

2019 MASTERs Conference

23075 IoT6

Simplifying TCP/IP Applications with MPLAB® Harmony

Hands-On Lab Manual

Instructors:

Martin Ruppert

Raji Shanmugasundaram

Niklas Larsson

Microchip Technology Inc.



MICROCHIP

MPLAB® Harmony TCP/IP Stack

UDP Module API Function List

Socket Management Functions

TCPIP_UDP_ServerOpen	Opens a UDP socket as a server.
TCPIP_UDP_ClientOpen	Opens a UDP socket as a client.
TCPIP_UDP_IsOpened	Determines if a socket was opened.
TCPIP_UDP_IsConnected	Determines if a socket has an established connection.
TCPIP_UDP_Bind	Bind a socket to a local address and port. This function is meant for client sockets. It assigns a specific source address and port for a socket.
TCPIP_UDP_RemoteBind	Bind a socket to a remote address This function is meant for server sockets.
TCPIP_UDP_Close	Closes a UDP socket and frees the handle.
TCPIP_UDP_OptionsGet	Allows getting the options for a socket such as current RX/TX buffer size, etc.
TCPIP_UDP_OptionsSet	Allows setting options to a socket like adjust RX/TX buffer size, etc
TCPIP_UDP_SocketInfoGet	Returns information about a selected UDP socket.
TCPIP_UDP_SocketNetGet	Gets the network interface of an UDP socket
TCPIP_UDP_SocketNetSet	Sets the network interface for an UDP socket
TCPIP_UDP_TxOffsetSet	Moves the pointer within the TX buffer.
TCPIP_UDP_SourceIPAddressSet	Sets the source IP address of a socket
TCPIP_UDP_BcastIPv4AddressSet	Sets the broadcast IP address of a socket Allows an UDP socket to send broadcasts.
TCPIP_UDP_DestinationIPAddressSet	Sets the destination IP address of a socket
TCPIP_UDP_DestinationPortSet	Sets the destination port of a socket
TCPIP_UDP_Disconnect	Disconnects a UDP socket and re-initializes it.
TCPIP_UDP_SignalHandlerDeregister	Deregisters a previously registered UDP socket signal handler.
TCPIP_UDP_SignalHandlerRegister	Registers a UDP socket signal handler.
TCPIP_UDP_Task Standard	TCP/IP stack module task function.

Transmit Data Functions

TCPIP_UDP_PutIsReady	Determines how many bytes can be written to the UDP socket.
TCPIP_UDP_TxPutIsReady	Determines how many bytes can be written to the UDP socket.
TCPIP_UDP_ArrayPut	Writes an array of bytes to the UDP socket.
TCPIP_UDP_StringPut	Writes a null-terminated string to the UDP socket.
TCPIP_UDP_Put	Writes a byte to the UDP socket.
TCPIP_UDP_TxCountGet	Returns the amount of bytes written into the UDP socket.
TCPIP_UDP_Flush	Transmits all pending data in a UDP socket.

Receive Data Transfer Functions

TCPIP_UDP_GetIsReady	Determines how many bytes can be read from the UDP socket.
TCPIP_UDP_ArrayGet	Reads an array of bytes from the UDP socket.
TCPIP_UDP_Get	Reads a byte from the UDP socket.
TCPIP_UDP_RxOffsetSet	Moves the read pointer within the socket RX buffer.
TCPIP_UDP_Discard	Discards any remaining RX data from a UDP socket.

Lab 1

Lab 2

Lab 3

Introduction

This Lab Manual provides the step by step procedure to complete two labs in the MASTERS 32075 IoT6 Class.

In Lab 1 we will open a TCP project, do some stack re-configuration and a connectivity check and in Lab 2 we will show an Application integration for local access, using the example of a Vending machine. Finally in Lab 3 we will make an Application integration for external access, using the example of a Weather Service.

Hardware Requirements

The following hardware is required:

- **SAM E70 Xpained Ultra** (Microchip Part Number: DM320113)
 - <http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=dm320113>



- **OLED1 Xpained Pro extension kit** (Microchip Part Number: ATOLED1-XPRO)



- **Cat 5 Ethernet Patch Cable**
- **USB Male A to USB Male B Micro Cable**

Software Requirements

The following software is required:

- **Microchip MPLAB X IDE v5.20**
 - <http://www.microchip.com/mplab>
- **Microchip MPLAB XC32 Compiler v2.15**
 - <http://www.microchip.com/mplab/compilers>
- **Microchip MPLAB Harmony 3**
 - <http://www.microchip.com/mplab/mplab-harmony>
- **Microchip MPLAB Harmony Configuration (MHC) Tool Plugin v3.3.0.1**
- **Tera Term v4.95**

Lab 1

Introduction

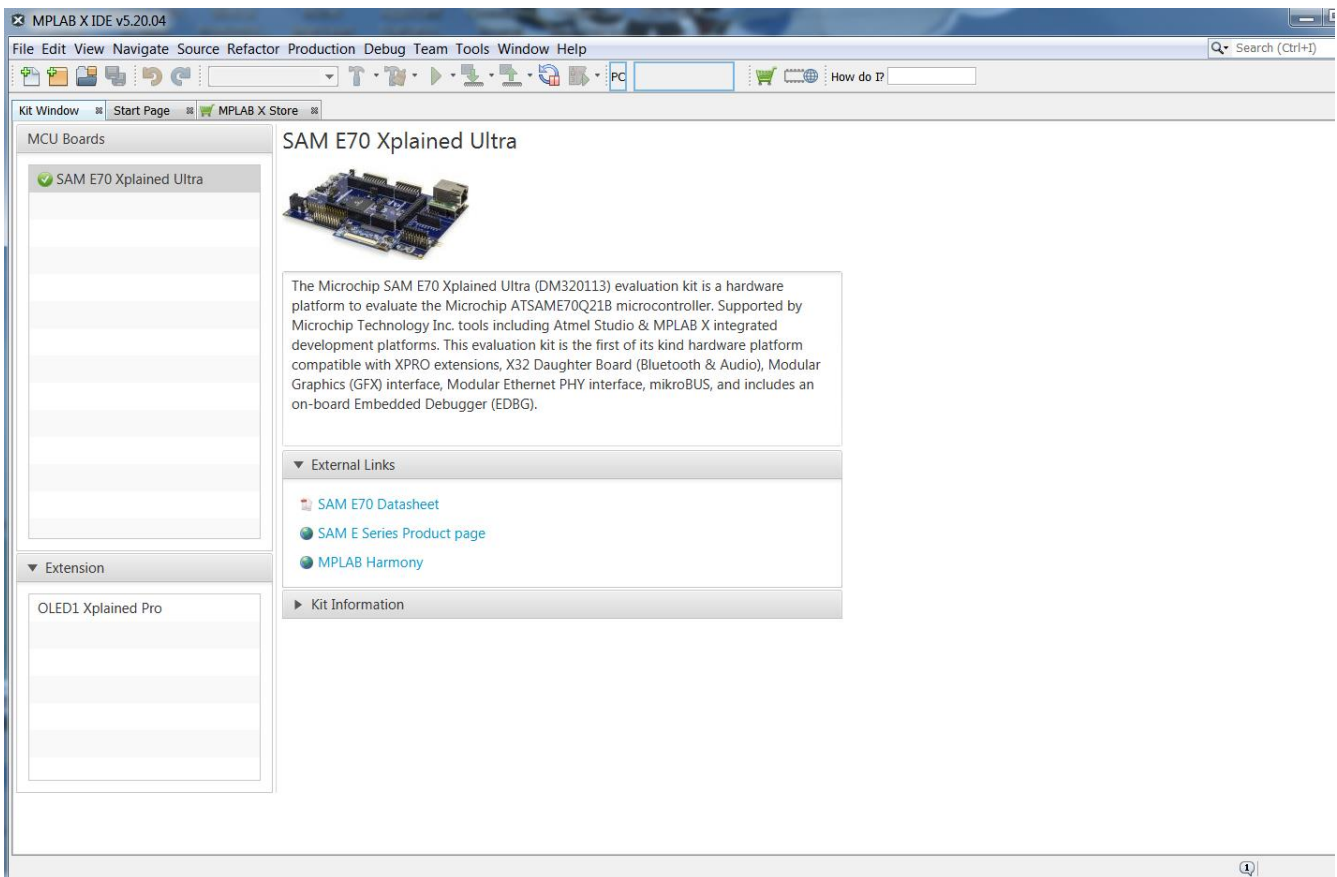
Lab 1 will show you how to open an existing TCP/IP MPLAB Harmony 3 Project and using the MPLAB Harmony Configuration (MHC) Tool. The project will incorporate basic TCP/IP functionality to allow the SAM E70 Xplained Ultra to connect to an Ethernet Network, along with a simple application to flash a “Heartbeat” LED every 500ms. Once the project is generated and programmed onto the development kit, you will use a number of techniques to validate that the PIC is connected to a network and determine its IP Address. The concepts that will be covered in this lab include:

- Open a SAM E70 MPLAB X Project
- Configuring the MPLAB Harmony path
- Configuring the TCP/IP Stack options, including:
 - Network Configuration of the Host Name
 - TCP/IP Services including Dynamic Host Configuration Protocol Client, ICMPv4 Server (for Ping testing) & Announce Discovery Tool
 - Bandwidth testing with “iperf”
- Configuring the Harmony Console and Command Service for monitoring and control of the TCP/IP stack via a Terminal Client running on a USB CDC Interface (Emulated RS232 COM Port).
Toggling the IO Pin that drives USER_LED0 on the SAM E70 Xplained Ultra
- Using the Windows Command Line Ping Tool and the Microchip TCP/IP Discovery tools to test connectivity of your SAM E70 Xplained Ultra
- Use the Console and Command System to get help on available TCPIP Commands and execute a command to get information about the network configuration.

Lab Procedure

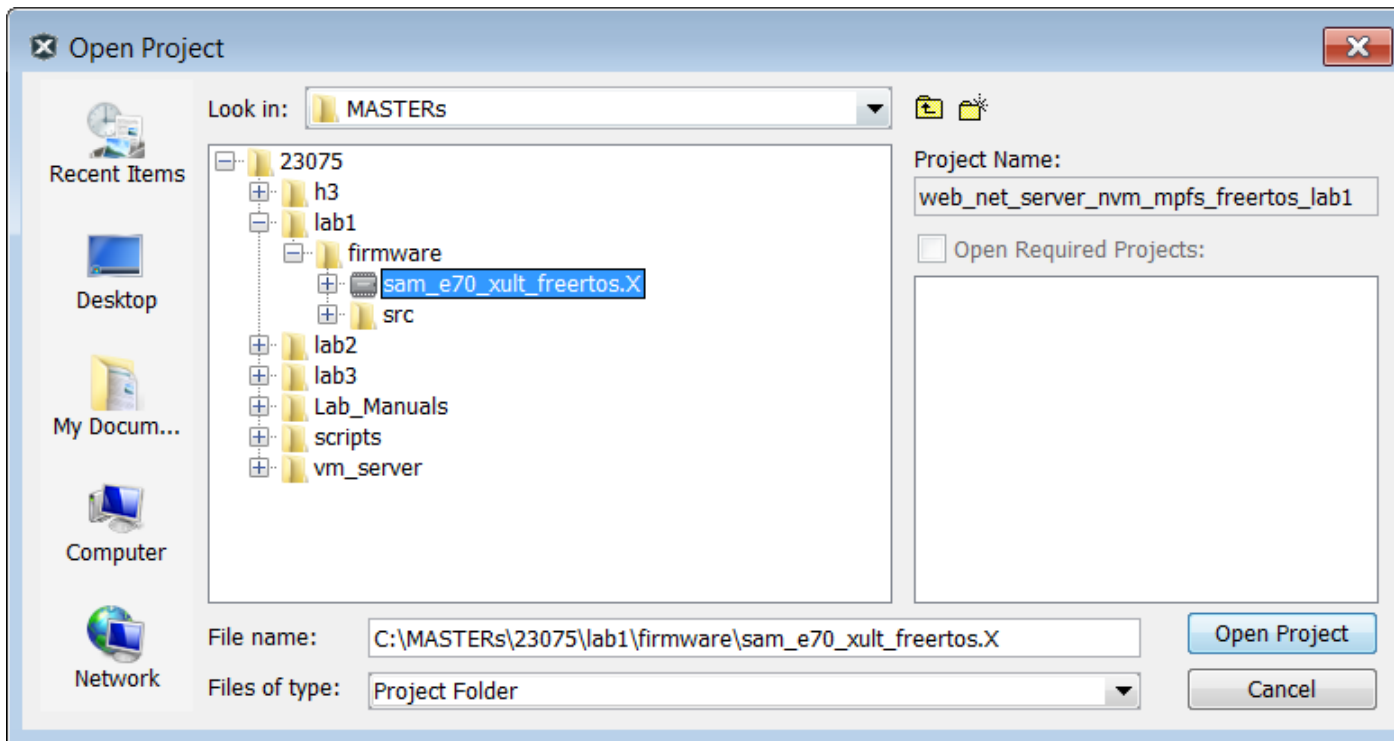
Starting MPLAB X IDE

- 1.1. Start MPLAB X IDE by double clicking on the MPLAB X IDE v5.20.04 icon found on the Windows desktop.

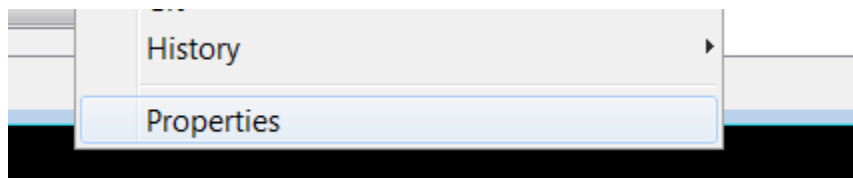


Project Load

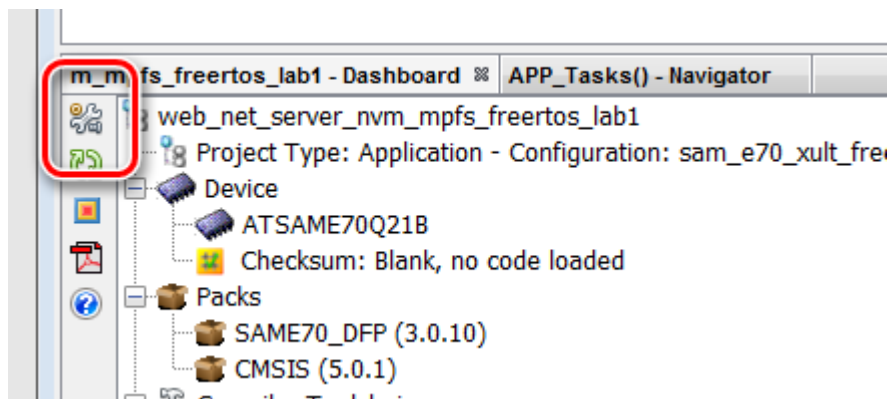
1. Open Project by choosing File>Open Project... from the main menu and select lab1 project



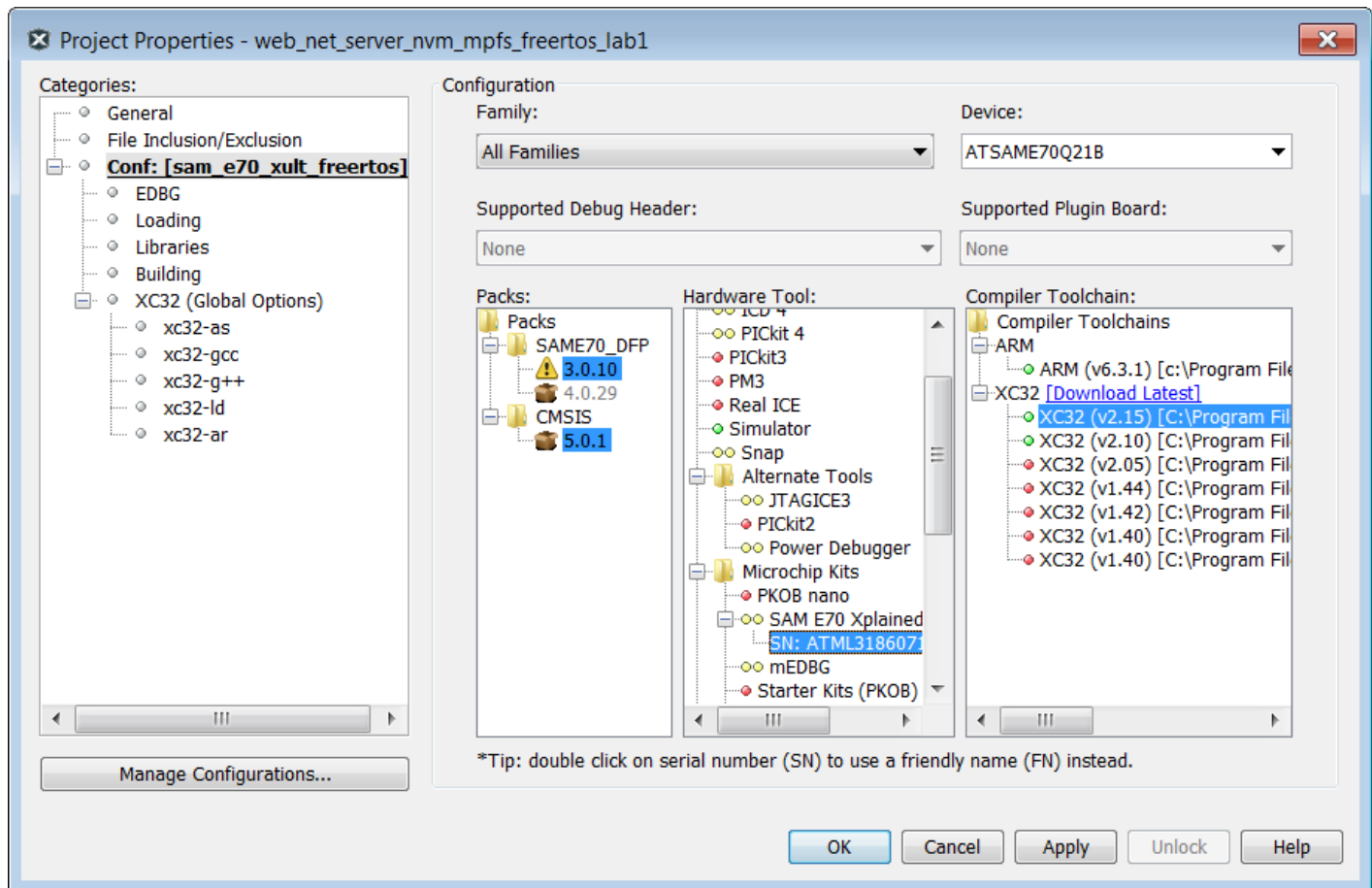
2. Open Project Properties by choosing
 - a. File>Project Properties from the main menu
 - b. Or select with a right click the project node in the project windows and select at the bottom



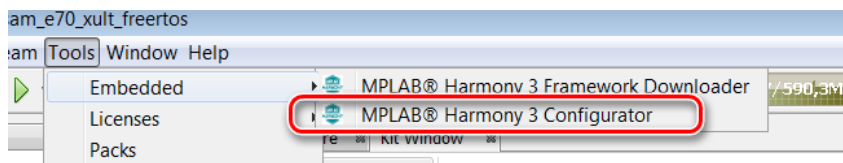
- c. Or select the toolbox in the Dashboard



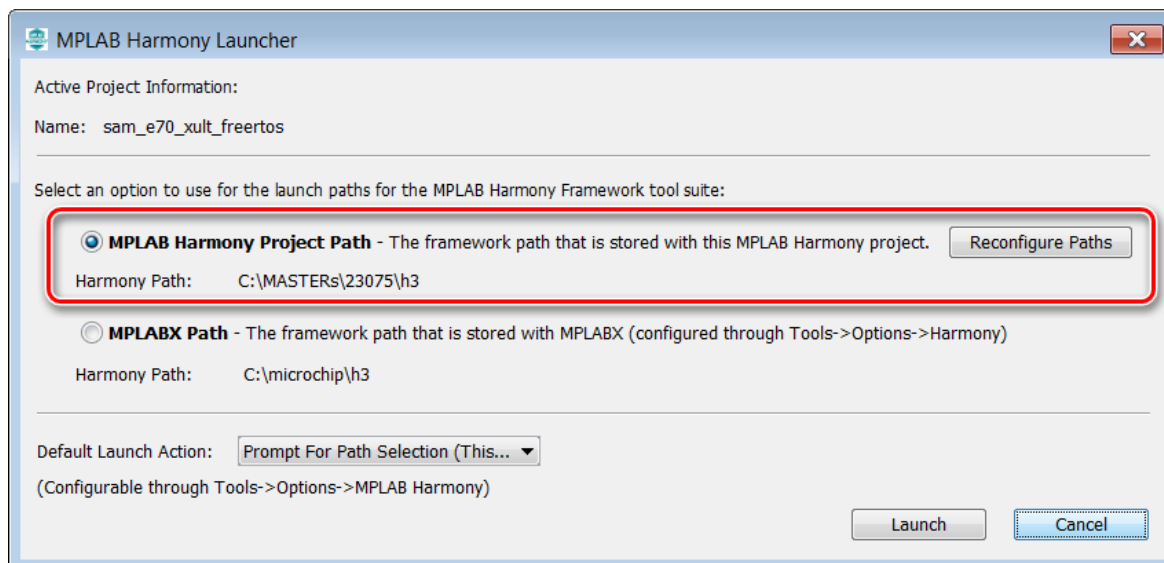
3. Select the XC32 Compiler v2.15
4. Select the SAME70 Xplained by clicking on the SN: Number



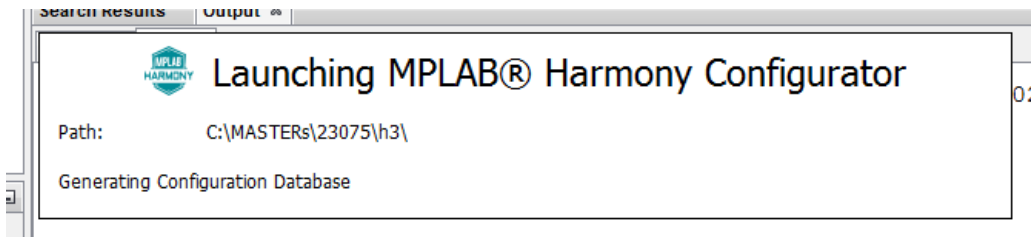
5. Select from the Tools Menu the Harmony 3 Configurator



6. Ensure the H3 Path is set to "C:\MASTERS\23075\h3\"

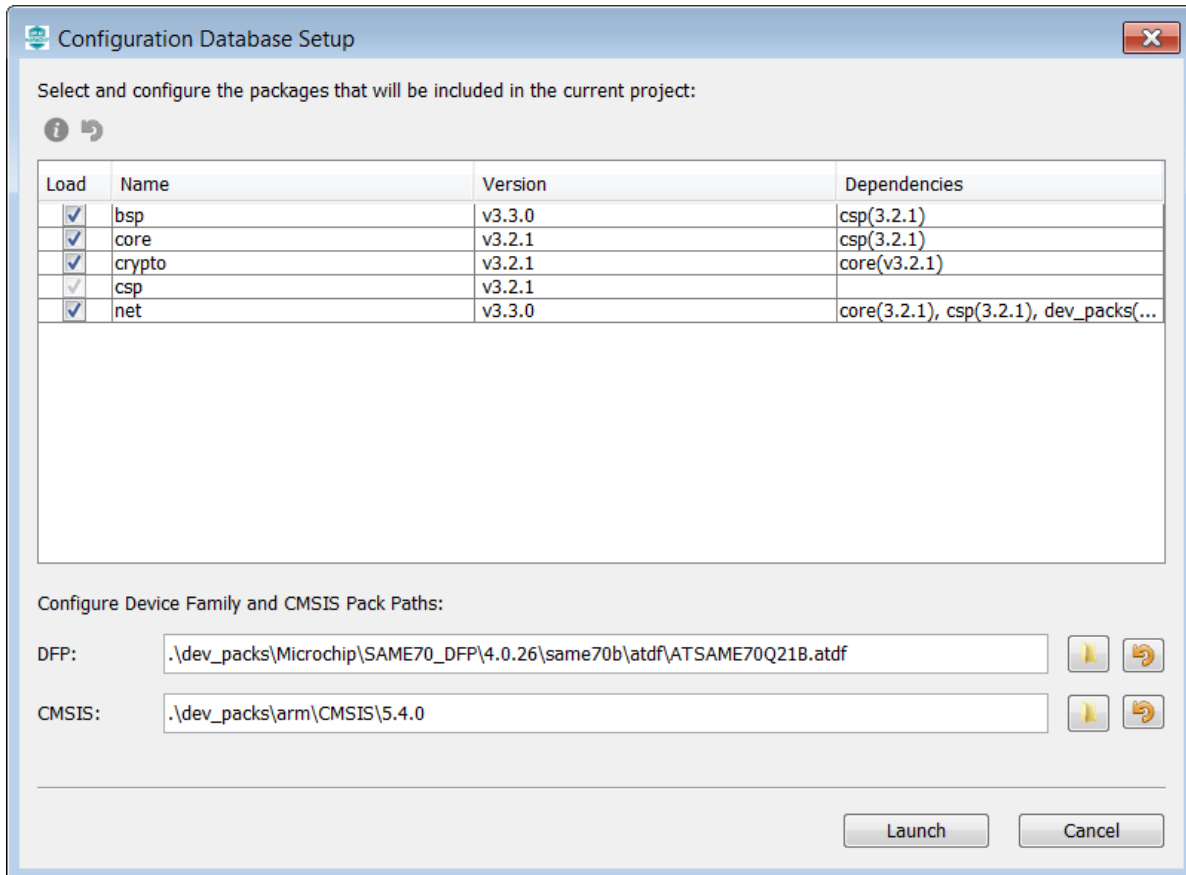


7. Select "Launch"



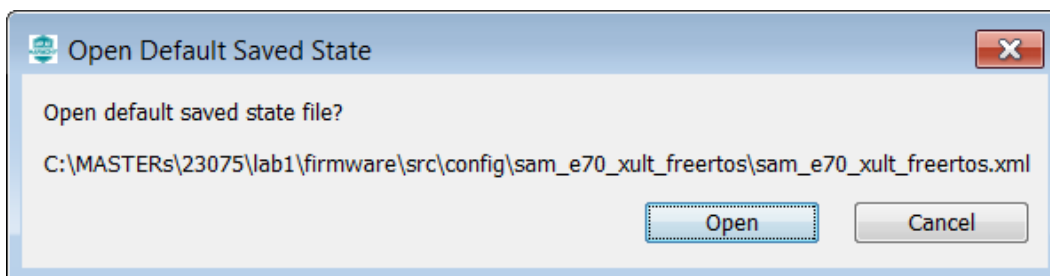
The first time the MHC is started, it can take up to 2 Minutes before the Configuration Database is prepared

8. In the next window the H3 parts and their used Version Numbers are displayed

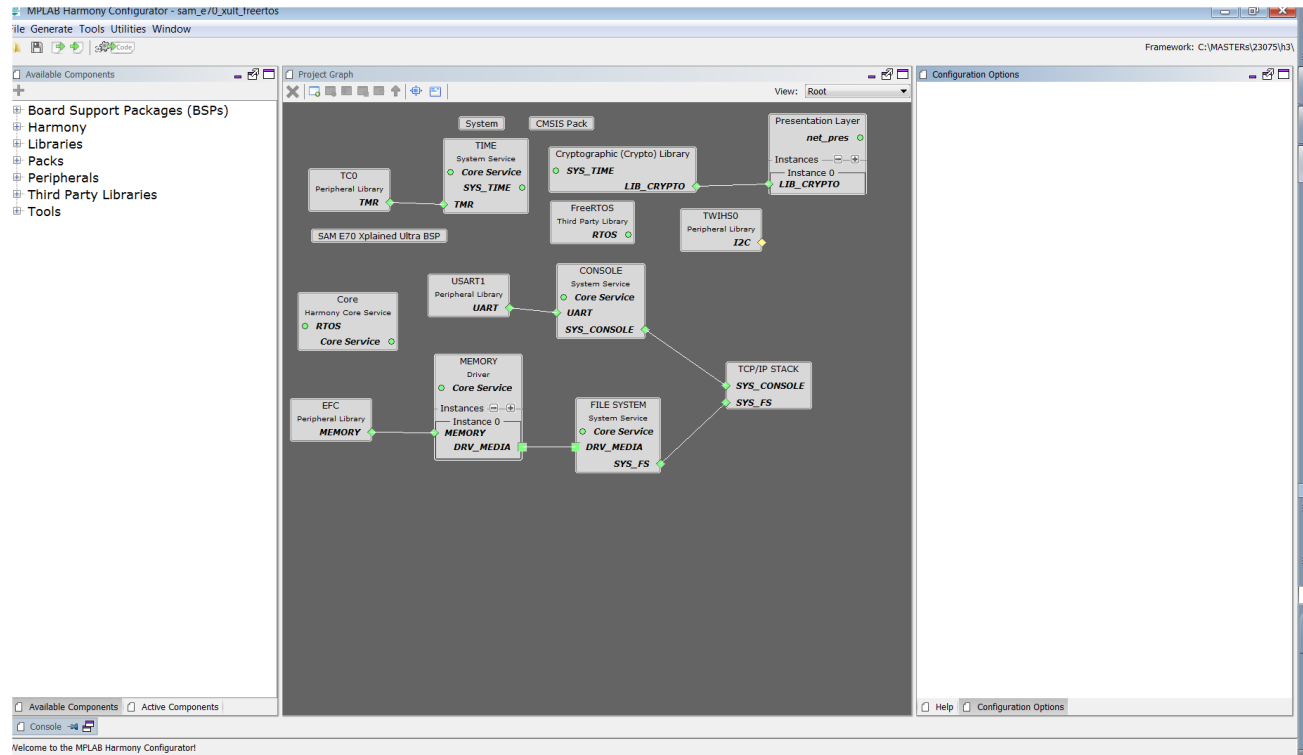


Click on "Launch"

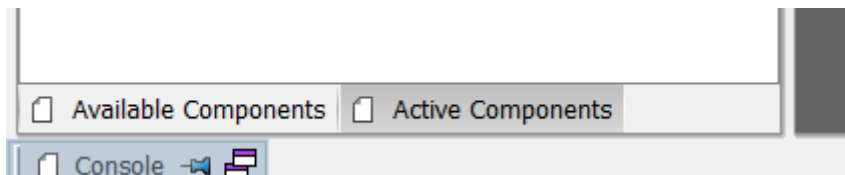
9. Open the saved state file



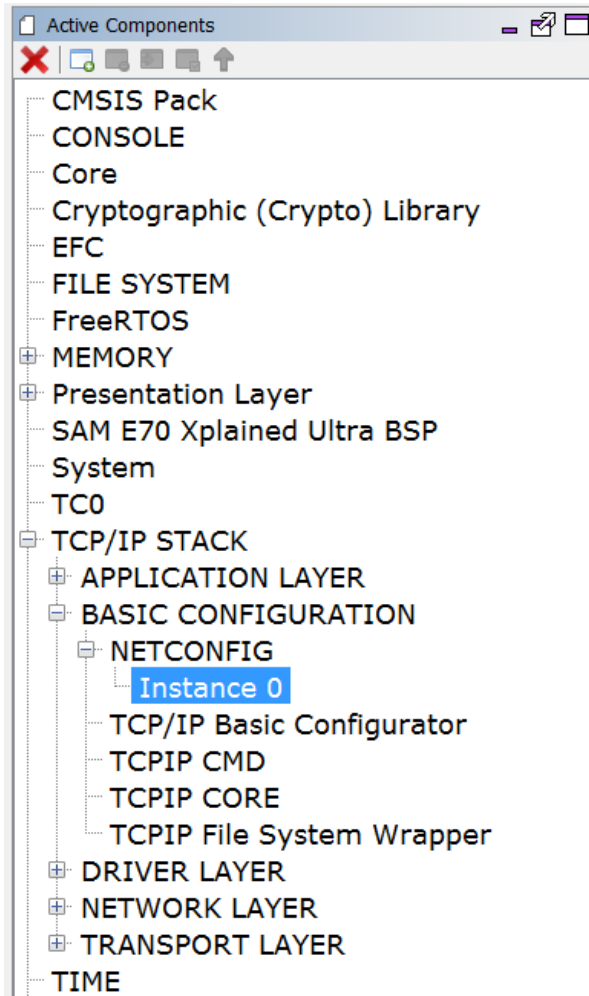
10. The MHC is up and running



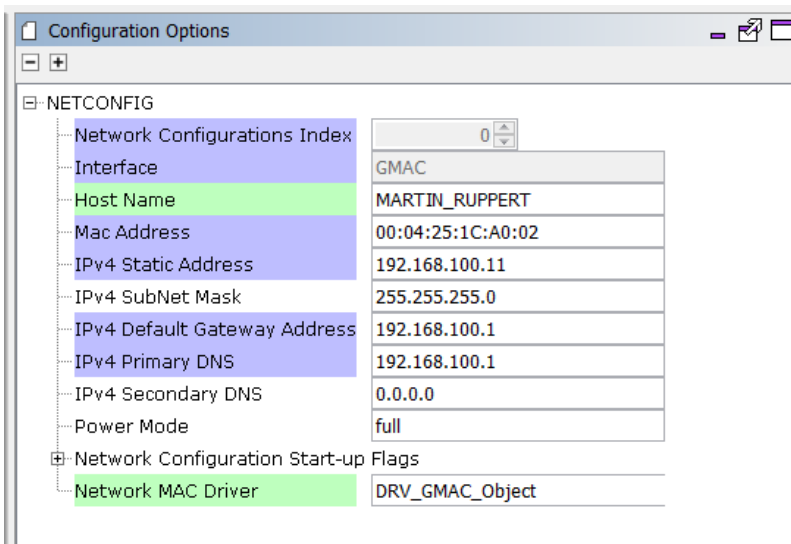
11. Select Active Components (left below)



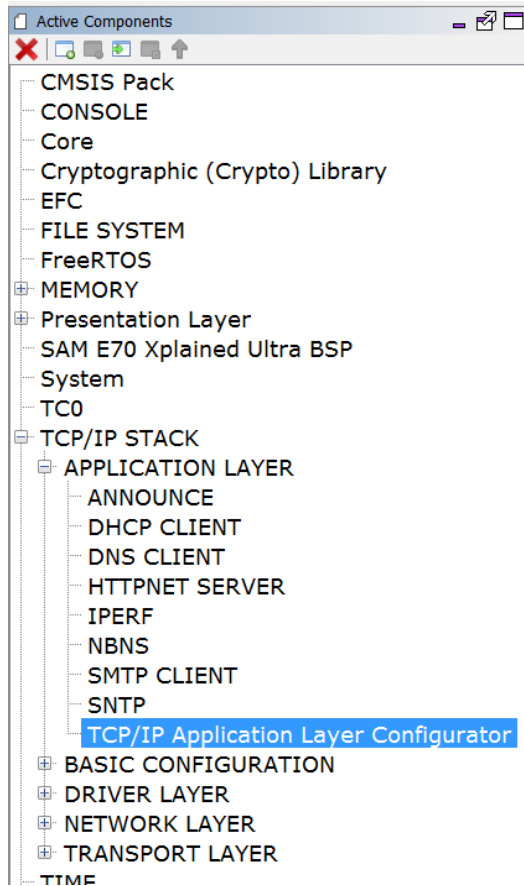
12. Select the Instance 0 in Active Components



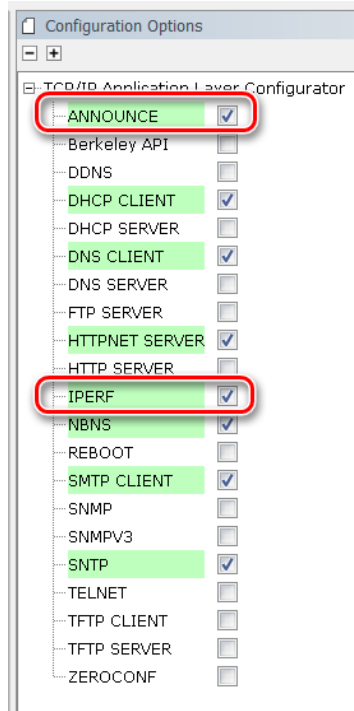
13. And change in the Configuration Options (on the right side) the Host Name to something meaningful for you.
The Host name can be identified in the Network.



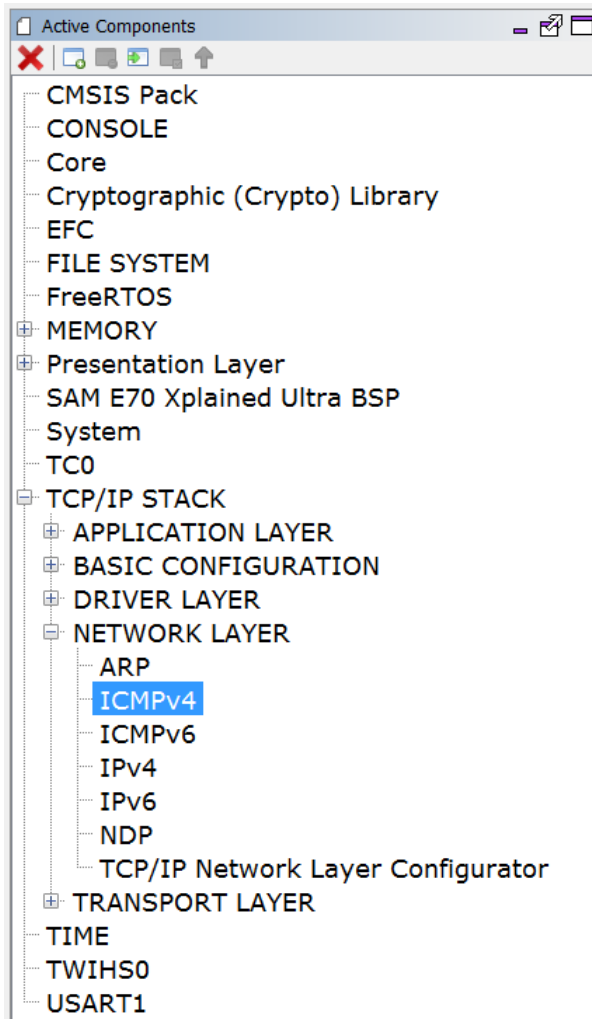
14. Select “TCP/IP Application Layer Configuration” in Active Components



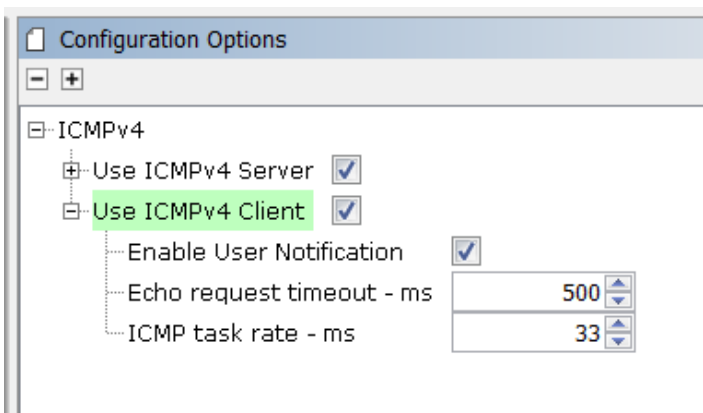
15. Ensure that ANNOUNCE and IPERF are selected



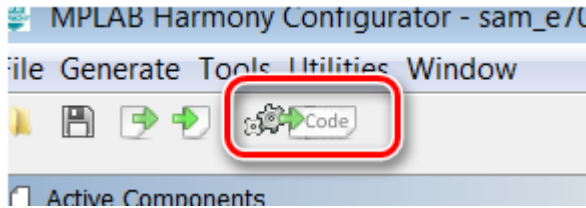
16. Select ICMPv4 in the Active Components



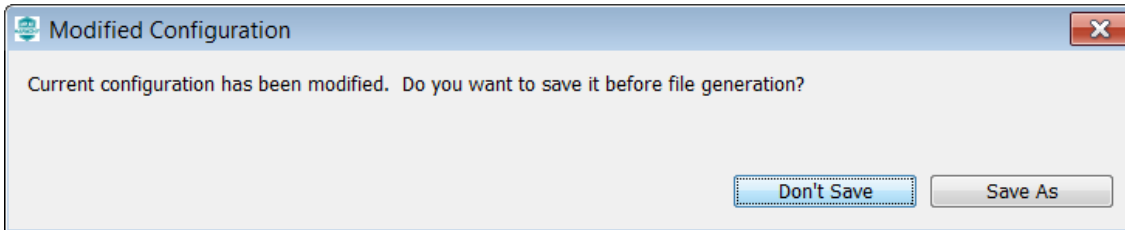
17. And ensure that the “Use ICMPv4 Client” is selected



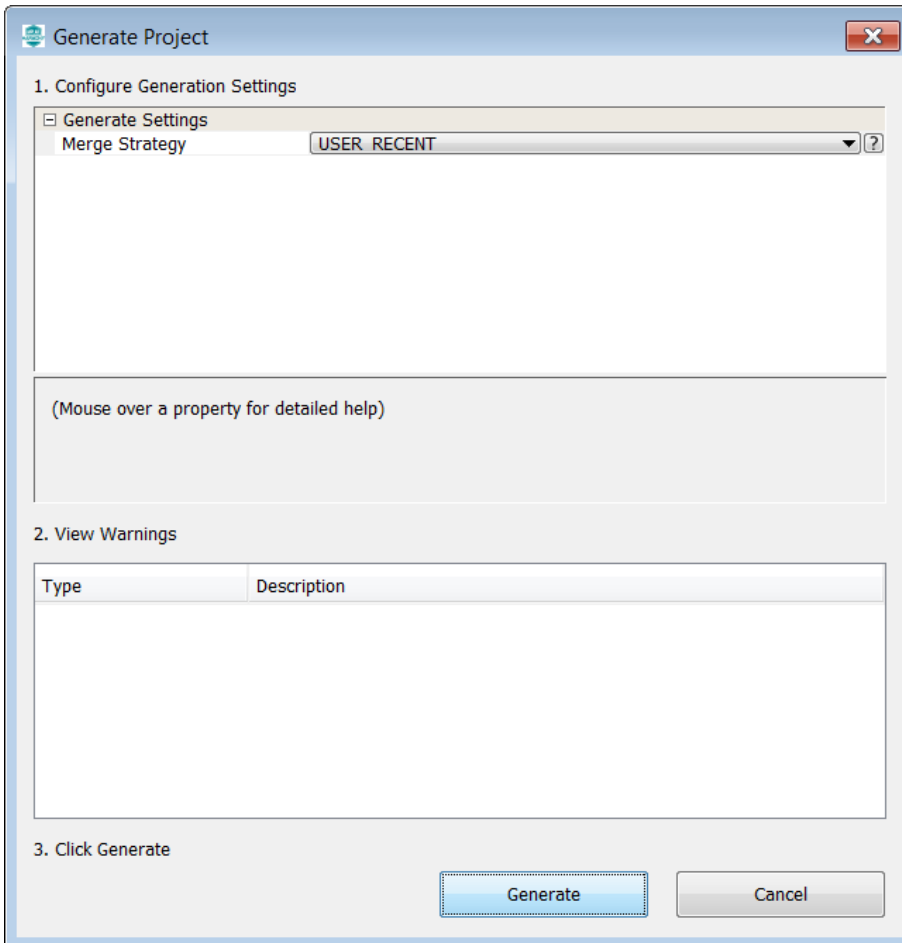
18. Select “Code” (Generate Code)



19. Select Don't Save

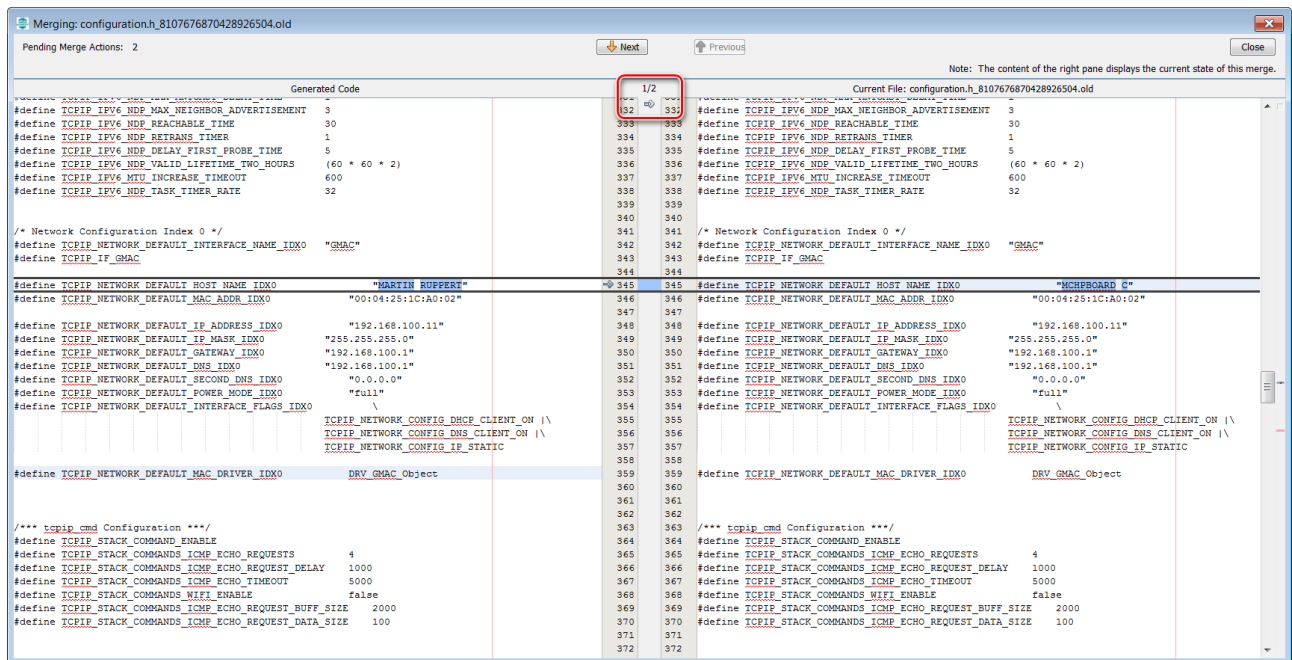


20. Select Generate

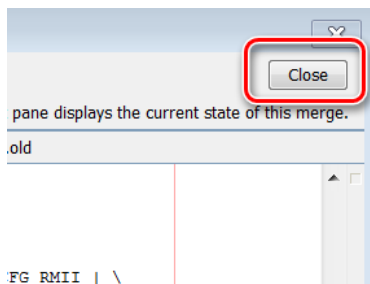


21. Some Files will be changed and the MHC is asking in a “diff” window, if the changes should be taken over.

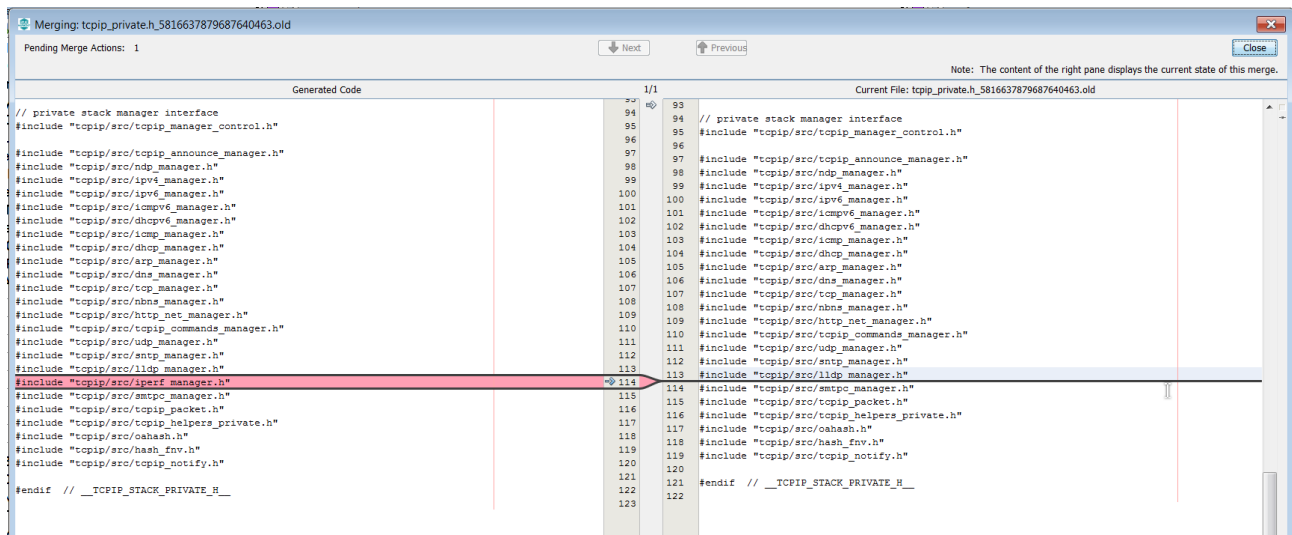
Accept all changes in the file by clicking on the Arrow in the middle above.



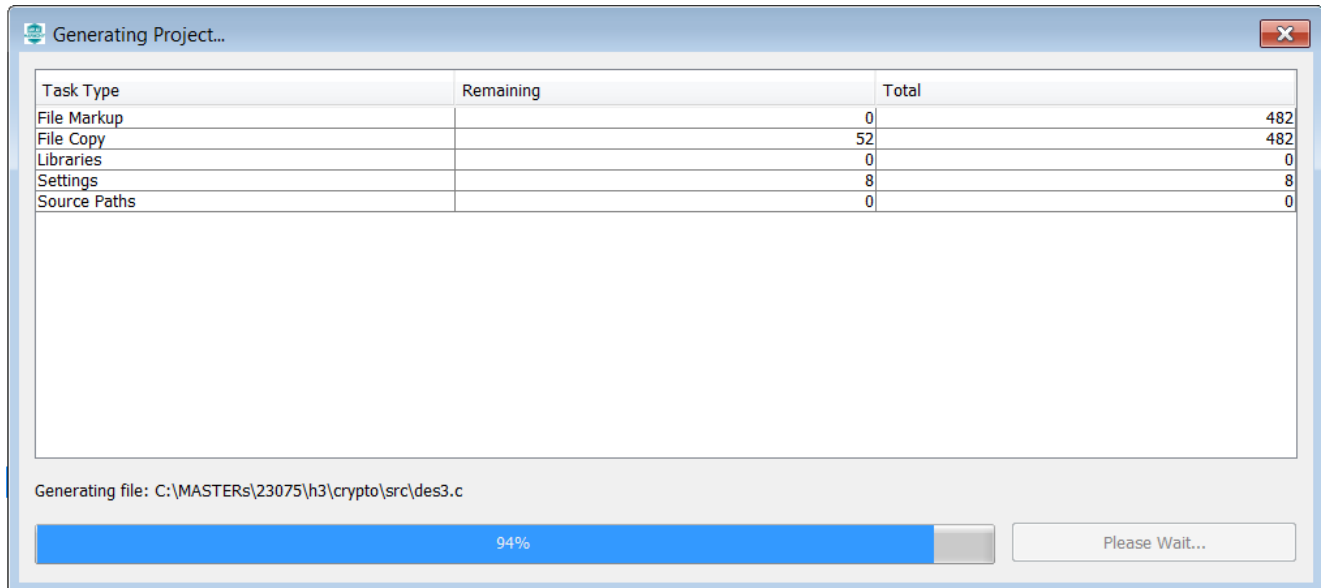
22. Then click on close in the upper right corner



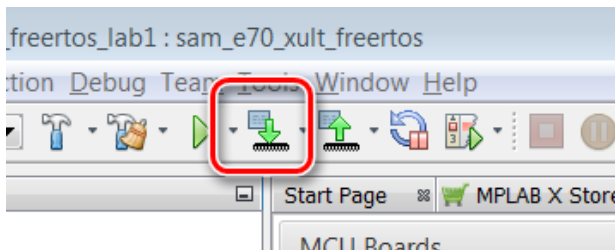
23. Same for the next 2 diff windows



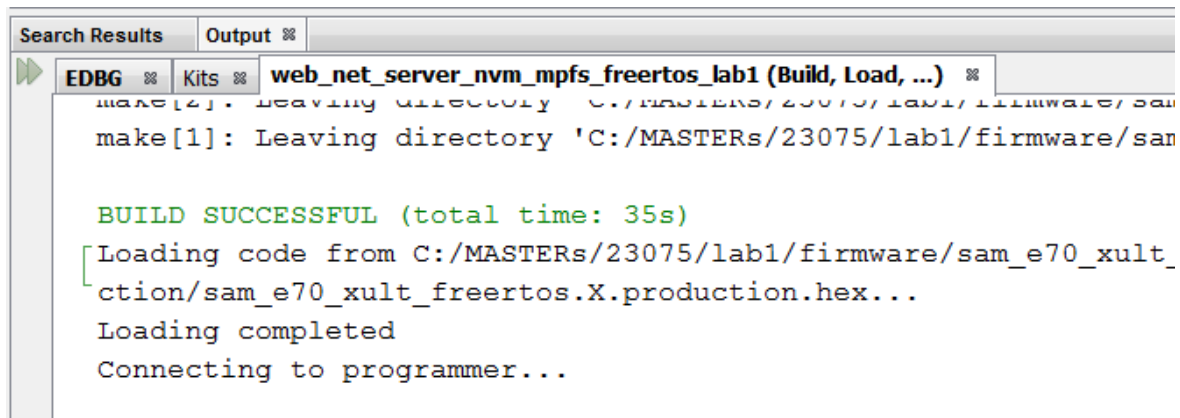
24. The whole process is displayed with a progress bar



25. Back again in the main window of MPLABX, click on the “Make and Program Device” button

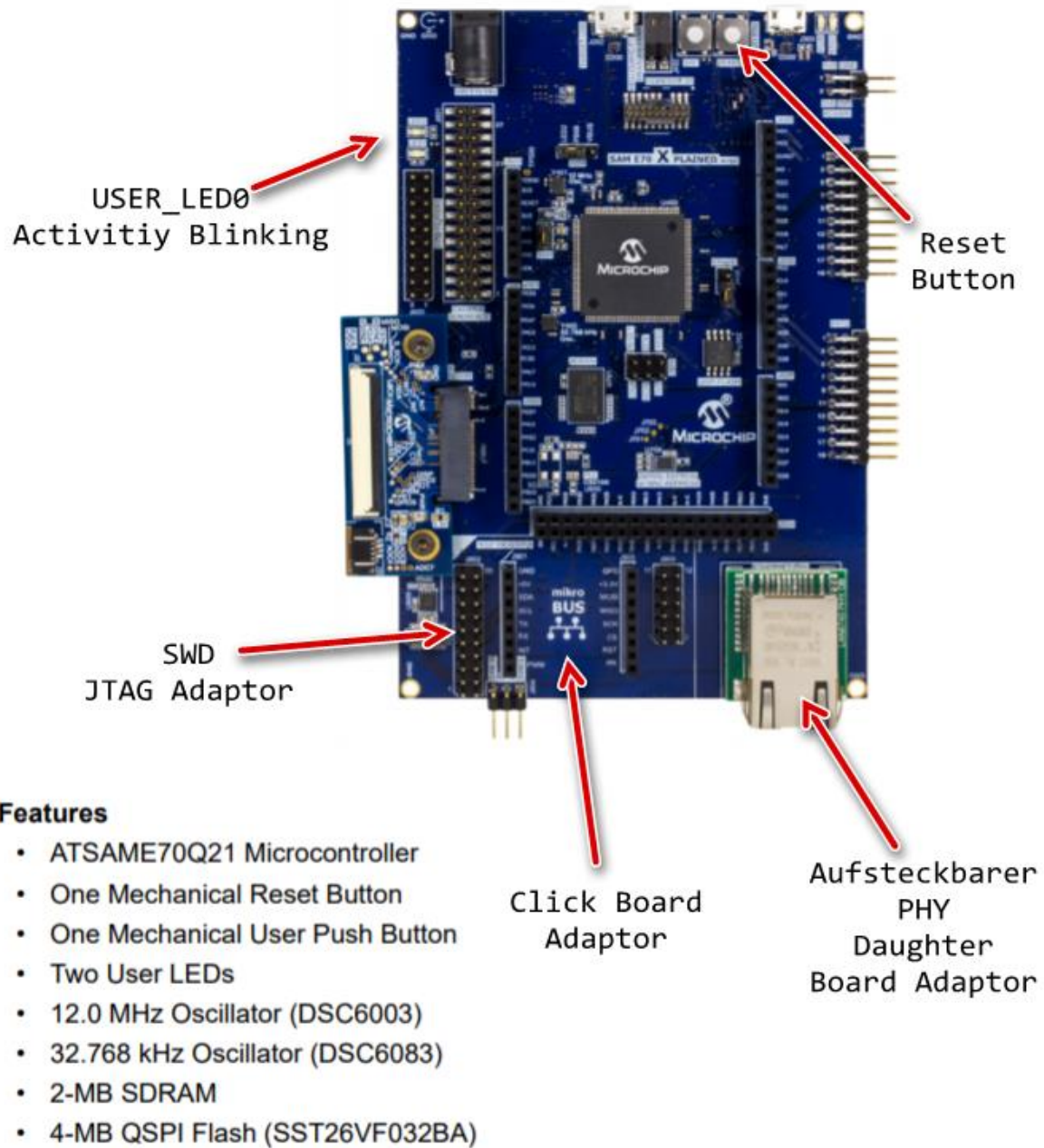


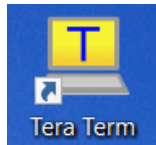
26. After successful build, the SAME70 is programmed automatically



27. The programming take about 30 seconds with on Board DebuggeCheck whether the USER_LED0 Activity is Blinking

Figure 1. SAM E70 Xplained Ultra Evaluation Kit with PHY Daughter Board

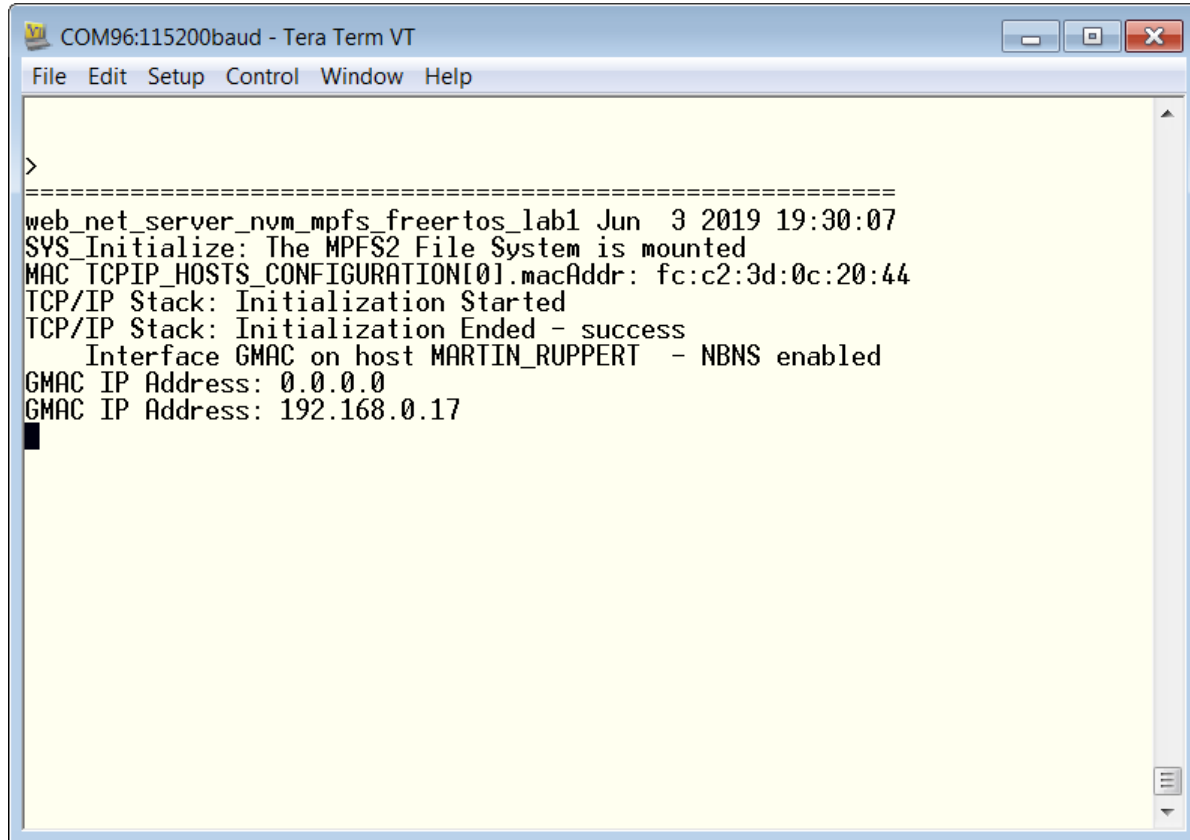




28. Open Terra Term Terminal Program and select under Setup->Serial Port the COM Port (in this case a COM96, but could be a different COM Port number in your case).

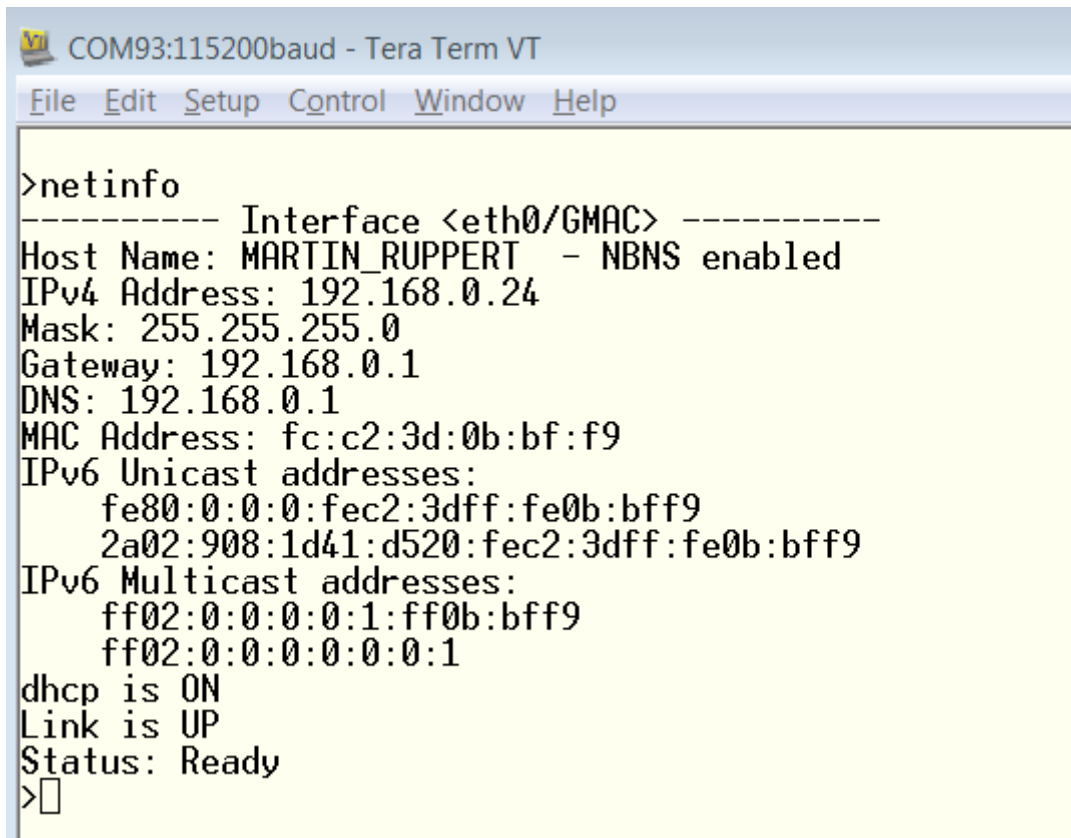
The press the Reset Button of the Board and let the Firmware restart. You should see the start message with the Lab name and the build timestamp at first, followed by the MAC Address from EEPROM.

29. After some small time, the IP Address should be change from 0.0.0.0 to a valid address

A screenshot of the Tera Term VT window. The title bar reads 'COM96:115200baud - Tera Term VT'. The menu bar includes 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The main text area shows the following output:

```
>
=====
web_net_server_nvm_mpfs_freertos_lab1 Jun  3 2019 19:30:07
SVS_Initialize: The MPFS2 File System is mounted
MAC TCP/IP_HOSTS_CONFIGURATION[0].macAddr: fc:c2:3d:0c:20:44
TCP/IP Stack: Initialization Started
TCP/IP Stack: Initialization Ended - success
    Interface GMAC on host MARTIN_RUPPERT - NBNS enabled
GMAC IP Address: 0.0.0.0
GMAC IP Address: 192.168.0.17
```

30. To check the basic information about the network enter the `netinfo` command and press **Enter**.



```
COM93:115200baud - Tera Term VT
File Edit Setup Control Window Help

>netinfo
----- Interface <eth0/GMAC> -----
Host Name: MARTIN_RUPPERT - NBNS enabled
IPv4 Address: 192.168.0.24
Mask: 255.255.255.0
Gateway: 192.168.0.1
DNS: 192.168.0.1
MAC Address: fc:c2:3d:0b:bf:f9
IPv6 Unicast addresses:
    fe80:0:0:0:fec2:3dff:fe0b:bff9
    2a02:908:1d41:d520:fec2:3dff:fe0b:bff9
IPv6 Multicast addresses:
    ff02:0:0:0:0:1:ff0b:bff9
    ff02:0:0:0:0:0:0:1
dhcp is ON
Link is UP
Status: Ready
>
```

31. A help shows the available commands

A screenshot of a Tera Term VT window titled 'COM93:115200baud - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The main text area shows the output of the 'help' command. It lists supported command groups (iperf, tcpip) and built-in commands (reset, q, help). Then, the 'help tcpip' command is entered, showing a detailed list of TCP/IP related commands like netinfo, defnet, dhcp, dhcps, zc11, setdns, setip, setgw, setbios, setmac, if, stack, heapinfo, ping, arp, and dnsc.

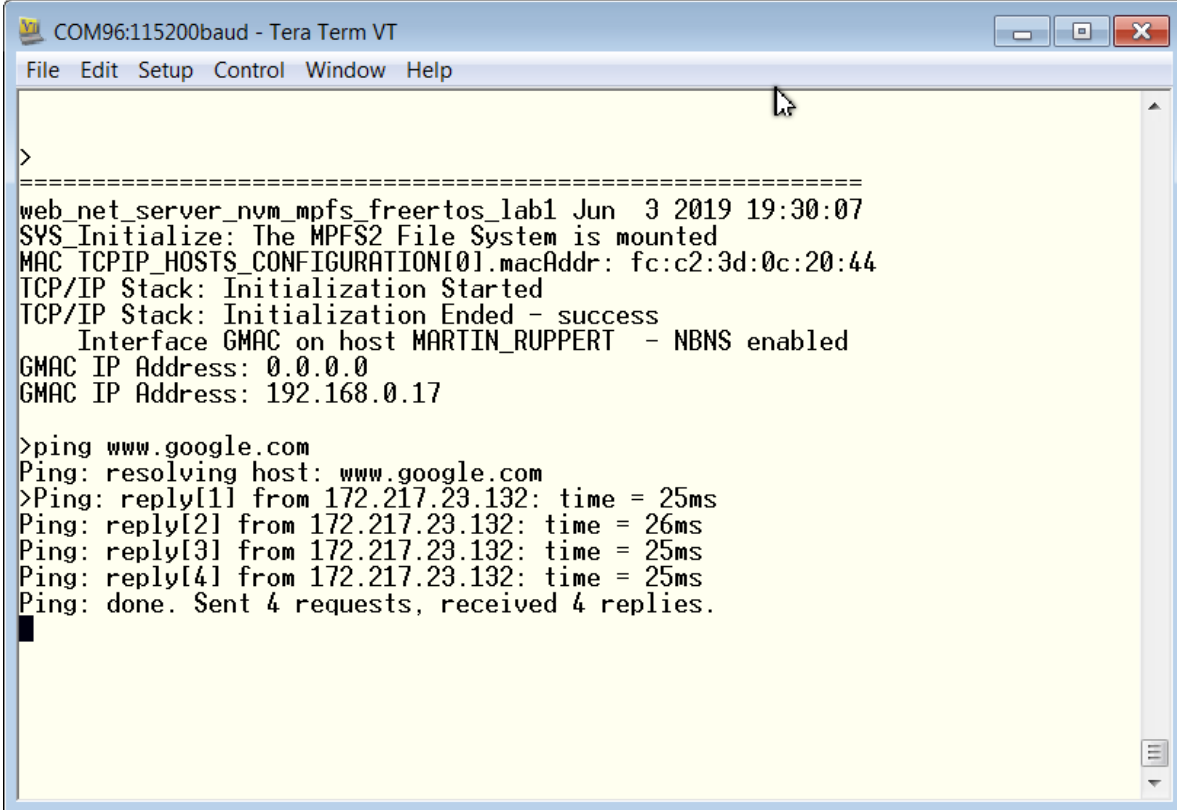
```
COM93:115200baud - Tera Term VT
File Edit Setup Control Window Help

>help

----- Supported command groups -----
*** iperf: iperf commands ***
*** tcpip: stack commands ***
----- Built in commands -----
*** reset: Reset host ***
*** q: quit command processor ***
*** help: help ***
>help tcpip

*** netinfo: Get network information ***
*** defnet: Set/Get default interface ***
*** dhcp: DHCP client commands ***
*** dhcps: Turn DHCP server on/off ***
*** zc11: Turn ZCLL on/off ***
*** setdns: Set DNS address ***
*** setip: Set IP address and mask ***
*** setgw: Set Gateway address ***
*** setbios: Set host's NetBIOS name ***
*** setmac: Set MAC address ***
*** if: Bring an interface up/down ***
*** stack: Stack turn on/off ***
*** heapinfo: Check heap status ***
*** ping: Ping an IP address ***
*** arp: ARP commands ***
*** dnsc: DNS client commands ***
***
```

32. As a first simple test you can ping an external Server or anything else you like (Maybe the board of your class neighbor?)

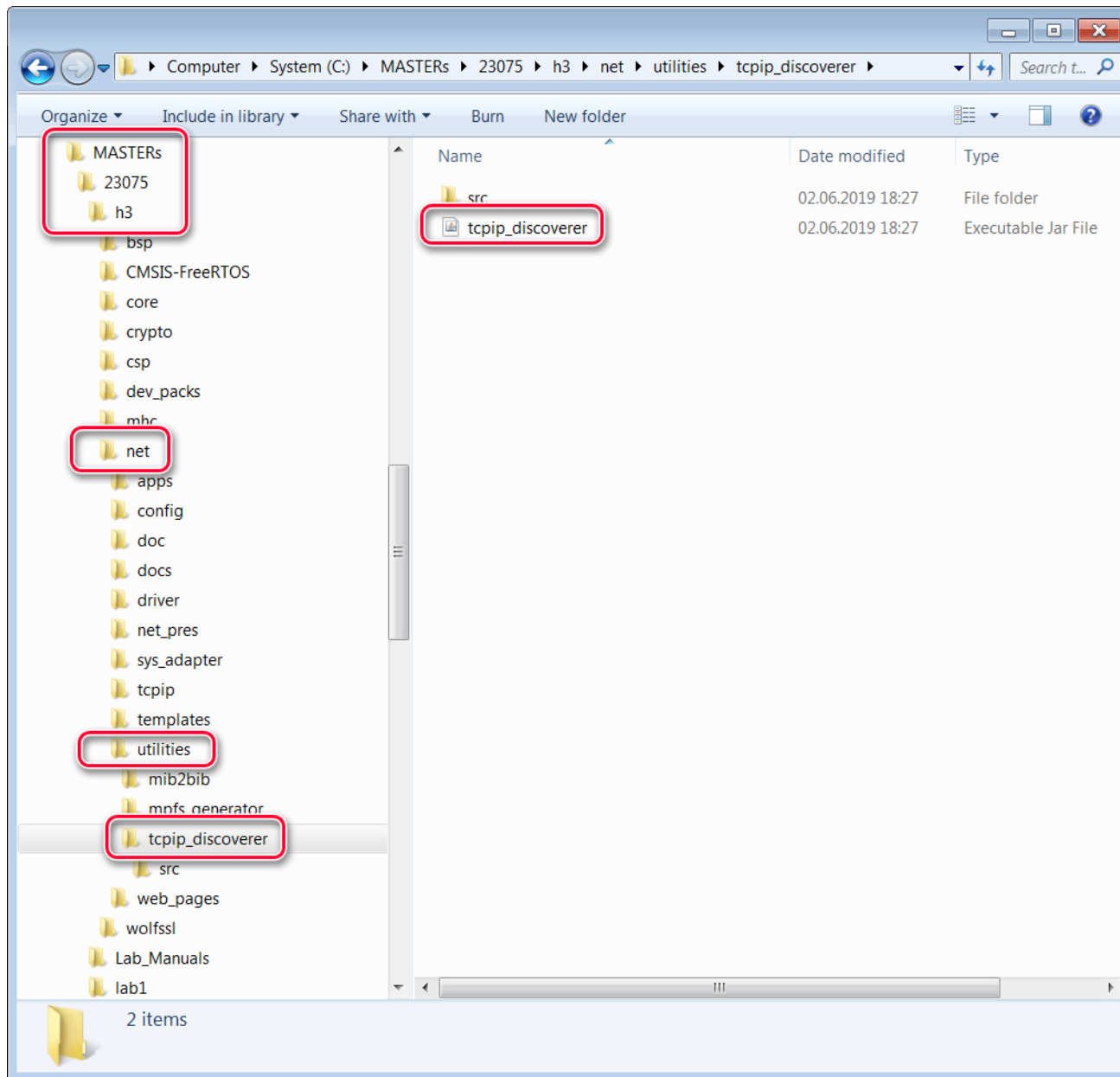


```
COM96:115200baud - Tera Term VT
File Edit Setup Control Window Help

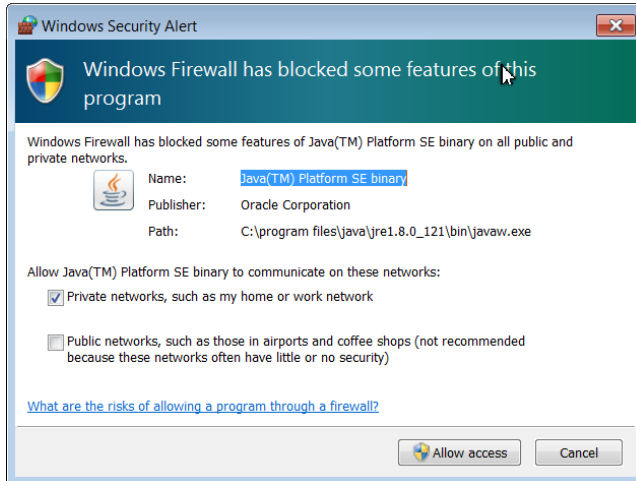
>
=====
web_net_server_nvm_mpfs_freertos_lab1 Jun  3 2019 19:30:07
SYS_Initialize: The MPFS2 File System is mounted
MAC TCP/IP_HOSTS_CONFIGURATION[0].macAddr: fc:c2:3d:0c:20:44
TCP/IP Stack: Initialization Started
TCP/IP Stack: Initialization Ended - success
Interface GMAC on host MARTIN_RUPPERT - NBNS enabled
GMAC IP Address: 0.0.0.0
GMAC IP Address: 192.168.0.17

>ping www.google.com
Ping: resolving host: www.google.com
>Ping: reply[1] from 172.217.23.132: time = 25ms
Ping: reply[2] from 172.217.23.132: time = 26ms
Ping: reply[3] from 172.217.23.132: time = 25ms
Ping: reply[4] from 172.217.23.132: time = 25ms
Ping: done. Sent 4 requests, received 4 replies.
```

33. Select the `tcpip_discoverer` tool from

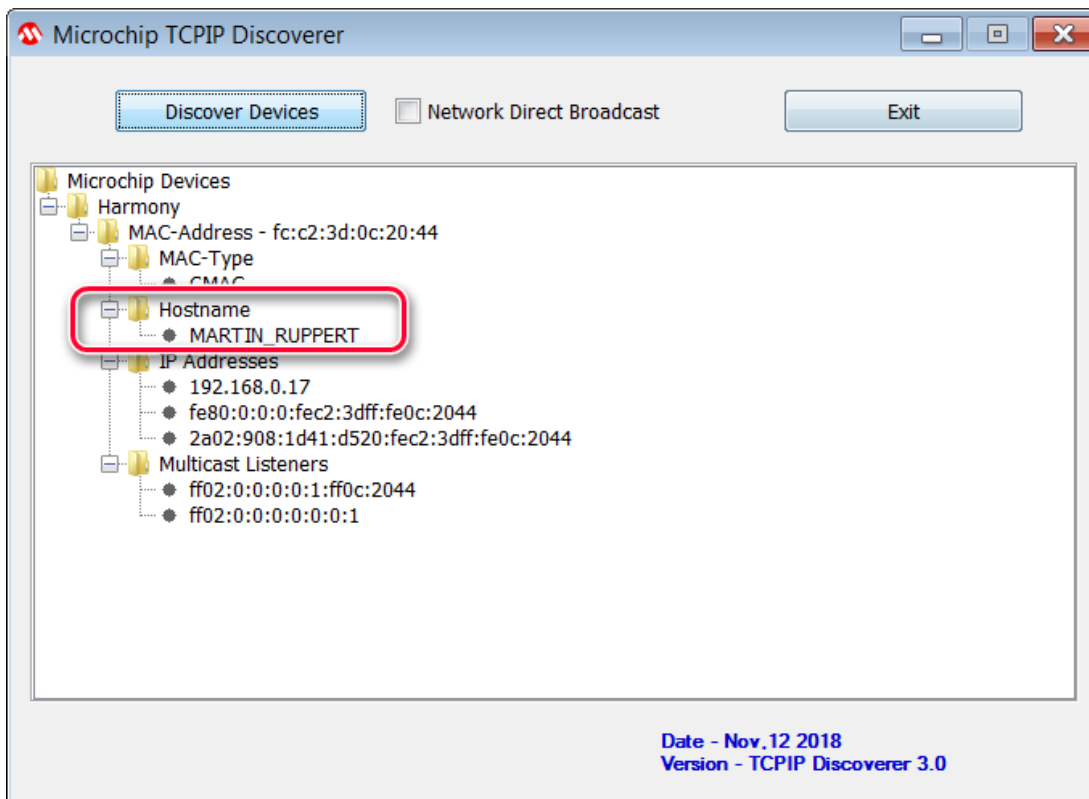


34. If Windows is asking for permissions allow the access



35. The TCP Discover should list all boards in the classroom. We have made this tool to help you to find your board in the network. The source codes of this tool (Java) are part of the H3.

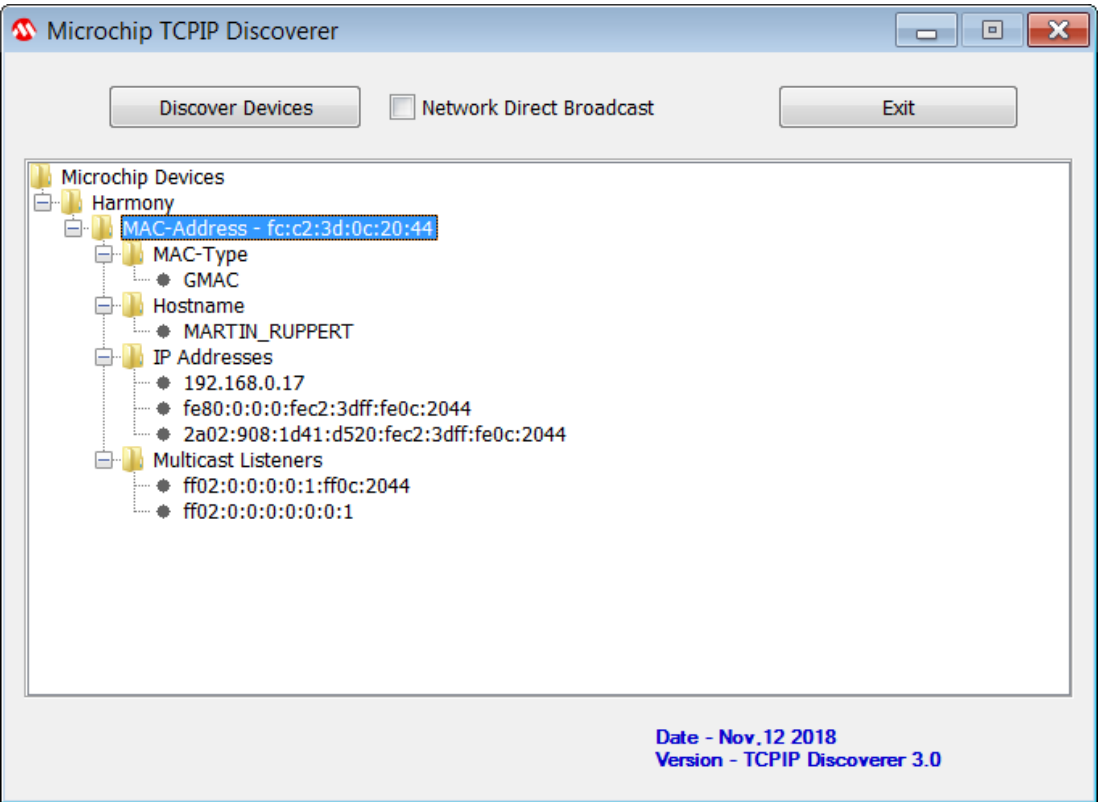
You can identify your board by the Host Name that has select in an earlier step. The Host Name is also known to the DHCP server and is listed in their typical Web Interfaces as a connected device.



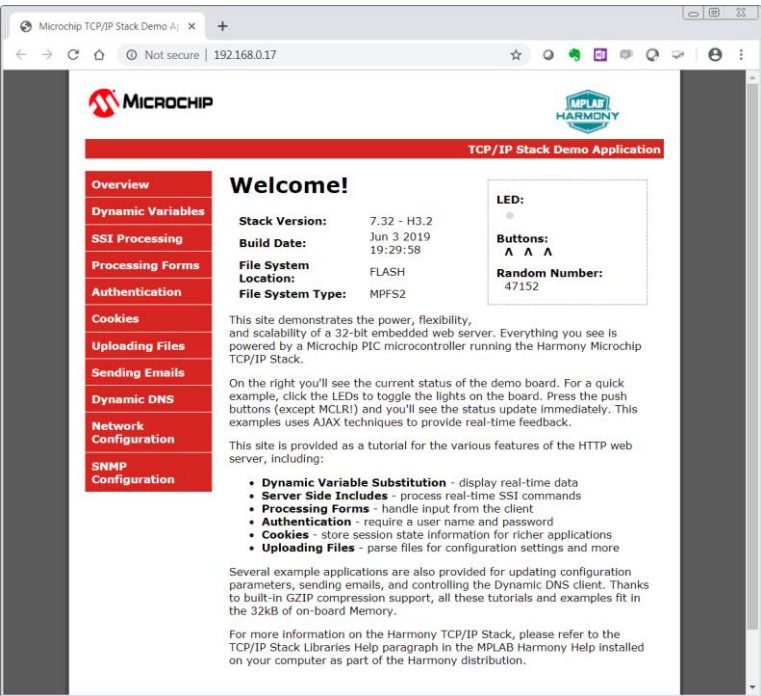
Press the **Discover Devices** button: The tool will send a UDP broadcast on port 30303, with the packet "Discovery, who is out there?" All H3 devices running the Announce service will respond to this broadcast, by sending a return broadcast on port 30303. The broadcast packet contains data on the type of interface used, the Host Name, MAC and IP Address. The Discover tool listens to all broadcasts on port 30303 and will show found devices under the Microchip

Devices tree. You can identify your device by looking for the host name that you entered in MHC Setup process. The Microchip TCP/IP Discoverer tool also shows the IP address for your board.

36. A double click on the **MAC-Address** line will put you in your default Internet Browser



37. And the Webpage is displayed. Please take some time an play with the sub menus to find out the capabilities of or H3 Web Server



37. Congratulations, you have completed Lab 1!

Harmony TCP/IP API Subset For Lab 2

TCP Socket Management Functions

TCPIP_TCP_ArrayGet Function

This function reads an array of data bytes from a TCP socket's RX buffer/FIFO. The data is removed from the FIFO in the process.

Function Prototype

```
uint16_t TCPIP_TCP_ArrayGet(  
    TCP_SOCKET hTCP,  
    uint8_t* buffer,  
    uint16_t len  
);
```

Preconditions

TCP is initialized.

Parameters

Parameter	Description
hTCP	The socket from which data is to be read.
buffer	Pointer to the array to store data that was read.
len	Number of bytes to be read.

Returns

Type	Description
uint16_t	The number of bytes read from the socket. If less than len , the RX FIFO buffer became empty or the socket is not connected.

TCPIP_TCP_ClientOpen Function

Provides a unified method for opening TCP client sockets. Sockets are created at the TCP module initialization, and can be claimed with this function and freed using **TCPIP_TCP_Abort** or **TCPIP_TCP_Close**. If the `remoteAddress != 0` (and the address pointed by `remoteAddress != 0`) then the socket will immediately initiate a connection to the remote host.

Function Prototype

```
TCP_SOCKET TCPIP_TCP_ClientOpen(  
    IP_ADDRESS_TYPE addType,  
    TCP_PORT remotePort,  
    IP_MULTI_ADDRESS* remoteAddress  
);
```

Preconditions

TCP is initialized.

Parameters

Parameter	Description
addType	The type of address being used. Valid values are: IP_ADDRESS_TYPE_IPV4 or IP_ADDRESS_TYPE_IPV6
remotePort	TCP port to connect to. The local port for client sockets will be automatically picked by the TCP module.
remoteAddress	The remote address to be used

Returns

Type	Description
TCP_SOCKET	Handle - Save this handle and use it when calling all other TCP APIs. If no sockets of the specified type were available to be opened, the handle will contain a value equal to INVALID_SOCKET .

TCPIP_TCP_Close Function

Graceful Option Set: If the graceful option is set for the socket (default), a TCPIP_TCP_Disconnect will be tried. If the linger option is set (default) the TCPIP_TCP_Disconnect will try to send any queued TX data before issuing FIN. If the FIN send operation fails or the socket is not connected the abort is generated.

Graceful Option Not Set: If the graceful option is not set, or the previous step could not send the FIN, a TCPIP_TCP_Abort is called, sending a RST to the remote node. Communication is closed, the socket is no longer valid and the associated resources are freed.

Function Prototype

```
void TCPIP_TCP_Close(
    TCP\_SOCKET hTCP
);
```

Preconditions

TCP socket should have been opened with TCPIP_TCP_ServerOpen/TCPIP_TCP_ClientOpen.

hTCP - valid socket

Parameters

Parameter	Description
hTCP	Handle to the socket to disconnect and close.

Returns

Type	Description
Void	None

TCPIP_TCP_GetIsReady Function

Call this function to determine how many bytes can be read from the TCP RX buffer. If this function returns zero, the application must return to the main stack loop before continuing in order to wait for more data to arrive.

Function Prototype

```
uint16_t TCPIP_TCP_GetIsReady(  
    TCP\_SOCKET hTCP  
);
```

Preconditions

TCP is initialized.

Parameters

Parameter	Description
hTCP	The socket to check.

Returns

Type	Description
uint16_t	The number of bytes available to be read from the TCP RX buffer.

TCPIP_TCP_IsConnected Function

This function determines if a socket has an established connection to a remote node. Call this function after calling `TCPIP_TCP_ServerOpen()/TCPIP_TCP_ClientOpen()` to determine when the connection is set up and ready for use.

Function Prototype

```
bool TCPIP_TCP_IsConnected(  
    TCP\_SOCKET hTCP  
);
```

Preconditions

TCP is initialized.

Parameters

Parameter	Description
hTCP	The TCP socket to check.

Returns

Type	Description
bool	True: the socket is connected False: the socket is disconnected

TCPIP_TCP_PutIsReady Function

Call this function to determine how many bytes can be written to the TCP TX buffer. If this function returns zero, the application must return to the main stack loop before continuing in order to transmit more data.

Function Prototype

```
uint16_t TCPIP_TCP_PutIsReady(  
    TCP\_SOCKET hTCP  
);
```

Preconditions

TCP is initialized.

Parameters

Parameter	Description
hTCP	The socket from which data is to be written.

Returns

Type	Description
uint16_t	The number of bytes available to be written in the TCP TX buffer.

TCPIP_TCP_StringPut Function

This function writes a null-terminated string to a TCP socket. The null-terminator is not copied to the socket.

Function Prototype

```
const uint8_t* TCPIP_TCP_StringPut(
    TCP_SOCKET hTCP,
    const uint8_t* Data
);
```

Preconditions

TCP is initialized.

Parameters

Parameter	Description
hTCP	The socket from which data is to be written.
const uint8_t*	Data

Returns

Type	Description
const uint8_t*	Pointer to the byte following the last byte written to the socket. If this pointer does not dereference to a NULL byte, the buffer became full or the socket is not connected.

TCPIP_TCP_WasReset Function

This function is a self-clearing semaphore indicating whether or not a socket has been disconnected since the previous call. This function works for all possible disconnections: a call to **TCPIP_TCP_Disconnect**, a FIN from the remote node, or an acknowledgment timeout caused by the loss of a network link. It also returns true after the first call to **TCPIP_TCP_Initialize**. Applications should use this function to reset their state machines.

Function Prototype

```
bool TCPIP_TCP_WasReset(
    TCP_SOCKET hTCP
);
```

Preconditions

TCP is initialized.

Parameters

Parameter	Description
hTCP	The TCP socket to check.

Returns

Type	Description
bool	true: the socket was disconnected since the previous call false: the socket remained connected since the previous call

UDP Socket Management Functions

TCPIP_UDP_ArrayGet Function

This function reads an array of bytes from the UDP socket, while adjusting the current read pointer and decrementing the remaining bytes available. TCPIP_UDP_GetIsReady should be used before calling this function to get the number of the available bytes in the socket.

Function Prototype

```
uint16_t TCPIP_UDP_ArrayGet(  
    UDP\_SOCKET hUDP,  
    uint8_t * cData,  
    uint16_t wDataLen  
);
```

Preconditions

UDP socket should have been opened with TCPIP_UDP_ServerOpen/TCPIP_UDP_ClientOpen.

hUDP - valid socket

Parameters

Parameter	Description
hUDP	UDP Socket Handle
cData	The buffer to receive the bytes being read. If NULL, the bytes are simply discarded
wDataLen	Number of bytes to be read from the socket.

Returns

Type	Description
uint16_t	The number of bytes successfully read from the UDP buffer. If this value is less than wDataLen, then the buffer was emptied and no more data is available.

TCPIP_UDP_Close Function

Closes a UDP socket and frees the handle. Call this function to release a socket and return it to the pool for use by future communications.

Function Prototype

```
void TCPIP_UDP_Close(  
    UDP\_SOCKET hUDP  
);
```

Preconditions

UDP socket should have been opened with TCPIP_UDP_ServerOpen/TCPIP_UDP_ClientOpen.

hUDP - valid socket

Parameters

Parameter	Description
hUDP	UDP Socket Handle

Returns

Type	Description
void	None

TCPIP_UDP_GetIsReady Function

This function will return the number of bytes that are available in the specified UDP socket RX buffer. The UDP socket queues incoming RX packets in an internal queue. If currently there is no RX packet processed (as a result of retrieving all available bytes with [TCPIP_UDP_ArrayGet](#), for example), this call will advance the RX packet to be processed to the next queued packet. If a RX packet is currently processed, the call will return the number of bytes left to be read from this packet.

Function Prototype

```
uint16_t TCPIP_UDP_GetIsReady(  
    UDP\_SOCKET hUDP  
);
```

Preconditions

UDP socket should have been opened with [TCPIP_UDP_ServerOpen](#)/[TCPIP_UDP_ClientOpen](#).

hUDP parameter is a valid socket

Parameters

Parameter	Description
hUDP	UDP Socket Handle

Returns

Type	Description
uint16_t	The number of bytes that can be read from the socket.

TCPIP_UDP_ServerOpen Function

Provides a unified method for opening UDP server sockets.

Function Prototype

```
UDP_SOCKET TCPIP_UDP_ServerOpen(
    IP_ADDRESS_TYPE addType,
    UDP_PORT localPort,
    IP_MULTI_ADDRESS* localAddress
);
```

Preconditions

UDP is initialized.

Parameters

Parameter	Description
IP_ADDRESS_TYPE addType	The type of address being used. IP_ADDRESS_TYPE_IPV4 or IP_ADDRESS_TYPE_IPV6 .
UDP_PORT localPort	UDP port on which to listen for connections
IP_MULTI_ADDRESS* localAddress	Local IP address to use. Can be 0 (NULL) if any incoming interface will do.

Returns

Type	Description
UDP_SOCKET	Handle - Save this handle and use it when calling all other UDP APIs. If no sockets of the specified type were available to be opened, the handle will contain a value equal to INVALID_SOCKET .

TCPIP_UDP_SocketInfoGet Function

This function will fill a user passed [UDP_SOCKET_INFO](#) structure with status of the selected socket

Function Prototype

```
bool TCPIP_UDP_SocketInfoGet(
    UDP_SOCKET hUDP,
    UDP_SOCKET_INFO* pInfo
);
```

Preconditions

UDP socket should have been opened with [TCPIP_UDP_ServerOpen\(\)](#)/[TCPIP_UDP_ClientOpen\(\)](#).

hUDP - valid socket

pInfo - valid address of a [UDP_SOCKET_INFO](#) structure

Parameters

Parameter	Description
hUDP	UDP Socket Handle
pInfo	Pointer to UDP_SOCKET_INFO to receive socket information

Returns

Type	Description
bool	true if call succeeded false if no such socket or invalid pinfo.

UDP_SOCKET_INFO Structure

Holds information about a UDP Socket

Structure

```
typedef struct {
    IP_ADDRESS_TYPE addressType;
    IP_MULTI_ADDRESS remoteIPAddress;
    IP_MULTI_ADDRESS localIPAddress;
    IP_MULTI_ADDRESS sourceIPAddress;
    IP_MULTI_ADDRESS destIPAddress;
    UDP_PORT remotePort;
    UDP_PORT localPort;
    TCPIP_NET_HANDLE hNet;
} UDP_SOCKET_INFO;
```

Members

Type	Member Name	Description
IP_ADDRESS_TYPE	addressType	address type of the socket
IP_MULTI_ADDRESS	remoteIPAddress	current socket destination address
IP_MULTI_ADDRESS	localIPAddress	current socket source address
IP_MULTI_ADDRESS	sourceIPAddress	source address of the last packet
IP_MULTI_ADDRESS	destIPAddress	destination address of the last packet
UDP_PORT	remotePort	Port number associated with remote node
UDP_PORT	localPort	local port number
TCPIP_NET_HANDLE	hNet	associated interface

Network Communications Controller Application

Code Modification Solutions

Item	Source Line	Description & Solution
1	253	<p>Open a UDP Server Instance, IPv4 address type, Port is defined in the ECS_BROADCAST_PORT constant, Accept connections on ANY interface. Store returned handle in the ecsUDPBroadcastHandle variable</p> <pre>ecsUDPBroadcastHandle = TCPIP_UDP_ServerOpen(IP_ADDRESS_TYPE_IPV4, ECS_BROADCAST_PORT, 0);</pre>
2	269	<p>Check if any data has been received on the UDP Socket</p> <pre>TCPIP_UDP_GetIsReady(ecsUDPBroadcastHandle)</pre>
3	296	<p>Close the UDP Server Instance</p> <pre>TCPIP_UDP_Close(ecsUDPBroadcastHandle);</pre>
4	316	<p>Open a TCP Client Instance, IPv4 Address Type, Destination Port is defined in ECS_TCP_PORT constant. IP Address is stored in ecsUDPSocketInfo.sourceIPAddress structure. Store returned handle in the ecsTCPCHandle variable. Hint: You must pass the address of ecsUDPSocketInfo.sourceIPAddress to the Harmony API that is used to open the TCP Client.</p> <pre>ecsTCPCHandle = TCPIP_TCP_ClientOpen(IP_ADDRESS_TYPE_IPV4, ECS_TCP_PORT, &ecsUDPSocketInfo.sourceIPAddress);</pre>
5	333	<p>Check if the TCP Client is connected to the Server</p> <pre>TCPIP_TCP_IsConnected(ecsTCPCHandle)</pre>
6	365	<p>Check if the TCP connection has dropped out any time between the current and previous execution of the NETWORKCOMS_MANAGE_TCP_CLIENT state.</p> <pre>TCPIP_TCP_WasReset(ecsTCPCHandle)</pre>
7	417	<p>Get the amount of free space available in the Transmit Buffer of the TCP Socket</p> <pre>TCPIP_TCP_PutIsReady(ecsTCPCHandle)</pre>
8	431	<p>Send the string stored in callReq_JSONPacket array to the TCP Server.</p> <pre>TCPIP_TCP_StringPut(ecsTCPCHandle, callReq_JSONPacket);</pre> <p>TCPIP_TCP_ArrayPut can also be used, but the length of the string needs to be specified e.g.</p> <pre>TCPIP_TCP_ArrayPut(ecsTCPCHandle, callReq_JSONPacket, strlen(callReq_JSONPacket));</pre> <p>Alternatively the size of the array can be specified, but only when the array size is equal to the string size.</p> <pre>TCPIP_TCP_ArrayPut(ecsTCPCHandle, callReq_JSONPacket, sizeof(callReq_JSONPacket));</pre>

MPLAB® Harmony TCP/IP Stack

TCP Module API Function List

Socket Management Functions

TCPIP_TCP_ServerOpen	Opens a TCP socket as a server.
TCPIP_TCP_ClientOpen	Opens a TCP socket as a client.
TCPIP_TCP_Close	Disconnects an open socket and destroys the socket handle, releasing the associated resources.
TCPIP_TCP_Connect	Connects a client socket.
TCPIP_TCP_Bind	Binds a socket to a local address.
TCPIP_TCP_RemoteBind	Binds a socket to a remote address.
TCPIP_TCP_IsConnected	Determines if a socket has an established connection.
TCPIP_TCP_WasReset	Self-clearing semaphore indicating socket reset.
TCPIP_TCP_Disconnect	Disconnects an open socket.
TCPIP_TCP_Abort	Aborts a connection.
TCPIP_TCP_OptionsGet	Allows getting the options for a socket like: current RX/TX buffer size, etc.
TCPIP_TCP_OptionsSet	Allows setting options to a socket like adjust RX/TX buffer size, etc.
TCPIP_TCP_SocketInfoGet	Obtains information about a currently open socket.
TCPIP_TCP_SocketNetGet	Gets the current network interface of an TCP socket.
TCPIP_TCP_SocketNetSet	Sets the interface for an TCP socket
TCPIP_TCP_SignalHandlerDeregister	Deregisters a previously registered TCP socket signal handler.
TCPIP_TCP_SignalHandlerRegister	Registers a TCP socket signal handler.
TCPIP_TCP_Task	Standard TCP/IP stack module task function.

Transmit Data Functions

TCPIP_TCP_Put	Writes a single byte to a TCP socket.
TCPIP_TCP_PutIsReady	Determines how much free space is available in the TCP TX buffer.
TCPIP_TCP_StringPut	Writes a null-terminated string to a TCP socket.
TCPIP_TCP_ArrayPut	Writes an array from a buffer to a TCP socket.
TCPIP_TCP_Flush	Immediately transmits all pending TX data.
TCPIP_TCP_FifoTxFullGet	Determines how many bytes are pending in the TCP TX FIFO.
TCPIP_TCP_FifoTxFreeGet	Determines how many bytes are free and could be written in the TCP TX FIFO.

Receive Data Transfer Functions

TCPIP_TCP_ArrayFind	Searches for a string in the TCP RX buffer.
TCPIP_TCP_Find	Searches for a byte in the TCP RX buffer.
TCPIP_TCP_Get	Retrieves a single byte to a TCP socket.
TCPIP_TCP_Peek	Peaks at one byte in the TCP RX buffer/FIFO without removing it from the buffer.
TCPIP_TCP_Discard	Discards any pending data in the RCP RX FIFO.
TCPIP_TCP_FifoRxFreeGet	Determines how many bytes are free in the RX buffer/FIFO.
TCPIP_TCP_FifoSizeAdjust	Adjusts the relative sizes of the RX and TX buffers.
TCPIP_TCP_FifoRxFullGet	Determines how many bytes are pending in the RX buffer/FIFO.
TCPIP_TCP_GetIsReady	Determines how many bytes can be read from the TCP RX buffer.
TCPIP_TCP_ArrayGet	Reads an array of data bytes from a TCP socket's RX buffer/FIFO.
TCPIP_TCP_ArrayPeek	Reads a specified number of data bytes from the TCP RX buffer/FIFO without removing them from the buffer.



MICROCHIP