



# MICROCHIP MASTERS

2019

## 23075 IoT6

# Simplifying TCP/IP Applications with MPLAB® Harmony



# Class Objectives

**When you walk out of this class you will be able to...**

- Understand the fundamentals of TCP/IP and how applications use them to create TCP/IP connections.
- Understanding Network analysis with the well-known tool Wireshark
- Learn the architecture and the fundamentals of the Harmony TCP/IP stack to interface your application with some common stack API's in a FreeRTOS Task Scheduler environment

# Agenda – Lecture

## Part 1: TCP/IP Fundamentals as a Refresher

## Part 2: MPLAB® Harmony TCP/IP Stack Overview

- TCP/IP Layers and Features
- Network Interface Options
- Processor Requirements

## Part 3: Using the MPLAB® Harmony TCP/IP Stack

- Creating a new project, stack configuration and **connectivity check (Lab 1)**
- MPLAB® Harmony TCP/IP Stack APIs
- Application integration for **local access**, using the example of a Vending machine (**Lab 2**)
- Application integration for **external access** using the example of a Weather Service (**Lab 3**)



# Agenda – Labs

- **Lab 1:** Creating a new project, Stack Configuration and **Connectivity Check**
- **Lab 2:** Application integration for **local access**, using the example of a Vending machine
- **Lab 3:** Application integration for **external access**, using the example of a Weather Service



# MICROCHIP MASTERS

2019

## Part 1: TCP/IP Fundamentals as a Refresher

- Five Layer Model and Applications
- TCP vs UDP



# Agenda

## Five Layer Model and Applications

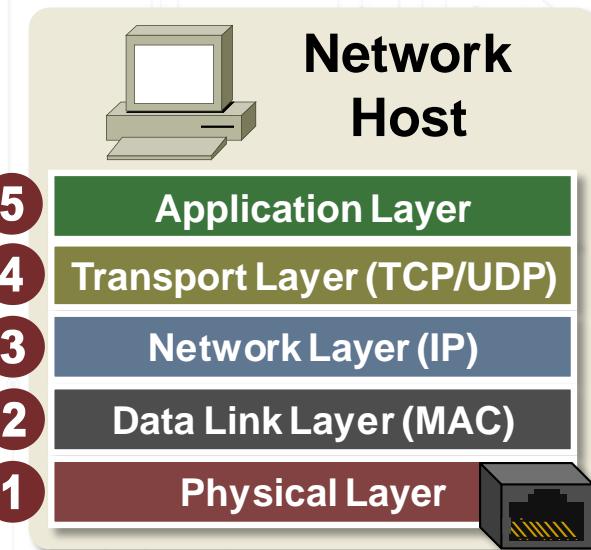
- **Five Layer Software Model**

- Application Layer
- Transport Layer
- Network Layer
- Data Link Layer
- Physical Layer

- **TCP vs. UDP**

- **TCP/IP Applications**

- DNS, NBNS, SNTP, DHCP, SNMP, Telnet, SMTP, HTTP





# Basic Needs for TCP/IP Transmissions

- Need to Specify:
  - Most reliable or fastest transmissions
  - Where we want the data delivered



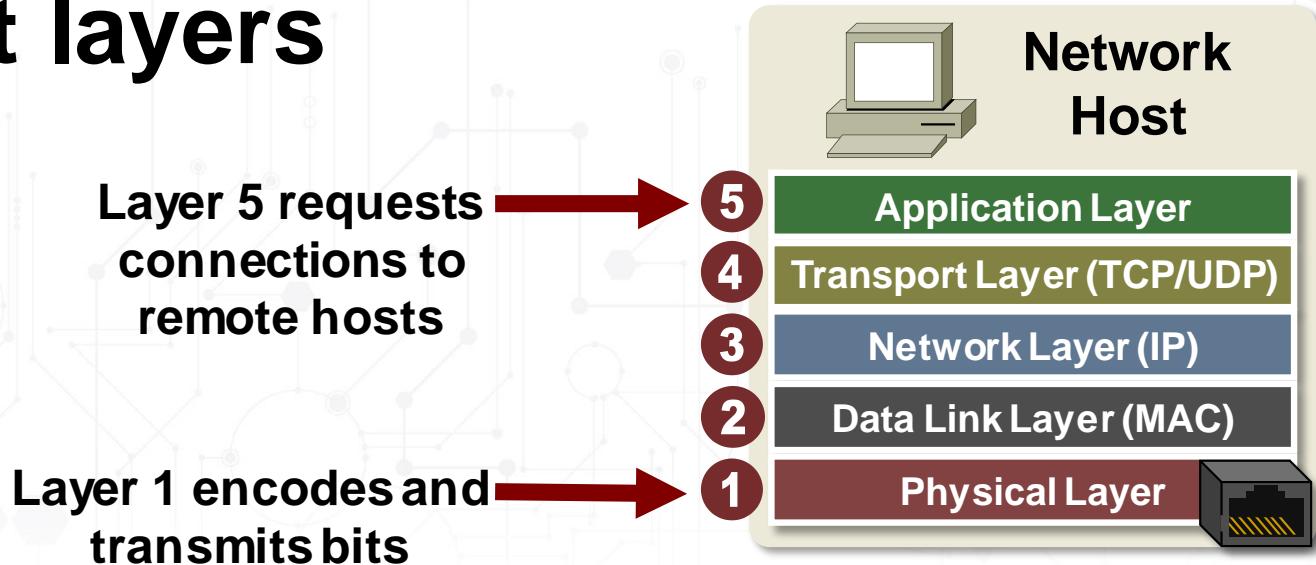
Needs to be sent with all transmitted data.



- Also need to physically transmit the data

# TCP/IP 5 Layer Model

- Each layer performs a specific task needed to move our data across the network
- Each layer only communicates with adjacent layers



# What Does Each Layer Do?

5

## Application Layer

The Application layer is the group of applications requiring network communications.

Host A  
Web Browser

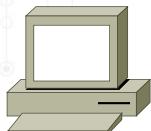
Generates the data and requests connections

Host B  
Web Server

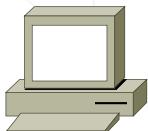
4

## Transport Layer (TCP/UDP)

The Transport layer establishes the connection between applications on different hosts.



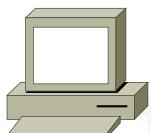
Establishes connections with remote host



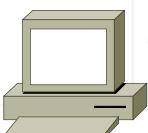
3

## Network Layer (IP)

The Network layer is responsible for creating the packets that move across the network.



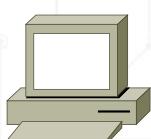
Transfers packets with virtual (IP) addresses



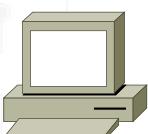
2

## Data Link Layer (MAC)

The Data Link layer is responsible for creating the frames that move across the network.



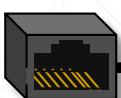
Transfers frames with physical (MAC) addresses



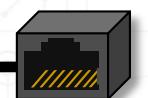
1

## Physical Layer

The Physical layer is the transceiver that drives the signals on the network.

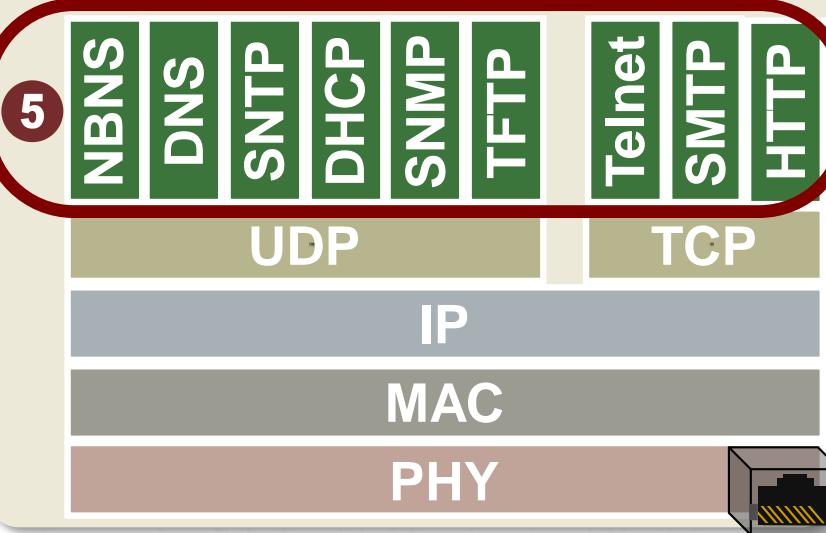
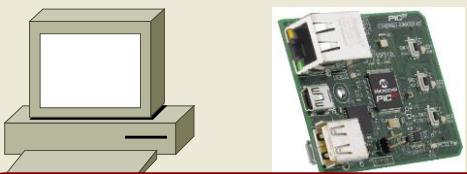


Transmits and receives bits



# Application Layer (Layer 5)

## Network Host



- **Layer 5 is where TCP/IP applications live**
- **Your application typically interacts with these applications**

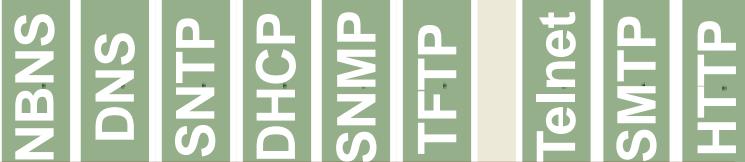
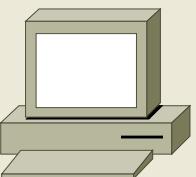


# Transport Layer

## (Layer 4)

- **Connects to remote hosts using either:**
  - TCP (Transfer Control Protocol)
  - UDP (User Datagram protocol)
- **Delivers data to and from applications**
- **Assigns port numbers to processes running in applications**

### Network Host



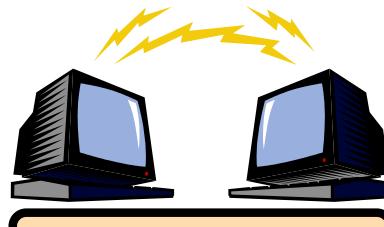
4



# TCP vs. UDP



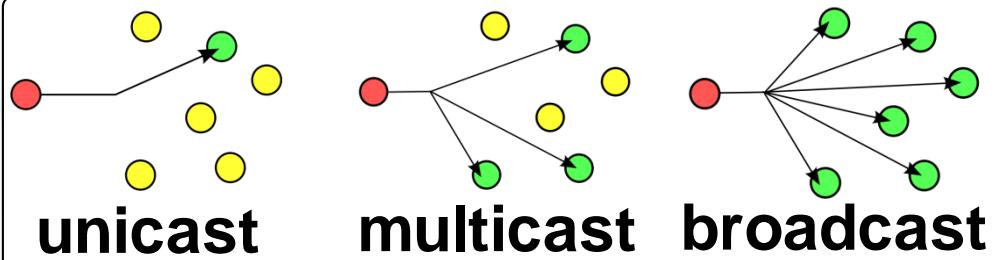
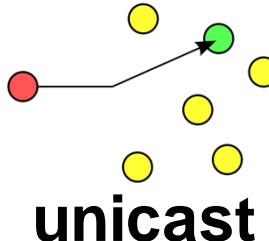
**TCP**



**UDP**

- Slower but reliable transfers
- Typical applications:
  - Email
  - Web browsing

- Fast but non-guaranteed transfers (“best effort”)
- Typical applications:
  - VoIP
  - Internet Radio



# TCP and UDP Header Formats

## Reference Slide

### TCP Segment Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Sequence Number							
64	Acknowledgment Number							
96	Data Offset	Res	Flags	Window Size				
128	Header and Data Checksum				Urgent Pointer			
160...	Options							

### UDP Datagram Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

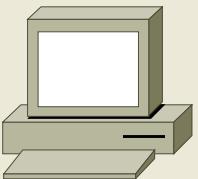


# Network Layer (Layer 3)



An IP address is 32 bits and looks like this:  
192.168.1.101

## Network Host



NBNS

DNS

SNTP

DHCP

SNMP

TFTP

Telnet

SMTP

HTTP

UDP

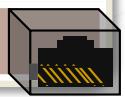
TCP

3

IP

MAC

PHY



- **Also known as the Internet layer**
- **Adds a header to the data received from the transport layer**
  - contains the source and destination IP addresses
- **Creates IP Packets**

# IPv4 Packet Header Format (Reference Slide)

## IPv4 Packet Header Format

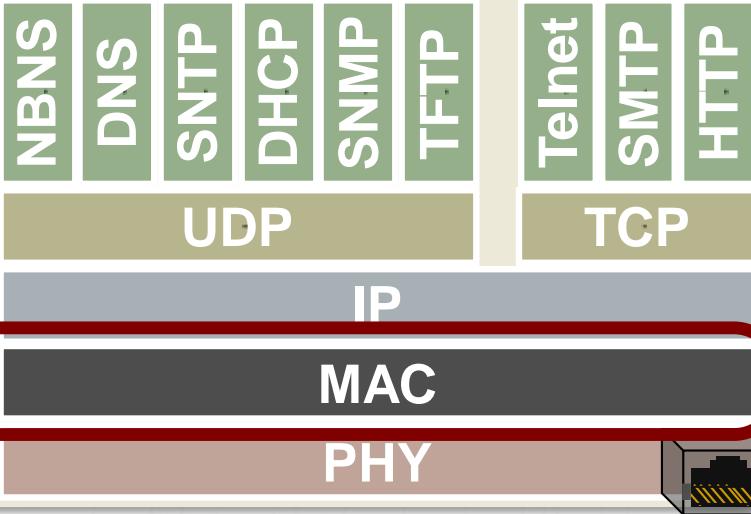
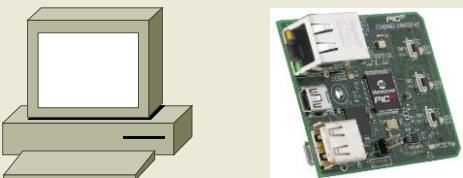
Bit #	0	7	8	15	16	23	24	31								
0	Version	IHL	DSCP	ECN	Total Length											
32	Identification					Flags	Fragment Offset									
64	Time to Live		Protocol			Header Checksum										
96	Source IP Address															
128	Destination IP Address															
160	Options (if IHL > 5)															

# Data Link Layer (Layer 2)



A MAC address is 48 bits and looks like this:  
F0:DE:F1:1E:E8:93

## Network Host



- **Uses a Media Access Controller to generate frames**
- **Adds a header to the packet**
  - Source and destination MAC addresses
- **Every host has at least one MAC address**

# Ethernet and Wi-Fi® Frame Formats (Reference Slide)

## Ethernet (802.3) Frame Format

7 bytes	1 byte	6 bytes	6 bytes	2 bytes	42 to 1500 bytes	4 bytes	12 bytes
Preamble	Start of Frame Delimiter	Destination MAC Address	Source MAC Address	Type	Data (payload)	CRC	Inter-frame gap



For TCP/IP communications,  
the payload for a frame is a  
packet



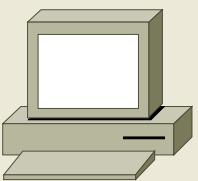
## Wi-Fi (802.11) Frame Format

2 bytes	2 bytes	6 bytes	6 bytes	6 bytes	2 bytes	6 bytes	0 to 2312 bytes	4 bytes
Frame Control	Duration	MAC Address 1 (Destination)	MAC Address 2 (Source)	MAC Address 3 (Router)	Seq Control	MAC Address 4 (AP)	Data (payload)	CRC



# Physical Layer (Layer 1)

## Network Host



NBNS

DNS

SNTP

DHCP

SNMP

TFTP

Telnet

SMTP

HTTP

UDP

TCP

IP

MAC

PHY



1

- Sends and receives signals on the physical wire or antenna
- Responsible for moving bits
- Found at the end of every network interface



# Transmit Data Using Network Layers

5

Application Layer

I want to download a web page  
from this address: 192.168.1.102

Message

4

Transport Layer (TCP/UDP)

Source Port = 31,244  
Destination Port = 80

I want to download a web page  
from this address: 192.168.1.102

Segment/Datagram

3

Network Layer (IP)

Source IP Addr = 192.168.1.101  
Dest IP Addr = 192.168.1.102

Source Port = 31,244  
Destination Port = 80

Message

Packet

2

Data Link Layer (MAC)

Source MAC Addr = 00:12:F1:1E:E8:93  
Dest MAC Addr = 00:04:A3:4D:1C:73

Src IP Addr  
Dest IP Addr

Src Port #  
Dest Port #

Message

Frame

1

Physical  
Layer





# Receive Data Using Network Layers

5

Application Layer

I want to download a web page  
from this address: 192.168.1.102

Message

4

Transport Layer (TCP/UDP)

Source Port = 31,244  
Destination Port = 80

I want to download a web page  
from this address: 192.168.1.102

Segment/Datagram

3

Network Layer (IP)

Source IP Addr = 192.168.1.101  
Dest IP Addr = 192.168.1.102

Source Port = 31,244  
Destination Port = 80

Message

Packet

2

Data Link Layer (MAC)

Source MAC Addr = 00:12:F1:1E:E8:93  
Dest MAC Addr = 00:04:A3:4D:1C:73

Src IP Addr  
Dest IP Addr

Src Port #  
Dest Port #

Message

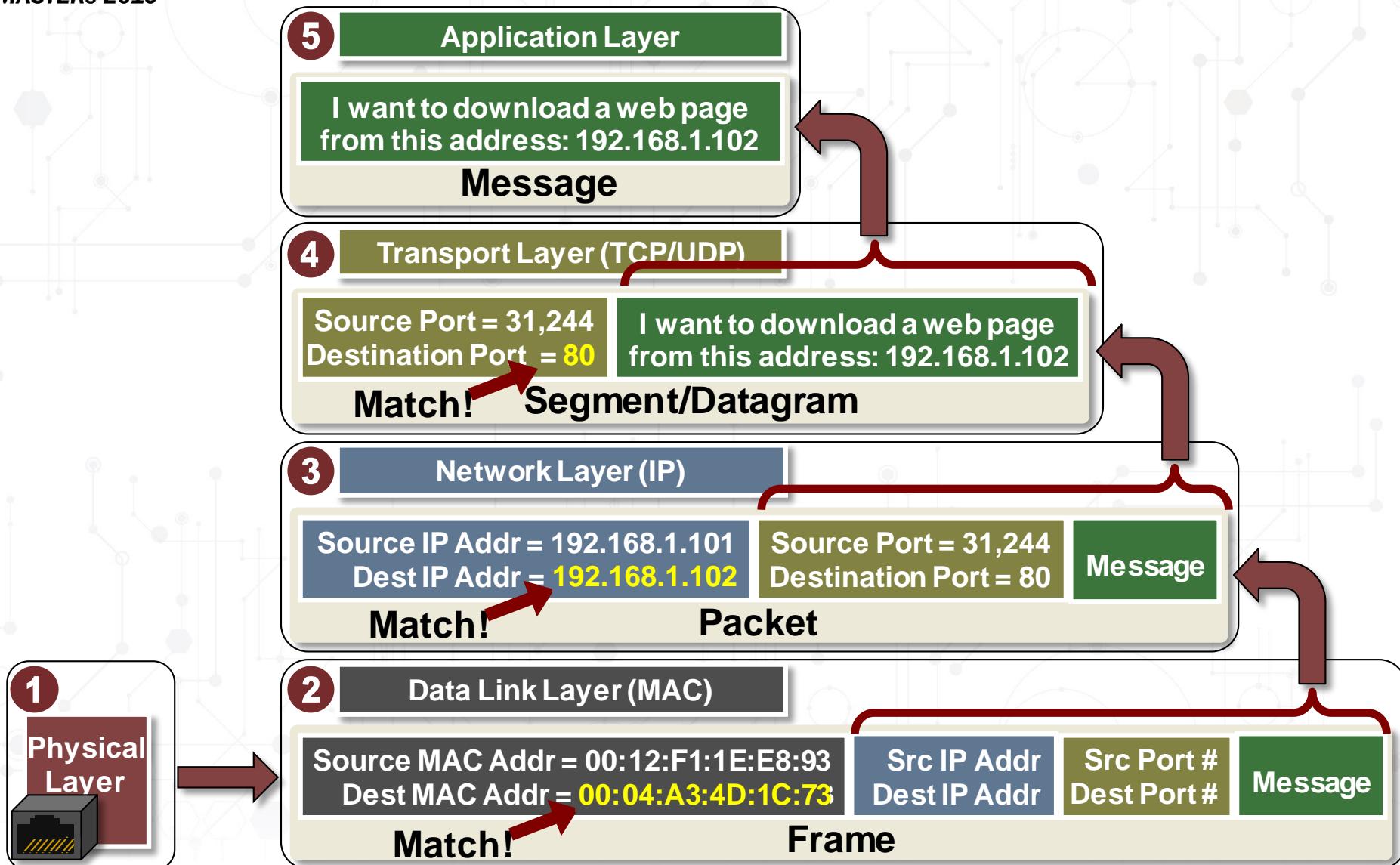
Frame

1

Physical  
Layer



# Receive Data Using Network Layers



# TCP/IP Protocol Stack

## (Terminology Reference)

Layer #	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc...	Messages	n/a
4	Transport	TCP/UDP	Segments/ Datagrams	Port #s
3	Network or Internet	IP	Packets	IP Address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

# Common TCP/IP Applications

Application	Description
DHCP	Dynamic Host Configuration Protocol assigns IP addresses
DNS	Domain Name System translates website names to IP addresses
HTTP	Hypertext Transfer Protocol used to transfer web pages
NBNS	NetBIOS Name Service translates local host names to IP addresses
SMTP	Simple Mail Transfer Protocol sends email messages
SNMP	Simple Network Management Protocol manages network devices
SNTP	Simple Network Time Protocol provides time of day
Telnet	Bi-directional text communication via a terminal application
TFTP	Trivial File Transfer Protocol used to transfer small amounts of data



# Knowledge Check



When the Transport layer receives a message from a remote host, how does it know which application process to deliver it to?

## Port Numbers



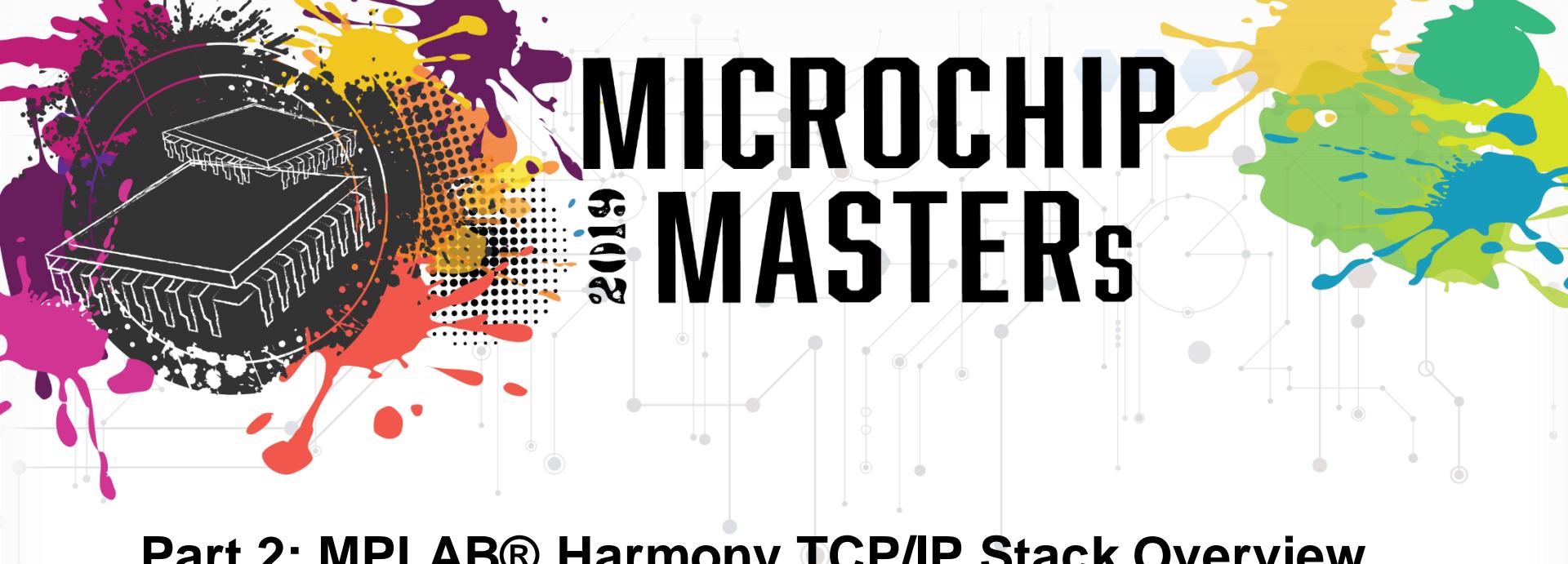
# Knowledge Check



**True or false:  
The Simple Mail Transport Protocol  
(SMTP) application uses the UDP  
transport protocol.**

**False**

**SMTP uses TCP to guarantee error free delivery.**



# MICROCHIP MASTERS

2019

## Part 2: MPLAB® Harmony TCP/IP Stack Overview

- TCP/IP Layers and Features
- Network Interface Options
- Processor Requirements





# MPLAB® Harmony TCP/IP Stack

- Provides a foundation for embedded network applications by handling most of the interaction required between the physical network interface and your application.
- Fully Implemented and supported by Microchip Engineers.
- Source code is included with the MPLAB® Harmony Distribution.
- Cost: Free\*



\*MPLAB Harmony is only licensed for use on Microchip Microcontrollers



# Architecture

PIC32 Microcontroller running MPLAB Harmony TCP/IP Stack

## PIC32 and SAM Microcontrollers

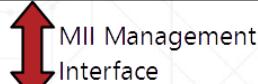
TCP/IP Stack  
Layers

Application,  
Presentation  
& Session

Transport

Internet

Network  
Interface



Ethernet Network  
Interface

External IC



WiFi Network  
Interface

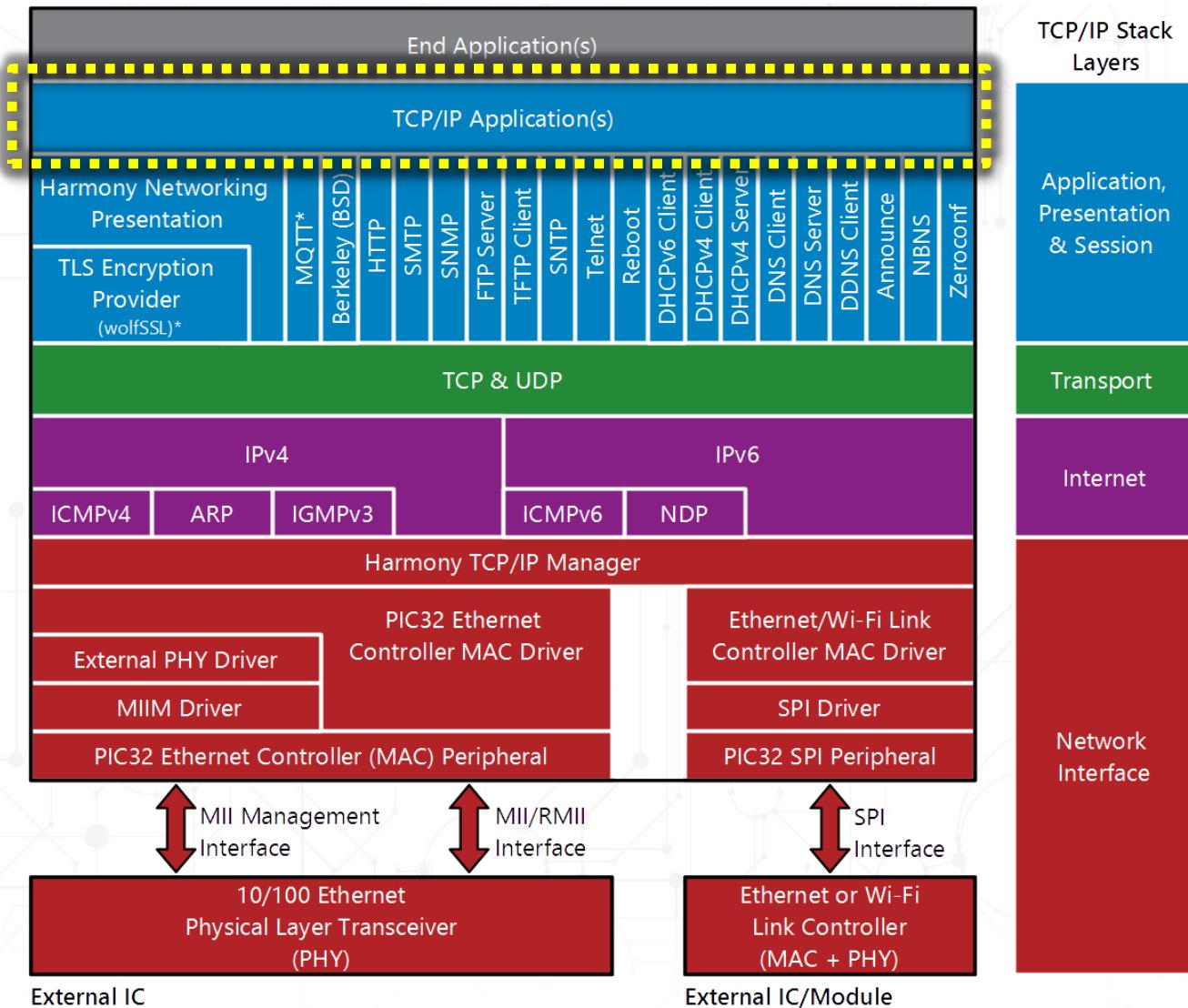
External IC/Module





# Application Layer

PIC32 MCU running MPLAB® Harmony TCP/IP Stack

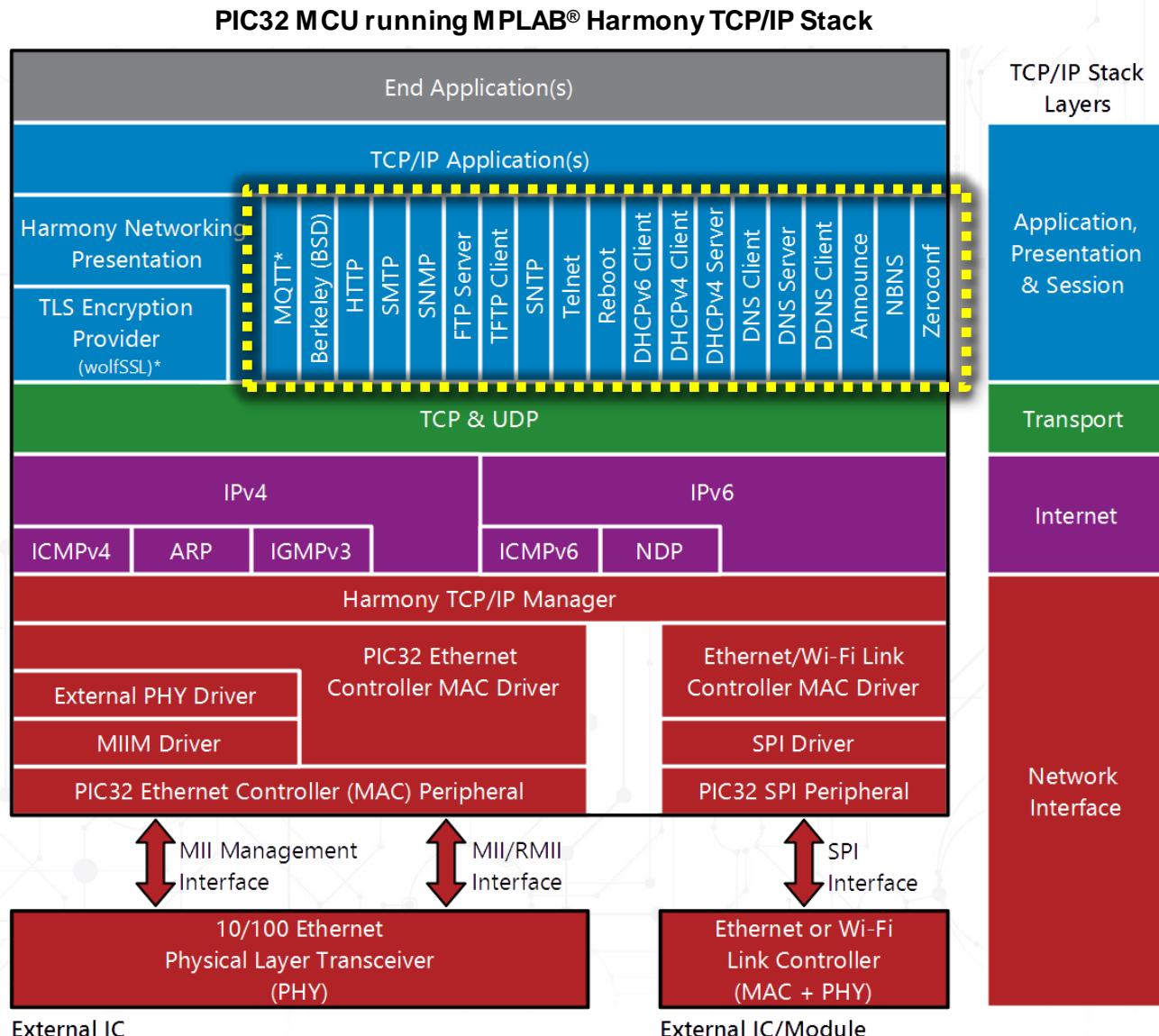




# Application Layer

- Networking Applications can consist of:
  - User implemented TCP/UDP Socket Applications.
  - Prewritten MPLAB® Harmony TCP/IP Application Modules (e.g. HTTP) .
- Multiple Networking Applications can run simultaneously.
- TCP/IP Stack can coincide with other MPLAB® Harmony Middleware (e.g. Graphics).

# Services and Applications





# Supported Services

- **DHCP:** *Dynamic Host Configuration Protocol*
  - Used for IP Address Assignment
  - Server and Client Support
- **DNS:** *Domain Name System*
  - Translates website name to IP Address
  - Server and Client Support
- **DDNS Client**
  - Discovery and registration of public IP address with a DDNS Provider on the internet.

# Supported Services

- **Announce**
  - Microchip TCP/IP Discovery Protocol used for identifying devices on a local network.
- **NBNS: NetBIOS Name Service**
  - Translates local host names to IP Addresses
- **Zeroconf (Bonjour/Avahi)**
  - A set of technologies to automatically create a usable network including IP Address assignment, distribution and resolution of hostnames, location of network services.



# Supported Services

- **MQTT: Message Queue Telemetry Transport**
  - Machine-to-machine (M2M) or "Internet of Things" connectivity protocol
  - Lightweight publish/subscribe messaging transport
  - Supported with third party **wolfMQTT** which is bundled with MPLAB® Harmony.
    - <https://www.wolfssl.com/wolfSSL/Products-wolfmqtt.html>



# Supported Applications

- **HTTP: Hypertext Transfer Protocol (Web Server)**
  - Allows for serving of Webpages
  - Webpages can be stored in Program Memory, SD Card, or External Flash
  - Supports Forms and Dynamic Variables



# HTTP Web Server

Microchip TCP/IP Stack Demo... X

192.168.1.16

**MICROCHIP**

**MPLAB HARMONY**

**TCP/IP Stack Demo Application**

**Welcome!**

**Overview**

**Dynamic Variables**

**Processing Forms**

**Authentication**

**Cookies**

**Uploading Files**

**Sending Emails**

**Dynamic DNS**

**Network Configuration**

**SNMP Configuration**

**Stack Version:** 7.25

**Build Date:** Jun 30 2016 08:43:35

**File System Location:** FLASH

**File System Type:** MPFS2

This site demonstrates the power, flexibility, and scalability of a 32-bit embedded web server. Everything you see is powered by a Microchip PIC microcontroller running the Harmony Microchip TCP/IP Stack.

On the right you'll see the current status of the demo board. For a quick example, click the LEDs to toggle the lights on the board. Press the push buttons (except MCLR!) and you'll see the status update immediately. This examples uses AJAX techniques to provide real-time feedback.

This site is provided as a tutorial for the various features of the HTTP web server, including:

- **Dynamic Variable Substitution** - display real-time data
- **Processing Forms** - handle input from the client
- **Authentication** - require a user name and password
- **Cookies** - store session state information for richer applications
- **Uploading Files** - parse files for configuration settings and more

Several example applications are also provided for updating configuration parameters, sending emails, and controlling the Dynamic DNS client. Thanks to built-in GZIP compression support, all these tutorials and examples fit in the 32kB of on-board Memory.

For more information on the Harmony TCP/IP Stack, please refer to the TCP/IP Stack Libraries Help paragraph in the MPLAB Harmony Help installed on your computer as part of the Harmony distribution.

Copyright © 2014 Microchip Technology, Inc.

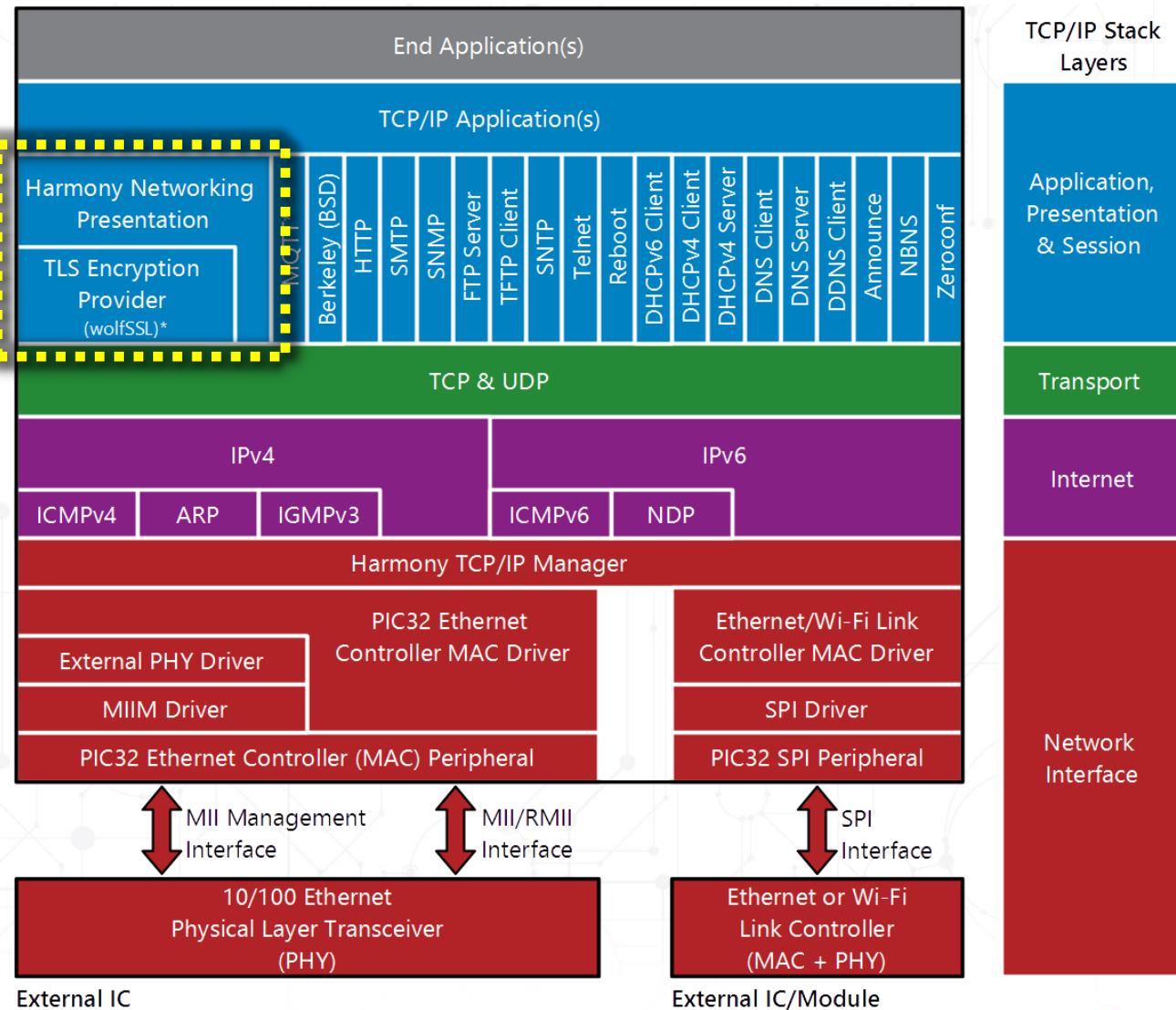
# Supported Applications

- **SMTP** *Simple Mail Transfer Protocol*
  - Allows E-Mails to be sent.
  - Client Support
- **SNMP** *Simple Network Management Protocol*
  - Allows for management of devices on a network.
- **FTP** *File Transfer Protocol*
  - Allows for transfer of files.
- **NTP** *Network Time Protocol*
  - Allows the time of day to be obtained from a precise timer server on the internet.



# Encryption Provider

PIC32 MCU running MPLAB® Harmony TCP/IP Stack



# Encryption Provider

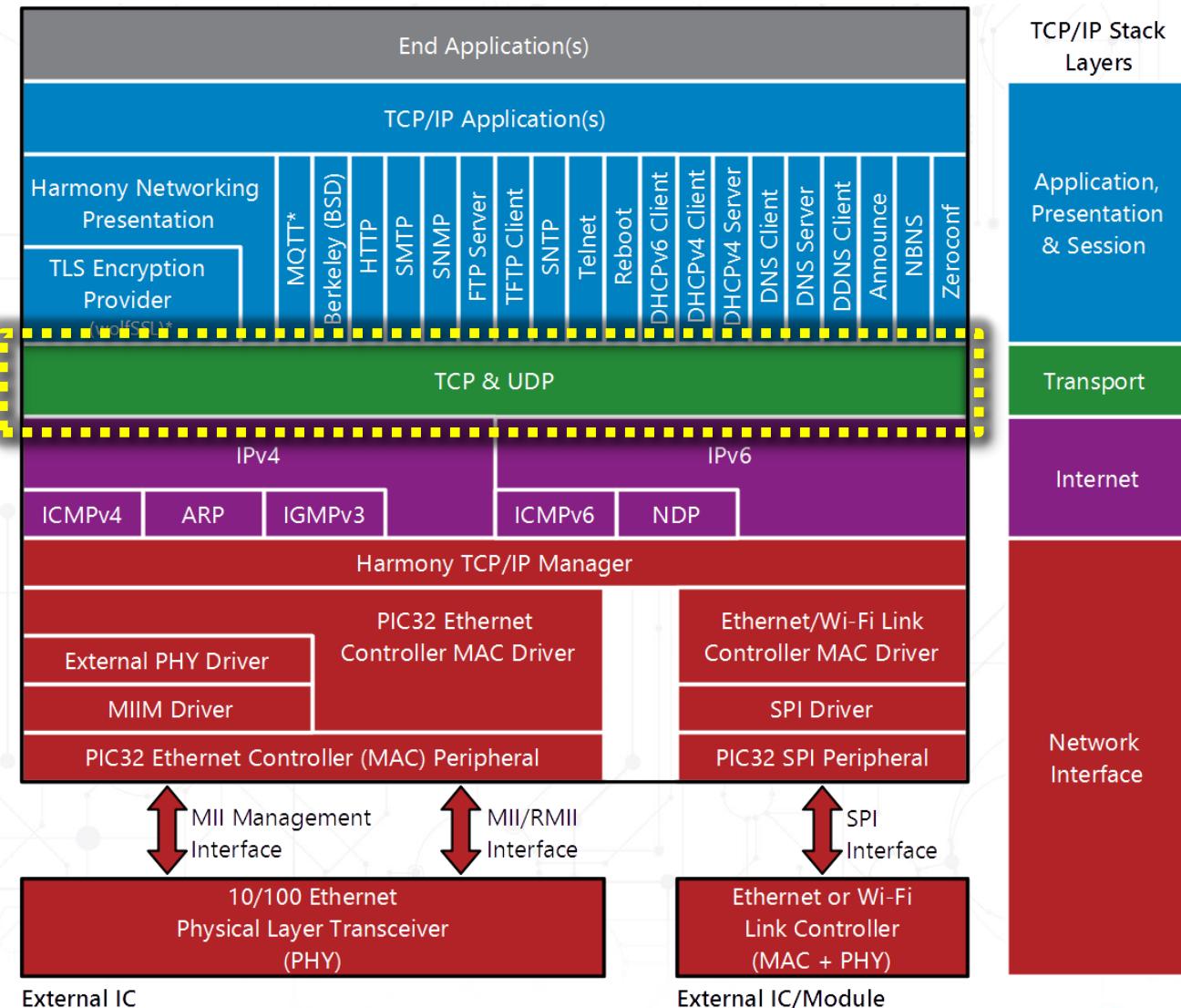
- **TLS from wolfSSL (3rd party provider)**
  - Written in ANSI C
  - Works with or without RTOS
  - TLS 1.2 and DTLS 1.2
  - Integrates directly into MPLAB® Harmony TCP/IP Stack
  - Distributed with MPLAB Harmony
  - Can be evaluated free of charge
  - Requires a commercial license purchase for production



**SSL= Secure Sockets Layer, TLS = Transport Layer**

# Transport Layer

PIC32 MCU running MPLAB® Harmony TCP/IP Stack





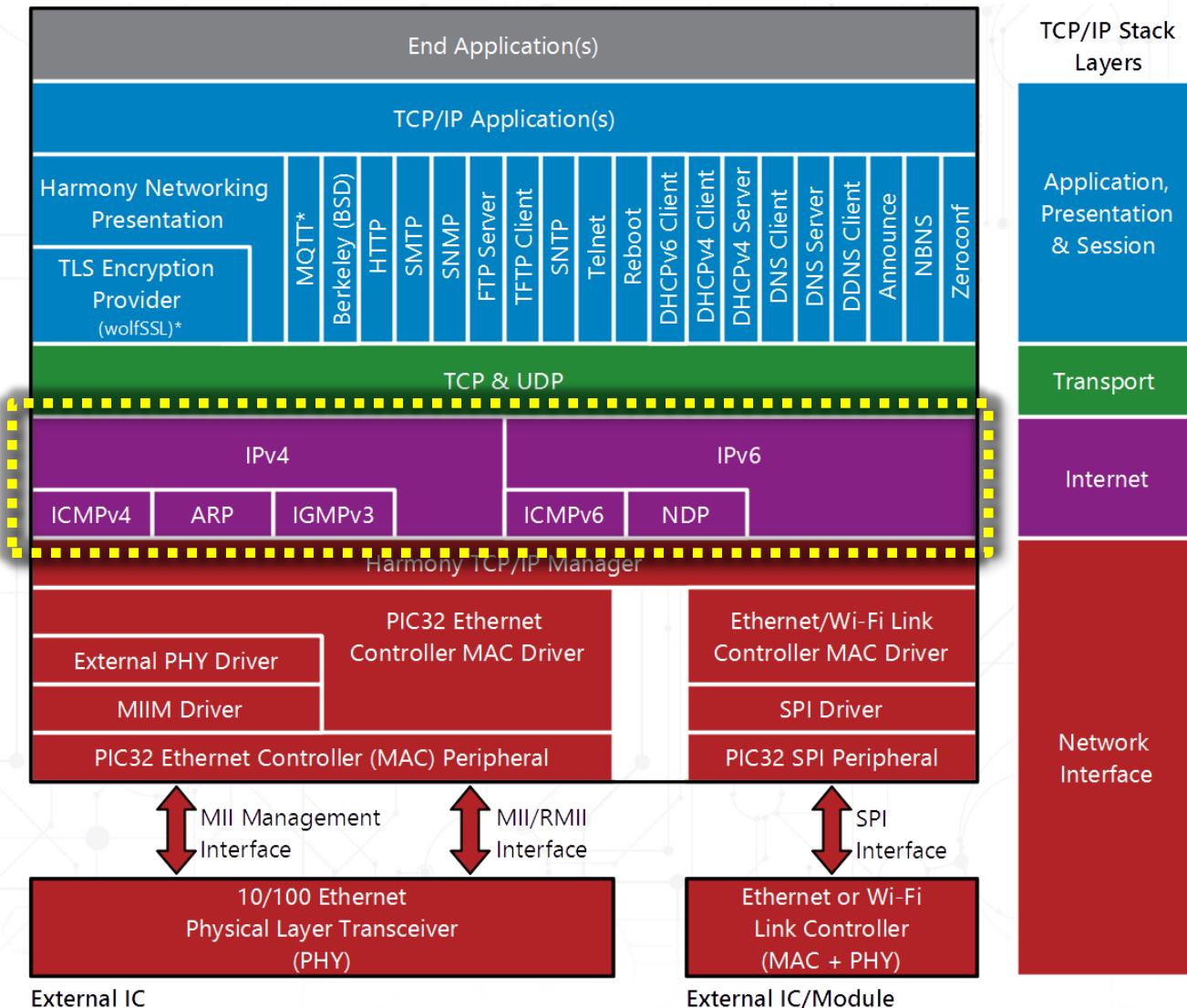
# Transport Layer

- Light-weight and high-performance implementations of the TCP and UDP transport layers.
- Various TCP and UDP configuration options are available at both project generation and run time (eg TX/RX buffer sizes).
- Simultaneous sockets are supported (max number is configurable).
- Simple APIs are used to create socket, check connection status, check TX/RX buffer status, read/write data, and close the socket.



# Internet Layer

PIC32 MCU running MPLAB® Harmony TCP/IP Stack





# Internet Layer

- IPv4 and IPv6 Support
- Both IPv4 and IPv6 can run simultaneously
- The IP Layer is transparent to all applications that use the built in Transport Layer functions.
- IP Datagrams can be sent directly using a special API bypassing higher layer protocols.



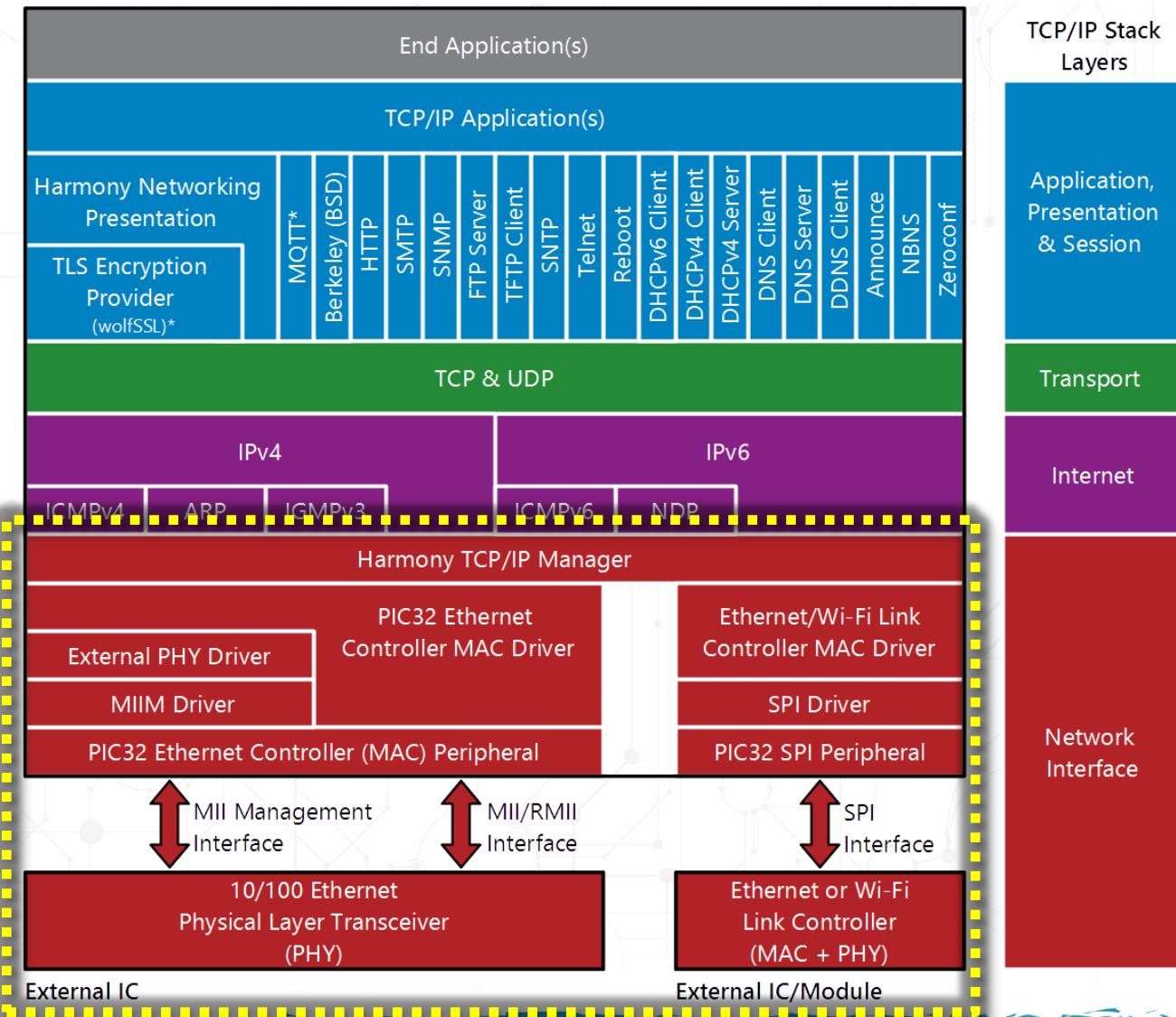
# IPv4 vs IPv6

- **IPv4**
  - Four 8-bit segments
  - Decimal numbers 0 to 255
  - Number of addresses: 32 bits = 4.3 billion
- **IPv6** **FE80:0000:0000:0000:75EA:CEFF:FE44:5101**
  - Eight 16-bit segments
  - Hex numbers 0 to FFFF
  - Number of addresses: 128 bits =  $3.4 \times 10^{38}$   
 $= 340,282,366,920,938,463,463,374,607,431,768,211,456$

192.168.1.1

# Network Interface Layer

PIC32 MCU running MPLAB® Harmony TCP/IP Stack



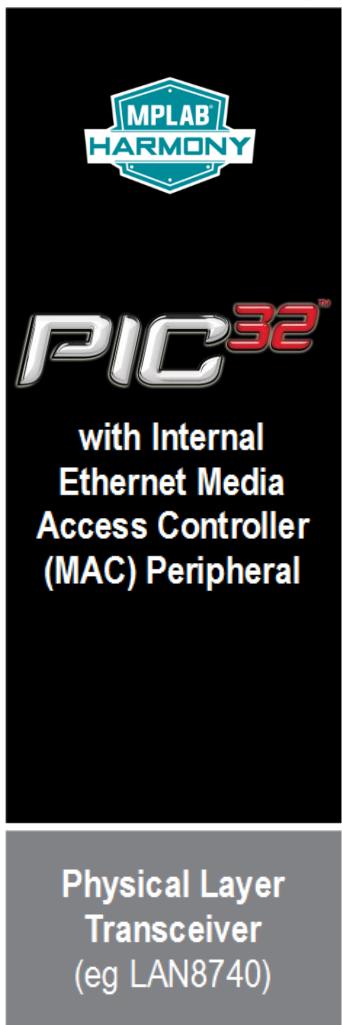
# Network Interface Features

- Multiple Concurrent Interfaces on one MCU e.g.
  - Ethernet + Wi-Fi®\*
  - Ethernet + Ethernet
- Drivers for supported interface devices are supplied in the TCP/IP stack.
- Interface options are selected and configured using MPLAB® Harmony Configuration (MHC) Tool.
- The TCP/IP Stack automatically initialises and manages the configured network interface(s).

\*In Harmony 2.06, the Wi-Fi drivers can only support a single Wi-Fi interface.

# Network Interface Options

Option 1  
Ethernet



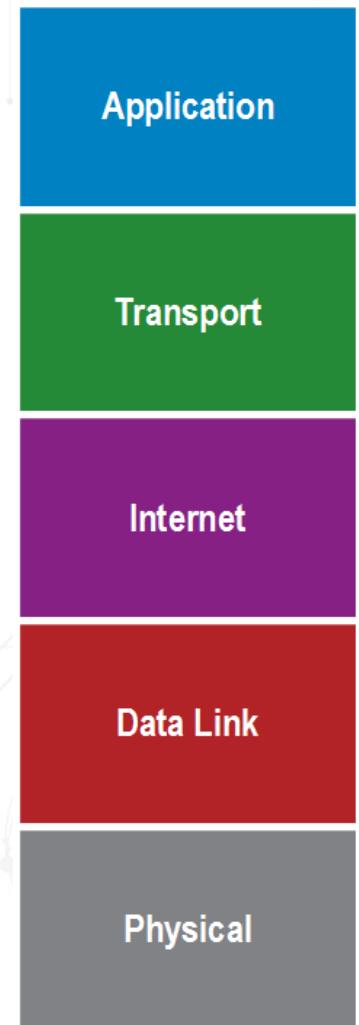
Option 2  
Ethernet



Option 3  
Wi-Fi



TCP/IP  
Stack

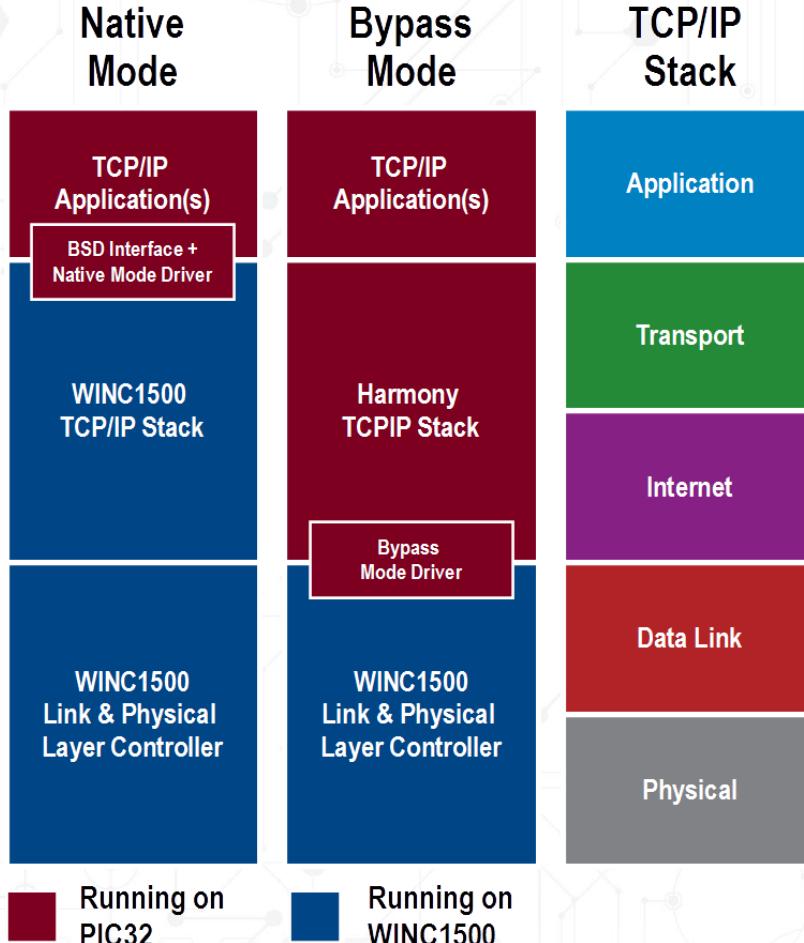


# Supported Interface Devices

- **Ethernet PHYs**
  - Only used with PIC32 devices that have internal Ethernet MAC Peripheral
  - **PHY Only:** LAN8700, LAN8720, LAN8740, KSZ8041 & KSZ8061
  - **PHY + Switch:** LAN9303 & KSZ8863
- **Standalone Ethernet Link Controllers**
  - ENC28J60 (SPI)
  - ENCx24J600 (SPI)
- **Wi-Fi®**
  - ATWINC1500 (SPI)
  - ATWILC1000 (SPI)



# WINC1500 Wi-Fi® Module *Operation Modes with Harmony*



## Two modes of operation:

- **Native Mode:** TCP/IP Stack residing on WINC1500 device is utilised
- **Bypass Mode:** Harmony TCP/IP Stack is utilised

Harmony TCP/IP Stack features are only available when WINC1500 is operated in Bypass mode.





# MAC Address

- All network interface devices must have a globally unique MAC address.
- Some devices are already supplied with a MAC Address, including:
  - PIC32 devices with Internal Ethernet MAC Peripheral
  - All Microchip Wi-Fi® modules
  - Microchip ENCx24J600 Ethernet Controller

# PIC32 MAC Address

- A globally unique MAC Address is programmed into a read only section of Flash Program Memory.
- This feature is only available on PIC32 Devices with an Internal Ethernet MAC Peripheral.
- The address is automatically loaded into the EMAC1Sx registers after reset.
- The address can be manually read using a Harmony API: **PLIB\_ETH\_MACGetAddress**.

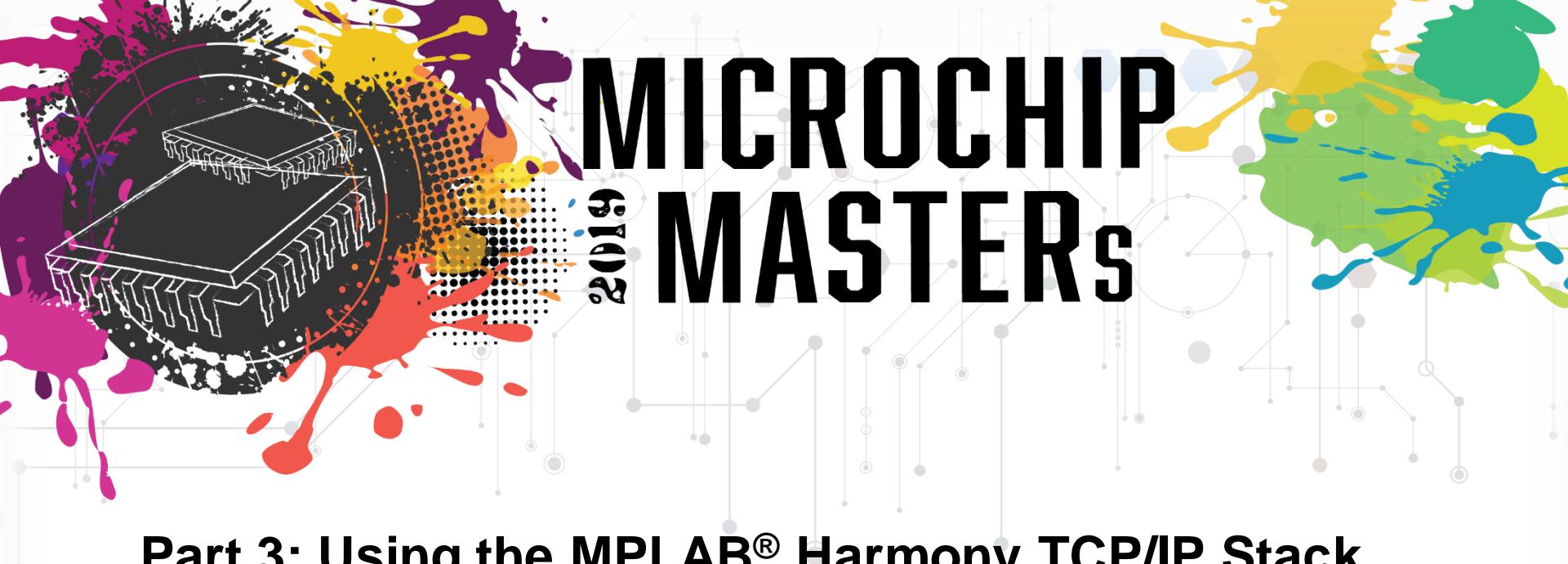
# TCP/IP Stack Features

- **Fully dynamic:**
  - Stack initialization/de-initialization
  - Interface up/down
  - Resource management
  - Module configuration
- **Run-time configuration via APIs or Console**
- **Interrupt driven operation**
- **RTOS compatible**



# TCP/IP Console

- **Network specific commands:**
  - Network info
  - Stack and interfaces up/down control
    - Whole stack or one interface at a time
  - IP addresses set/get
  - DHCP, NetBIOS names, etc.
  - Wi-Fi® commands included
  - TCP/IP Heap statistics
  - Socket Debugging



# MICROCHIP MASTERS

2019

## Part 3: Using the MPLAB® Harmony TCP/IP Stack

- Open an example project
- Stack (Re)Configuration
- Connectivity Check
- MPLAB® Harmony TCP/IP Stack APIs
- Tools





# Harmony V3

## Device Configuration

### Initialize Device

- Clocks
- Pins
- Memory
- Vectors
- Startup Code
- Essentials...

### Peripheral Libraries

#### Use Peripherals

- Custom Initialization
- Clean & direct
- MCC-Like
- Low overhead
- Custom generated

### Drivers & Services

#### Share Resources & Peripherals

- Built on PLIBs
- Interoperable
- Simple interfaces
- Advanced capabilities

### Powerful Middleware

#### Add MW Stacks

- Graphics
- TCP/IP Networking
- USB Host & Device
- Cryptography
- More...

### Real-Time OS

#### Optimize CPU Usage

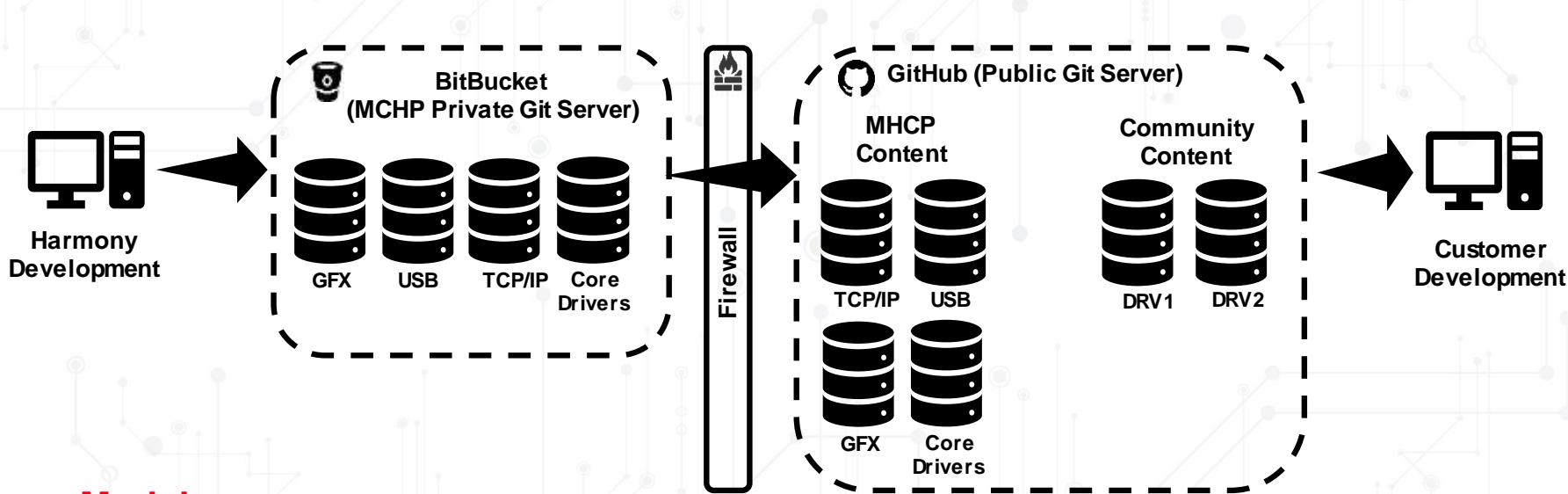
- FreeRTOS Included
  - OS Abstraction Layer (OSAL)
  - Preemptive scheduling

I don't care about libraries. Just get the MCU initialized!

Give me a total system solution!

# Harmony V3

## MPLAB Harmony V3 on GitHub



- **Modular:**
  - Individual libraries released when ready. No need to wait for complete Harmony release
  - Customer selects desired modules. No need to download everything.
- **Flexible:**
  - Ability to quickly add/modify project configuration to meet evolving customer needs
- **Open:**
  - Facilitates community content. Customer can select desired content with package manager.



# MHC Git

Configuration Database Setup

Select and configure the packages that will be included in the current project:

Load	Name	Version	Dependencies
<input checked="" type="checkbox"/>	bsp	v3.3.0	csp(3.2.1)
<input checked="" type="checkbox"/>	core	v3.3.0	csp(3.2.1)
<input checked="" type="checkbox"/>	crypto	v3.2.1	core(v3.2.1)
<input checked="" type="checkbox"/>	csp	v3.2.1	
<input checked="" type="checkbox"/>	net	v3.3.0	core(3.2.1), csp(3.2.1), dev_packs(...)

Configure Device Family and CMSIS Pack Paths:

DFP:

CMSIS:

Configuration Database Setup

Select and configure the packages that will be included in the current project:

Load	Name	Version	Dependencies
<input type="checkbox"/>	audio	v3.1.0	core(3.0)
<input type="checkbox"/>	bootloader	v3.0.0	csp(3.0)
<input checked="" type="checkbox"/>	bsp	v3.2.0	core(3.0)
<input type="checkbox"/>	bt	v3.1.0	core(3.0)
<input checked="" type="checkbox"/>	core	v3.3.0	csp(3.2.1)
<input checked="" type="checkbox"/>	crypto	v3.2.0	core(v3.2.0)
<input checked="" type="checkbox"/>	csp	v3.3.0	
<input type="checkbox"/>	gfx	v3.1.0	core(3.0)
<input type="checkbox"/>	micrium_uicos3	v3.0.0	core(3.0)
<input type="checkbox"/>	motor_control	v3.2.0	csp(v3.2.0)
<input checked="" type="checkbox"/>	net	v3.3.0	core(3.2.1), csp(3.2.1), dev_packs(...)
<input type="checkbox"/>	touch	v3.2.0	csp(v3.3.0)
<input type="checkbox"/>	usb	v3.2.0	core(3.0)

Configure Device Family and CMSIS Pack Paths:

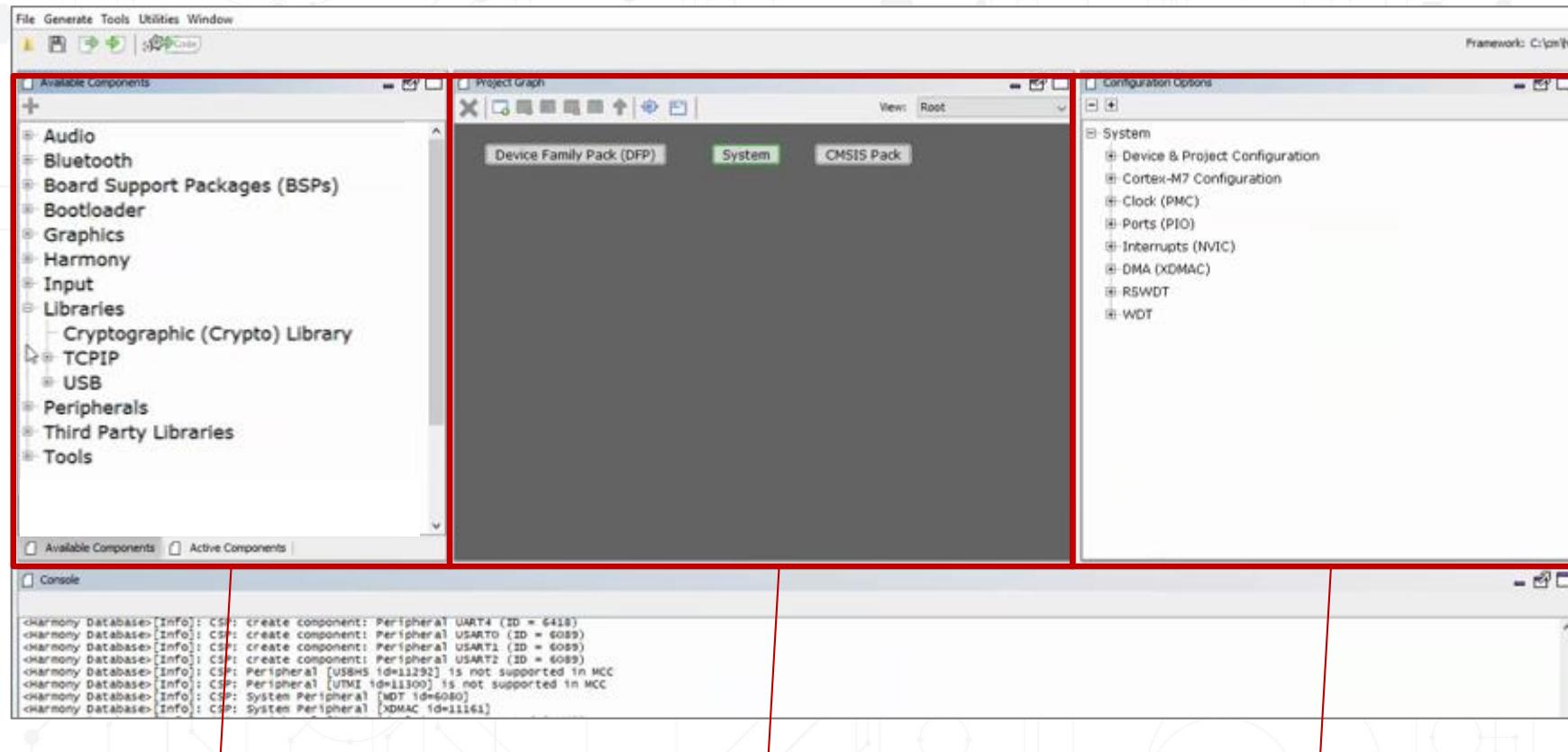
DFP:

CMSIS:



# Harmony V3

## MPLAB Harmony Configurator



**Available Components  
(Peripherals, Libraries,..)**

**Project Graph**

**Configuration Options**



# Harmony V3

## MPLAB Harmony Configurator

The screenshot shows the MPLAB Harmony Configurator software interface. On the left, the "Available Components" pane lists various peripheral components: I2SC, PWM, QSPI, RSTC, RTC, RTT, SDRAMC, SMC, SPI, SSC, SUPC, TC, TC2, TC3, TRNG, TWIHS, and UUART. A red dashed box highlights the SUPC, TC, TC2, TC3, and TRNG components. A red arrow points from this highlighted group to the "Project Graph" pane in the center. The "Project Graph" pane displays a "Device Family Pack (DFP)" containing a "USART0 Peripheral Library" (UART) and a "System" pack containing "TC0 Peripheral Library" (TMR) and "TC1 Peripheral Library" (TMR). A green arrow points from the highlighted components in the Available Components pane to the USART0 component in the Project Graph. The "Configuration Options" pane on the right shows the configuration for USART0, including Interrupt Mode (checked), Select Clock Source (MCK), Clock Source Value (150,000,000), Baud Rate (9,600), OverSampling (16 Times), Data (8 BIT), Parity (NO), and Stop (1BIT). The "Console" pane at the bottom shows the following log output:

```
<Harmony Database>: Info: CSP: create component: Peripheral USART2 (ID = 6089)
<Harmony Database>: Info: CSP: Peripheral [USBHS id=11292] is not supported in MCC
<Harmony Database>: Info: CSP: Peripheral [UTMI id=11300] is not supported in MCC
<Harmony Database>: Info: CSP: System Peripheral [WDT id=6080]
<Harmony Database>: Info: CSP: System Peripheral [XDHAC id=11161]
<Harmony Database>: Info: CSP: Peripheral [EUSIS id=1] is not supported in MCC
```



# Harmony V3

Peripheral  
initialization only. No  
Driver!  
Access completely  
managed from end-

The screenshot shows the MPLAB Harmony Configurator interface. On the left is a sidebar with a tree view of available components:

- Audio
- Bluetooth
- Board Support Packages (BSPs)
- Bootloader
- Graphics
- Harmony
- Input
- Libraries
  - Cryptographic (Crypto) Library
  - TCPIP
  - USB
- Peripherals
- Third Party Libraries
- Tools

The main area is titled "Project Graph" and shows a "Device Family Pack (DFP)" node expanded. Inside it, the "USART0 Peripheral Library" is selected, and its "UART" component is highlighted with a yellow diamond marker. Below the graph are tabs for "System" and "CMSIS Pack".

A terminal window at the bottom displays log messages related to component creation and configuration:

```
<Harmony Database>[Info]: create component: USB Device HID
<Harmony Database>[Info]: create component: USB Device MSD
<Harmony Database>[Info]: create component: USB Host
<Harmony Database>[Info]: create component: USB Host MSD
<Harmony Database>[Info]: create component: USB Host CDC
<Harmony Database>[Info]: create component: USB Host HID
<Harmony Database>[Info]: create component: USB Host Audio
<Info>: Startup tasks completed
<Configuration Database>[Info]: Loading Interrupt Manager for ATSAME7Q21B
<Configuration Database>[Info]: Loading SYSTICK for ATSAME7Q21B
<Configuration Database>[Info]: Loading DMA Manager for ATSAME7Q21B
<Configuration Database>[Info]: Loading RSDT for ATSAME7Q21B
<Configuration Database>[Info]: Starting RTT for ATSAME7Q21B
<Configuration Database>[Info]: Running USART0
<Plugin>[Info]: Loading plugin: C:\cm\h3\csp\archv..\peripheral\clksame_e70\plugin\clockmanager.jar
<Plugin>[Info]: Loading plugin: C:\cm\h3\csp\archv..\peripheral\mpu\plugin\MPUmanager.jar
<Plugin>[Info]: Loading plugin: C:\cm\h3\csp\archv..\peripheral\xdmac_11161\plugin\xdmananager.jar
<Plugin>[Info]: Loading plugin: C:\cm\h3\csp\archv..\peripheral\nvic\plugin\NVICmanager.jar
<Plugin>[Info]: Loading plugin: C:\cm\h3\csp\archv..\peripheral\pio_11004\plugin\SAME70pinmanager.jar
<Harmony Database>[Info]: Loading System Services for ATSAME7Q21B
<Harmony Database>[Info]: Loading Pin Manager for ATSAME7Q21B
```

Welcome to the MPLAB Harmony Configurator!

MPLAB Harmony Configurator



# Harmony V3

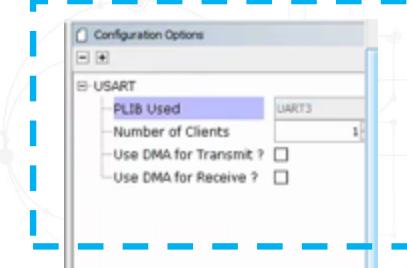
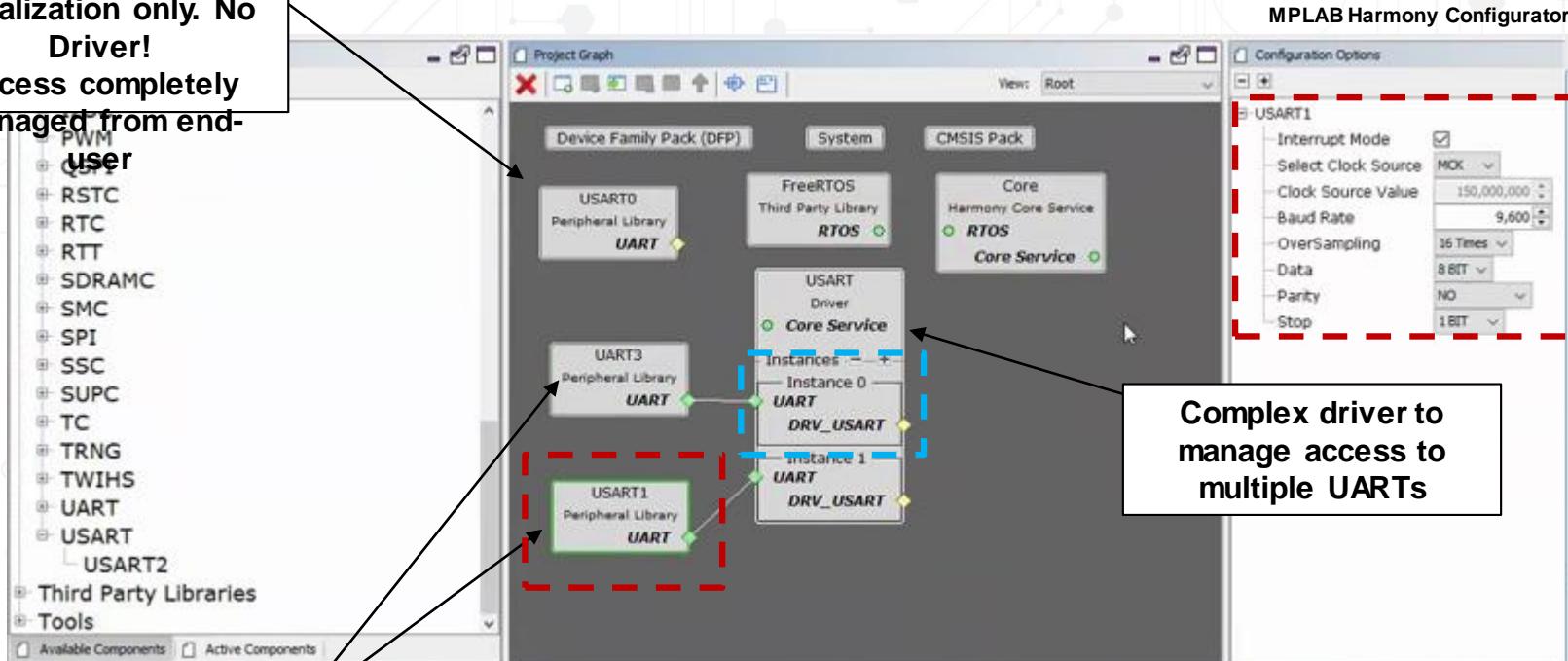
Peripheral initialization only. No Driver!  
Access completely managed from end-user

PWM  
QSPI  
RSTC  
RTC  
RTT  
SDRAMC  
SMC  
SPI  
SSC  
SUPC  
TC  
TRNG  
TWIHS  
UART  
USART  
USART2

Third Party Libraries  
Tools

Available Components Active Components

Peripheral initialization only.





# Harmony V3

The screenshot shows the MPLAB Harmony IDE interface. The left sidebar displays a file tree for a project named 'est'. The main workspace shows a code editor with C code for USART0 initialization. The code includes comments explaining the configuration of USART0 mode, baud rate, and instance objects. A cursor is visible over the line of code setting the USART0 baud rate.

```
166     (void)dummyData;
167
168     return;
169 }
170
171 void USART0_Initialize( void )
172 {
173     /* Reset USART0 */
174     USART0_REGS->US_CR = (US_CR_RSTRX_Msk | US_CR_RSTTX_Msk | US_CR_RSTSTA_Msk);
175
176     /* Enable USART0 */
177     USART0_REGS->US_CR = (US_CR_TXEN_Msk | US_CR_RXEN_Msk);
178
179     /* Configure USART0 mode */
180     USART0_REGS->US_MR = (US_MR_USCLKS_MCK | US_MR_CHRL_8_BIT | US_MR_USART_PAR_NO | US_MR_USART_NBSTOP_1_BIT | (0 << US_MR_USART_OVER_Pos));
181
182     /* Configure USART0 Baud Rate */
183     USART0_REGS->US_BRGR = US_BRGR_CD(976);
184
185     /* Initialize instance object */
186     usart0Obj.rxBuffer = NULL;
187     usart0Obj.rxSize = 0;
188     usart0Obj.rxProcessedSize = 0;
189     usart0Obj.rxBusyStatus = false;
```

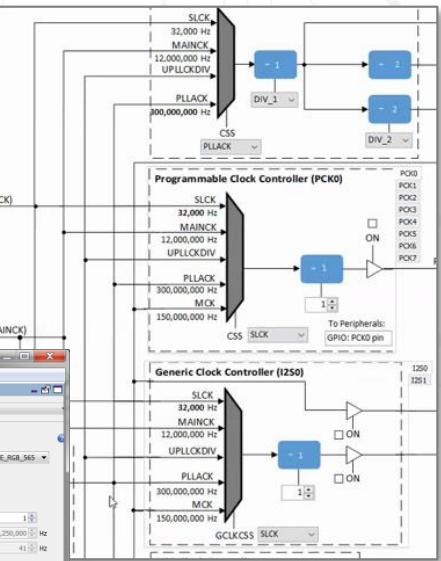
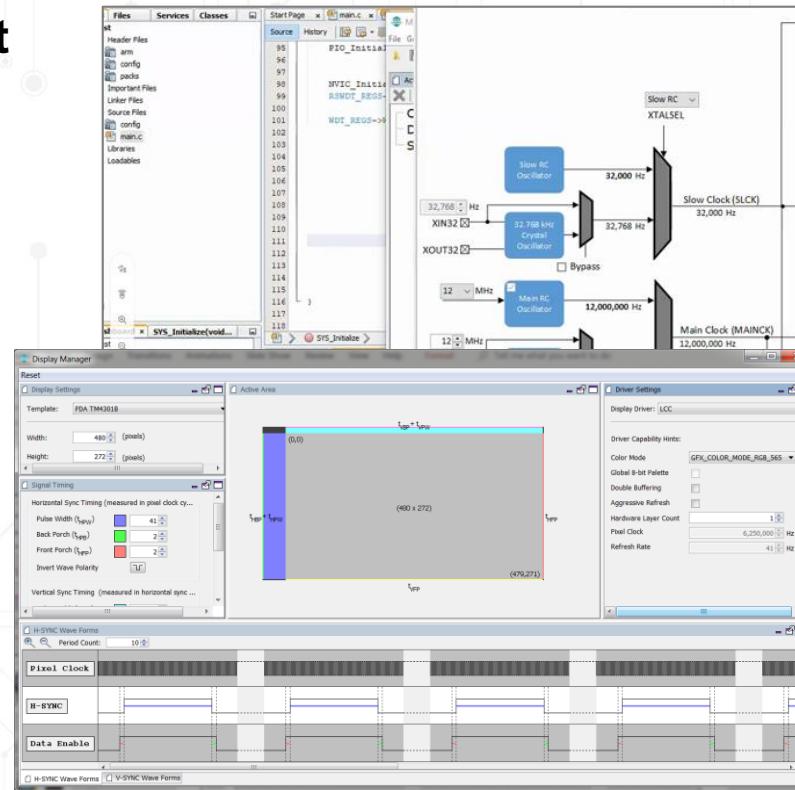
The status bar at the bottom indicates the current function is 'USART0\_Initialize' and the address is '0x00'.

# Harmony V3

Keep + improve the good things from Harmony V2



- Clock Initialization
- Pinout/I/O Management
  - Graphic
- TCP/IP and SSL1.3
  - Security
  - USB
  - Audio
  - ...





# MHC

MPLAB Harmony Configurator - sam\_v71\_xult\_freertos

File Generate Tools Utilities Window

Framework: C:\microchip\h3\

Project Graph

View: Root

Active Components

- CMSIS Pack
- CONSOLE
- Core
- Cryptographic (Crypto) Library
- Device Family Pack (DFP)
- EFC
- FILE SYSTEM
- FreeRTOS
- MEMORY
- Presentation Layer
- SAM V71 Xplained Ultra BSP
- System
- TCO
- TCP/IP STACK
  - APPLICATION LAYER
  - BASIC CONFIGURATION
    - NETCONFIG
      - Instance 0
        - TCP/IP Basic Configurator
        - TCPPIP CMD
        - TCPPIP CORE
        - TCPPIP File System Wrapper
    - DRIVER LAYER
    - NETWORK LAYER
    - TRANSPORT LAYER
- TIME
- USART1

Project Graph

Device Family Pack (DFP) System CMSIS Pack

Core Harmony Core Service RTOS Core Service

TC0 Peripheral Library TMR

TIME System Service Core Service SYS\_TIME Core Service

Cryptographic (Crypto) Library SYS\_TIME LIB\_CRYPTO

FreeRTOS Third Party Library RTOS

UART1 Peripheral Library USART

MEMORY Driver Core Service

EFC Peripheral Library MEMORY

PRESENTATION LAYER net\_pres

Instances Instance 0 LIB\_CRYPTO

CONSOLE System Service Core Service

UART SYS\_CONSOLE

FILE SYSTEM System Service Core Service

DRV\_MEDIA SYS\_FS

TIME

USART1

Configuration Options

NETCONFIG

Network Configurations Index	0
Interface	GMAC
Host Name	MARTIN_RUPPERT
Mac Address	00:04:25:1C:A0:02
IPv4 Static Address	192.168.100.11
IPv4 SubNet Mask	255.255.255.0
IPv4 Default Gateway Address	192.168.100.1
IPv4 Primary DNS	192.168.100.1
IPv4 Secondary DNS	0.0.0.0
Power Mode	full

Network Configuration Start-up Flags

Network MAC Driver DRV\_GMAC\_Object

Available Components Active Components

Welcome to the MPLAB Harmony Configurator!



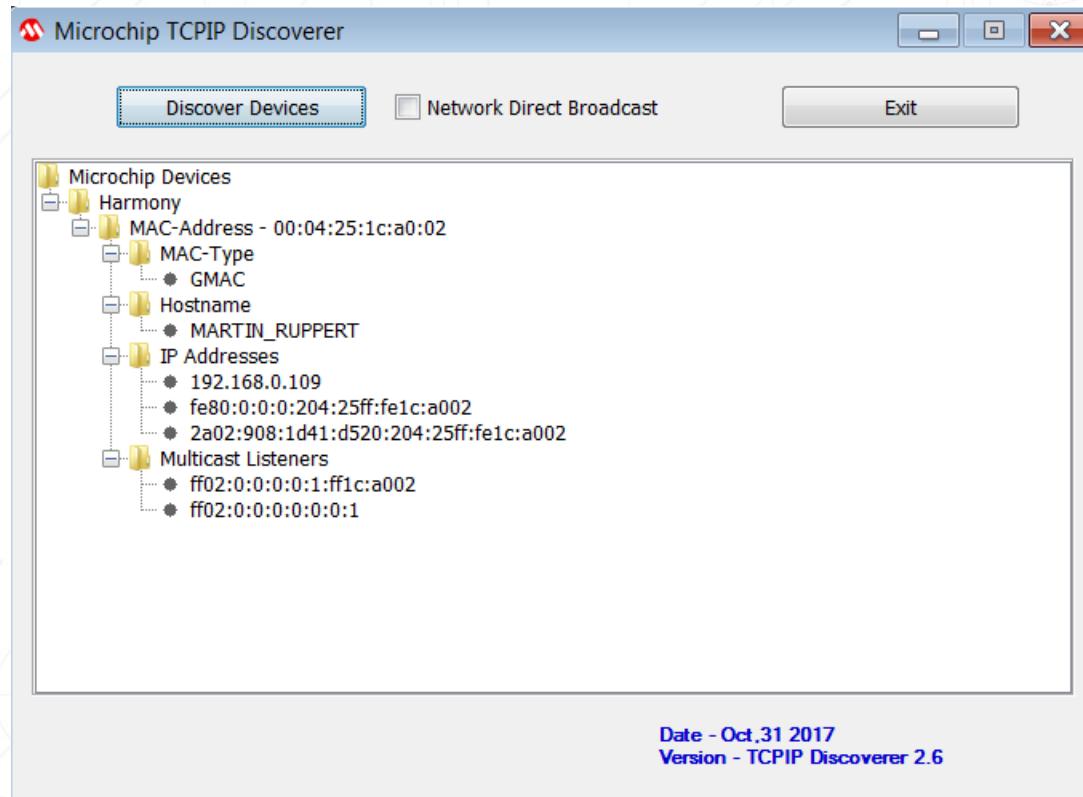
# Terminal

```
COM55:115200baud - Tera Term VT
File Edit Setup Control Window Help
TCP/IP Stack: Initialization Started

>TCP/IP Stack: Initialization Ended - success
SYS_Initialize: The MPFS2 File System is mounted
    Interface GMAC on host MARTIN_RUPPERT - NBNS enabled
GMAC IP Address: 0.0.0.0
GMAC IP Address: 192.168.0.109
```



# TCPIP Discoverer



Date - Oct.31 2017

Version - TCPIP Discoverer 2.6



# HTTP2 Webserver

192.168.0.109

**Welcome!**

**Stack Version:** 7.32 - H3.2    **Build Date:** Mar 15 2019  
**File System Location:** FLASH    **File System Type:** MPFS2

**LED:**    
**Buttons:** V V A  
**Random Number:** 22064

This site demonstrates the power, flexibility, and scalability of a 32-bit embedded web server. Everything you see is powered by a Microchip PIC microcontroller running the Harmony Microchip TCP/IP Stack.

On the right you'll see the current status of the demo board. For a quick example, click the LEDs to toggle the lights on the board. Press the push buttons (except MCLR!) and you'll see the status update immediately. This examples uses AJAX techniques to provide real-time feedback.

This site is provided as a tutorial for the various features of the HTTP web server, including:

- **Dynamic Variable Substitution** - display real-time data
- **Processing Forms** - handle input from the client
- **Authentication** - require a user name and password
- **Cookies** - store session state information for richer applications
- **Uploading Files** - parse files for configuration settings and more

Several example applications are also provided for updating configuration parameters, sending emails, and controlling the Dynamic DNS client. Thanks to built-in GZIP compression support, all these tutorials and examples fit in the 32kB of on-board Memory.

For more information on the Harmony TCP/IP Stack, please refer to the TCP/IP Stack Libraries Help paragraph in the MPLAB Harmony Help installed on your computer as part of the Harmony distribution.



# Using RTOS with MPLAB Harmony



# Why use RTOS?

## □ Priority based task scheduling

- Ensures better responsiveness to events
- A pre-emptive scheduler always runs the highest priority task that is ready to run, allowing hard real time tasks to meet its deadlines
- Removes dependency of tasks on the execution time of other functions running in the “super loop”

## □ Efficient CPU usage

- Does not waste CPU cycles by running tasks waiting on events/ resources
- Idle task is run when there are no application tasks ready to run. This allows putting CPU in low power modes

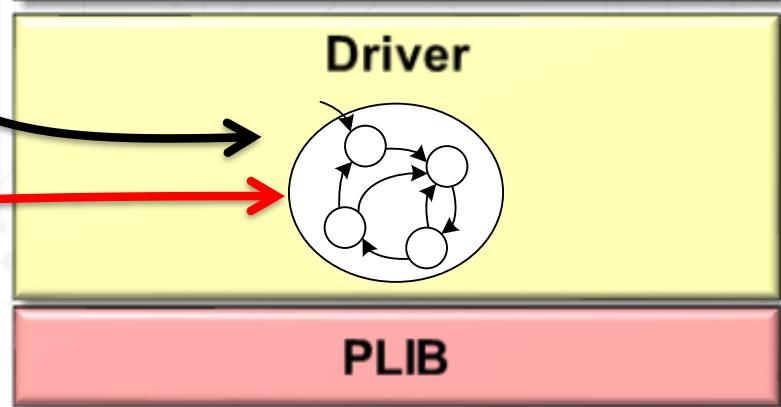
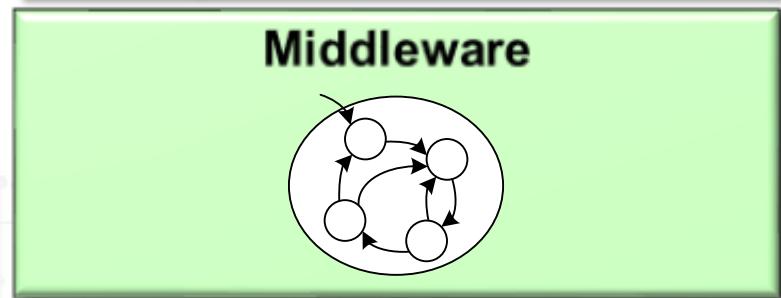
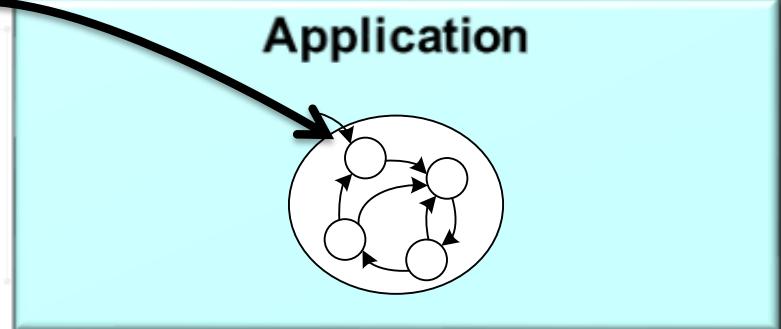


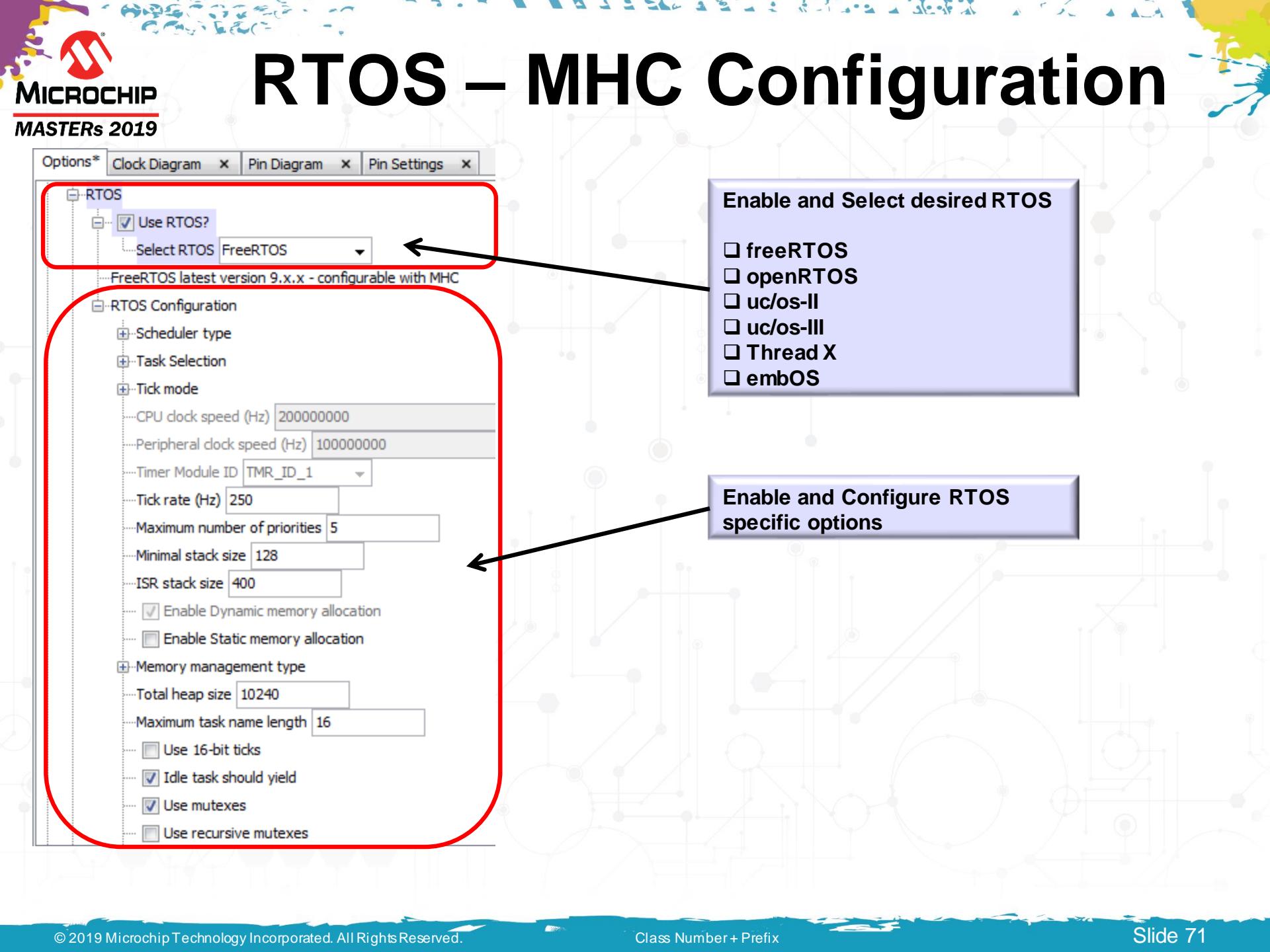
# RTOS Driven

```
static void _SENSORTASK_Tasks(void)
{
    while(1)
    {
        SENSORTASK_Tasks();
        vTaskDelay(10); /*Give time back*/
    }
}
```

```
void _SYS_Tasks (void *param)
{
    while(1)
    {
        DRV_SPI_Tasks(sysObj.spiObjectIdx0);
        vTaskDelay(10); /*Give time back*/
    }
}
```

```
void SPI_ISR(void)
{
    DRV_SPI_Tasks(sysObj.spiObjectIdx0);
}
```





# RTOS – MHC Configuration

Options \* Clock Diagram Pin Diagram Pin Settings

**RTOS**

Use RTOS?  
Select RTOS: FreeRTOS

FreeRTOS latest version 9.x.x - configurable with MHC

**RTOS Configuration**

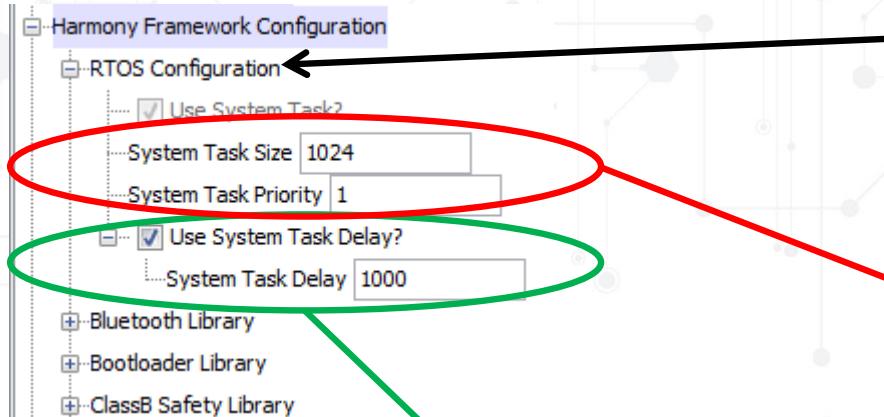
- Scheduler type**
- Task Selection**
- Tick mode**
- CPU clock speed (Hz)**: 200000000
- Peripheral clock speed (Hz)**: 100000000
- Timer Module ID**: TMR\_ID\_1
- Tick rate (Hz)**: 250
- Maximum number of priorities**: 5
- Minimal stack size**: 128
- ISR stack size**: 400
- Enable Dynamic memory allocation
- Enable Static memory allocation
- Memory management type**
- Total heap size**: 10240
- Maximum task name length**: 16
- Use 16-bit ticks
- Idle task should yield
- Use mutexes
- Use recursive mutexes

Enable and Select desired RTOS

- freeRTOS
- openRTOS
- uc/os-II
- uc/os-III
- Thread X
- embOS

Enable and Configure RTOS specific options

# RTOS – MHC Configuration



RTOS configuration for the **SYS\_Tasks** thread.

```
void SYS_Tasks ( void )
{
    /* Create OS Thread for Sys Tasks. */
    xTaskCreate((TaskFunction_t) _SYS_Tasks,
                "Sys Tasks",
                1024, NULL, 1, NULL);
    ....
}
```

```
static void _SYS_Tasks ( void)
{
    while(1)
    {
        /* Maintain system services */
        SYS_CONSOLE_Tasks(sysObj.sysConsole0);

        /* Maintain Device Drivers */

        /* Maintain Middleware */

        /* Task Delay */
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Block the task for 1000 milliseconds. Allows other tasks to run.



# RTOS – MHC Configuration

RTOS configuration for a driver instance which is configured for polled mode operation

Standalone – Run the driver task as a separate RTOS thread

```
void SYS_Tasks ( void )
{
    /* Create OS Thread for DRV_SPI Instance 0 Tasks. */

    xTaskCreate((TaskFunction_t) _DRV_SPI_IDX0_Tasks,
                "DRV_SPI Instance 0 Tasks",
                1024, NULL, 1, NULL);
    ...
}
```

```
void _DRV_SPI_IDX0_Tasks(void)
{
    while(1)
    {
        DRV_SPI_Tasks(sysObj.spiObjectIdx0);

        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Combined with System Tasks – Run the driver task as part of the `_SYS_Tasks` thread

```
static void _SYS_Tasks ( void )
{
    while(1)
    {
        /* Maintain system services */
        SYS_CONSOLE_Tasks(sysObj.sysConsole0);

        /* Maintain Device Drivers */
        DRV_SPI_Tasks(sysObj.spiObjectIdx0);

        /* Maintain Middleware */

        /* Task Delay */
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

© 2019 Microchip Technology Incorporated. All Rights Reserved.

Class Number + Prefix

Slide 73

# System Initialization & Tasks

```

int main(void)
{
    SYS_Initialize(NULL);

    while(true)
    {
        SYS_Tasks();
    }

    return(EXIT_VALUE);
}

void SYS_Tasks ( void )
{
    /* Create OS Thread for Sys Tasks. */
    xTaskCreate((TaskFunction_t) _SYS_Tasks, "Sys Tasks",1024, NULL, 3, NULL);
    xTaskCreate((TaskFunction_t) _USB_Tasks, "USB Tasks",1024, NULL, 3, NULL);

    /* Create OS Thread for SENORTASK Tasks. */
    xTaskCreate((TaskFunction_t) _SENSORTASK_Tasks, "SENSORTASK Tasks", 1024, NULL, 2,
NULL);

    /* Create OS Thread for EEPROMTASK Tasks. */
    xTaskCreate((TaskFunction_t) _EEPROM_Tasks, "EEPROM Tasks", 1024, NULL, 1, NULL);
    .....

    /*****
     * Start RTOS *
     *****/
    vTaskStartScheduler(); /* This function never returns. */
}

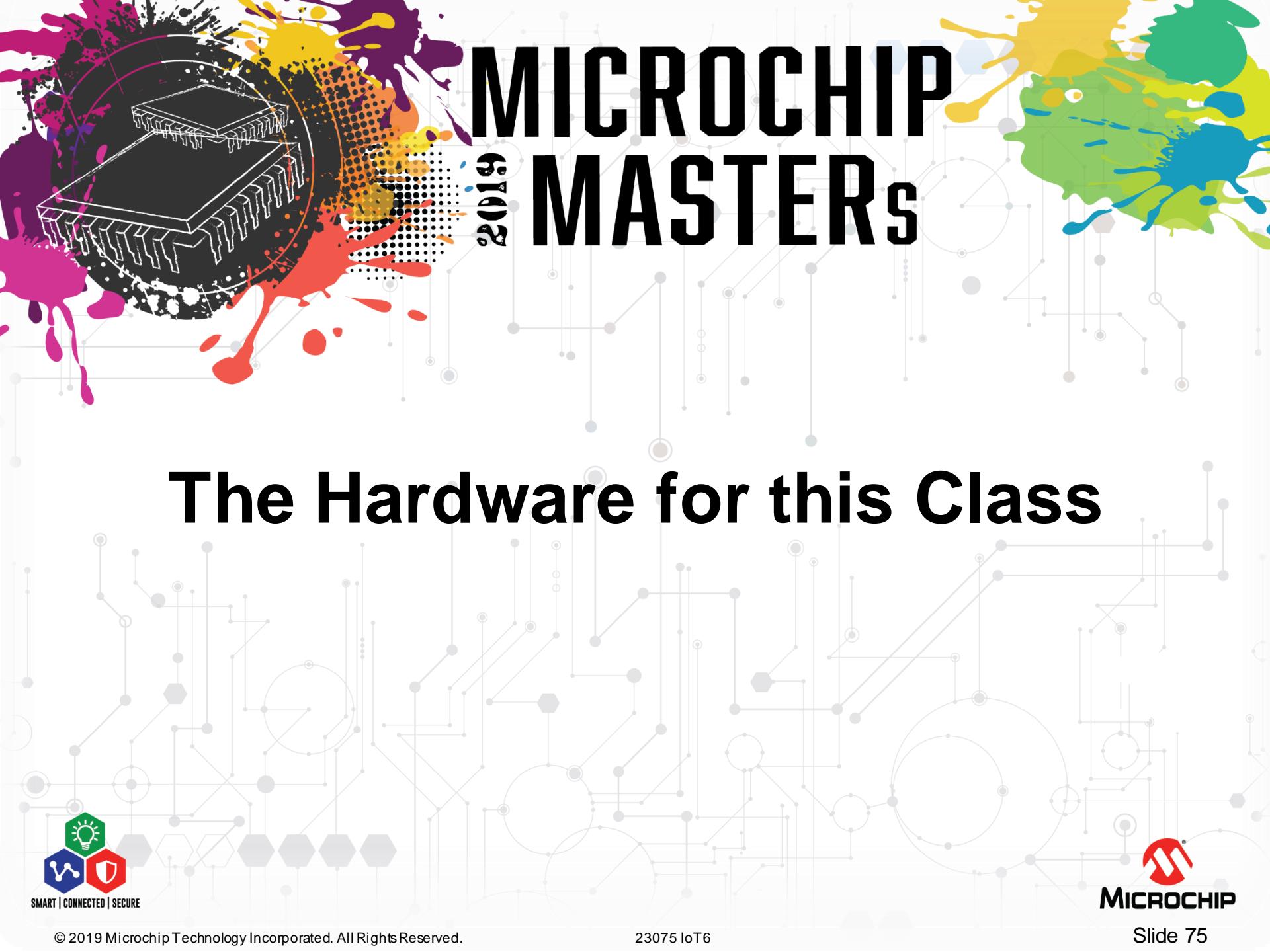
```

```

void SYS_Initialize( void* data )
{
    SYS_CLK_Initialize( &clkInit );
    BSP_Initialize();
    sysObj.spiObjectIdx0 = DRV_SPI_Initialize (DRV_SPI_INDEX_0,
&drvSpi0InitData);
    sysObj.drvTmr0      = DRV_TMR_Initialize(DRV_TMR_INDEX_0,
&drvTmr0InitData);
    .....

    /*create Tasks*/
    APP_Initialize();
}

```



# MICROCHIP MASTERS

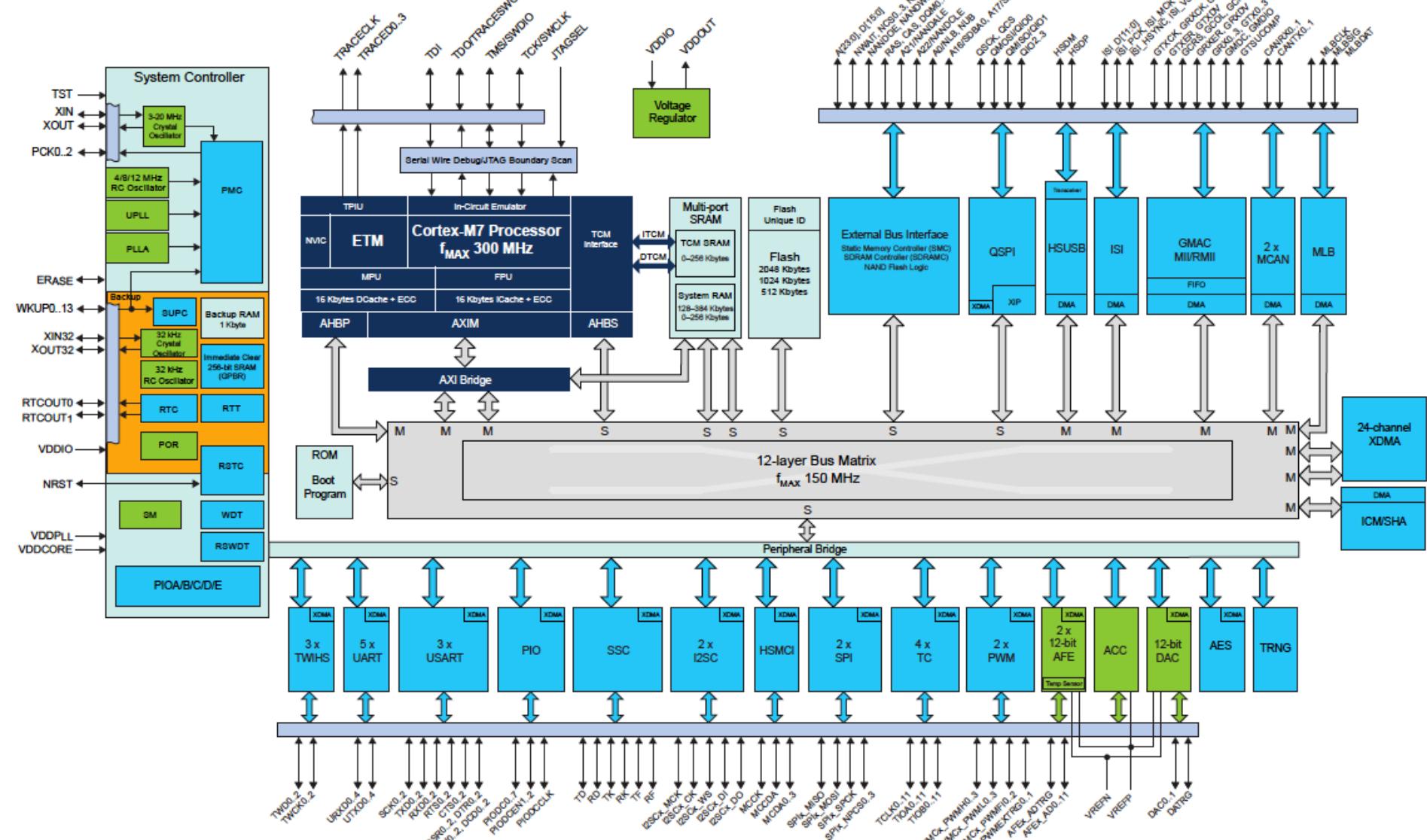
2019

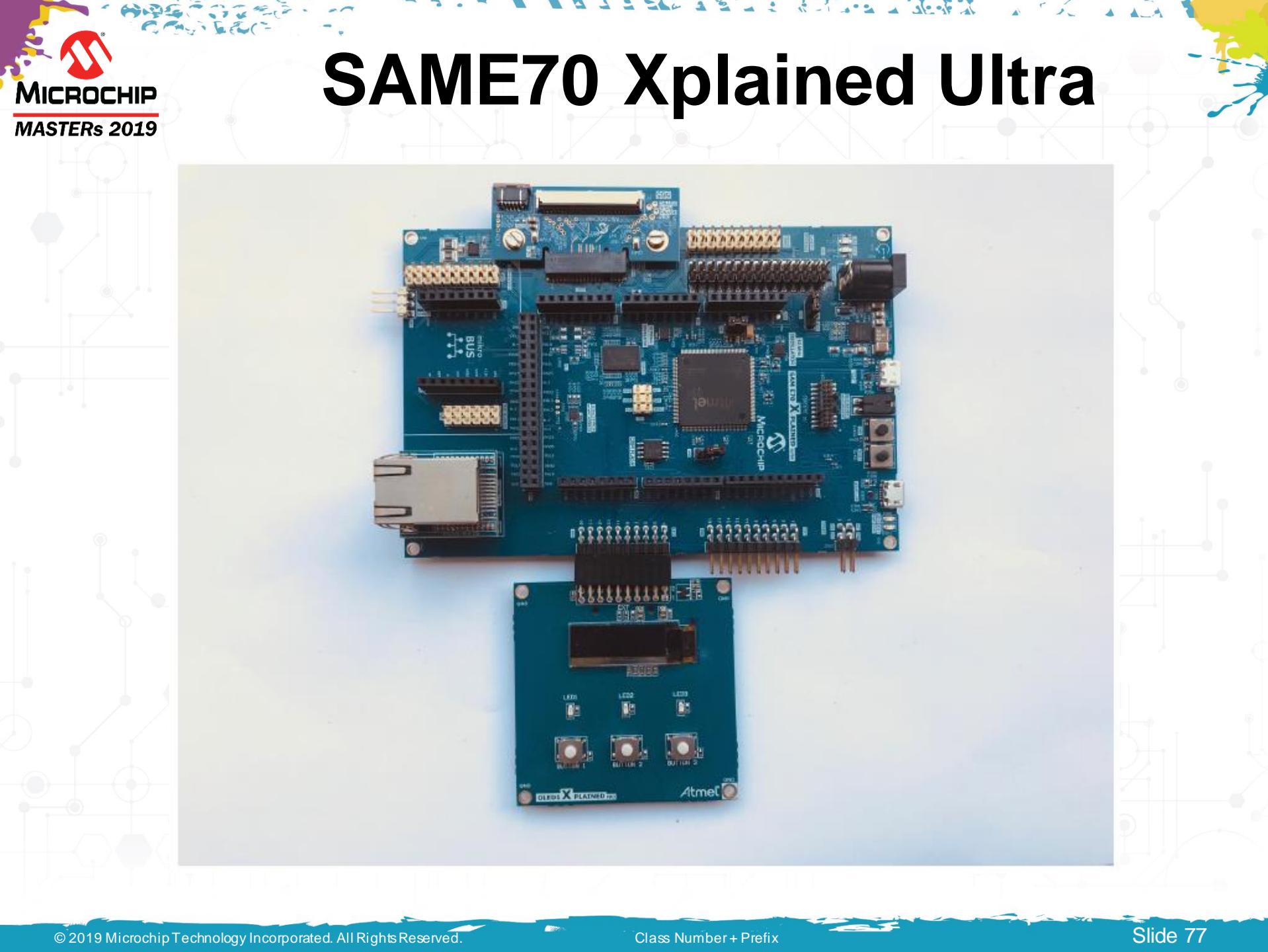
## The Hardware for this Class



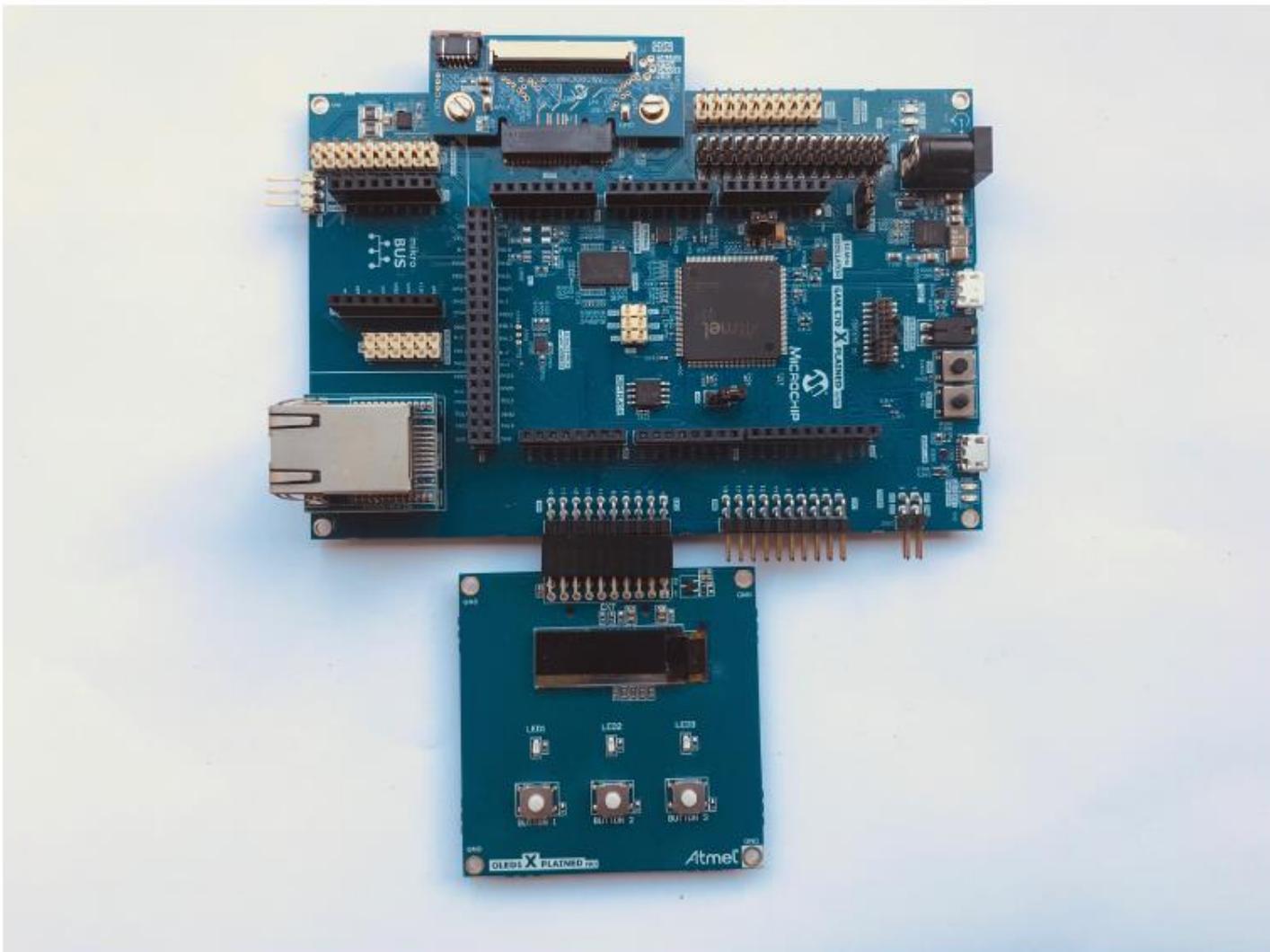


# SAME70

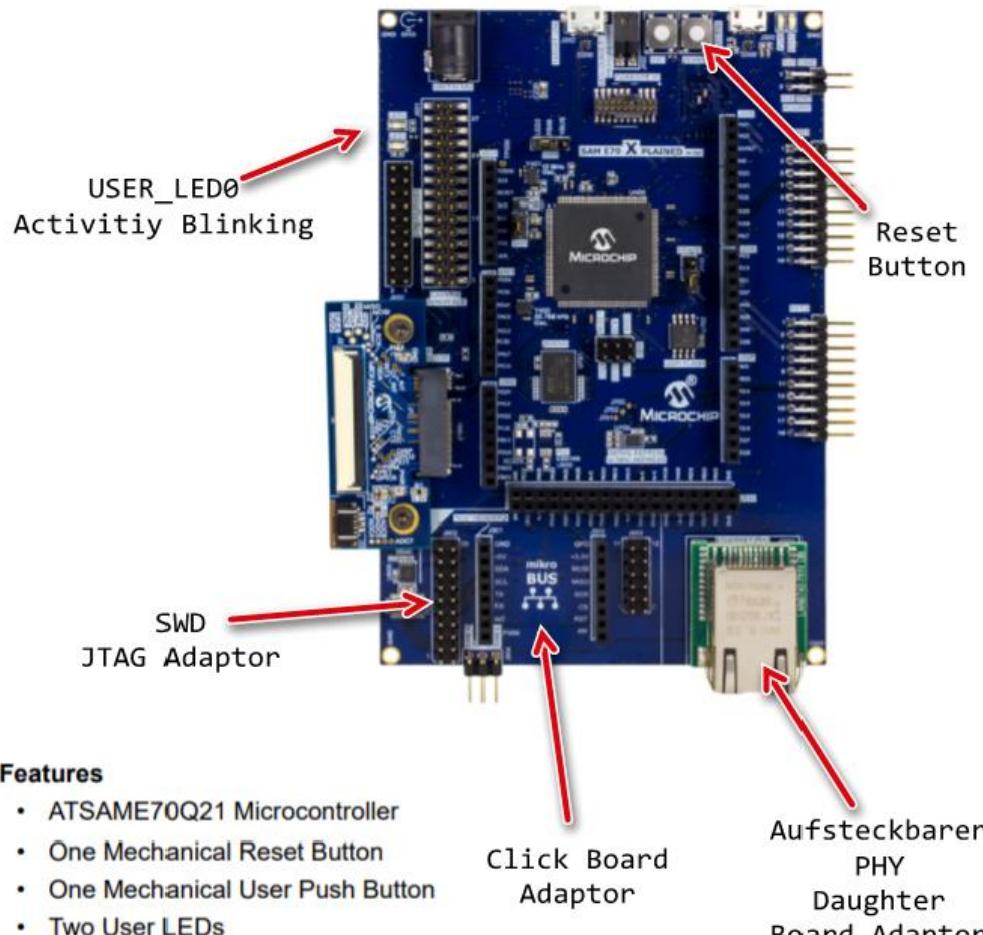




# SAME70 Xplained Ultra



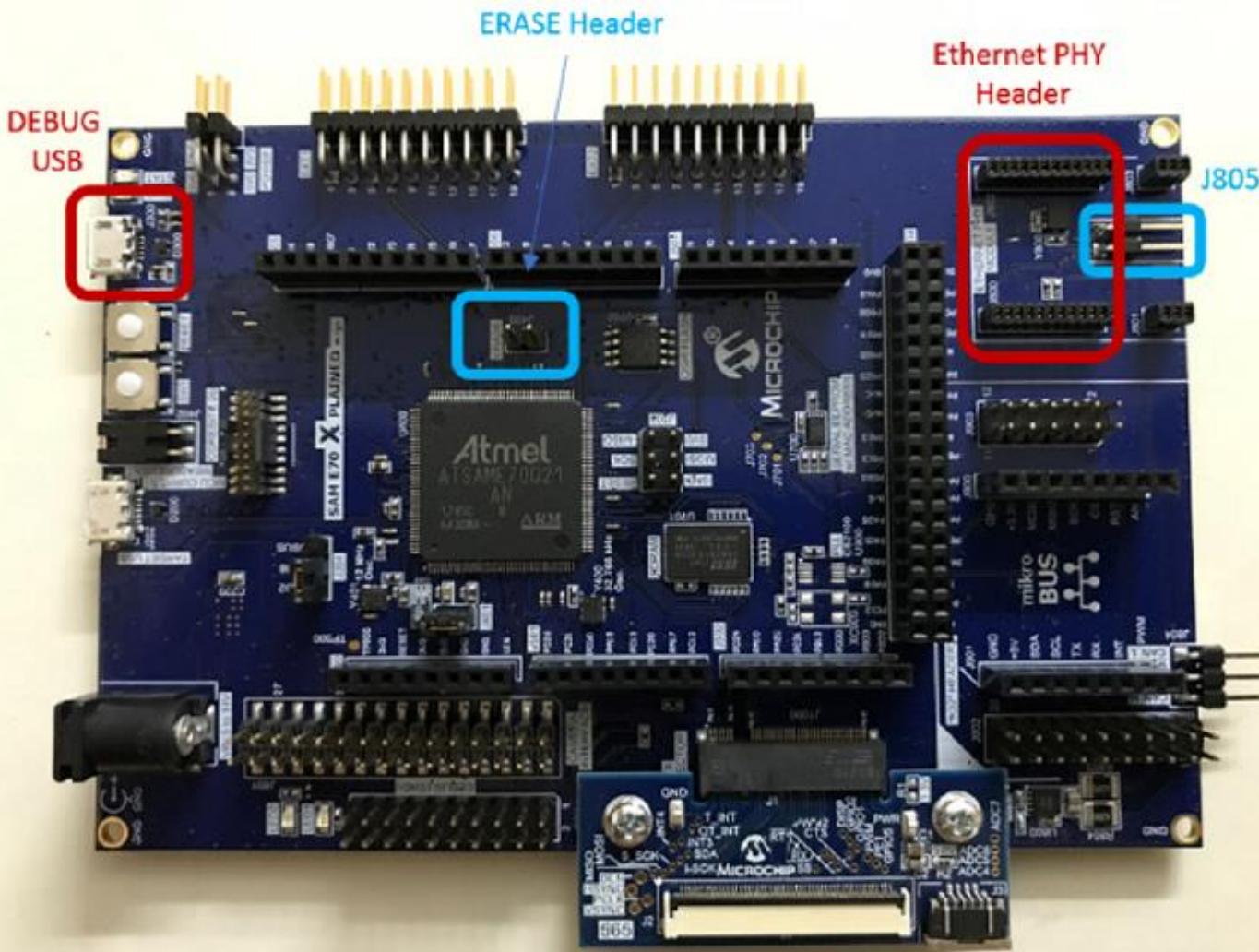
# SAME70 Xplained Ultra

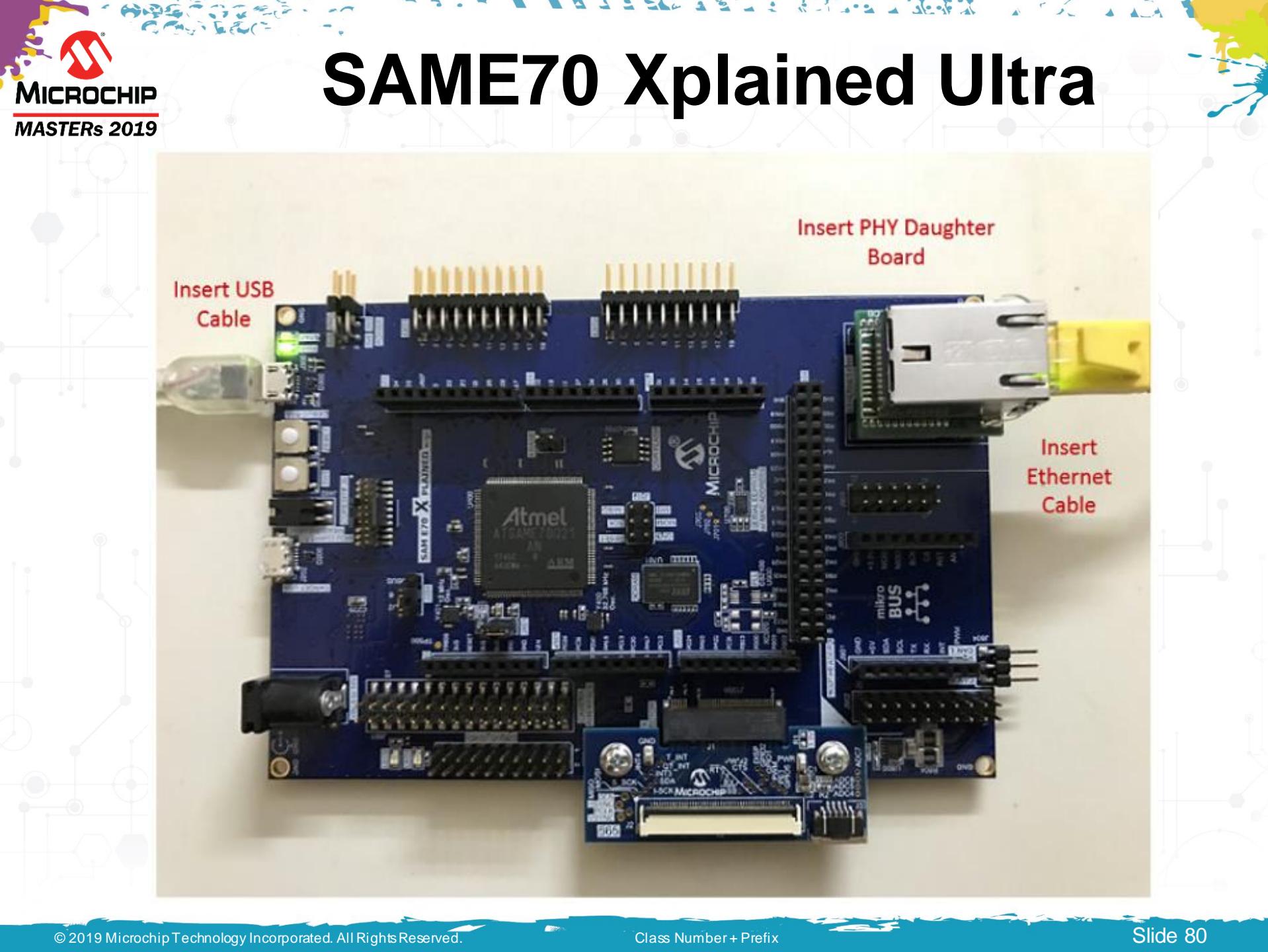


## Features

- ATSAME70Q21 Microcontroller
- One Mechanical Reset Button
- One Mechanical User Push Button
- Two User LEDs
- 12.0 MHz Oscillator (DSC6003)
- 32.768 kHz Oscillator (DSC6083)
- 2-MB SDRAM
- 4-MB QSPI Flash (SST26VF032BA)

# SAME70 Xplained Ultra





# SAME70 Xplained Ultra

# Now it is time for Lab1

A walk through an existing TCP Application,  
adding some changes with  
Microchip Harmony Configurator  
And playing with the tools



# Using the MPLAB® Harmony TCP/IP Stack **BERKELEY MODULE**



# Berkeley Socket

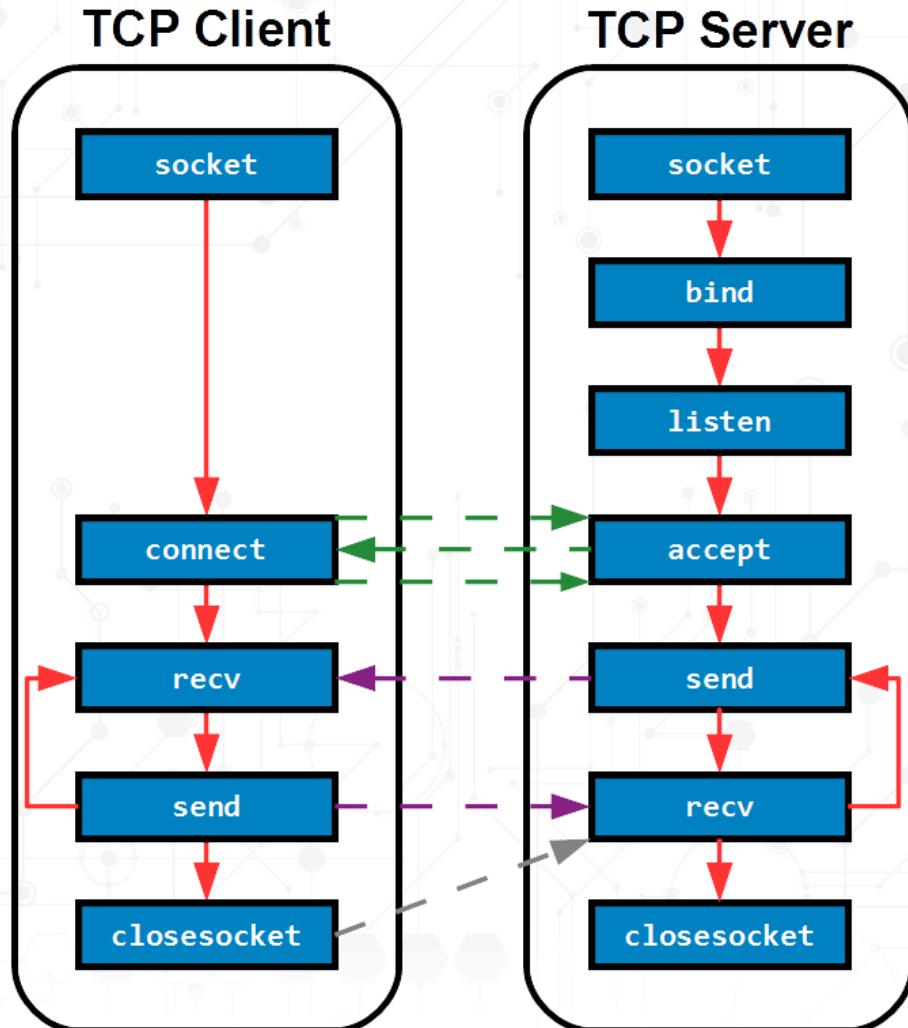
- Berkeley sockets is an application programming interface (API) for Internet sockets and Unix domain sockets, used for inter-process communication (IPC). It is commonly implemented as a library of linkable modules. It originated with the 4.2BSD Unix operating system, released in 1983.
- A socket is an abstract representation (handle) for the local endpoint of a network communication path. The Berkeley sockets API represents it as a file descriptor (file handle) in the Unix philosophy that provides a common interface for input and output to streams of data.



# Berkeley APIs

API Name	Host	Description
<b>socket</b>	Server/Client	Creates a new BSD socket.
<b>bind</b>	Server	Assigns the local address of the communication endpoint.
<b>connect</b>	Client	Assigns the address of the peer communications endpoint, and establishes a connection between the end points.
<b>listen</b>	Server	Sets the specified socket in a listen mode and ready to accept connection requests.
<b>accept</b>	Server	Accept connection requests queued for a listening socket.
<b>send</b>	Server/Client	Send outgoing data on an already connected socket.
<b>recv</b>	Server/Client	Receive incoming data that has been queued for a socket.
<b>closesocket</b>	Server/Client	Closes an existing socket.

# BSD Socket Application Flow



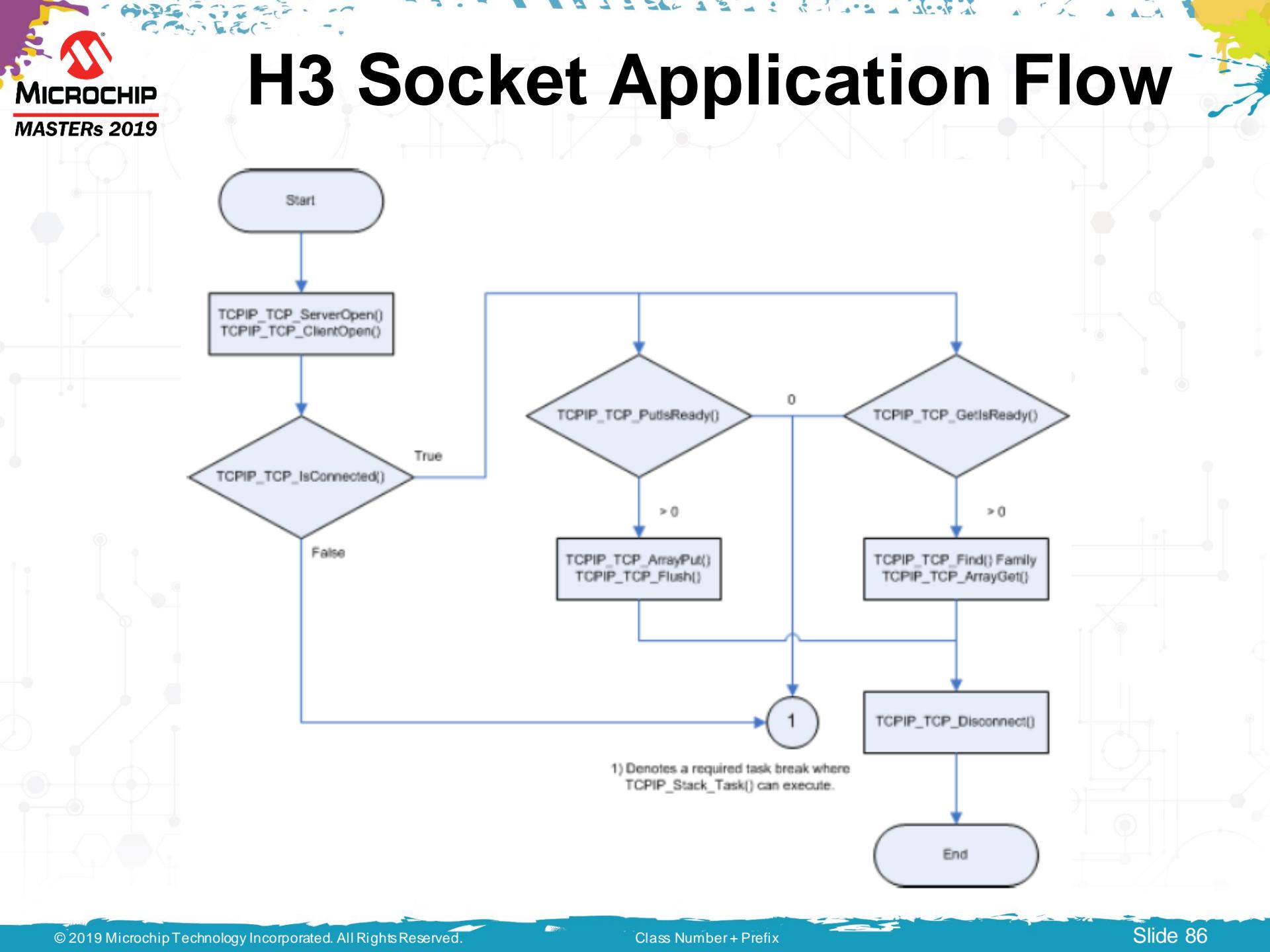
Server and Client both create a socket that uses the TCP Protocol.

Client attempts to establish a TCP connection with the Server using a 3-Way Handshake.

Server Sends data to Client, recv returns a value > 0.

Client Sends data to Server, recv returns a value > 0.

Client Closes Socket, recv returns a value of 0. Server closes Socket.





# Tips

- The Berkeley APIs available in the Harmony Framework can only operate in non-blocking mode.
- TCP/IP Applications using Berkeley Module APIs should be implemented using a non-blocking state machine.
- Example application source code can be found in the app.c file in the following Harmony TCP/IP Demonstration Projects:
  - *berkeley\_tcp\_client*
  - *berkeley\_tcp\_server*
  - *berkeley\_udp\_client*
  - *berkeley\_udp\_server*



# Now it is time for Lab2

## A complete Vending Machine Service Application



# LAB 2

- **Vending Machine**
- **Libraries to establish communication between a TCP Client and Server**
- **Dynamic variables and GET request from an HTTP client handled in Harmony HTTP server**

# LAB 2- TASK 1

## Hardware

- SAM E70 Xplained Ultra Evaluation Kit
- OLED1 Xplained Pro Extension Kit

E70 WEB SERVER

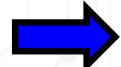


/Vending Machine



COM14 - Tera Term VT  
 File Edit Setup Control Window Help

```
>
=====
web_net_nvram_mpfs_freertos_labi Jun 6 2019 09:01:39
SYS_Initialize: The MPFS2 File System is mounted
MAC TCPPIP_HOSTS_CONFIGURATION[0].macAddr: fc:c2:3d:0d:21:d7
TCP/IP Stack: Initialization Started
TCP/IP Stack: Initialization Ended - success
    Interface GMAC on host RAJI_SHAN - NBNS enabled
<null> -sends message to the server when a Bay is empty
GMAC IP Address: 0.0.0.0
GMAC IP Address: 10.13.33.76
```



TCP/IP Stack Demo Application

### Vending Machine Demo

The GET method appends the data to the end of the URI. You'll see this data following the question mark (?) in your browser's address bar. Data sent via GET is automatically decoded, and stored in the current HTTP connection data buffer. Your application will handle the data in the `TCPIP_HTTP_GetExecute` callback. `TCPIP_HTTP_ArgGet` function provides an easy method to retrieve submitted values for processing.

As an example, this GET form gets count of the VM items as user input and updates the table

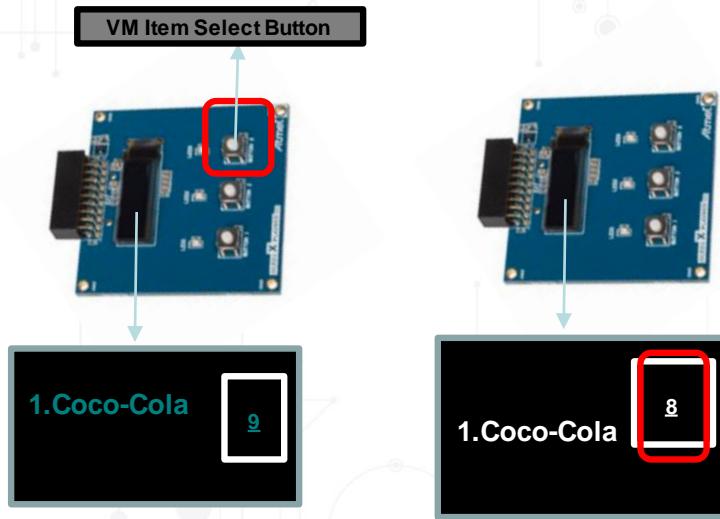
ITEM	COUNT	UPDATE				
Coca-Cola	Coca-Cola Diet	Pepsi	Dr Pepper	Sprite	Fanta	Dasani
9	9	9	9	9	9	9

Copyright © 2018 Microchip Technology, Inc.



# LAB2 – TASK1

## Server updates the VM Client



TCP/IP Stack Demo Application

### Vending Machine Demo

The GET method appends the data to the end of the URI. You'll see this data following the question mark (?) in your browser's address bar. Data sent via GET is automatically decoded, and stored in the current HTTP connection data buffer. Your application will handle the data in the TCPIP\_HTTP\_GetExecute callback. TCPIP\_HTTP\_ArgGet function provides an easy method to retrieve submitted values for processing.

As an example, this GET form gets count of the VM items as user input and updates the table

ITEM	1	COUNT	0	UPDATE
------	---	-------	---	--------

VM TRACKER

Coca-Cola	Coca-Cola Diet	Pepsi	Dr Pepper	Sprite	Fanta	Dasani
8	9	9	9	9	9	9

Copyright © 2018 Microchip Technology, Inc.



## Vending Machine Demo

The **GET** method appends the data to the end of the URI. You'll see this data following the question mark (?) in your browser's address bar. Data sent via **GET** is automatically decoded, and stored in the current **HTTP** connection data buffer. Your application will handle the data in the **TCPPIP\_HTTP\_GetExecute** callback. **TCPPIP\_HTTP\_Arget** function provides an easy method to retrieve submitted values for processing.

As an example, this **GET** form gets count of the VM items as user input and updates the table

ITEM	COUNT	UPDATE
0	0	PDATE
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	Pepper
7	7	
8	8	
9	9	

VM TRACKER						
Coca-Cola	Coca-Cola Diet	Pepsi	Sprite	Fanta	Dasani	None
8	9	9	9	9	9	9

Copyright © 2018 Microchip Technology, Inc.

## Vending Machine Demo

The **GET** method appends the data to the end of the URI. You'll see this data following the question mark (?) in your browser's address bar. Data sent via **GET** is automatically decoded, and stored in the current **HTTP** connection data buffer. Your application will handle the data in the **TCPPIP\_HTTP\_GetExecute** callback. **TCPPIP\_HTTP\_Arget** function provides an easy method to retrieve submitted values for processing.

As an example, this **GET** form gets count of the VM items as user input and updates the table

ITEM	COUNT	UPDATE
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	

VM TRACKER						
Coca-Cola	Coca-Cola Diet	Pepsi	Dr Pepper	Sprite	Fanta	Dasani
8	9	9	9	4	9	9

Copyright © 2018 Microchip Technology, Inc.

**4. Dr Pepper**

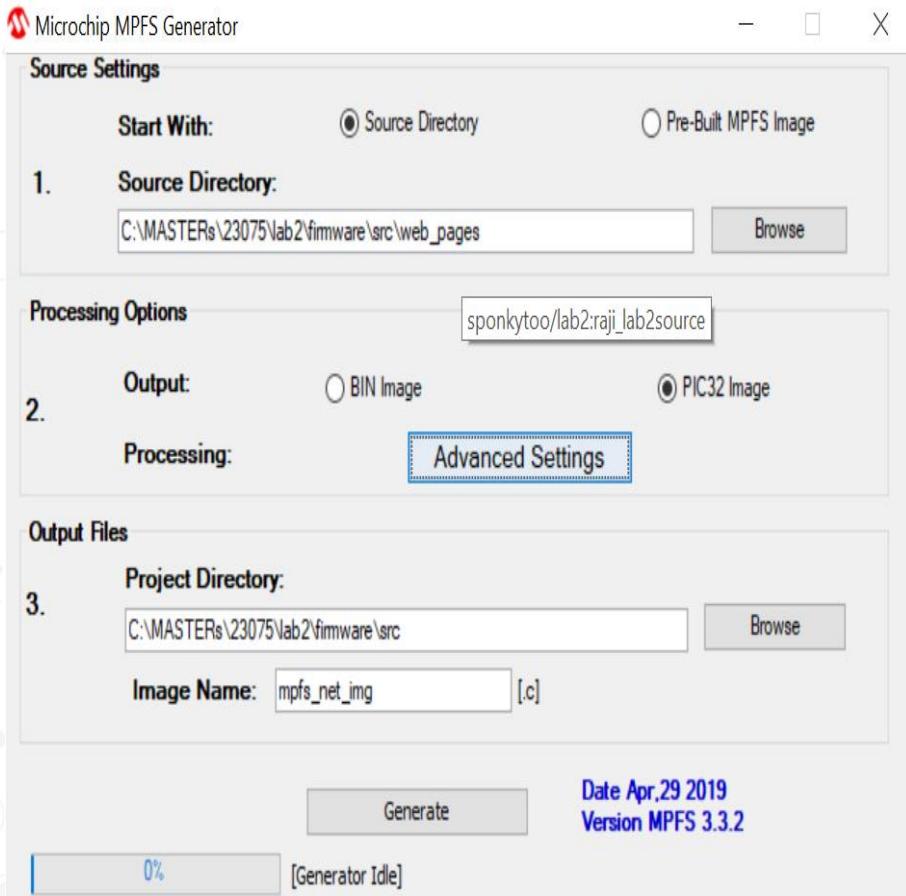
**4**



# VM Server

```
martinruppert — minicom - sudo — 80x24
MSG:2727 from fc:c2:3d:0c:20:44 : 1.Coca-Cola is empty
MSG:2728 from fc:c2:3d:0c:20:44 : 2.Diet-Coke is empty
MSG:2729 from fc:c2:3d:0c:20:44 : 3.Pepsi is empty
MSG:2730 from fc:c2:3d:0c:20:44 : 4.Dr Pepper is empty
MSG:2731 from fc:c2:3d:0c:20:44 : 5.Sprite is empty
MSG:2732 from fc:c2:3d:0c:20:44 : 6.Fanta is empty
MSG:2733 from fc:c2:3d:0c:20:44 : 7.Dasani is empty
MSG:2734 from fc:c2:3d:0c:20:44 : 1.Coca-Cola is empty
MSG:2735 from fc:c2:3d:0c:20:44 : 2.Diet-Coke is empty
MSG:2736 from fc:c2:3d:0c:20:44 : 3.Pepsi is empty
MSG:2737 from fc:c2:3d:0c:20:44 : 4.Dr Pepper is empty
MSG:2738 from fc:c2:3d:0c:20:44 : 5.Sprite is empty
MSG:2739 from fc:c2:3d:0c:20:44 : 6.Fanta is empty
MSG:2740 from fc:c2:3d:0c:20:44 : 7.Dasani is empty
MSG:2741 from fc:c2:3d:0c:20:44 : 1.Coca-Cola is empty
MSG:2742 from fc:c2:3d:0c:20:44 : 2.Diet-Coke is empty
MSG:2743 from fc:c2:3d:0c:20:44 : 3.Pepsi is empty
MSG:2744 from fc:c2:3d:0c:20:44 : 4.Dr Pepper is empty
MSG:2745 from fc:c2:3d:0c:20:44 : 5.Sprite is empty
MSG:2746 from fc:c2:3d:0c:20:44 : 6.Fanta is empty
MSG:2747 from fc:c2:3d:0c:20:44 : 7.Dasani is empty
MSG:2748 from fc:c2:3d:0c:20:44 : 1.Coca-Cola is empty
MSG:2749 from fc:c2:3d:0c:20:44 : 2.Diet-Coke is empty
```

# LAB 2 – MPFS Generator



- Packages web pages into efficient storage format
- **Microchip MPFS Utility**
- Can include the utility on the build process



# Using RTOS with MPLAB Harmony



# Why use RTOS?

## □ Priority based task scheduling

- Ensures better responsiveness to events
- A pre-emptive scheduler always runs the highest priority task that is ready to run, allowing hard real time tasks to meet its deadlines
- Removes dependency of tasks on the execution time of other functions running in the “super loop”

## □ Efficient CPU usage

- Does not waste CPU cycles by running tasks waiting on events/ resources
- Idle task is run when there are no application tasks ready to run. This allows putting CPU in low power modes

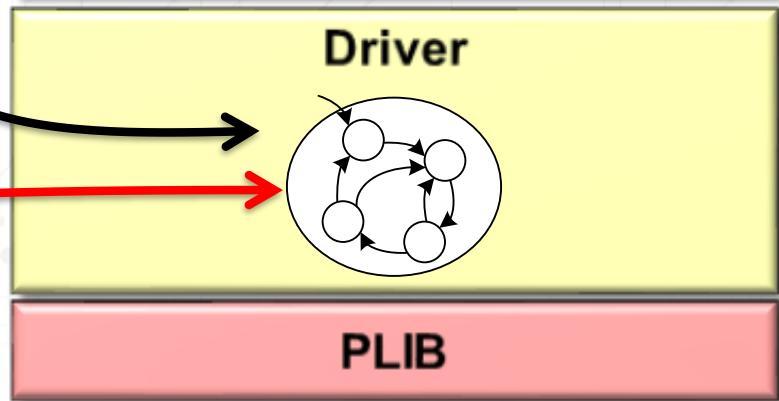
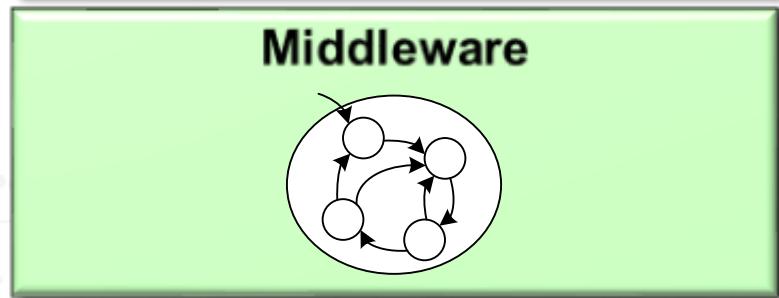
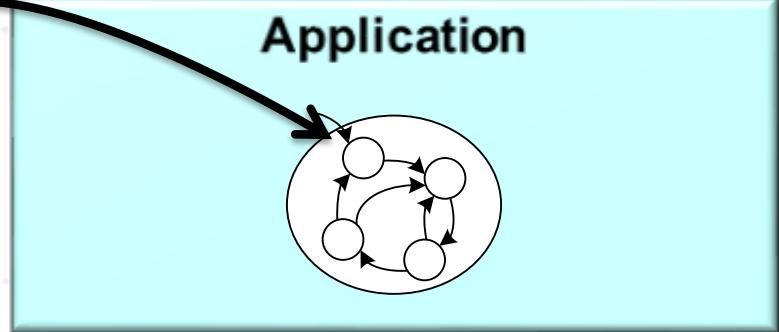


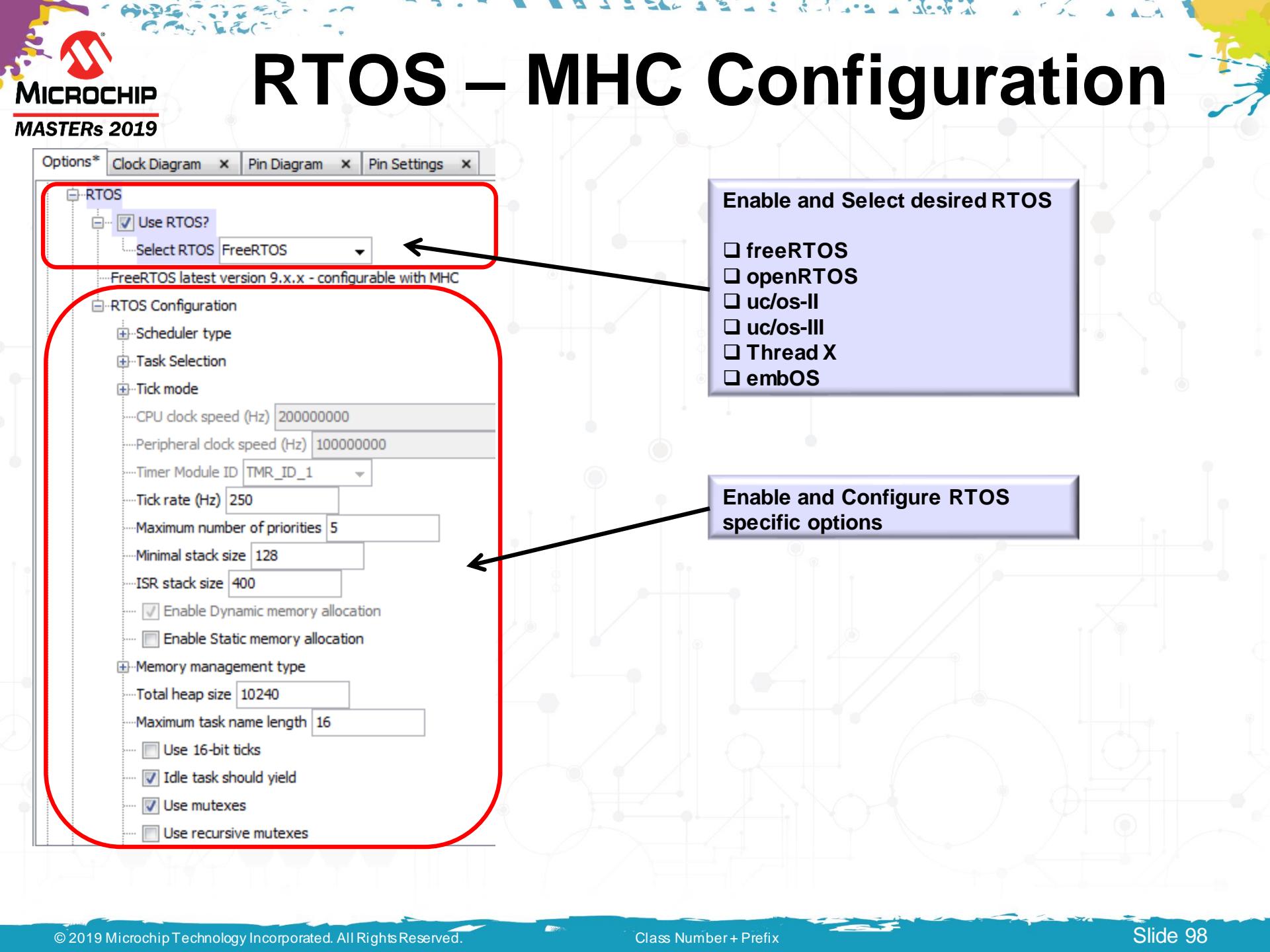
# RTOS Driven

```
static void _SENSORTASK_Tasks(void)
{
    while(1)
    {
        SENSORTASK_Tasks();
        vTaskDelay(10); /*Give time back*/
    }
}
```

```
void _SYS_Tasks (void *param)
{
    while(1)
    {
        DRV_SPI_Tasks(sysObj.spiObjectIdx0);
        vTaskDelay(10); /*Give time back*/
    }
}
```

```
void SPI_ISR(void)
{
    DRV_SPI_Tasks(sysObj.spiObjectIdx0);
}
```





# RTOS – MHC Configuration

Options \* Clock Diagram × Pin Diagram × Pin Settings ×

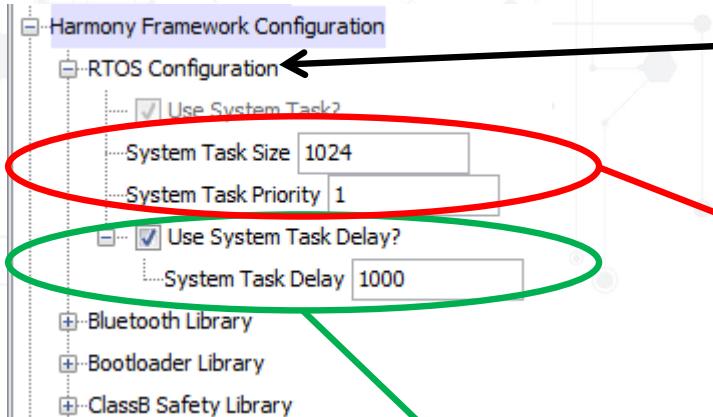
Use RTOS?  
Select RTOS: FreeRTOS

FreeRTOS latest version 9.x.x - configurable with MHC

**RTOS Configuration**

- Scheduler type
- Task Selection
- Tick mode**
- CPU clock speed (Hz) 200000000
- Peripheral clock speed (Hz) 100000000
- Timer Module ID TMR\_ID\_1
- Tick rate (Hz) 250
- Maximum number of priorities 5
- Minimal stack size 128
- ISR stack size 400
- Enable Dynamic memory allocation
- Enable Static memory allocation
- Memory management type**
- Total heap size 10240
- Maximum task name length 16
- Use 16-bit ticks
- Idle task should yield
- Use mutexes
- Use recursive mutexes

# RTOS – MHC Configuration



Block the task for 1000 milliseconds. Allows other tasks to run.

RTOS configuration for the **SYS\_Tasks** thread.

```
void SYS_Tasks ( void )
{
    /* Create OS Thread for Sys Tasks. */
    xTaskCreate((TaskFunction_t) _SYS_Tasks,
                "Sys Tasks",
                1024, NULL, 1, NULL);
    ....
}
```

```
static void _SYS_Tasks ( void)
{
    while(1)
    {
        /* Maintain system services */
        SYS_CONSOLE_Tasks(sysObj.sysConsole0);

        /* Maintain Device Drivers */

        /* Maintain Middleware */

        /* Task Delay */
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```



# RTOS – MHC Configuration

RTOS configuration for a driver instance which is configured for polled mode operation

Standalone – Run the driver task as a separate RTOS thread

```
void SYS_Tasks ( void )
{
    /* Create OS Thread for DRV_SPI Instance 0 Tasks. */

    xTaskCreate((TaskFunction_t) _DRV_SPI_IDX0_Tasks,
                "DRV_SPI Instance 0 Tasks",
                1024, NULL, 1, NULL);
    ...
}
```

```
void _DRV_SPI_IDX0_Tasks(void)
{
    while(1)
    {
        DRV_SPI_Tasks(sysObj.spiObjectIdx0);

        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Combined with System Tasks – Run the driver task as part of the `_SYS_Tasks` thread

```
static void _SYS_Tasks ( void )
{
    while(1)
    {
        /* Maintain system services */
        SYS_CONSOLE_Tasks(sysObj.sysConsole0);

        /* Maintain Device Drivers */
        DRV_SPI_Tasks(sysObj.spiObjectIdx0);

        /* Maintain Middleware */

        /* Task Delay */
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

© 2019 Microchip Technology Incorporated. All Rights Reserved.

Class Number + Prefix

Slide 100

# System Initialization & Tasks

```

int main(void)
{
    SYS_Initialize(NULL);

    while(true)
    {
        SYS_Tasks();
    }

    return(EXIT_VALUE);
}

void SYS_Tasks ( void )
{
    /* Create OS Thread for Sys Tasks. */
    xTaskCreate((TaskFunction_t) _SYS_Tasks, "Sys Tasks",1024, NULL, 3, NULL);
    xTaskCreate((TaskFunction_t) _USB_Tasks, "USB Tasks",1024, NULL, 3, NULL);

    /* Create OS Thread for SENORTASK Tasks. */
    xTaskCreate((TaskFunction_t) _SENSORTASK_Tasks, "SENSORTASK Tasks", 1024, NULL, 2,
NULL);

    /* Create OS Thread for EEPROMTASK Tasks. */
    xTaskCreate((TaskFunction_t) _EEPROM_Tasks, "EEPROM Tasks", 1024, NULL, 1, NULL);
    .....

    /*****
     * Start RTOS *
     *****/
    vTaskStartScheduler(); /* This function never returns. */
}

```

```

void SYS_Initialize( void* data )
{
    SYS_CLK_Initialize( &clkInit );
    BSP_Initialize();
    sysObj.spiObjectIdx0 = DRV_SPI_Initialize (DRV_SPI_INDEX_0,
&drvSpi0InitData);
    sysObj.drvTmr0      = DRV_TMR_Initialize(DRV_TMR_INDEX_0,
&drvTmr0InitData);
    .....

    /*create Tasks*/
    APP_Initialize();
}

```



# Data Protocol

- Data exchanged through a TCP/UDP Socket can use a custom protocol or a standard protocol such as HTTP or MQTT.
- Data can be formatted using a standard data-interchange format such as JSON or XML.

# Data Protocol Examples

Example Sensor Data: “ID: AZ2355, Temperature: 25C, Time: 08:56, Type: K Thermocouple”

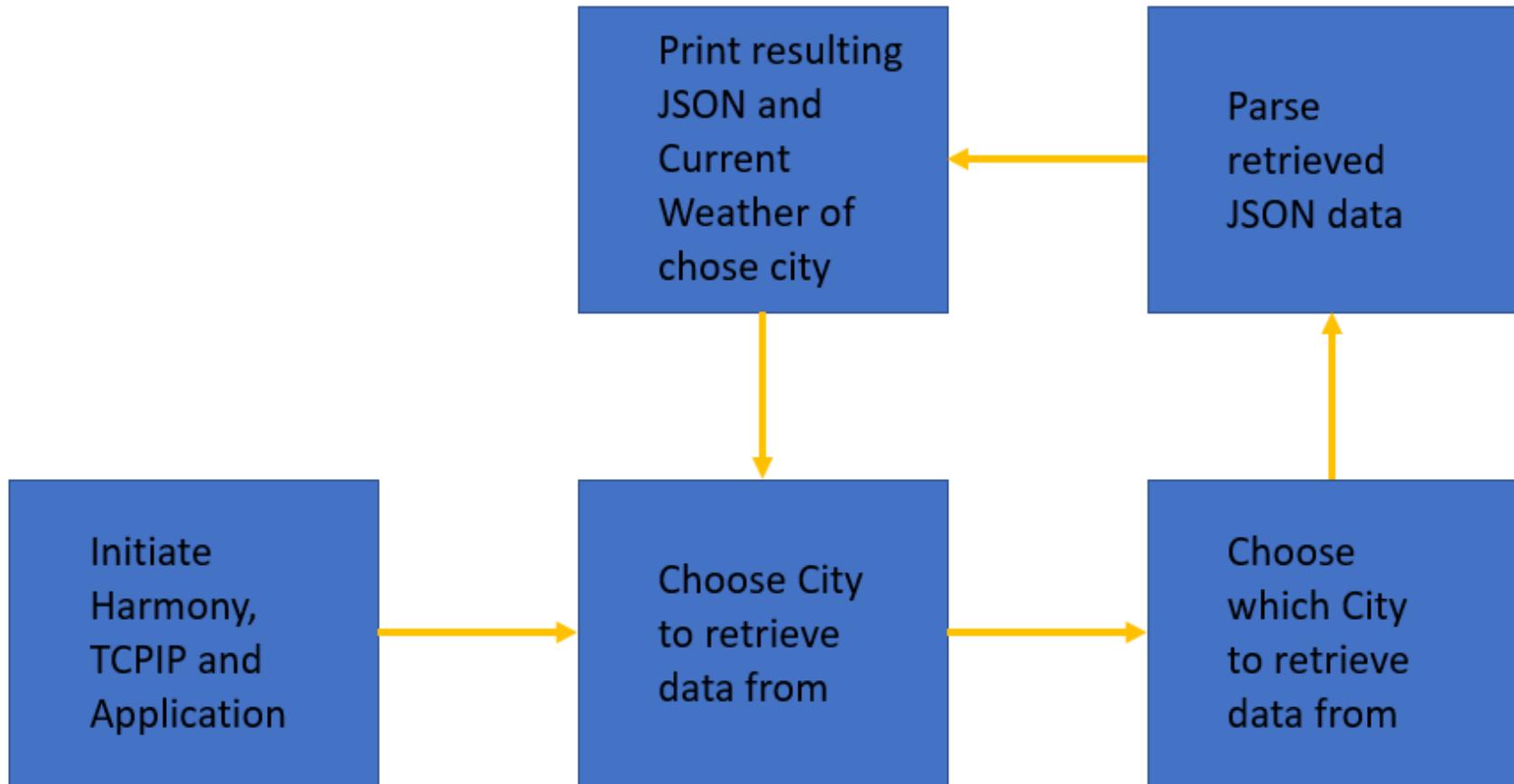
Custom Protocol	Custom over HTTP	JSON	XML
<AZ2355 25C 08:56 KTC>	<pre>POST/sensorData/temperature.html HTTP/1.0 User-Agent: HTTPTool/1.0 Content-Type: application/x-www-form-urlencoded Content-Length: 23</pre> <AZ2355 25C 08:56 KTC>	<pre>{   "sensor": {     "id": "AZ2355",     "type": "kthermocouple"   },   "readings": [     {       "time": "08:56",       "temperature": 25,       "unit": "C"     }   ] }</pre>	<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;root&gt;   &lt;readings&gt;     &lt;element&gt;        &lt;temperature&gt;25&lt;/temperature&gt;        &lt;time&gt;08:56&lt;/time&gt;         &lt;unit&gt;C&lt;/unit&gt;       &lt;/element&gt;     &lt;/readings&gt;     &lt;sensor&gt;       &lt;id&gt;az2355&lt;/id&gt;       &lt;type&gt;kthermocouple&lt;/type&gt;     &lt;/sensor&gt;   &lt;/root&gt;</pre>



# Now it is time for Lab3

## Getting the Weather Report

# Pulling the Weather Data



# Pulling the Weather Data

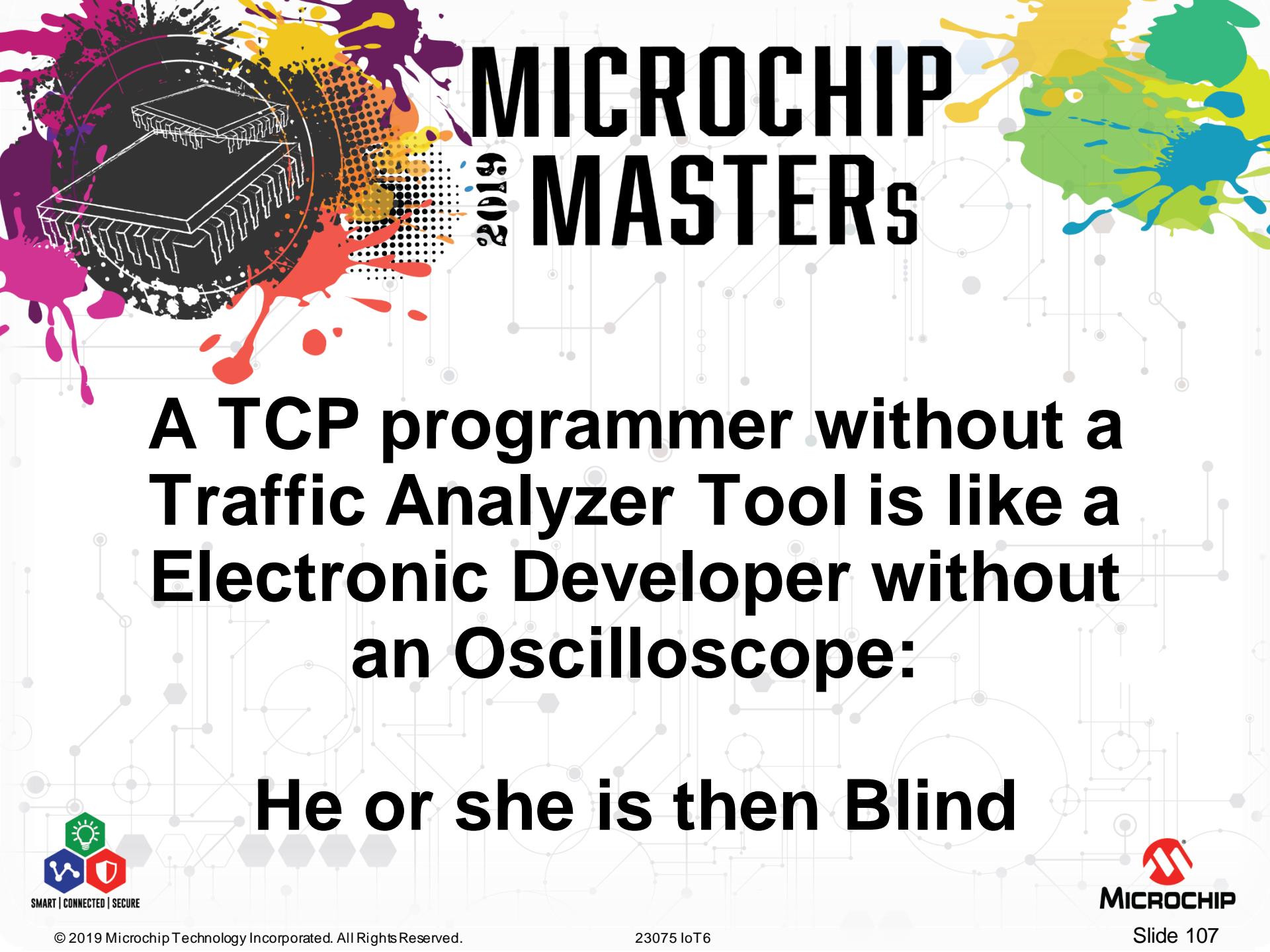
```
martinruppert — minicom - sudo — 80x26
=====
request weather lab3 Jun 10 2019 12:27:31
MAC TCPIP_HOSTS_CONFIGURATION[0].macAddr: fc:c2:3d:0c:20:44
TCP/IP Stack: Initialization Started
TCP/IP Stack: Initialization Ended - success
    Interface GMAC on host MCHPBOARD_C      - NBNS disabled
GMAC IP Address: 192.168.0.19
Waiting for command type: requestWeather <city>

[>rw Phoenix

>cityBuffer: Phoenix
Starting connection

Connection Closed
resultingJson:
  {"coord":{"lon":-112.08,"lat":33.45}, "weather":[{"id":801,"main":"Clouds","des

Current Weather in Phoenix
Humidity: 13
Pressure: 1016
Temperature: 34.22
Main Weather: Clouds
```



# **MICROCHIP MASTERS**

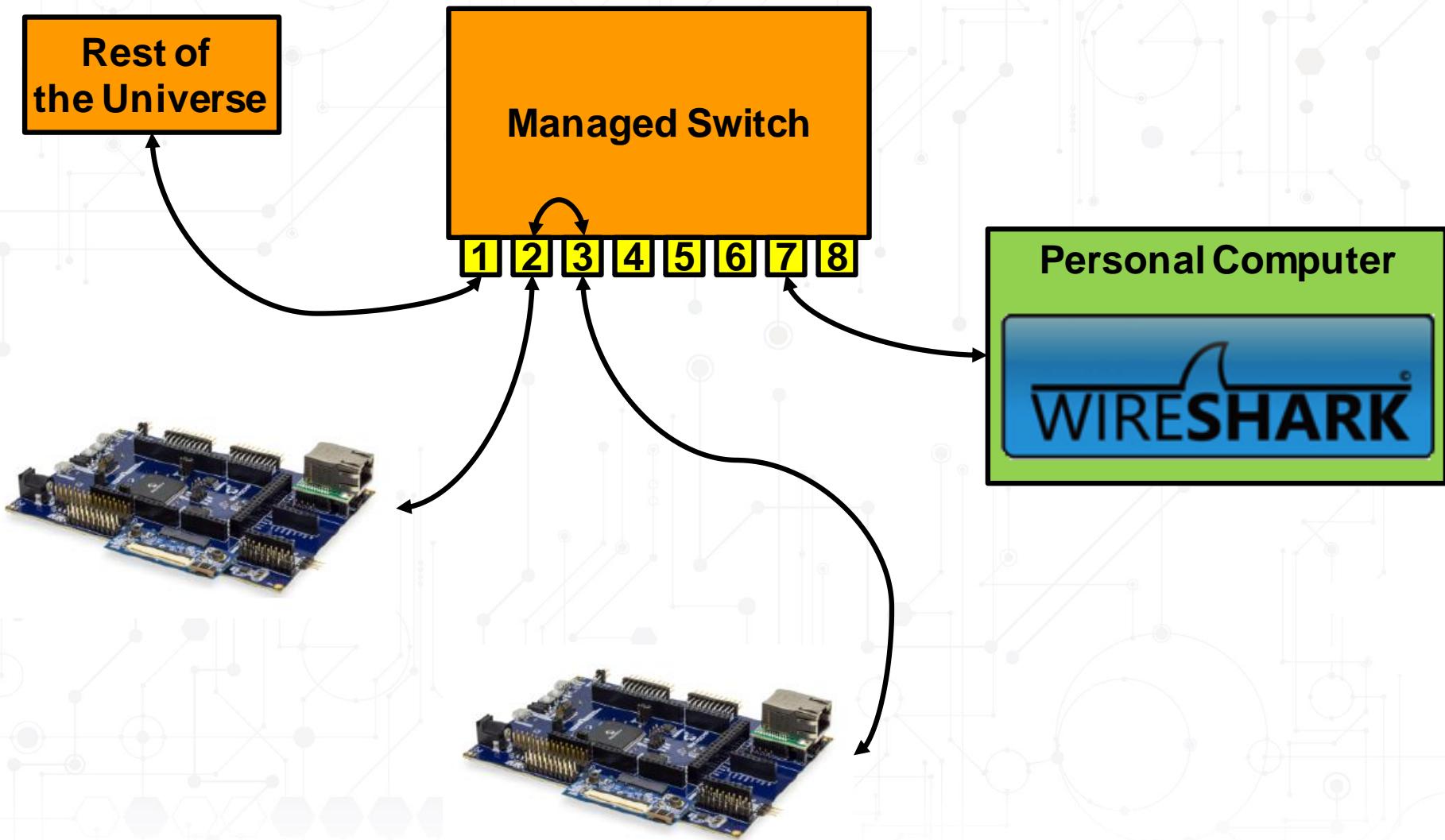
2019

**A TCP programmer without a  
Traffic Analyzer Tool is like a  
Electronic Developer without  
an Oscilloscope:**

**He or she is then Blind**



# How to Analyze the Traffic





MICROCHIP  
MASTERs 2019



# Configuration Software for Managed Switch

The image shows two windows of the ProSAFE Plus Configuration Utility for a NETGEAR GS108Ev2 switch.

**Top Window (Switch Selection):**

- Title: ProSAFE Plus Configuration Utility
- NETGEAR logo: Connect with Innovation™
- Menu bar: Network, System, VLAN, QoS, Help
- Sub-menu: Switch Selection
- Message: Please select a switch to configure
- Table: Discovered Switches

Product	Switch Name	MAC Address	IP Address	Located on IP Network
GS108Ev2	MySwitch	10:0d:7f:bb:ae:91	192.168.0.101	192.168.0.192

- Language selection: Select Language: English, QUIT

**Bottom Window (Port Mirroring Configuration):**

- Title: ProSAFE Plus Configuration Utility-GS108Ev2-MySwitch
- NETGEAR logo: Connect with Innovation™
- Menu bar: Network, System, VLAN, QoS, Help
- Sub-menu: Status, Maintenance, Monitoring, Port Mirroring, Port Statistics, Mirroring, Cable Tester
- Language selection: Select Language: English, QUIT
- Panel: Port Mirroring Configuration

Mirroring	Enable
Source Port	01    02    03
Port	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
Destination Port	07

- Port List: 04, 05, 06, 07, 08

# Wireshark Filter

Ethernet: en0

eth.addr == fc:c2:3d:0c:20:44

No.	Time	Source	Destination	Protocol	Leng	Info
11	1.303395	192.168.0.19	192.168.0.108	TCP	60	51094 → 80 [SYN] Seq=0 Win=512 Len=0 MSS=1460
12	1.304437	192.168.0.108	192.168.0.19	TCP	60	80 → 51094 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0 MSS=1460
13	1.305409	192.168.0.19	192.168.0.108	TCP	60	51094 → 80 [ACK] Seq=1 Ack=1 Win=512 Len=0
14	1.307340	192.168.0.19	192.168.0.108	TCP	107	[TCP segment of a reassembled PDU]
15	1.308279	192.168.0.19	192.168.0.108	TCP	60	51094 → 80 [FIN, ACK] Seq=54 Ack=1 Win=512 Len=0
16	1.308400					en=0
17	1.309324					en=0
18	1.311302					512 Len=0
19	1.312324					en=0
20	3.286390					S=1460

Frame 14: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface en0  
**eth.addr == fc:c2:3d:0c:20:44**

- ▶ Frame 14: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface en0
- ▶ Ethernet II, Src: (00:0c:29:4f:00:00) [00:0c:29:4f:00:00], Dst: (00:00:00:00:00:00) [00:00:00:00:00:00]
- ▶ Internet Protocol Version 4, Src: 192.168.0.19 (192.168.0.19), Dst: 192.168.0.108 (192.168.0.108)
- ▶ Transmission Control Protocol

```

Source Port: 51094
Destination Port: 80
[Stream index: 2]
[TCP Segment Len: 53]
Sequence number: 1      (relative sequence number)
[Next sequence number: 54      (relative sequence number)]
Acknowledgment number: 1      (relative ack number)
Header Length: 20 bytes
▶ Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 512]
[Window size scaling factor: -2 (no window scaling used)]
▶ Checksum: 0x1662 [validation disabled]
Urgent pointer: 0
▶ [SEQ/ACK analysis]
TCP segment data (53 bytes)

```

0000	00 04 25 1c a0 02	fc c2 3d 0c 20 44	08 00 45 00	..%..... =. D..E.
0010	00 5d 00 41 00 00	64 06 c4 8a c0 a8 00	13 c0 a8	..].A..d. .......
0020	00 6c c7 96 00 50	bb b3 d3 f3 e0 b0 e1 a6 50	18	.l...P.. .....P.
0030	02 00 16 62 00 00	4d 53 47 3a 38 32	33 20 66 72	...b..MS G:823 fr
0040	6f 6d 20 66 63 3a	63 32 3a 33 64 3a	30 63 3a 32	om fc:c2 :3d:0c:2
0050	30 3a 34 34 20 3a	20 31 2e 43 6f 63	61 2d 43 6f	0:44 : 1 .Coca-Co
0060	6c 61 20 69 73 20	65 6d 70 74 79		la is em pty

wireshark\_pcappng\_en0\_20190610171027\_P8zh8Z

Packets: 69 · Displayed: 27 (39.1%) · Dropped: 0 (0.0%) · Profile: Default



# Payload Data

Wireshark · Follow TCP Stream (tcp.stream eq 2) · wireshark\_pcappng\_en0\_201906101710...

MSG:823 from fc:c2:3d:0c:20:44 : 1.Coca-Cola is empty

1 client pkt(s), 0 server pkt(s), 0 turn(s).

Entire conversation (53 bytes) Show data as ASCII Stream 2

Find: Find Next

Help Hide this stream Print Save as... Close

0000 00 04 25 1c a0 02 fc c2 3d 0c 20 44 08 00 45 00 ..%..... =. D..E.
0010 00 5d 10 41 00 00 64 06 c4 8a c0 a8 00 13 c0 a8 .].A..d. .......
0020 00 6c c7 96 00 50 bb b3 d3 f3 e0 b0 e1 a6 50 18 .l...P. .....P.
0030 02 00 16 62 00 00 4d 53 47 3a 38 32 33 20 66 72 ...b..MS G:823 fr
0040 6f 6d 20 66 63 3a 63 32 3a 33 64 3a 30 63 3a 32 om fc:c2 :3d:0c:2
0050 30 3a 34 34 20 3a 20 31 2e 43 6f 63 61 2d 43 6f 0:44 : 1 .Coca-Co
0060 6c 61 20 69 73 20 65 6d 70 74 79 la is em pty

A data segment used in reassembly of a lower-level protocol (tcp.segment\_data), 53 bytes | Packets: 69 · Displayed: 9 (13.0%) · Dropped: 0 (0.0%) | Profile: Default



**Many thanks for attending this class**

**And...**

**...let us make this world a better place with  
TCPIP**



**MICROCHIP**

**MASTERs 2019**

**SOFTWARE:**

You may use Microchip software exclusively with Microchip products. Further, use of Microchip software is subject to the copyright notices, disclaimers, and any license terms accompanying such software, whether set forth at the install of each program or posted in a header or text file.

Notwithstanding the above, certain components of software offered by Microchip and 3rd parties may be covered by "open source" software licenses – which include licenses that require that the distributor make the software available in source code format. To the extent required by such open source software licenses, the terms of such license will govern.

**NOTICE & DISCLAIMER:**

These materials and accompanying information (including, for example, any software, and references to 3rd party companies and 3rd party websites) are for informational purposes only and provided "AS IS." Microchip assumes no responsibility for statements made by 3rd party companies, or materials or information that such 3rd parties may provide.

MICROCHIP DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING ANY IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY DIRECT OR INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND RELATED TO THESE MATERIALS OR ACCOMPANYING INFORMATION PROVIDED TO YOU BY MICROCHIP OR OTHER THIRD PARTIES, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THE DAMAGES ARE FORESEEABLE. PLEASE BE AWARE THAT IMPLEMENTATION OF INTELLECTUAL PROPERTY PRESENTED HERE MAY REQUIRE A LICENSE FROM THIRD PARTIES.

**TRADEMARKS:**

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INCnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MIWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omnipresent Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQL, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2019, Microchip Technology Incorporated, All Rights Reserved.