MASTERS 24

EMPOWERING INNOVATION

# 24037 NET1 Demonstration and Lab

## Unlocking the Potential of 10BASE-T1S: A Comprehensive Guide to Understanding, Developing and Evaluating Single Pair Ethernet with MPLAB® Harmony for Automotive and Industrial Networking

August 2024

**April Graham, Laurent Manhès, Wolfgang Wittmer**

MICROCHIP

# Lab: Learning Objectives

## What is the Purpose or Goal of this Class/Lab?

- When you join us in the “Unlocking the Potential of 10BASE-T1S” class, you will walk out of here knowing how to demonstrate a pre-configured Multi-drop 10BASE-T1S Network
- You will use MPLAB® Harmony to setup and configure a small 10BASE-T1S network and reconfigure your application's network settings and understand their impact on the communication in 10BASE-T1S networks.
- You will program your own application onto a microcontroller evaluation board to establish 10BASE-T1S network communication
- You will build an application on top of your project to add an OLED display to show the network and PLCA parameters
- You will then extend the physical network, that is the UTP bus line, without the need for any switches or repeaters.

# 24037 NET1: Overall Class Agenda (3 hours)

- **10BASE-T1S Presentation (80 minutes)**
- **Break (10 minutes)**

- **10BASE-T1S Network Demonstration (20 minutes)**
  - Understand the demo which uses a pre-configured multi-drop 10BASE-T1S network
  - Understand how to use the tools which are used to verify network performance

- **10BASE-T1S Network Lab (60 minutes)**
  - Use MPLAB® Harmony to setup and configure a small 10BASE-T1S network.
  - Reconfigure your application's network settings and understand their impact on the communication in 10BASE-T1S networks.
  - Program your own application onto a microcontroller evaluation board to establish 10BASE-T1S network communication.
  - Build an application on top of your project to add an OLED display to show the network and PLCA parameters
  - Extend the physical network, that is the UTP bus line, without the need for any switches or repeaters.

- **Q+A/Feedback (10 minutes)**

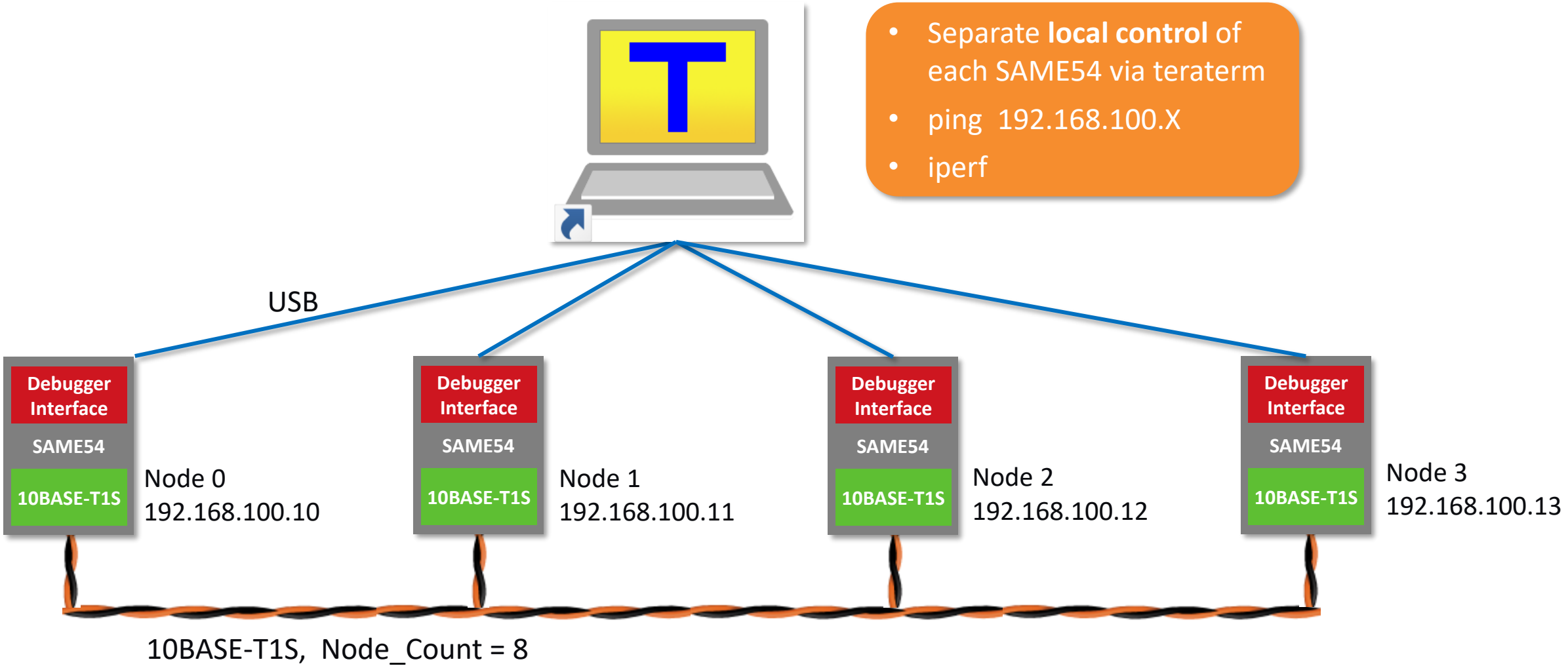
# 24037 NET1
# 10BASE-T1S Network Demonstration

# Demonstration: Learning Objectives

## What will you be able to do when you finish the activity?

- Attendees will know how to demonstrate a pre-configured Multi-drop 10BASE-T1S Network
- Understand how to use the tools which are used to verify network performance,
- Understand the effects of multidrop network loading on bandwidth utilization

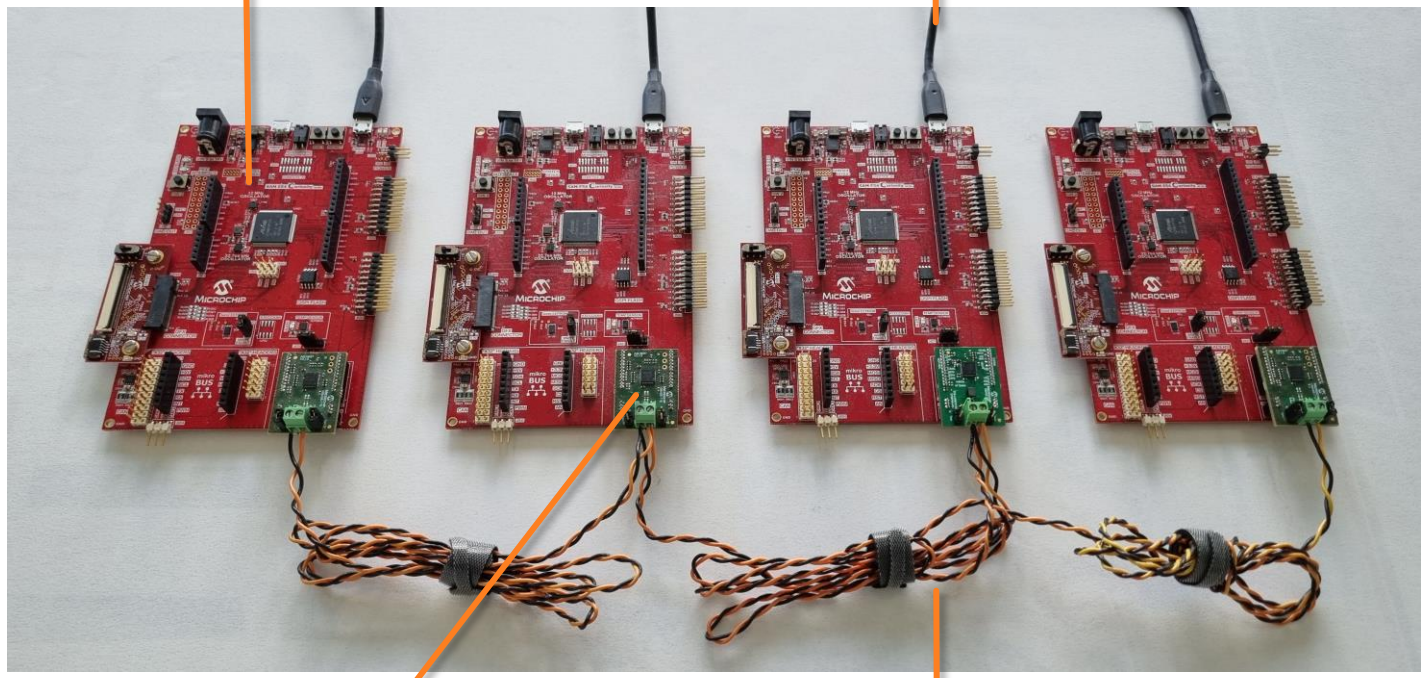
# Demo System: 10BASE-T1S



# Demo Setup

SAME54 Eval board

USB Connection to PC

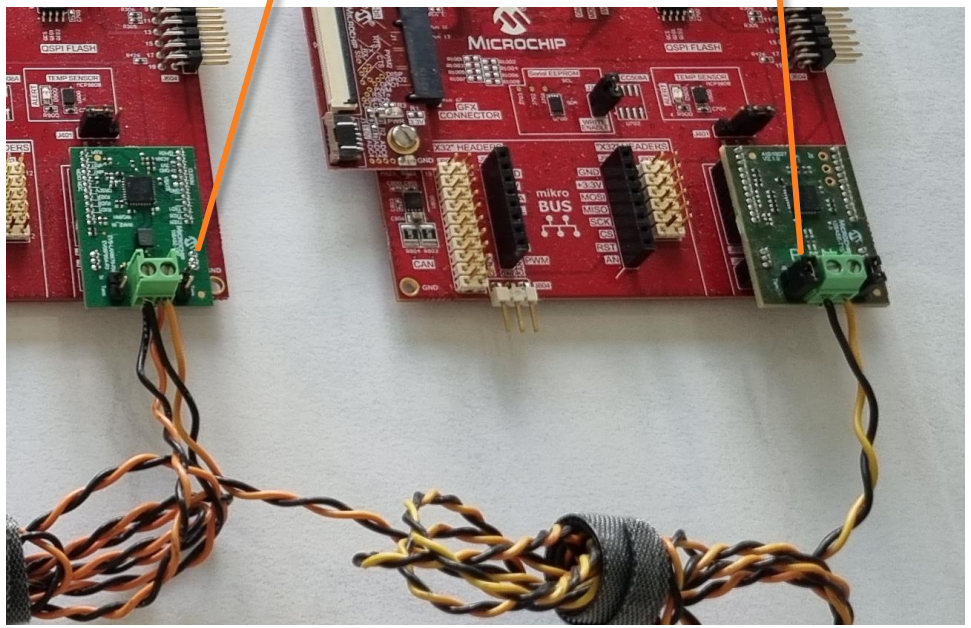


LAN8670 PHY

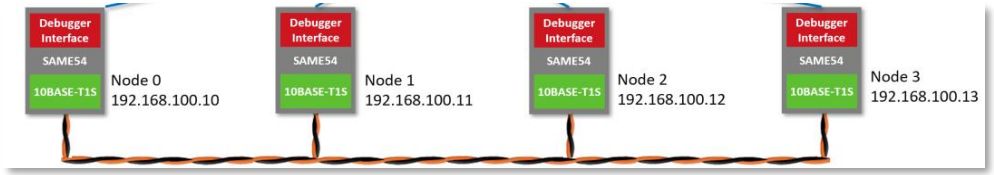
10BASE-T1S Bus

No Termination, no jumpers at drop nodes

Termination at end nodes



# Ping Statistics




COM13 - Tera Term VT

File Edit Setup Control Window Help

```
>
>
>ping 192.168.100.10
>Ping: reply[1] from 192.168.100.10: time = 1ms
Ping: reply[2] from 192.168.100.10: time = 1ms
Ping: reply[3] from 192.168.100.10: time = 1ms
Ping: reply[4] from 192.168.100.10: time = 1ms
Ping: done. Sent 4 requests, received 4 replies.

>ping 192.168.100.10 s 1200
>Ping: reply[1] from 192.168.100.10: time = 2ms
Ping: reply[2] from 192.168.100.10: time = 3ms
Ping: reply[3] from 192.168.100.10: time = 2ms
Ping: reply[4] from 192.168.100.10: time = 3ms
Ping: done. Sent 4 requests, received 4 replies.
```

COM9 - Tera Term VT

File Edit Setup Control Window Help

```
>ping 192.168.100.10 n 50 t 5
>Ping: reply[1] from 192.168.100.10: time = 1ms
Ping: reply[2] from 192.168.100.10: time = 1ms
Ping: reply[3] from 192.168.100.10: time = 1ms
Ping: reply[4] from 192.168.100.10: time = 1ms
Ping: reply[5] from 192.168.100.10: time = 1ms
Ping: reply[6] from 192.168.100.10: time = 1ms
Ping: reply[7] from 192.168.100.10: time = 1ms
Ping: reply[8] from 192.168.100.10: time = 1ms
Ping: reply[9] from 192.168.100.10: time = 1ms
Ping: reply[10] from 192.168.100.10: time = 1ms
```

- Round trip time = from sending ECHO to receiving ECHO-REPLY
- Normal round trip time in an idle network < 1ms

- Round trip time is larger because ECHO/REPLY packets are large (1200 bytes + headers)
- Ethernet frame of 1500 Bytes takes 1.2 ms

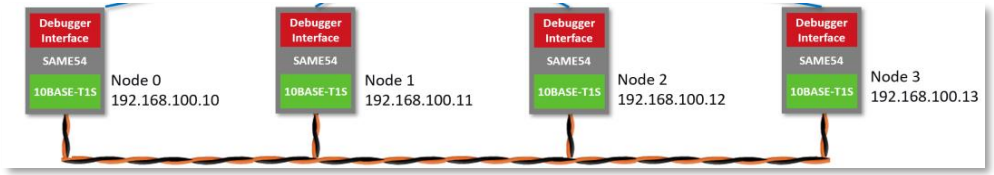
- Round trip time is small even when ECHO/REPLY packets are sent every 5 ms, only one connection on the bus





MICROCHIP

# Ping Statistics




COM13 - Tera Term VT

File Edit Setup Control Window Help

```
Ping: reply[89] from 192.168.100.11: time = 1ms
Ping: reply[90] from 192.168.100.11: time = 1ms
Ping: reply[91] from 192.168.100.11: time = 1ms
Ping: reply[92] from 192.168.100.11: time = 1ms
Ping: reply[93] from 192.168.100.11: time = 1ms
Ping: reply[94] from 192.168.100.11: time = 1ms
Ping: reply[95] from 192.168.100.11: time = 1ms
Ping: reply[96] from 192.168.100.11: time = 1ms
Ping: reply[97] from 192.168.100.11: time = 1ms
Ping: reply[98] from 192.168.100.11: time = 1ms
Ping: reply[99] from 192.168.100.11: time = 1ms
Ping: reply[100] from 192.168.100.11: time = 1ms
Ping: done. Sent 100 requests, received 100 replies.

>
>ping 192.168.100.11 n 100 t 20 s 500
```

COM9 - Tera Term VT

File Edit Setup Control Window Help

```
Ping: reply[93] from 192.168.100.10: time = 1ms
Ping: reply[94] from 192.168.100.10: time = 1ms
Ping: reply[95] from 192.168.100.10: time = 1ms
Ping: reply[96] from 192.168.100.10: time = 1ms
Ping: reply[97] from 192.168.100.10: time = 1ms
Ping: reply[98] from 192.168.100.10: time = 1ms
Ping: reply[99] from 192.168.100.10: time = 1ms
Ping: reply[100] from 192.168.100.10: time = 1ms
Ping: done. Sent 100 requests, received 100 replies.

>
>ping 192.168.100.10 n 100 t 20 s 500
```

- Round trip time is small (1 ms) even when ECHO/REPLY packets are sent every 20 ms, having 500 Bytes payload, two connections

# Iperf Basics

- **Iperf client** sends UDP-packets (or TCP-packets to an iperf server
- Iperf client tries to use as much bandwidth as possible (or as configured)
- Iperf client shows TX statistics

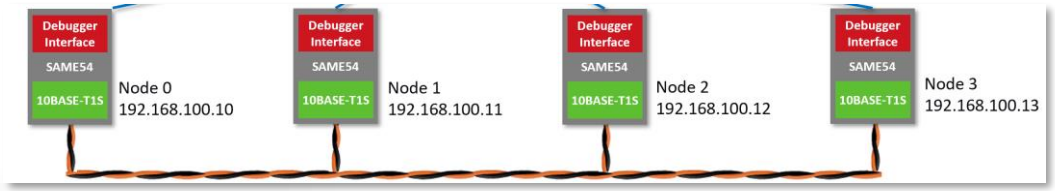
- iperf -c <target server IP>
  - -u traffic over UDP
  - -l <number of UDP data bytes>
  - -t 0 sending forever

- ***iperf -c 192.168.100.33 -u -l 1400 -t 0***
  client sends UDP packets to the address 192.168.100.33, having a length of 1400 bytes UDP payload plus headers.
- Sending is stopped by ***iperfk***

- **Iperf Server** receives UDP or TCP packets from an iperf client.
- Iperf server shows RX statistics
- ... Many more features...

- ***Iperf -s -u***
  starts an iperf server which listens on UDP

# Iperf Bandwidth




COM11 - Tera Term VT — **Node 0**

File Edit Setup Control Window Help

```
IPv4 Address: 192.168.100.10
Mask: 255.255.255.0
Gateway: 192.168.100.1
DNS1: 192.168.100.1
DNS2: 0.0.0.0
MAC Address: 00:04:25:1c:a0:10
default IP address is ON
dhcp is enabled
Link is UP
Status: Ready
>iperf -s -u

iperf: Starting session instance 0
------------------------------------------------------------
iperf: Server listening on UDP port 5001
>
iperf: instance 0 session started ...
    - Local  192.168.100.10 port 5001 connected with
    - Remote 192.168.100.11 port 61875
    - [ 0-  1 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 1-  2 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 2-  3 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 3-  4 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 4-  5 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 5-  6 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 6-  7 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 7-  8 sec]   0/ 802 ( 0%)    9422 Kbps
    - [ 8-  9 sec]   0/ 803 ( 0%)    9434 Kbps
    - [ 9- 10 sec]   0/ 802 ( 0%)    9422 Kbps
    - [0.0- 10.0 sec]   0/ 8037 ( 0%)    9424 Kbps
iperf: instance 0 completed ...
iperf instance 0: Rx done. Socket closed.
iperf instance 0: Ready for the next session.
```

Node0 runs iperf server

COM14 - Tera Term VT — **Node 1**

File Edit Setup Control Window Help

```
>iperf -c 192.168.100.10 -u

iperf: Starting session instance 0
iperf: Using the default interface!
>    - RemoteNode MAC: 0 4 25 1c a0 10
------------------------------------------------------------
iperf: Client connecting to 192.168.100.10, UDP port 5001

iperf: instance 0 started ...
    - Local  192.168.100.11 port 61875 connected with
    - Remote 192.168.100.10 port 5001
    - Target rate = 10000000 bps, period = 1 ms
    - [ 0-  1 sec]   0/ 805 ( 0%)    9467 Kbps
    - [ 1-  2 sec]   0/ 804 ( 0%)    9446 Kbps
    - [ 2-  3 sec]   0/ 804 ( 0%)    9455 Kbps
    - [ 3-  4 sec]   0/ 804 ( 0%)    9446 Kbps
    - [ 4-  5 sec]   0/ 804 ( 0%)    9455 Kbps
    - [ 5-  6 sec]   0/ 804 ( 0%)    9446 Kbps
    - [ 6-  7 sec]   0/ 804 ( 0%)    9455 Kbps
    - [ 7-  8 sec]   0/ 803 ( 0%)    9443 Kbps
    - [ 8-  9 sec]   0/ 804 ( 0%)    9455 Kbps
    - [0.0- 10.0 sec]   0/ 8037 ( 0%)    9452 Kbps
    ------------------------------------------
    - [0.0- 10.1 sec]   0/ 8046 ( 0%)    9358 Kbps
iperf: instance 0 completed ...iperf instance 0: Tx done.
Socket closed.
iperf: instance 0 completed.
```

- Node1 runs iperf client over UDP
- lost / total datagrams
- bandwidth used by this connection
- Only one transmission on the network: node1 => node0
- iperf client on node1 can use the full bandwidth.
- iperf counts only the UDP payload to calculate the BW: ~9.4 Mbits/sec

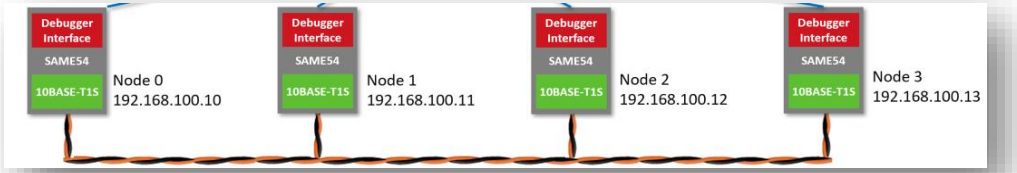




MICROCHIP

# Iperf Bandwidth

## Two connections, same packet length




COM14 - Tera Term VT — **Node 1**

File Edit Setup Control Window Help

```
- Local  192.168.100.11 port 61288 connected with
- Remote 192.168.100.10 port 5001
- Target rate = 10000000 bps, period = 1 ms
- [ 0-  1 sec]   0/ 805 ( 0%)    9467 Kbps
- [ 1-  2 sec]   0/ 804 ( 0%)    9455 Kbps
- [ 2-  3 sec]   0/ 803 ( 0%)    9443 Kbps
- [ 3-  4 sec]   0/ 420 ( 0%)    4929 Kbps
- [ 4-  5 sec]   0/ 403 ( 0%)    4735 Kbps
- [ 5-  6 sec]   0/ 403 ( 0%)    4730 Kbps
- [ 6-  7 sec]   0/ 403 ( 0%)    4730 Kbps
- [ 7-  8 sec]   0/ 403 ( 0%)    4730 Kbps
- [ 8-  9 sec]   0/ 403 ( 0%)    4730 Kbps
- [0.0- 10.0 sec]   0/ 5245 ( 0%)    6168 Kbps
```

COM9 - Tera Term VT — **Node 3**

File Edit Setup Control Window Help

```
- Local  192.168.100.13 port 49639 connected with
- Remote 192.168.100.12 port 5001
- Target rate = 10000000 bps, period = 1 ms
- [ 0-  1 sec]   0/ 403 ( 0%)    4739 Kbps
- [ 1-  2 sec]   0/ 401 ( 0%)    4716 Kbps
- [ 2-  3 sec]   0/ 401 ( 0%)    4716 Kbps
- [ 3-  4 sec]   0/ 401 ( 0%)    4716 Kbps
- [ 4-  5 sec]   0/ 401 ( 0%)    4716 Kbps
```

- Iperf-server running on node0 (192.168.100.10) and node2
- Node1 already sends to node0
- As soon as node3 starts sending to node2 as well, the bandwidth for each of the two connections goes down to ~4.7 Mbits/sec
- The bandwidth and number of frames per second is shared evenly among the two transmissions, that is between the two senders (clients)





MICROCHIP

# Iperf Bandwidth

## Two connections, different packet lengths


COM13 - Tera Term VT — **Node 2**

File Edit Setup Control Window Help

```
    - [40- 41 sec]   0/ 1147 ( 0%)    9176 Kbps
    - [41- 42 sec]   0/ 1147 ( 0%)    9176 Kbps
    - [42- 43 sec]   0/ 1147 ( 0%)    9176 Kbps
    - [43- 44 sec]   0/ 1147 ( 0%)    9176 Kbps
    - [44- 45 sec]   0/ 1147 ( 0%)    9176 Kbps
    - [45- 46 sec]   0/ 773 ( 0%)     6178 Kbps
    - [46- 47 sec]   0/ 741 ( 0%)     5928 Kbps
    - [47- 48 sec]   0/ 741 ( 0%)     5928 Kbps
    - [48- 49 sec]   0/ 741 ( 0%)     5928 Kbps
    - [49- 50 sec]   0/ 741 ( 0%)     5928 Kbps
    - [50- 51 sec]   0/ 741 ( 0%)     5928 Kbps
    - [51- 52 sec]   0/ 741 ( 0%)     5928 Kbps
    - [52- 53 sec]   0/ 1121 ( 0%)    8968 Kbps
    - [53- 54 sec]   0/ 1147 ( 0%)    9176 Kbps
    - [54- 55 sec]   0/ 1147 ( 0%)    9176 Kbps
    - [55- 56 sec]   0/ 1147 ( 0%)    9176 Kbps
```

COM9 - Tera Term VT — **Node 3**

File Edit Setup Control Window Help

```
iperf: instance 0 started ...
    - Local  192.168.100.13 port 51019 connected with
    - Remote 192.168.100.10 port 5001
    - Target rate = 10000000 bps, period = 0 ms
    - [ 0-  1 sec]   0/ 743 ( 0%)     2969 Kbps
    - [ 1-  2 sec]   0/ 741 ( 0%)     2964 Kbps
    - [ 2-  3 sec]   0/ 741 ( 0%)     2964 Kbps
    - [ 3-  4 sec]   0/ 741 ( 0%)     2964 Kbps
i    - [ 4-  5 sec]   0/ 741 ( 0%)     2964 Kbps
per    - [ 5-  6 sec]   0/ 741 ( 0%)     2964 Kbps
fk
```

- Node2 sends to node0,
  using a packet length of 1000 Bytes
- iperf -c 192.168.100.10 -u -l 1000 -t 0
- More packets/s but less bandwidth compared to larger packets because of more UDP headers

- Node3 starts sending as well,
  using a packet length of 500 Bytes only.

- With PLCA the available transmit opportunities are shared evenly among the two transmissions.
  ~ 740 packets/sec
- However, the bandwidth is shared
  in the proportion of the packet lengths,
  2:1 in this example





MICROCHIP

# 24037 NET1
# 10BASE-T1S Lab

# Lab: Learning Objectives

## What will our clients be able to do when they finish the activity?

- Use MPLAB® Harmony to setup and configure a small 10BASE-T1S network.
- Reconfigure your application's network settings and understand their impact on the communication in 10BASE-T1S networks.
- Program your own application onto a microcontroller evaluation board to establish 10BASE-T1S network communication.
- Build an application on top of your project to add an OLED display to show the network and PLCA parameters
- Extend the physical network, that is the UTP bus line, without the need for any switches or repeaters.

# LAB Part 1: 10BASE-T1S using MPLAB® Harmony

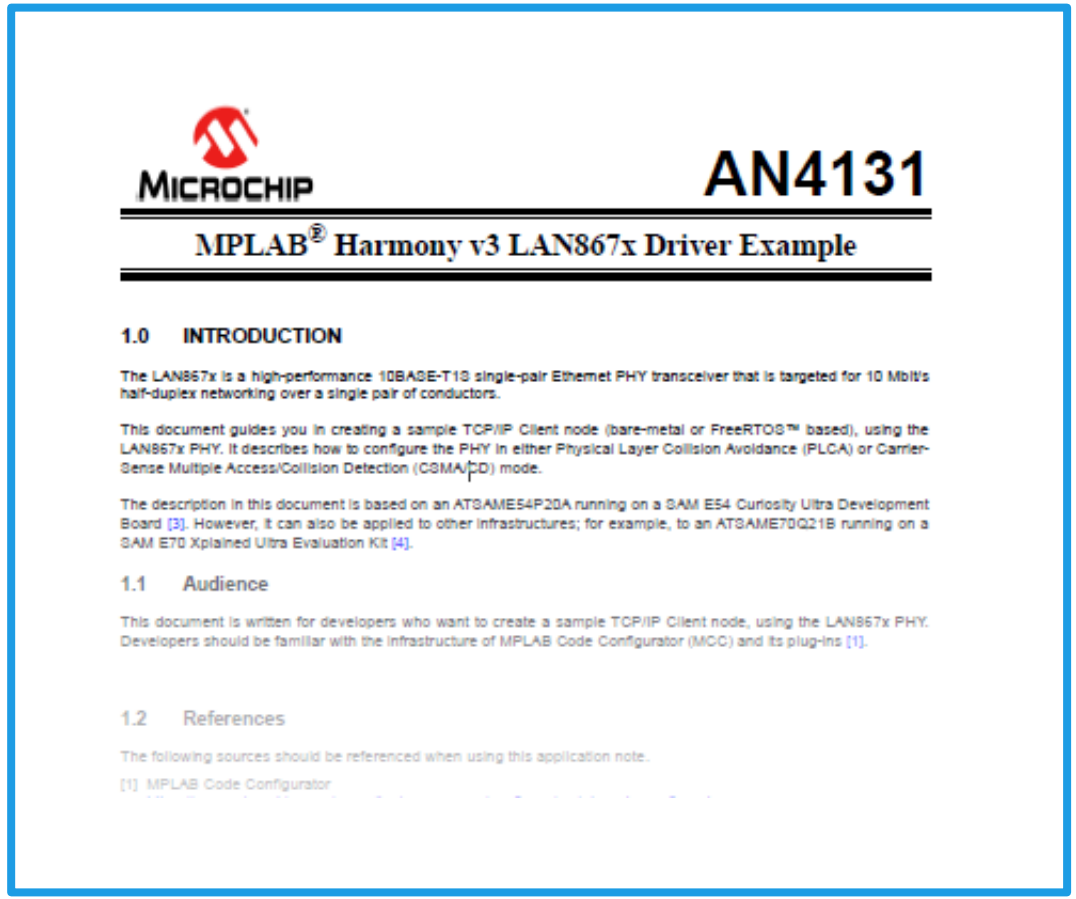
## Project Creation

# LAB: Learning Objectives

- **Use MPLAB® Harmony to setup and configure a small 10BASE-T1S network.**
- **Reconfigure your application's network settings and understand their impact on the communication in 10BASE-T1S networks.**
- **Program your own application onto a microcontroller evaluation board to establish 10BASE-T1S network communication.**
- **Build an application on top of your project to add an OLED display to show the network and PLCA parameters**
- **Extend the physical network, that is the UTP bus line, without the need for any switches or repeaters.**

# Goal of this Hands-On Training

- Present a demonstration of an example of Single Pair Ethernet communication over 10BASE-T1S
- Use LAN8670 RMII evaluation board with SAME54 Curiosity Ultra
- Based on Application Note AN4131



MICROCHIP

AN4131

MPLAB® Harmony v3 LAN867x Driver Example

1.0 INTRODUCTION

The LAN867x is a high-performance 10BASE-T1S single-pair Ethernet PHY transceiver that is targeted for 10 Mbit/s half-duplex networking over a single pair of conductors.

This document guides you in creating a sample TCP/IP Client node (bare-metal or FreeRTOS™ based), using the LAN867x PHY. It describes how to configure the PHY in either Physical Layer Collision Avoidance (PLCA) or Carrier-Sense Multiple Access/Collision Detection (CSMA/CD) mode.

The description in this document is based on an ATSAME54P20A running on a SAM E54 Curiosity Ultra Development Board [3]. However, it can also be applied to other infrastructures; for example, to an ATSAME70Q21B running on a SAM E70 Xplained Ultra Evaluation Kit [4].

1.1 Audience

This document is written for developers who want to create a sample TCP/IP Client node, using the LAN867x PHY. Developers should be familiar with the infrastructure of MPLAB Code Configurator (MCC) and its plug-ins [1].

1.2 References

The following sources should be referenced when using this application note.

[1] MPLAB Code Configurator

- Link to AN4131 is given here and is also available on LAN8670 Product Page





MICROCHIP

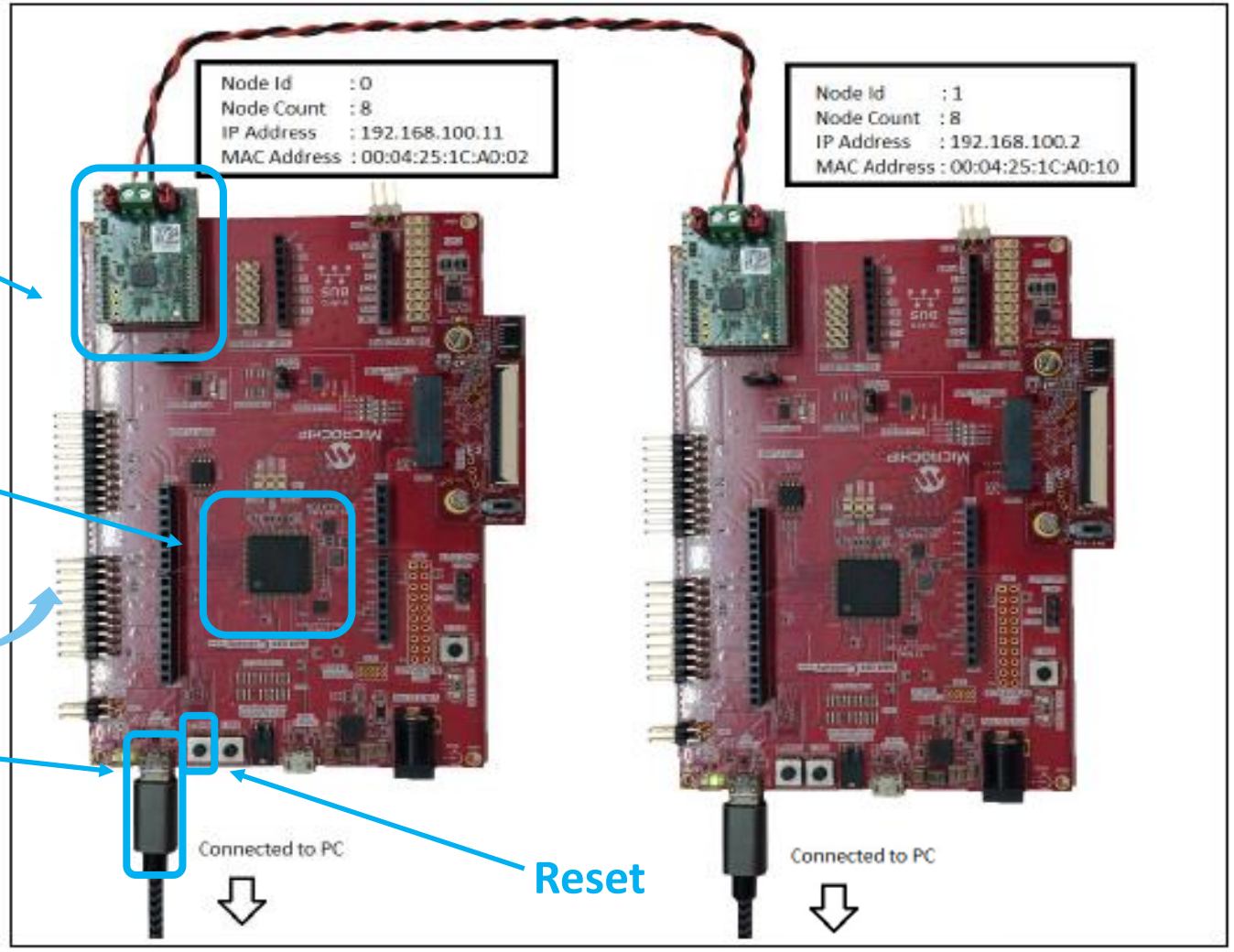
# LAN8670 RMII evaluation board with SAME54 Curiosity Ultra

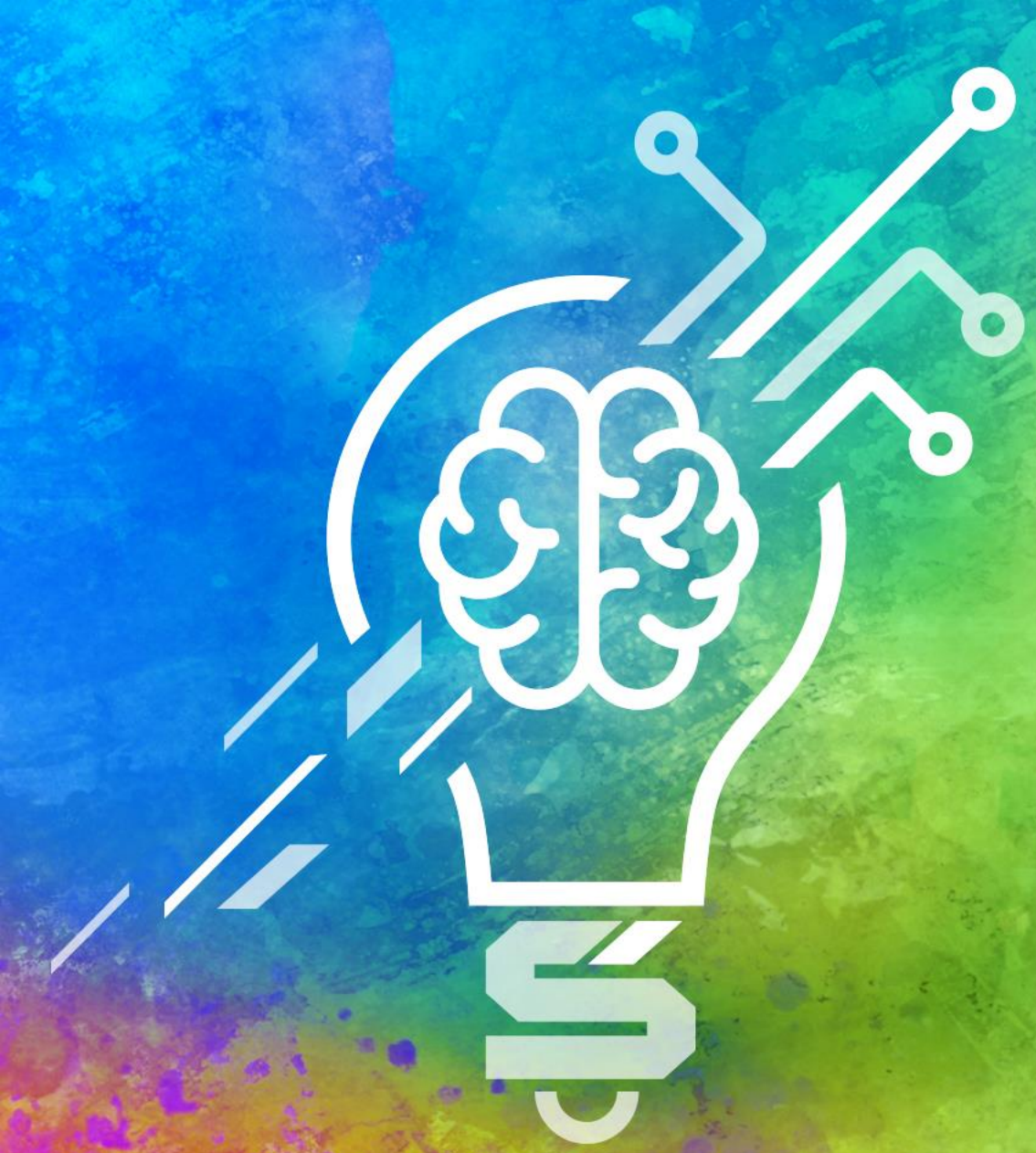
Ethernet Interface with RMII mounted with EVB-LAN8670-RMII to 10BASE-T1S interface card

ATSAME54P20A

On board EDBG debugger

USB Debugger Interface



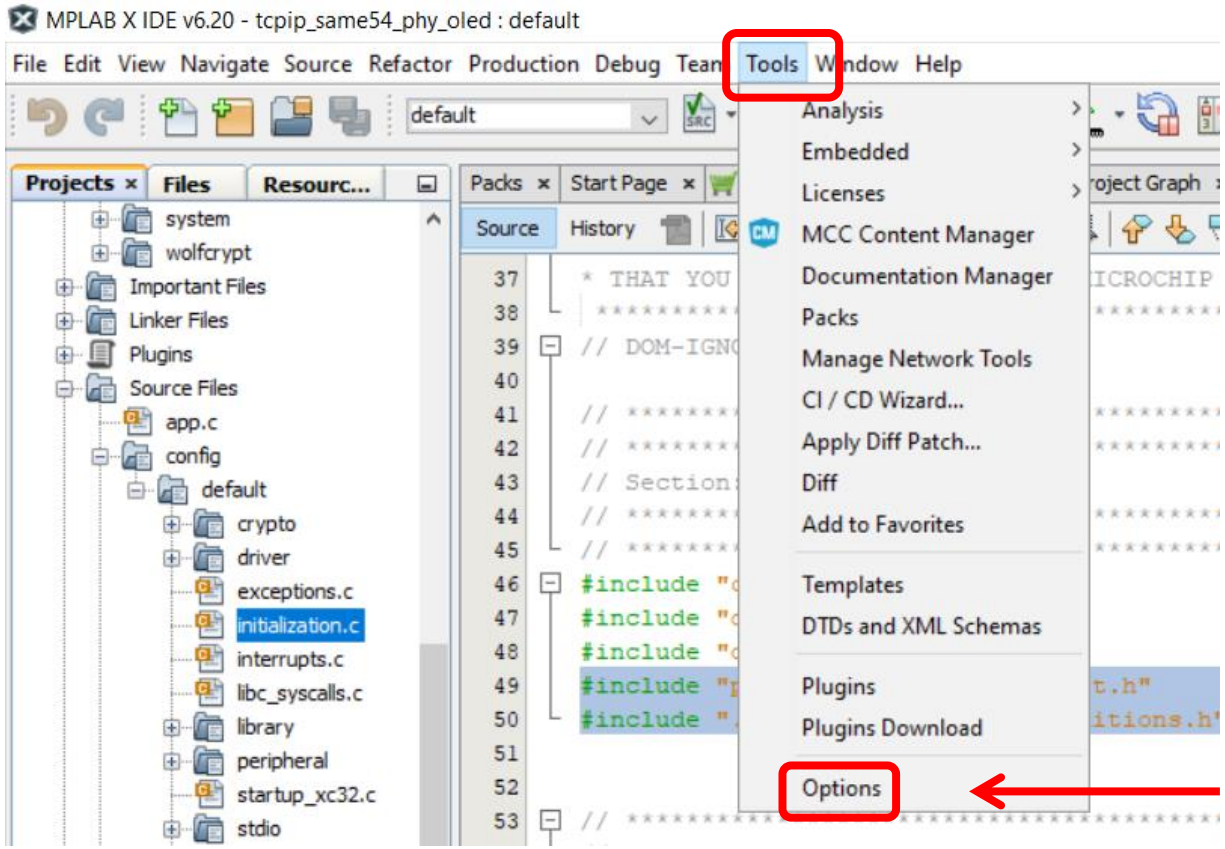


# Project Setup

## Step-by-Step Guide to Creating the Project

# Select the Harmony Repository

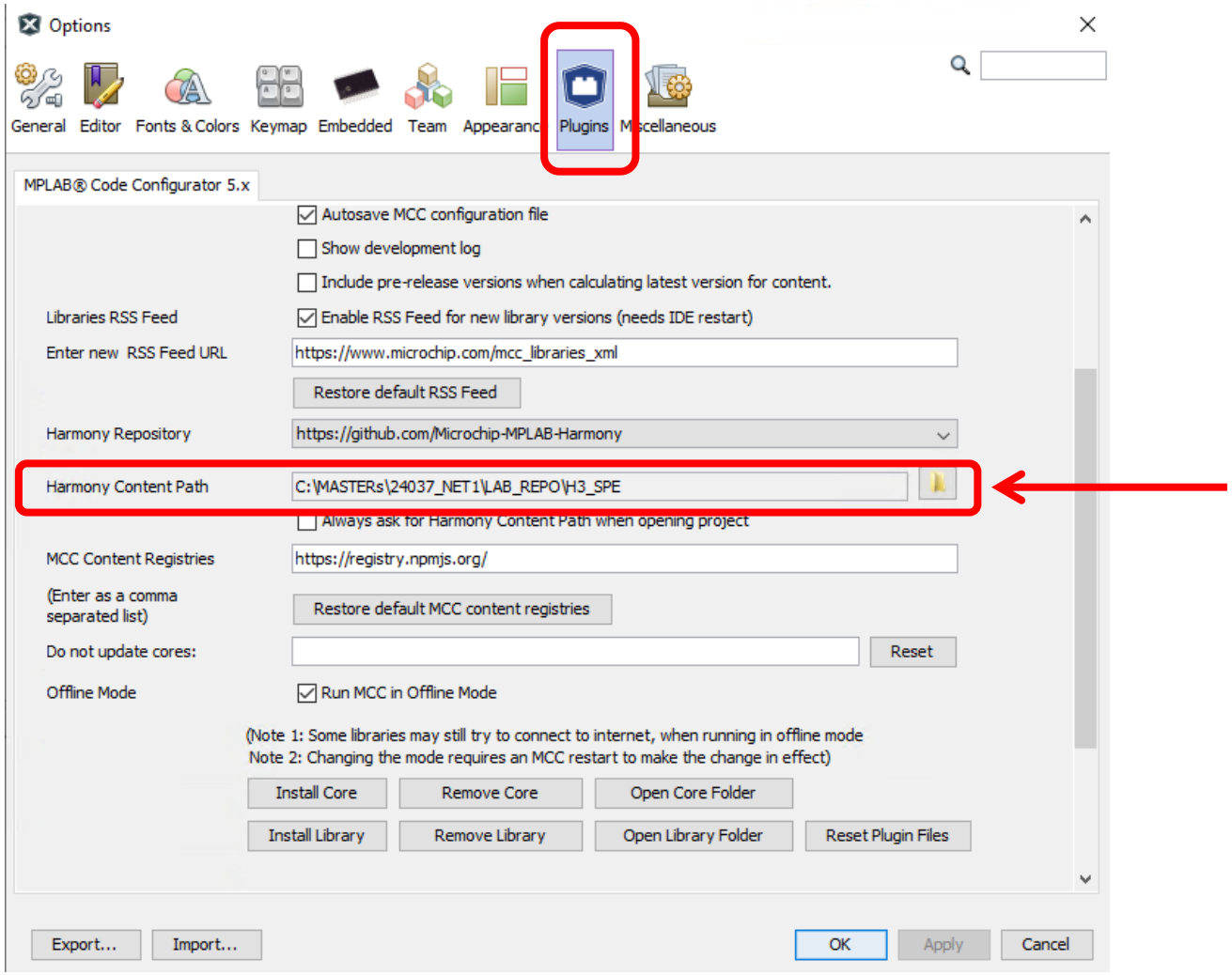
In MPLAB® X, go **Tools > Options**



# Select the Harmony Repository

Under Plugins, change the Harmony Content Path so that it is:

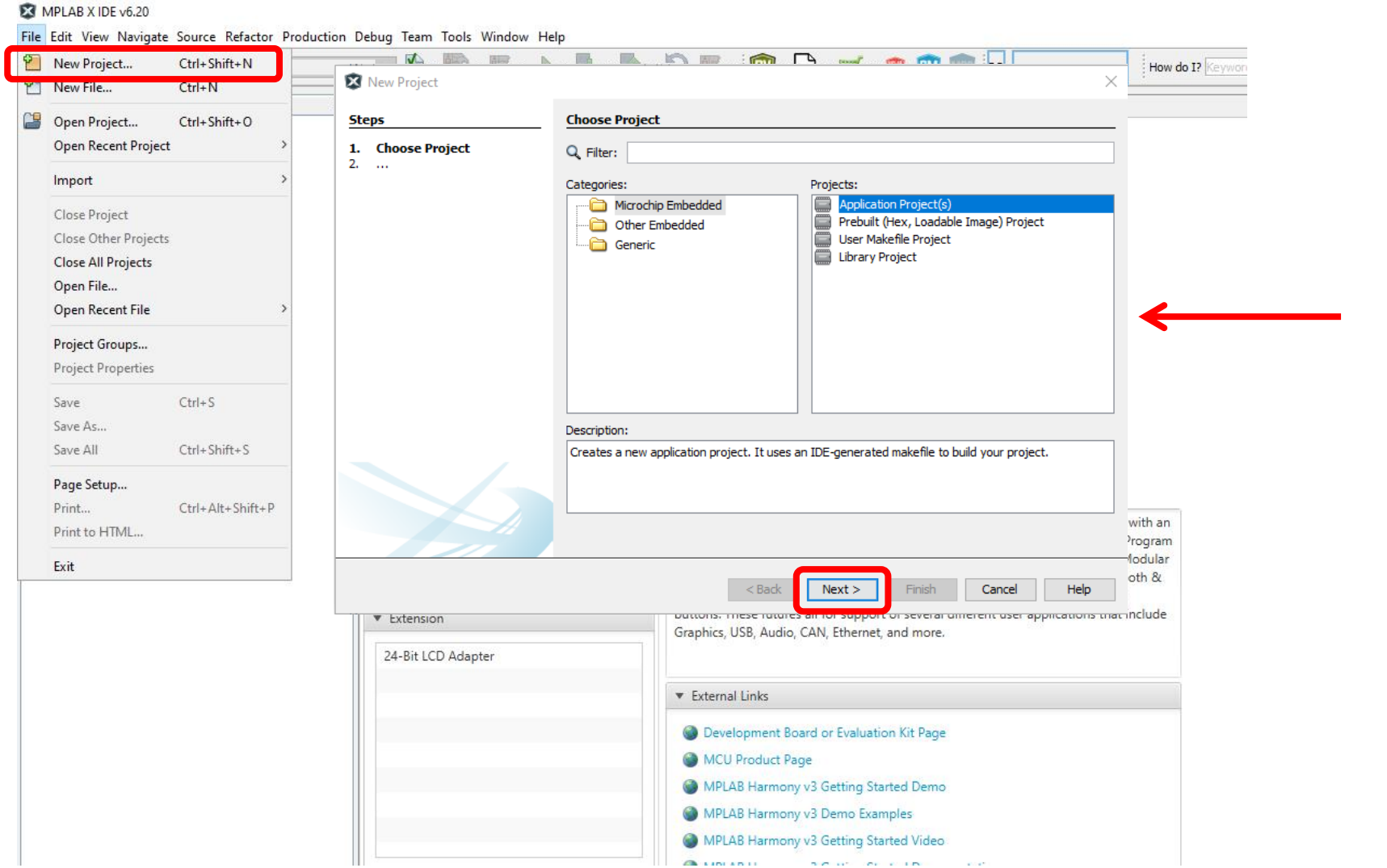
C:\MASTERs\24037_NET1\LAB_REPO\H3_SPE



# Create New Project

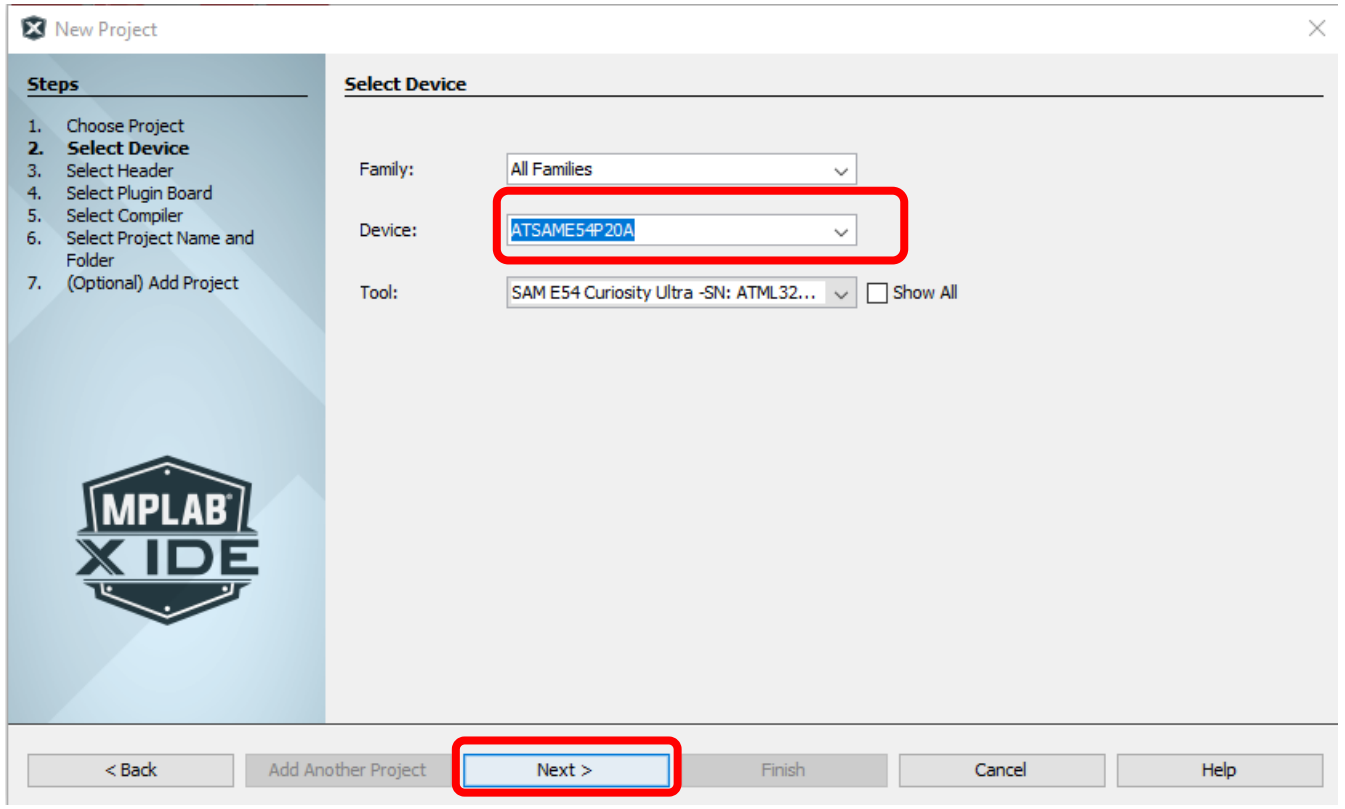
In MPLAB® X, go **File > New Project…**

Select **Application Project(s)** and click **Next>**



# Select Device and Tool

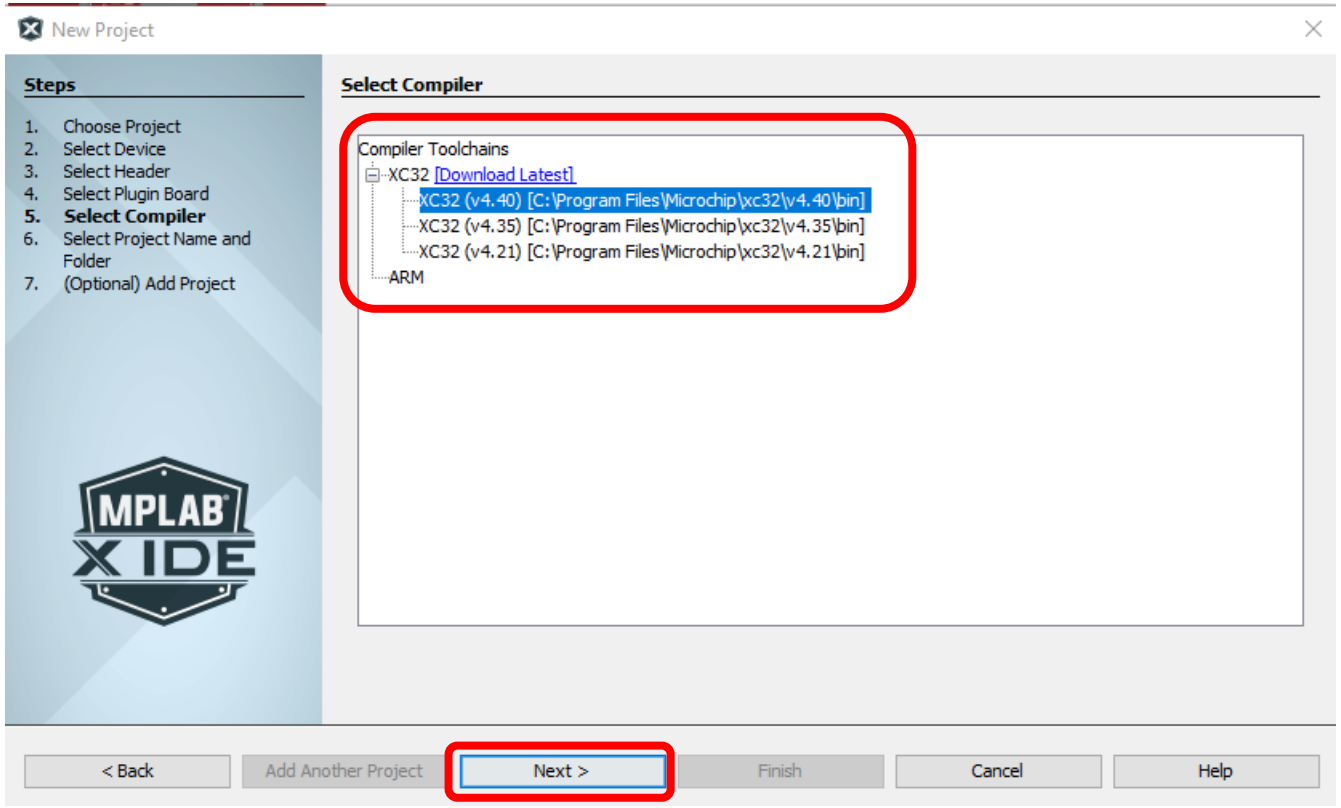
Enter the name of the device which is ATSAME54P20A and click **Next>**



# Select Compiler

Select the XC32 (v4.40) Compiler and click **Next>**

**Note:** there may be other compiler versions visible, or not, just ignore the others



# Create a Project Name

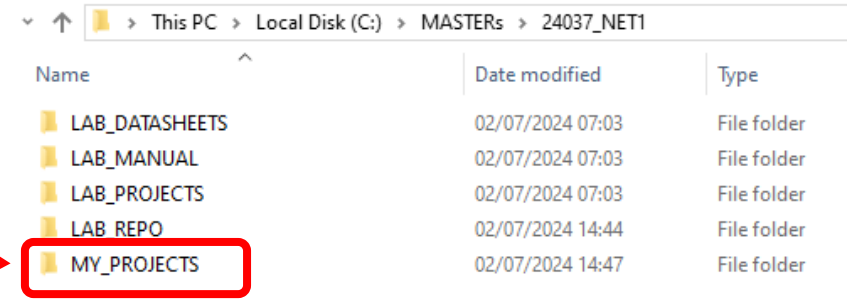
On your PC, an area has been prepared for your work. It is located in C:\MASTERs\24037_NET1 and is named “MY_PROJECTS”

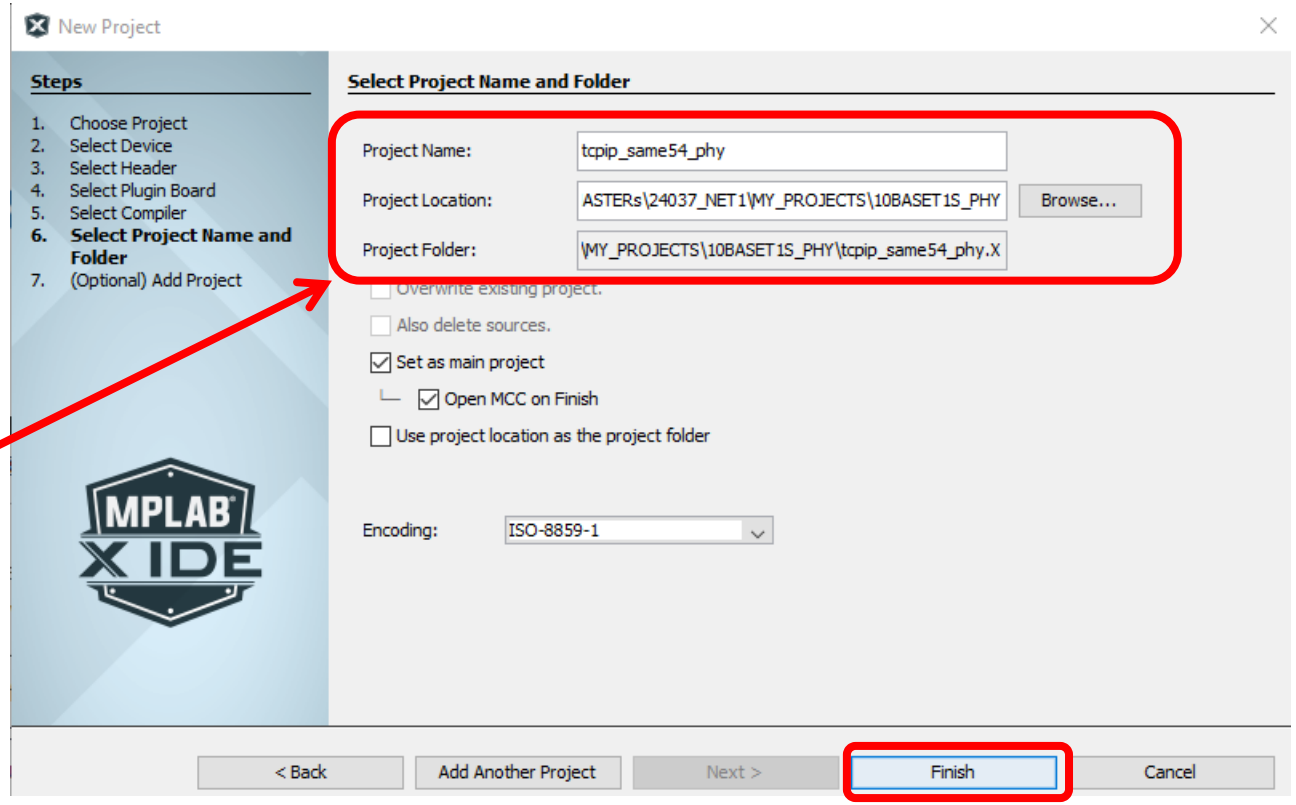
You will need to create a folder inside of MY_PROJECTS called “10BASET1S_PHY”. The project will be located in the 10BASET1S_PHY folder.

Enter a Project Name for your project – in this example we have called it “tcpip_same54_phy”

For Project Location, Browse to the 10BASET1S_PHY (or equivalent) that you have created on your PC.

Click Finish.

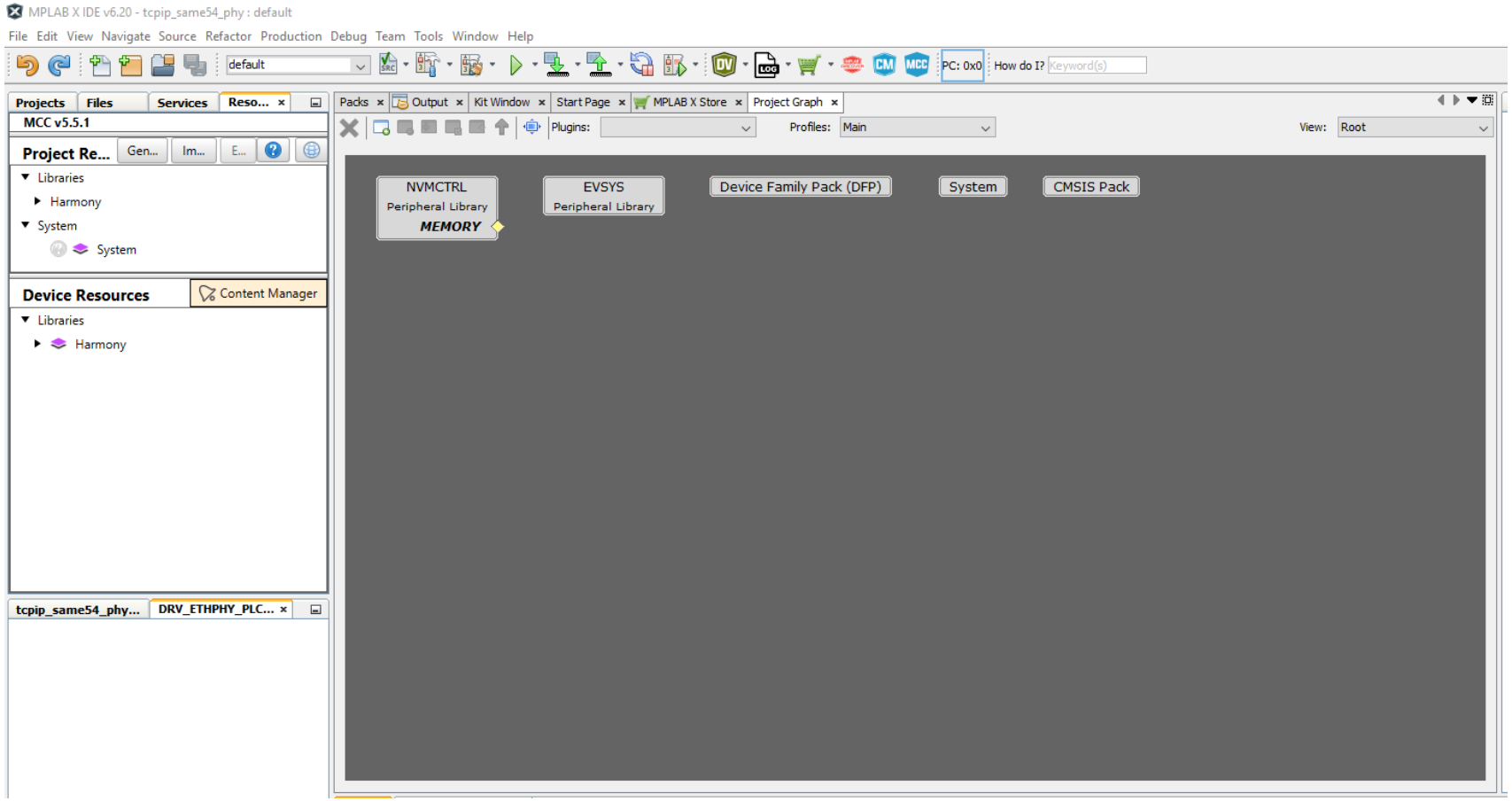


# Your Project is Created!

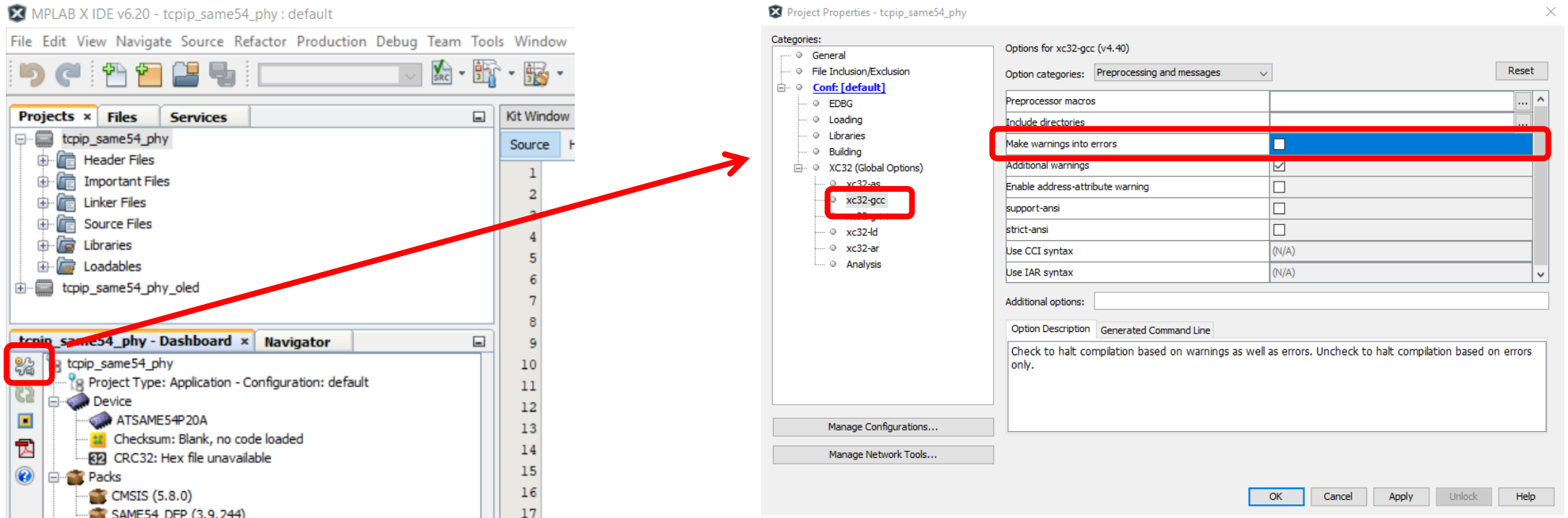
MPLAB® X opens MPLAB® Code Configurator and you can see the Project Graph

This will already be populated with some components which are essential for the device you have selected



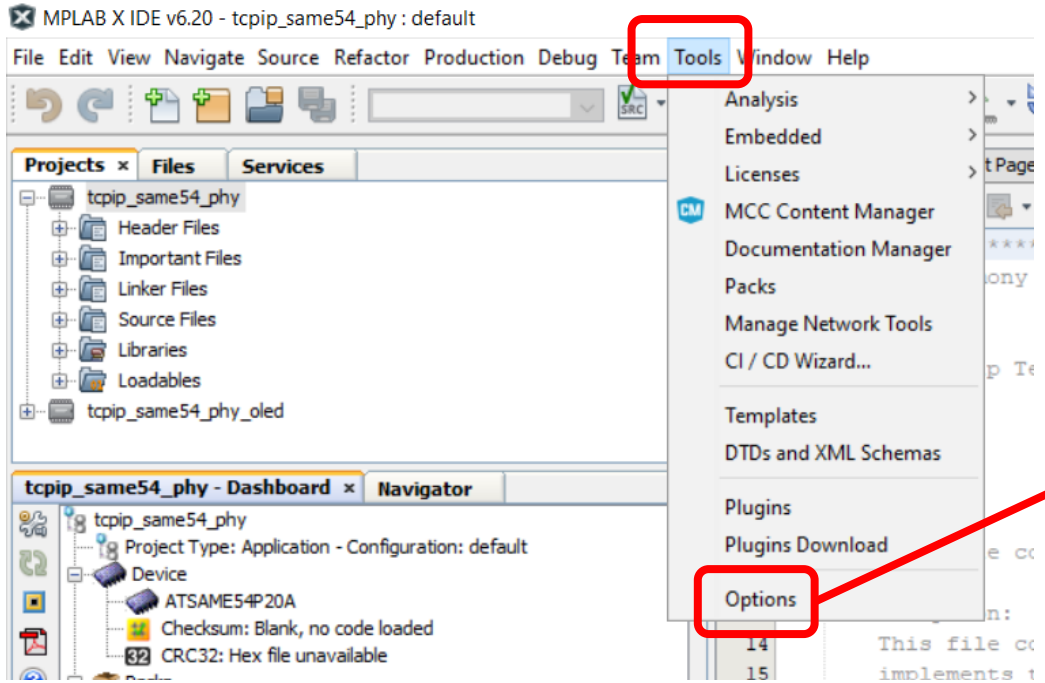
# Additional settings that could cause problems!

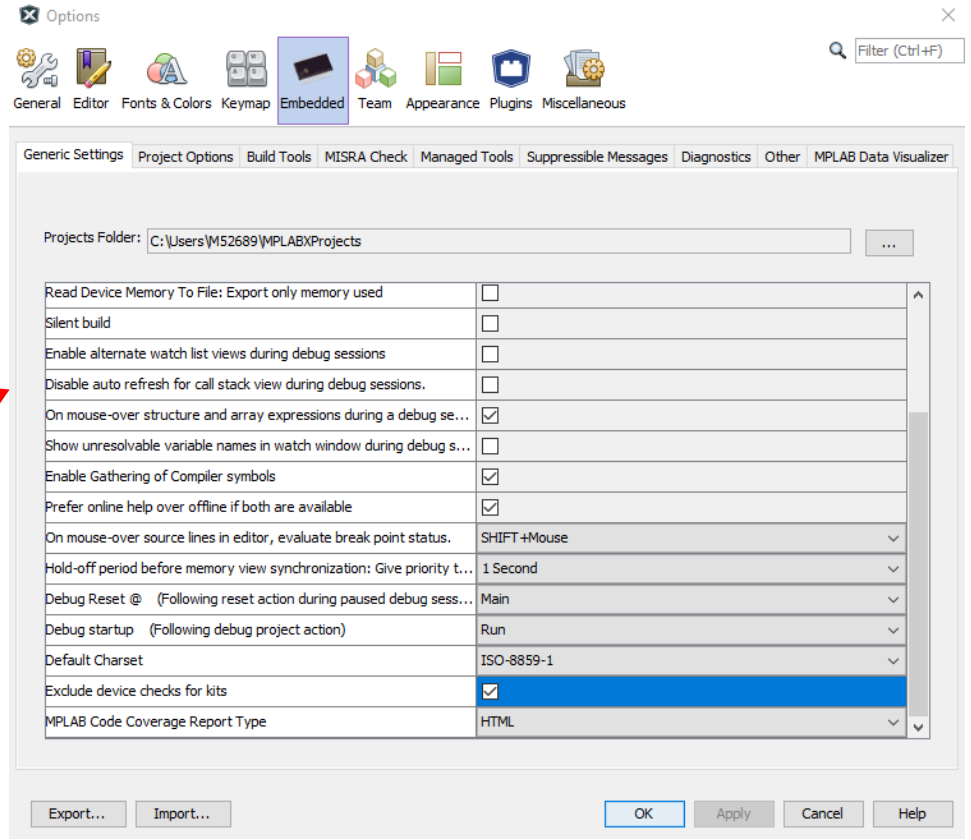
In Project Properties, under **xc32-gcc**, uncheck
“Make warnings into errors”

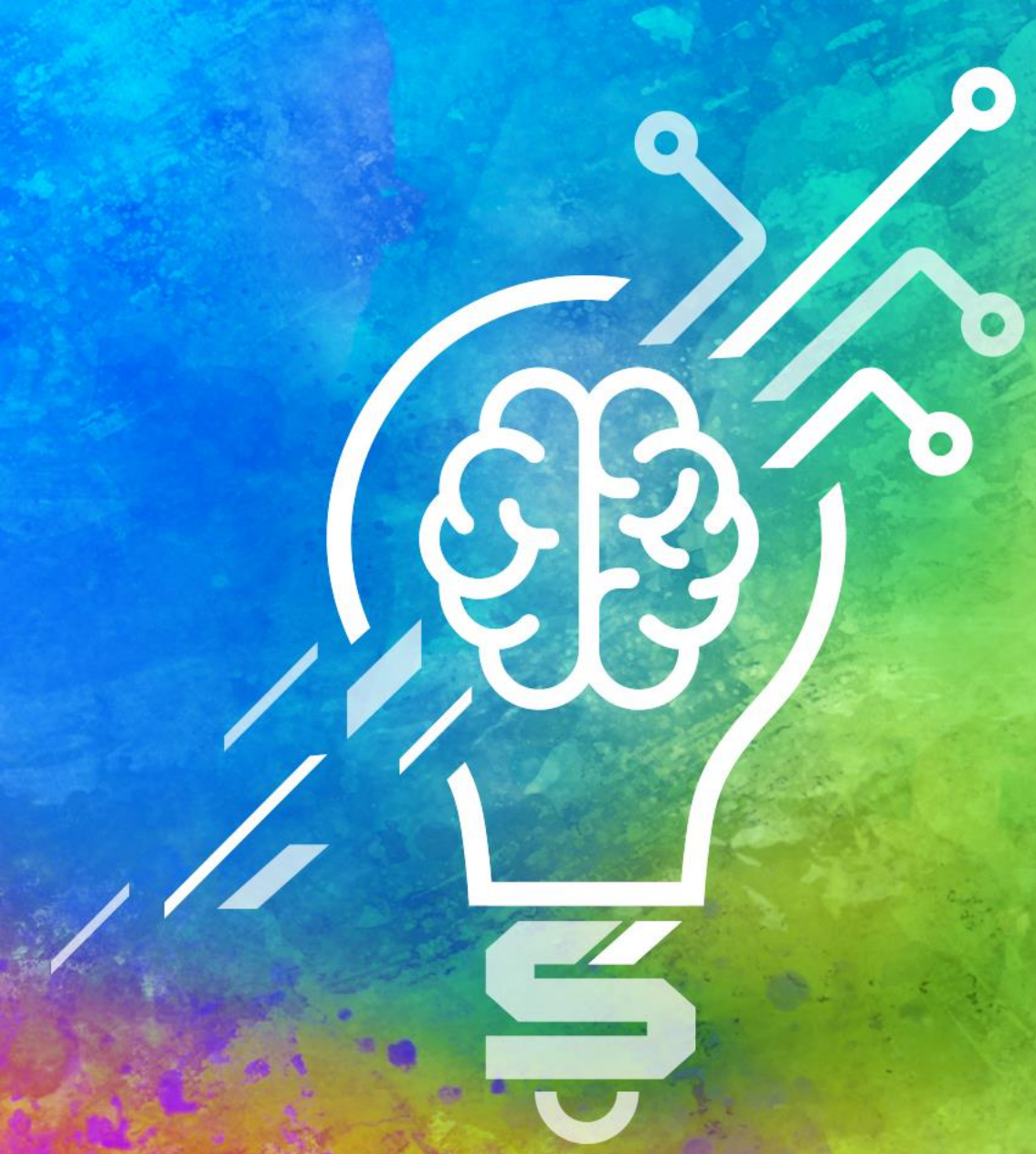


# Additional settings that could cause problems!

In Options > Embedded, check “Exclude device checks for kits”



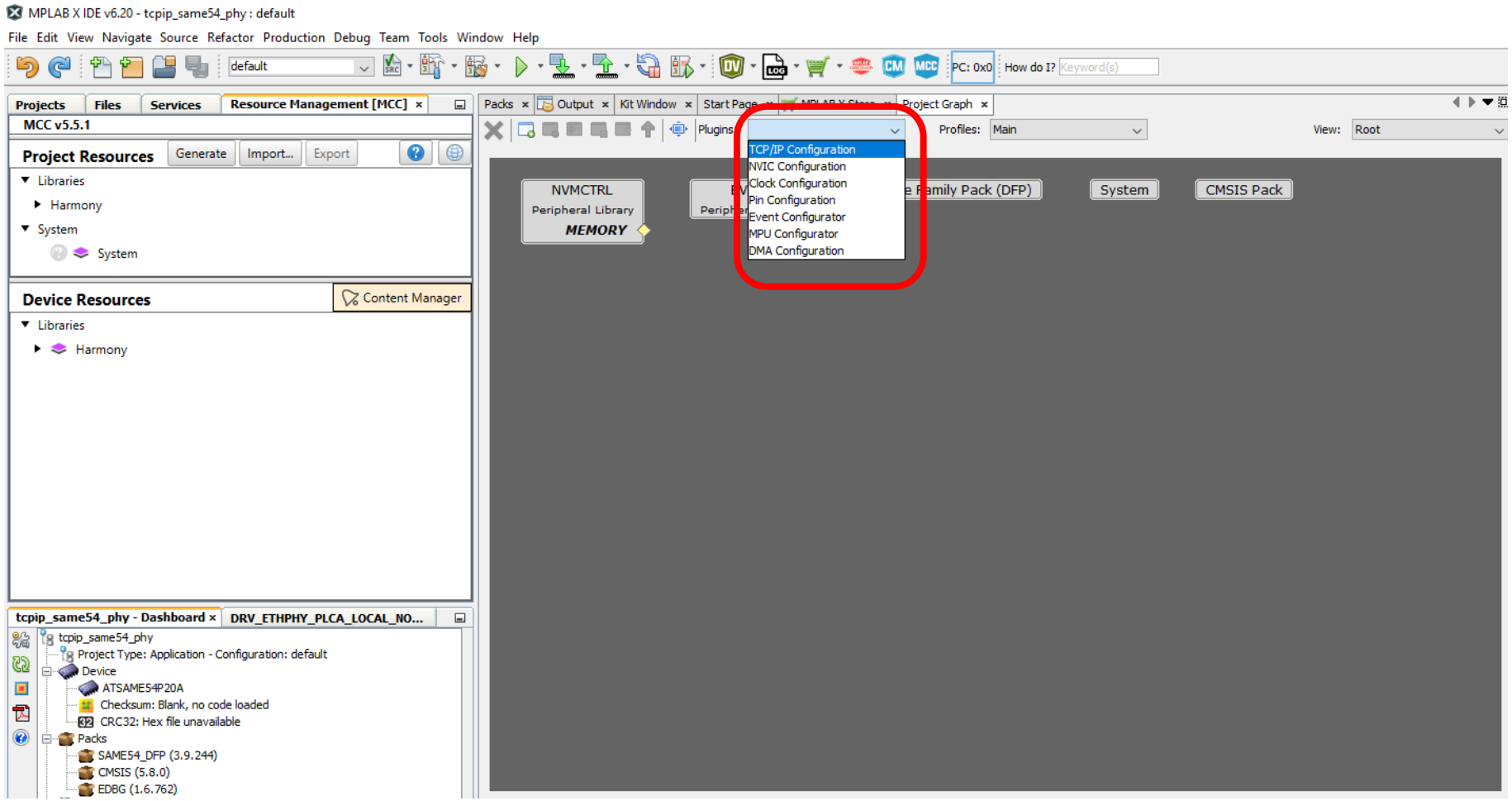





# Configuring the Project in MCC

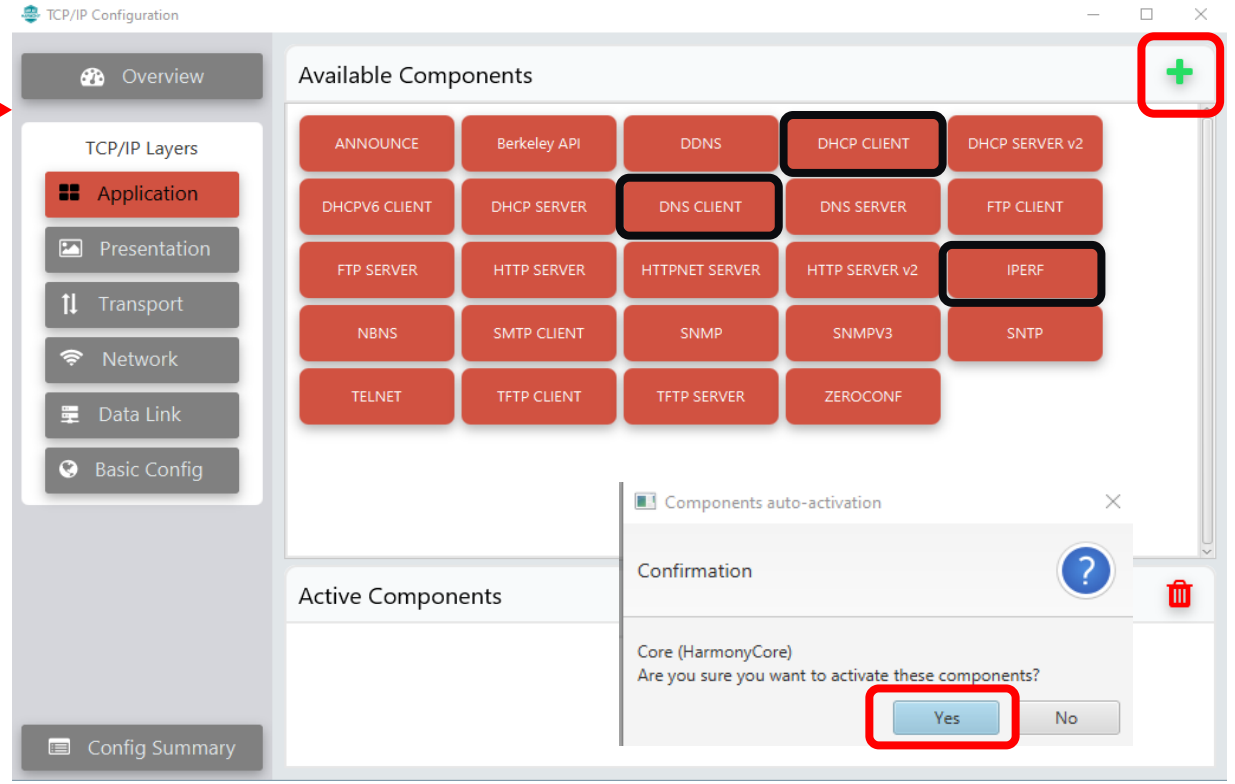
# Open MCC and the TCP/IP Configuration Tool

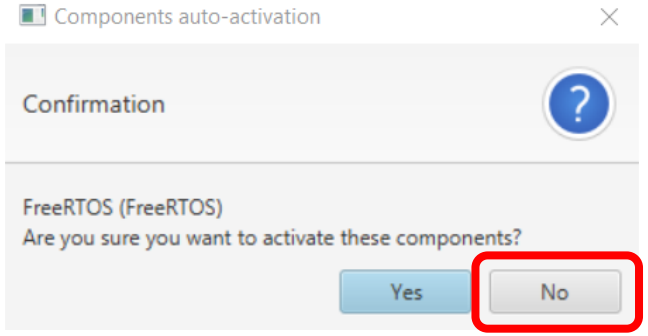
- **Start by opening the TCP/IP Configuration Plugin**



# TCP/IP: Application Layer

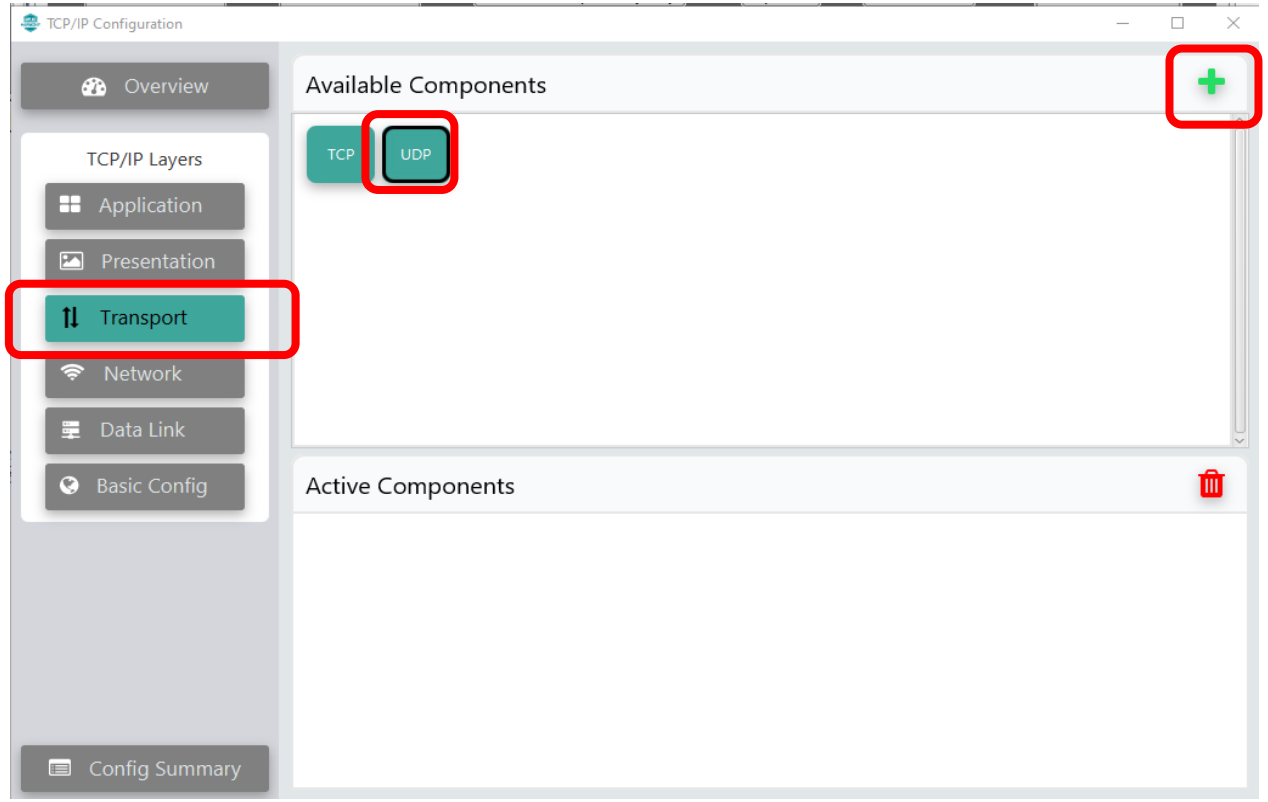
- In the TCP/IP Configuration tool, in the Application Layer, Select IPERF and click +
- Click Yes to activate the Harmony Core component, and No for FreeRTOS (not needed for this demo)
- Also add DNS CLIENT and DHCP CLIENT



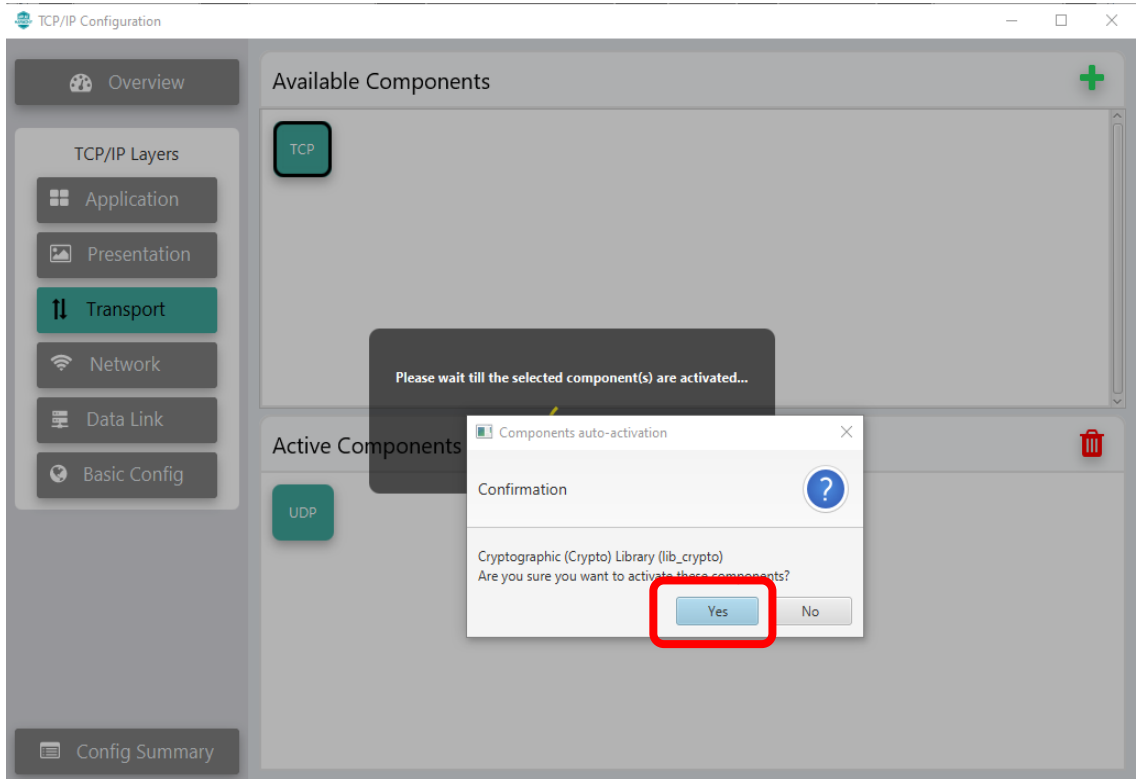
# TCP/IP: Transport Layer

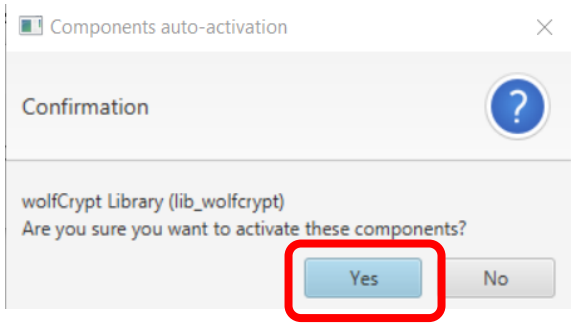
- **In the Transport Layer, add the UDP Component in the same way**



# TCP/IP: Transport Layer

- **Add the TCP Component, which needs the Crypto library and the WolfSSL libraries, click Yes to add these**

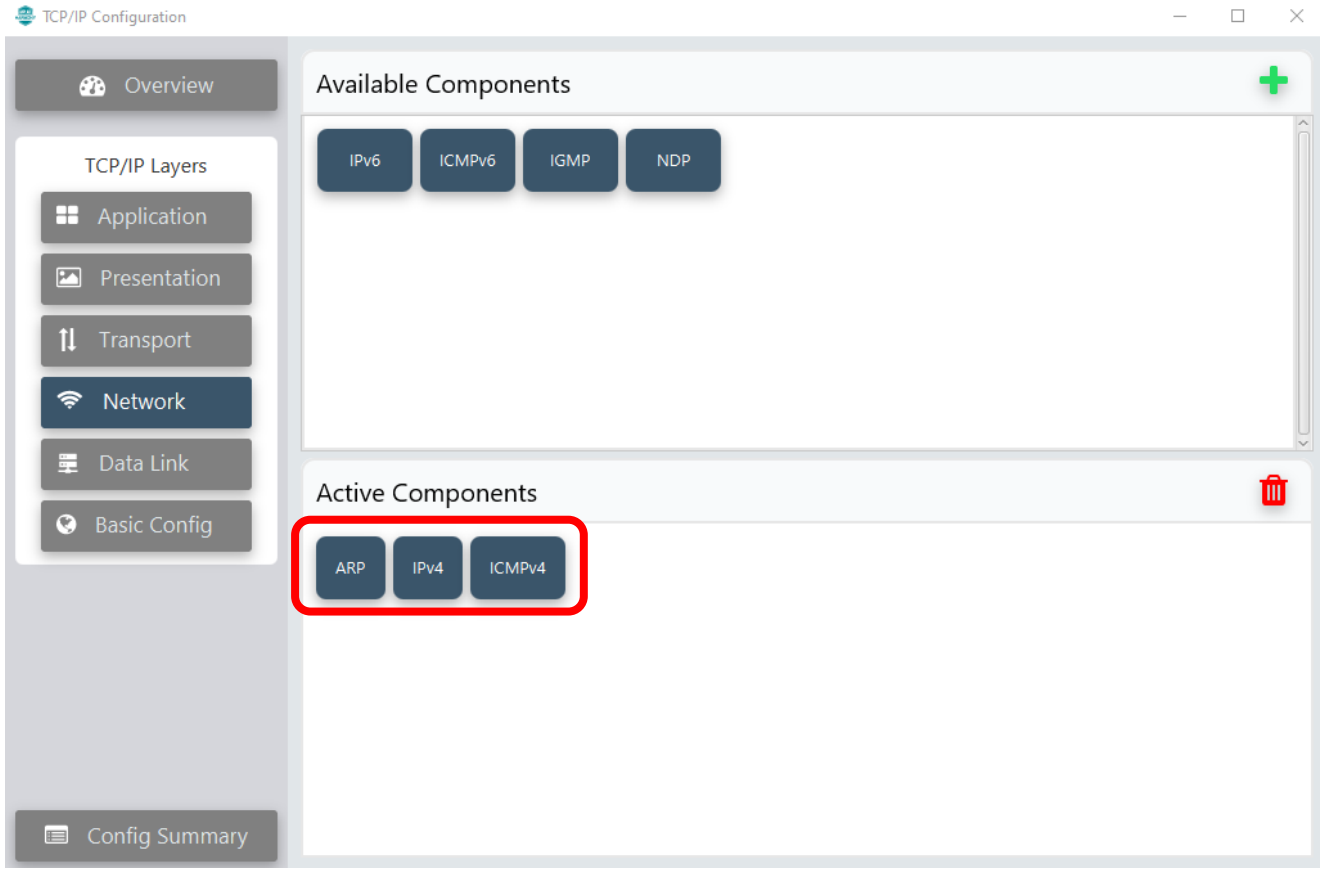



**Note: this pop-up sometimes is hidden behind the various MPLAB® X windows, you may not see it until a later point, just make sure to click Yes when it appears**

# TCP/IP: Network Layer

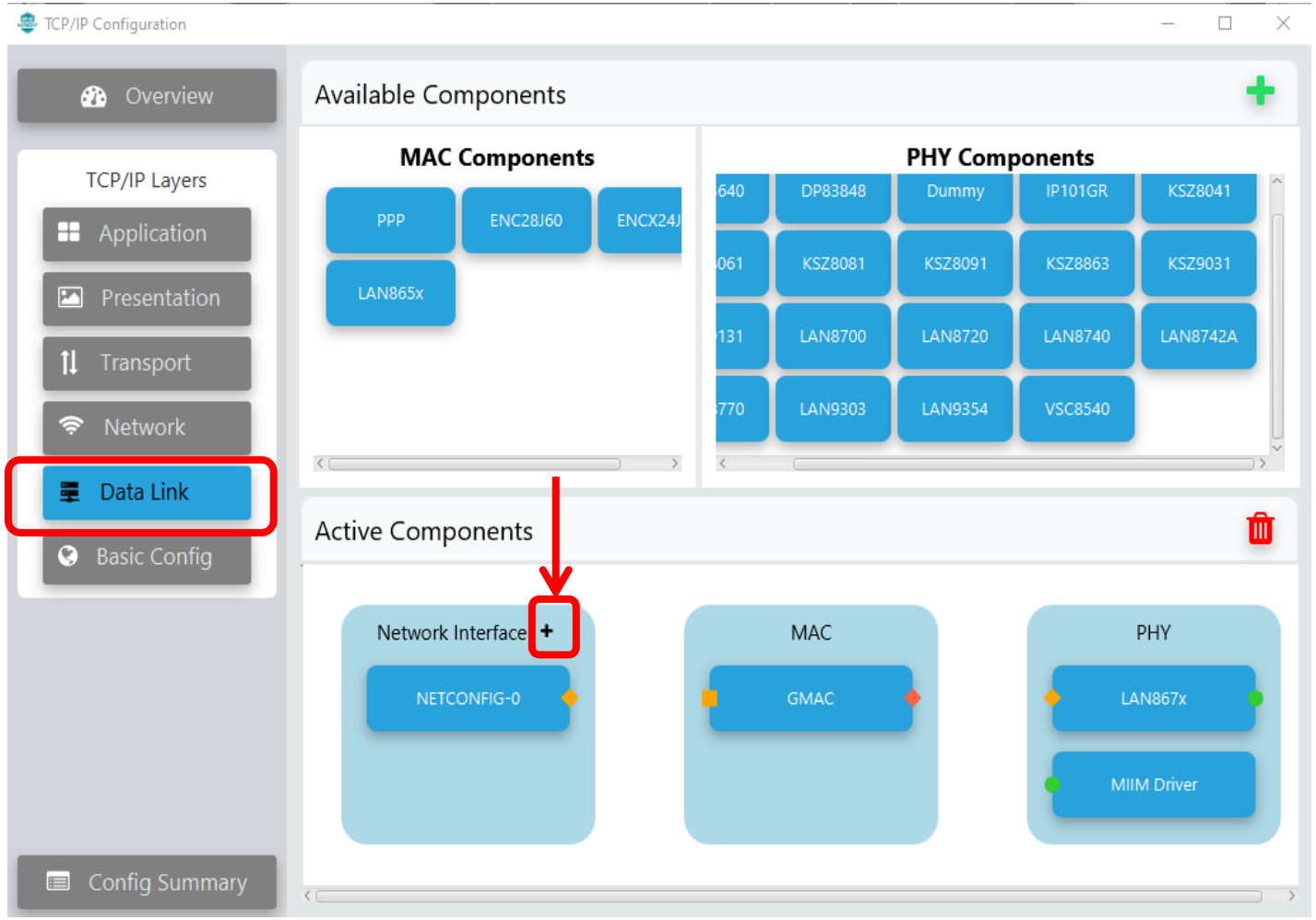
- **In the Network Layer, add the ARP, IPv4 and ICMPv4 components**



# TCP/IP: Data Link Layer

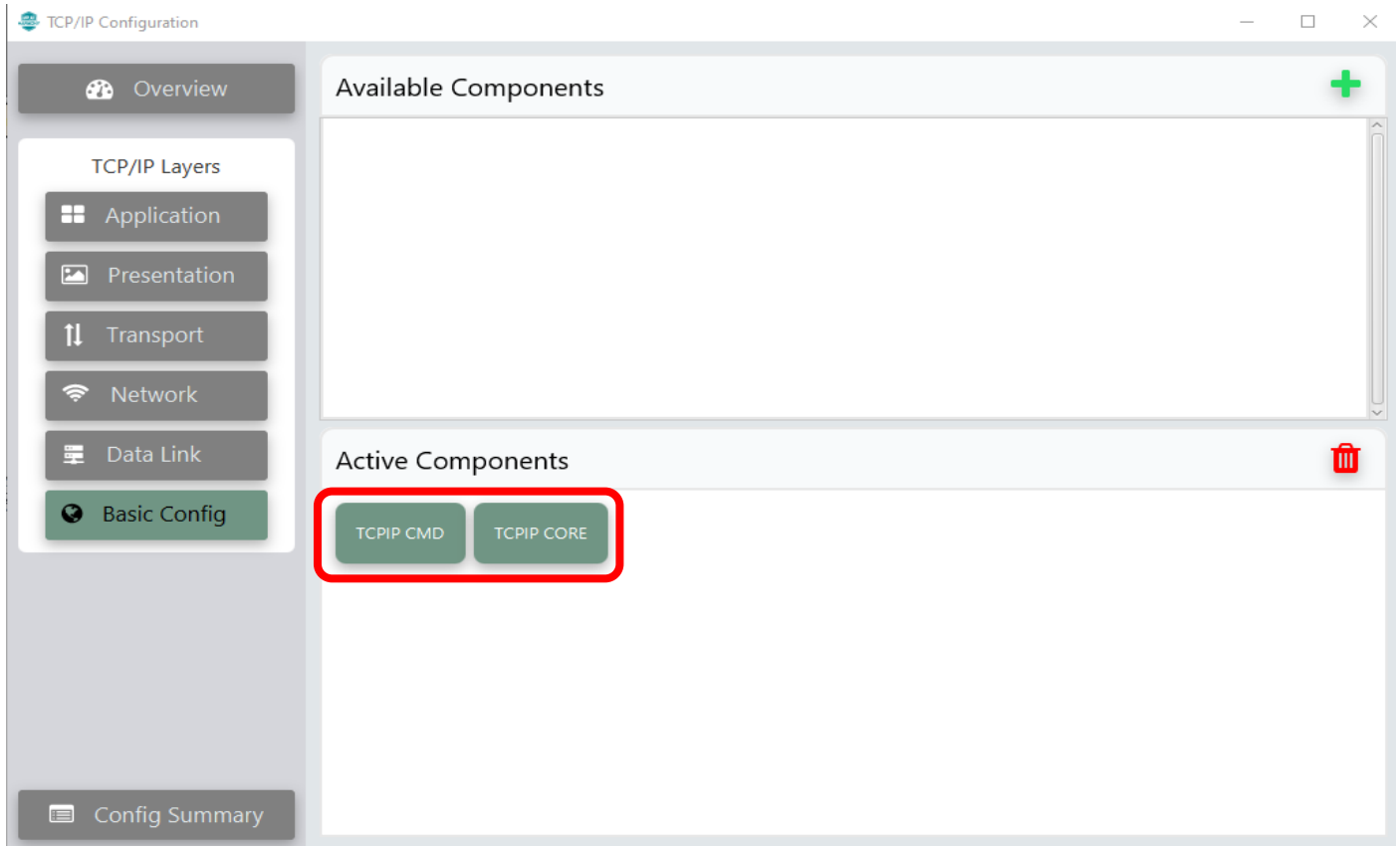
- **In the Data Link Layer, from the MAC Components box, add the GMAC component. From the PHY Components box, add the LAN867x and MIIM Driver components**

- **Click the “+” beside Network Interface to add the NETCONFIG-0 component**



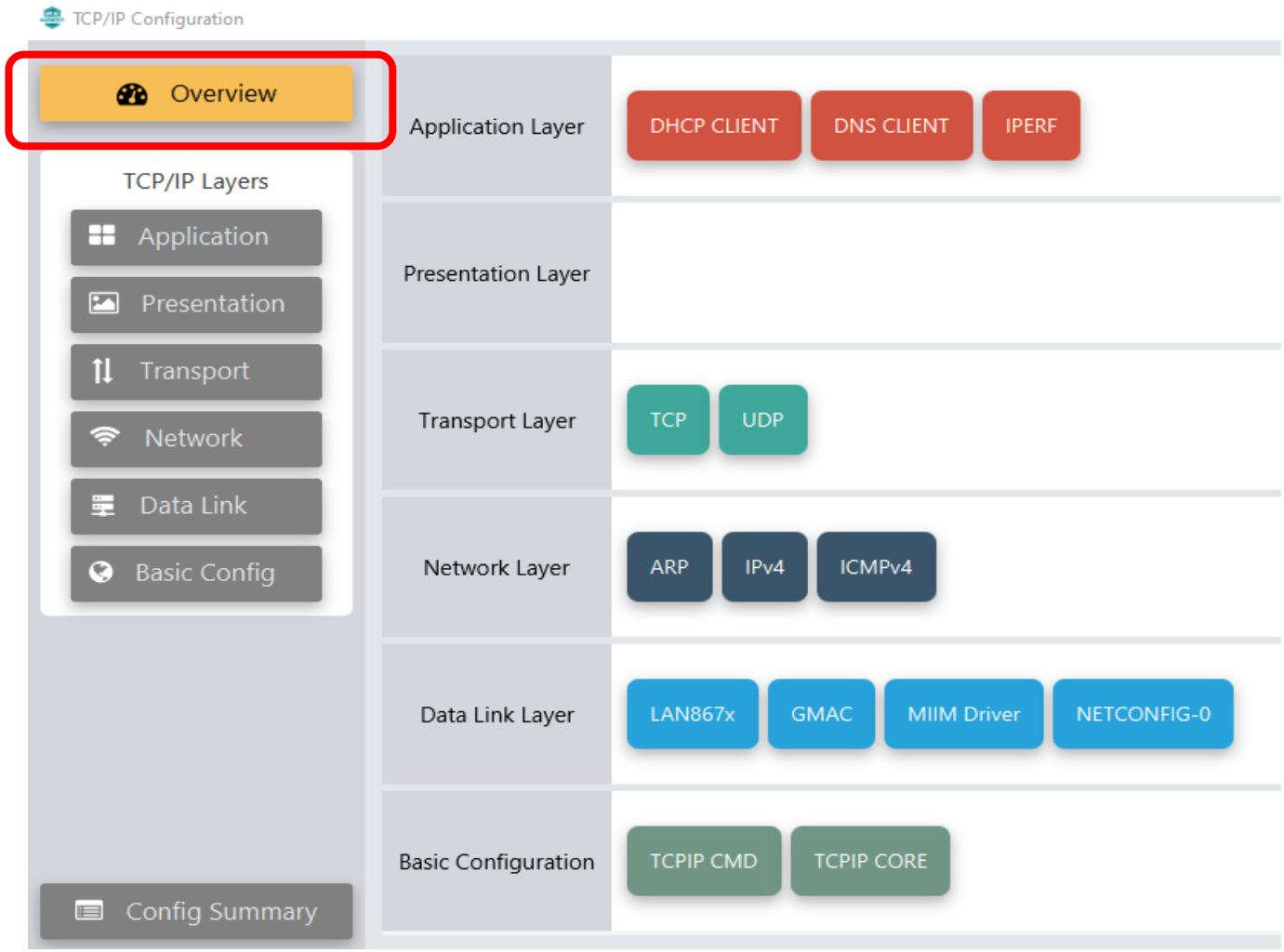
# TCP/IP: Basic Config

- **In Basic Config, the TCPIP CORE is there already**
- **Add the TCPIP CMD component**



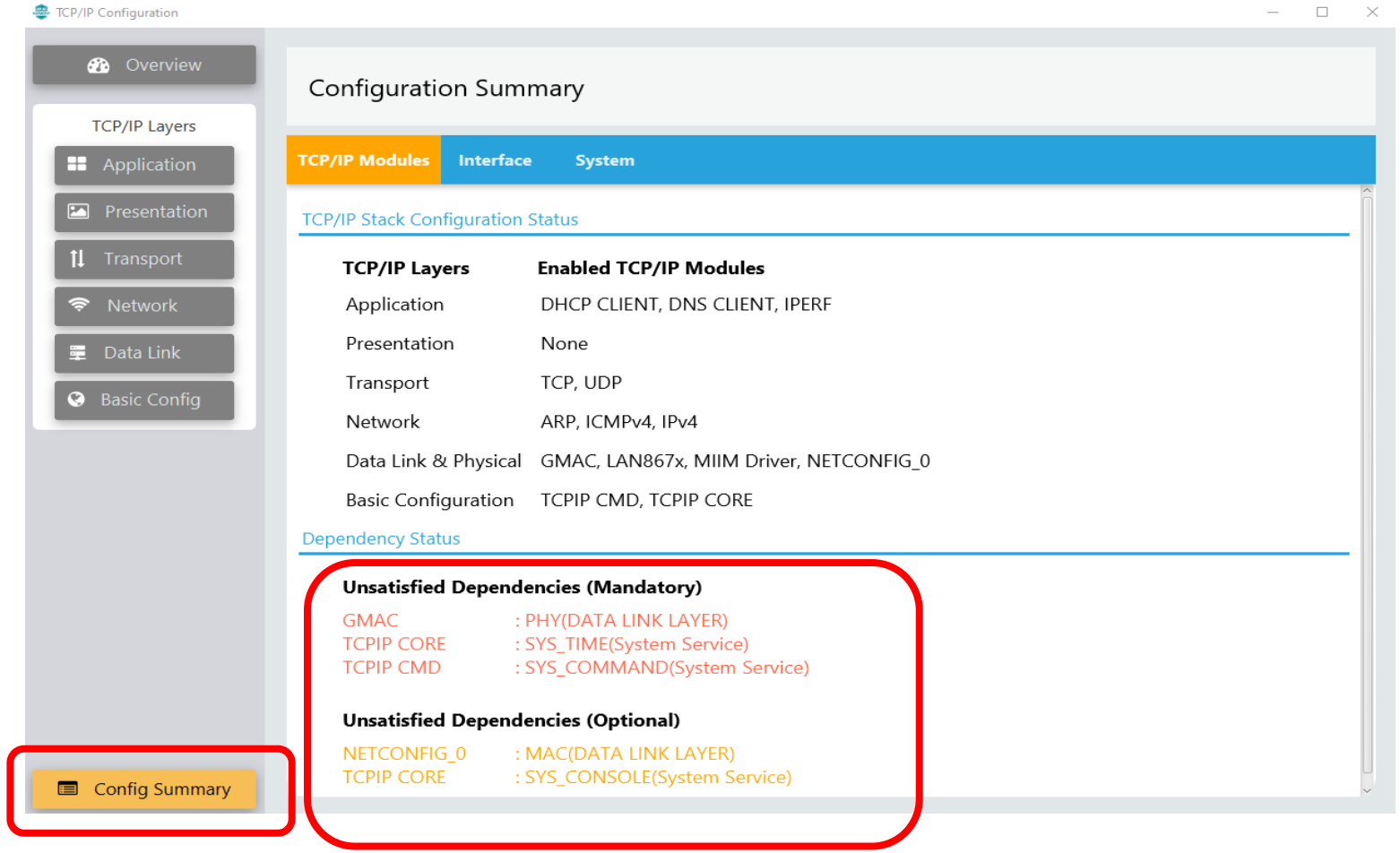
# Overview

- **The Overview of the TCP/IP Configurator should now be populated with the necessary components for the stack**



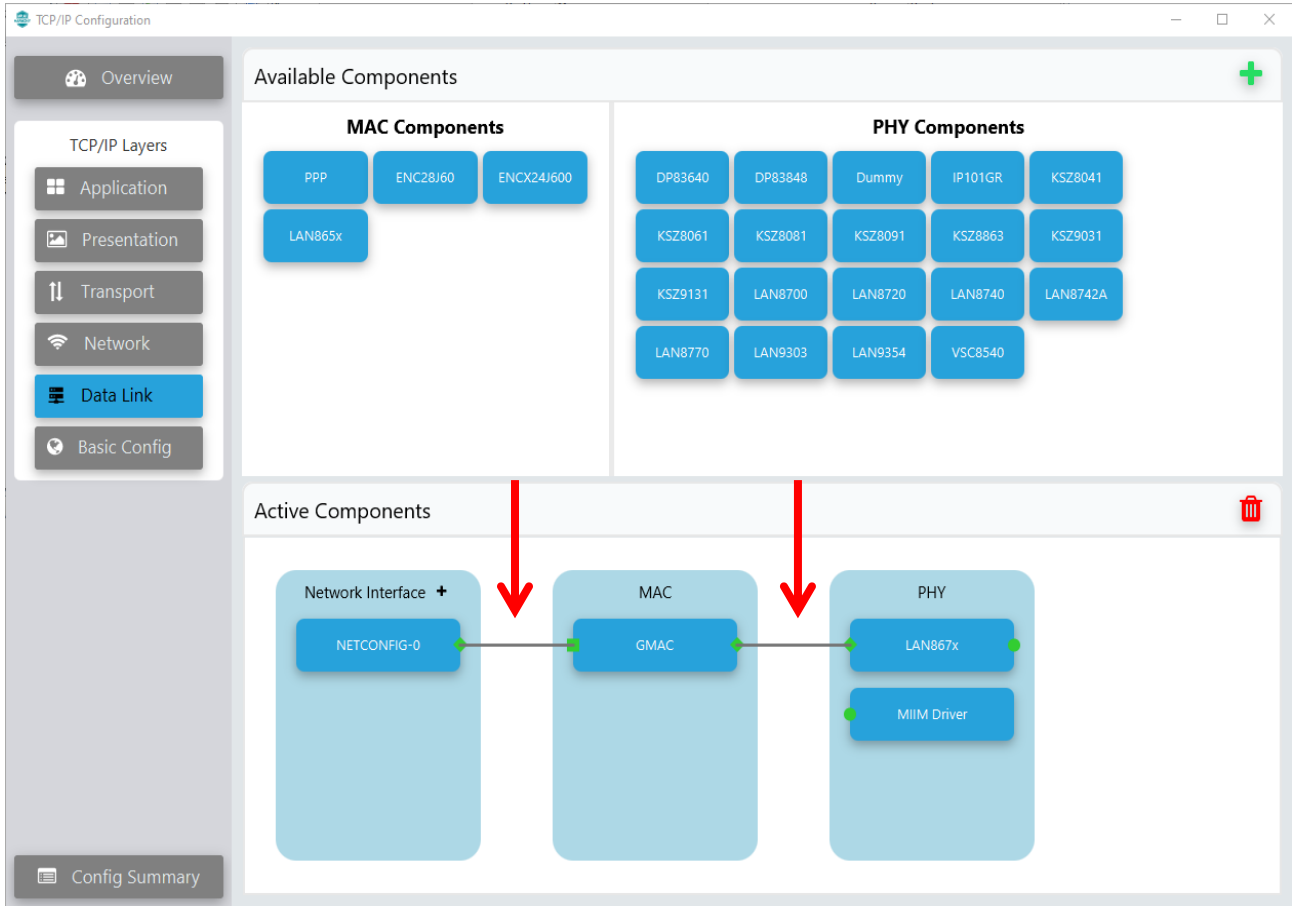
# Config Summary

- Check the Config Summary to see what has been included
- Note the Unsatisfied Dependencies, these are elements of the project that the components require in order to work for this project, and they must be satisfied.



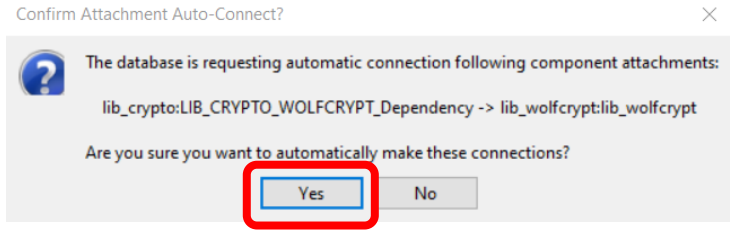
# TCP/IP Data Link Dependencies

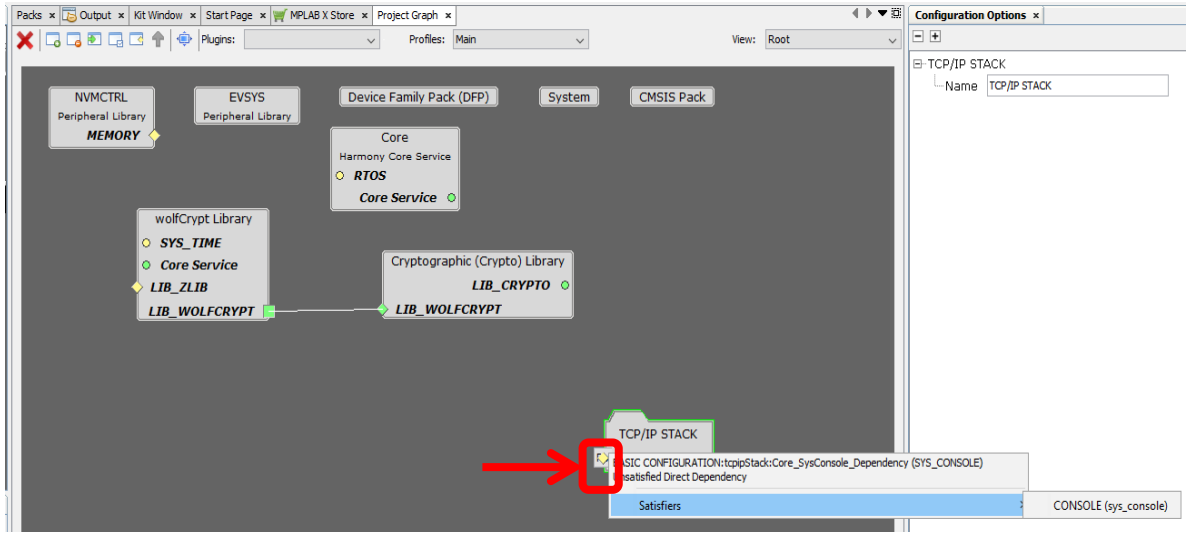
- **In the Data Link Layer, connect NETCONFIG-0 to GMAC**
- **Connect GMAC to LAN867x PHY driver**
  - To do these connections, simply use the mouse to “draw” a line between the red/orange dots on the components



# System Console

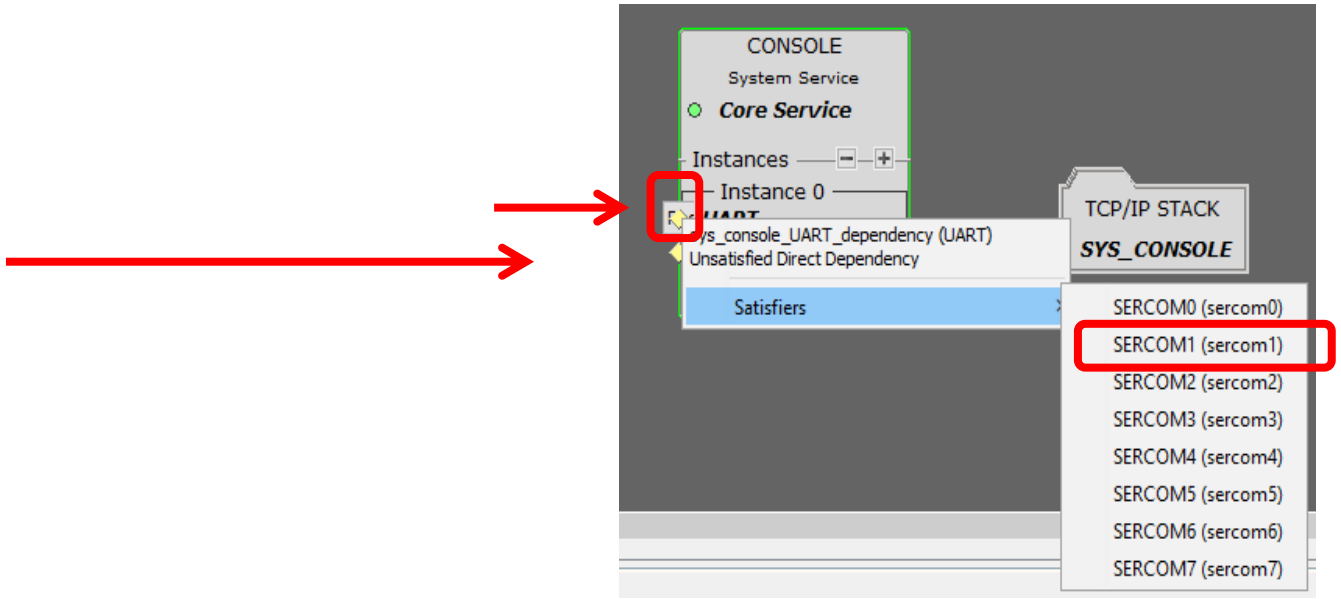
- **Return to the Project Graph, you should see the TCP/IP Stack and the WolfCrypt components. (You may need to accept the Wolfcrypt dependency confirmation popup at this point if you have not already)**
- **Use the Project Graph to add the System Console to the project**
  - Do this by right-clicking on the yellow diamond “SYS_CONSOLE” and select Satisfiers > CONSOLE (sys_console)



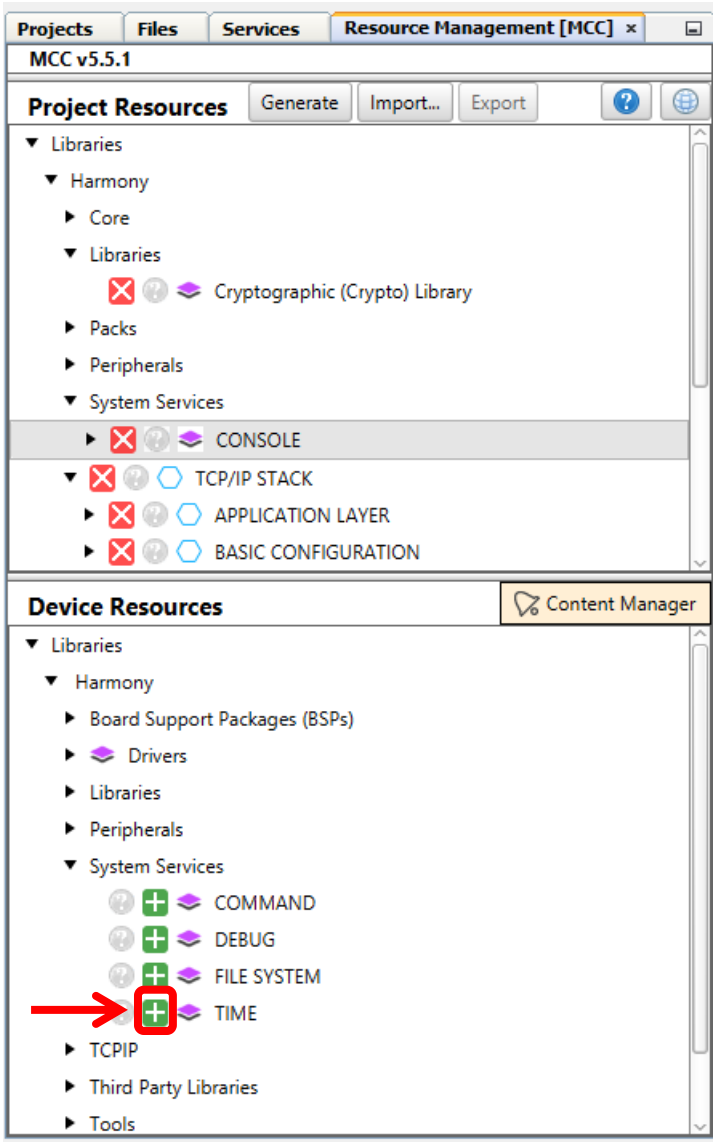
# System Console

- The System Service Console is now added to the project graph
- This Console needs a UART instance, right click on the yellow diamond labelled “UART” and select SERCOM1

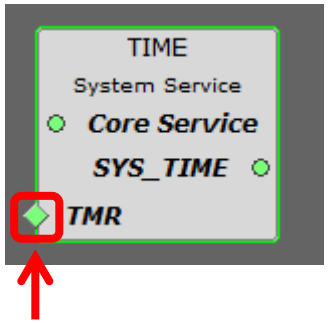


# TIME System Service

- Add the TIME System Service from the Device Resources window, which is located in Libraries > Harmony > System Services
- Do this by clicking on the “+” beside the TIME system service



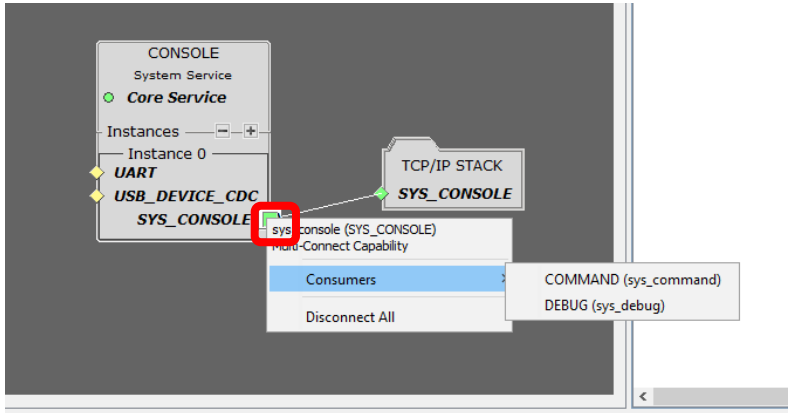

- Right click on the yellow diamond labelled “TMR” and select TC0 as the timer for the project (the diamond will turn green when TC0 is added)

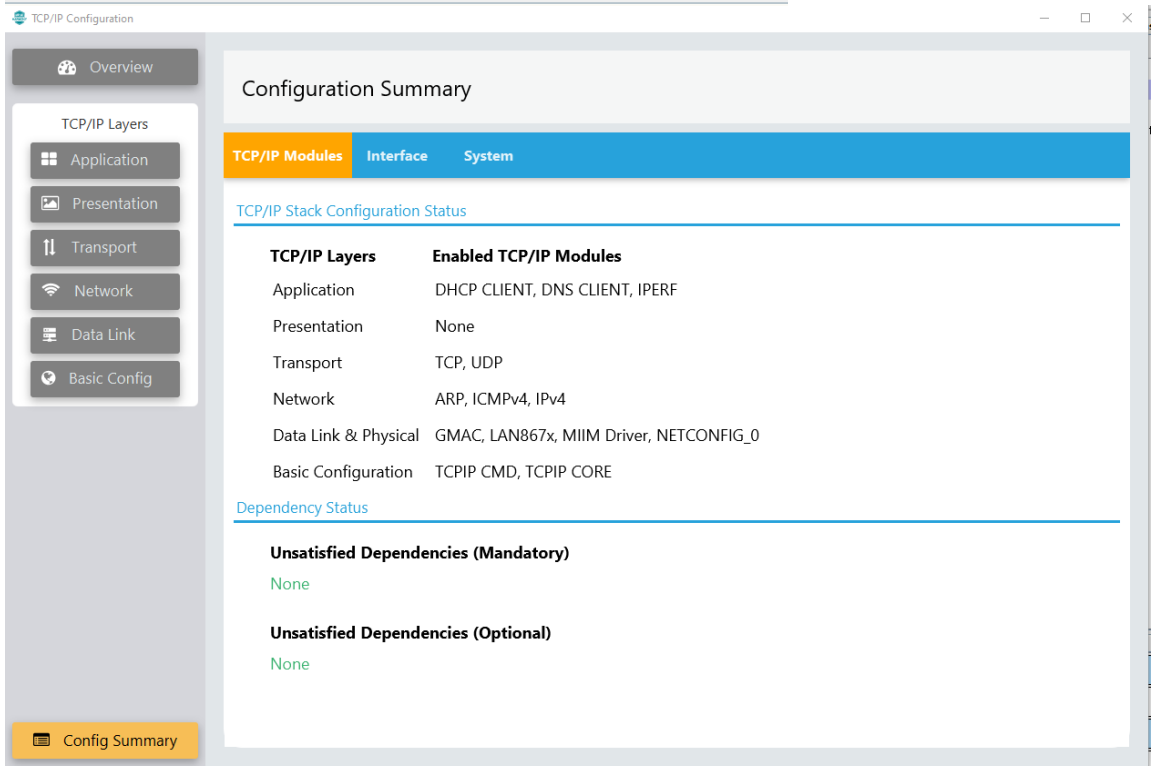


# COMMAND and DEBUG

- Add the COMMAND (sys_command) and DEBUG (sys_debug) as Consumers for the Console System Service, along with the TCP/IP Stack.
- Do this by right clicking on the green square labelled “SYS_CONSOLE” and selecting these components, one at a time

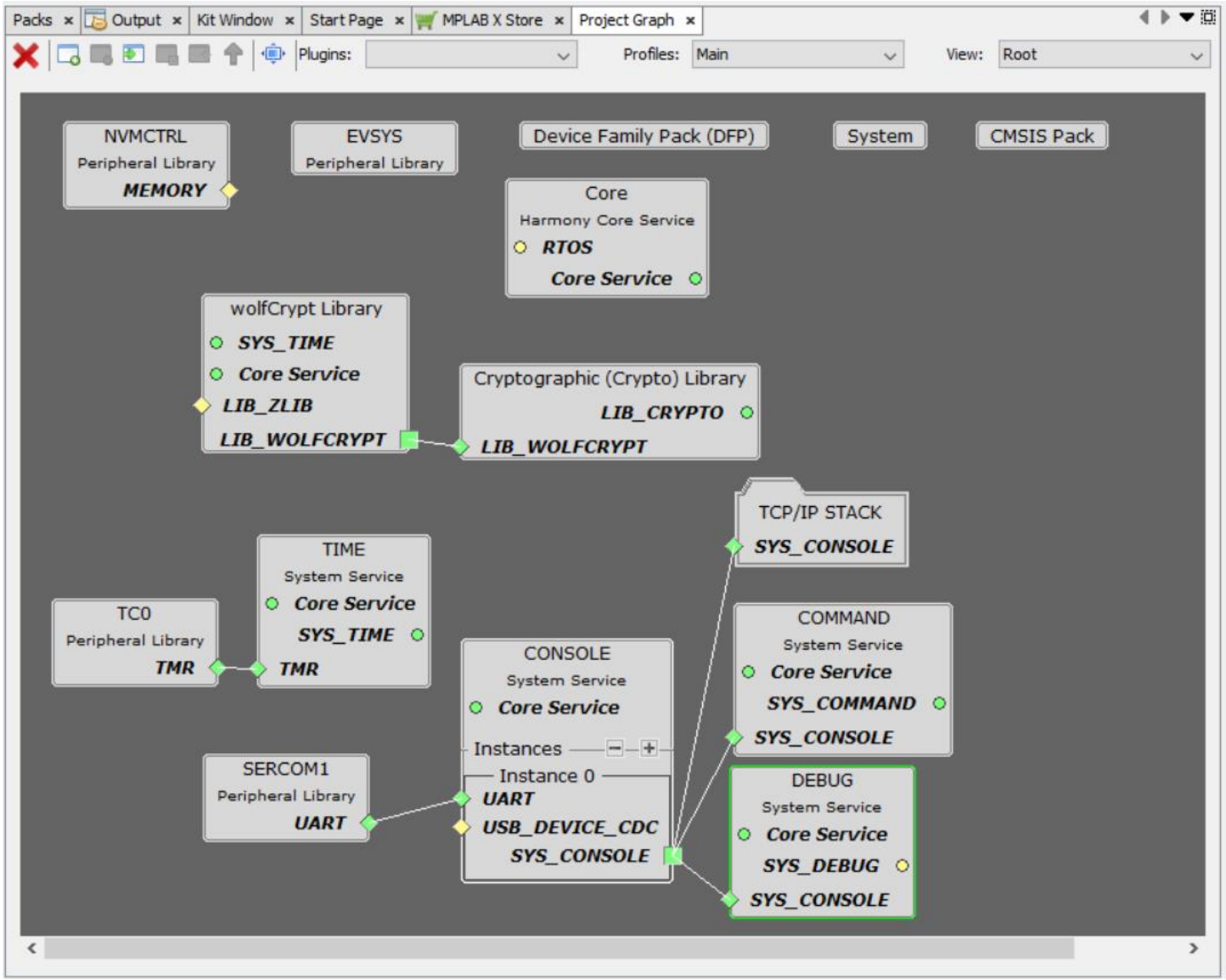
- The TCP/IP Config Summary should now have no unsatisfied dependencies
- Close the TCP/IP Configuration Tool at this point



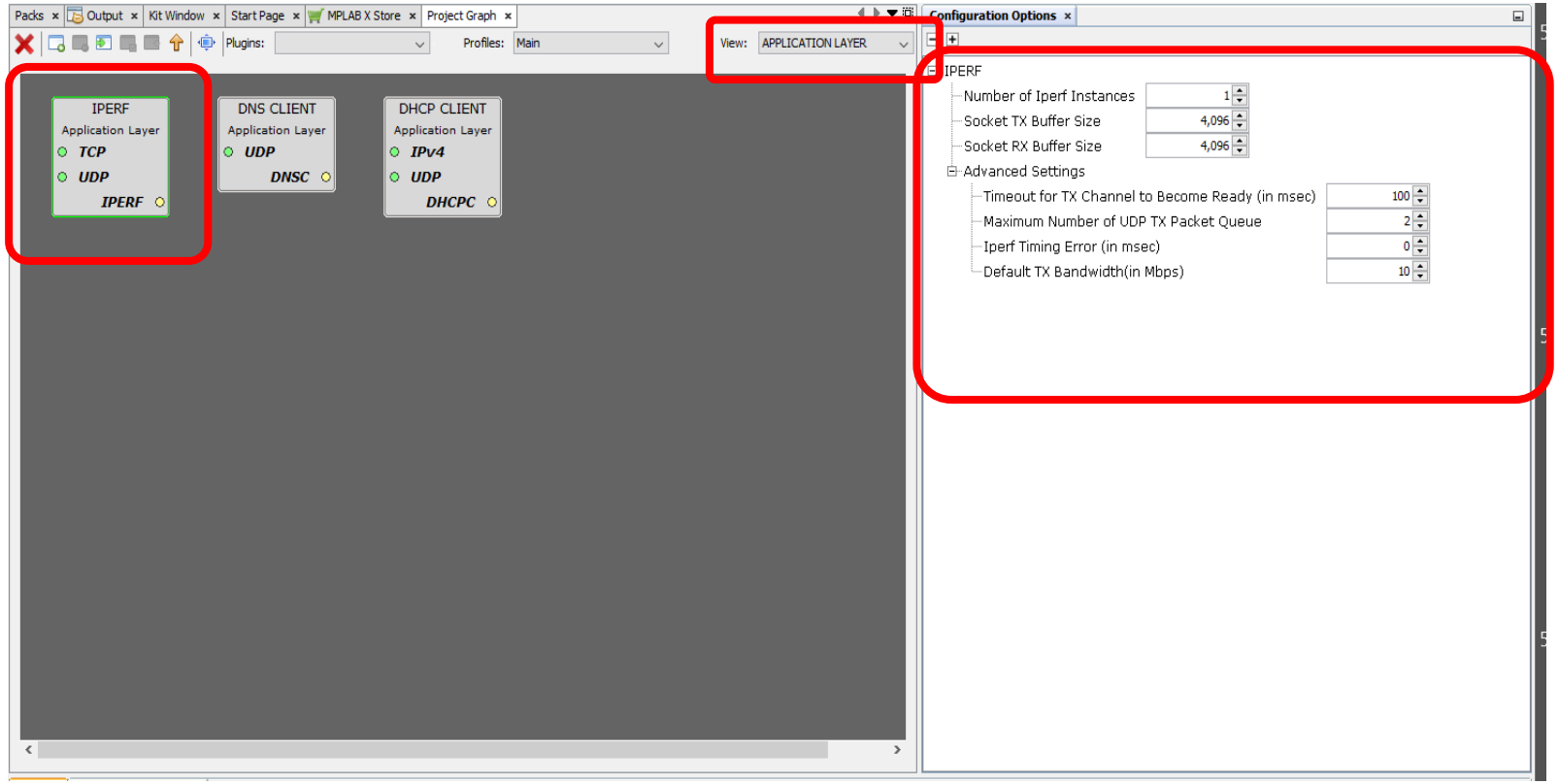
# Project Graph

- **The project graph should now contain all of the necessary components**



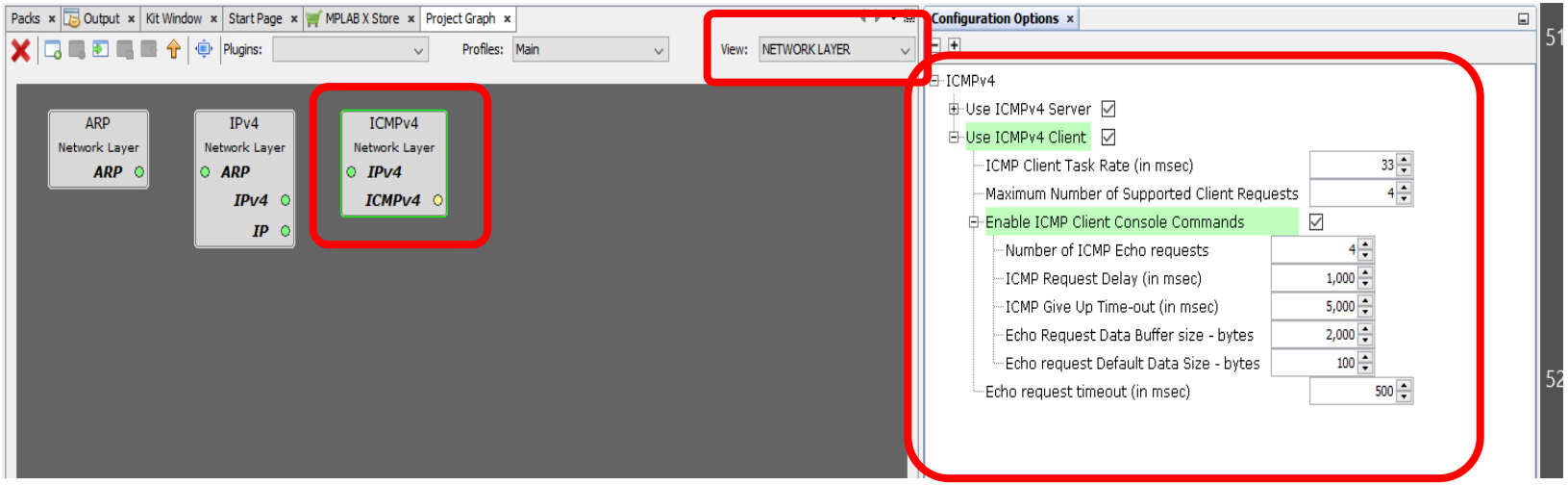
# Application Layer Settings

- **In the Application Layer, Select the IPERF component, and change the speed to 10Mbps**



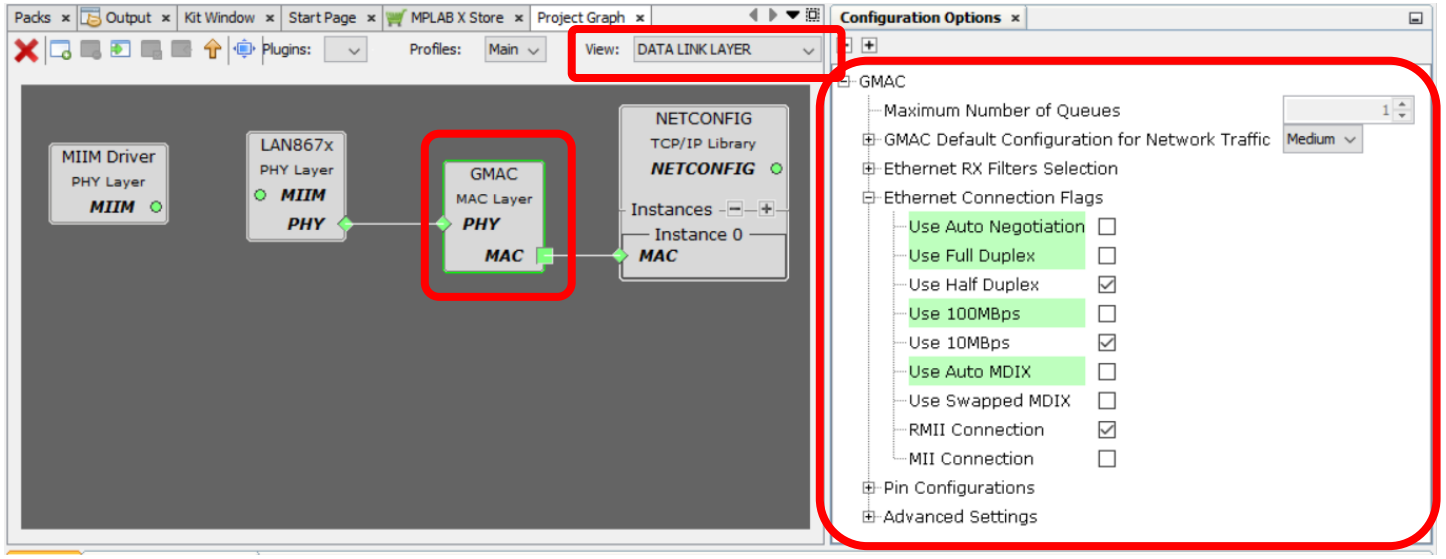
# Network Layer Settings

- In the Network Layer, for the ICMPv4 component, select to Use ICMPv4 Client, and Enable ICMP Client Console Commands

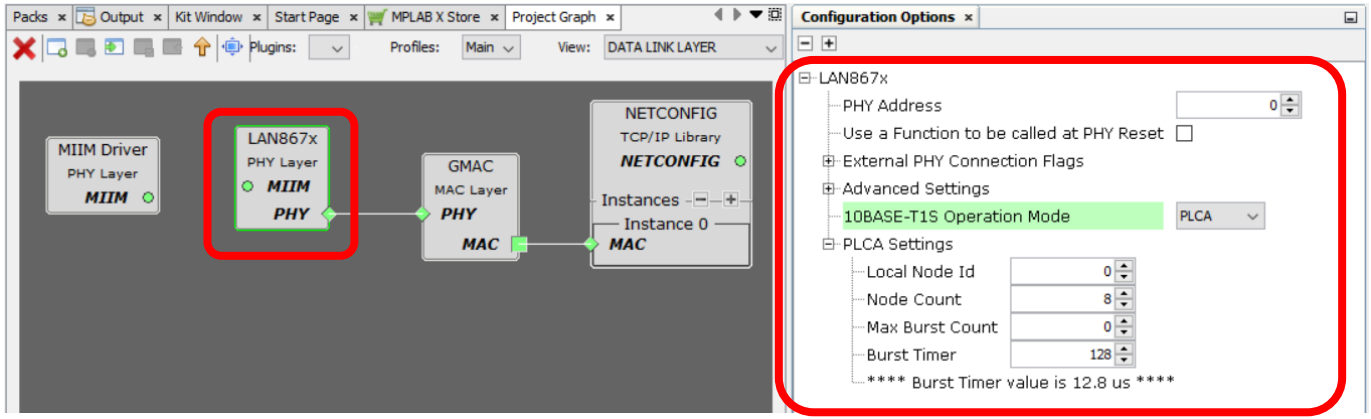


# Data Link Layer Settings

- In the Data Link Layer, click on the GMAC component and set the Ethernet Connection Flags as shown

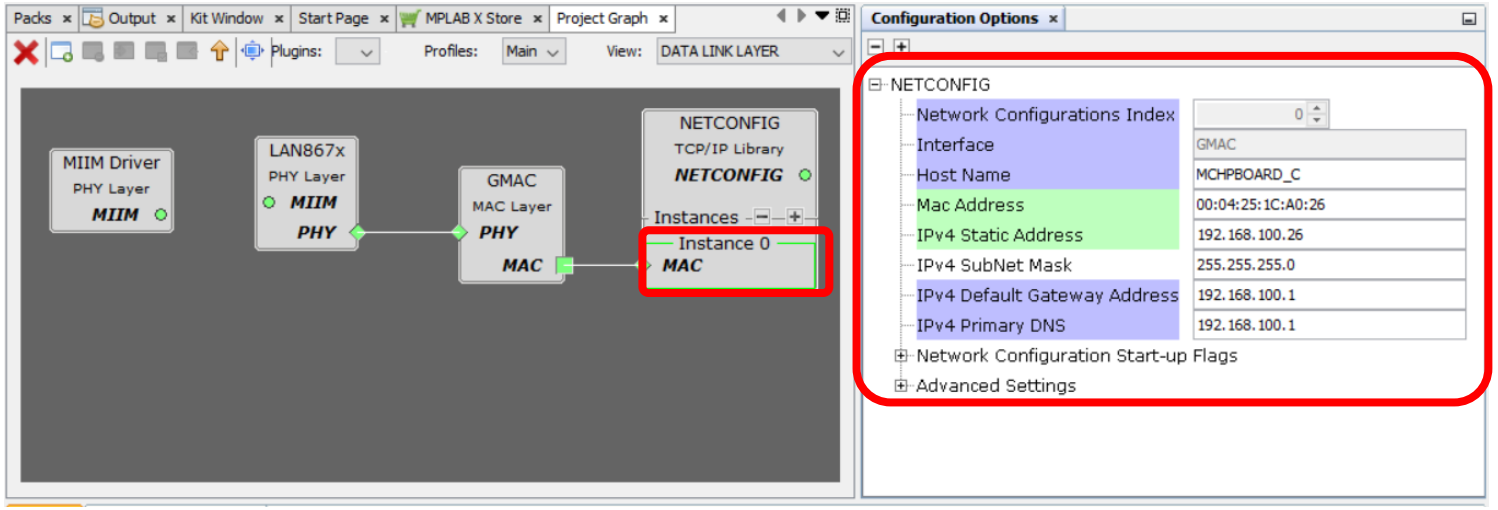



- Click on the LAN867x PHY Layer component and select the Advanced Settings to use PLCA, leave the other settings as is



# Network Configuration

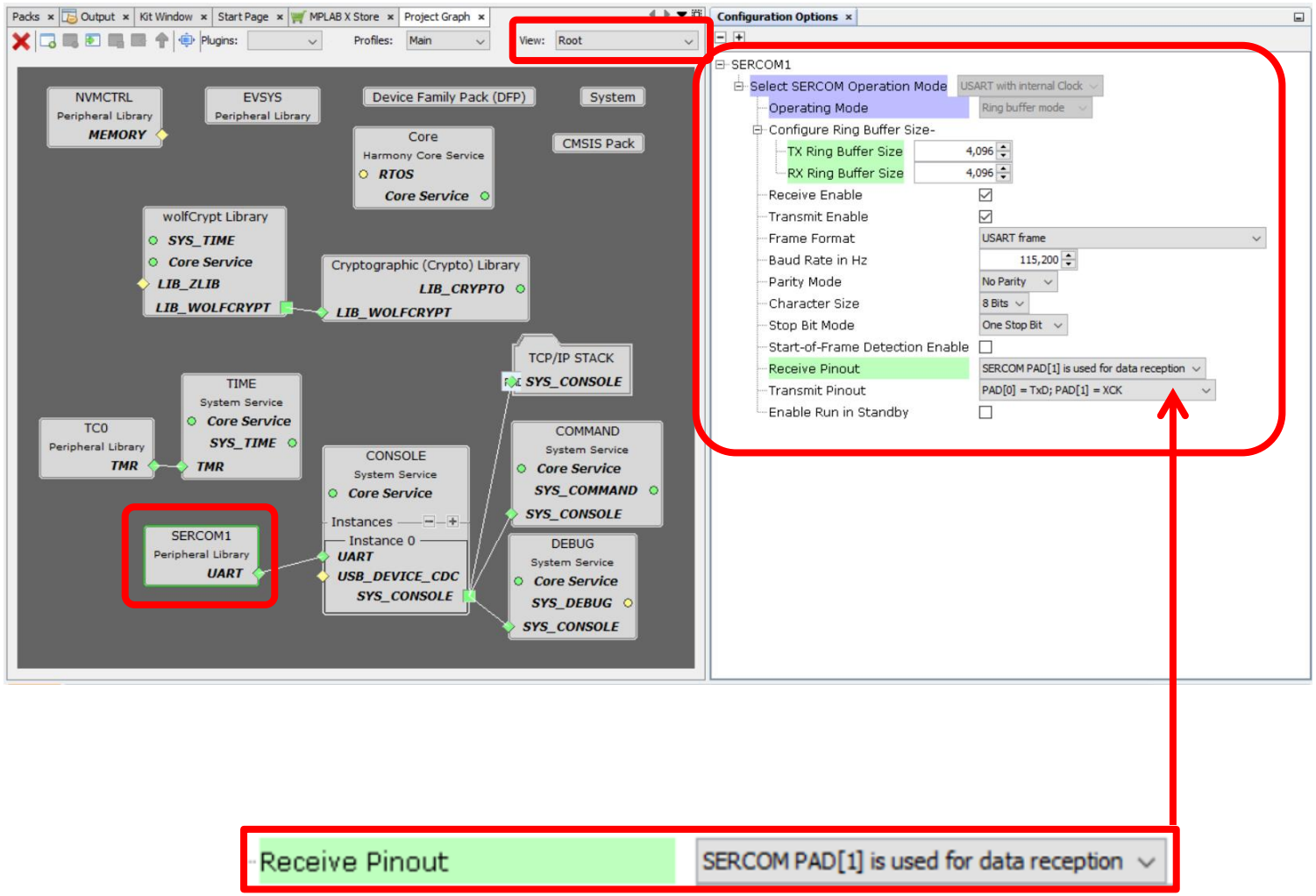
- Observe the network configuration settings in the NETCONFIG component by clicking on the Instance 0 part of this component, and change the MAC address and IP Address.
- Each of the two boards will need distinct MAC addresses
- They will also need distinct IP Addresses which will need to be on the same network subnet with distinct node numbers, here “26” is used



# SERCOM Configuration

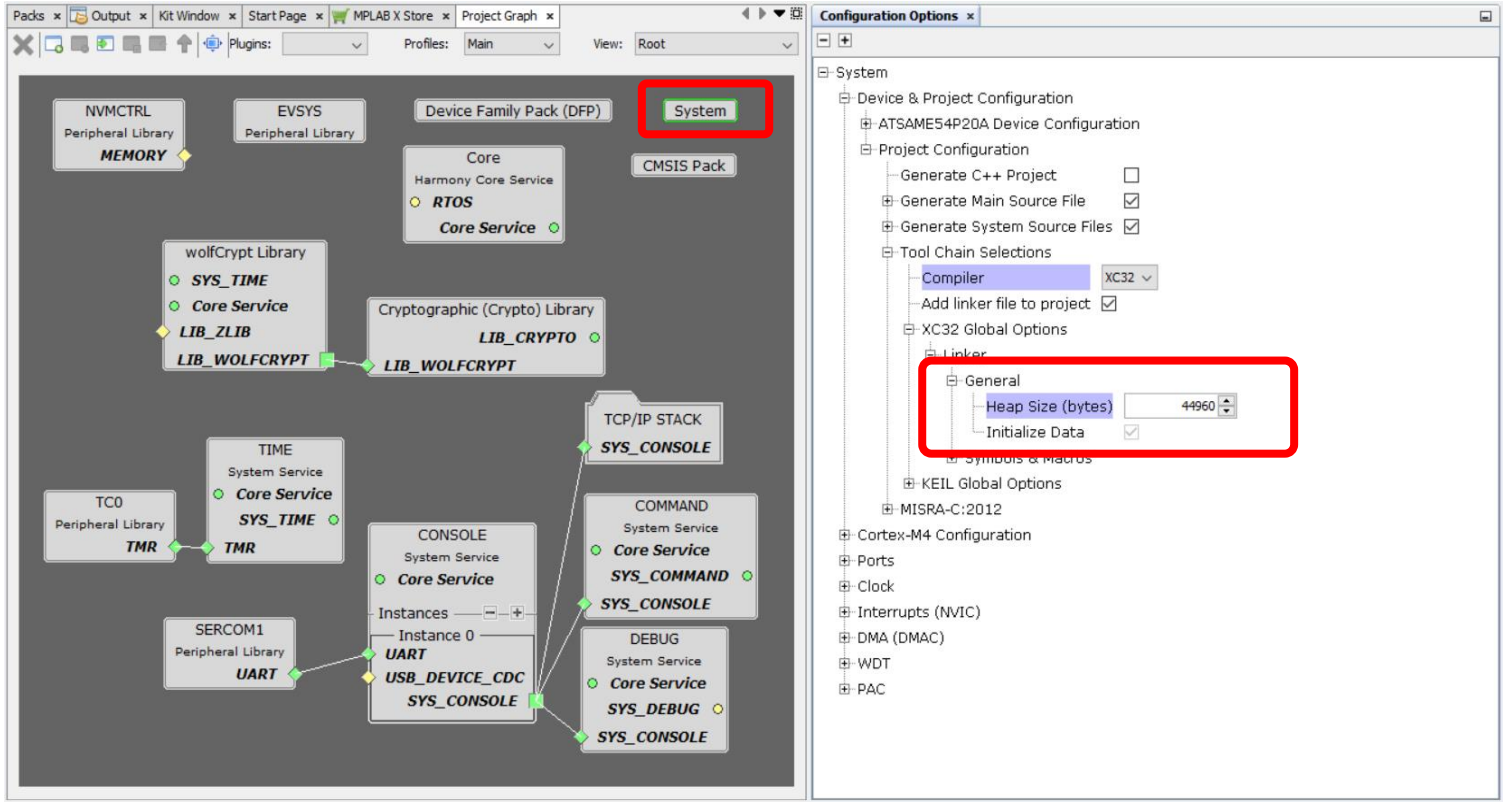
- **In the Root layer of the Project Graph, configure SERCOM1 to increase the TX and RX Ring Buffer Size to 4096**
- **Set the Receive Pinout to:**

  “SERCOM PAD[1] is used for data reception”



# Heap Size

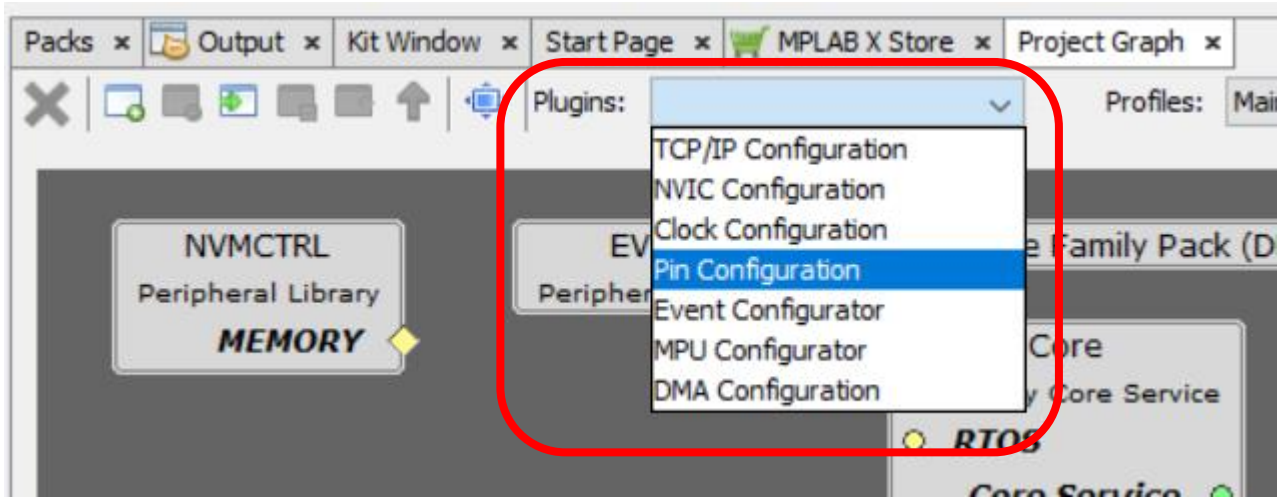
- **Select the System component, and change the Heap Size to 44960 bytes as shown**




**Note:** if you usually use the Project Properties > XC32(Global Options) > xc32-ld settings to set the Heap Size – be careful, as this may be overwritten when you generate the code for the project – make sure to check the Project Properties before you compile the project

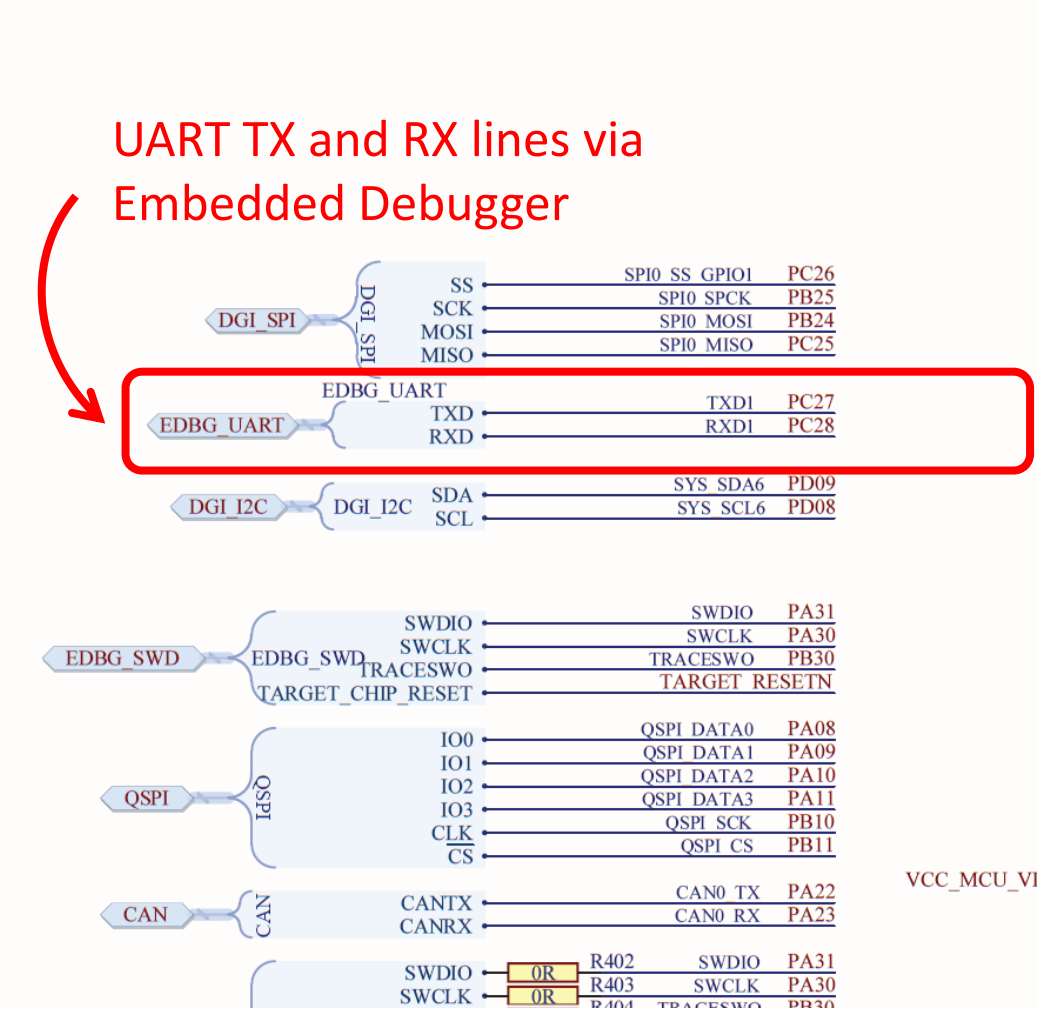
# Pin Configuration Tool

- **Open the Pin Configuration Tool in the Plugins dropdown menu**

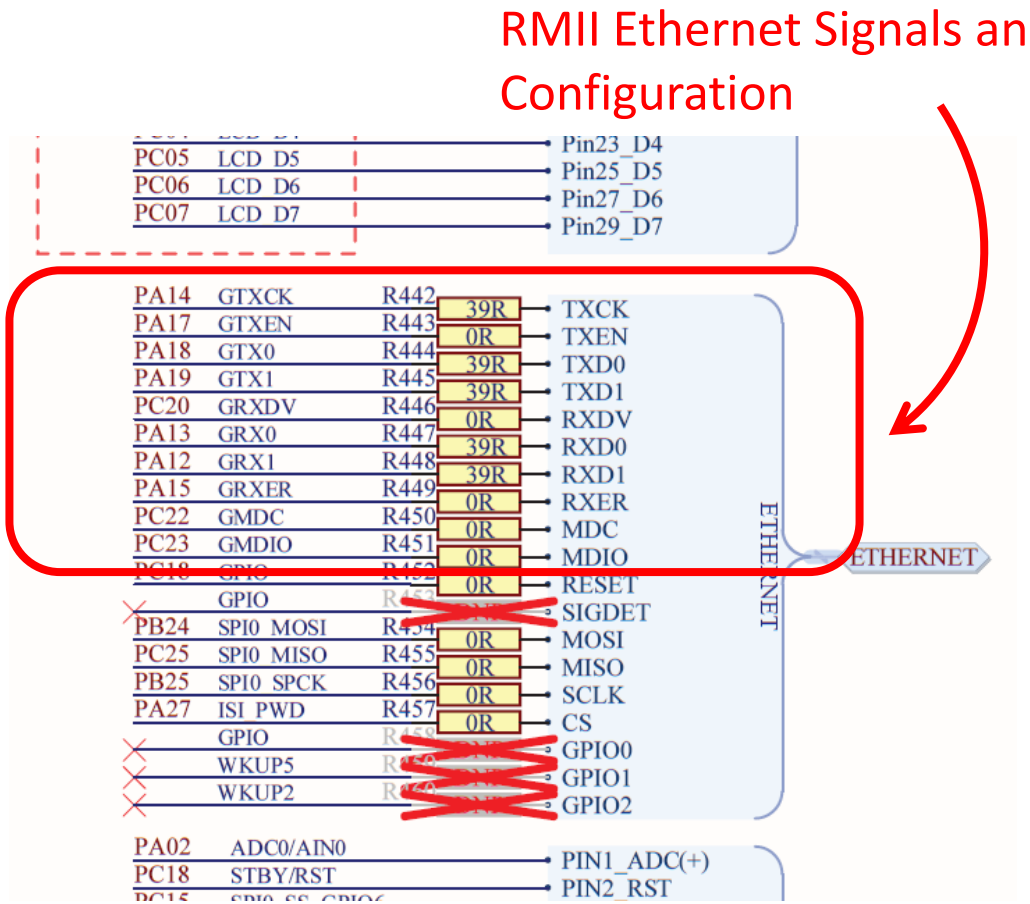


# These are some snapshots from the Curiosity Board schematic to check for the pin connections for the ethernet and UART pins

## UART Pin Configuration

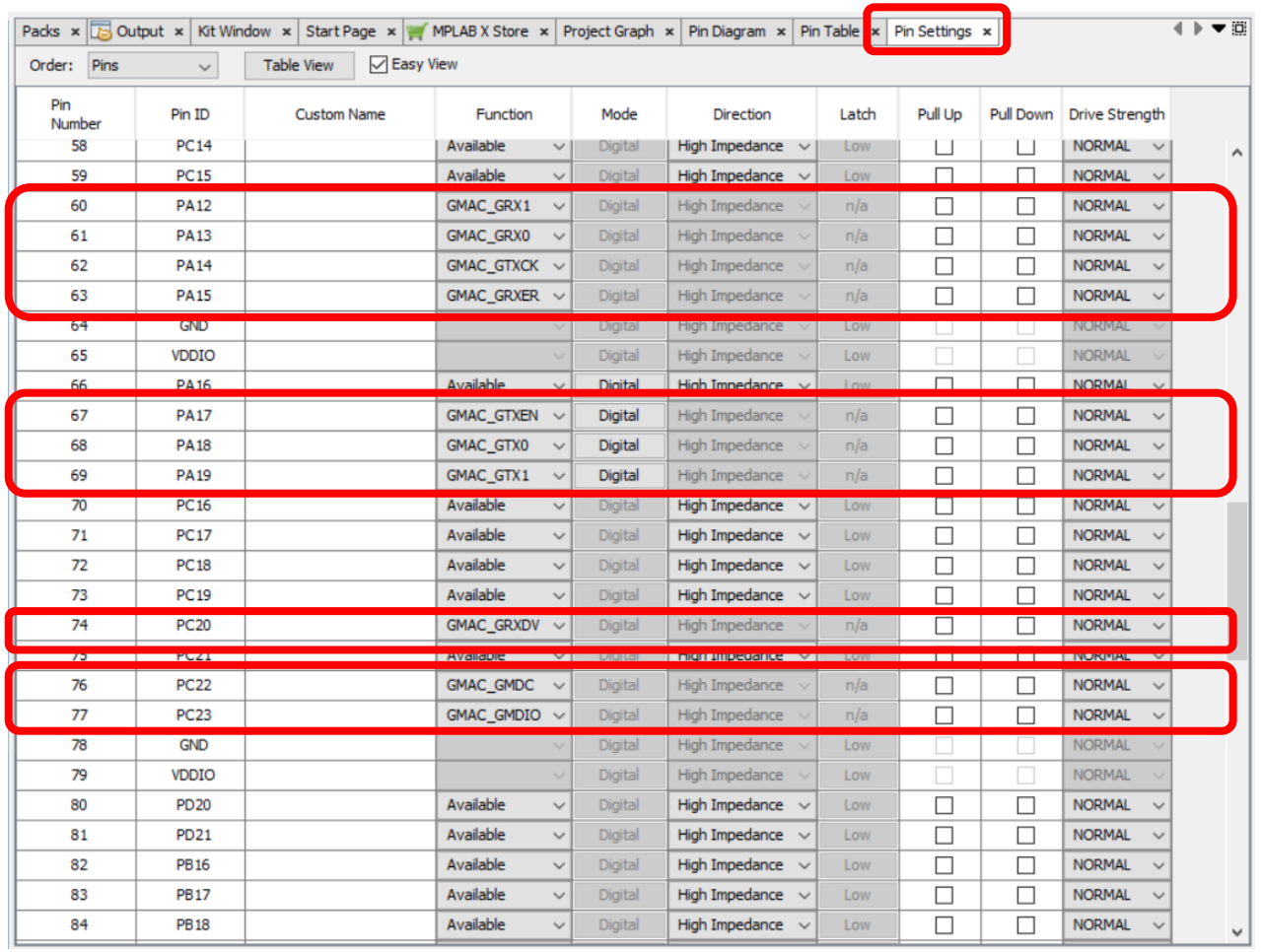



## GMAC Pin Configuration



# Pin Configuration Tool

- Having checked the schematic to see the connections on the board, use the Pin Configuration Tool to set up the pins.
- The Pin Settings tab is the easiest way to do this. Select the values shown for PA12, PA13, PA14, PA15, PA17, PA18, PA19, PC20, PC22 and PC23, to match with the board connections.



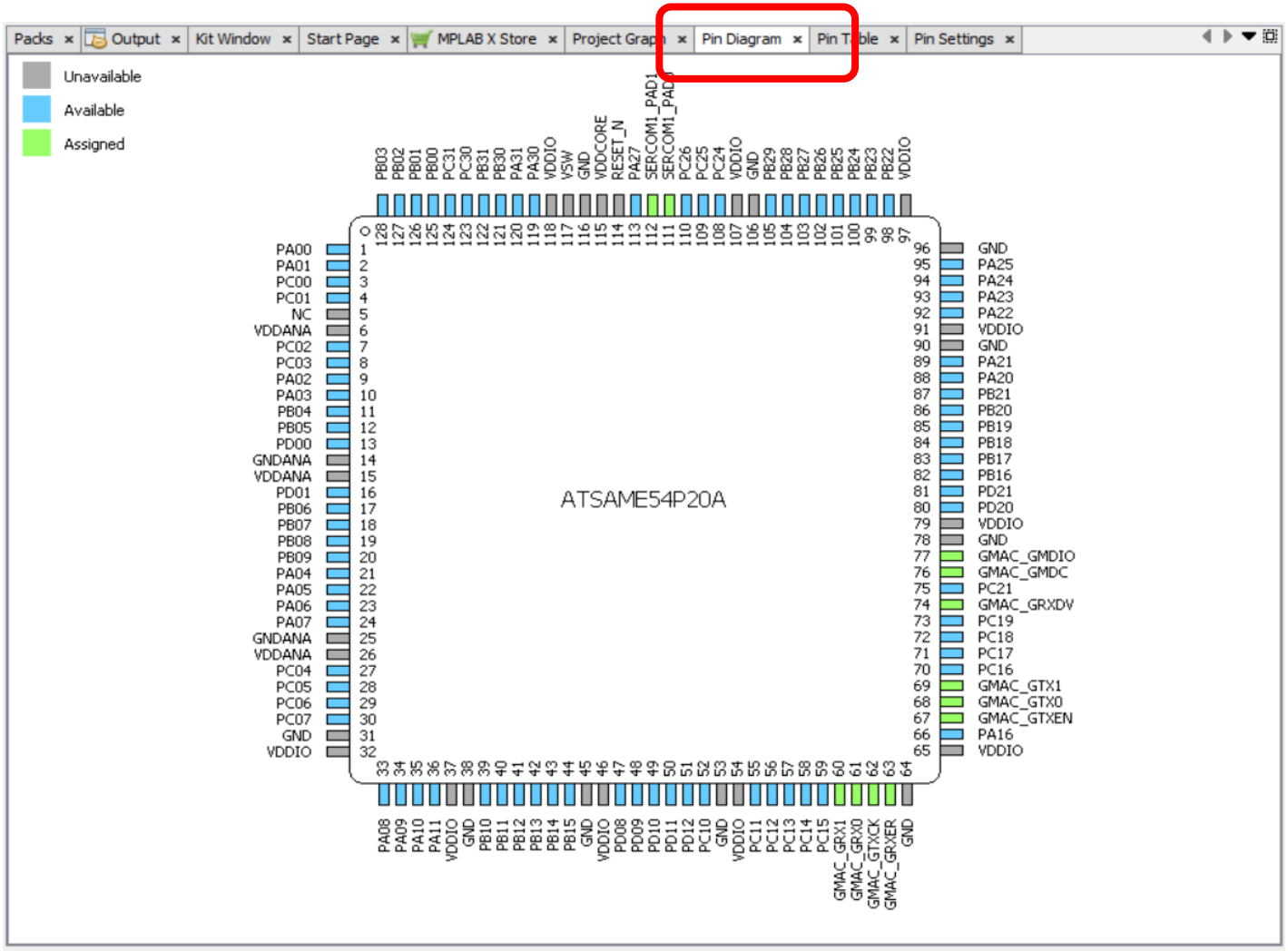

| Pin Number | Pin ID | Custom Name | Function | Mode | Direction | Latch | Pull Up | Pull Down | Drive Strength |
|---|---|---|---|---|---|---|---|---|---|
| 58 | PC14 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 59 | PC15 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 60 | PA12 | | GMAC_GRX1 | Digital | High Impedance | n/a | | | NORMAL |
| 61 | PA13 | | GMAC_GRX0 | Digital | High Impedance | n/a | | | NORMAL |
| 62 | PA14 | | GMAC_GTXCK | Digital | High Impedance | n/a | | | NORMAL |
| 63 | PA15 | | GMAC_GRXER | Digital | High Impedance | n/a | | | NORMAL |
| 64 | GND | | | Digital | High Impedance | Low | | | NORMAL |
| 65 | VDDIO | | | Digital | High Impedance | Low | | | NORMAL |
| 66 | PA16 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 67 | PA17 | | GMAC_GTXEN | Digital | High Impedance | n/a | | | NORMAL |
| 68 | PA18 | | GMAC_GTX0 | Digital | High Impedance | n/a | | | NORMAL |
| 69 | PA19 | | GMAC_GTX1 | Digital | High Impedance | n/a | | | NORMAL |
| 70 | PC16 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 71 | PC17 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 72 | PC18 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 73 | PC19 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 74 | PC20 | | GMAC_GRXDV | Digital | High Impedance | n/a | | | NORMAL |
| 75 | PC21 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 76 | PC22 | | GMAC_GMDC | Digital | High Impedance | n/a | | | NORMAL |
| 77 | PC23 | | GMAC_GMDIO | Digital | High Impedance | n/a | | | NORMAL |
| 78 | GND | | | Digital | High Impedance | Low | | | NORMAL |
| 79 | VDDIO | | | Digital | High Impedance | Low | | | NORMAL |
| 80 | PD20 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 81 | PD21 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 82 | PB16 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 83 | PB17 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 84 | PB18 | | Available | Digital | High Impedance | Low | | | NORMAL |

- Set the UART connections on pins PC27 and PC28

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 111 | PC27 | | SERCOM1_PAD0 | Digital | High Impedance | n/a | | | NORMAL |
| 112 | PC28 | | SERCOM1_PAD1 | Digital | High Impedance | n/a | | | NORMAL |

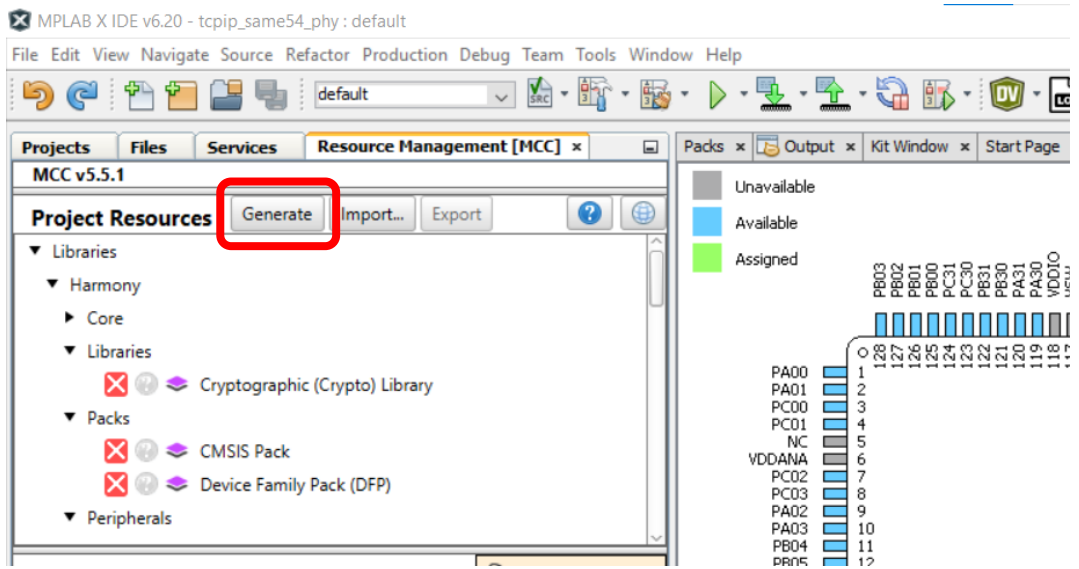
# Pin Diagram

- **The Pin Diagram should now look as shown – the pins that are green are the ones that have been configured using the Pin Settings**

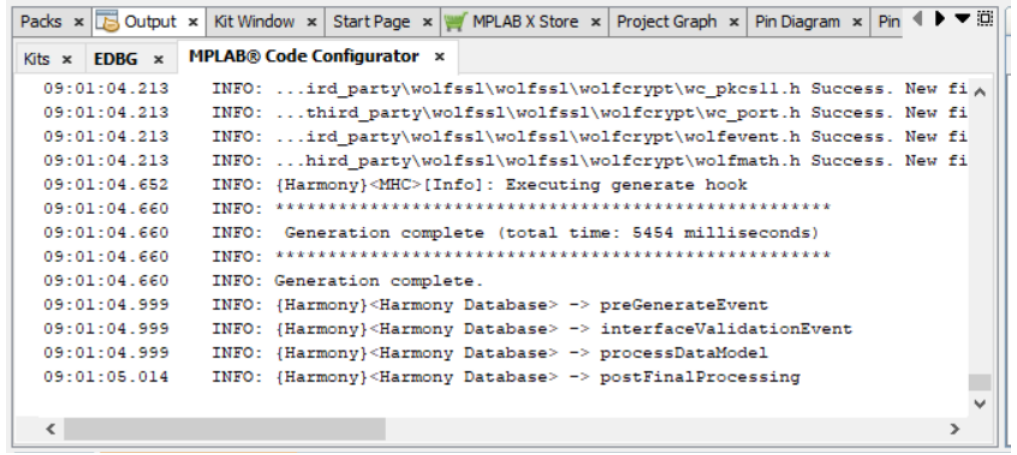


# Generate the Code

- **In Resource Management[MCC], use the Generate button to create the code for the project**

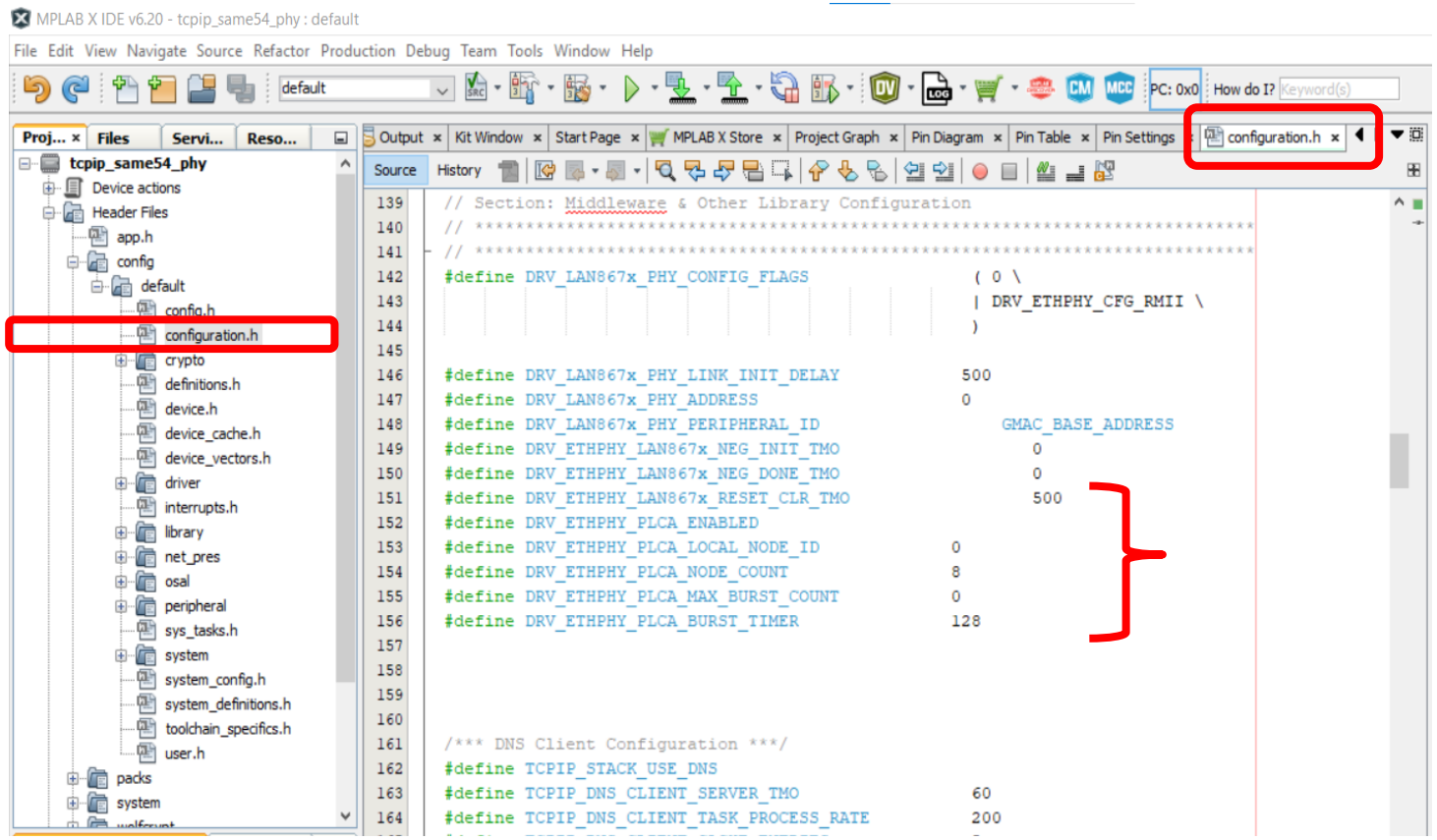



- **Watch the MPLAB® Code Configurator window for messages**



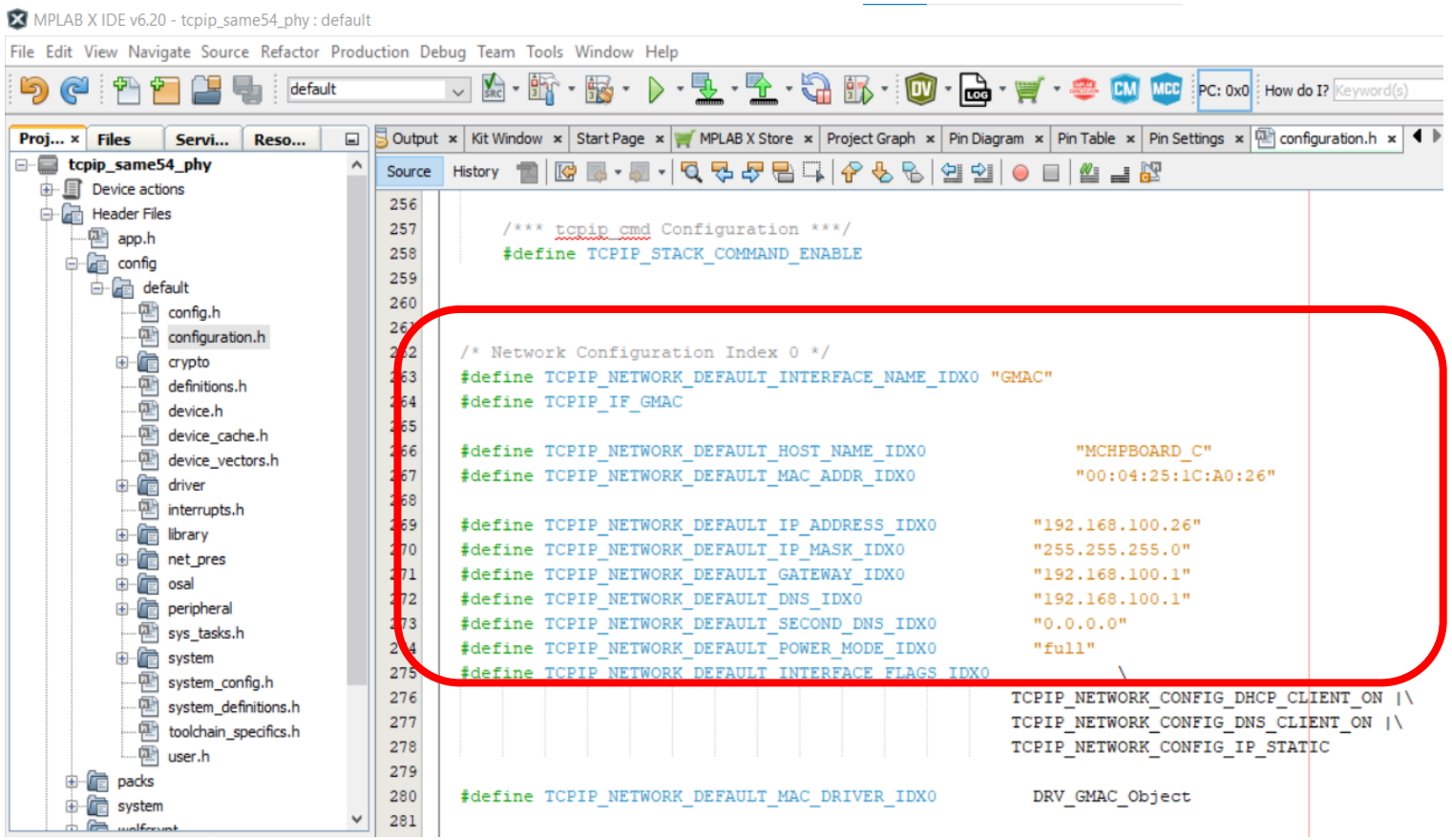
# Configuration.h

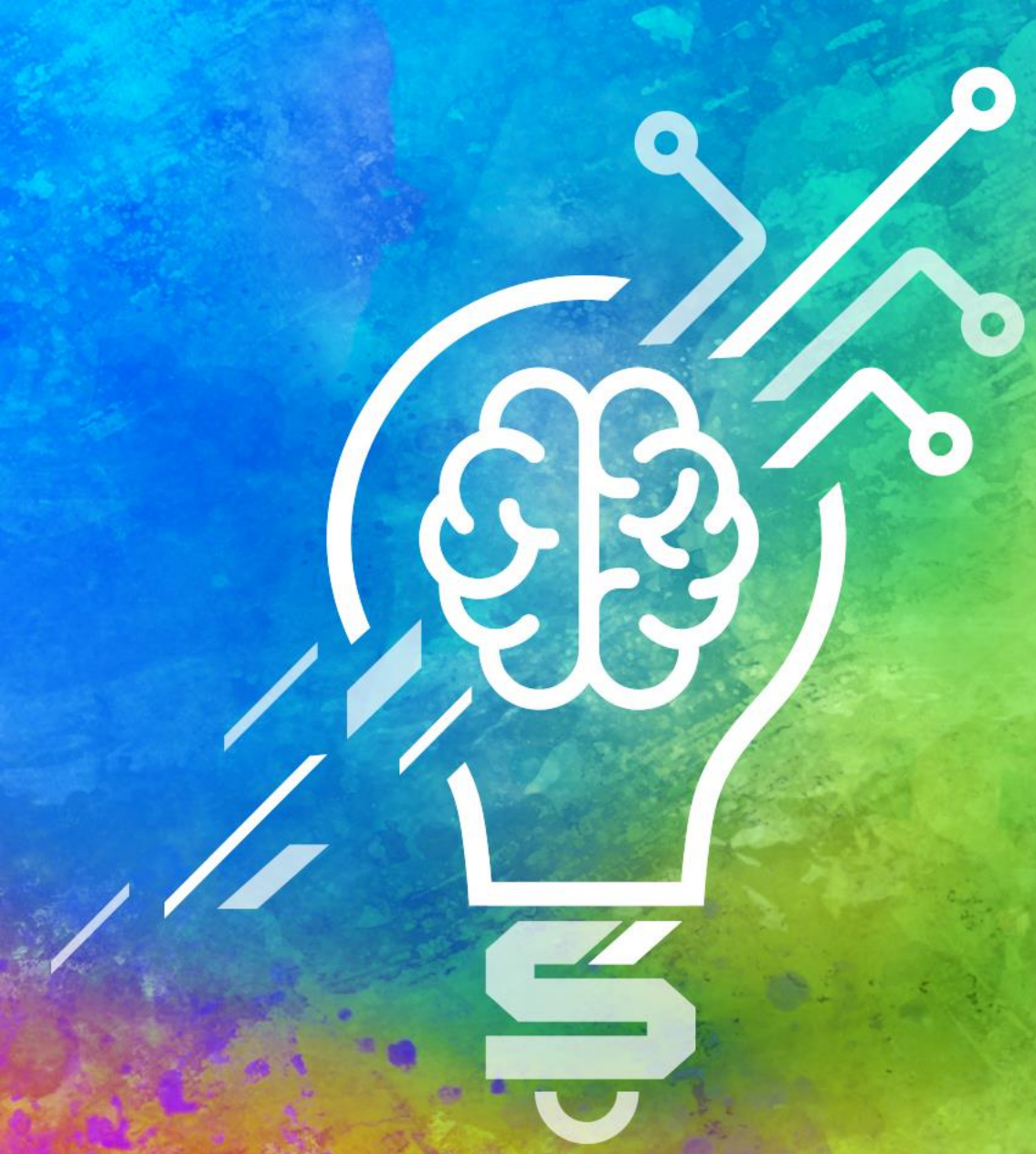
- **The Project is now populated**
- **Check the configuration.h Header File**
  - The PLCA settings and the Network Configuration settings can be seen in this file
  - Check that the PLCA settings are correct
  - These can be changed in this file if necessary
    - But any changes in this file will not be reflected back into the MCC Project Graph settings



# Configuration.h

- Check that the IP Address and MAC Address are correct
- These can be changed in this file if necessary
  - Again any changes in this file will not be reflected back into the MCC Project Graph settings





# LAB Part 2: 10BASE-T1S Device Programming and Network Evaluation

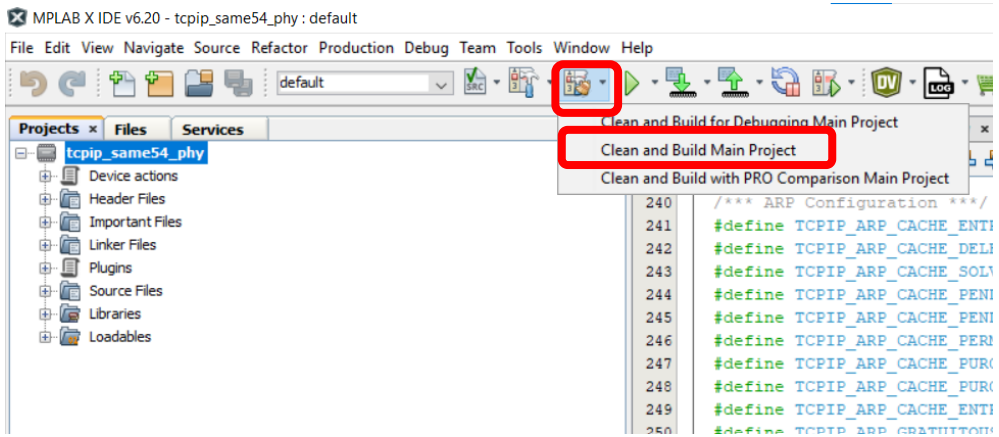
## Programming and Testing your 10BASE-T1S Network

# LAB: Learning Objectives

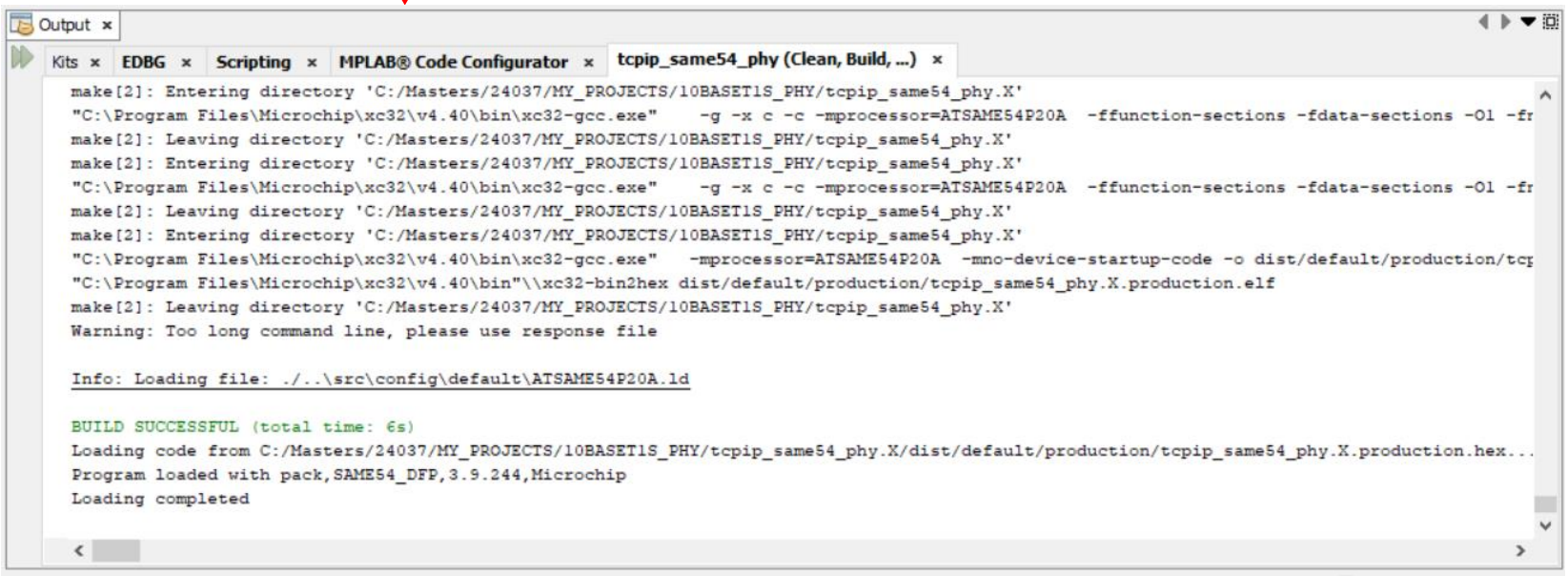
- Use MPLAB® Harmony to setup and configure a small 10BASE-T1S network.
- Reconfigure your application's network settings and understand their impact on the communication in 10BASE-T1S networks.
- Program your own application onto a microcontroller evaluation board to establish 10BASE-T1S network communication.
- Build an application on top of your project to add an OLED display to show the network and PLCA parameters
- Extend the physical network, that is the UTP bus line, without the need for any switches or repeaters.

# Build the Project and Program the Device

- **Clean and Build the Project**



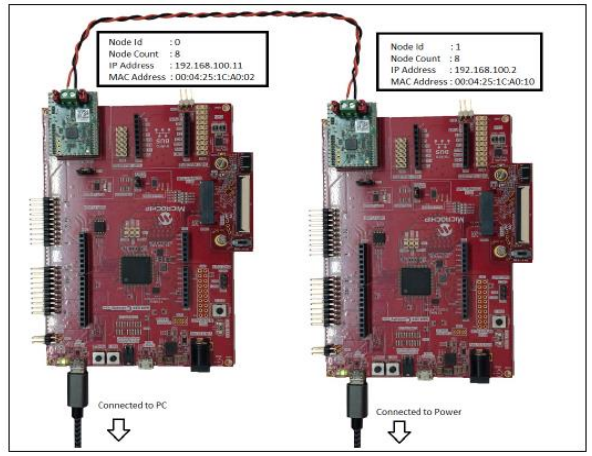

- **Check the output of the Build to make sure it was successful, similar to what is shown.**



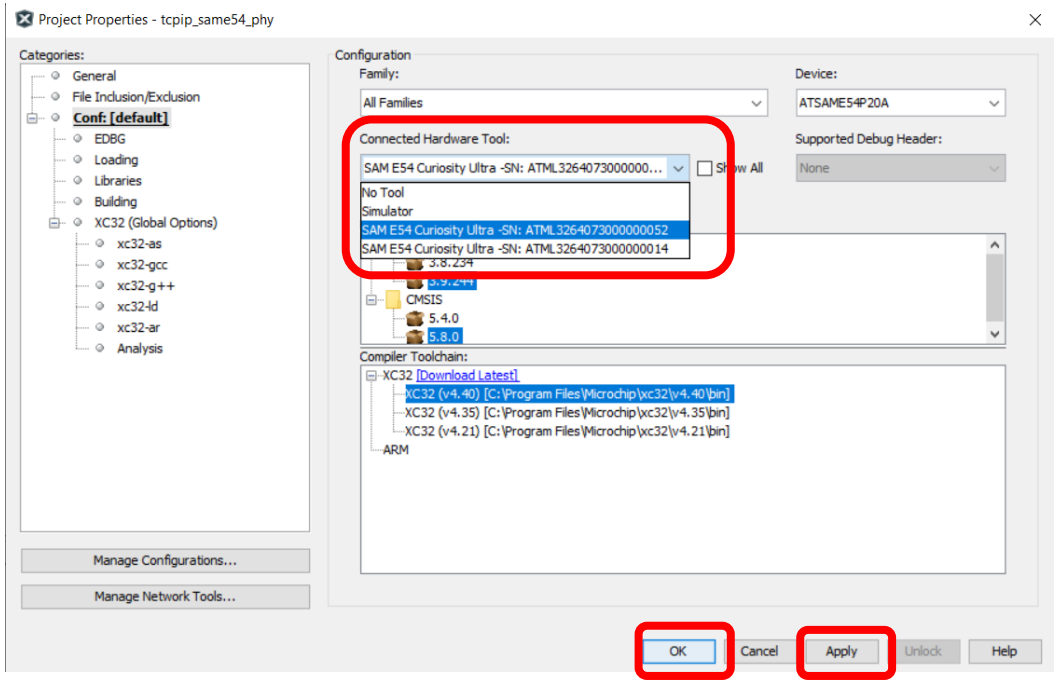

**Run out of time? Just use the Demo Project "C:\MASTERs\24037_NET1\LAB_PROJECTS\10BASET1S_PHY\tcpip_same54_phy.X" and use this to program your boards.**

# Program the Node 0 Board

- Connect the boards to the PC with the USB Cables



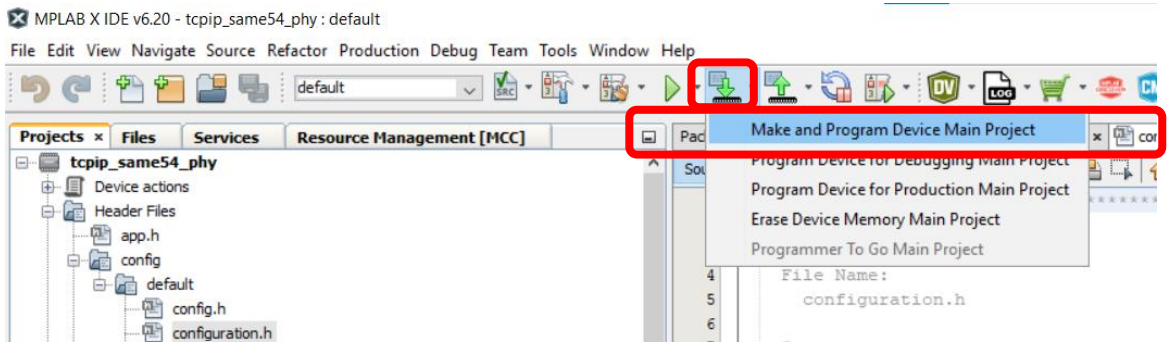

- In the Project Properties, check the Connected Hardware Tool list dropdown menu and select which board to program
- This board will be the PLCA Node 0 board
- Click on Apply then OK

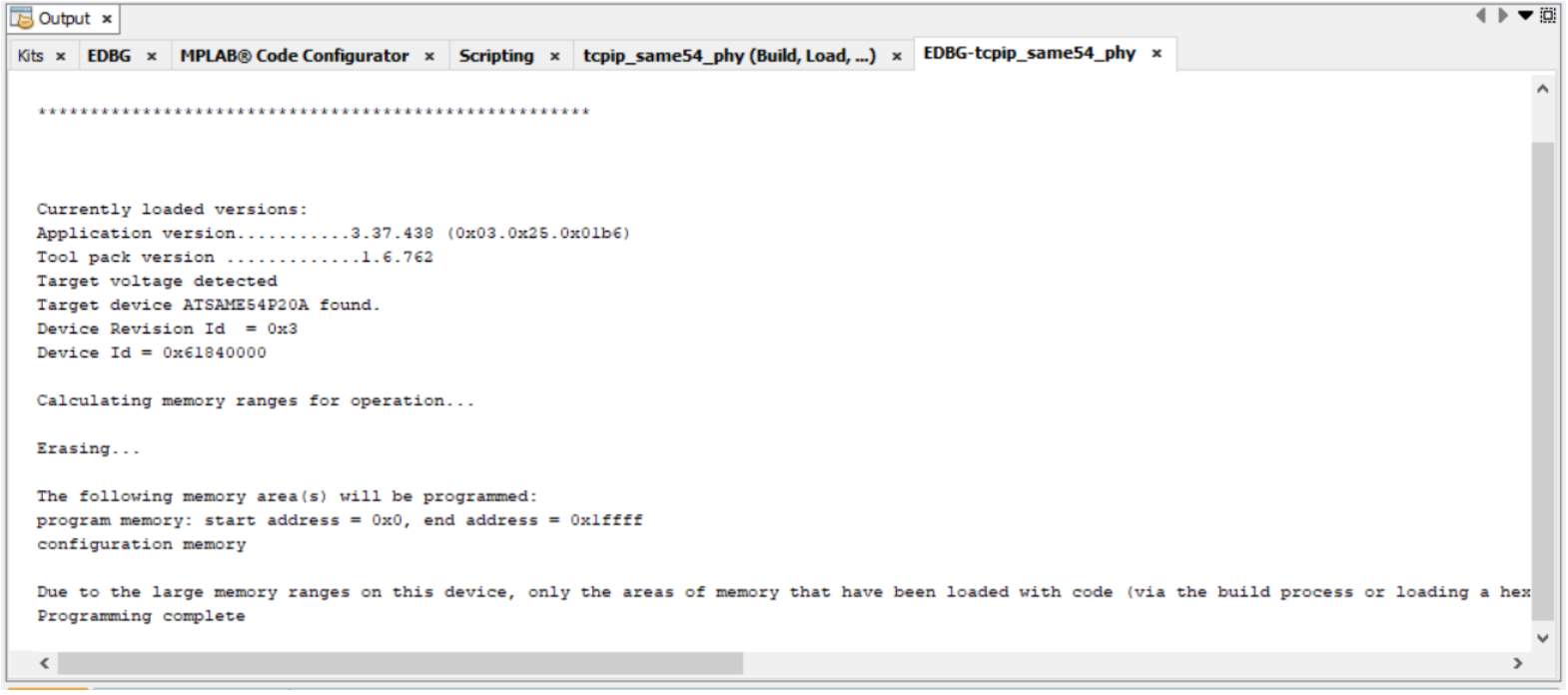


# Program the Node 0 Board

- **Use Make and Program Device to program the board.**

- **Check the output to make sure there were no errors**



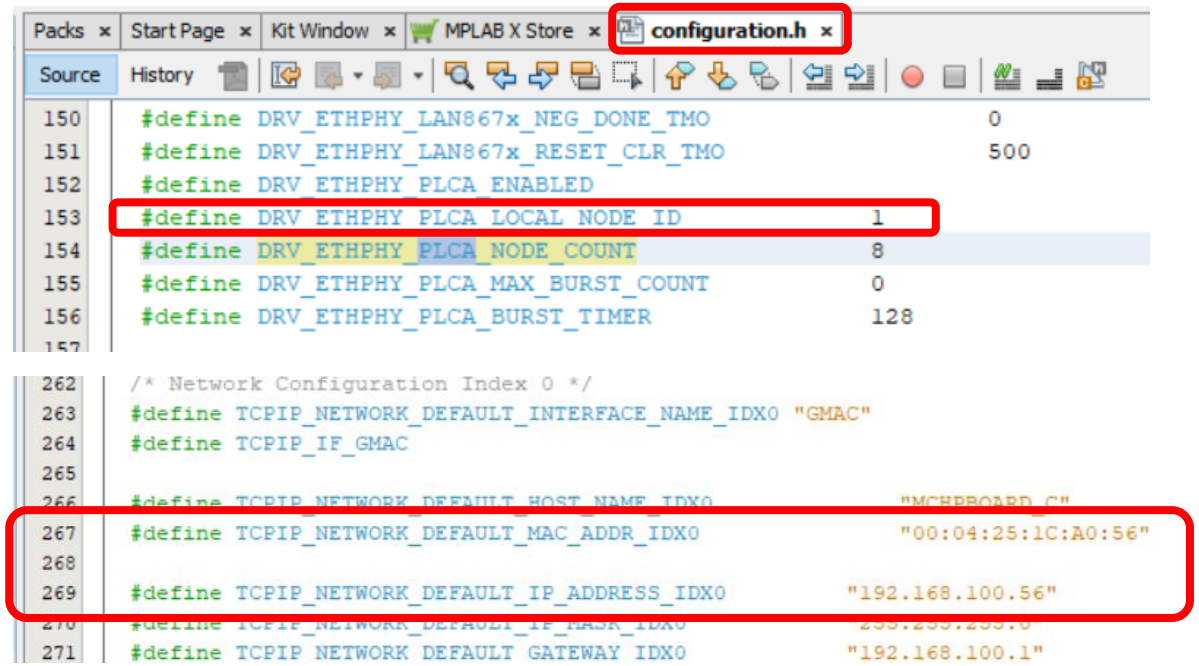



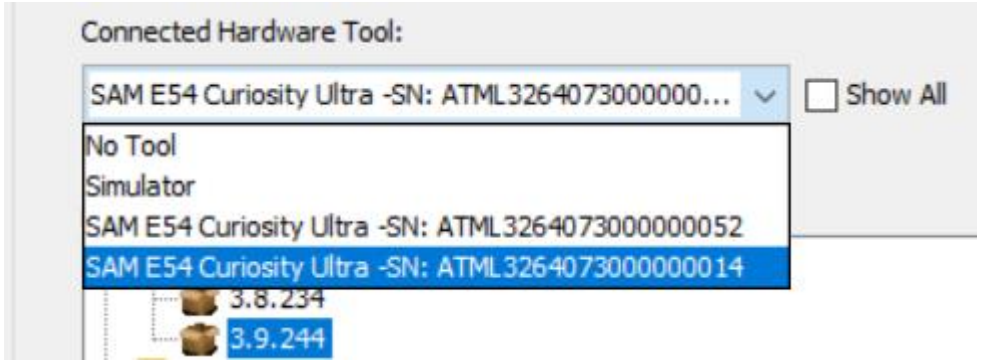



MICROCHIP

# Program the Node 1 Board

- In configuration.h, change the Node ID to 1 and change the MAC and IP Address so that they are distinct from the Node 0 Board
- In Project Properties, change the Connected Hardware Tool to the other board
- Repeat the “Make and Build Main Project” and “Make and Program Device” steps
- The boards are now programmed

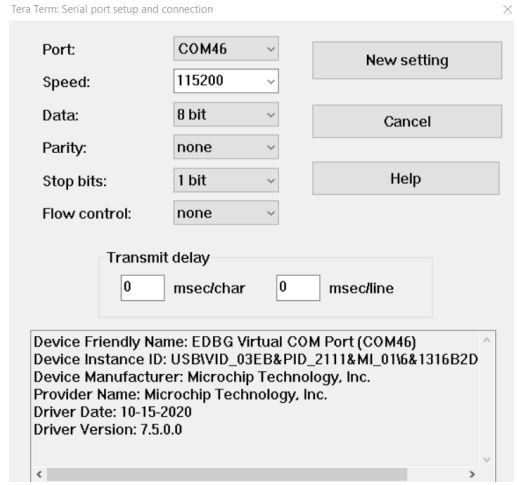


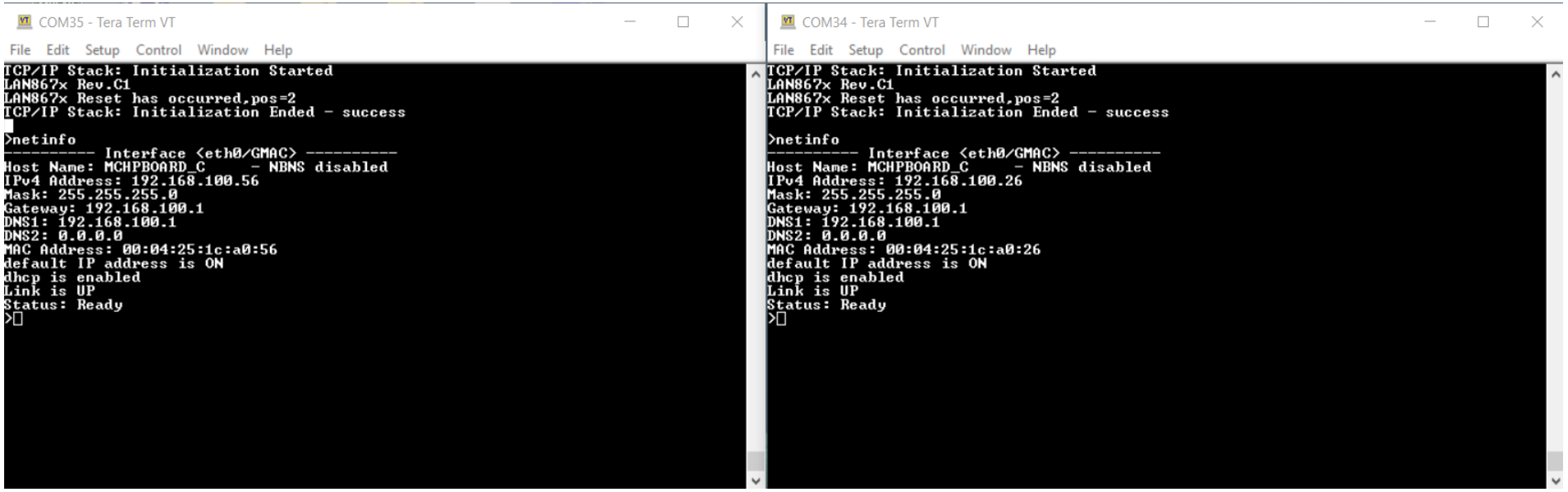




- Again any changes in configuration.h file will not be reflected back into the MCC Project Graph settings

# Test

- **Open a serial connection to both boards using the settings shown**
- **Reset both boards using the Reset button on the board**
  - You should see the TCP/IP Stack Initialization messages to show the stack is working
- **Use the "`netinfo`" command to show the Network Configuration**
  - Tip – wait a few moments after resetting the boards to allow the link to establish



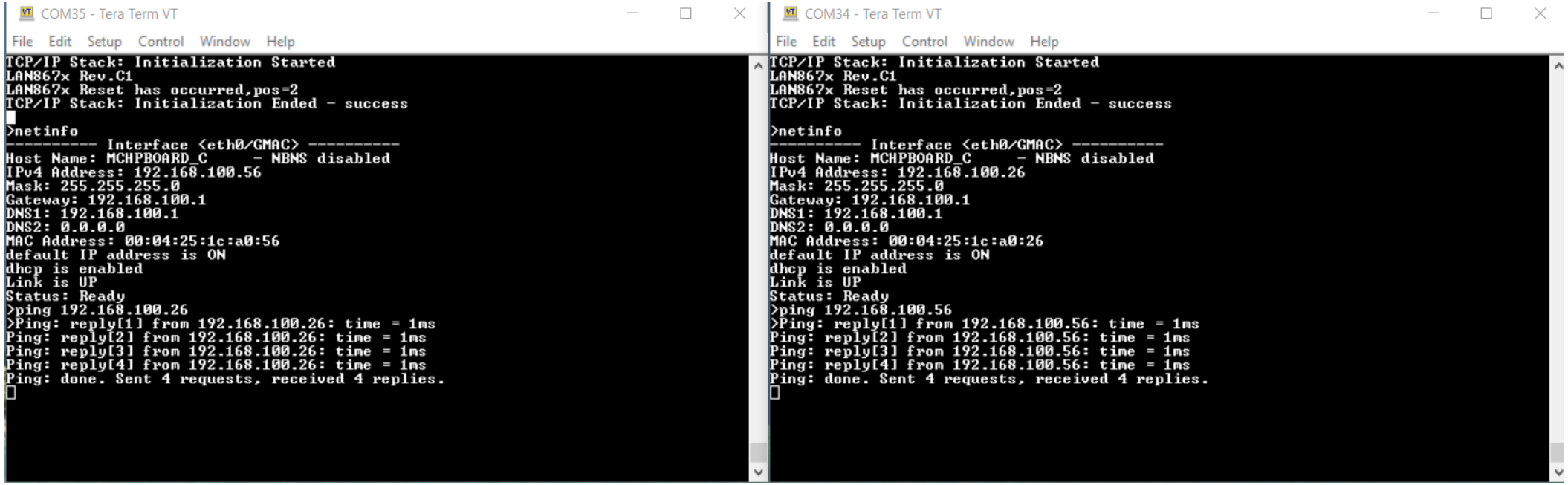



# Ping test

- **Use the "`ping`" command to check the link**

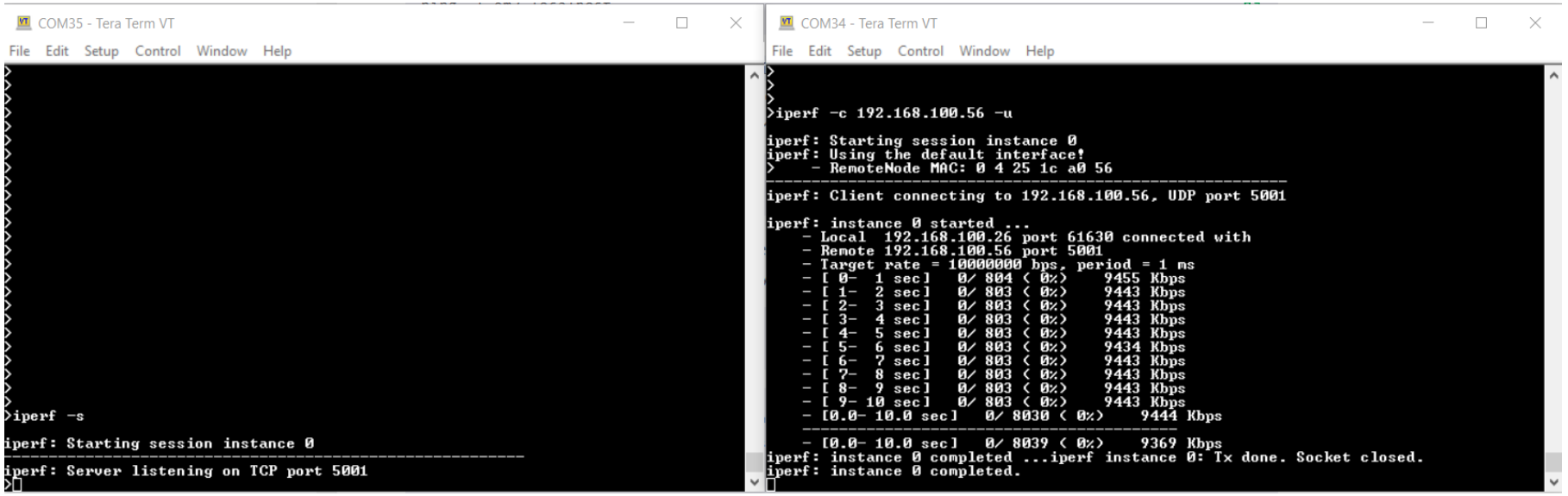
**Hint: there are a few options you can use with ping:**

| Option | Description |
|---|---|
| ping stop | Can be used any time during a ping to stop the ping |
| ping <address> n nPings | To send nPings number of pings |
| ping <address> t msPeriod | Change the time interval between pings |
| ping <address> s size | Specify the packet size |



# iperf

- On one terminal type "`iperf -s`" to initiate the iperf server on that node
- On the other terminal type "`iperf -c 192.168.100.xx -u -l 1418`" to test the speed of the link
- Hint use "-t 0" option to run iperf indefinitely
- Try other ethernet frame length options other than 1418, such as 700 and 300 to show the reduction of the bandwidth when the length of the frame is limited in this way



# LAB Part 3: 10BASE-T1S Project OLED Extension

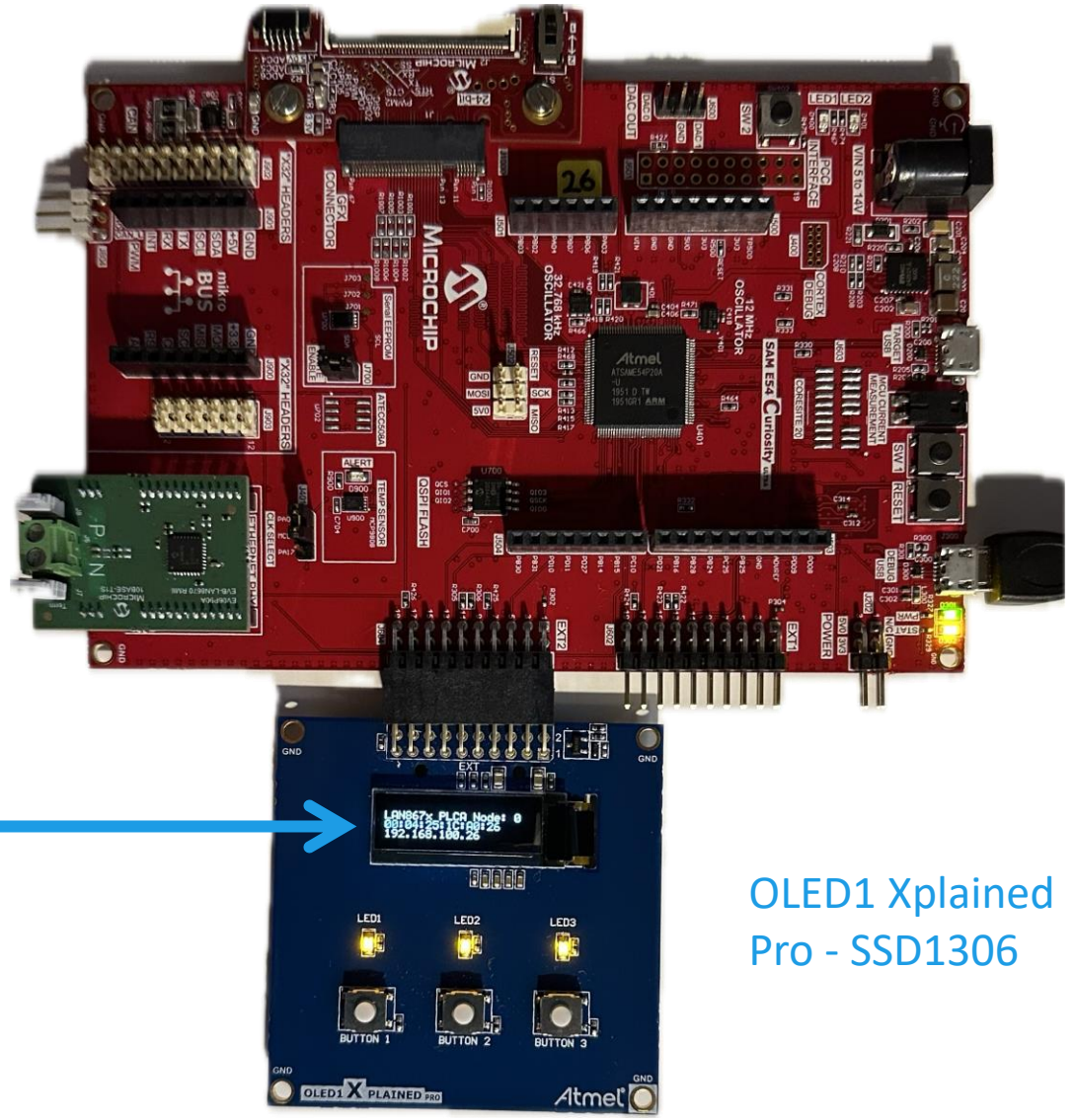
## Build an Application on your Project

# LAB: Learning Objectives

- Use MPLAB® Harmony to setup and configure a small 10BASE-T1S network.
- Reconfigure your application's network settings and understand their impact on the communication in 10BASE-T1S networks.
- Program your own application onto a microcontroller evaluation board to establish 10BASE-T1S network communication.
- Build an application on top of your project to add an OLED display to show the network and PLCA parameters
- Extend the physical network, that is the UTP bus line, without the need for any switches or repeaters.

# OLED Integration with 10BASE-T1S

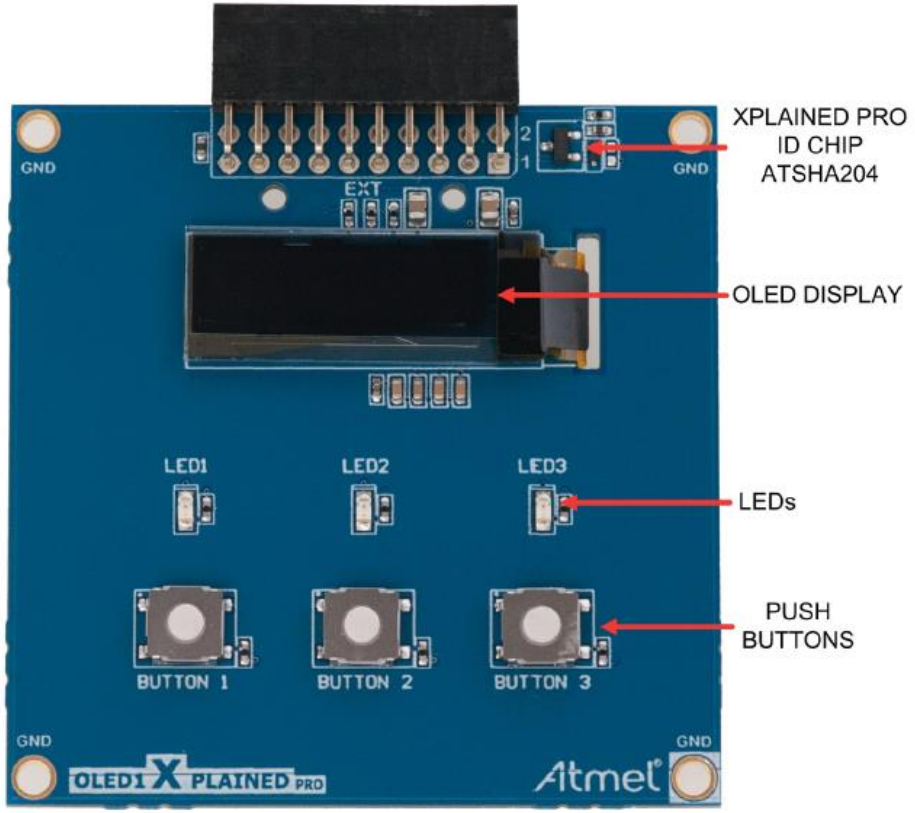
- The project presented here will enable the addition of a display to the hardware setup
- This will be useful for inspecting the IP address, MAC address and PLCA Node ID of each board.


OLED1 Xplained Pro - SSD1306

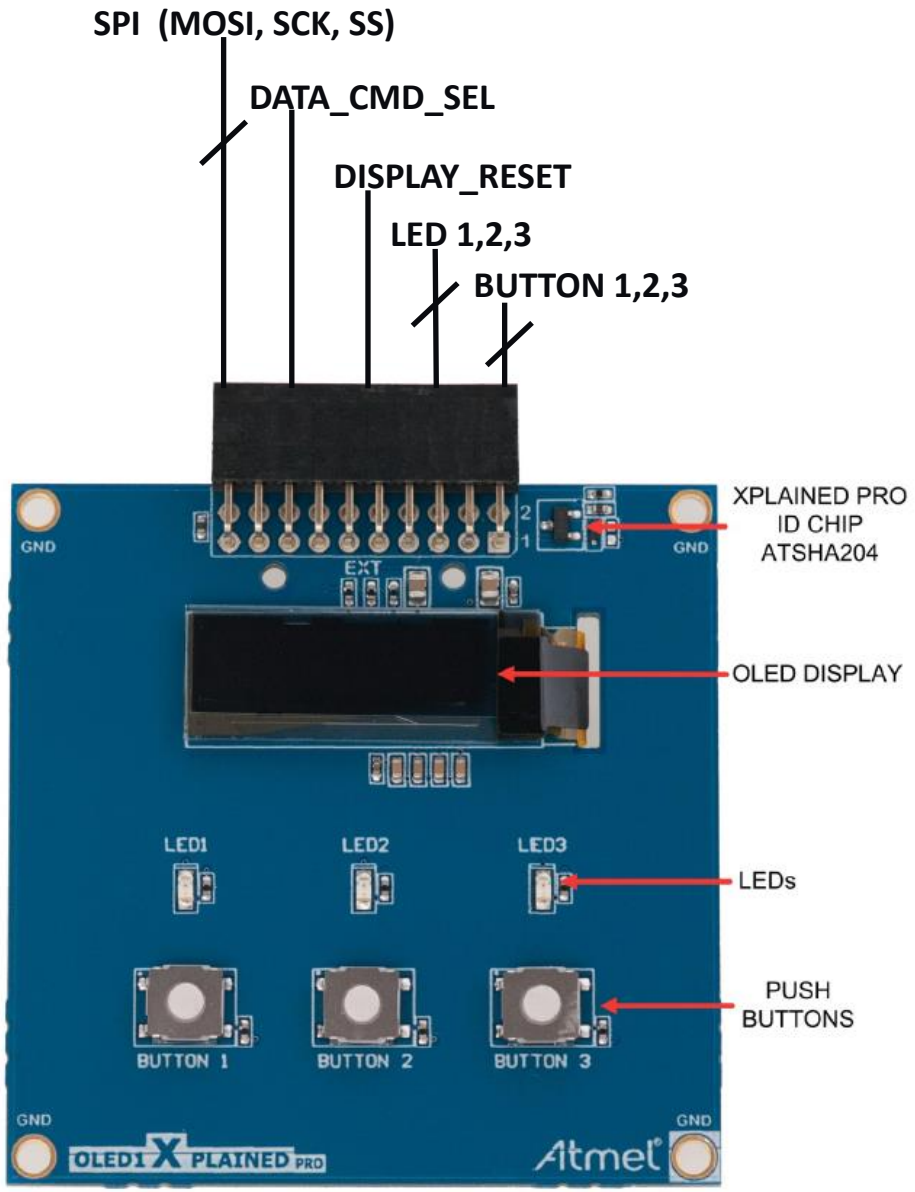
# OLED1 Overview

- The OLED display is driven by the SSD1306 controller, which takes input in SPI format and converts it to 8 bit parallel to drive the display itself.
- The ATSAME54 will need to have an SPI interface driver to send data to this bus
- We will add this in Harmony, and also the applicable driver for the controller
- We will then write a simple application that will send the IP Address, MAC Address and PLCA Node ID to the display



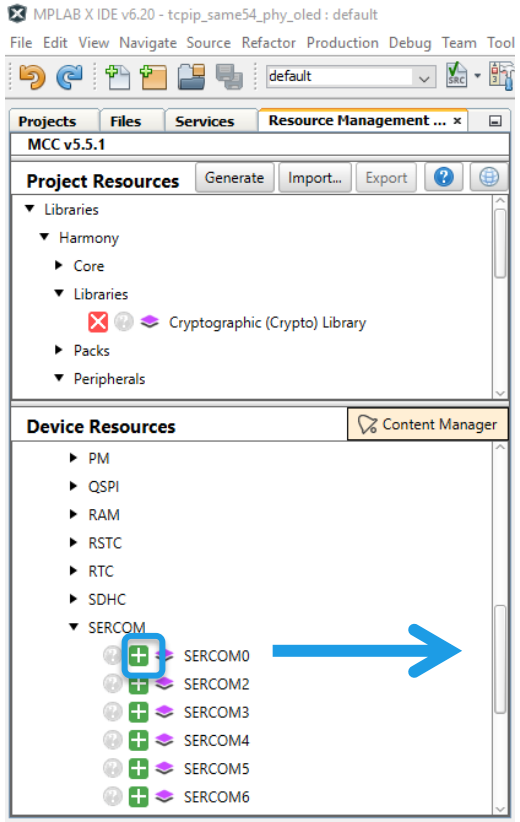
# OLED1 Overview

- The ATSAME54 will need to configure the pins to drive and receive input from the OLED1 Xplained board
- SPI is sent from the driver on the ATSAME54 to the controller
- DATA_CMD_SEL tells the controller if the SPI traffic is data for the display, or configuration information
- DISPLAY_RESET resets the controller
- LED 1,2,3 are driven from the ATSAME54 to the LEDs, and are active low
- BUTTON 1,2,3 are inputs from the OLED1 Explained board to the ATSAME54, and will be configured as input pins on the ATSAME54

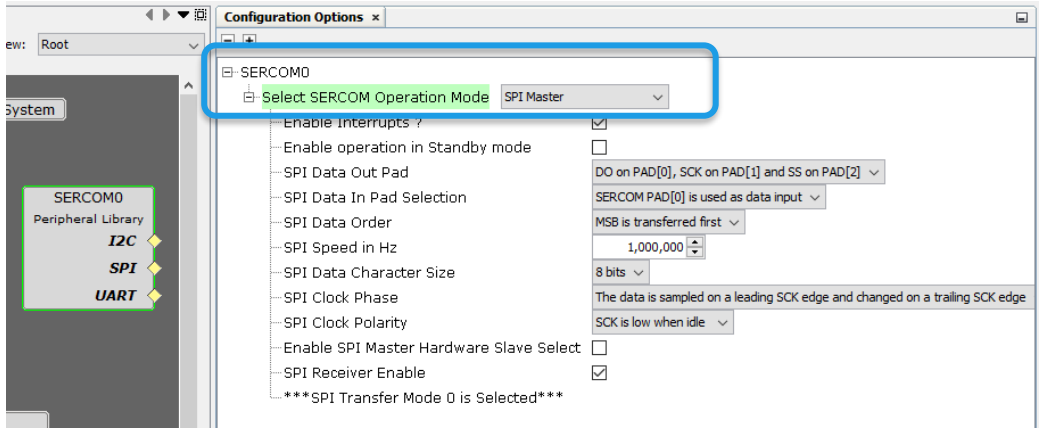


# Adding the SPI Device to your 10BaseT1S Project

- In Device Resources, under Libraries > Harmony > Peripherals > SERCOM, use the + symbol beside SERCOM0 to add it to the project
- In the Project Graph, select the SERCOM0 component and configure it as SPI Master

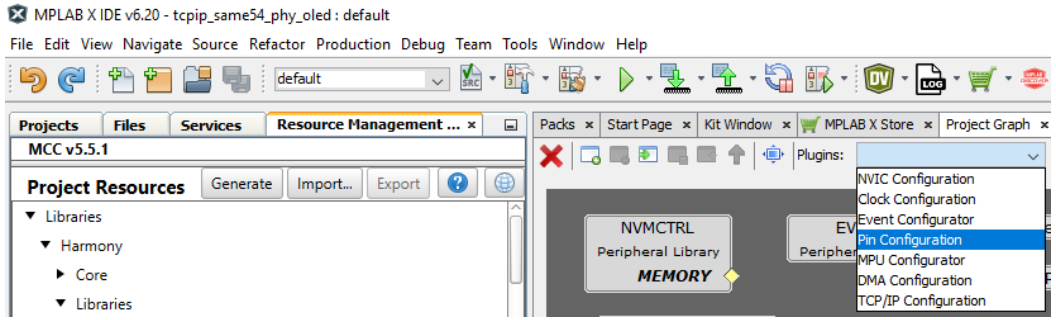



Note: In this example, we have created another identical project named tcpip_same54_phy_oled but you can simply continue with your original project that you have just created



# Use the Pin Configuration Tool to set up the pins for the OLED Xplained board

- As before, we will use the Pin Configuration tool to set up the extra pins that are needed



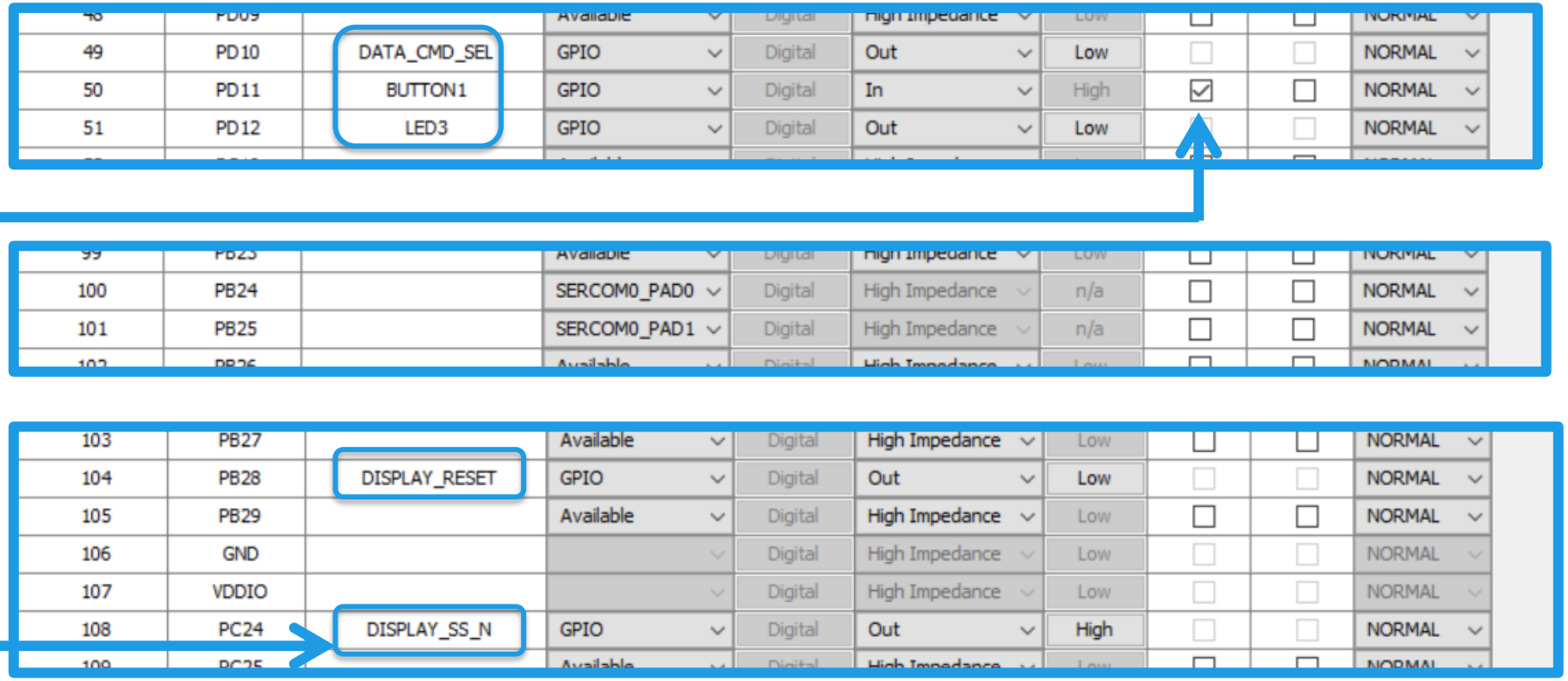

Note : the names used here need to be added, as they are used by the code in the application to refer to the configured pins

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 33 | PA08 | BUTTON2 | GPIO | Digital | In | High | ☑ | ☐ | NORMAL |
| 34 | PA09 | BUTTON3 | GPIO | Digital | In | High | ☑ | ☐ | NORMAL |
| 43 | PB14 | LED1 | GPIO | Digital | Out | Low | ☐ | ☐ | NORMAL |
| 44 | PB15 | LED2 | GPIO | Digital | Out | Low | ☐ | ☐ | NORMAL |

# Configure Pins

Pay attention to the Direction and Pull Up/Pull Down details, so that these pins are enabled with the correct functionality

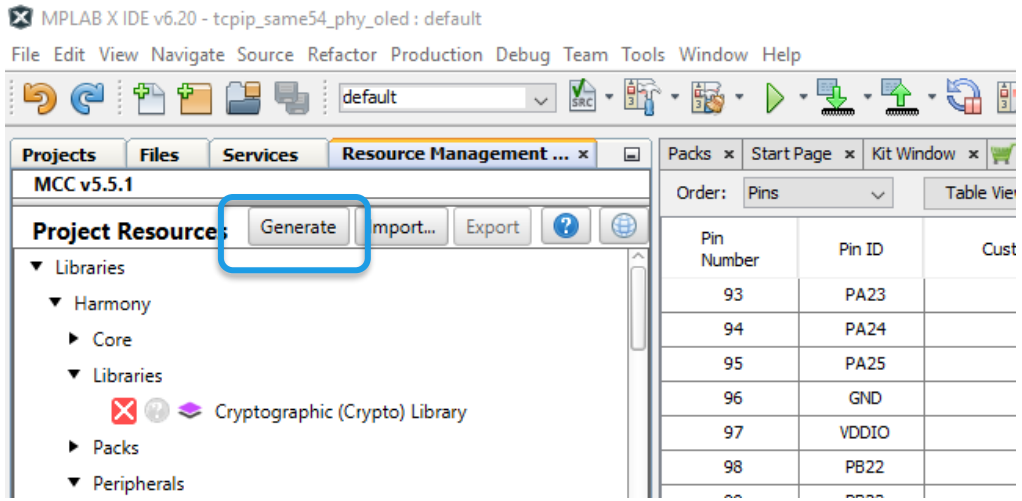
Note : the names used here need to be added, as they are used by the code in the application to refer to the configured pins



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 49 | PD10 | DATA_CMD_SEL | GPIO | Digital | Out | Low | | | NORMAL |
| 50 | PD11 | BUTTON1 | GPIO | Digital | In | High | ☑ | | NORMAL |
| 51 | PD12 | LED3 | GPIO | Digital | Out | Low | | | NORMAL |
| 100 | PB24 | | SERCOM0_PAD0 | Digital | High Impedance | n/a | | | NORMAL |
| 101 | PB25 | | SERCOM0_PAD1 | Digital | High Impedance | n/a | | | NORMAL |
| 103 | PB27 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 104 | PB28 | DISPLAY_RESET | GPIO | Digital | Out | Low | | | NORMAL |
| 105 | PB29 | | Available | Digital | High Impedance | Low | | | NORMAL |
| 106 | GND | | | Digital | High Impedance | Low | | | NORMAL |
| 107 | VDDIO | | | Digital | High Impedance | Low | | | NORMAL |
| 108 | PC24 | DISPLAY_SS_N | GPIO | Digital | Out | High | | | NORMAL |

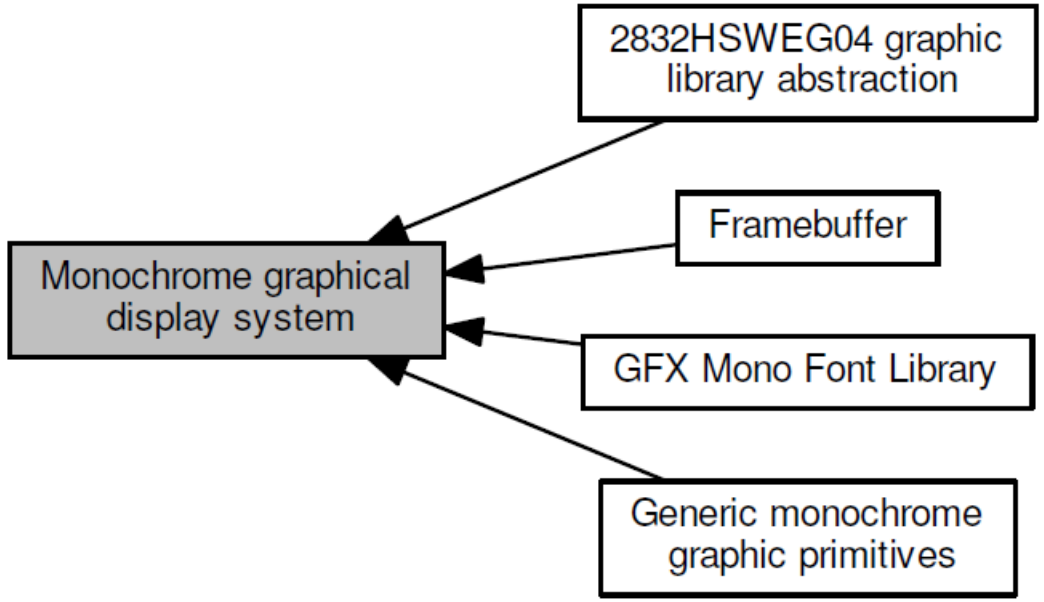
# Regenerate the Code

- **Use the Generate button to re-generate the code**

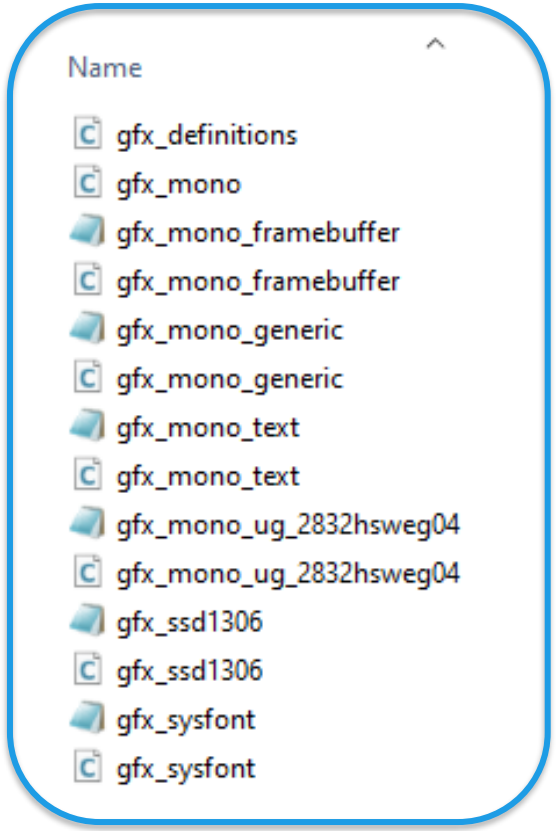


# Importing the graphics libraries into the project

- **The graphics library files need to be copied over from the pre-prepared project**
- **This project is located in C:\MASTERs\24037_NET1\LAB_PROJECTS\10BASET1S_PHY_OLED , and named tcpip_same54_phy_oled.X**

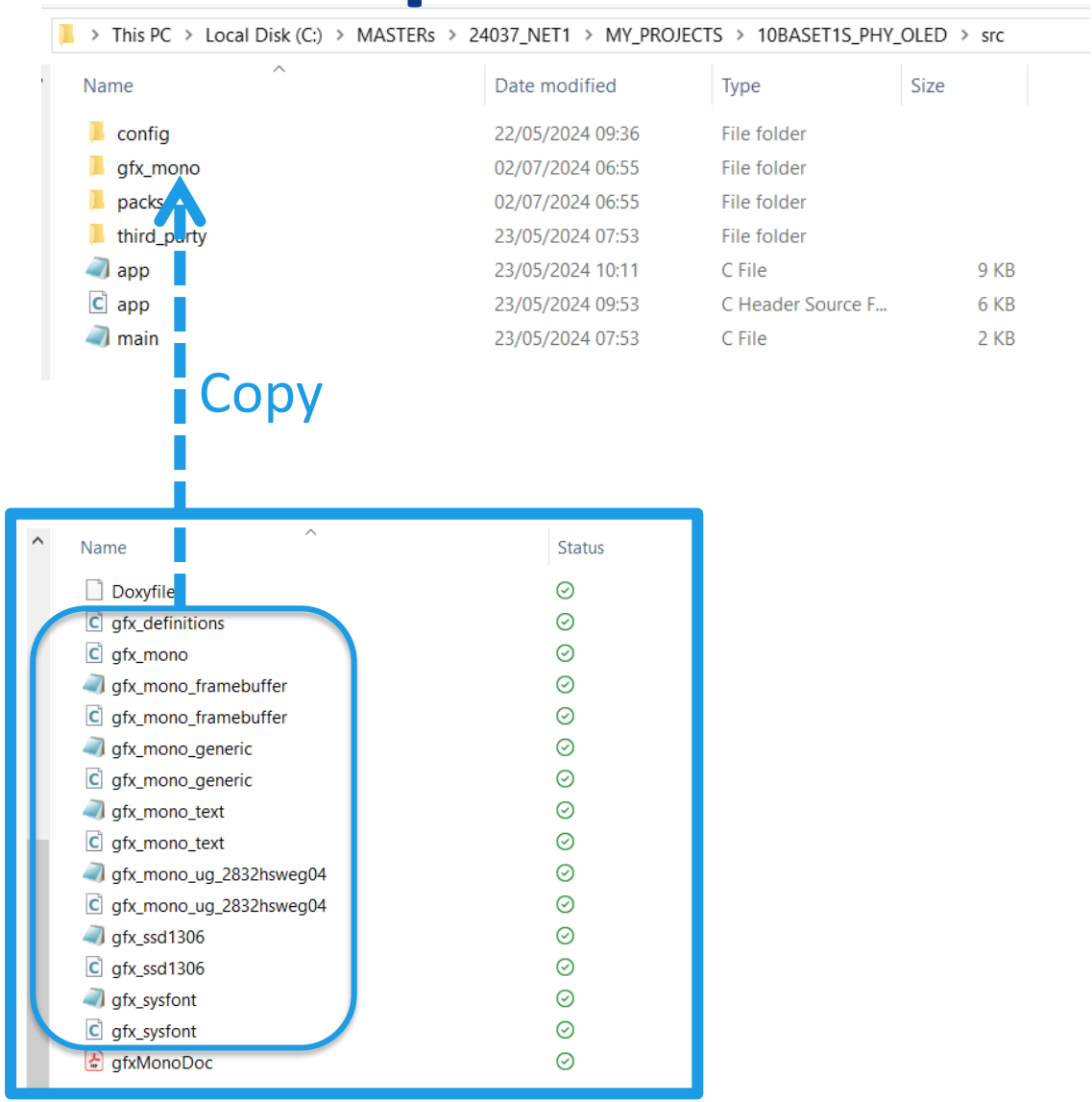



**Navigate to src/gfx_mono to see the graphics files:**



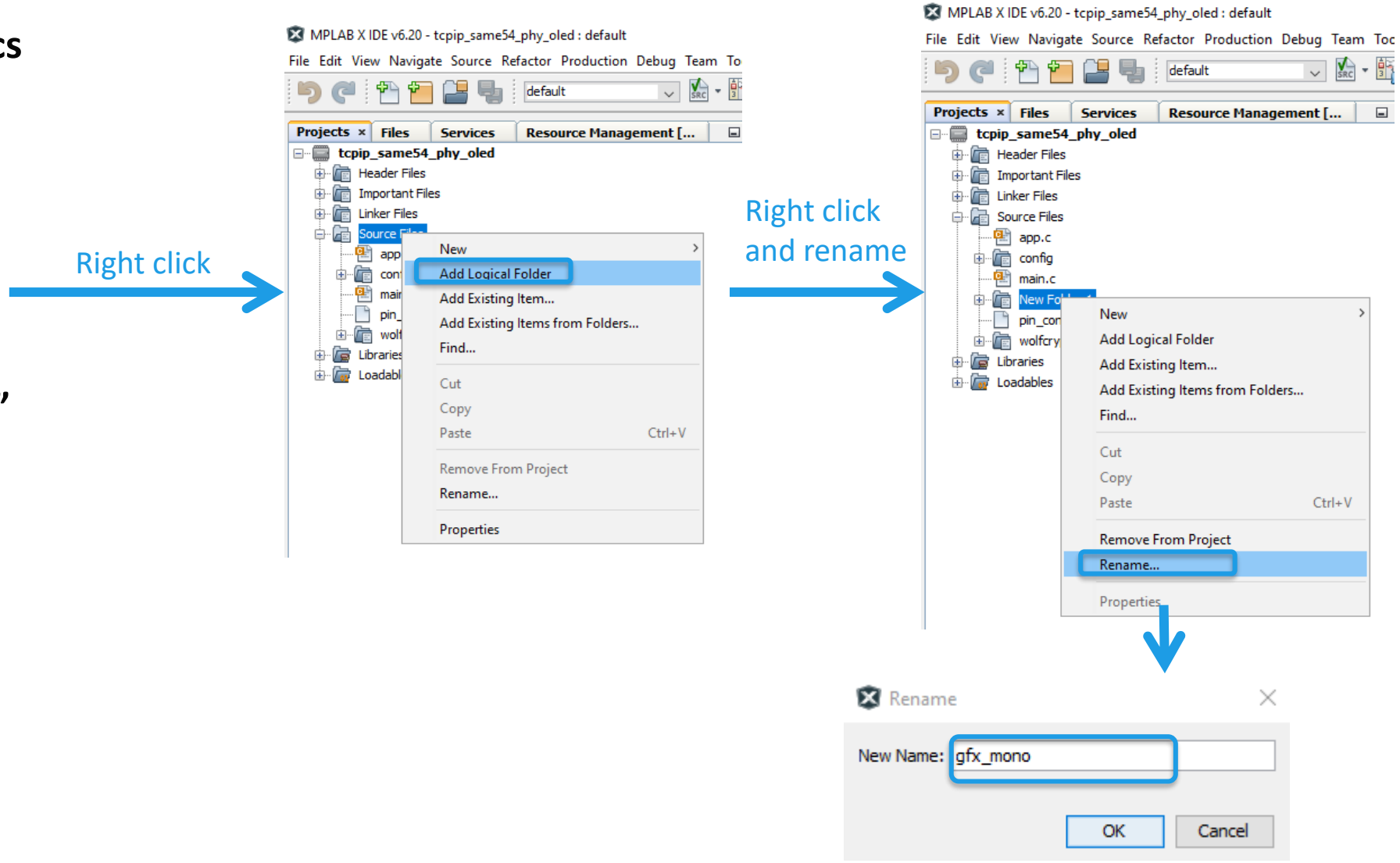
# Copy over Files with Windows Explorer

- Navigate using Windows Explorer to where you have created your project
- Descend into the src directory and create a folder named gfx_mono
- From the example project, copy in the files for the graphics library into this folder



# Add the files to the Project in MPLAB® X

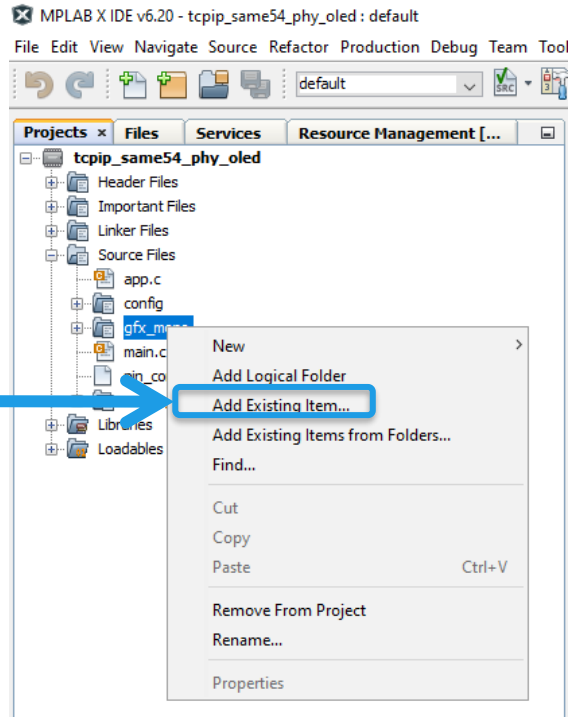
- In MPLAB® X, the graphics library files need to be added to the project.
- To do this, create a “Logical Folder” within the Source Files folder in your project
- Rename it to “gfx_mono”



# Add the files to the Project in MPLAB® X (contd)

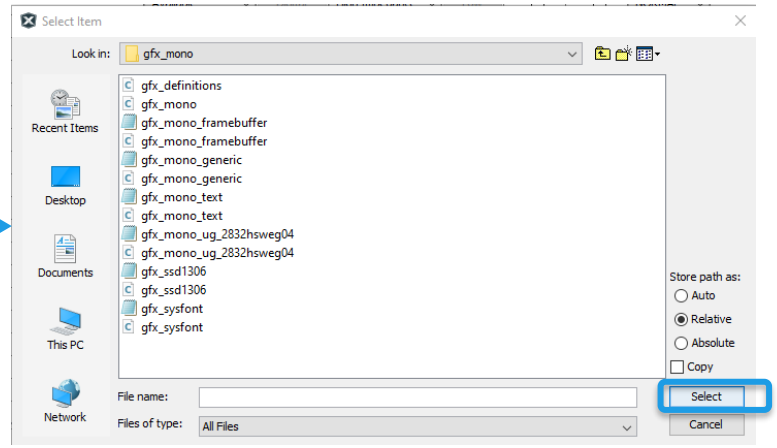
- Then add the files to this project
- This tells MPLAB® X that these files are part of the project and will be included in the compilation and build

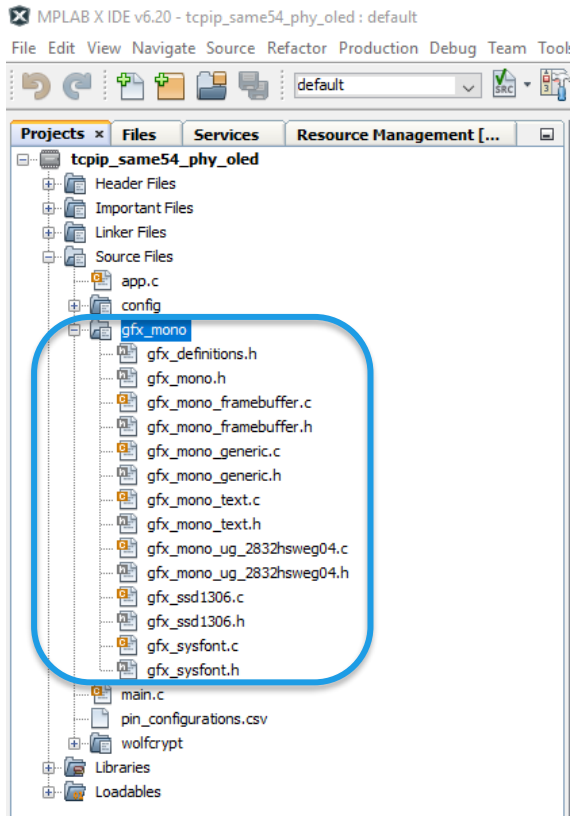
Check in the project folder in MPLAB® X that the files were added correctly:




Right click

Select all the files and click Select

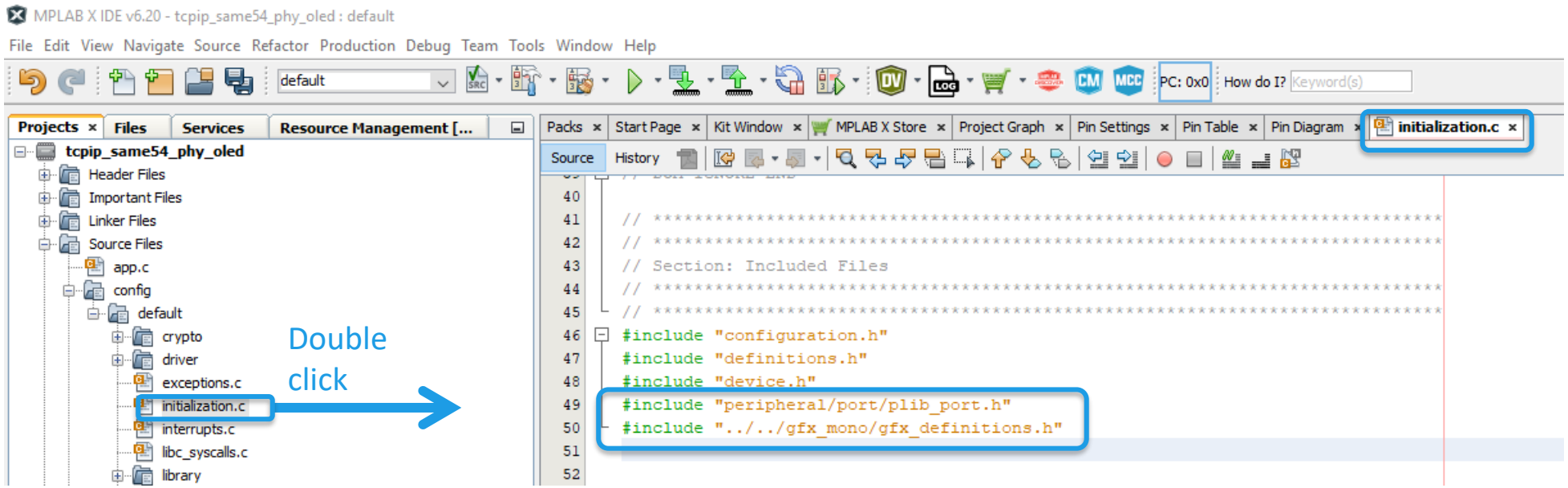


# Add the Code for the gfx_mono Initialization

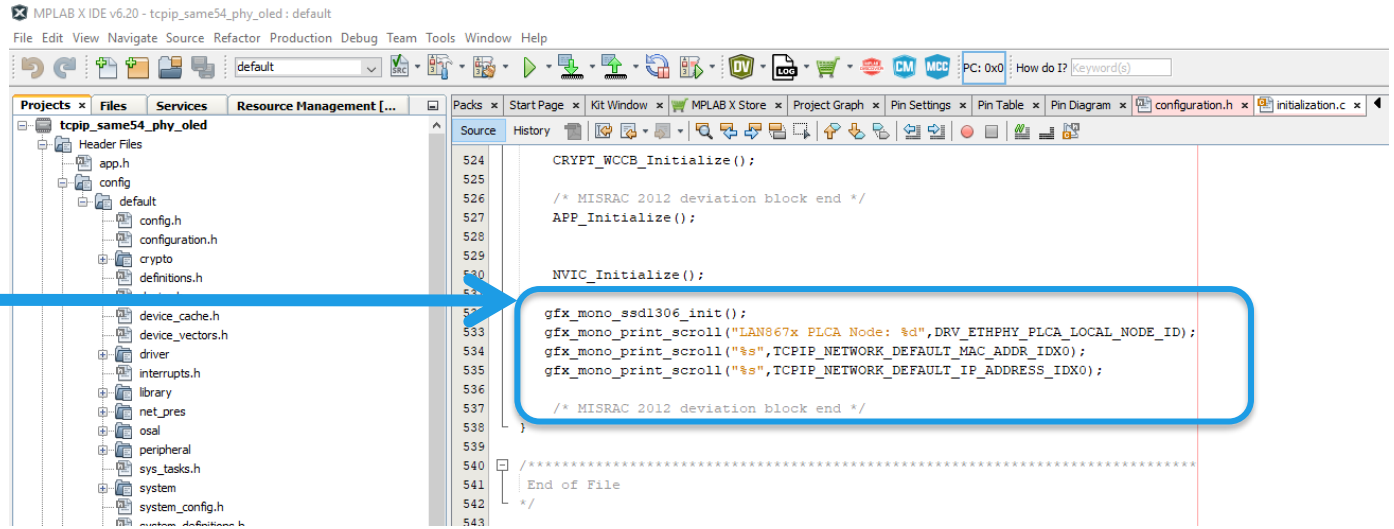
**In initialization.c add includes at the top**

```
#include "peripheral/port/plib_port.h"
#include "../../gfx_mono/gfx_definitions.h"
```

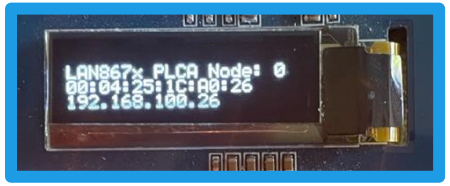


# Add the Code for the gfx_mono Initialization

In initialization.c add code at the end of SYS_Initialize() :

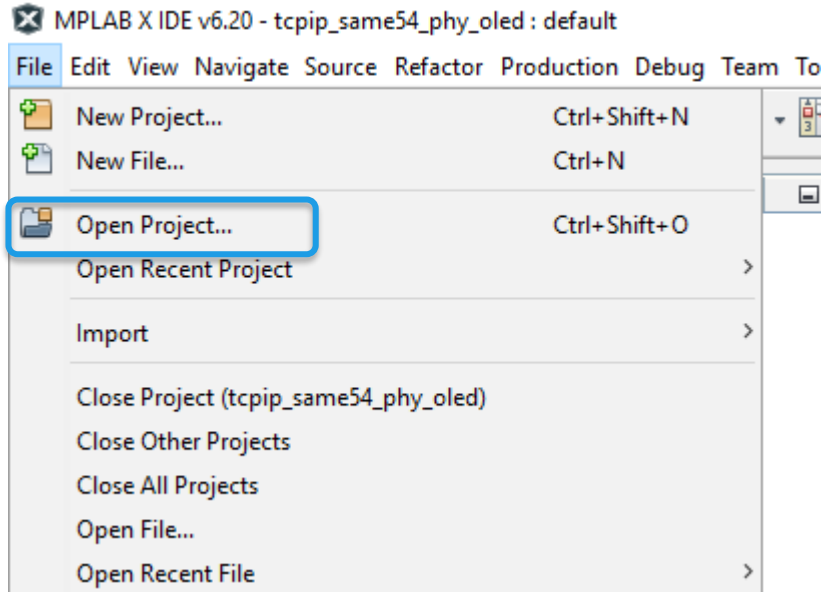



```
void SYS_Initialize ( void* data )
{
    ...
    ...
    APP_Initialize();
    NVIC_Initialize();

    gfx_mono_ssd1306_init();
    gfx_mono_print_scroll("LAN867x PLCA Node: %d",DRV_ETHPHY_PLCA_LOCAL_NODE_ID);
    gfx_mono_print_scroll("%s",TCPIP_NETWORK_DEFAULT_MAC_ADDR_IDX0);
    gfx_mono_print_scroll("%s",TCPIP_NETWORK_DEFAULT_IP_ADDRESS_IDX0);

}
```



# Enable Register Write

- Now we will add some code to configure one of the pin registers in the PHY so that the PHY daughter board can work together with the OLED daughter board on the SAM Curiosity board
- In MPLAB® X, open the Demo Project which is located in C:\MASTERs\24037_NET1\LAB_PROJECTS\10BASET1S_PHY_OLED, named tcpip_same54_phy_oled.X

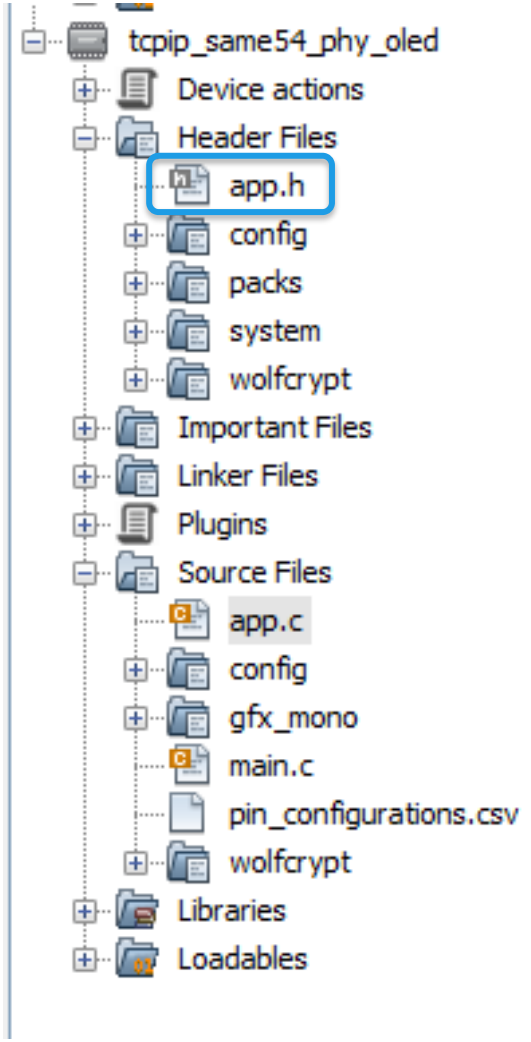



```
60
61  typedef enum
62  {
63      /* Application's state machine's initial state. */
64
65      APP_MIIM_INIT = 0,
66      APP_WAIT_STACK_INIT,
67      APP_READ_OPERATION_MODE,
68      APP_READ_PLCA_CONFIGURATION,
69      APP_WRITE_PLCA_CONFIGURATION,
70      APP_MIIM_CLOSE,
71      APP_STATE_SERVICE_TASKS,
72  } APP_STATES;
73
74
```

# Enable Register Write

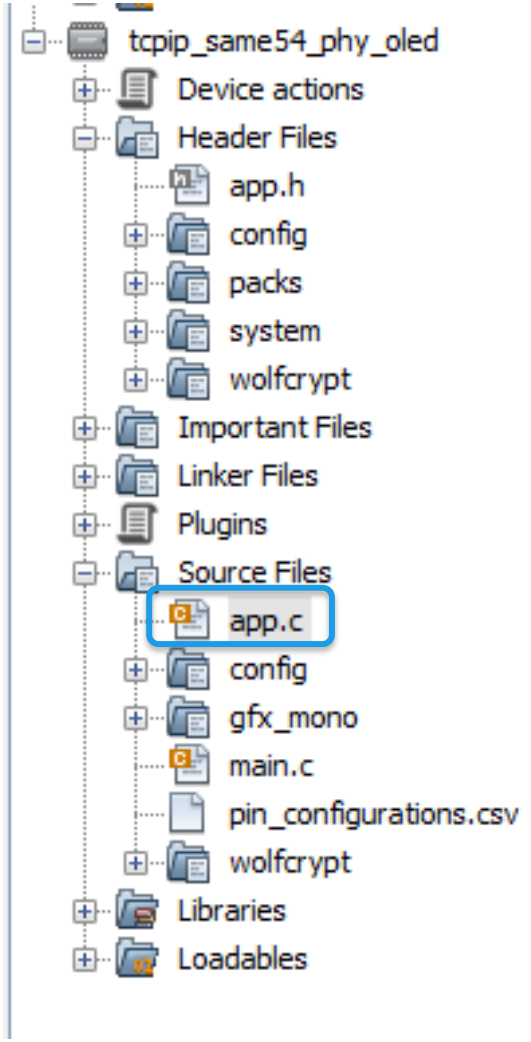
- In your own project, open the Header File app.h, and also open the Header File app.h from the Demo Project.
- Copy the enumeration function code “APP_STATES” shown here to replace the APP_STATES enumeration function in the app.h of your own project

```
60
61  typedef enum
62  {
63      /* Application's state machine's initial state. */
64
65      APP_MIIM_INIT = 0,
66      APP_WAIT_STACK_INIT,
67      APP_READ_OPERATION_MODE,
68      APP_READ_PLCA_CONFIGURATION,
69      APP_WRITE_PLCA_CONFIGURATION,
70      APP_MIIM_CLOSE,
71      APP_STATE_SERVICE_TASKS,
72  } APP_STATES;
73
74
```



# Enable Register Write

- In your own project, open the Source File app.c, and also open the Source File app.c from the Demo Project.
- This time copy over the entire contents of this file to replace the contents of the app.c in your own project
- The functions in this file write to the registers in the LAN8670 to configure one of the pins correctly to work with the OLED daughter board.



# State Machine for Register Read/Write

- The state machine in app.c enables read and write access to the registers in the LAN8760 PHY.
- A register write is required in order to configure one of the PHY pins to high impedence so that it does not interfere with the operation of the OLED, this is the part of the code that does this:

```
151
152         case APP_READ_OPERATION_MODE:
153         {
154           opRes = Write_Phy_Register(&clientObj, 0, 0x1F0011, 0xC000);
155
156           if (opRes < 0) {
157             SYS_CONSOLE_PRINT("Setting PHY_PINCTRL failed\r\n", data);
158
159           } else if (opRes == DRV_MIIM_RES_OK) /* Check operation is completed. */ {
160             appData.state = APP_MIIM_CLOSE;
161           }
162           break;
163         }
164
```

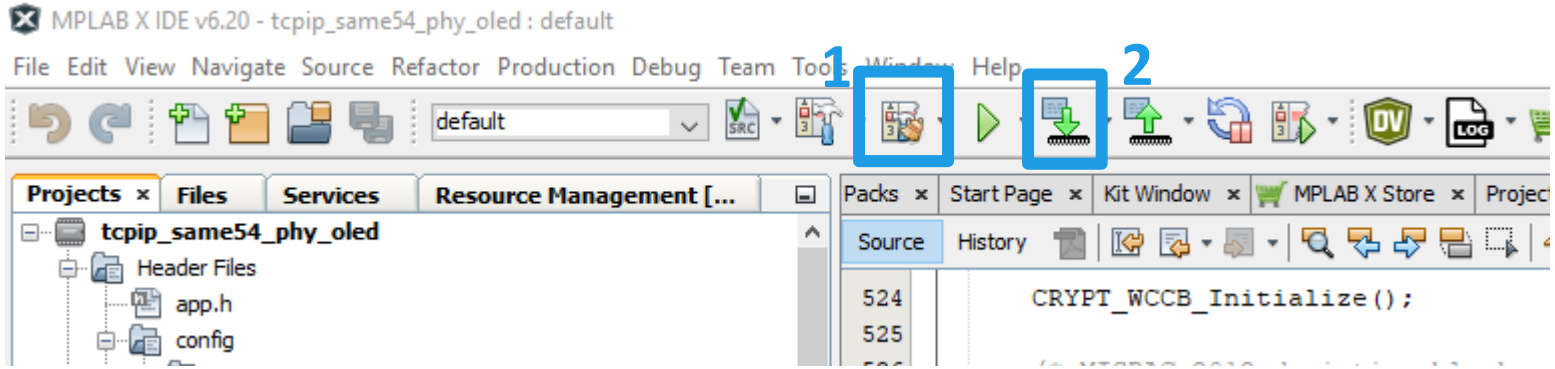




MICROCHIP

# Next step...program the boards

**Clean and Build Project (1) , than Program the first board (2):**

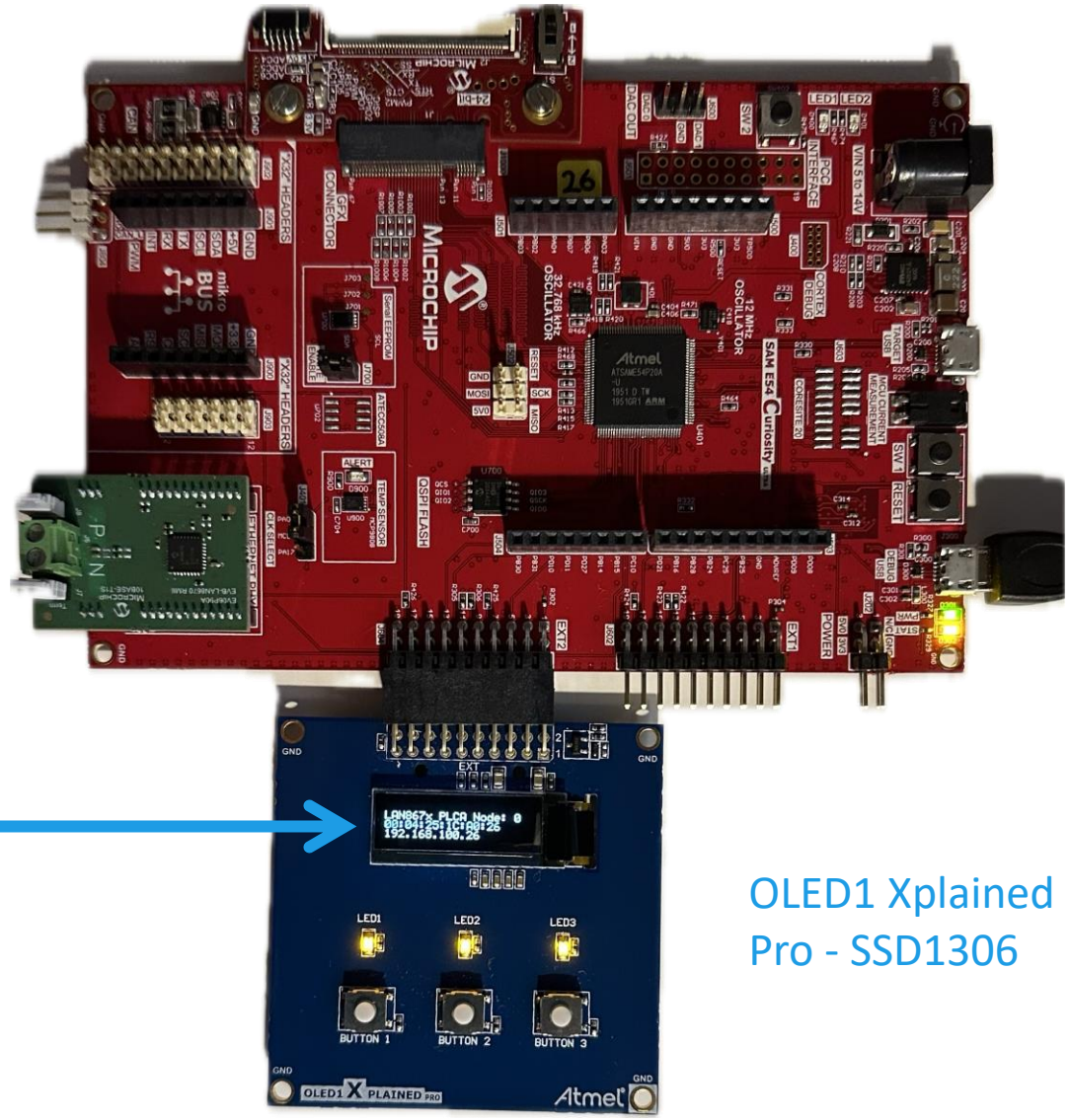



**After this, program the second board – but don't forget to change the MAC and IP address, along with the PLCA Node ID in configuration.h (as before) before you program the second board.**

**Run out of time? Just use the Demo Project "C:\MASTERs\24037_NET1\LAB_PROJECTS\10BASET1S_PHY_OLED\tcpip_same54_phy_oled.X" and use this to program your boards.**

# Result

- **The OLED will now show the IP Address, MAC Address and PLCA Node ID for each board**


OLED1 Xplained Pro - SSD1306

# LAB Part 4: 10BASE-T1S Network Expansion

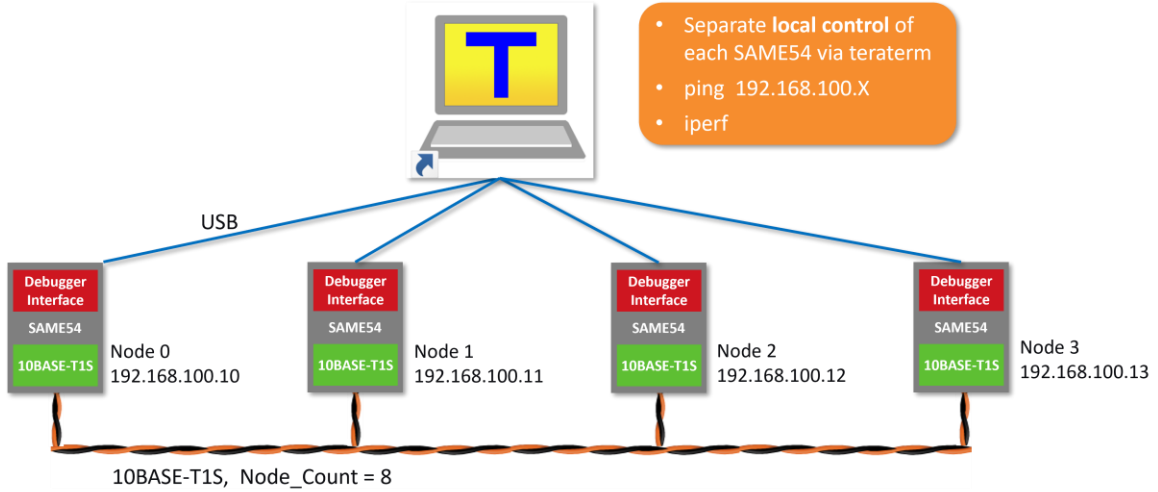
## Create a Multi-drop Topology

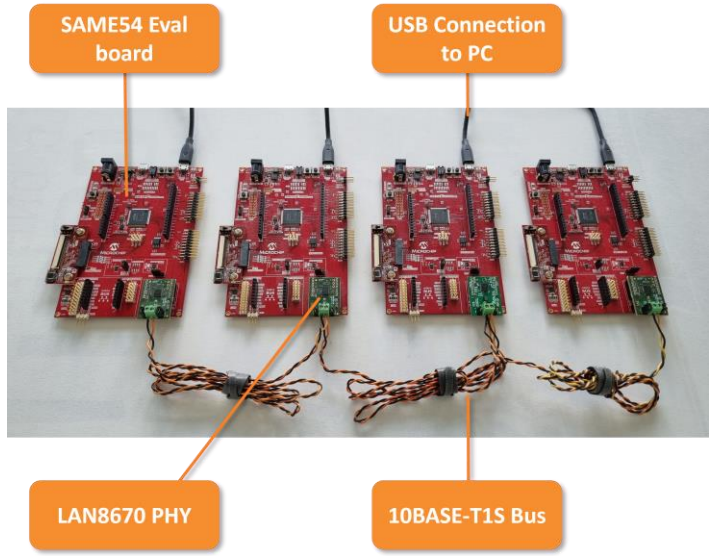
# Learning Objectives

- Use MPLAB® Harmony to setup and configure a small 10BASE-T1S network.
- Reconfigure your application's network settings and understand their impact on the communication in 10BASE-T1S networks.
- Program your own application onto a microcontroller evaluation board to establish 10BASE-T1S network communication.
- Build an application on top of your project to add an OLED display to show the network and PLCA parameters
- Extend the physical network, using an Unshielded Twisted Pair bus line, without the need for any switches or repeaters.

# Extending the Network

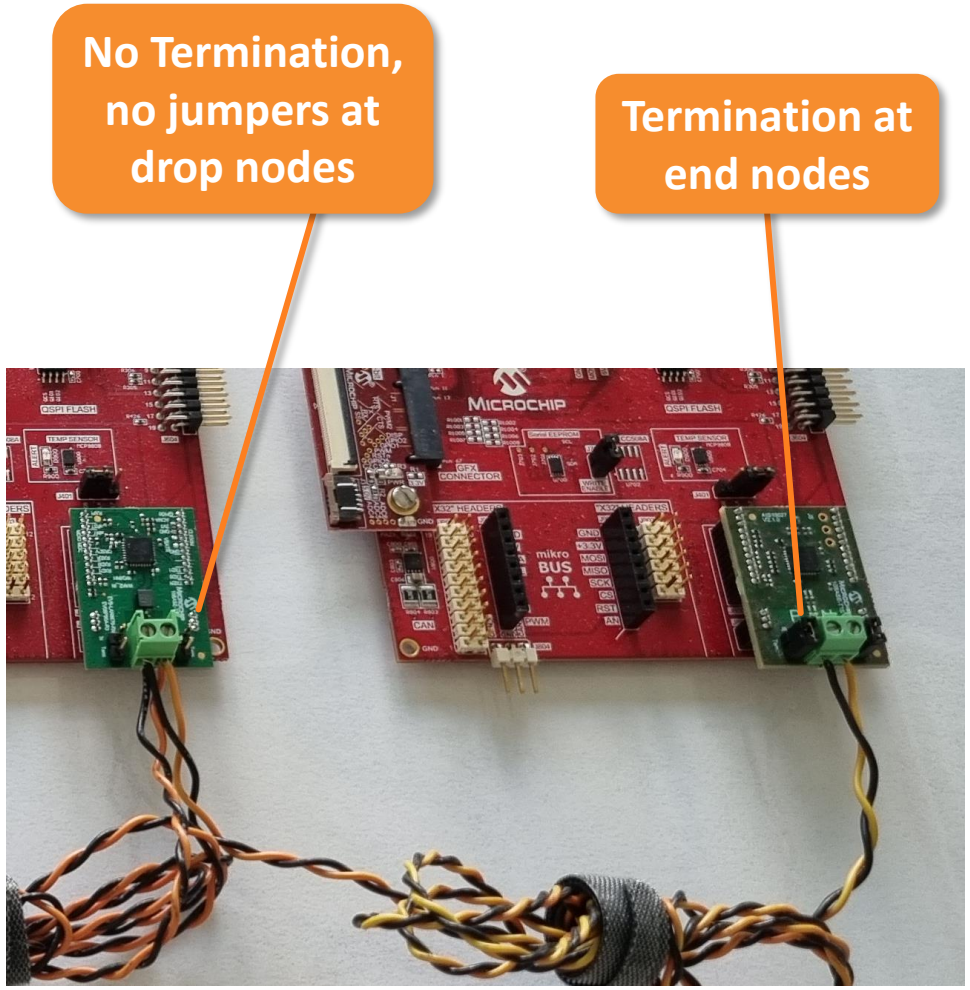
- You can work with a neighbour, or neighbours, to set up a multidrop network
- Use your own pair of boards and theirs, to set up 4, 6, 8 or more nodes
- Connect each board either to one laptop, or to a number of laptops for the terminal window using the USB Cables

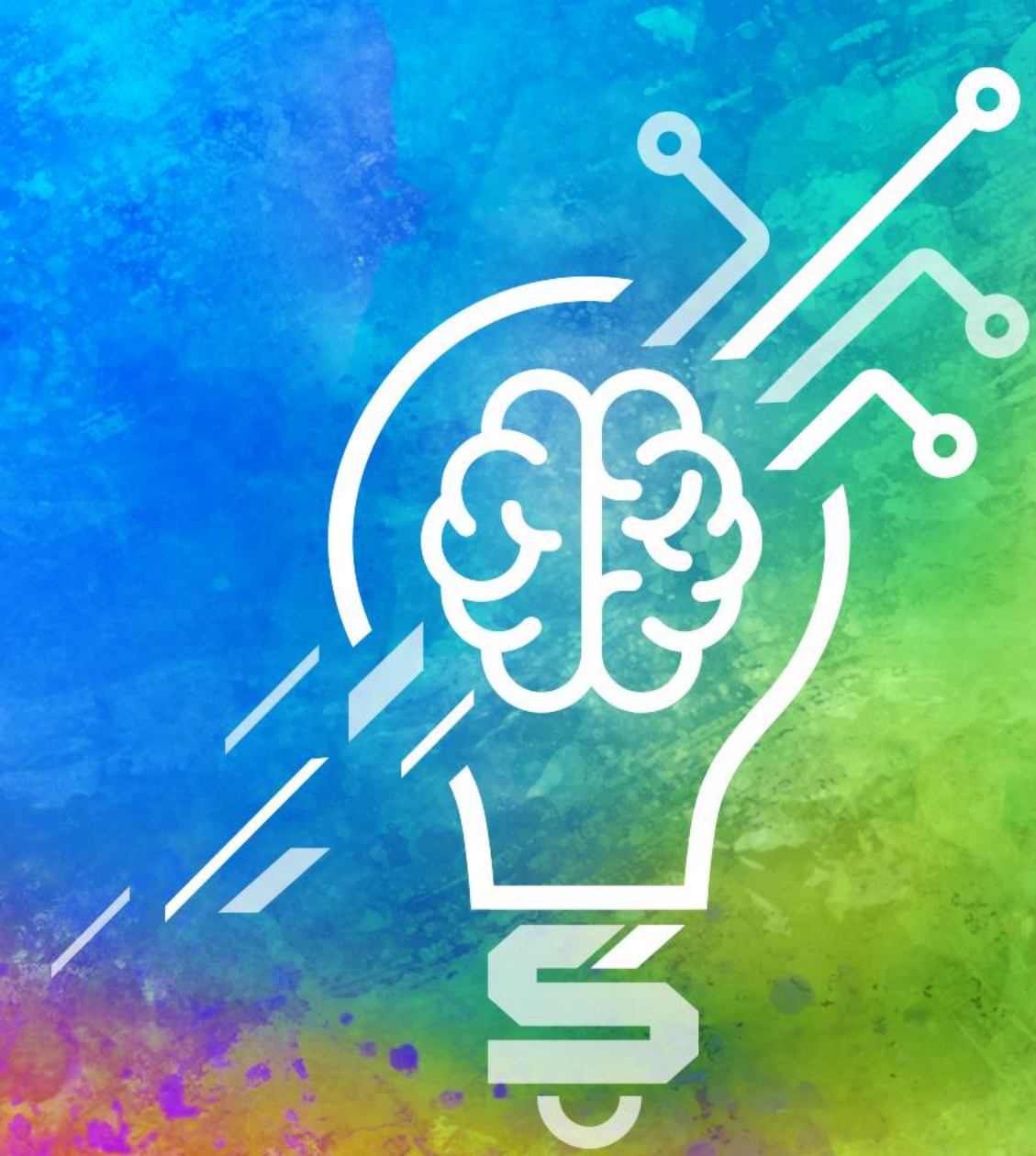


# Extending the Network

- **Don’t forget a few things:**
  - Each node needs to be re-programmed so that it has a unique IP Address and MAC Address on the multidrop network
  - Each network needs one coordinator which is PLCA Node 0
  - Each other node needs to have a unique PLCA Node number, which must be less than the total number of nodes that the system is configured for
  - Remove the termination jumpers, except for the end nodes

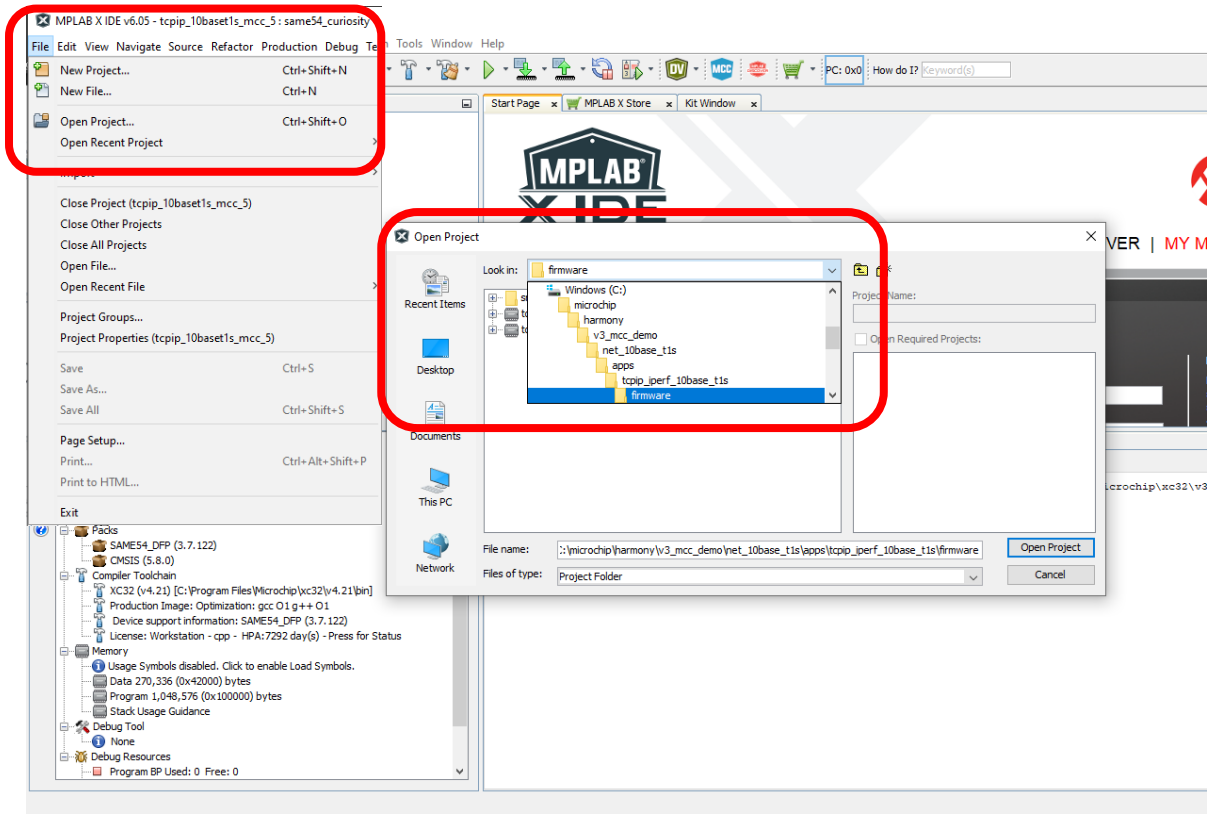


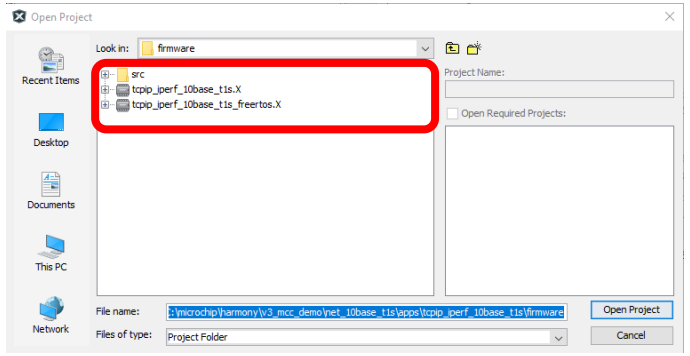


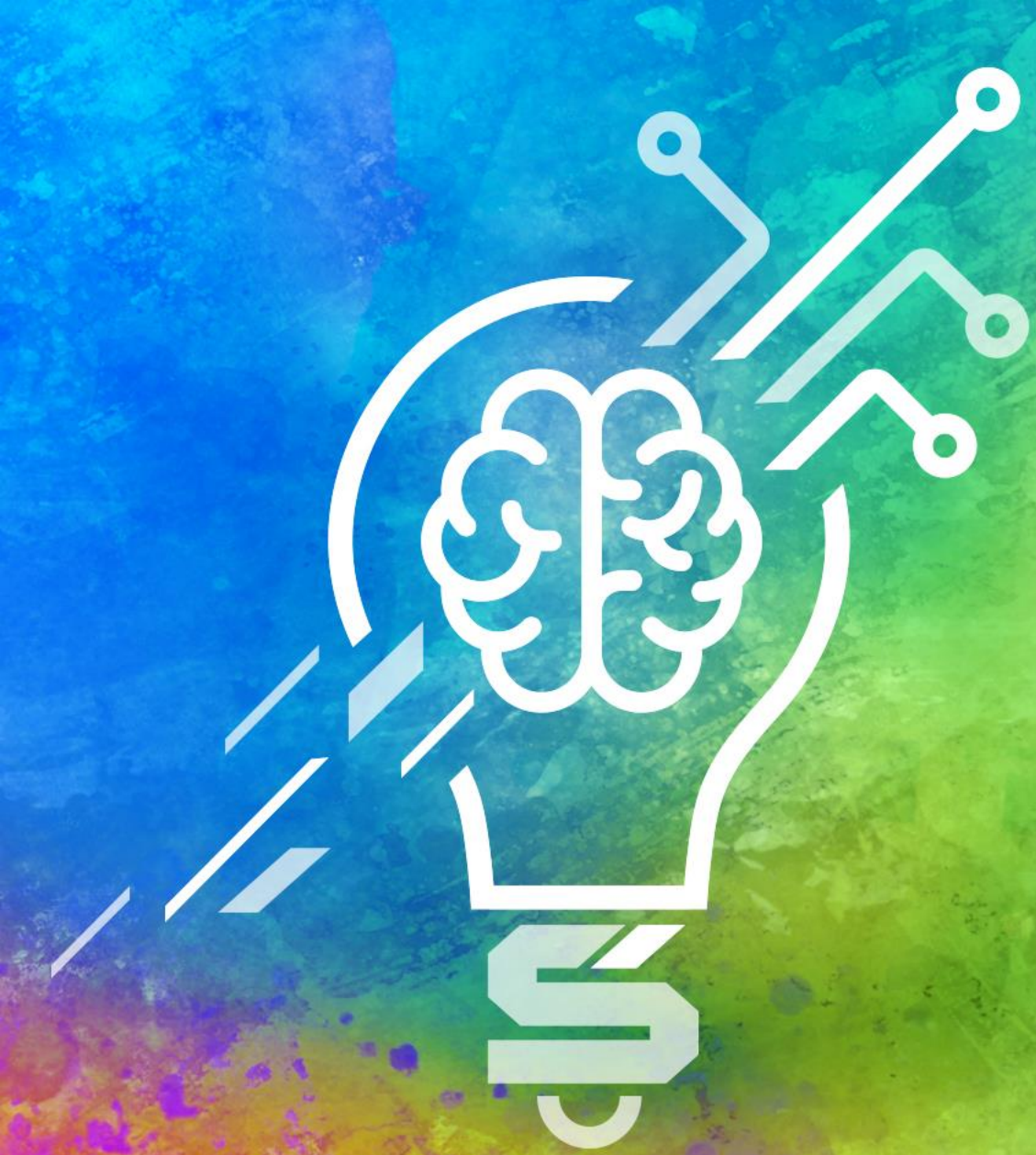
# Useful Links and References

# Demo Project

- **There is another demo project checked into the Github Harmony repository that is in the net_10base_t1s library**
- **Use File > Open Project and navigate to this library, and choose from the freertos or non-freertos versions**
  - Note: in order to generate and compile these projects, you will still need to have downloaded the necessary packages as described in the project setup





# LAB Appendix:
# Equipment and Tools List

# Equipment List

## Evaluation Boards

| Evaluation Board | Link | Photo | Quantity Needed |
|---|---|---|---|
| SAM E54 CURIOSITY ULTRA DEVELOPMENT BOARD | https://www.microchipdirect.com/dev-tools/DM320210 |  | 2 per attendee |
| ATOLED1-XPRO - OLED1 Xplained Pro extension kit | https://www.microchipdirect.com/dev-tools/ATOLED1-XPRO |  | 2 per attendee |
| EVB-LAN8670-RMII | https://www.microchipdirect.com/dev-tools/EV06P90A |  | 2 per attendee |

# Equipment List

## Accessories

| Evaluation Board | Photo | Quantity Needed |
|---|---|---|
| UTP (Unshielded Twisted Pair Wires) |  | 2 per attendee |
| Small Screwdriver |  | 1 per attendee |
| Micro USB Cables | | 2 per attendee |