NCN26010XMNEVK 10BASE-T1S SPE Evaluation Kit User's Manual

EVBUM2833/D

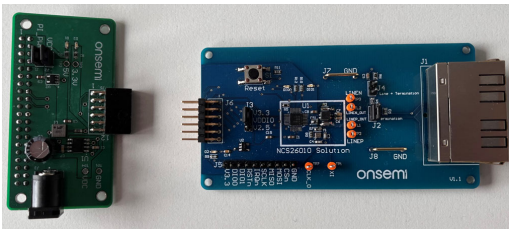

Figure 1. Evaluation Board Photo

Introduction

The NCN26010XMNEVK 10BASE−T1S MACPHY evaluation kit is a set of PCBs designed to allow customers access to **onsemi**'s NCN26010 SPI enabled 10BASE−T1S MACPHY. Its main purpose is to demonstrate the MACPHY's basic functionality and doubles as a lab tool to allow customers to develop their own embedded software drivers for the NCN26010 device.

Features

The evaluation kit consists of the NCN26010BMNEVB bridge board and the NCN26010XMNEVB 10BASE−T1S evaluation board.

The bridge board provides a power connector that allows feeding the boards with a DC supply of anything between 8 V and 28 V. For full operation, when a Raspberry Pi Model 3 is connected to the board, the power supply needs to be able to provide 10 W of continuous power.

The NCN26010XMNEVB evaluation board, carrying the NCN26010 has two RJ45 connectors that allow attaching a twisted single pair data cable. For ease of use, standard cat 5 Ethernet cables can be connected to the RJ45 connectors. Note that only one pair (connected to pin 1 and 2) of the RJ45 connector is used.

The SPI connection can be done through a PMOD connector on the NCN26010 evaluation board, allowing the connection to MCU demo and eval boards of various vendors. One example of such an Eval board is **onsemi**'s RSL10 BDK−GEVK.

To allow monitoring the SPI traffic, the EVB features a 0.1" pitched pin header that is easily accessible. This header could also be used to connect a logic analyzer, allowing tracing of the SPI signals while in full operation together with a (not supplied) Raspberry Pi or any other suitable Host MCU or SBC.

The bridge board not only acts as a power supply for the NCN26010 evaluation board, but also powers a Raspberry Pi that can be connected to the 40−pin female connector, bringing the SPI and interrupt request signals from the NCN26010 MACPHY to the corresponding pins on the Raspberry Pi SBC.

After trying different models of Raspberry Pi single board computers, onsemi got best results with Raspberry Pi 3B and 3A+ models running "off the shelf" Raspberry OS with some software tools installed.

See section "Raspberry Pi Setup and Operation" for a list of suggested tools.

When using a Raspberry Pi model 3 with a 10/100 Ethernet interface, T1S can be bridged with this port allowing connection to a regular PC.

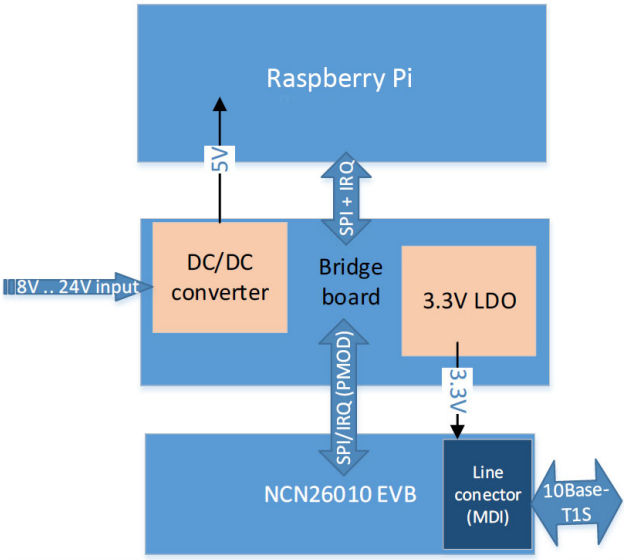

Figure 2. Simplified Block Diagram

APPLICATIONS INFORMATION

For details of the NCN26010BMNEDB and NCN20610XMNEVB boards that make the NCN26010XMNEVK, please consult the user guides of these boards.

The 10BASE−T1S evaluation board needs a host computer to perform any network communication. The NCN26010 performs the functions of a MAC (media access controller) and a PHY (physical layer) device.

The upper layer protocols need to be provided by software running on the host. The host will also have to handle the SPI communication between the MACPHY and the host, since the MACPHY operates as an SPI slave device.

In addition, the MACPHY provides an IRQ signal that signals new RX data and other events that need to be handled by the host.

Kit Content

An Evaluation Kit contains one NCN26010XMNEVB 10BASE−T1S MAPHY evaluation bard and one NCN26010BMNEVB bridge board which can be used to connect Raspberry Pi SBCs. Note that the Raspberry Pi is not supplied as part of the kit and needs to be externally sourced by the user of this kit.

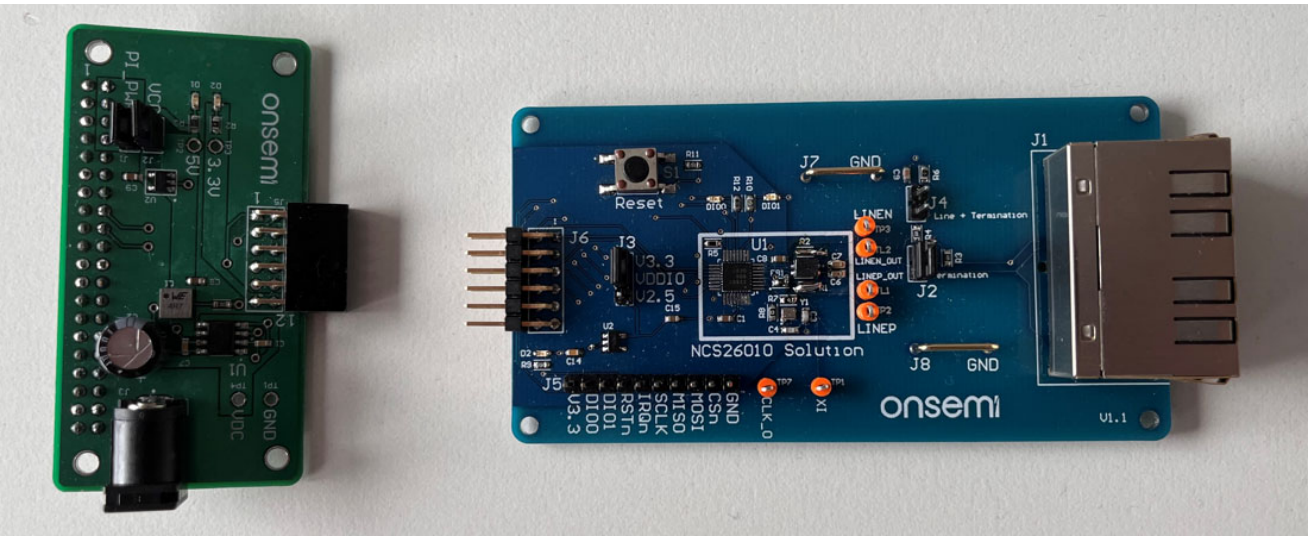

Figure 3. NCN26010XMNEVK Kit Content

Figure 4 shows an assembled Evaluation Kit including a Raspberry Pi.

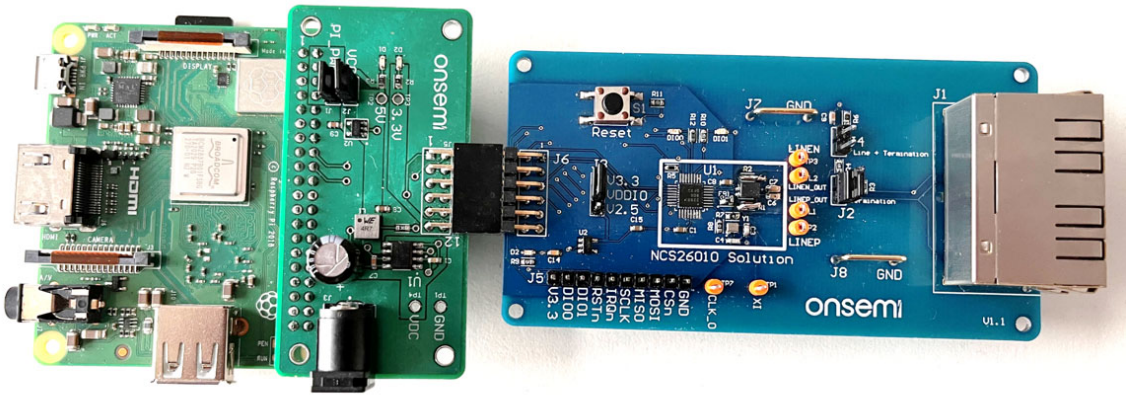

Figure 4. NCN26010XMNEVK Connected to a Raspberry Pi 3A+

Raspberry Pi Setup and Operation

By the nature of Ethernet, the board will only be functional when connected to a host computer which provides the upper layer protocols (like ARP, ICMP and TCP/IP or alike).

For simplicity, **onsemi** decided to use a proven, Linux based platform for this purpose to minimize the amount of development and get to a working solution quickly.

The Raspberry Pi is not provided and needs to be provided by the user.

onsemi recommends using a Raspberry Pi 3 B+ or Raspberry Pi 3 A+.

Preparing the Raspberry Pi

A 16 GB micro SD card is required as mass storage on the Raspberry Pi SBC.

The following steps need to be executed to make the Raspberry Pi work as expected:

1. Obtain Raspberry OS in its latest version and install it on the SD card as explained on the Raspberry Foundations web site at https://www.raspberrypi.org/software/
2. When running headless (without a monitor attached), create an empty file called “ssh” in the /boot directory of the SD card in which the Raspberry OS is installed.
   There is no harm to enable SSH even if you intend to run a Desktop environment
3. Boot the Raspberry Pi (one for every board), login, and run
   sudo raspi−config
   a. Using “1 System Options” → “S4 Hostname”, set a UNIQUE host name

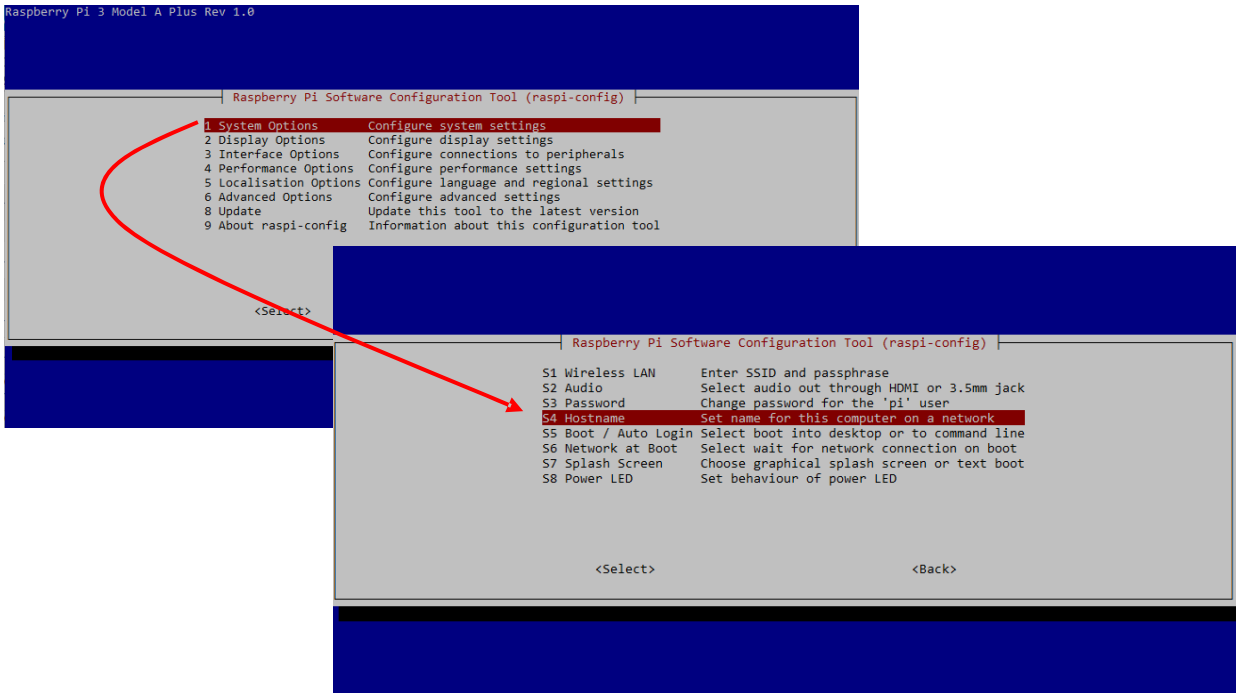

Figure 5.

b. Enable the SPI interface by using “3 Interface Options” → “P4 SPI”

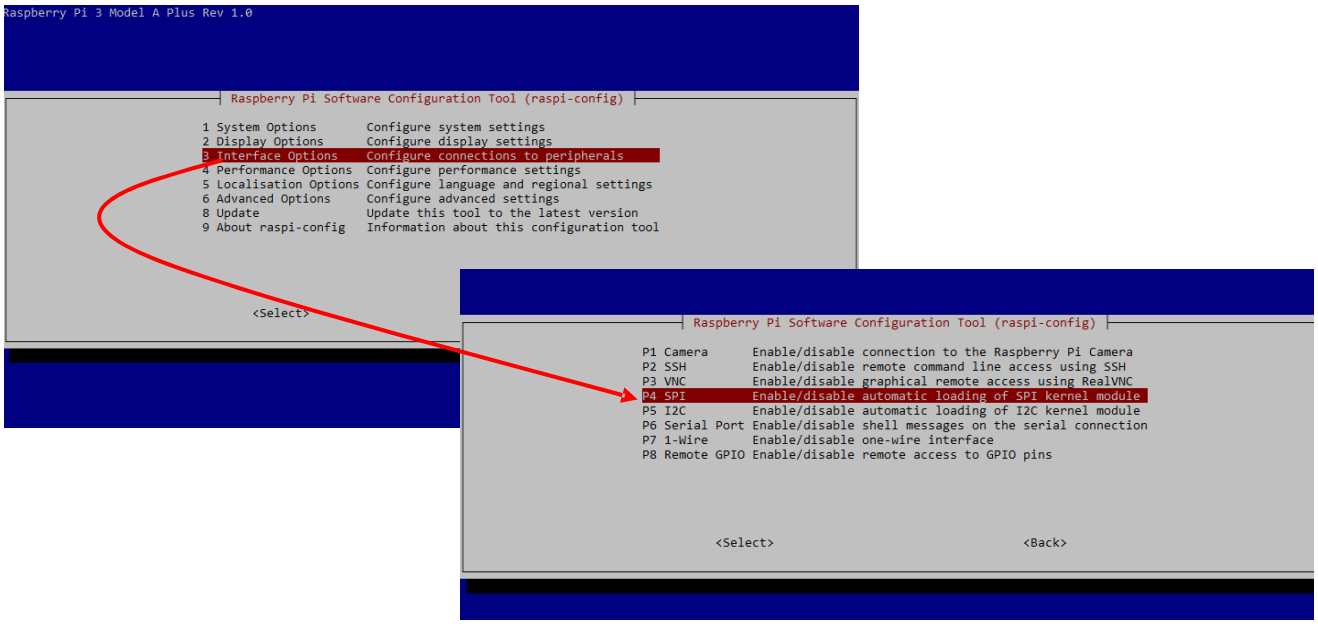

Figure 6.

c. Depending on the specific application needs, enable or disable the Camera−Interface, I2C, UART or the Remote GPIO and appropriately configure all other settings.

4. Reboot the Raspberry Pi

5. Install additional software

For performance measurements onsemi suggests installing “iperf3” which allows you to measure data throughput between two nodes

The Raspberry Pi needs to be connected to the internet to install software from the Raspberry Pi repository

Open a console on the Raspberry (either via ssh or using the desktop) and type:

sudo apt−get update
sudo apt−get install iperf3

Note that the update command only needs to be done once.

Install all other software tools following the same approach.

Tools that might be needed:

- tcpdump
- Wireshark
- Bridge tools, for bridging the T1S with the 100Base TX port of the Raspberry Pi.
  This would allow you to integrate a regular PC into the T1S subnet using one Raspberry Pi as a bridge
- A DHCP server of your choice if you want a bridge to configure a connected PC automatically

Preparing, Compiling, and Running the Demo Application

The demo application prepared to run on Raspberry Pi is a user space “driver” that provides routines to read and write Ethernet frames to/from the 10BASE−T1S connected MACPHY and allows reading and writing configuration registers, allowing testing features provided by NCN26010

Among others these features are:

- Switching between PCLA and CSMD/CD mode of operation
- Configure various aspect of the PLCA mode (ID, burst rate, max IDs)
- Enable/Disable ENI (enhanced noise immunity) mode

The T1_MACPHYDriver program makes use of the tun/tap interface provided by Linux.

This “virtual” network interface allows layer 2 Ethernet frames to be exchanged between a user space program and the Linux kernel. The Linux kernel provides the upper layer protocols above layer 2 of the OSI−ISO. Namely, this is the TCP/IP stack.

Once up and running, the 10BASE−T1S MACPHY is presented to the Linux operating system on the “tap0” interface and will act like any other network interface in the system.

Compiling

onsemi’s demo application runs in a Linux console. The application is available pre−compiled on request or it can be downloaded at https://www.onsemi.com

Users do not have to compile the application themselves, instead they can just copy the program into the home directory of the Raspberry Pi.

On request, onsemi can make available the source code of this application. The source code will contain instructions on how to compile the application.

Running the Program

Before the program can be started, the tap interface needs to be initialized on the Raspberry Pi.

To do this, execute these commands. Note that all these need to run as root.

Listing 1: TAP CONFIGURATION

```
sudo ip tuntap add tap0 mode tap user pi
sudo ifconfig tap0 10.1.1.3 netmask 255.255.255.0 hw ether 60:c0:bf:01:01:03 txqueuelen 10
sudo route add default gw 10.1.1.1 tap0
sudo ethtool −s tap0 duplex half speed 10
```

Note that every station needs to be assigned a unique IP address and a MAC address. For experiments with Packet Filtering, Packet sniffing (using Wireshark) it is beneficial to choose hostname, IP address and MAC address to be somewhat related.

In the above example, we named the host “pi03” assigned it a 10.1.1.3 address and made the last significant three bytes of the MAC address match the last three bytes of the IP address.

Also, note that we set the tap 0 to half duplex and its speed to 10 Mbit/s. This way we avoid the queues in the Linux TCP/IP stack to send data that the MACPHY will never be able to handle.

Once the tap is initialized, start the T1S driver as root:

sudo ./T1S

When running, the application accepts a small number of commands to make changes to the MACPHYs configuration on the fly.

Commands

1. Read a register:
   To read a MACPHY register, type:
   R <MMS> <Address>

Example:

```
Enter command in HEX> r 0 0001
Command Entered is r 0 0001

Before Read- Memory map: 0, Reg address : 0x0001
Register value for register 0x0001 from MMS 0 is 0x7E818061
Enter command in HEX>
```

2. Write to a register
   To write a register type:
   W <MMS> <Address> <32 bit data in hex format>

Example:

```
Enter command in HEX> w 1 0000 00000103
Command Entered is w 1 0000 00000103

Before Write- Memory map: 1, Reg address : 0x0000, Reg value 0x00000103
Register value read after writting for register 0x0000 from MMS 1 is 0x00000103
Enter command in HEX>
```

3. Reporting Link (MAC) statistics
   The MACPHY has several statistics counter (see product preview for details)
   To read all these registers in one go and get a human readable report of the counted events, simply type:
   S

Example:

```
MMS1 - Address 0x0030 -> Sent Bytes Low:  412
MMS1 - Address 0x0031 -> Sent Bytes High:  0
MMS1 - Address 0x0032 -> Frames Sent OK: 4
MMS1 - Address 0x0030 -> Sent Bytes Low:  412
MMS1 - Address 0x0031 -> Sent Bytes High:  0
MMS1 - Address 0x0032 -> Frames Sent OK: 4
MMS1 - Address 0x0033 -> Broadcast Frames sent OK: 0
MMS1 - Address 0x0034 -> Multicast Frames Sent OK: 0
MMS1 - Address 0x0035 -> 64-Byte Frames Sent OK: 2
MMS1 - Address 0x0036 -> 65 - 127 Byte frames sent OK 0
MMS1 - Address 0x0037 -> 128 - 255 Byte Frames sent OK: 2
MMS1 - Address 0x0038 -> 256 - 511 Byte Frames sent OK: 0
MMS1 - Address 0x0039 -> 510 - 1023 Byte Frames sent OK: 0
MMS1 - Address 0x003a -> 1024 or larger byte Frames sent OK: 0
MMS1 - Address 0x003b -> Frames aborted due to TX-buffer underflow: 0
MMS1 - Address 0x003c -> Frames transmitted after singel collision: 0
MMS1 - Address 0x003d -> Frames transmitted after multiple collisions: 0
MMS1 - Address 0x003e -> Frames transmitted after excessive collisions: 0
MMS1 - Address 0x003f -> Frames transmitted after deferal: 0
MMS1 - Address 0x0040 -> CRS de-assertions during frame transmission Errors: 0
MMS1 - Address 0x0041 -> Receive Bytes Counter Low: 36846689
MMS1 - Address 0x0042 -> Received Bytes Counter high 0
MMS1 - Address 0x0043 -> Frames received OK: 33840
MMS1 - Address 0x0044 -> Broadcast frames received OK: 12
MMS1 - Address 0x0045 -> Mulitcast frames received OK: 49
MMS1 - Address 0x0046 -> 64 Byte Frames resceived OK: 524
MMS1 - Address 0x0047 -> 65 - 127 Byte frames received OK: 6864
MMS1 - Address 0x0048 -> 128-255 Byte frames received OK: 323
MMS1 - Address 0x0049 -> 256-511 Byte frames received OK: 140
MMS1 - Address 0x004a -> 512-1023 Byte Frames recevied OK: 198
MMS1 - Address 0x004b -> 1024 Byte Frames received OK: 25791
MMS1 - Address 0x004c -> Droped Frames too short errors: 0
MMS1 - Address 0x004d -> Dropped frame to long errors: 0
MMS1 - Address 0x004e -> droped frame due to wrong FCS errors: 0
MMS1 - Address 0x004f -> symbol errors during frame reception: 0
MMS1 - Address 0x0050 -> Align errors during frame reception: 0
MMS1 - Address 0x0051 -> RX buffer overflow orrors 151
MMS1 - Address 0x0052 -> RX dropped frame count: 0
Enter command in HEX>
```

4. Quit the “driver”
   To stop the driver, simply type:
   Q

Command Line Options

The driver offers two command line options

1. PLCA

   To make starting in PLCA operation simpler than having to configure the correct registers manually, a command line option has been added to the program
   For starting the head node, use:
   −plca 0 <max ID>
   Where <max ID> is the number of the station with the highest PLCA ID + 1
   For starting a normal plca node use:
   −plca <ID>
   Where <ID> is the PLCA ID you want to assign. Note that the ID must be larger than zero and smaller than
   max ID
   0<= ID <= max ID

2. ServerMode

   When you intend to start the driver at startup or in the background,
   "ServerMode" as a command line option. This will disable **the interactive user interface**.

Connecting a Microcontroller

The NCN26010XMNEVB evaluation board is designed to allow customers to use a Microcontroller of their choice and interface with the MACPHY driver using the SPI interface.

This connection can either be done through the PMOD connector on the evaluation board or through the debug and monitoring connecter on the lower edge of the evaluation board. For details, please consult the NCN26010XMNEVB user guide, available at https://www.onsemi.com

On request, onsemi can provide some example implementations for select number of different MCUs, **onsemi**'s RSL10 BLE radio chip with integrated low power Cortex M3 being one of them.

A suitable Microcontroller needs to operate the SPI at >= 15 MHz and support interrupts though GPIO to be able to react to service requests issued by the MACPHY through the IRQ signal.

Setting Up a Network

To setup a functional T1S network, connect a twisted pair unshielded cable on the J1 connectors (dual RJ45). J1 makes a physical connection of the two RJ45 sockets, allowing users to use standard ethernet cables or crimp their one SPE cables using cheap plastic unshielded JR45 connectors.

Note that you can connect more than two boards to the same cable. Per IEEE standard, a maximum cable length of 25m and eight stations can be used on the same cable.

Results may vary with different wire gauges and PHY settings (i.e. ENI mode should give a higher count of possible stations and/or an extended reach).

When assigning the stations with IP addresses of the same network, you can use Linux standard tools like ping to check if there is a connection between all the stations.

In addition, you can point Wireshark to the tap0 interface to monitor the network traffic.

Note that by default, the driver program operates the MACPHYs in "promiscuous" mode, where every station forwards all frames to the TCP/IP stack, regardless of its destination address.

You can configure address filtering to limit the traffic on the SPI interface especially in situation where you use microcontrollers that might not have the performance to deal with the amount of traffic on the SPI when using "promiscuous" mode.

See datasheet and application note for details on how the address filtering works.

Custom configurations can be put into a "MACPHY_Configfile.txt" configuration file. If a file with that name is in the same directory as the actual application program, the program executes register writes contained in that file upon start up, hence configuring the MACPHY accordingly.

Here is a sample content:

```
# NCN26010 MACPHY Config File
# reset the device
0 0003 00000001
# enable MAC RX and TX, enable addr filtering
1 0000 00010103
# PHY register, bring link up
0 FF00 00001000
# set LEDs to RX and TX activity
c 0012 8D8F
# Set PLCA node ID to 2 and node count to 15
4 CA02 00000F02
# enable PLCA (head node)
4 CA01 00008000
# basic configuration CONFI0 register
# frame starts at CS
# 64 byte chunk size
# frame aligned with start of chunk
# sync bit set --- ready to go
0 0004 0000BC06
# MAC Filter, allow Broadcast and unicast
# mac Address is 60:C0:BF:01:01:03
1 0010 BF010103
1 0011 800060C0
# enable PLCA precedence mode
# 4 8002 00008800
```

NCN26010XMNEVB Schematic

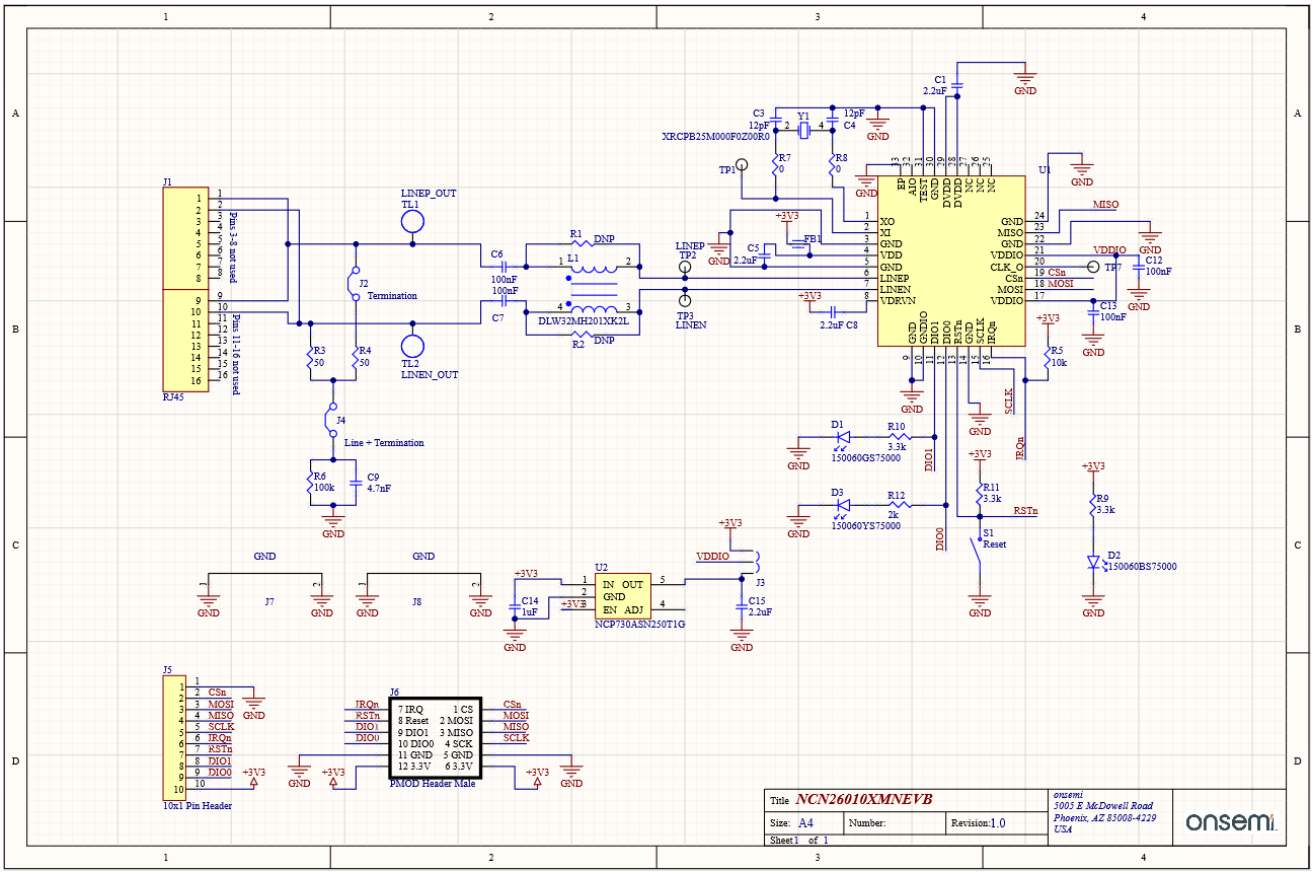

Figure 7. NCN26010XMNEVB Schematic

NCN26010BMNEVB Bridge Board Schematic

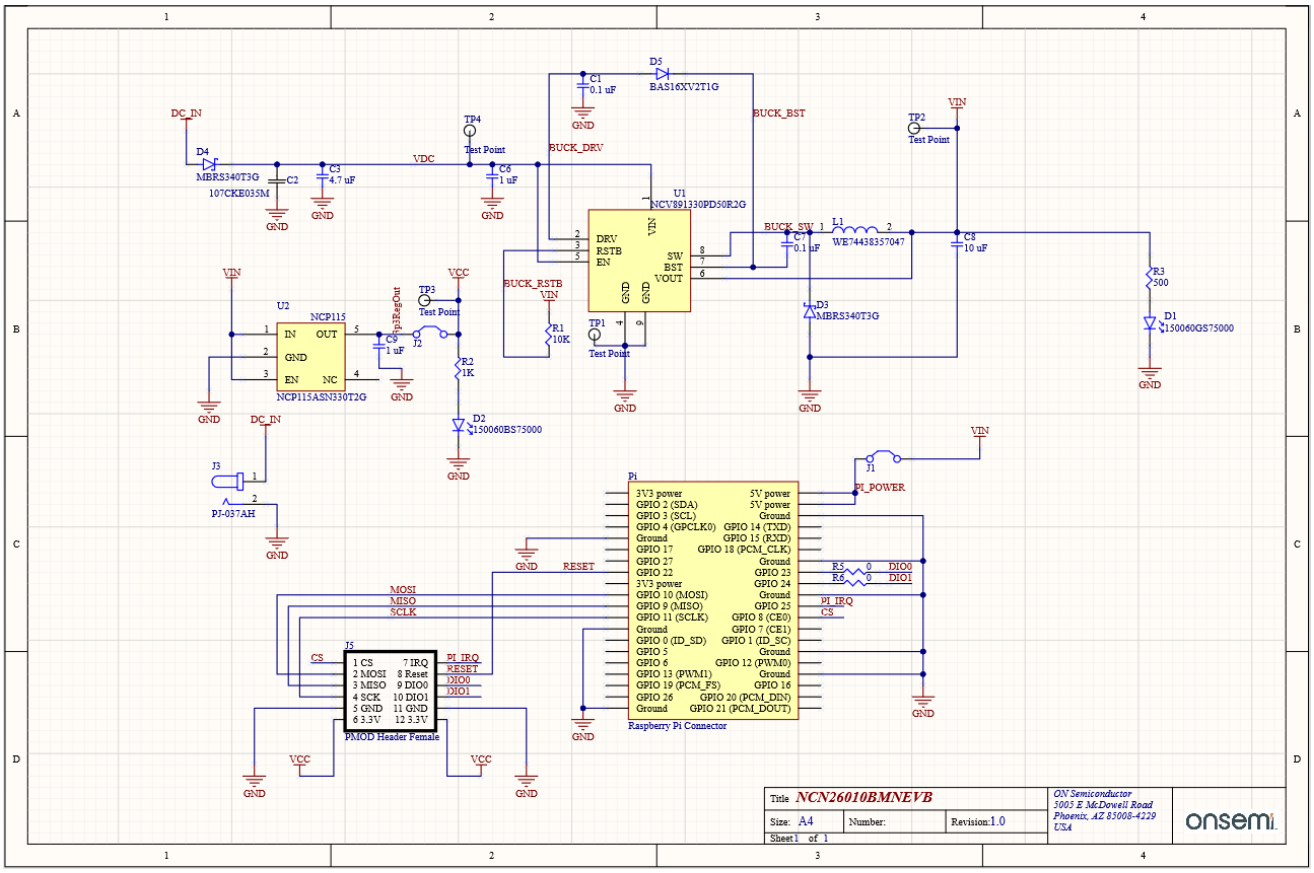

Figure 8. NCN26010BMNEVB Bridge Board Schematic