USB2T1S-C

Menu

1. General Description:

10BASE-T1S is an Ethernet standard designed specifically for short-distance, low-cost applications and is widely used in industrial automation, automotive electronics, and the Internet of Things (IoT). 10BASE-T1S is part of the IEEE 802.3cg standard, representing a single-pair twisted-pair Ethernet technology.
10BASE: Indicates a transmission rate of 10 Mbps.
T1: Refers to the use of a single-pair twisted pair for communication.
S: Represents support for short-reach applications.

The USB2T1S-C is a Hi-Speed USB to 10BASE-T1S Ethernet network convertor.Industrial-grade design and built-in ESD protection. By using a single-pair twisted pair for data transmission, supporting a maximum transmission distance of up to 25 meters. It supports multipoint communication, allowing multiple devices to share the same twisted pair (similar to a bus structure). Additionally, it uses IEEE 802.3's PLCA (Physical Layer Collision Avoidance) technology to prevent signal collisions and improve communication efficiency.

INNOMAKER USB2T1S-C has strong anti-interference capability to adapt to the noise in industrial and automotive environments. It is suitable for communication between controllers, sensors, and actuators in industrial automation, supporting industrial Ethernet standards such as PROFINET and EtherCAT. It is also applicable to automotive networks for in-vehicle low-speed communication (e.g., body control modules and sensor networks) and supports automotive Ethernet architectures. Additionally, it is ideal for IoT and building automation industries, serving as a low-cost, low-power, and highly efficient data transmission tool.

2. Features

1. Hi-Speed USB 2.0 to 10 Mbps Ethernet convertor, to interconnect a USB interface with a IEEE 802.3cg 10BASE-T1S Ethernet network interface. The convertor serves as a network card that connects applications via USB 2.0 to the 10BASE-T1S network interface.

2. Support multidrop mixing segments and physical layer collision avoidance (PLCA)

3. 10BASE-T1S single-pair Ethernet physical layer transceiver LAN8670

4. Industrial-grade design with built-in ESD protection devices ensures strong anti-interference capabilities, protecting signal lines within the device from electrostatic discharge (ESD) and surge interference.

5. The device uses a USB Type-A connector and a standard DB9 terminal. A converter board is included with the package, which allows configuration of two 49.9 Ω termination resistors and selection of wiring conversion methods.

3. Hardware Description

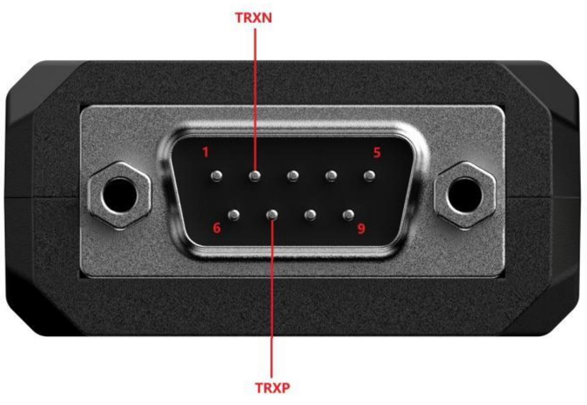
3.1 General Information

Items	Description
Power Input Requirement	Power is supplied through the USB port, 5V/200mA
Power Consumption	＜1W
Weight	78 g
Operating Temperature	-40-85 Celsius
T1S Phy Chip	LAN8670C2
USB to TX Phy Chip	LAN9500AI

3.2 Interface

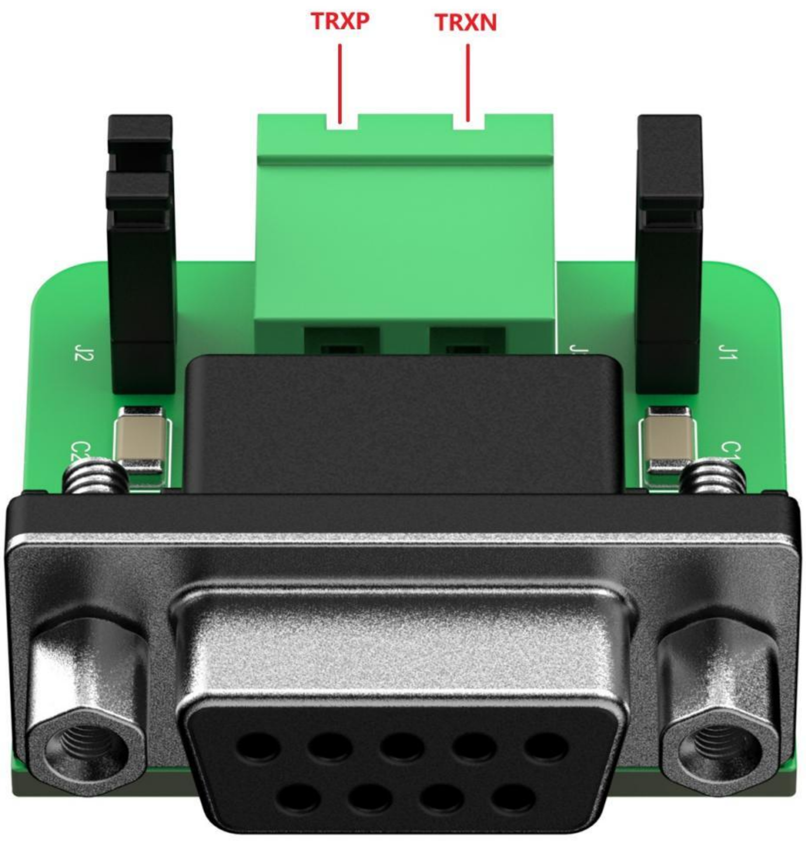
(1) DB9 Port

DB9 Port	T1S Port	Description
PIN 2	TRX_N	TRX_N (Negative terminal), Refers to the negative terminal of the signal, typically used to connect to the negative terminal of the communication line or device.
PIN 7	TRX_P	TRX_P (Positive terminal): Refers to the positive terminal of the signal, typically used to connect to the positive terminal of the communication line or device.
PIN 3 and PIN 6	GND	The reserved ground pin is not needed for grounding in actual T1S communication.



(2) Terminal

The termination jumpers (J1 and J2) are used for enabling two 49.9 Ohm edge termination at the ends of a 10BASE-T1S segment.Both jumpers must be closed to enable the edge termination.



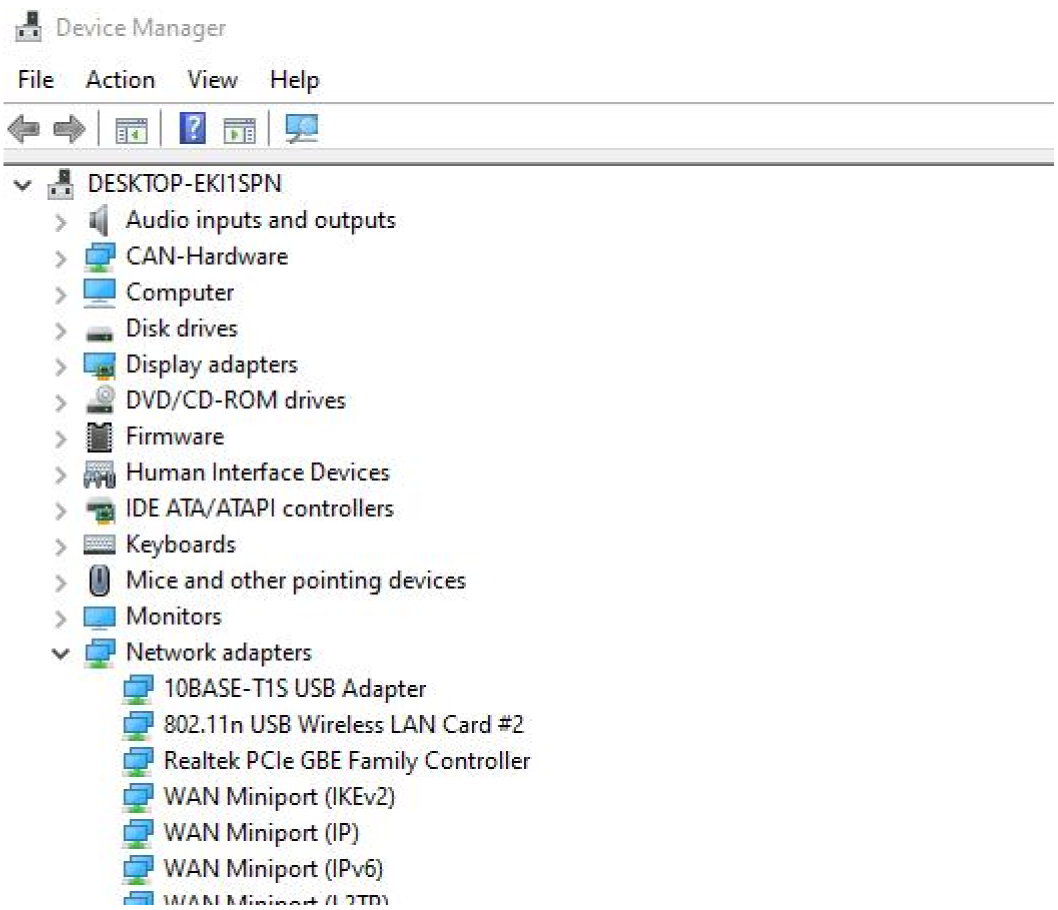
4. WIndows

Microchip provides ready-made drivers for Windows 10 and Linux, and the Windows 10 driver has been tested and works on Windows 11. If you have more questions about the drivers, you can contact **http://www.microchip.com/support**.

4.1 Windows Driver

Download and install the driver from the following link:
https://github.com/INNO-MAKER/10BaseToT1S

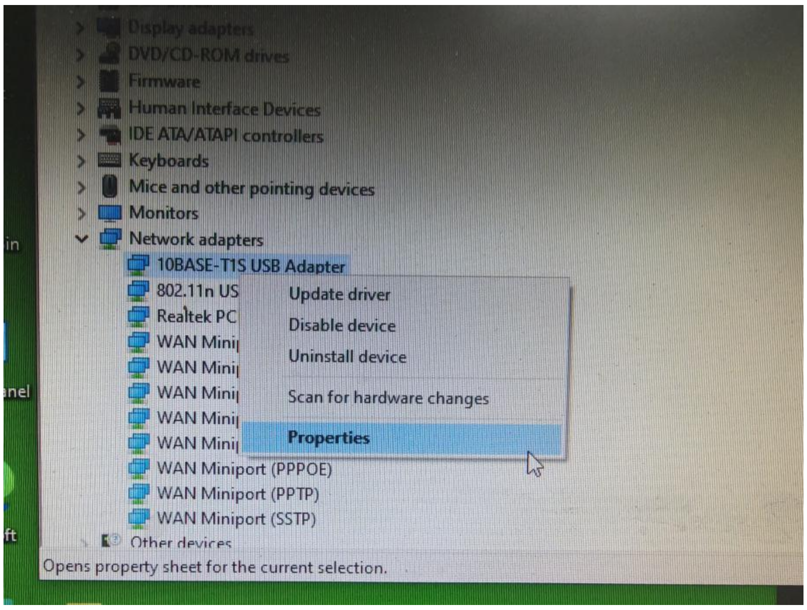
After successful installation, open the Windows Device Manager, and under the Network Adapters tab, you will see a device named "10BASE-T1S USB Adapter" without any exclamation mark.

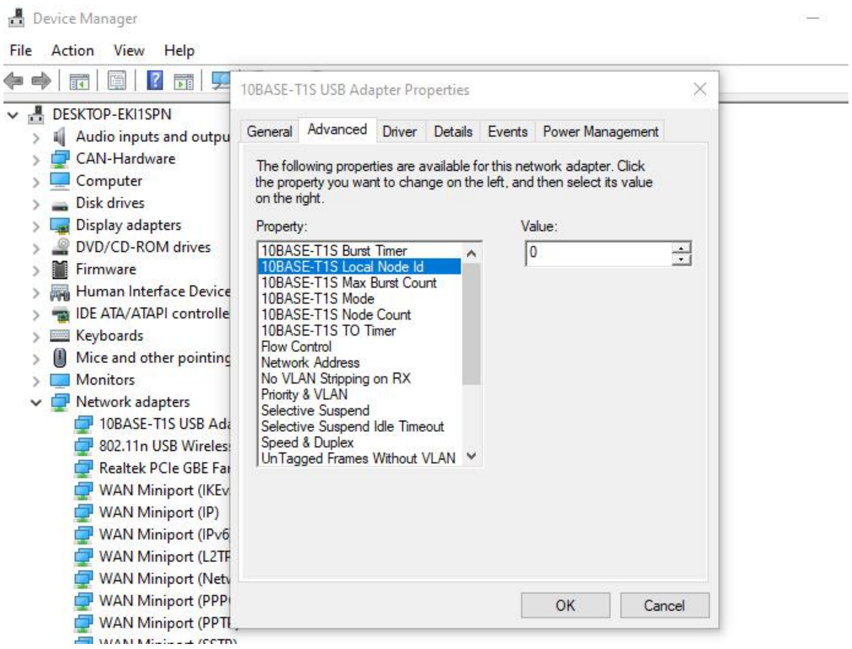


```
DESKTOP-EKI1SPN
  Audio inputs and outputs
  CAN-Hardware
  Computer
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  Firmware
  Human Interface Devices
  IDE ATA/ATAPI controllers
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
    10BASE-T1S USB Adapter
    802.11n USB Wireless LAN Card #2
    Realtek PCIe GBE Family Controller
    WAN Miniport (IKEv2)
    WAN Miniport (IP)
    WAN Miniport (IPv6)
```

4.2 Windows Node ID Setting

Right-click on the device name, then select "Properties" → "Advanced" → "10BASE-T1S Local Node Id". In the "Value" box, you can set the Node ID. Each T1S must have a Node 0, which is the master node. Also, the node IDs must not be duplicated.





The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property:
10BASE-T1S Burst Timer
10BASE-T1S Local Node Id
10BASE-T1S Max Burst Count
10BASE-T1S Mode
10BASE-T1S Node Count
10BASE-T1S TO Timer
Flow Control
Network Address
No VLAN Stripping on RX
Priority & VLAN
Selective Suspend
Selective Suspend Idle Timeout
Speed & Duplex
UnTagged Frames Without VLAN

Value: 0

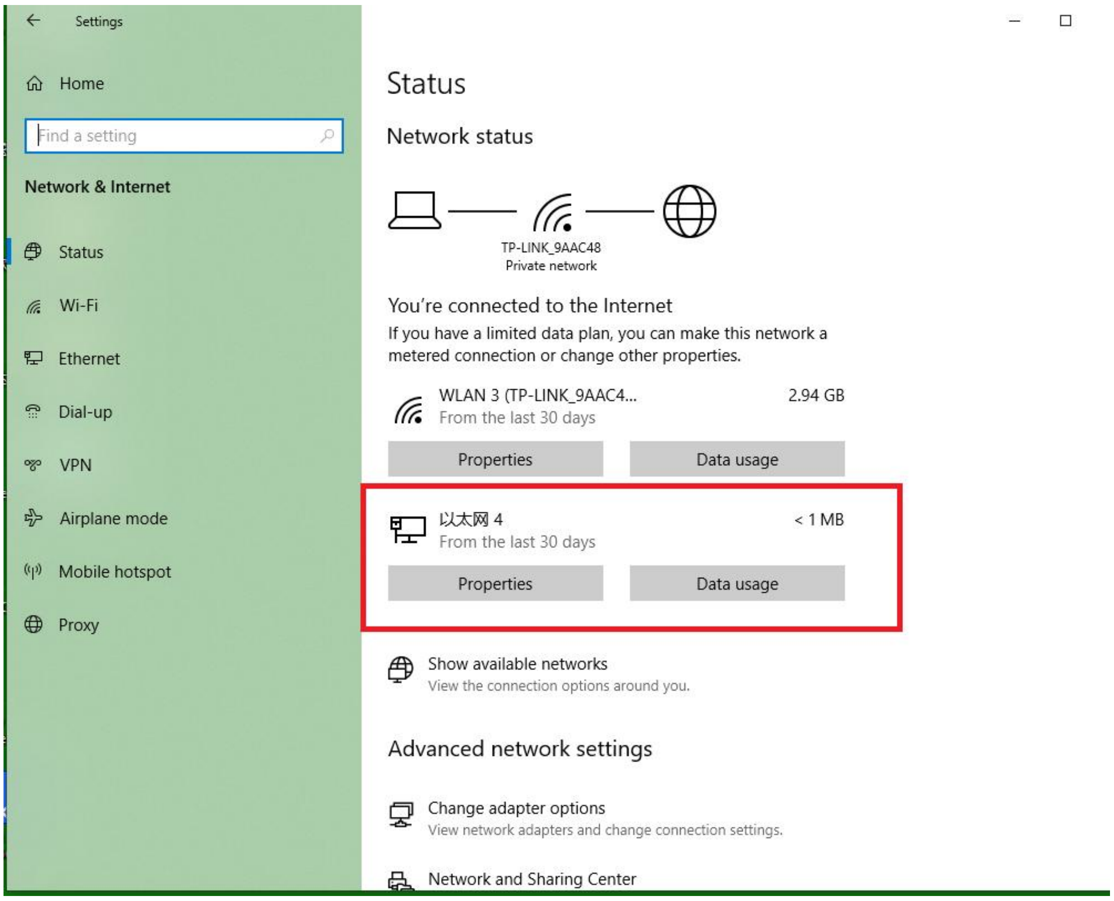
4.3 Windows Testing

After the driver installation is complete, you can treat this device as a 10M network interface and use it with almost all network software. Here, I will use the iperf3 tool for a simple demonstration. It is a well-established tool, and you can find plenty of usage information on Google, so I won't go into detailed explanations but will just provide the basic test command. This tool needs to be run through the command line on Windows.

(1) Connect the device to the existing T1S network. If you don't have one, you can purchase two USB2T1S-C devices for testing. Note that N should be connected to N, and P should be connected to P.

(2) Set the node. There must be one Node 0 as the master in the network, and the remaining nodes must not be duplicated.

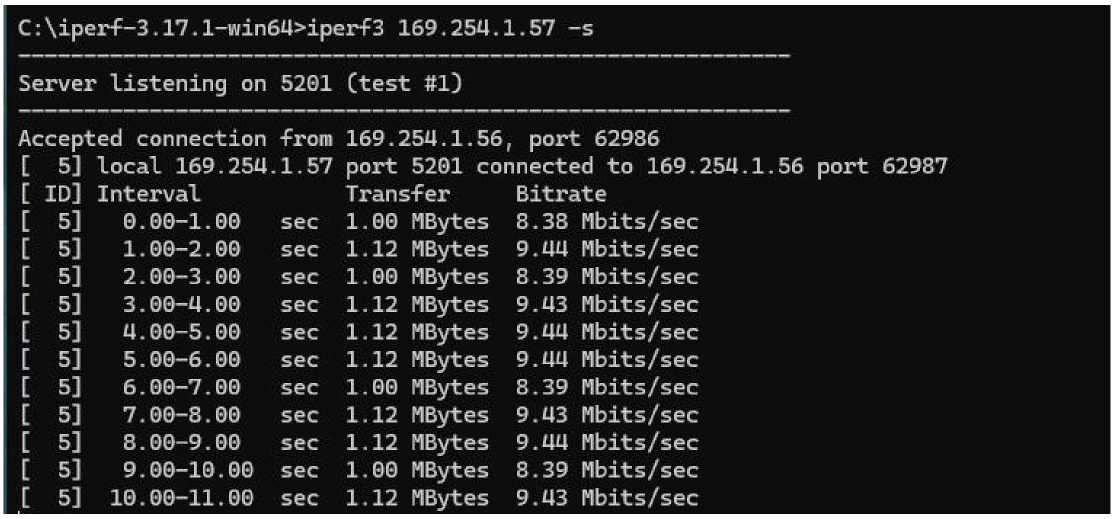
(3) In Windows, select "Settings" → "Network & Internet," and here you will see a new network device. Since the network IP address of the other T1S device I am using is 169.254.1.56, I will manually set the device's IP address to 169.254.1.57 here for easier communication.



(4) Run iperf3 in CMD, and use the following command to set the USB2T1S-C as the server, and

the other device works as client.

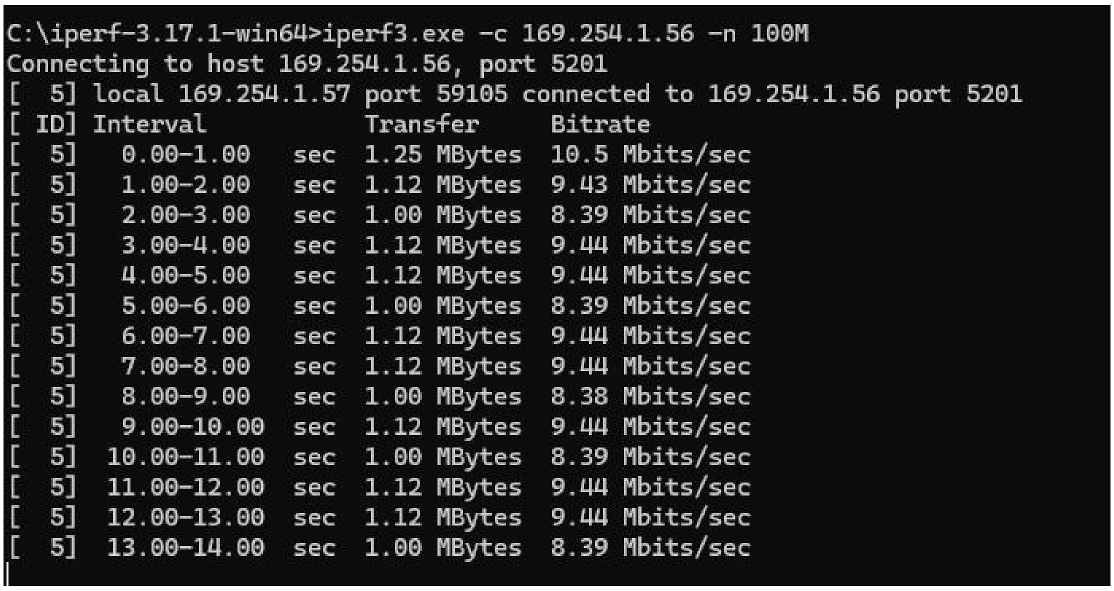
iperf3 169.254.1.57 -s



```
C:\iperf-3.17.1-win64>iperf3 169.254.1.57 -s
-----------------------------------------------------------
Server listening on 5201 (test #1)
-----------------------------------------------------------
Accepted connection from 169.254.1.56, port 62986
[  5] local 169.254.1.57 port 5201 connected to 169.254.1.56 port 62987
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  1.00 MBytes  8.38 Mbits/sec
[  5]   1.00-2.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   2.00-3.00   sec  1.00 MBytes  8.39 Mbits/sec
[  5]   3.00-4.00   sec  1.12 MBytes  9.43 Mbits/sec
[  5]   4.00-5.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   5.00-6.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   6.00-7.00   sec  1.00 MBytes  8.39 Mbits/sec
[  5]   7.00-8.00   sec  1.12 MBytes  9.43 Mbits/sec
[  5]   8.00-9.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   9.00-10.00  sec  1.00 MBytes  8.39 Mbits/sec
[  5]  10.00-11.00  sec  1.12 MBytes  9.43 Mbits/sec
```

(5) Run iperf3 in CMD, and use the following command to set the USB2T1S-C as the client.the other device works as server

iperf3 -c 169.254.1.56 -n 100M



```
C:\iperf-3.17.1-win64>iperf3.exe -c 169.254.1.56 -n 100M
Connecting to host 169.254.1.56, port 5201
[  5] local 169.254.1.57 port 59105 connected to 169.254.1.56 port 5201
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  1.25 MBytes  10.5 Mbits/sec
[  5]   1.00-2.00   sec  1.12 MBytes  9.43 Mbits/sec
[  5]   2.00-3.00   sec  1.00 MBytes  8.39 Mbits/sec
[  5]   3.00-4.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   4.00-5.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   5.00-6.00   sec  1.00 MBytes  8.39 Mbits/sec
[  5]   6.00-7.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   7.00-8.00   sec  1.12 MBytes  9.44 Mbits/sec
[  5]   8.00-9.00   sec  1.00 MBytes  8.38 Mbits/sec
[  5]   9.00-10.00  sec  1.12 MBytes  9.44 Mbits/sec
[  5]  10.00-11.00  sec  1.00 MBytes  8.39 Mbits/sec
[  5]  11.00-12.00  sec  1.12 MBytes  9.44 Mbits/sec
[  5]  12.00-13.00  sec  1.12 MBytes  9.44 Mbits/sec
[  5]  13.00-14.00  sec  1.00 MBytes  8.39 Mbits/sec
```

(6) By using the above two commands, you can see the communication results of the USB2T1S-C device. The speed will be approximately 8-9 Mbits/sec. The speed may vary depending on the connecting cables, test system, and testing tools.

4.4 Windows Trouble shooting

In Windows, apart from wiring issues, there are a few other things to check if device communication fails:

(1) The driver is not installed correctly. Check the device manager to see if the device driver is installed successfully.

(2) The node is not set; ensure that there is at least one node with ID 0 on the bus, and there are no ID conflicts in the network.

(3) The Windows firewall or antivirus software is not turned off.

(4) The IP address is not set correctly. Ensure that the USB2T1S-C device is on the same network segment as the external network or device.

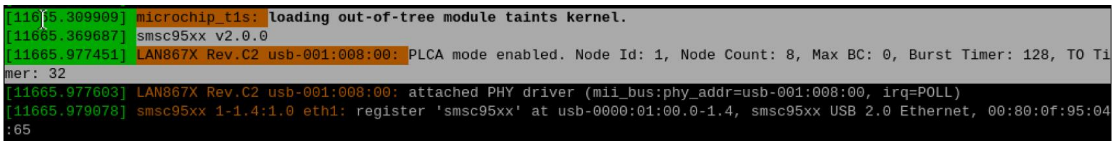
5.LINUX

5.1 Linux Driver

Download and install the driver from the following link:
https://github.com/INNO-MAKER/10BaseToT1S

The following demonstration will use Raspberry Pi + Linux 6.6.31 kernel. Please note that you must compile and add the driver to the system. After installing the driver, insert the device into USB port. Use the dmesg command to check if the driver installation was successful. This is important because even if the driver is not installed successfully, many Linux systems have default drivers that allow the device to function properly. However, when using these default drivers, a large number of packet losses may occur.

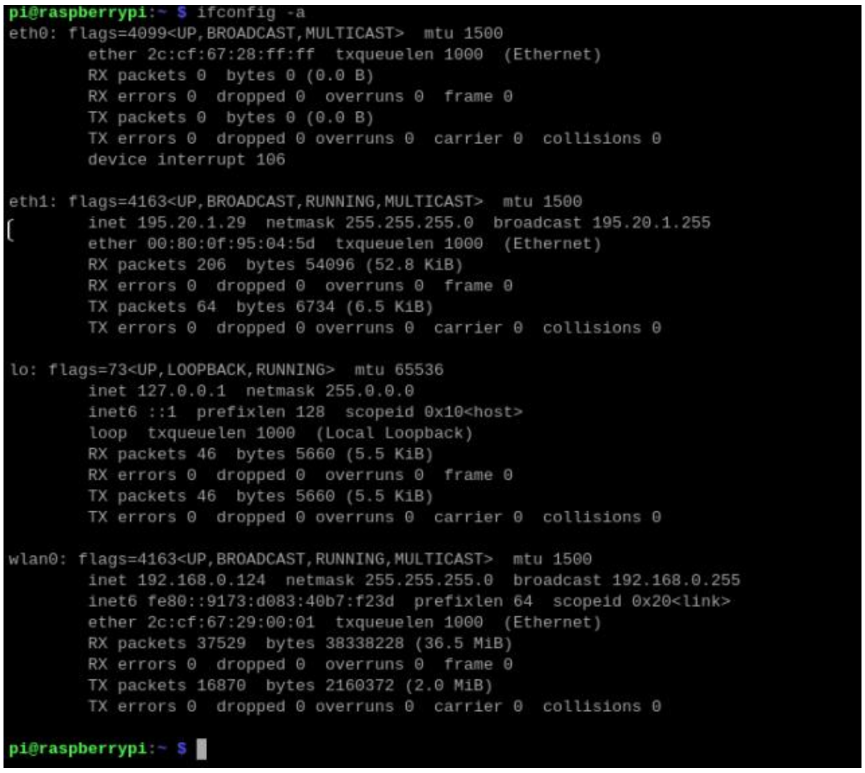
demsg



```
[11665.309909] microchip_t1s: loading out-of-tree module taints kernel.
[11665.369687] smsc95xx v2.0.0
[11665.977451] LAN867X Rev.C2 usb-001:008:00: PLCA mode enabled. Node Id: 1, Node Count: 8, Max BC: 0, Burst Timer: 128, TO Ti
mer: 32
[11665.977603] LAN867X Rev.C2 usb-001:008:00: attached PHY driver (mii_bus:phy_addr=usb-001:008:00, irq=POLL)
[11665.979078] smsc95xx 1-1.4:1.0 eth1: register 'smsc95xx' at usb-0000:01:00.0-1.4, smsc95xx USB 2.0 Ethernet, 00:80:0f:95:04
:65
```

You will see an additional network interface in LINUX, corresponding to eth1 on my system.

Ifconfig -a

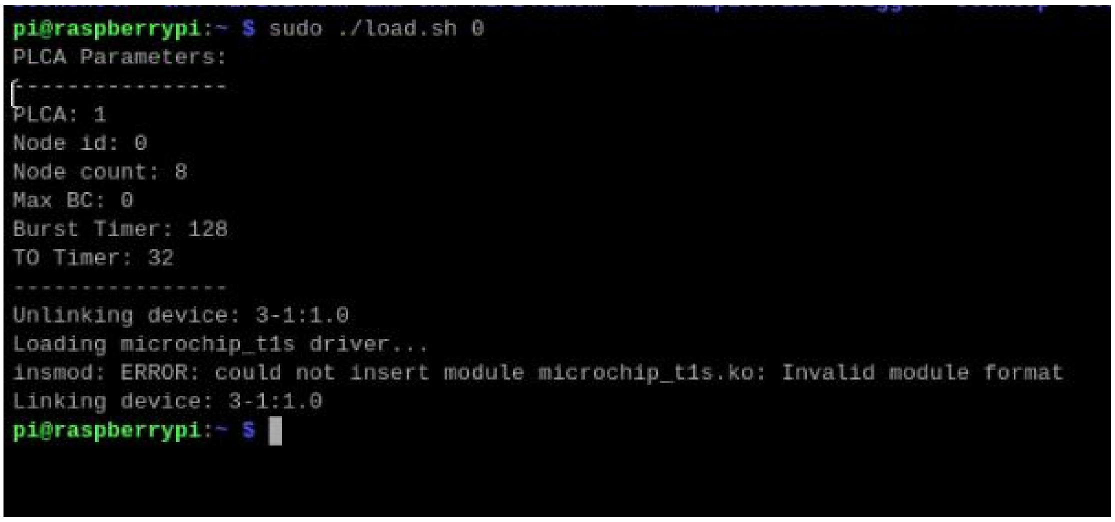


```
pi@raspberrypi:~ $ ifconfig -a
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 2c:cf:67:28:ff:ff  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 106

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 195.20.1.29  netmask 255.255.255.0  broadcast 195.20.1.255
        ether 00:80:0f:95:04:5d  txqueuelen 1000  (Ethernet)
        RX packets 206  bytes 54096 (52.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 64  bytes 6734 (6.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 46  bytes 5660 (5.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 46  bytes 5660 (5.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.124  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::9173:d083:40b7:f23d  prefixlen 64  scopeid 0x20<link>
        ether 2c:cf:67:29:00:01  txqueuelen 1000  (Ethernet)
        RX packets 37529  bytes 38338228 (36.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16870  bytes 2160372 (2.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $ 
```

5.2 Linux Node ID Setting

After the driver installation, Run the load.sh script in the driver package and add the node number to set the current device's node. There must be one Node 0 as the master in the network, and the remaining nodes must not be duplicated.

sudo ./load.sh 0



```
pi@raspberrypi:~ $ sudo ./load.sh 0
PLCA Parameters:
----------------
PLCA: 1
Node id: 0
Node count: 8
Max BC: 0
Burst Timer: 128
TO Timer: 32
----------------
Unlinking device: 3-1:1.0
Loading microchip_t1s driver...
insmod: ERROR: could not insert module microchip_t1s.ko: Invalid module format
Linking device: 3-1:1.0
pi@raspberrypi:~ $
```

5.3 Linux Testing

After the driver installation is complete, you can treat this device as a 10M network interface and use it with almost all network software. Here, I will use the iperf3 tool for a simple demonstration. It is a well-established tool, and you can find plenty of usage information on Google, so I won't go into detailed explanations but will just provide the basic test command.

(1) Connect the device to the existing T1S network. If you don't have one, you can purchase two USB2T1S-C devices for testing. Note that N should be connected to N, and P should be connected to P.

(2) Here, I am using 2 Raspberry Pi 4 and 2 10BASE-T1S devices for communication. Through the test-s.py file, one is set with IP: 195.20.1.29, node: 1, and starts the server mode of the iperf3 test tool. The other is set with IP: 195.20.1.31, node: 0, and starts the client mode of the iperf3 test tool. I have written the setup and communication startup process in a Python script for convenient one-time execution. Alternatively, it can also be done by entering commands one by one through the command line.

The content of test-s.py is as follows:

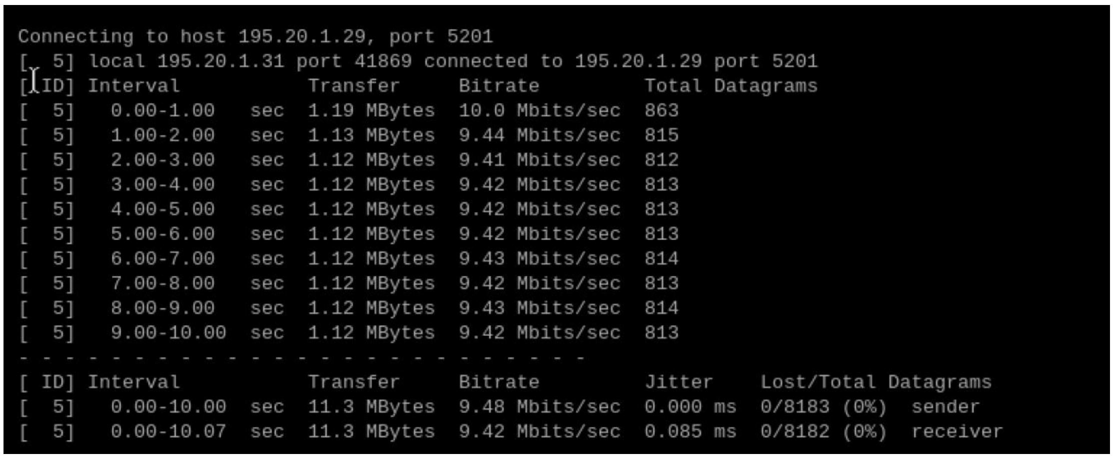
```
os.system('sudo killall iperf3')
os.system('sudo /home/pi/load.sh 0')
os.system('sudo ifconfig eth1 195.20.1.29')
os.system('sudo ifconfig eth1')
os.system('sudo iperf3 -s eth1')
```

The content of test-c.py is as follows:

```
#UDP TEST
os.system('sudo killall iperf3')
os.system('sudo /home/pi/load.sh 1')
os.system('sudo ifconfig eth1 195.20.1.31')
os.system('sudo ifconfig eth1')
os.system('sudo iperf3 -c 195.20.1.29 -b 10M -u')

#TCP TEST
#os.system('sudo killall iperf3')
#os.system('sudo /home/pi/load.sh 1')
#os.system('sudo ifconfig eth1 195.20.1.31')
#os.system('sudo ifconfig eth1')
#os.system('sudo iperf3 -c 195.20.1.29 -b 10M')
```

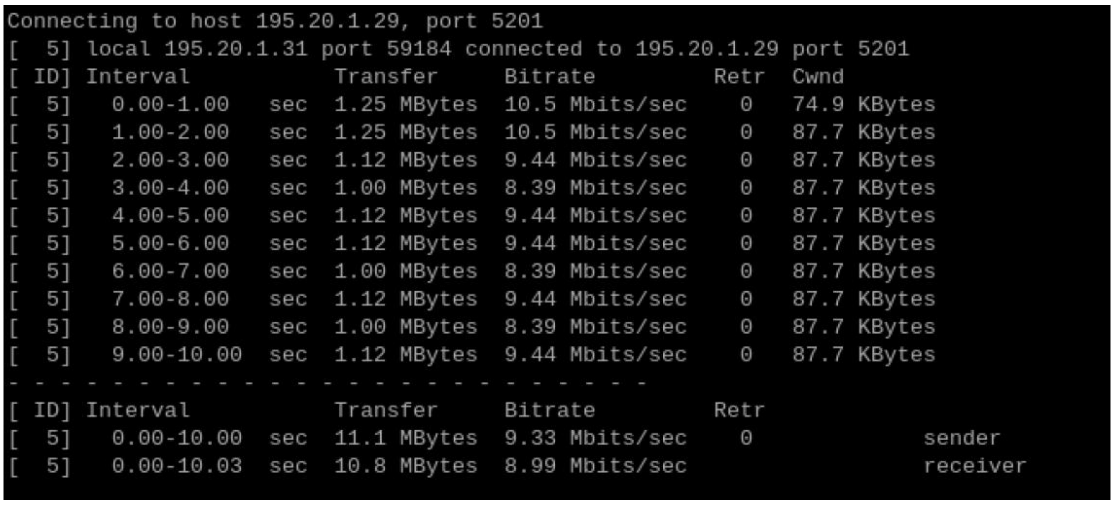
(3) Enter the file path in the command line on two Raspberry Pis, and run sudo python3 test-s.py and sudo python3 test-c.py respectively. You will see the following UDP and TCP/IP test results.

UDP test result:



```
Connecting to host 195.20.1.29, port 5201
[  5] local 195.20.1.31 port 41869 connected to 195.20.1.29 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  1.19 MBytes  10.0 Mbits/sec  863
[  5]   1.00-2.00   sec  1.13 MBytes  9.44 Mbits/sec  815
[  5]   2.00-3.00   sec  1.12 MBytes  9.41 Mbits/sec  812
[  5]   3.00-4.00   sec  1.12 MBytes  9.42 Mbits/sec  813
[  5]   4.00-5.00   sec  1.12 MBytes  9.42 Mbits/sec  813
[  5]   5.00-6.00   sec  1.12 MBytes  9.42 Mbits/sec  813
[  5]   6.00-7.00   sec  1.12 MBytes  9.43 Mbits/sec  814
[  5]   7.00-8.00   sec  1.12 MBytes  9.42 Mbits/sec  813
[  5]   8.00-9.00   sec  1.12 MBytes  9.43 Mbits/sec  814
[  5]   9.00-10.00  sec  1.12 MBytes  9.42 Mbits/sec  813
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-10.00  sec  11.3 MBytes  9.48 Mbits/sec  0.000 ms  0/8183 (0%)  sender
[  5]   0.00-10.07  sec  11.3 MBytes  9.42 Mbits/sec  0.085 ms  0/8182 (0%)  receiver
```

TCP/IP test result:



```
Connecting to host 195.20.1.29, port 5201
[  5] local 195.20.1.31 port 59184 connected to 195.20.1.29 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  1.25 MBytes  10.5 Mbits/sec    0   74.9 KBytes
[  5]   1.00-2.00   sec  1.25 MBytes  10.5 Mbits/sec    0   87.7 KBytes
[  5]   2.00-3.00   sec  1.12 MBytes  9.44 Mbits/sec    0   87.7 KBytes
[  5]   3.00-4.00   sec  1.00 MBytes  8.39 Mbits/sec    0   87.7 KBytes
[  5]   4.00-5.00   sec  1.12 MBytes  9.44 Mbits/sec    0   87.7 KBytes
[  5]   5.00-6.00   sec  1.12 MBytes  9.44 Mbits/sec    0   87.7 KBytes
[  5]   6.00-7.00   sec  1.00 MBytes  8.39 Mbits/sec    0   87.7 KBytes
[  5]   7.00-8.00   sec  1.12 MBytes  9.44 Mbits/sec    0   87.7 KBytes
[  5]   8.00-9.00   sec  1.00 MBytes  8.39 Mbits/sec    0   87.7 KBytes
[  5]   9.00-10.00  sec  1.12 MBytes  9.44 Mbits/sec    0   87.7 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  11.1 MBytes  9.33 Mbits/sec    0             sender
[  5]   0.00-10.03  sec  10.8 MBytes  8.99 Mbits/sec                  receiver
```

5.4 Linux Trouble shooting

In Linux, apart from wiring issues, there are a few other things to check if device communication fails:

(1) The driver is not compiled and loaded.

(2) The node is not set; ensure that there is at least one node with ID 0 on the bus, and there are no ID conflicts in the network.

(3)If there is a large amount of packet loss and retries during communication, use the dmesg command to check if the driver is installed correctly. When the driver is not properly installed, some Linux systems' default drivers can still recognize the device and establish communication, but they will experience significant packet loss.

(4)Confirm whether the network interface is eth0, eth1, or another one. Choose the corresponding network interface for operation.

(5)The IP address is not set correctly. Ensure that the USB2T1S-C device is on the same network segment as the external network or device.

(6) DHCP may automatically assign or change IP addresses, which is a common issue on Raspberry Pi. When using tools like IPERF3, this can cause the IP address to suddenly be lost or changed. Therefore, disable DHCP or assign a static IP to the device. When encountering sudden communication failure, use the ifconfig -a command to check if the IP address has been lost.

6.User Manual Version Descriptions

Version	Description	Date	E-mail
V1.0		2024.11.04	support@inno-maker.com sales@inno-maker.com

If you have any suggestions, ideas, codes and tools please feel free to email to me. Look forward to your letter and kindly share.