشه آزاد گذاشت اما برای اینکه امکان این رو داشته باشیم که در صورت نیاز احتمال ایجاد نویز روی این پین ها را کاهش بدیم طبق این گفته دیتاشیت میتونیم مقاومت PD برای این پین ها بذاریم ولی به صورت دیفالت مونتاژی نباشن و فقط جاش رو لحاظ کرده باشیم.
یعنی با پسوند NP__ لحاظ کنیم

SIGNAL NAME	TYPE	PIN #	DESCRIPTION
PHYAD0 (COL) PHYAD1 (RXD_0) PHYAD2 (RXD_1) PHYAD3 (RXD_2) PHYAD4 (RXD_3)	S, O, PU S, O, PD	35 36 37 38 39	**PHY ADDRESS [4:0]:** The DP83848x provides five PHY address pins, the state of which are latched into the PHYCTRL register at system Hardware-Reset. The DP83848x supports PHY Address strapping values 0 (<00000>) through 31 (<11111>) **A PHY Address of 0 puts the part into the MII Isolate Mode.** The MII isolate mode mus be selected by strapping Phy Address 0; changing to Address 0 by register write will not pu the Phy in the MII isolate mode. Refer to *Section 6.4.4* for additional information. PHYAD0 pin has weak internal pullup resistor. PHYAD[4:1] pins have weak internal pulldown resistors.
AN0 (LED_LINK) AN1 (LED_SPEED)(1)	S, O, PU S, O, PU	22 21	These input pins control the advertised operating mode of the device according to the following table. The value on these pins are set by connecting them to GND (0) or VCC (1) through 2.2-kΩ resistors. These pins should NEVER be connected directly to GND or VCC. The value set at this input is latched into the DP83848x at Hardware-Reset. The float/pulldown status of these pins are latched into the Basic Mode Control Registe and the Auto_Negotiation Advertisement Register during Hardware-Reset. The default for DP83848x is 11 because these pins have an internal pullup.

AN1(1)	AN0	Advertised Mode
0	0	10BASE-T, Half/full-Duplex
0	1	100BASE-TX, Half/full-Duplex
1	0	10BASE-T, Half-Duplex 100BASE-TX, Half-Duplex
1	1	10BASE-T, Half/Full-Duplex 100BASE-TX, Hal/Full-Duplex

MII_MODE (RX_DV)	S, O, PD	32	**MII MODE SELECT:** This strapping option determines the operating mode of the MAC Data Interface. Default operation (No pullup) will enable normal MII Mode of operation. Strapping MII_MODE high will cause the device to be in RMII mode of operation. Because the pin includes an internal pulldown, the default value is 0. The following table details the configuration:

MIL_MODE	MAC Interface Mode
0	MII Mode
1	RMII Mode

LED_CFG (CRS/CRS_DV)	S, O, PU	33	**LED CONFIGURATION:** This strapping option determines the mode of operation of the LED pins. Default is Mode 1. Mode 1 and Mode 2 can be controlled through the strap option. All modes are configurable through register access. See Table 6-2 for LED Mode Selection.
MDIX_EN (RX_ER)	S, O, PU	34	**MDIX ENABLE:** Default is to enable MDIX. This strapping option disables Auto-MDIX. An external pulldown will disable Auto-MDIX mode.

(1) AN1 (LED_SPEED) is only available on the DP83848 I/K

MCU:

- آیا برای محاسبه ی جریان مصرفی برای stm32 لازمه جریان dynamic رو هم لحاظ کنیم؟

بله اگر خیلی دقیق بخوایم محاسبه کنیم لازمه در نظر بگیریم اگر هم دیدیم میزان جریان داینامیک اضافه شده در مقابل کل جریان خیلی نیست میشه صرف نظر کرد. برای typ ظاهرا میشه از stmcube هم کمک گرفت البته خودم تا به حال این کار رو نکردم ببینم چه طور میشه استفاده کرد بهتون میگم. برای محاسبه max هم میشه از جداول بخش peripherals استفاده کرد به اضافه همین جریان داینامیک

- All I/O pins are in input mode with a static value at V_{DD} or V_{SS} (no load).
- All peripherals are disabled except if it is explicitly mentioned.
- The Flash memory access time is adjusted both to f_{HCLK} frequency and V_{DD} range (see *Table 19: Limitations depending on the operating power supply range*).
- When the regulator is ON, the voltage scaling and over-drive mode are adjusted to f_{HCLK} frequency as follows:
  - Scale 3 for $f_{HCLK} \leq 144$ MHz
  - Scale 2 for $144 \text{ MHz} < f_{HCLK} \leq 168$ MHz
  - Scale 1 for $168 \text{ MHz} < f_{HCLK} \leq 216$ MHz. The over-drive is only ON at 216 MHz.
- When the regulator is OFF, the V12 is provided externally as described in *Table 18: General operating conditions*:
- The system clock is HCLK, $f_{PCLK1} = f_{HCLK}/4$, and $f_{PCLK2} = f_{HCLK}/2$.
- External clock frequency is 25 MHz and PLL is ON when f_{HCLK} is higher than 25 MHz.
- The typical current consumption values are obtained for $1.7 \text{ V} \leq V_{DD} \leq 3.6$ V voltage range and for $T_A = 25$ °C unless otherwise specified.
- The maximum values are obtained for $1.7 \text{ V} \leq V_{DD} \leq 3.6$ V voltage range and a maximum ambient temperature (T_A) unless otherwise specified.
- For the voltage range $1.7 \text{ V} \leq V_{DD} \leq 3.6$ V, the maximum frequency is 180 MHz.

Table 25. Typical and maximum current consumption in Run mode, code with data processing running from ITCM RAM, regulator ON

Symbol	Parameter	Conditions	f_{HCLK} (MHz)	Typ	Max(1)			Unit
					T_A = 25 °C	T_A = 85 °C	T_A = 105 °C	
I_{DD}	Supply current in RUN mode	All peripherals enabled(2)(3)	216	193	221(4)	258(4)	-	mA
			200	179	207	244	279	
			180	159	176(4)	210(4)	238(4)	
			168	142	156	187	211	
			144	122	135	167	190	
			60	49	55	81	103	
			25	23	28	54	76	
		All peripherals disabled(3)	216	95	107(4)	153(4)	-	
			200	88	100	146	180	
			180	78	88(4)	122(4)	147(4)	
			168	70	78	109	133	
			144	60	68	99	123	
			60	24	29	55	76	
			25	12	16	42	63	

1. Guaranteed by characterization results, unless otherwise specified.

یک مثال در یکی از بردها برای محاسبه ماکزیمم :

Signal	Component	Max Input Capacitance (Pf)	Cext (Pf)	I/O Toggling Frequncy (MHz)	Typ. I/O Switching Current on STM32 Pin (mA)
I2C_SCL	PCA9548	21	30	100-400 KHz	0.3
I2C_SDA					0.3
CLKIN	74HC165PW	3.5	10	48	3.1
DATAIN					3.1
LD	74HC595PW	3.5	10	20	1.8
SHCP					1.8
STCP					1.8
OE_B					1.8
RST_B					1.8
CTRLs	SI5345	2	10	2	1.6
CTRLs	74HC164D	3.5	10	20	3.6

- آیا برای پروتکل I2c از پایه I2c_smba هم استفاده میکنیم؟

خیر تا به الان همچین نیازمندی اعلام نشده

- در میکروکنترلر stm32 VDDUSB چه کاربردی دارد؟

ظاهرا برای کاربردهایی مثل استفاده از usb می باشد که در صورت استفاده نکردن میشه به همون VDD وصل کرد

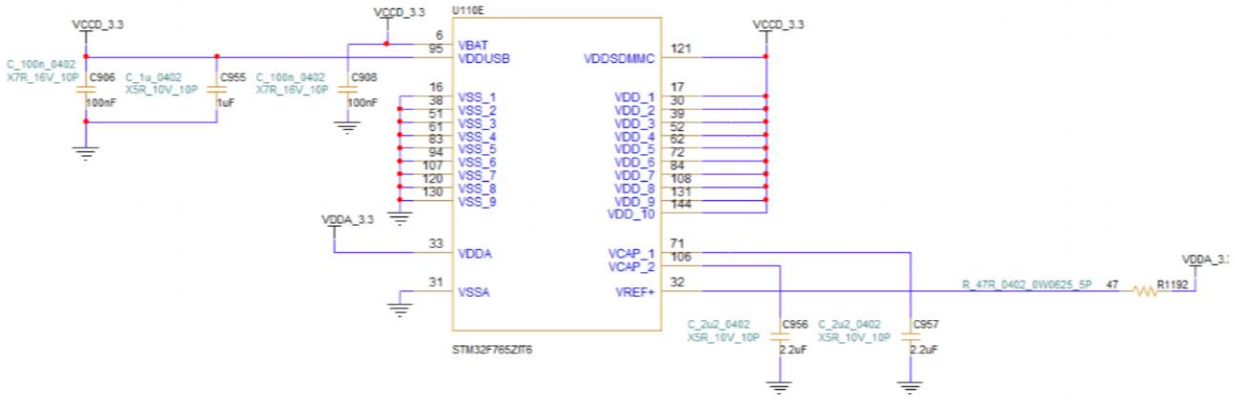
2.17 Power supply schemes

- V_{DD} = 1.7 to 3.6 V: external power supply for I/Os and the internal regulator (when enabled), provided externally through V_{DD} pins.
- V_{SSA}, V_{DDA} = 1.7 to 3.6 V: external analog power supplies for ADC, DAC, Reset blocks, RCs and PLL. V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.
- V_{BAT} = 1.65 to 3.6 V: power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when V_{DD} is not present.

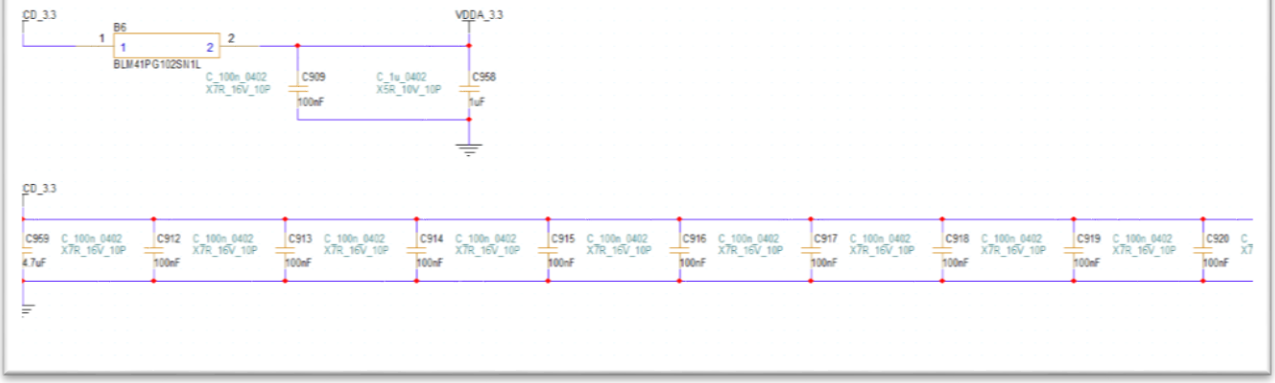
Note: *V_{DD}/V_{DDA} minimum value of 1.7 V is obtained when the internal reset is OFF (refer to Section 2.18.2: Internal reset OFF). Refer to Table 3: Voltage regulator configuration mode versus device operating mode to identify the packages supporting this option.*

- $V_{DDSDMMC}$ can be connected either to V_{DD} or an external independent power supply (1.8 to 3.6V) for SDMMC2 pins (clock, command, and 4-bit data). For example, when the device is powered at 1.8V, an independent power supply 2.7V can be connected to $V_{DDSDMMC}$.When the $V_{DDSDMMC}$ is connected to a separated power supply, it is independent from V_{DD} or V_{DDA} but it must be the last supply to be provided and the first to disappear. The following conditions $V_{DDSDMMC}$ must be respected:
  - During the power-on phase (V_{DD} < V_{DD_MIN}), $V_{DDSDMMC}$ should be always lower than V_{DD}
  - During the power-down phase (V_{DD} < V_{DD_MIN}), $V_{DDSDMMC}$ should be always lower than V_{DD}
  - The $V_{DDSDMMC}$ rising and falling time rate specifications must be respected
  - In operating mode phase, $V_{DDSDMMC}$ could be lower or higher than V_{DD}: All associated GPIOs powered by $V_{DDSDMMC}$ are operating between $V_{DDSDMMC\_MIN}$ and $V_{DDSDMMC\_MAX}$.
- V_{DDUSB} can be connected either to V_{DD} or an external independent power supply (3.0 to 3.6V) for USB transceivers (refer to *Figure 4* and *Figure 5*). For example, when the device is powered at 1.8V, an independent power supply 3.3V can be connected to V_{DDUSB}. When the V_{DDUSB} is connected to a separated power supply, it is independent from V_{DD} or V_{DDA} but it must be the last supply to be provided and the first to

STM32 POWER



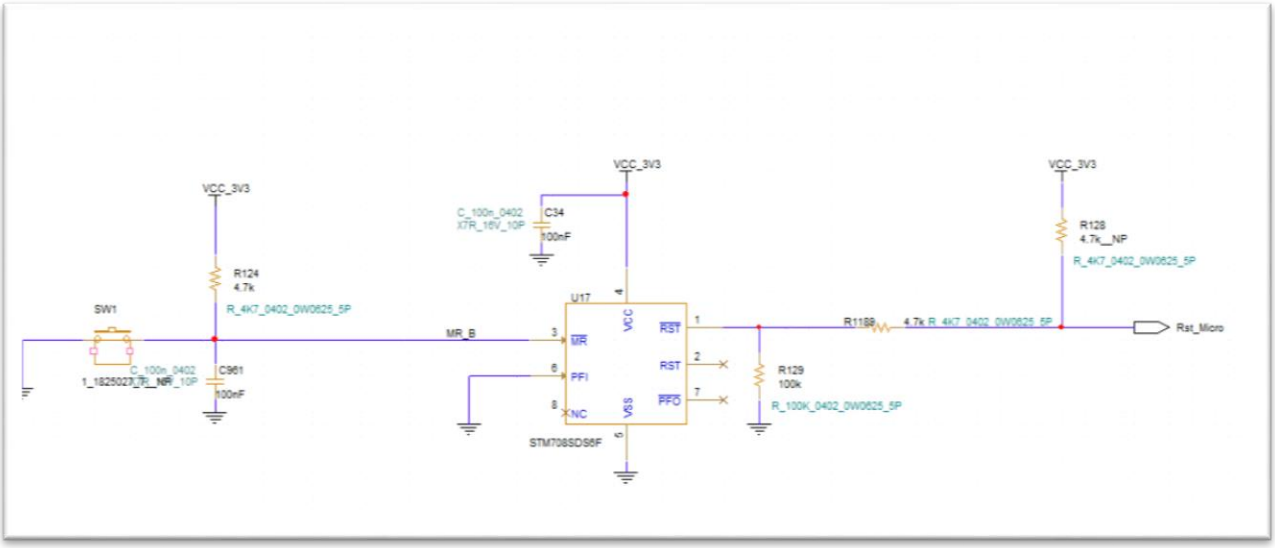
See AN1709 P.30-33
for PCB Design Guidelines

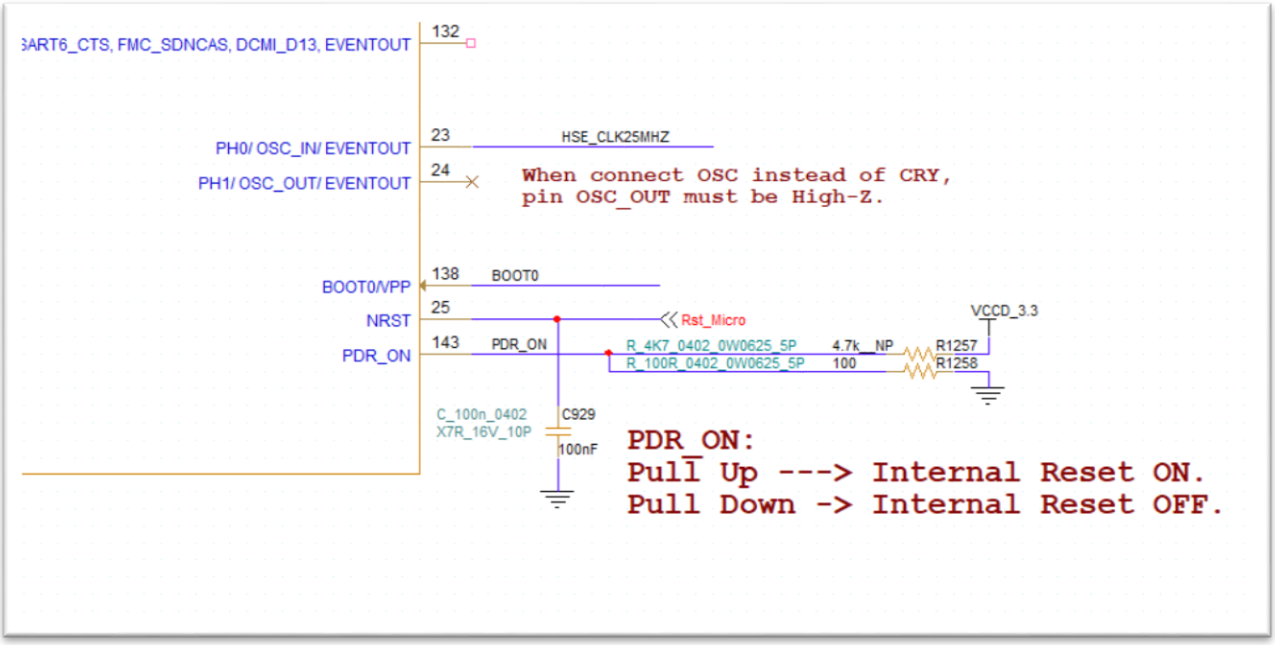


- قرار گرفتن کلید ریست stm32 و مدار مرتبط روی برد بالا یا پایین

بخشی از کلید ریست و مدارش در برد پایین می باشد

معمولا برای پروتکشن بیشتر برای ریست از چیپ STM708SDS6F استفاده میشود. ولی در صورت نداشتن حساسیت بالا میشه از طراحی حذف کرد. از طرفی فک میکنم محدودیت فضا هم در این برد داشته باشیم. مگر اینکه فعلا بذاریم و بعدا اگر دچار محدودیت فضا یا هزینه یا عدم حساسیت هستیم حذف کنیم ولی حس اولیه م این هست که روی این برد نباشه





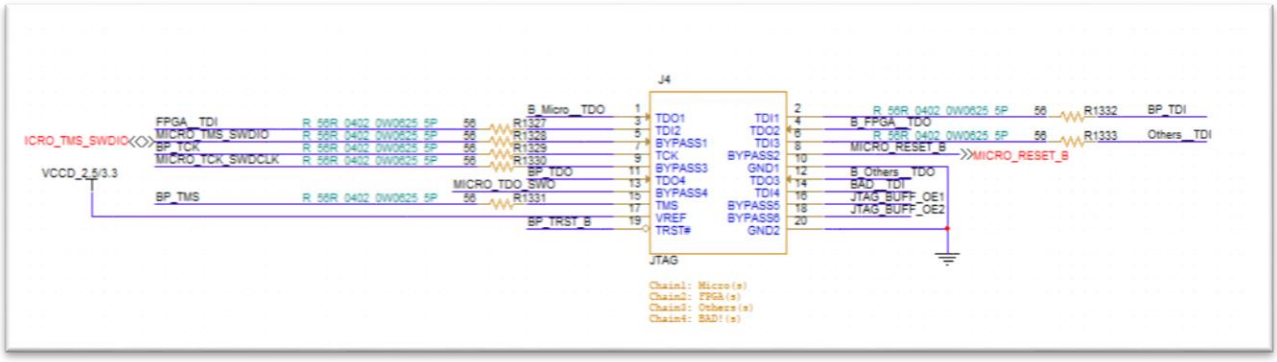
- آیا برای پروگرم کردن stm32 نیازه یه مدار جانبی داشته باشیم؟

در بررسی اولیه از بلوک دیاگرام میکرو تصورم این بود که شاید uart کفایت کنه اما بعد از کمی بررسی به نظر میرسه شاید محدودیت هایی ایجاد کنه و برای پروگرمینگ پیچیده تر برای core میکروی stm32 بهتره همون SWD رو داشته باشیم

یه کانکتور مخصوص برای همین مسیرهای جیتگ و پروگرمینگ داریم که میشه اونو کنار میکرو بذاریم

چیزی که ذهنمو مشغول کرده این هست که شاید بازم محدودیت فضا و مکانیکی داشته باشیم اینکه روی یه برد پلاگین بخواد یه کانکتور نسبتا بلند جیتگ هم قرار بگیره. مگر اینکه این سیگنال ها رو هم بدیم پایین فقط سورس ترمینیشن یعنی مقاومت سری با این سیگنال ها در نزدیکی پین ها بذاریم برای بحث های تطبیق امپدانس و کاهش رفلکشن در برد. در بررسی بعدی باید ببینیم روی همین کانکتوری که داریم جا برای این سیگنال ها داریم یا نه

مثال مدار جانبی به صورت زیر است



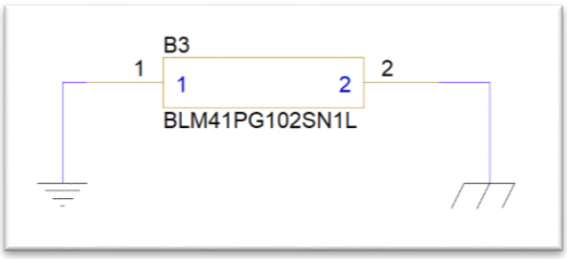
Here we decided to add sixth page as an JTAG Page

Ports:

- قرار دادن GND_EARTH در طراحی

در برد زمین سیگنال ها برای بسته شدن مدار GND می باشد و بدنه و شیلدها earth و با یک BLM در یک نقطه می توانیم این دو را بهم وصل میکنیم

برای ترانسفورمری هم که گذاشتیم چون نقش ایزولاسیون هم داره به نظرم سمت آنالوگ مدار بهتره رفرانس EARTH داشته باشه و سمت دیجیتال GND که زمین های این دو بخش جدا باشن



UART:

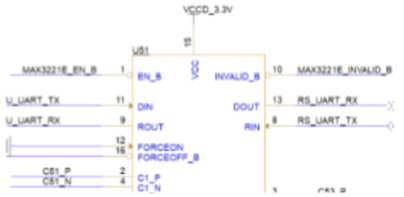
- چرا نیازه برای قطعه LP82453NL خازن و مقاومت گذاشته بشه.

این مقاومت و خازن ها می توانند نقش فیلتر و اصلاح عملکرد و شکل سیگنال در لایه ای که داریم اطلاعات رو انتقال میدن، داشته باشن. مثلا مقاومت ها یا ترکیبشون با خازن می توانند به عنوان ترمینیشن هایی برای بحث های تطبیق امپدانس و کاهش رفلکشن های سیگنال در سر دایورسنگی های مدار باشند و خازن ها برای موارد نویزی و EMC و EMI می توانند نقش مثبت داشته باشن

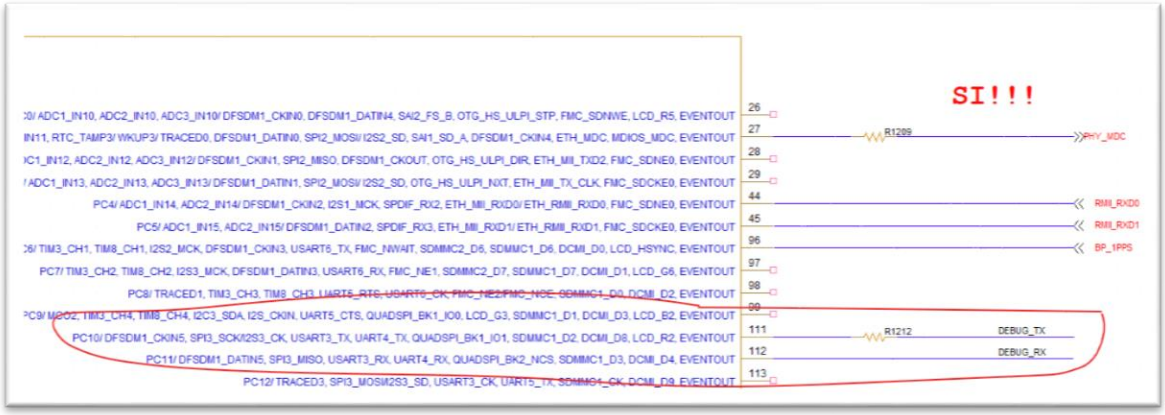
- برای پروتکل NPI آیا پین های MRDY , SRDY نیازه قرار داده بشه؟

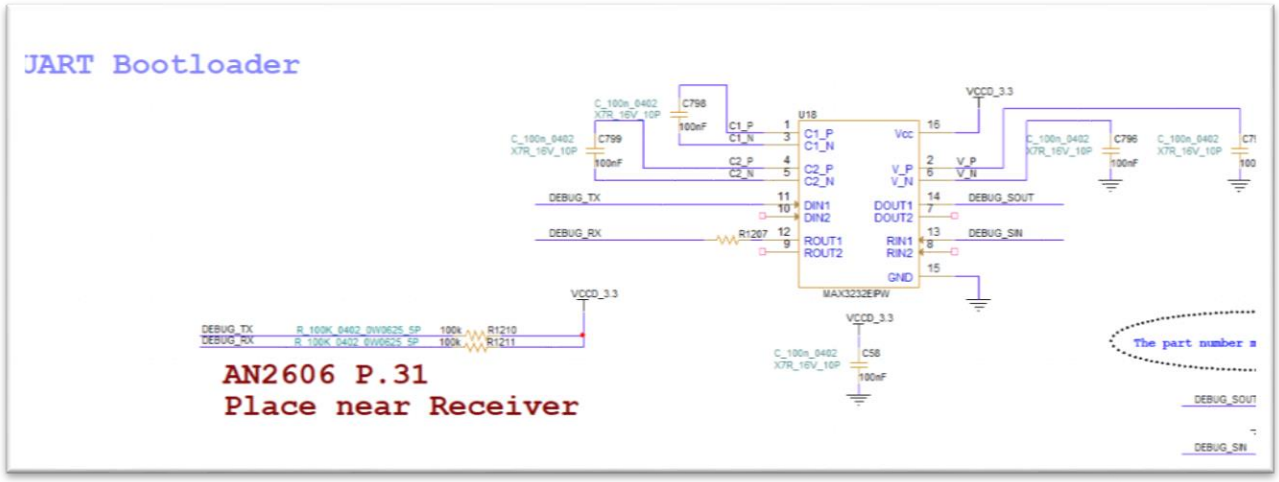
در کاربردهای قبلی استفاده نشده است

- در UART این پین ها به چی وصل می شوند؟(MCU?) آیا مدار جانبی می خواهند؟



بهتر است از قطعه max3232 با مدار زیر استفاده شود





Here we decided to change URT part to MAX3232EPIW.

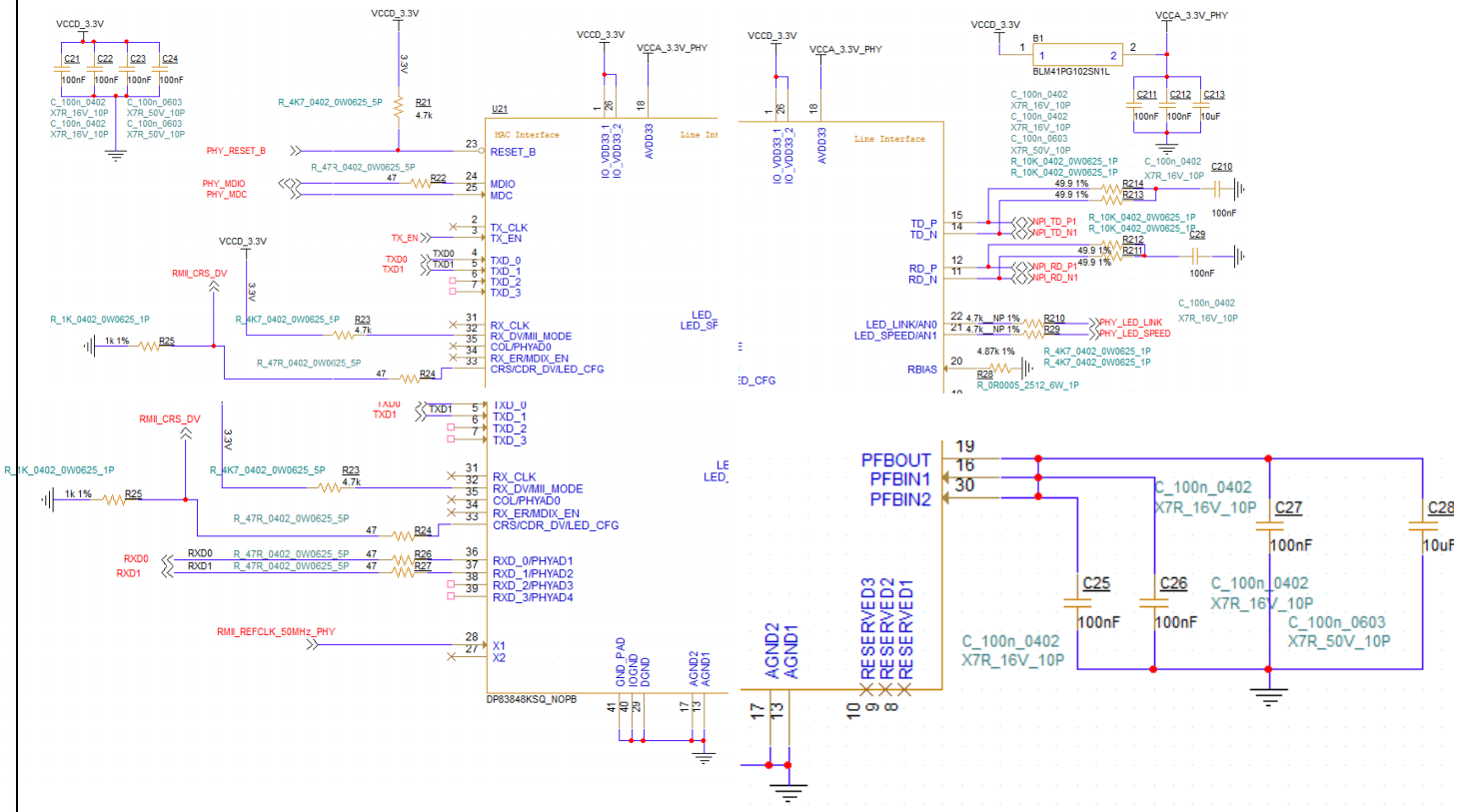
Second Design:

EEPROM:

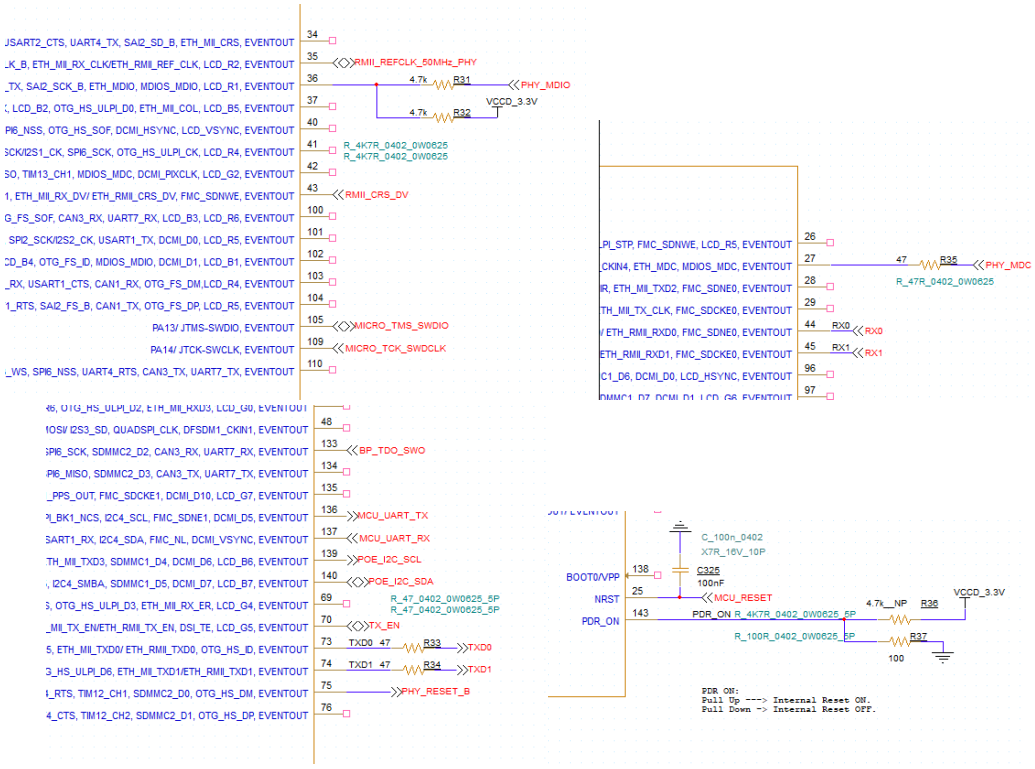
Not changed

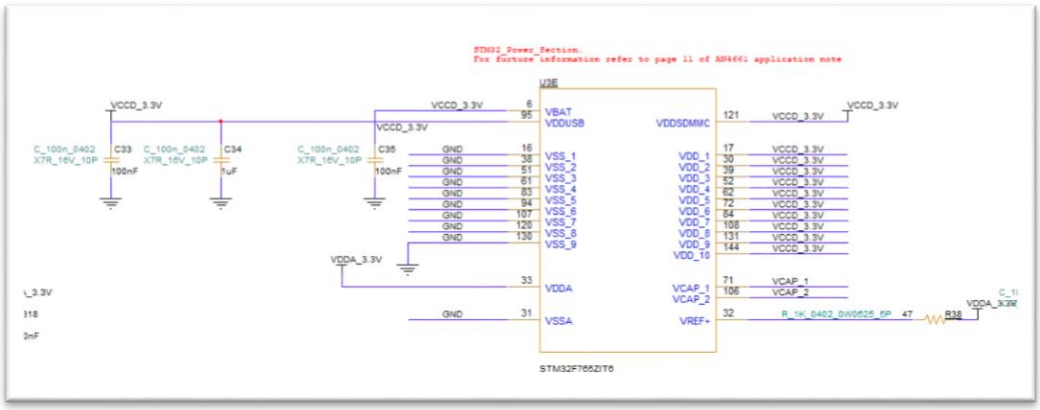
Ethernet:

We deleted TXD bus because we just need two bites for RMII.



MCU:



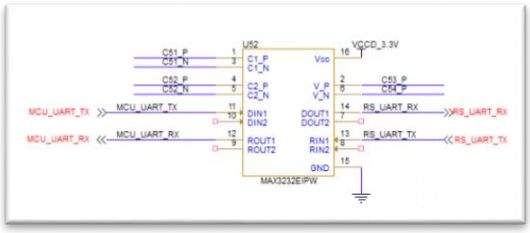


We didn't put picture of coupling capacitors but it's as the picture in the milestone part.

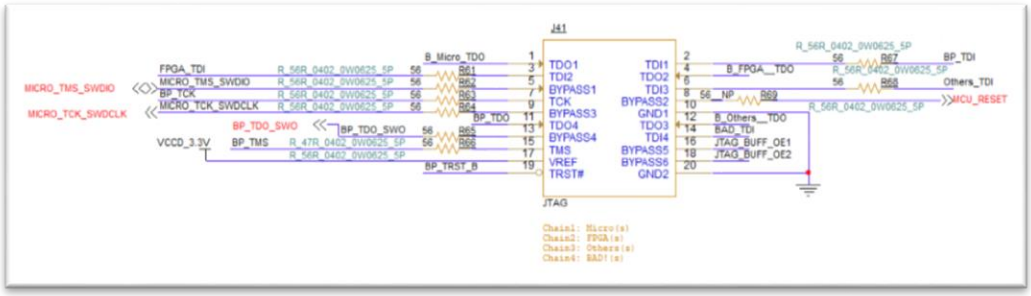
Ports:

Not changed

UART:



JTAG:



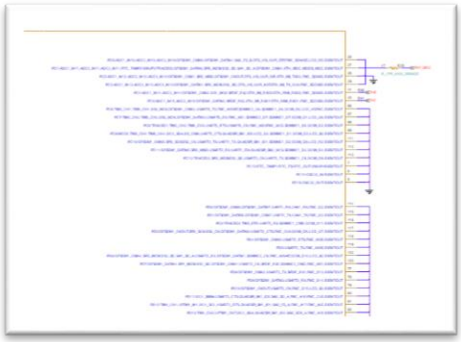
Review:

Ethernet:

- We checked between MCU and Ethernet for Resistors because they should be in input side.
- LED off-pages Removed and the pins connected to the earth

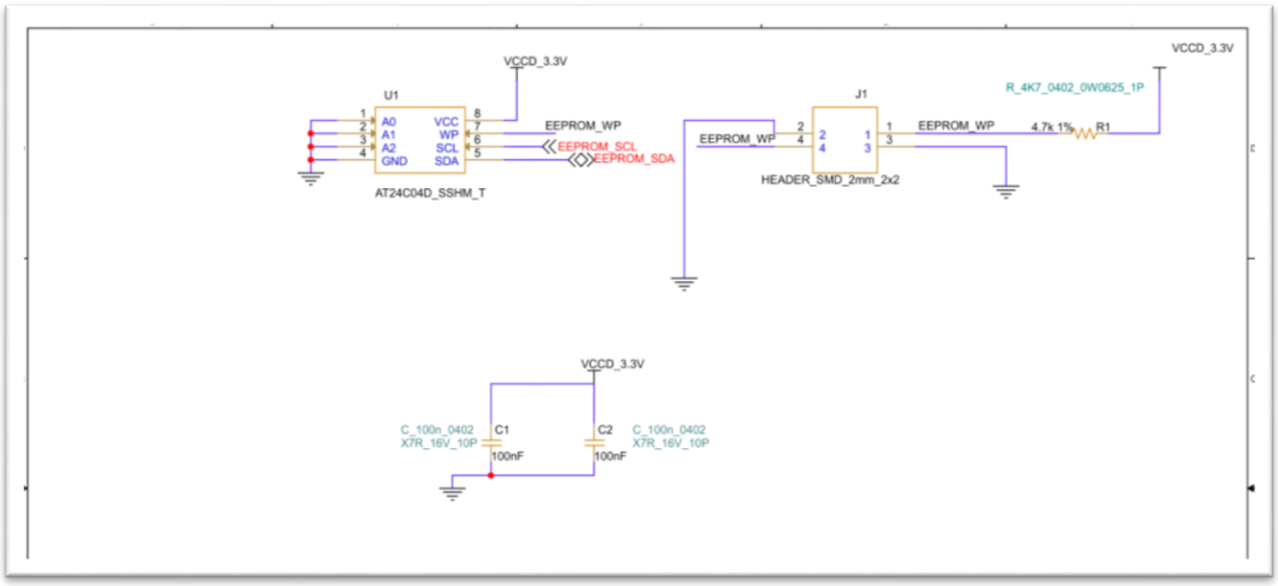
MCU:

- We connected all unused pins to GND as the datasheet told.
  For example:

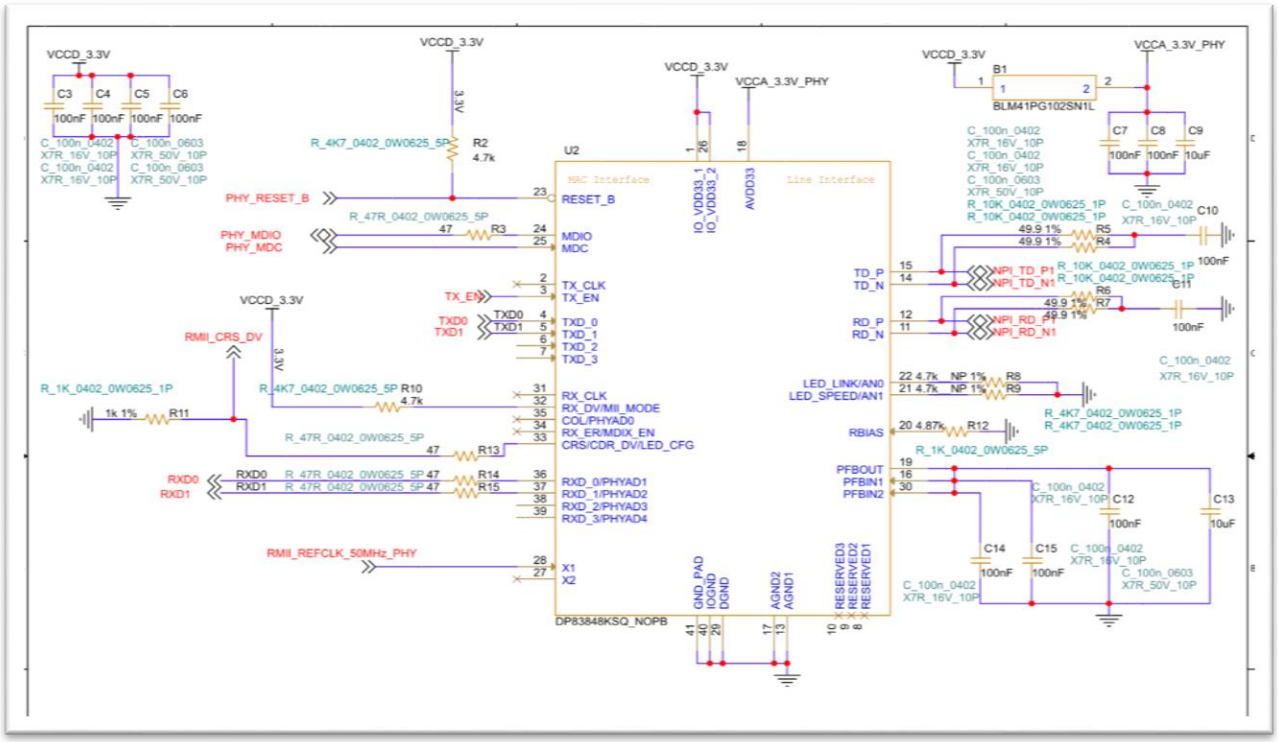


Final Design in cadence:

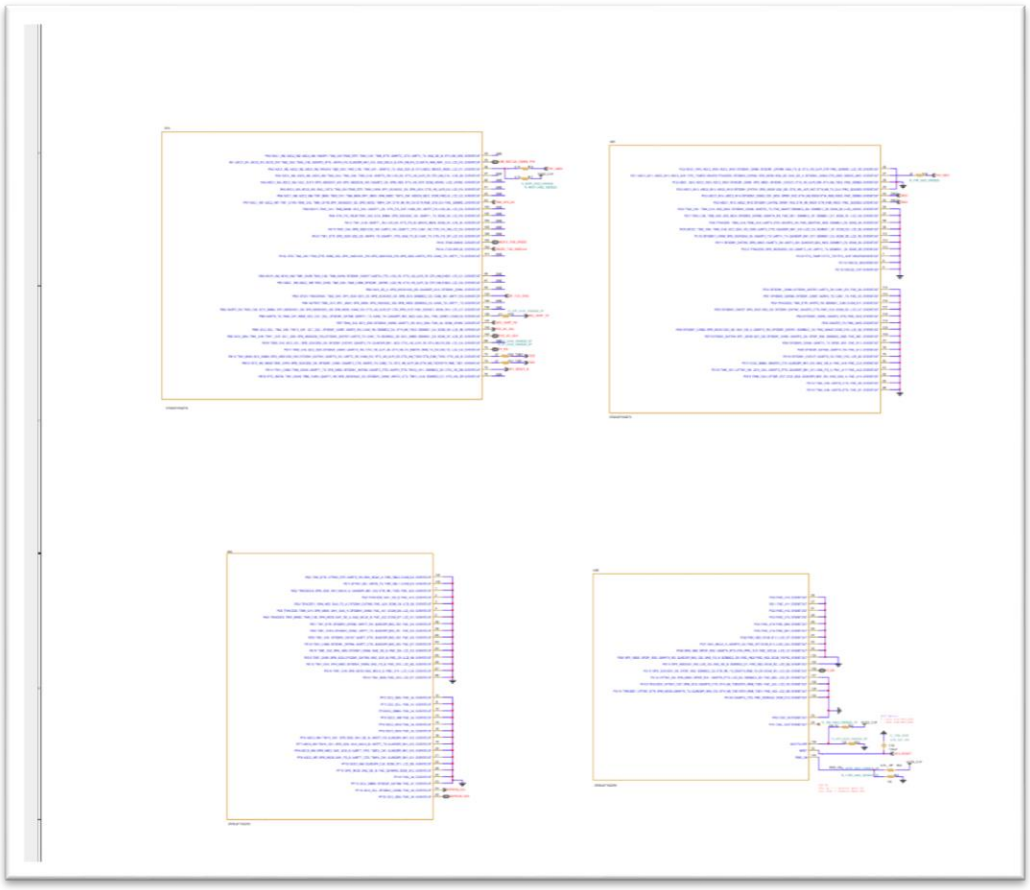
1.eeprom



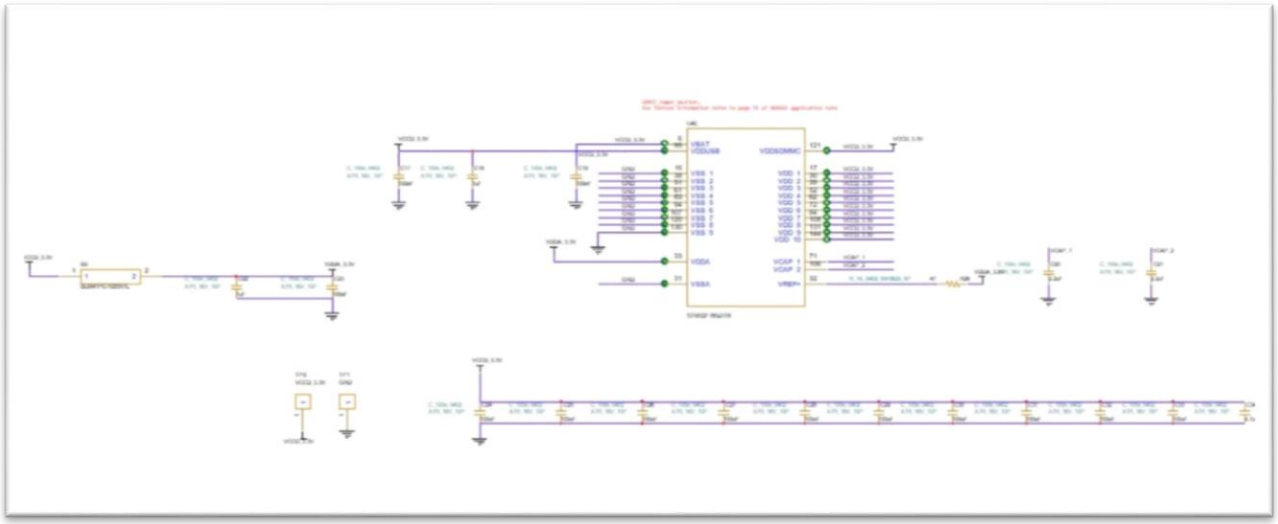
2.PHY



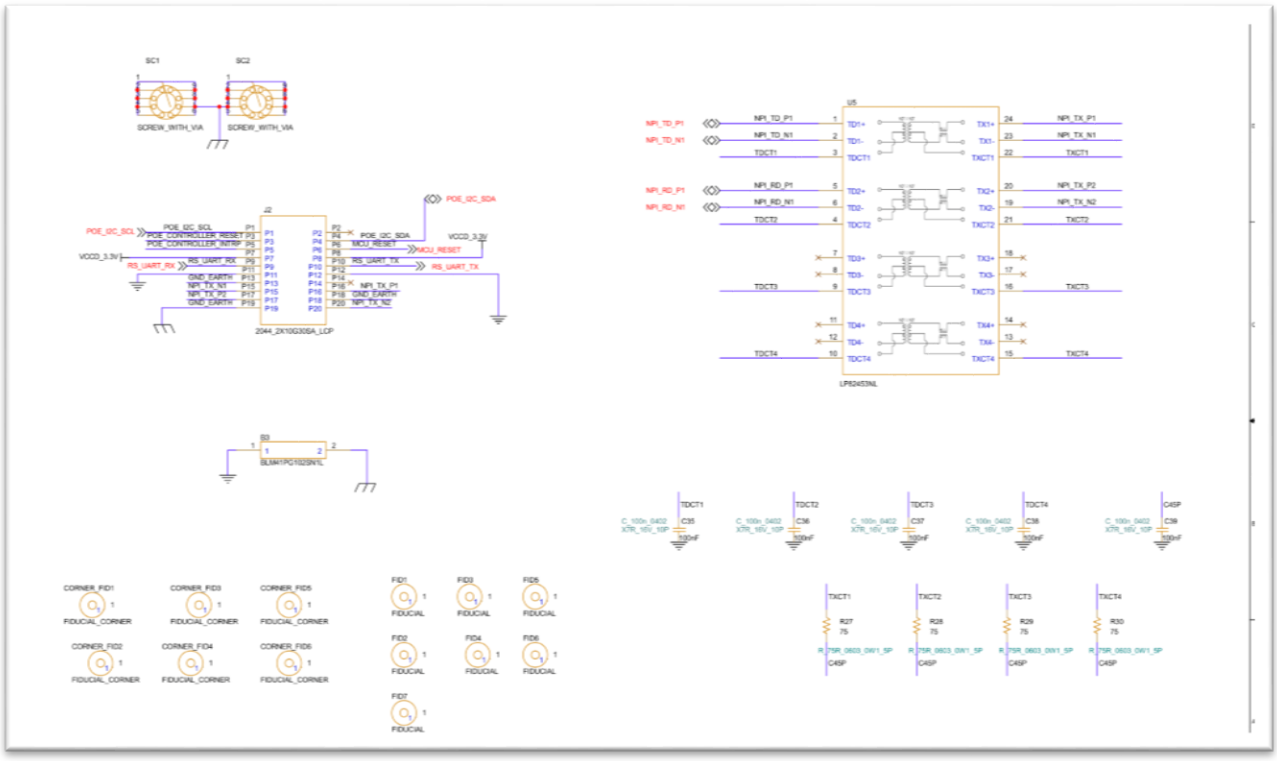
3.MCU



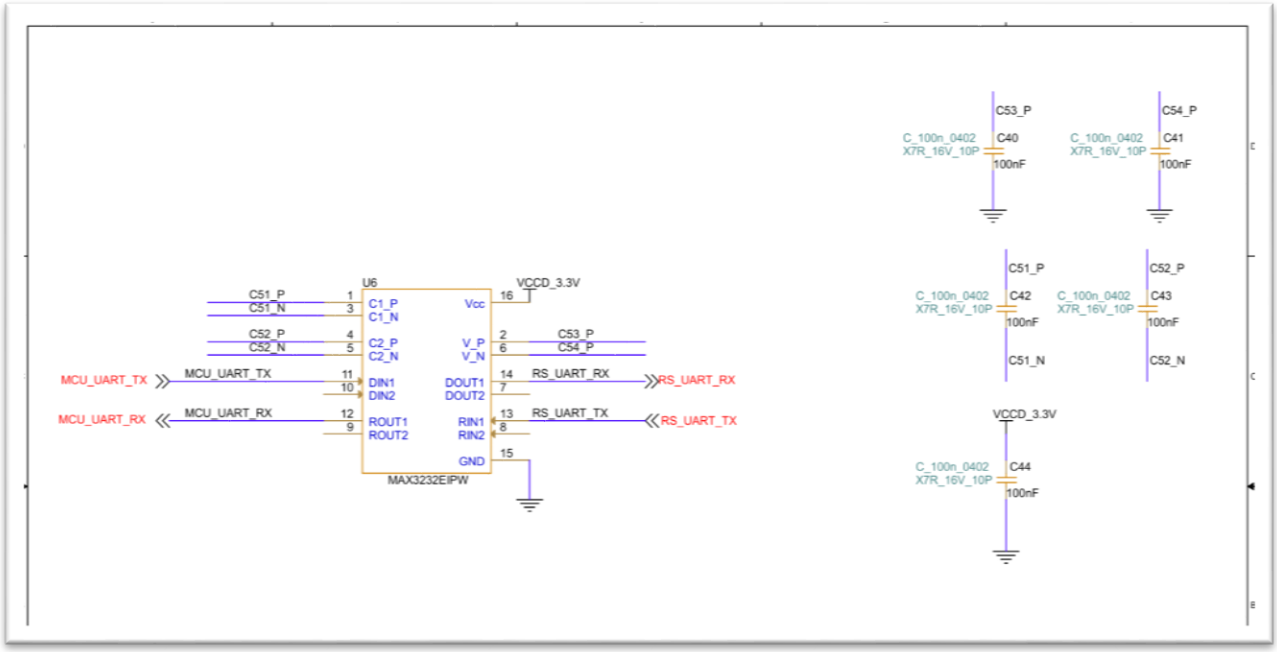
MCU Power



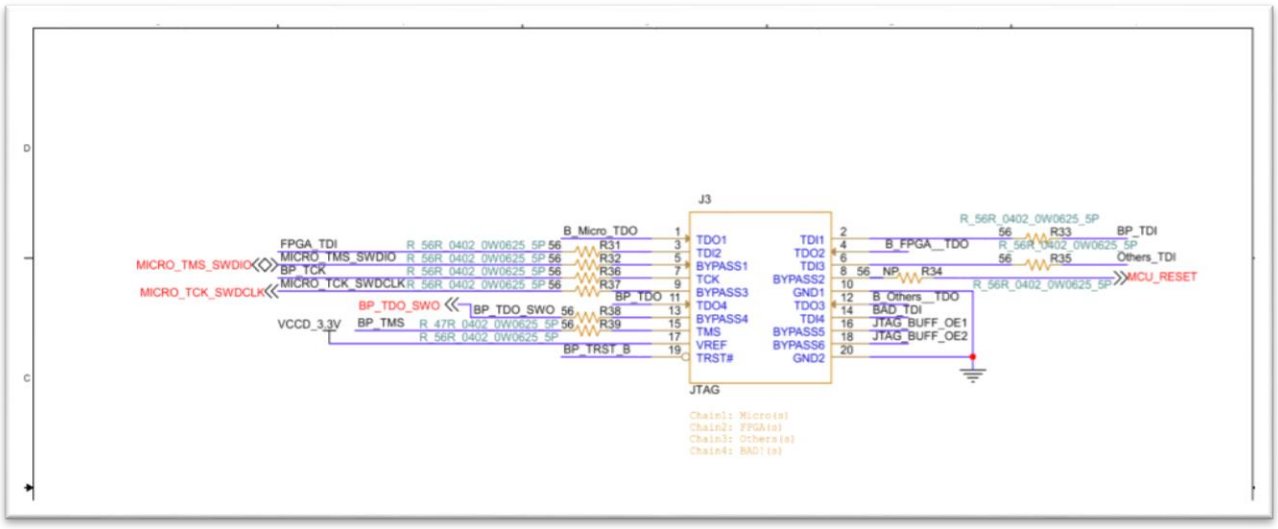
4.Port



5.RS-232 Driver/Receiver



6.JTAG

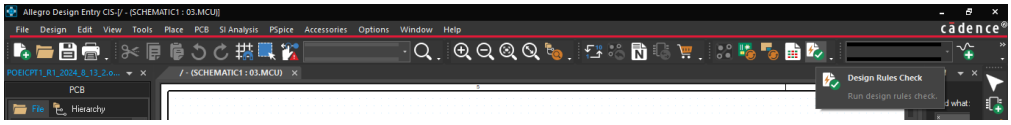

J3
B_Micro_TDO
FPGA_TDI
MICRO_TMS_SWDIO
BP_TCK
MICRO_TCK_SWDCLK
BP_TDO_SWO
VCCD_3.3V
BP_TMS
BP_TDO
BP_TRST_B
R31
R32
R36
R37
R38
R39
TDO1
TDI2
BYPASS1
TCK
BYPASS3
TDO4
BYPASS4
TMS
VREF
TRST#
TDI1
TDO2
TDI3
BYPASS2
GND1
TDO3
TDI4
BYPASS5
BYPASS6
GND2
B_FPGA_TDO
B_Others_TDO
BAD_TDI
JTAG_BUFF_OE1
JTAG_BUFF_OE2
R33
R35
R34
BP_TDI
Others_TDI
MCU_RESET
JTAG

Phase 3: PCB Design

PCB design:

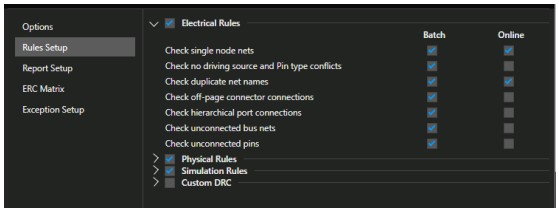
Before PCB:

Before making layout, we should do some checks in schematic part. Checking rules and connections.

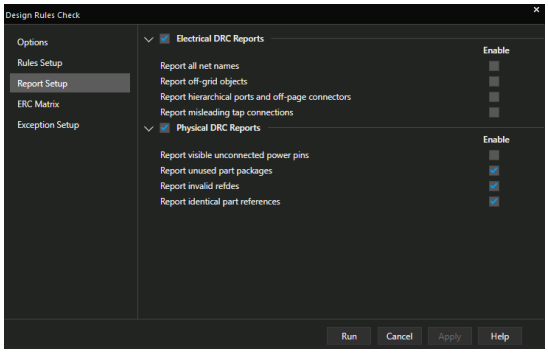


At the first we set ERC matrix. It is good for find and setup all possible the errors in our board. For example, you can set error on input/output connection or unconnected.

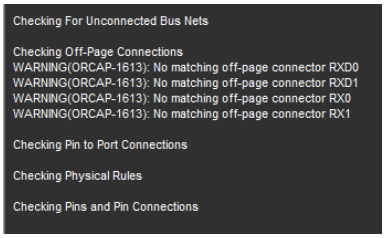
Then we can enable checking all electrical and physical rules in our design.



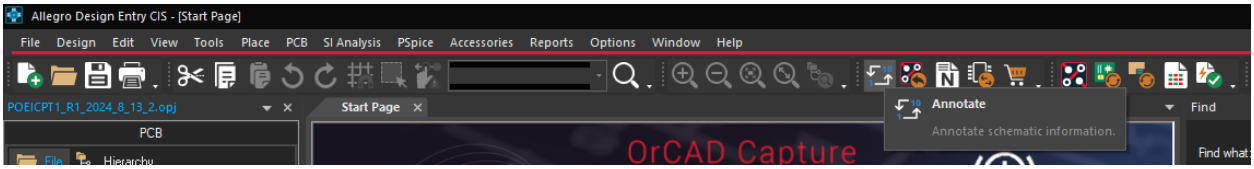
At the end we setup reporting options to show more options from our schematic.



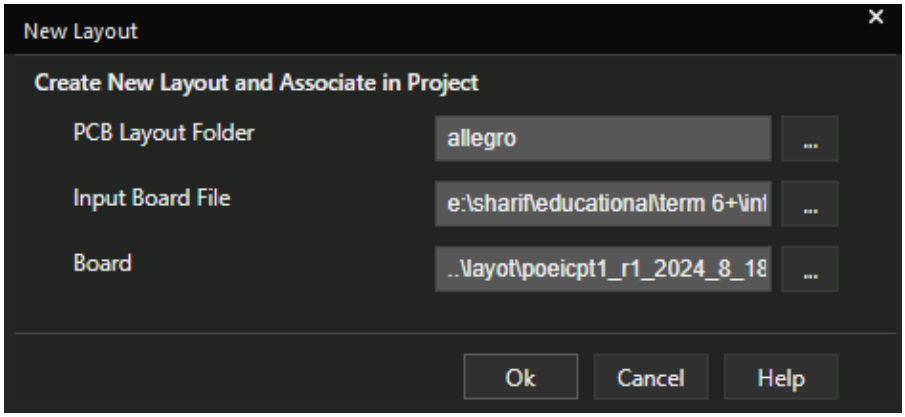
After clicking on the Run, Cadence start checking and show the results in session log.
We put two images as an example:



We need to name our elements in order in all of the board with smaller and standard numbers (For example we had R313 and it was a little weird number). For this we used annotate:



Now is time to make layout:



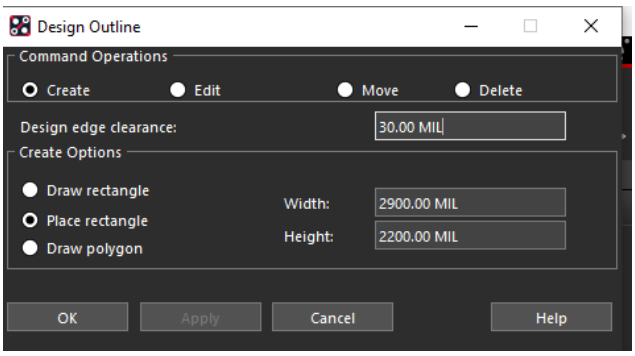
(Every time you make changes in schematic you should do like this and you should update your layout. But first time just make a file as an output)

Starting to design:

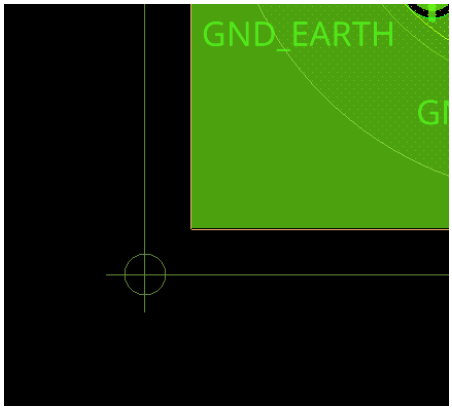
Our board has two layers and we will tell about designing in two layers (top and bottom). But we can add new layer if we want.

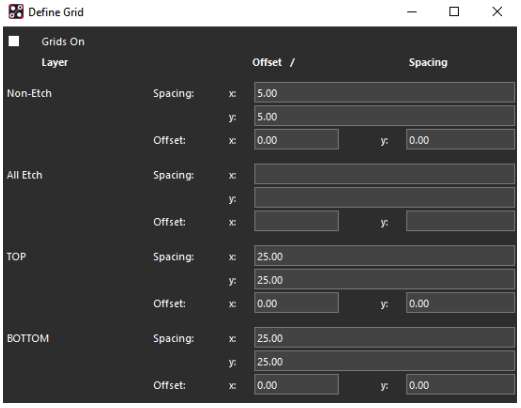
- Size and distance in cadence is based on MIL (MILLI Inches) but you can change it.
- At the first you need to draw an outline for your board

(Our board should be 7.5cm x 5.5cm and we converted it to MILS)
2900 X 2200 MILS, 30MIL Edge Clearance
(Edge Clearance is for show where you should draw your wires and place elements and your main board (electrical part) is on that so you can't design out of it.

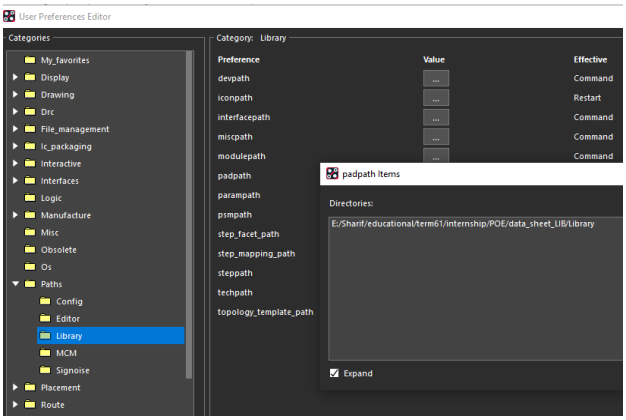


Then you set your coordinates and grid (Typically in left bottom corner of outline)
Default was 100 Mil but we changed it to 5.

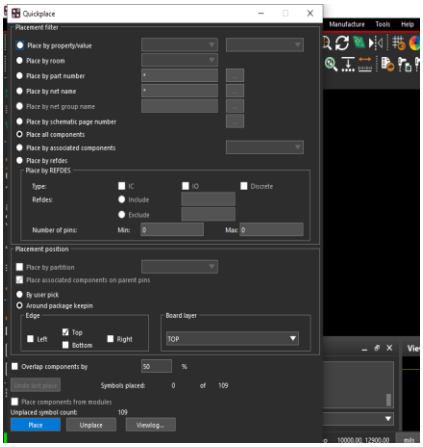




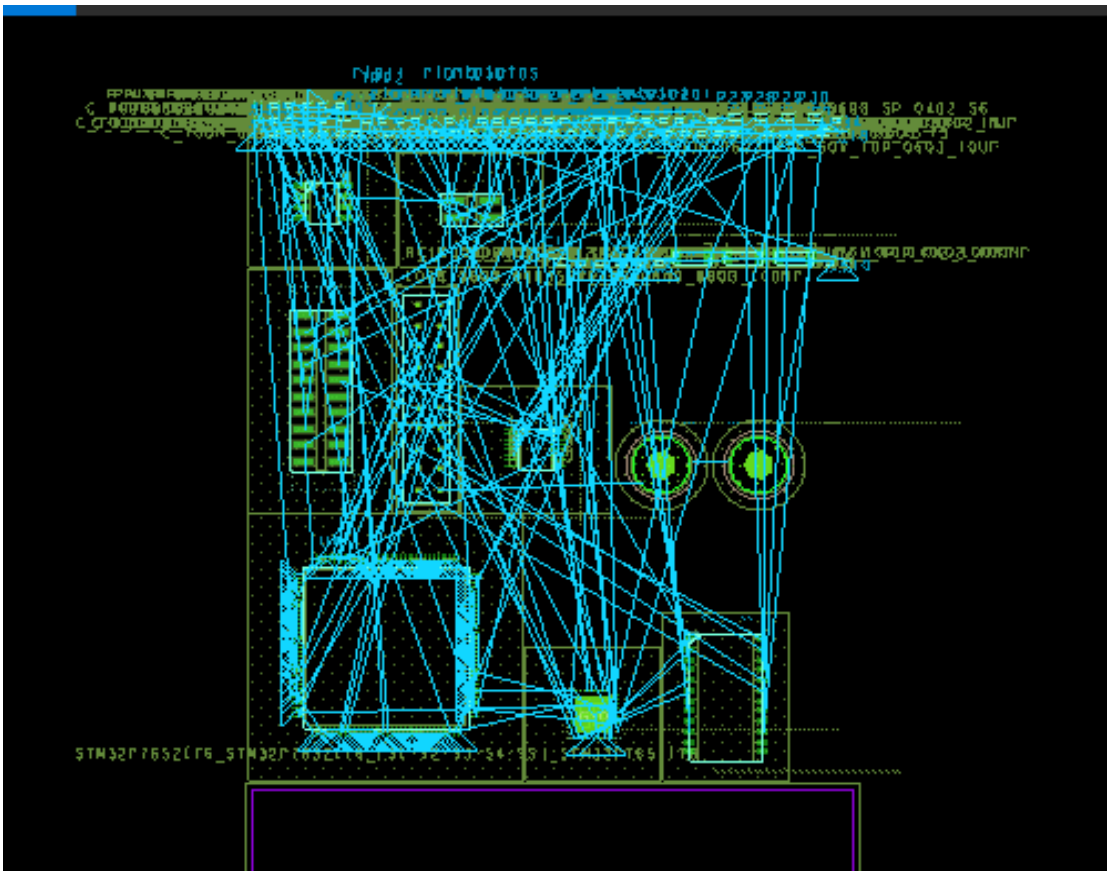
Here you need to have devices footprints and you should add them to your board file. (Be careful about naming rules and name of folders in directory (specially space). Because it may cause problems in your PCB file)



After that you place your parts in PCB space. Cadence itself can count our symbols and also it can say how many of them got problem in recognition for show in PCB space. (For Example, we got problem with our fiducials



At the end you place parts like this

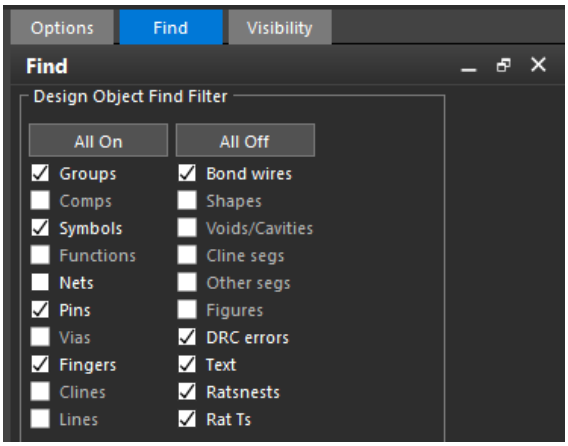


Designing features:

There are important and useful menus on the right side. At the first we are going to talk about **Menus.**

Menus:

Find menu:



This menu is for showing and clicking in cadence. You can show or filter any part you want.

Some of options are very useful and you should enable them for many usages like moving and deleting:

Symbols:

Moving/deleting any resistor, capacitor, IC, …

Shapes:

Moving/deleting outline and a part of an IC

Pins:

Measuring distance

Drawing wire, vias, …

Vias:

Moving/deleting vias

Clines:

Moving/deleting wire between two pins completely.

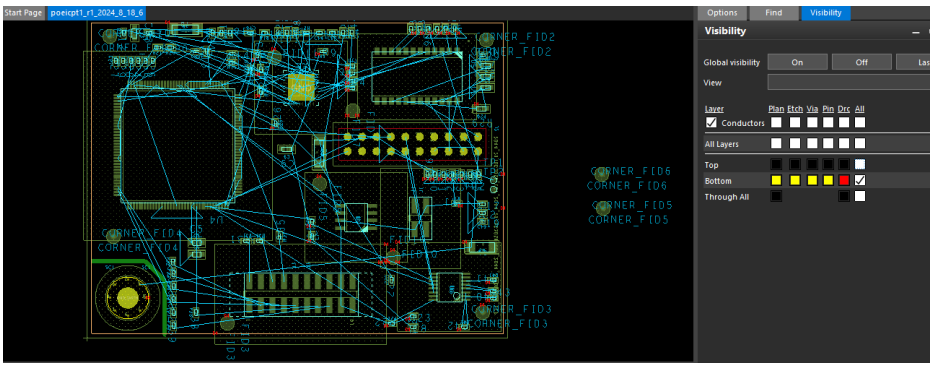
Cline segs:

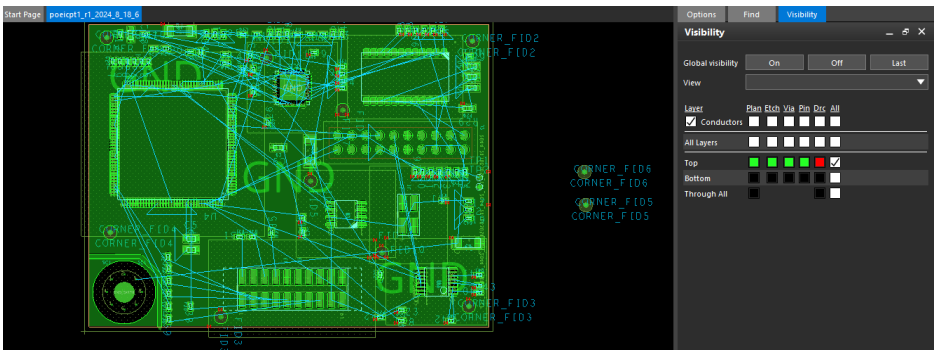
Moving/deleting wire from part by part from its angular point.

(Note: If you want to change the wire way without cutting connection of two pins turn on Cline segs in etch mode.)

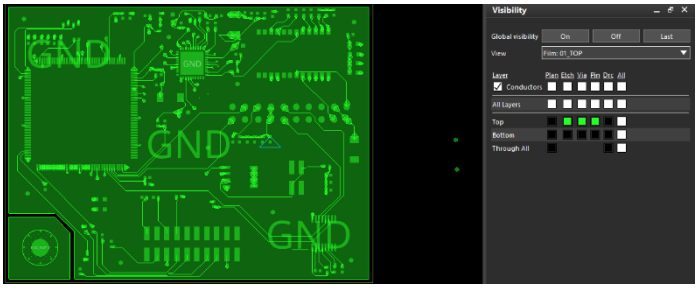
Visibility Menu:

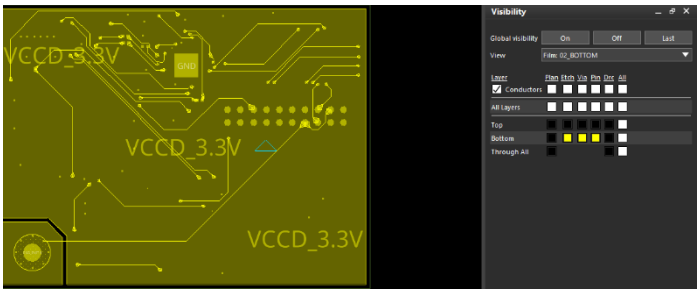
This for setup which parts you want to see and which you need to get hidden. It has options like find menu but as you know our board has top and bottom layer, so you can also setup to see just one of them and it can help you to design them separately like this:

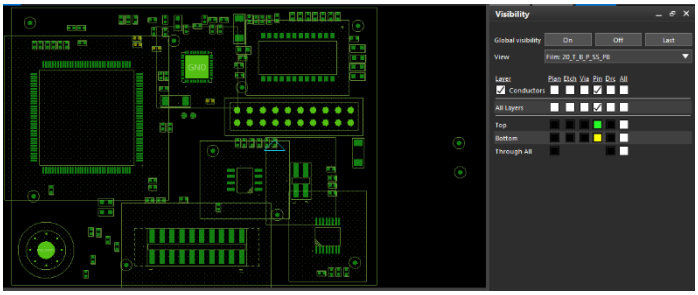


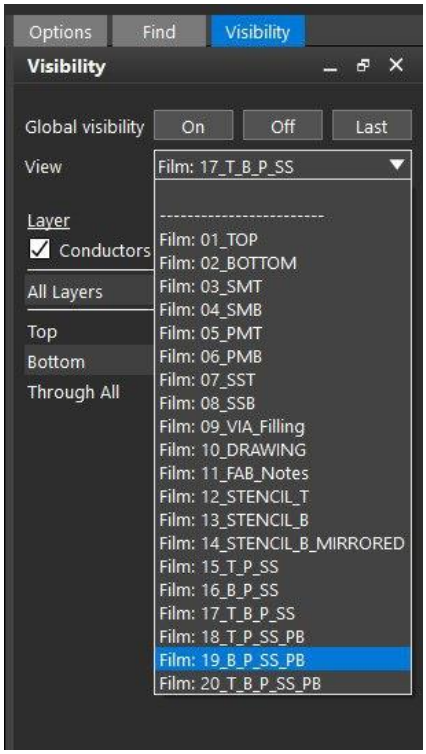


There is another tool in visibility menu. View part for add sampled templates(films). We added our company's samples which are more useful in designing. It has top and bottom showing as you see and also we can have placements without rats with T_B_P_SS.





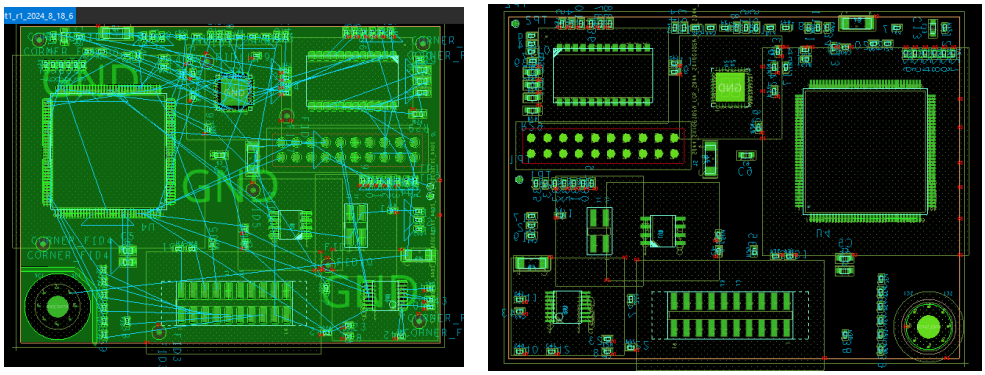




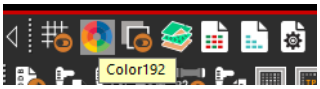
General features:

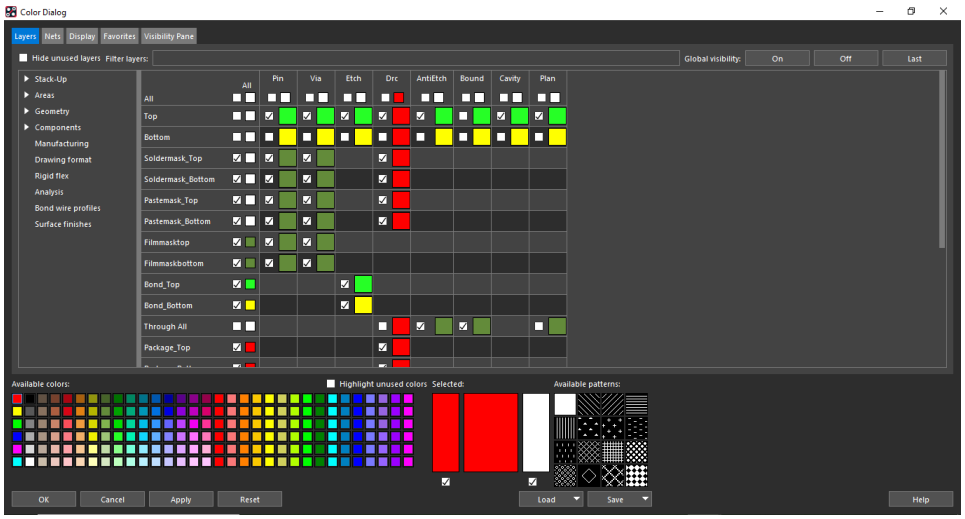
Hiding rats:

Also use zoom fit to fit your outline to your design space

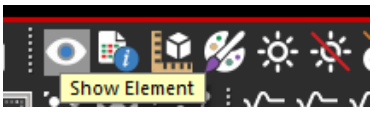


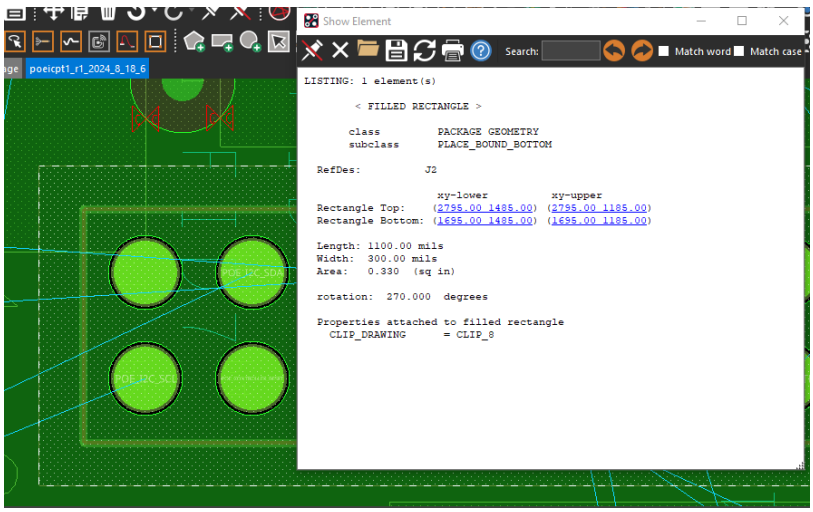
Coloring parts of your design for make it more understandable:



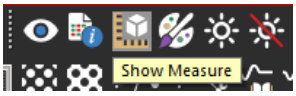


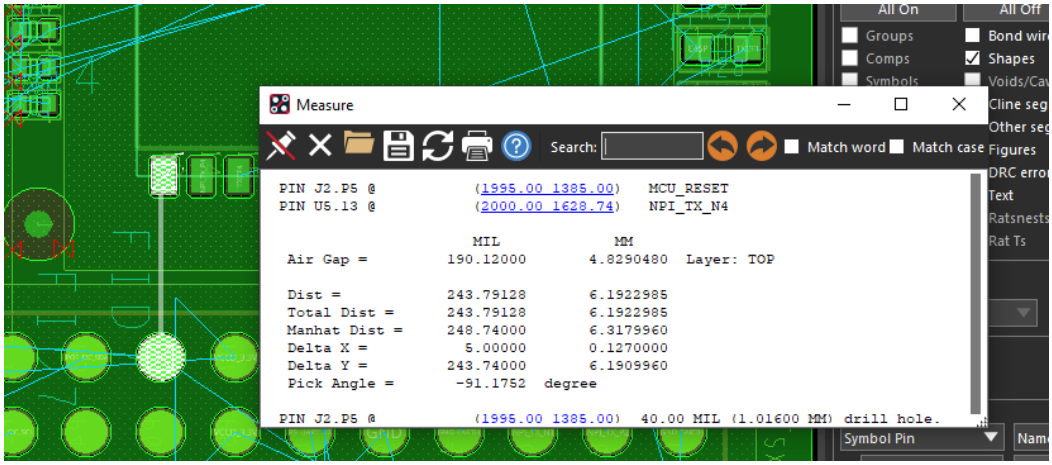
Know about elements:





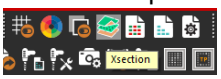
Measure distance between to selected elements:

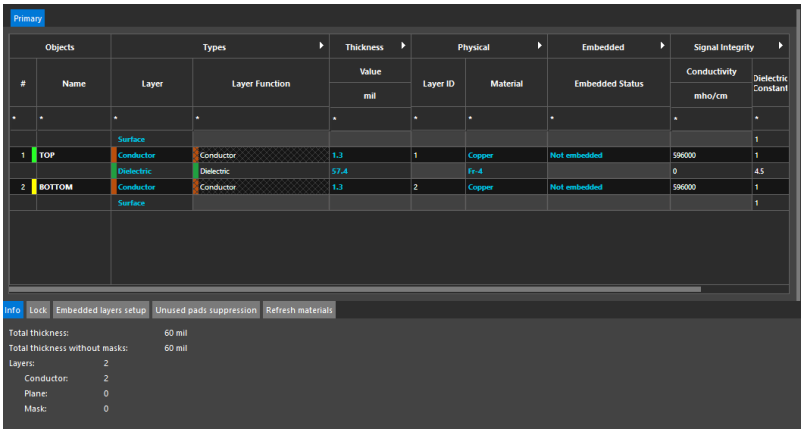




Xsection:

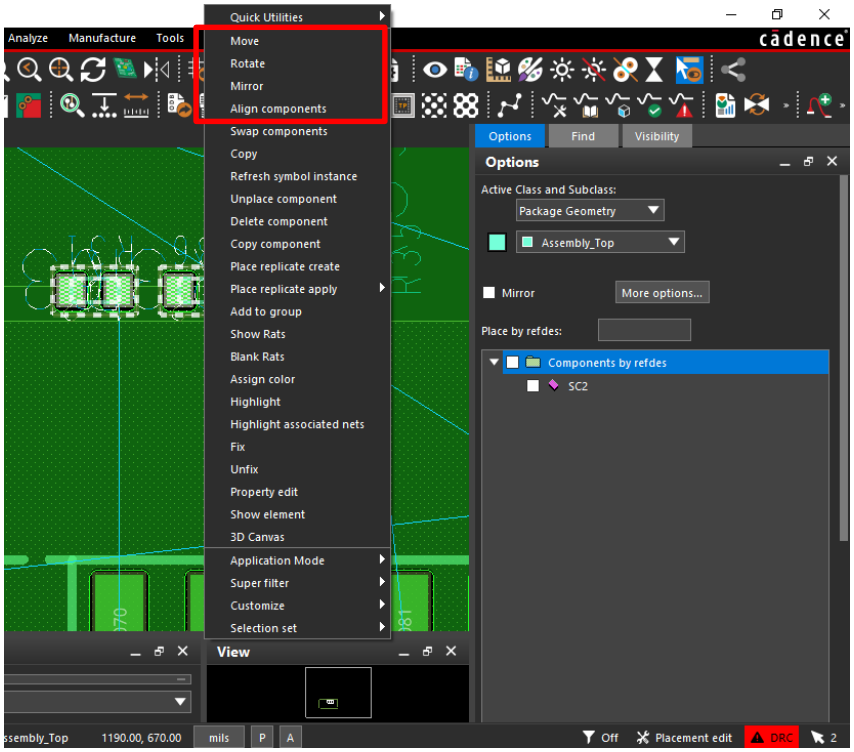
For setting thickness of our board and its layers. Our board has two layers which needs one dielectric layer and two conductor layer. As standard, we put 1.3 mil for conductor and 57.4 mil for dielectric; because our total board can be 60 mil depends on PCB producer company.





Placement Mode:

We use this mode for place element. It has many features we are going to tell some important ones. (Find menu is very useful for this part because you can click exactly on the element you want)

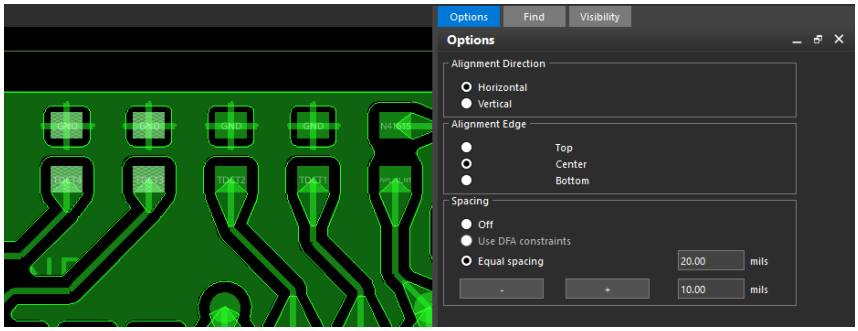


Move: changing place

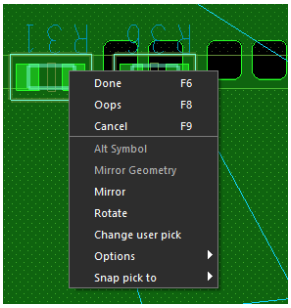
Rotate: rotating elements 90 degrees.

Mirror: rotating elements 90 degrees.

Align components: With selecting some elements and choosing this feature, you can setup their distance and it helps you to tidy your elements.



There is general thing for cadence. After doing anything for changing mode, you should confirm it with “done”. If you want, you can redo just your last action with “oops” and with cancel you completely ignored you last action since last time you pressed done.

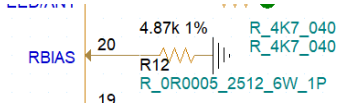


Debugging schematic with first place:

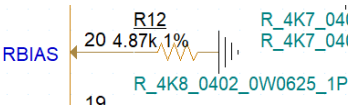
During we were trying to place our elements, we found some problems which was related to our schematic. So we turned back to schematic file.

Ethernet:

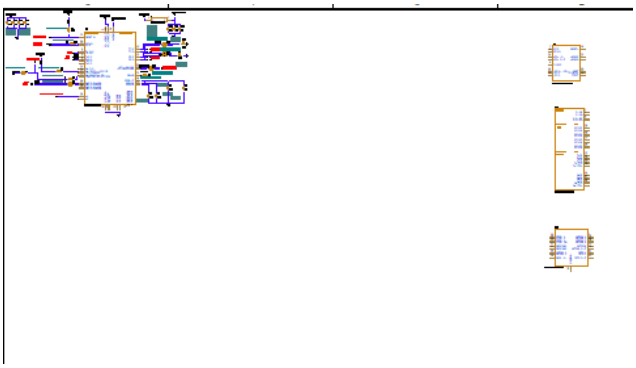
- **Resistor type was not right and it was very big for our board and power**
  Wrong:



  Correct:

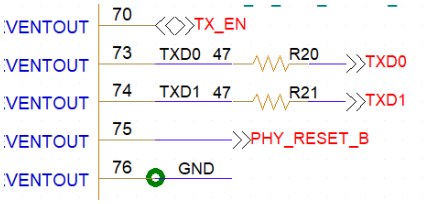


- **We had extra part in Ethernet page that was planted but is hidden because of the page size**

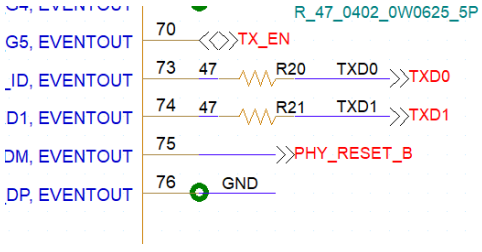


MCU:

- **Naming of TXD was wrong (both side of resistor is same voltage)**
  Wrong:

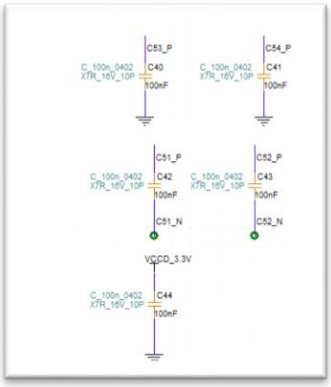


  Correct:



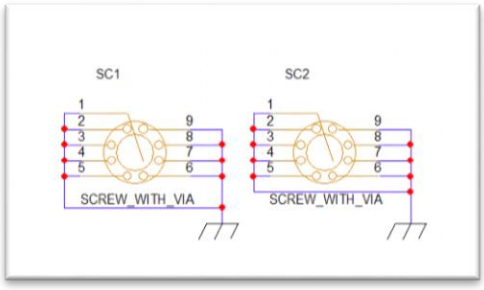
UART:

- **Name of the nets was wrong (C51_N , C52_N)**



Ports:

- **We added fiducials and screw to this page**

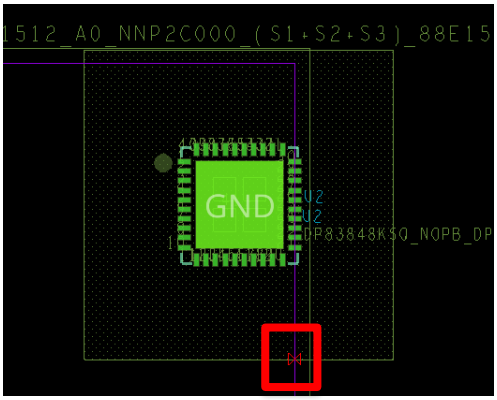


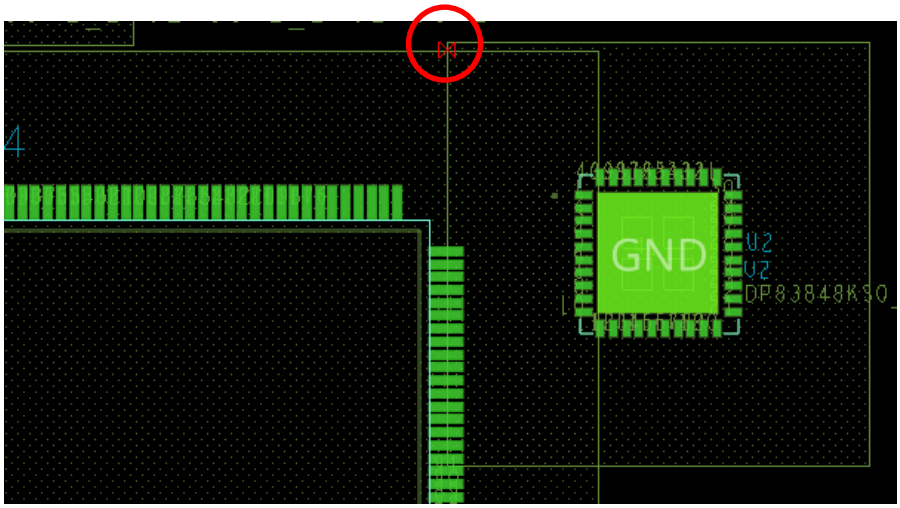
Standards of placement:

- Distance:

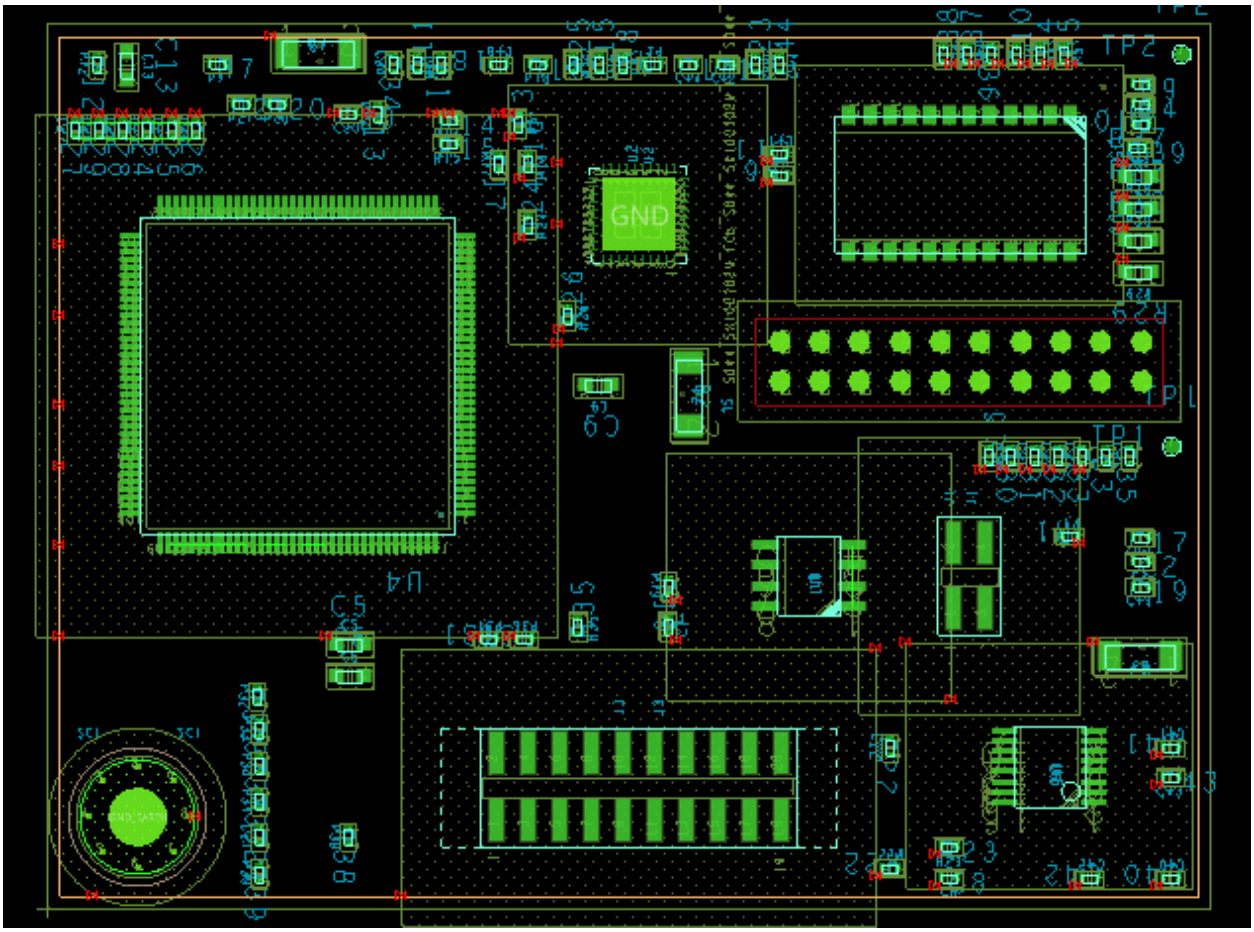
- **Between resistors:** 20MILS
- **Between resistors and IC:** 100 MILS
- **Between two SMD IC:** 150 MILS
- **Between IC and Through whole:** 200 MILS
- **Corner fiducials distance from outline corners:** 3 mm

And images below are for some errors of placing.



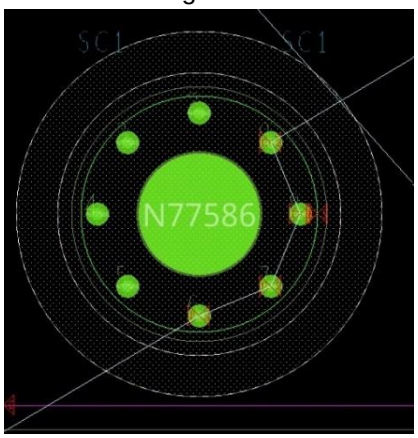


One of our placements:



Screw route keep out:

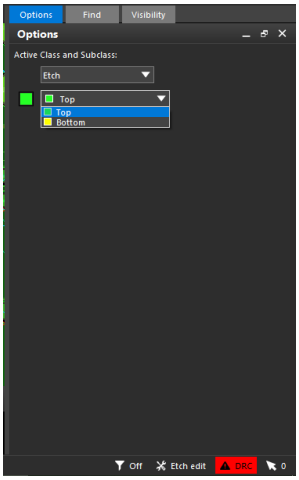
To set the screw and its surrounding area as GND_EARTH, two shapes need to be removed: the route keep-out shapes. These shapes prevent connecting wires to the screw.

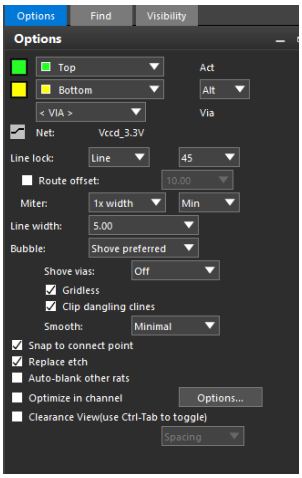


Etch mode

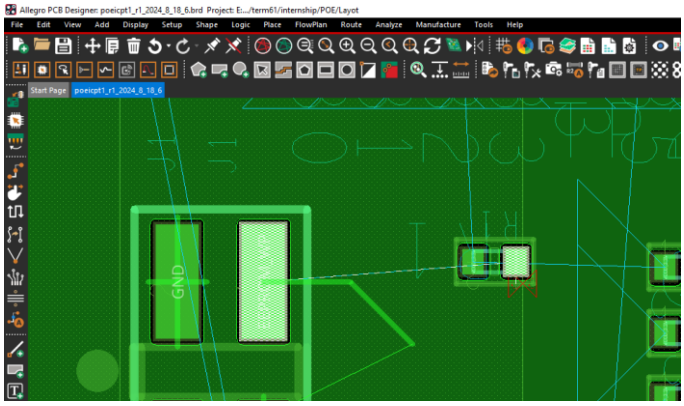
Drawing Wire:

After placing all elements now it is time to connect them. Etch mode is for this state. You can choose the layer where you want to draw wire. You should also set your wire width, because it should handle your currents and not get burnt and other physical and electrical conditions (More details in etch mode standards) It also depends on PCB company standards. Previous researches showed that is better to give **45** degrees angle to your wires if you want to turn and change the direction.





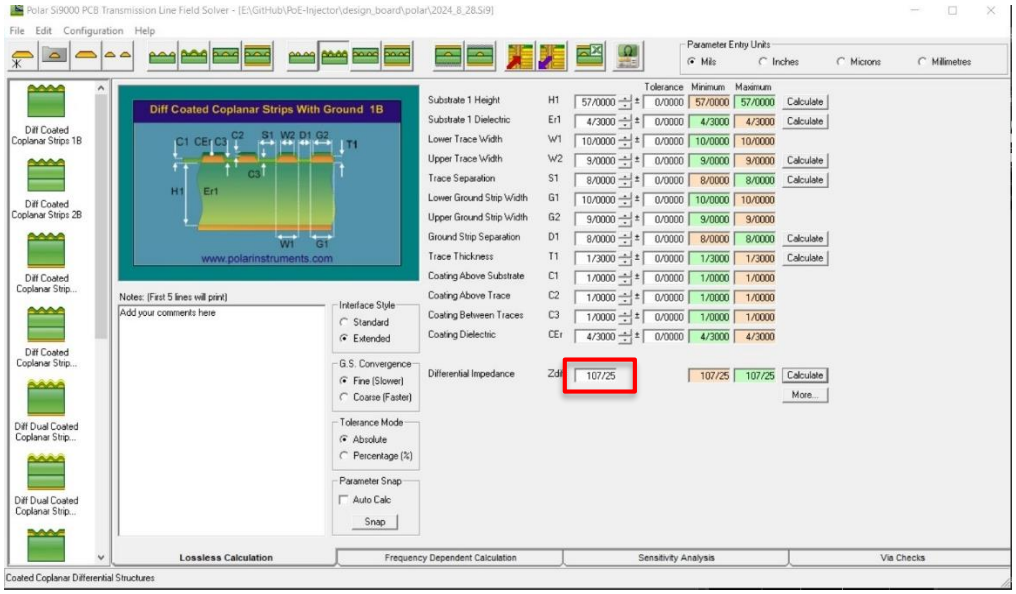
After setting etch mode options, you should enable “pins” from find menu and you wire will be like this:



Etch Part Standards:

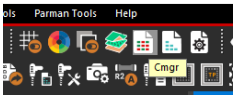
Polar software:

As we said we need to find width for our wires for this we use Polar software to calculate widths and impedance of wires.

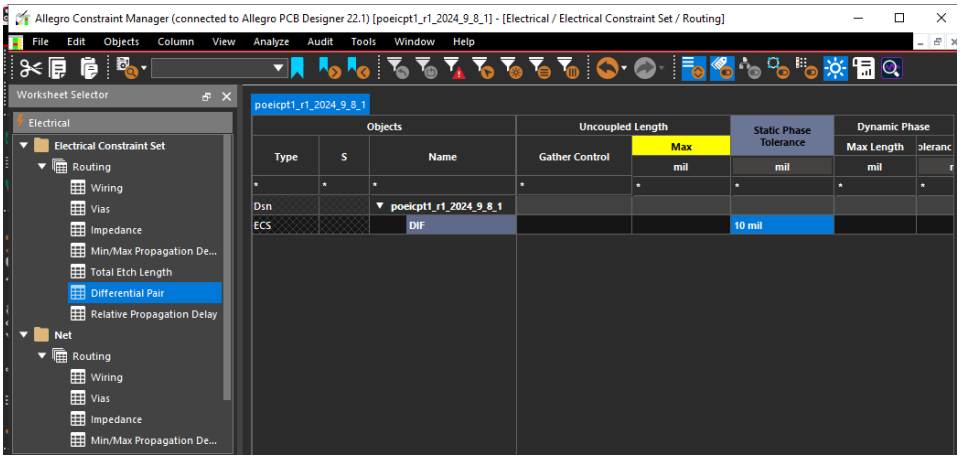


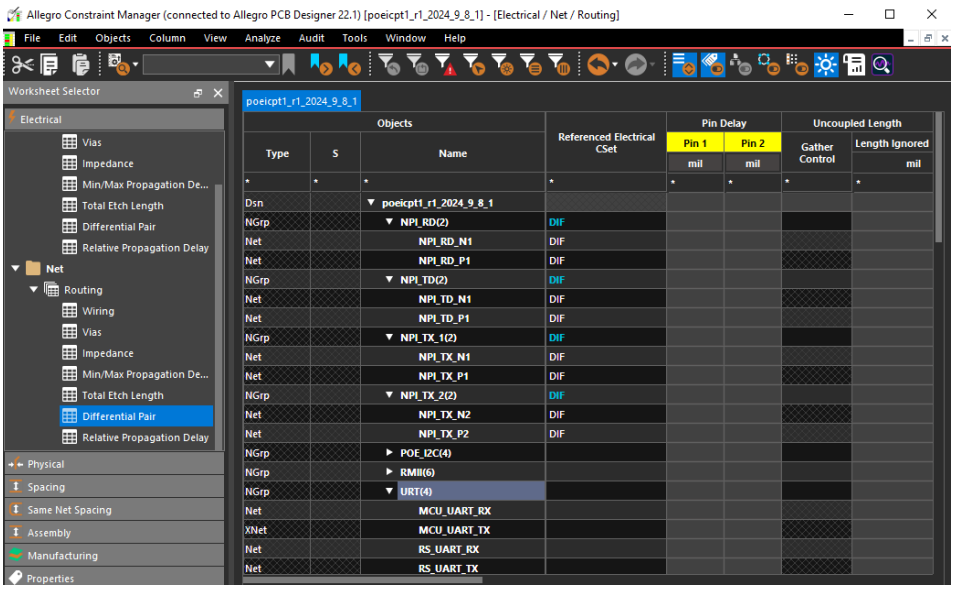
Constraint Management:

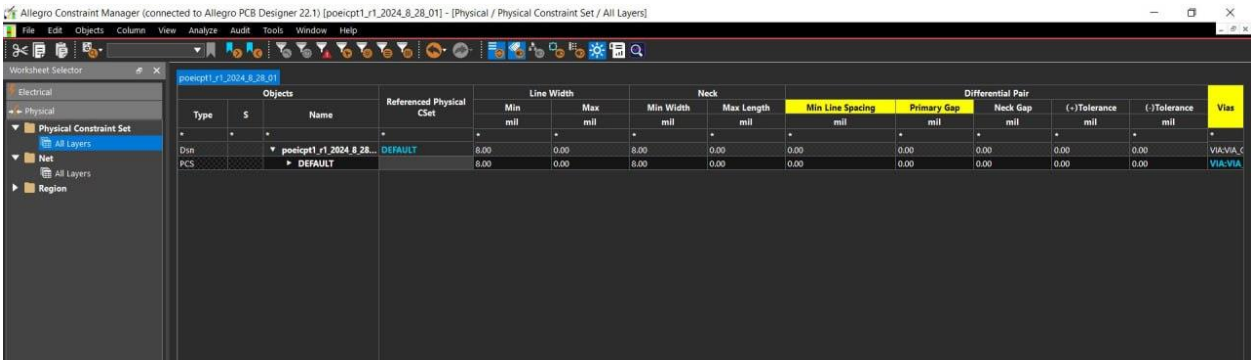
Then we set constraint to prevent errors and it makes our design more accurate.



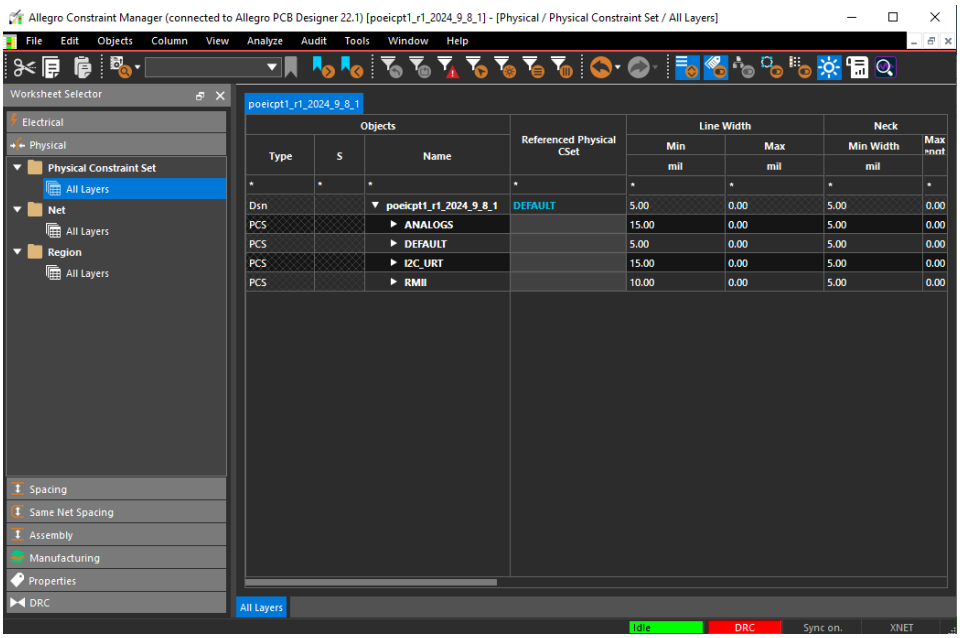
Electrical part:

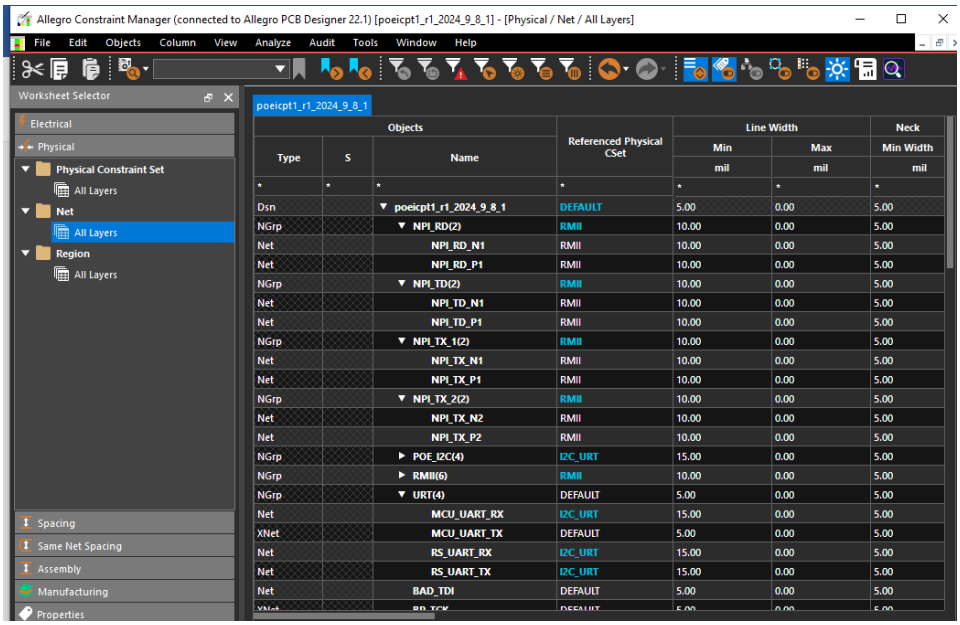




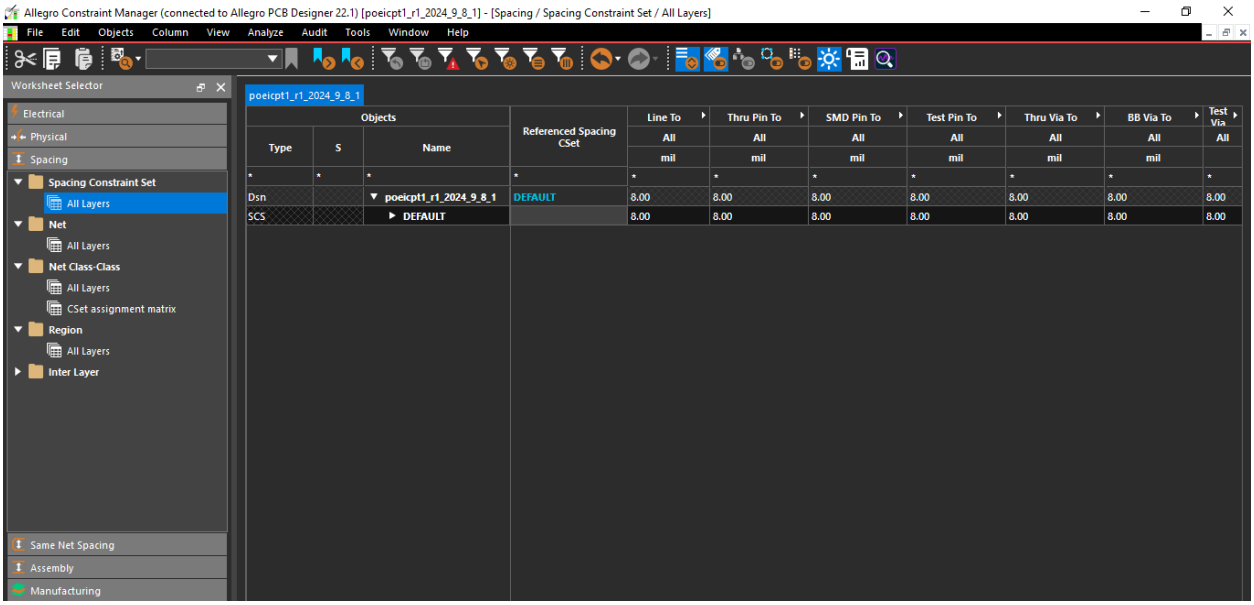


Physical part:





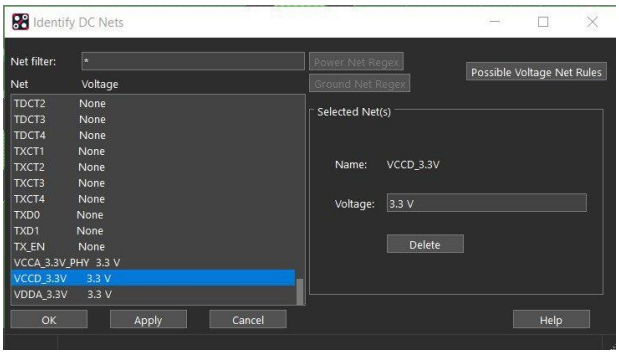
Spacing part:

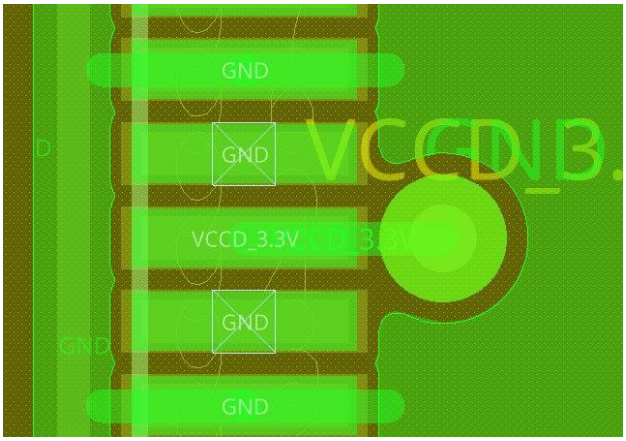


In this part we set some constraints about the distance from one object to another. it can be the distance between two wires or the distance between pin and wire or other options. It prevents overlapping and unwanted connections in manufacturing process.

Checking option:

You can find all the nets with same name with (Logic -> Identify DC nets) and is useful for any kind of checking and debugging.

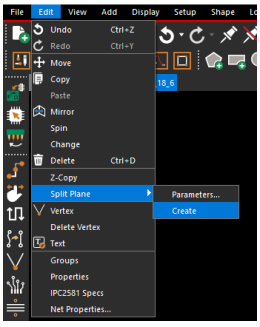


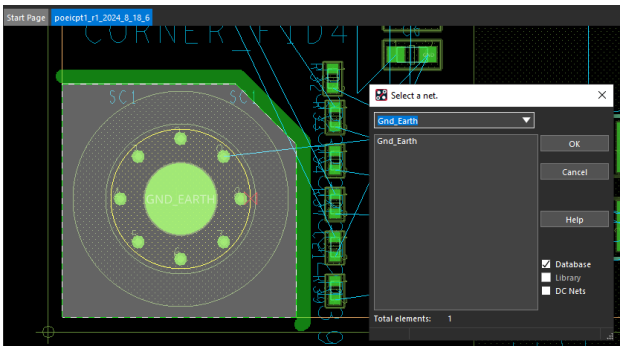


Plane for GND, GND_EARTH, VCC:

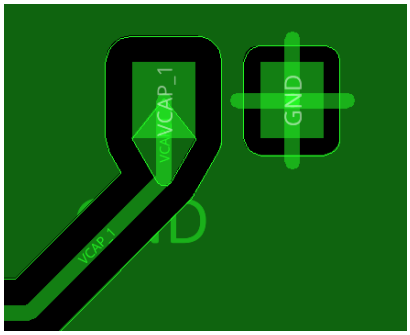
We do this because it not good to use wires for power and also it makes our PCB less complicated.
At the first we need to device your planes, so we need to draw anti-etch in etch mode. (For new plane it is necessary)

Then you can name your plane to any voltage of nets you want. We decided to put our GND_EARTH on screw as image below. Other part of top layer is GND plane and bottom layer is completely VCC plane

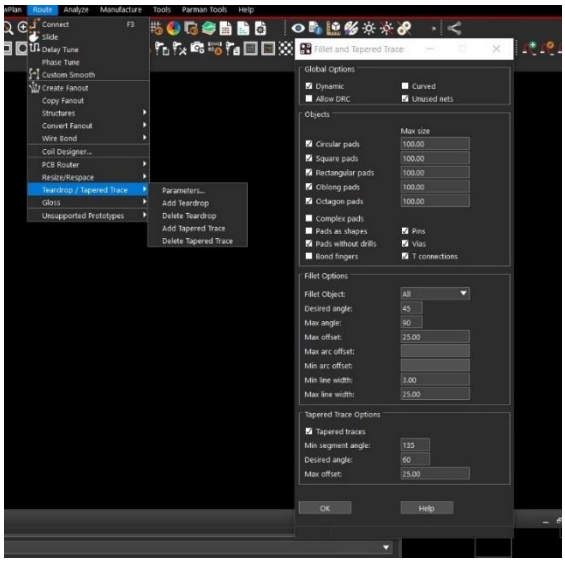




After setting planes, your wires get anti-etch around them and GND parts get connected to your plane as shown below:



Teardrop shaped:



The Teardrop feature in the Route menu of PCB design software like Cadence Allegro is used to enhance the reliability of the PCB by adding a teardrop-shaped transition between a trace and a pad or via. The purpose of this feature includes:

Key Purposes of Teardrops:

1. **Mechanical Strength:** Teardrops reduce stress concentrations at the junction of traces and pads/vias, making the PCB more resistant to mechanical stress, especially during drilling or thermal cycling.
2. **Improved Manufacturability:** During fabrication, small misalignments in drilling can damage the trace-to-pad connection. Teardrops provide a smoother transition, preventing broken connections caused by misaligned holes.
3. **Electrical Integrity:** In high-frequency circuits, teardrops help avoid abrupt changes in trace width, improving signal flow and reducing impedance discontinuities.
4. **Aesthetic and Flow:** Teardrops give the layout a more polished appearance and better routing flow by gradually merging traces with pads and vias.

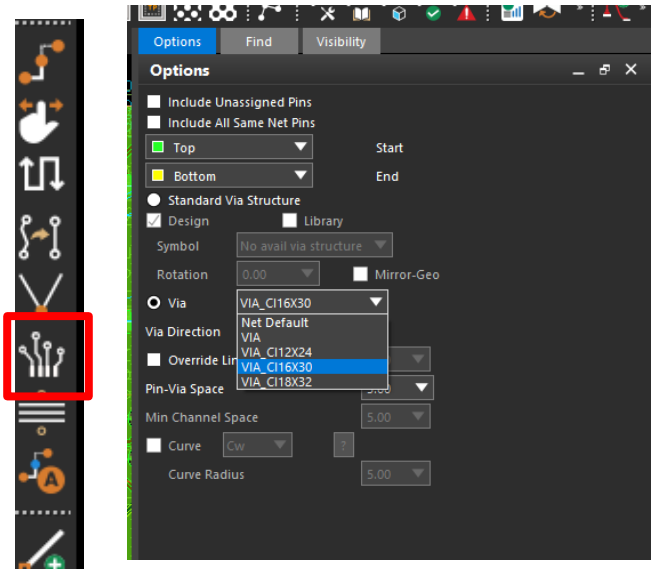
VIAs:

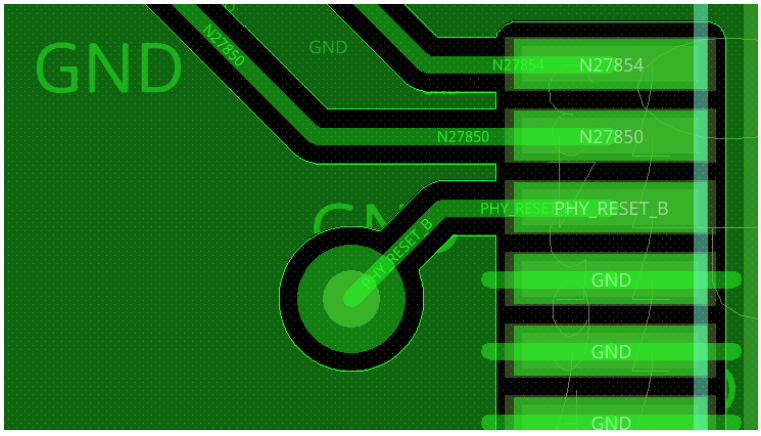
It is a hole in the board to have connection between pins in to different layers.

In our board we used it for:

- **VCC (Because our VCC plane is on the bottom layer) bottom resistors.**

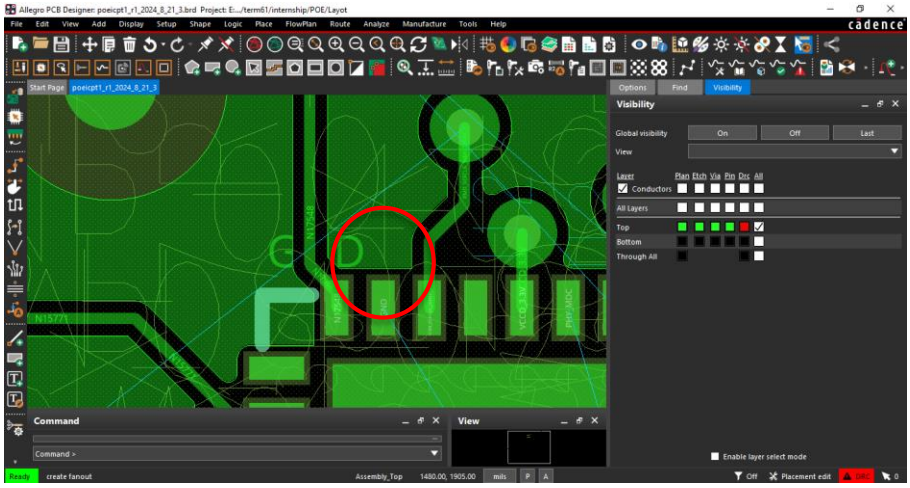
- **For pin which need to be connect from bottom layer**
- **Bottom resistors**





Special problem we got:

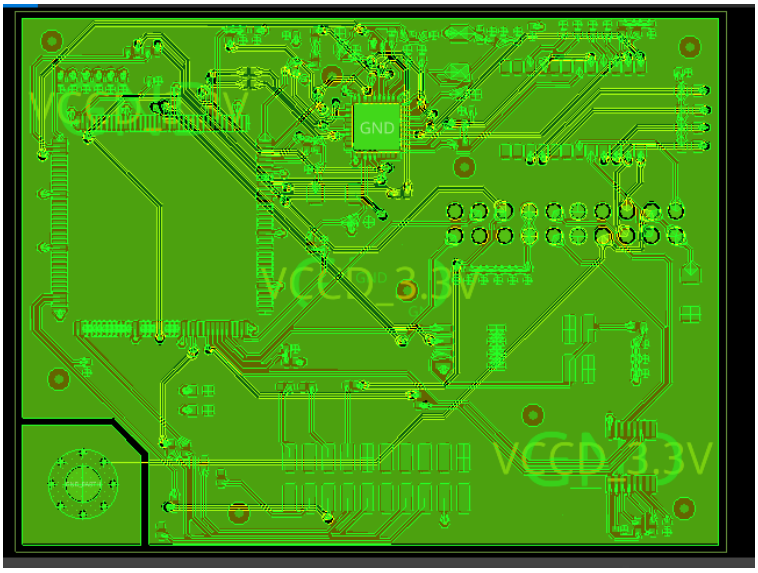
Some GND pins isn't connected to plane because of anti-etch of other pins wires.

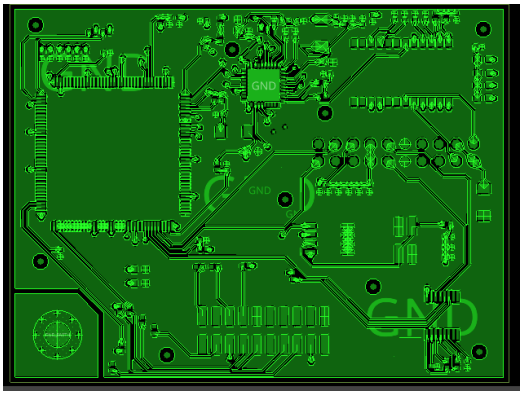


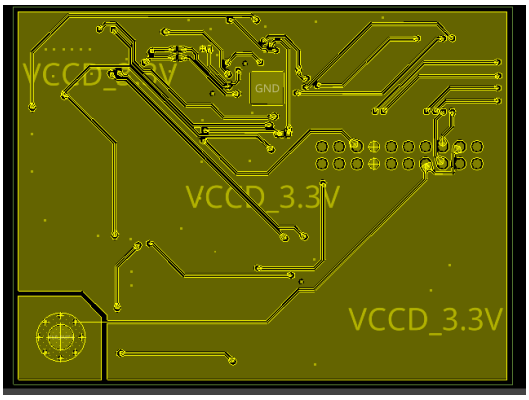
First Designing in cadence:

Our group members designed a different boards

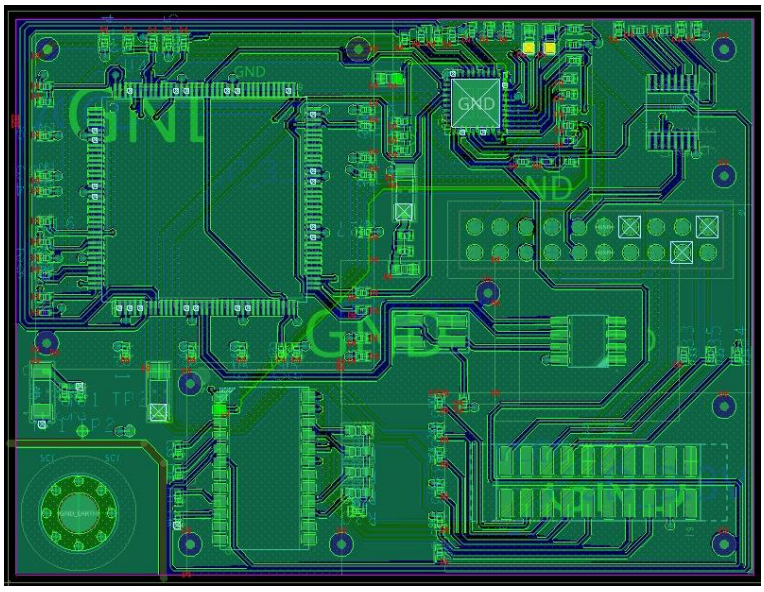
Afshari's board:



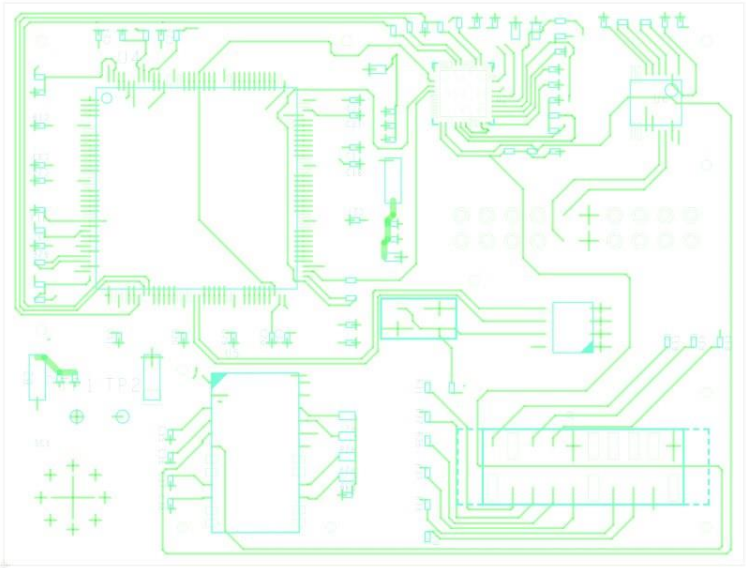


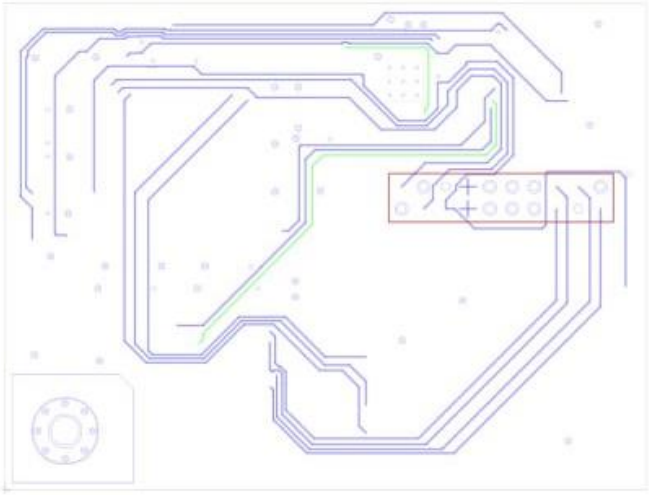


Savlani's board:

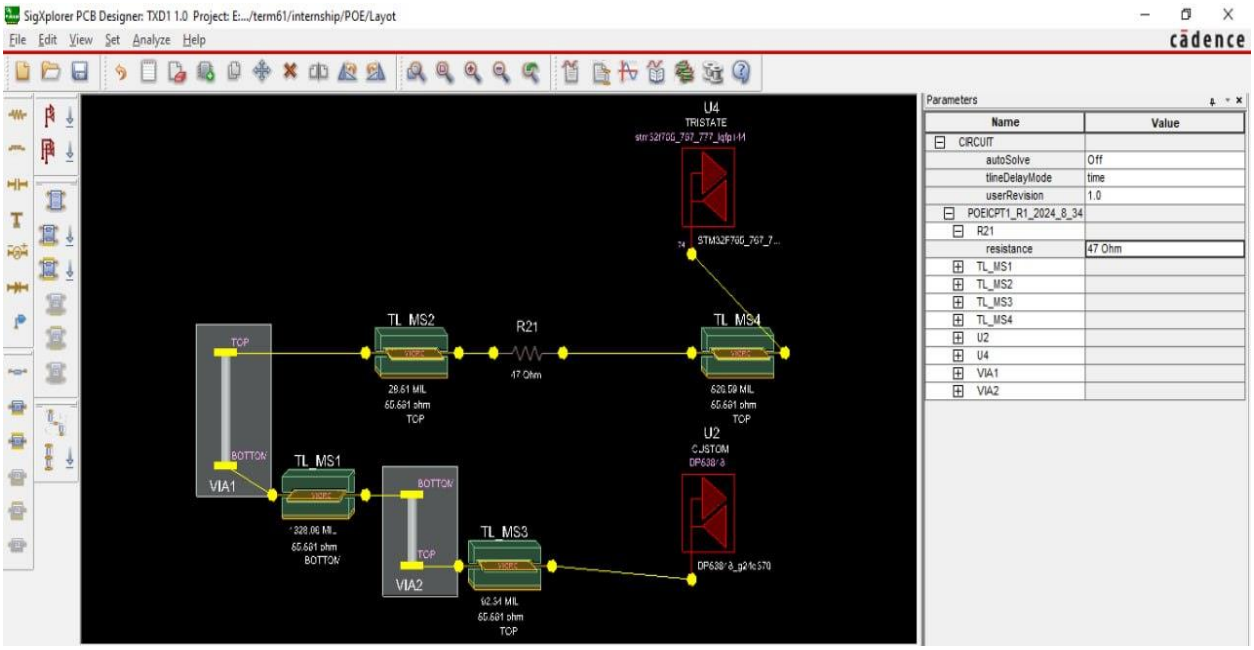


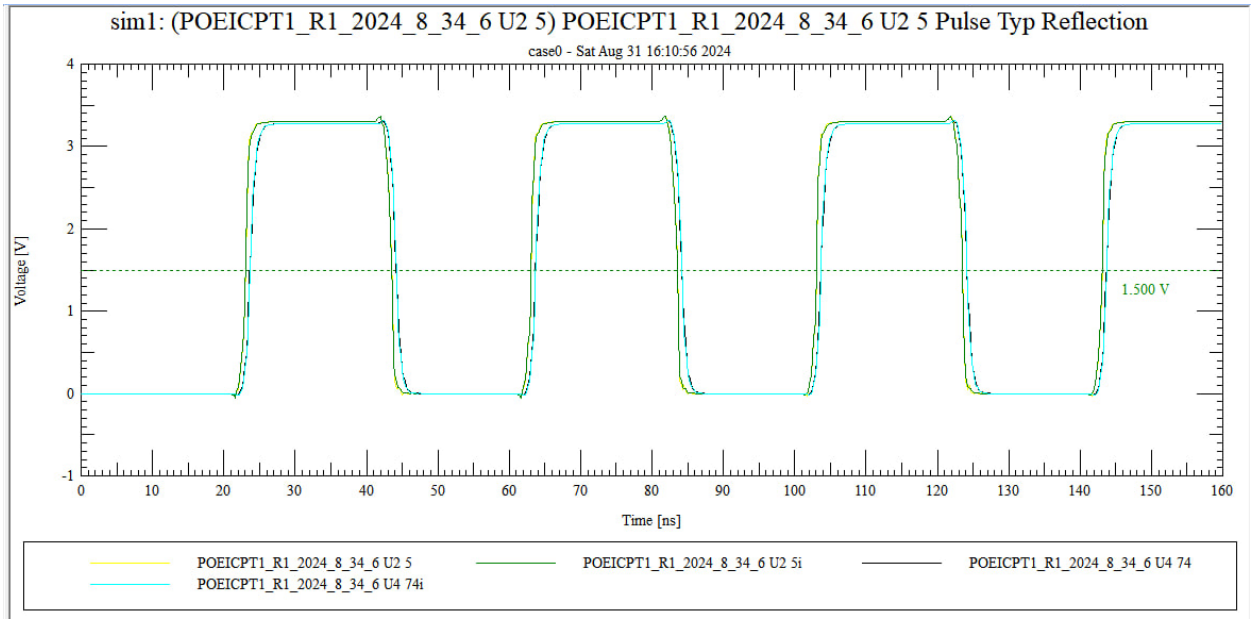
Miss. Savlani board has less errors and used 43 vias but Mr. Afshari has 94 vias.
Also in simulation we got better result from Savlani's board. So we get PDF export from Savlani's board and its top and bottom is like this:





Simulation:

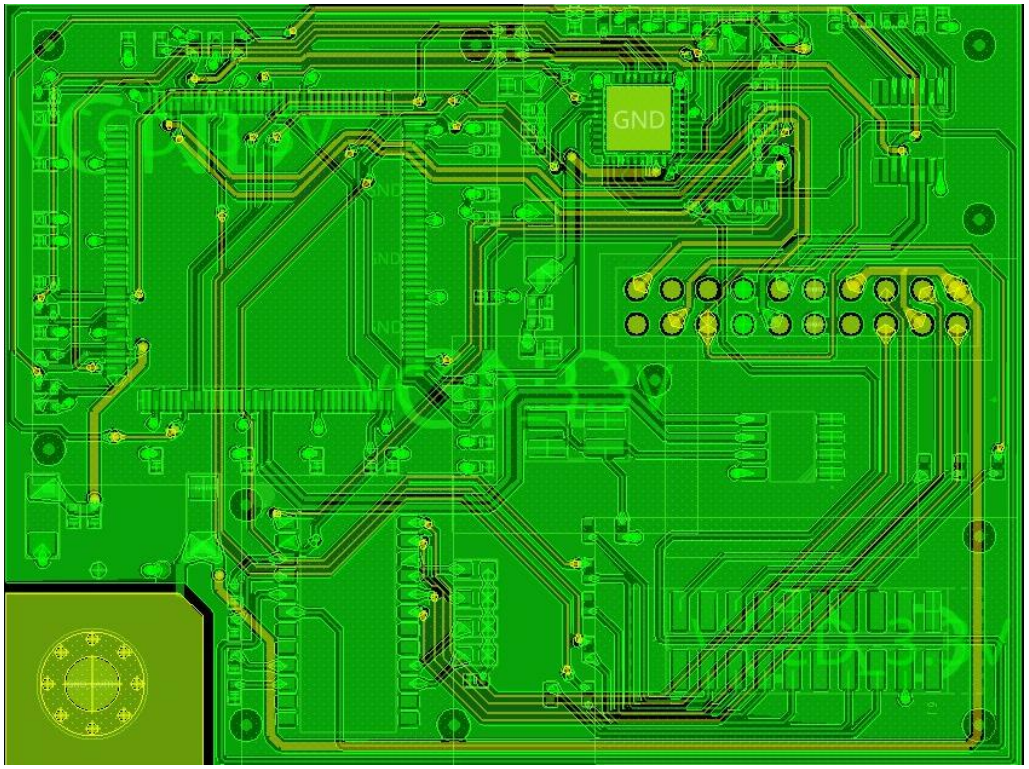




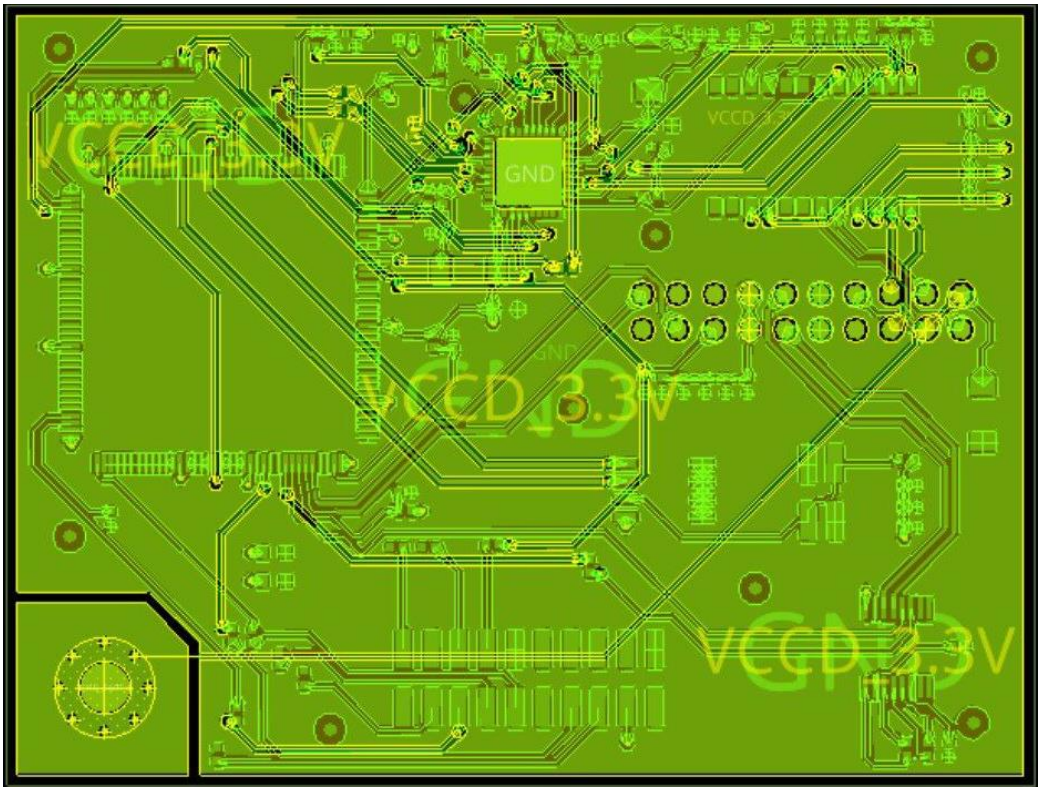
It is possible to simulate data transfer between two ICs in PCB design with using cadence simulation feature. After adding models to application and choosing related module for desired pin, the application provides diagram about the quality of data transferring and impact of the value of source destination resistors and mismatch in wire impedances.

Final Design in cadence:

Savlani's board:



Afshari's board:



Project Planning:

Phases history:

Phase 1:
Duration: 5 Mordad to 12 Mordad
Task: Find theoretical information and design primary circuit for controller on the paper
Phase 2:
Duration: 13 Mordad to 6 Shahrivar
Task: Design primary schematic on the cadence
Phase 3:
Duration: 7 Shahrivar to 17 Shahrivar
Task: Debug schematic and design PCB on the cadence

Agile working:

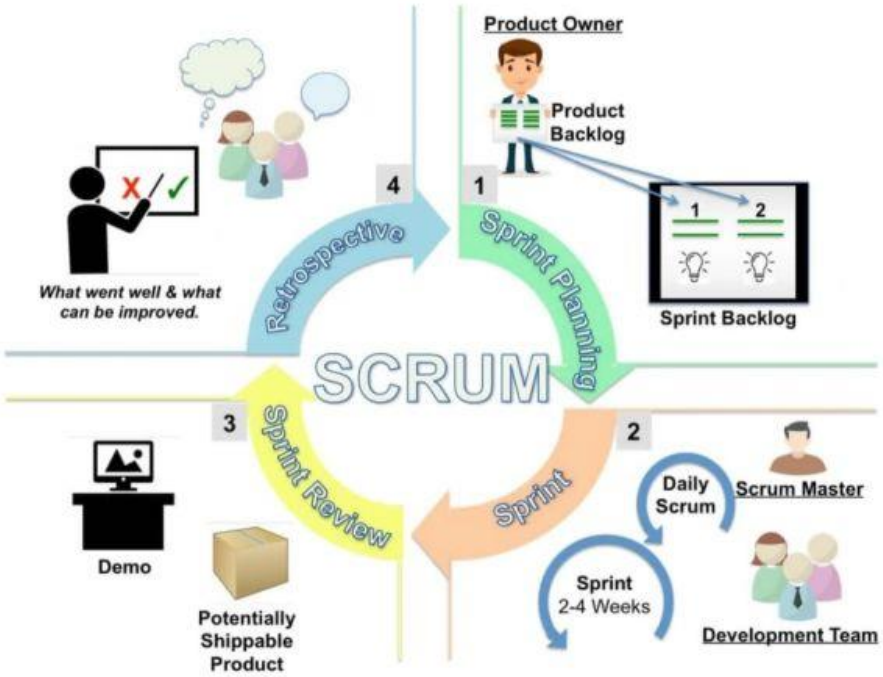
Scrum team:

The scrum team is a fundamental aspect of the scrum framework. It typically consists of the following roles:

Development team: These are professionals with the skills required to deliver a potentially shippable product increment. The development team is responsible for the actual work of creating the product.

Product owner: The product owner represents the interests of the stakeholders and is responsible for managing the product backlog and prioritizing work.

Scrum master: The scrum master is pivotal in facilitating the scrum process and ensuring that the team adheres to scrum principles.



Scrum master vs. project manager:

The scrum master's non-technical counterpart is the project manager. Both roles focus on the "how" of getting work done and solving workflow problems through process improvement. Are both roles required to manage agile projects successfully? The short answer is no.

While a traditional project manager and a professional scrum master are responsible for helping their teams get work done, their approaches are vastly different. Project managers set project milestones, report on team progress, and facilitate effective communication. However, they do so from a place of control. Conversely, scrum masters help teams enhance and streamline the processes by which they achieve their goals. They do so as a team member or collaborator—not by exerting total control. The best scrum teams are self-organizing and, therefore, don't react well to micromanagement.

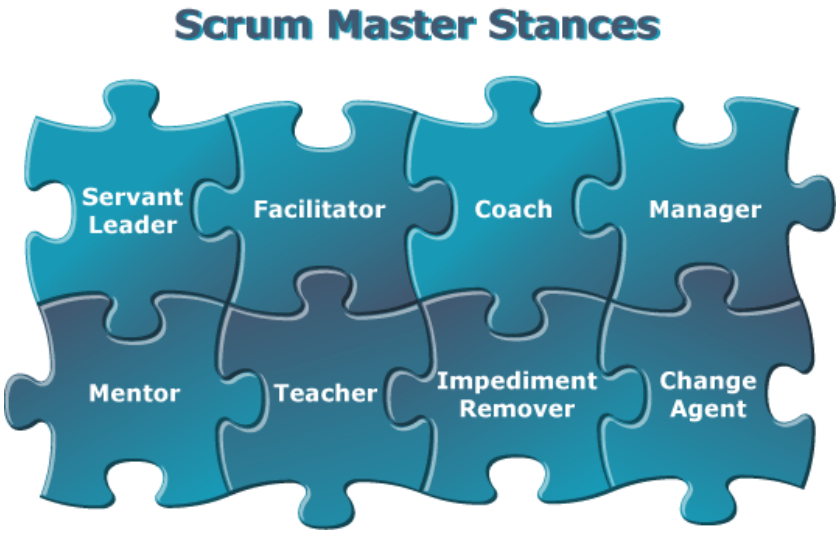
These are just a few of the possible configurations of scrum team management. Some companies make do with all of these roles, some have one or none at all.

Scrum master (Mehrad Afshari Nazari):

Who is Scrum master?

Scrum Masters do a lot of critical work that helps the team and the organization.

Scrum Master's main responsibility is to organize the development team to make progress based on the project's main priorities. In other words, the Scrum Master guides the team in the right direction.
To be successful, Scrum Master must wear different hats that depend on the situation or challenge the team is facing. In any given situation, a Scrum Master utilizes their important soft skills to act as a Servant Leader, Facilitator, Coach, Manager, Mentor, Teacher, Impediment Remover or Change Agent, depending on the situation at hand.



As a servant leader:
May aim to share positive energy and motivation with others and encourage the development of them.
As a facilitator:
Planning, guiding and managing a group meetings or events to ensure that it meets its goals.
As a couch:
They help the team members in self-management and cross-functionality
When a team member faces a challenge that stops work from completing a sprint, it is up to the Scrum Master to remove the obstacles.
As a teacher and mentor:

Maybe They have some technical experiences and scientific knowledge about the project and can give a hint.

As an impediment remover:

The Scrum Master helps the team to resolve conflicts and work together. It also helps to remove obstacles that block the team's progress.

As a change Agent:

Acts as a catalyst for the change management process..

As a program manager:

They should ensure that all Scrum events take place and are positive, productive, and kept within the time box. The Scrum Master is required to hold short and precise meetings so that useful information can be transferred to the team members in a short period of time. For example, this person leads the team's daily stand-up so that people talk about what they've done and present their daily schedule.

Scrum framework

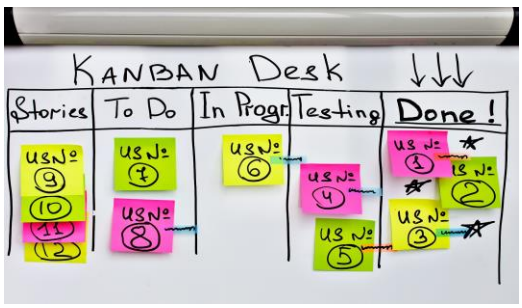
The scrum framework is a structured approach to agile project management methodology. It consists of several components:

Sprints: Sprints are time-boxed iterations, typically two-to-four weeks in length, where the development team works to complete a set of planned work items.

Product backlog: The product backlog is a prioritized list of all the features, enhancements, and bug fixes the product needs to address. The product owner manages and maintains this backlog.

Sprint backlog: The sprint backlog is a subset of the product backlog, containing the work items selected for a specific sprint.(**To do list**)

Here we have a format of table it names as Kanban Table which is for writing and grouping your task.

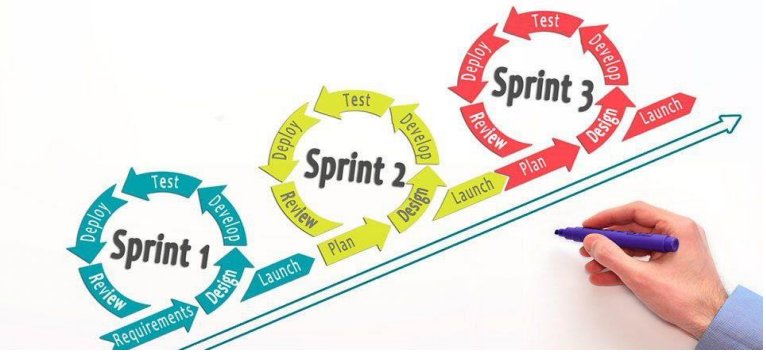


Daily standups: This is a short meeting that is held every day at a fixed time and place. The purpose of the daily standing meeting is to check in with the team and see how they are doing. Team members each briefly report on what they worked on yesterday, what they are working on today, and what obstacles they are facing. The Scrum Master facilitates the meeting and helps it stay on track.

Sprint planning meeting: This is a meeting that is held at the beginning of each sprint. The purpose of the sprint planning meeting is to decide what work will be done in the sprint. The team works together to estimate the work and create a sprint backlog. The scrum master facilitates the meeting and helps the team stay focused on the sprint goals.

Sprint review meeting: This is a meeting held at the end of each sprint. The purpose of the sprint review meeting is to show the stakeholders the work done in the sprint. The team presents the work and stakeholders provide feedback. The Scrum Master facilitates the meeting and helps it stay on track. (But in small groups the scrum master himself can be stakeholder)

Sprint retrospective meeting: This is a meeting held at the end of each sprint. The purpose of the sprint review meeting is to review the sprint and identify ways to improve.



The team will discuss what went well, what could have been better, and what they plan to do differently in the next sprint. The Scrum Master facilitates the meeting and helps the team stay focused on continuous improvement. The scrum framework emphasizes flexibility, adaptability, and continuous improvement, making it a popular choice for managing complex projects in an agile manner.

There is another role that is important for scrum master:

As a Product manufacturer:

The Scrum Master must communicate with the Product Owner to ensure a complete understanding of the project goals and scope of activities. These people explore different techniques for work with the product owner to maximize value and also they have liability to the product owner. because he or she needs to establish empirical product planning for a complex environment in some cases.

How I did it:

First we made a Telegram group with our mentor, Mrs Shatterzadeh. After that, we talked about meeting times in the company department. We decided to come on Saturdays and wednesdays. Then we tried to write sprints and tasks for our deadlines.

Sprints:

We have two weekly sprints for every week in phases:

1- Saturday to Wednesday

2- Wednesday to Saturday.

End of the meeting day, we check our situation and we approve some remote tasks to do until the next meeting. For example:

7 Mordad to 10 Mordad:

At Office: Debug installation problems + Learn designing schematic in Cadence 22.1 + Talk about theoretical things (parts and devices) with mentor.

At Home: Learn designing schematic in Cadence 22.1 + Try to design primary diagram blocks.

14 Mordad to 17 Mordad:

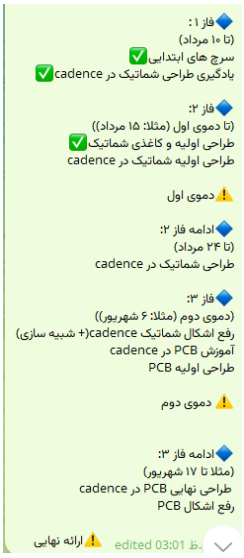
At Office: Starting to design schematic in Cadence and improve it with mentor

At Home: Complete phase two of the report and get ready for the first demo presentation + completing schematic design.

To do list:

It needs to be said that we were writing a task list and our theoretical results in the telegram group.First, We made agreements on structure and some symbols for writing our tasks list. To prioritize our tasks, we put warning sign for important ones and also have some colors. Also we write our needs and questions as a task for a mentor and that was not a bad idea.

For example:



Example of our "to do list" as kanban table:

Week1(Sprint 1&2)	To Do	Doing	Test and debug	Done
Savlani	-Instal cadence -Search about PoE -Finish proposals	-Installing cadence -Search about PoE -Writing proposals	Problem in zip file - -	No Yes Yes
Afshari	-Instal cadence -Search about PoE	-Clearing storage -Search about PoE	Problem in zip file -	No Yes
Mrs.Shaterzadeh	- Send educational content for cadence	-	-	Yes

Week3(Sprint 5&6)	To Do	Doing	Test and debug	Done
Savlani	-write report for project - Power point of Demo 1 -design schematic in cadence	-Designing ports, UART	Complete connections and STM32	Yes
Afshari	-write report for project -design schematic in cadence	-Designing Ethernet	Complete connections and STM32	Yes
Mrs. Shaterzadeh	- sending libraries and standards of designing	-	Checking designed circuit in the group and meeting	Yes

Week5(Sprint 9&10)	To Do	Doing	Test and debug	Done
Savlani	-Power of stm32 -Update URT - Add EEPROM to report	-Update URT	Schematic finished PCB started	Yes Yes NO
Afshari	-JTAG -Update Ethernet -Reset internal stm32 -Write control board part in report - Write Phase2 of report	-JTAG -Reset internal stm32 -Writing control board part in report - Writing Phase2 of report	Schematic finished PCB started	Yes Yes Yes Yes Yes
Mrs. Shaterzadeh	PCB footprint	Updating footprints	Debug PCB	Yes

Reviews:
Daily: Everyday 10 PM on Telegram group
Sprints: End of the meeting days
Weekly: After 2 sprints left in one week

Product Owner (Arezoo Savlani):

A Product Owner is a key role in a project, particularly within Agile frameworks like Scrum. The Product Owner acts as the bridge between the development team and the stakeholders, ensuring that the final product meets the needs of the users and the business. Here are the main responsibilities and attributes of a Product Owner:

Responsibilities:

1. **Defining the Product Vision:**
    - Articulate the overall vision and direction for the product.
    - Ensure the team and stakeholders have a clear understanding of the goals and the desired outcomes.
2. **Managing the Product Backlog:**
    - Create, prioritize, and maintain the product backlog, a dynamic list of tasks, features, and improvements.
    - Ensure that the backlog items are clearly defined, well-understood, and prioritized based on business value, customer needs, and feasibility.
3. **Stakeholder Engagement:**
    - Act as the main point of contact for stakeholders, gathering requirements, feedback, and ensuring their needs are met.
    - Communicate the product vision and progress to stakeholders, managing their expectations and involvement.
4. **Collaboration with the Development Team:**
    - Work closely with the development team to ensure they understand the product backlog items.
    - Participate in Sprint Planning, Reviews, and Retrospectives to provide clarity and feedback.
5. **Maximizing Product Value:**
    - Make decisions about the product's direction, features, and release plans to maximize the value delivered.
    - Balance the needs of users, the business, and technical feasibility.
6. **Accepting Work:**
    - Review and accept the work completed by the development team, ensuring it meets the acceptance criteria and the definition of done.

Attributes:

Customer-Centric: Focused on understanding and addressing the needs and problems of the users.

Decisive: Capable of making tough decisions quickly to keep the project moving forward.

Communicative: Excellent at communicating with both technical and non-technical stakeholders.

Analytical: Able to analyze data and feedback to make informed decisions about the product's direction.

Visionary: Capable of seeing the big picture while also managing the details.

Importance in Agile Projects:

- **Adaptability:** The Product Owner helps the team adapt to changes in requirements or market conditions.
- **Continuous Improvement:** By constantly refining the backlog based on feedback, the Product Owner ensures the product evolves effectively.
- **Ownership and Accountability:** The Product Owner is responsible for the success of the product, which drives them to make decisions that align with business goals and user needs.

Actions Taken by Product owner

- Ensure clarity about the project's purpose, customer needs, and the tools and devices to be used through sessions with the mentor.
- Review the project with another member in the presence of the mentor to reach a common understanding.
- Understand the timeline and expectations of the mentor for implementing the project.
- Provide useful articles to bring the team's knowledge to an appropriate level.
- Obtain the Cadence file, an application for designing boards, and follow the steps for installation. Watch educational videos and ensure all team members can follow them. In case of misunderstandings or lack of knowledge about hardware definitions or app functions, clarify the situation and solve the problems.
- Check project planning with the Scrum Master to ensure that the overall plan provides enough time for implementing each sprint, and reorder some tasks to maintain a smooth workflow.

Development team:

Because our group has just two people and we are not too many, scrum master and product owner are developers too.

Sessions with mentor:

At Hardware Office: Saturdays and Wednesday

Remotely: Telegram group in daily reviews

Demos:

First: (17 Mordad)

Phase 1 completely Done and presented

Phase 2 only up to first design presented

Second: (6 Shahrivar)

Phase 2 completely Done and presented

Phase 3 completely Done and presented

We have found some problems for debug after second demo in phase 2,3

Couching:

In this internship we had opportunity to learn about what is coaching and experiencing coaching sessions. It was really helpful to increasing our soft skills and also growing our personal knowledge.

Conclusion:

At the conclusion of our internship, we held a presentation to showcase our progress and finalize the project review. A group of mentors and key figures from the company came together to evaluate both our technical abilities and the overall quality of the project. During the presentation, we discussed the project's objectives, future direction, and shared insights from our experience working within the organization.(For example, we need to think more about MCU oscillator and NPI framework)

We were proud to have achieved the project's goals and successfully presented our final design.

Appendix:

- Site for making Block diagram:
  Block Diagram Maker | Free Block Diagram Online | Lucidchart
- Project's GitHub repository:
  GitHub PoE-Injector
- Project question document:
  questions - schematic - Poe team
- internship hours – Arezoo Savlani
  Parman weekly report - ArezooSavlani
- internship hours – Mahrad Afshari Nazari
  Parman weekly report - MahradAfshari

Special thanks to:

Mrs. Shaterzadeh for her help as our mentor. Mr. Kashani as HR and Internship Head. Mr. Rashidi and Mr. Mazaheri as coordinators. Mr. Olfat as Internship Supervisor. And the Hardware department of Parman Company for their kind and helpful support.