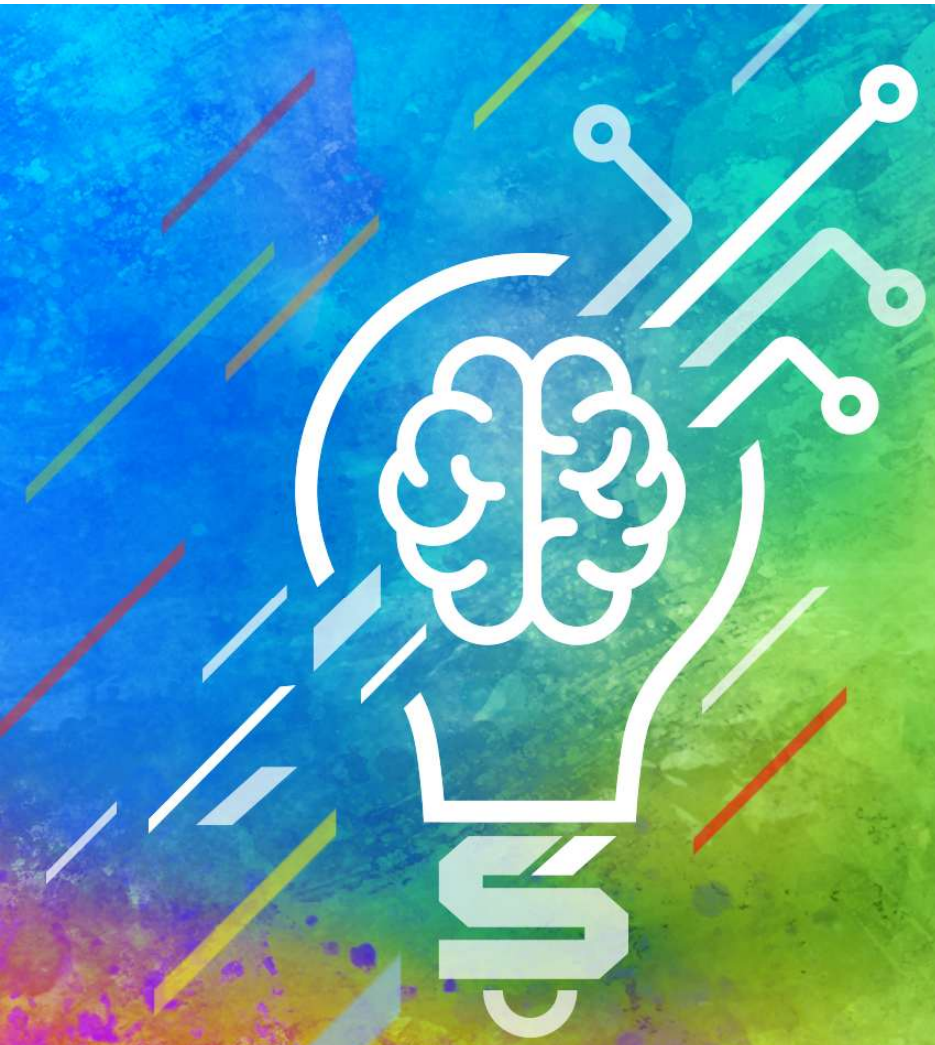


MASTERS 24

EMPOWERING INNOVATION

FRM06
Using Zephyr OS with Microchip Devices

Daniel Tang

MICROCHIP

Westborough Office

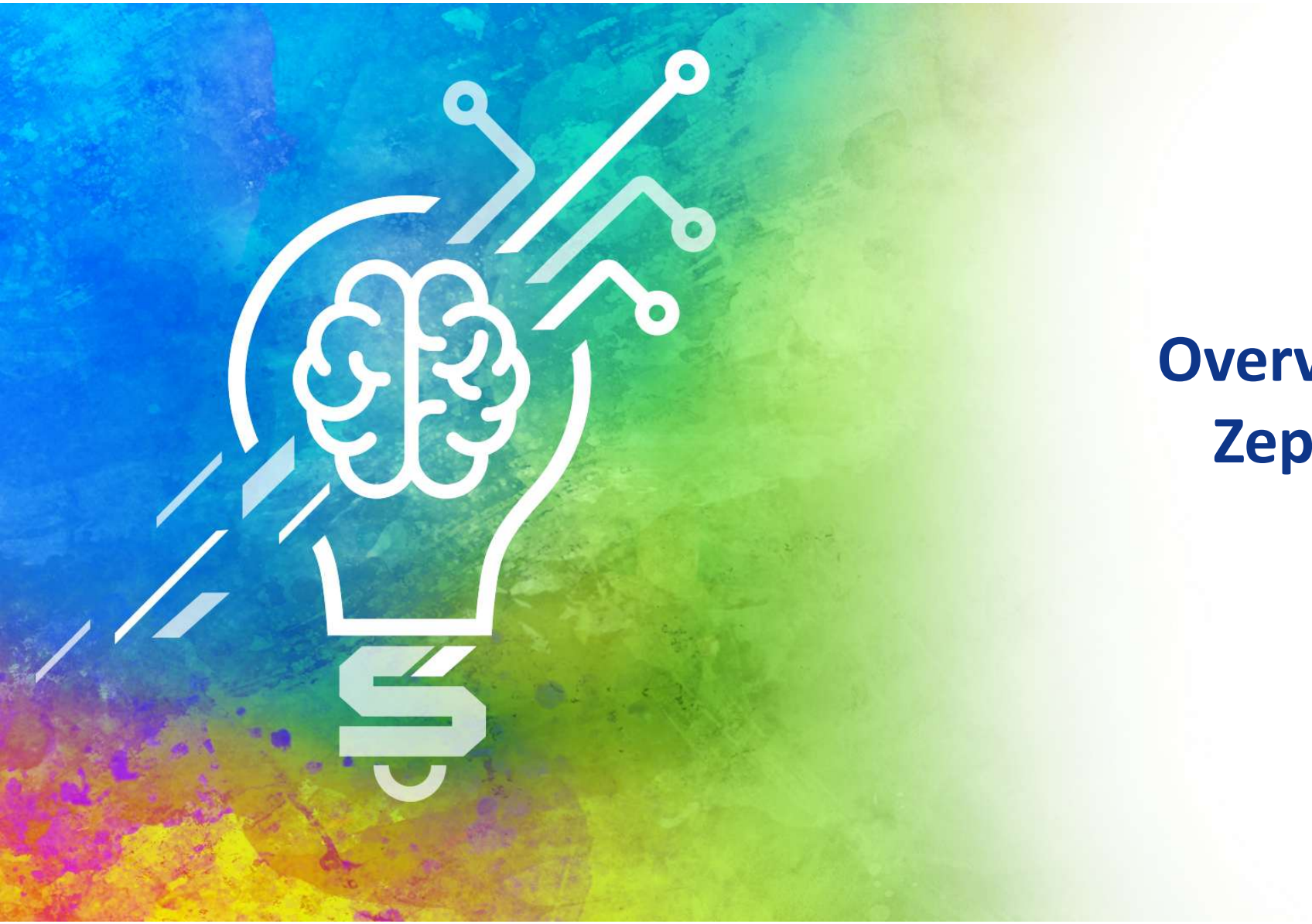
- Wifi
  - PolarFire / coolestfpga (WPA2)
- Food (Microsemi Conference Room)
  - Breakfast/Coffee
  - Lunch
- Restrooms
  - Out towards elevators down main hallway on left
- MASTERs
  - August 10-13, 2026 – Scottsdale Arizona (Marriot Desert Ridge)
    - RTOS track (4 Zephyr Classes)
    - Wireless, Security, Motor Control, Networking, AI/ML, FuSA, FPGA, HMI, Software tools

Class notes and finished labs Github

- https://github.com/dtang-microchip/ZephyrGettingStartedLab
  - Completed labs in their own branches (lab1, lab2, lab3)
  - LabManual PDF
  - WINC1500 Firmware "Updater"

Agenda

- **Overview of Zephyr OS**
  - Zephyr OS SDK, West and Zephyr OS Project Folder
  - Finding Zephyr OS APIs
  - Lab 1 – Blinky/Creating Your First Project
- **Zephyr OS Kernel Services**
  - Tasks and Inter-process Communication
  - Lab 2 – Creating threads and queues
- **Zephyr OS Devicetree**
  - Devicetree Overview and Vocabulary
  - .dtsi files and overlays
  - Lab 3 – Using Console to enable Wifi Connections with WINC1500
- **Zephyr OS Services and Finding Support**
- **Summary**



Overview of Zephyr OS

What is an RTOS?

Basic Superloop Code

```
#define LOOPDELAY          100ul

void main() {
        setupBoard();
        setupPeripherals();

        while(true) {
                // Gather Sensor Data
                temperature = getTemperature(ADC_CHAN_1);
                current = getCurrent(ADC_CHAN_2);
                voltage = getVoltage(ADC_CHAN_3);

                // Act on current sensor data
                checkFaults(temperature, current, voltage);
                serviceMotor(temperature, current, voltage);
                serviceUI();
                delayMS(LOOPDELAY);
        }
}
```


MICROCHIP

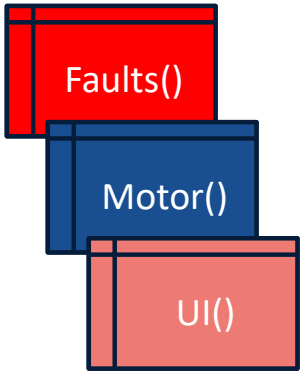
RTOS Code - Threads

```
void main() {
        setupBoard();
        setupPeripherals();

        createTask(Temperature);
        createTask(Current);
        createTask(Voltage);
        createTask(Faults);
        createTask(Motor);
        createTask(UI);

        startScheduler();

        while(true) {
                runScheduler();
        }
}
```

```
void Temperature() {
        while(true) {
                t = getADC(CH1);
                convertTemp(t, T);
                taskDelayMS(1000);
        }
```

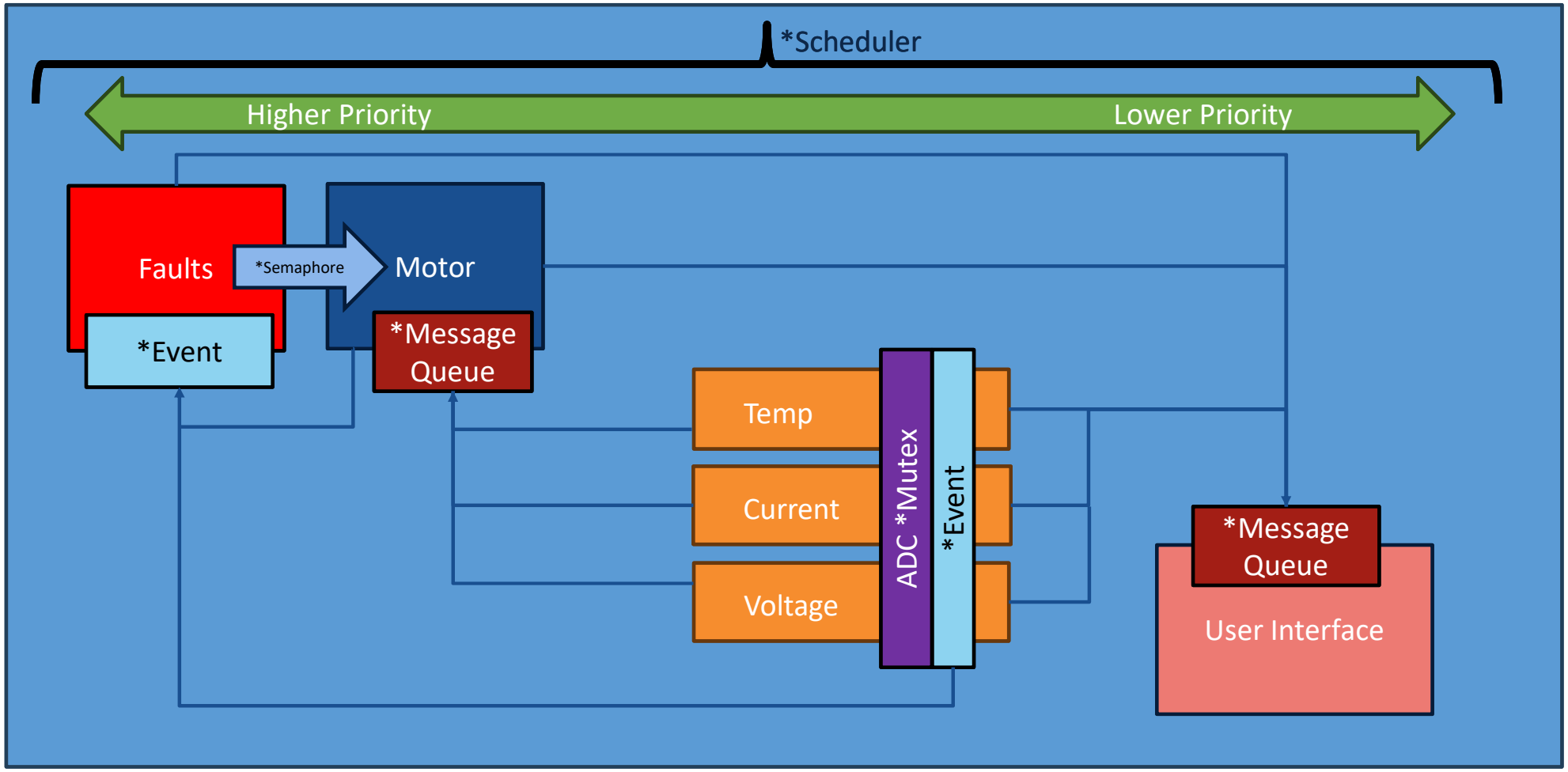
```
void Current() {
        while(true) {
                i = getADC(CH2);
                convertCurr(i, I);
                taskDelayMS(10);
        }
```

```
void Voltage() {
        while(true) {
                v = getADC(CH3);
                convertVolt(v, V);
                taskDelayMS(100);
        }
}
```

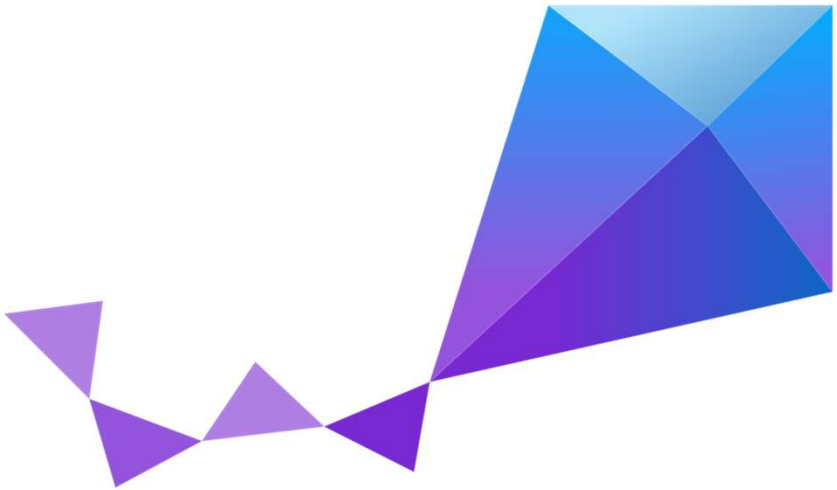


MICROCHIP

RTOS – System Architecture



What is Zephyr OS



- Real Time OS (RTOS)
- Board Support Packages for many devices
- Kernel Services and OS Services
- Supported by Linux Foundation and several large vendors
- Open Source Software

Zephyr OS SDK

- Toolchain (Compiler, Linker, Programmer)
- CMake Based build environment
- Hardware managed by DeviceTree
- Board Support Packages (.dtsi, openOCD config, etc)
- QEMU and Twister
- **Getting Started:**
  - https://docs.zephyrproject.org/latest/develop/getting_started/index.html

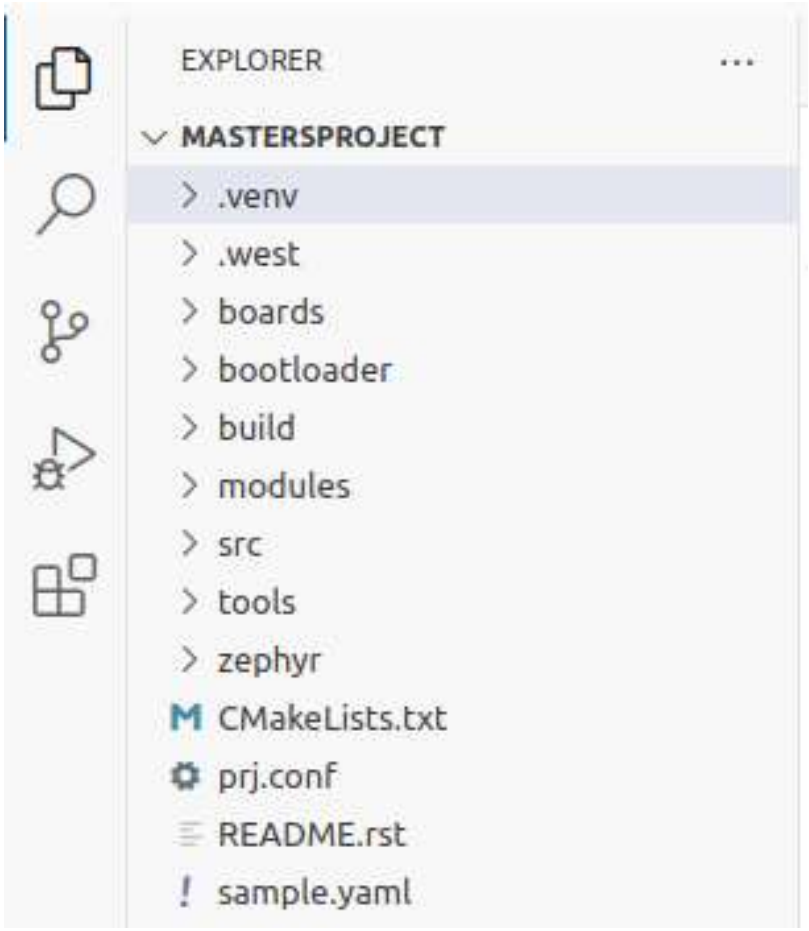
MICROCHIP

Zephyr OS *west* meta-tool

- “Swiss-army knife” tool for working with a Zephyr OS project
  - **west init**: Create a new project folder and download Zephyr OS code (This has already been done to save lab time)
  - **west build**: Build and compile code for a target
  - **west flash**: Flash code to a target device
  - **west debug**: Open a debug session to a target device
  - Lots more… https://docs.zephyrproject.org/latest/develop/west/index.html
- Python tool, typically installed in a VENV
  - From your project directory:
    `$>.venv/Script/Activate.ps1`
- west [common-opts] <command> [opts] <args>
  - Ex: `west build -p always -b samd21_xpro /samples/basic/blinky`
- West is pluggable and extendable to customize for your own project needs

MICROCHIP

Zephyr OS Project Folder



- **To Create a Custom Project, Add/Create:**
  - src/main.c
  - boards
  - CMakeLists.txt
  - prj.conf
  - Optional:
    - Readme.rst
    - sample.yaml
    - git init
    - .gitignore

MICROCHIP

Debugging a Zephyr OS Project

- printk();
  - Uses semihosting
  - picocom or similar
- west debug
  - `help`
  - `tui enable`
- VSCode launch.json
  - Requires cortex-debug

MICROCHIP

Zephyr OS APIs

- https://docs.zephyrproject.org/latest/doxygen/html/index.html

- Demo – Navigate the Zephyr OS webpage

Lab 1: Build and Deploy a Sample Application, then create a new project tree

```
(.venv) PS C:\Users\C75166\zephyrproject> west build -p always -b same54_xpro .\zephyr\samples\basic\blinky\
-- west build: making build dir C:\Users\C75166\zephyrproject\build pristine
-- west build: generating a build system
Loading Zephyr default modules (Zephyr base).
-- Application: C:/Users/C75166/zephyrproject/zephyr/samples/basic/blinky
-- CMake version: 3.27.9
-- Found Python3: C:/Users/C75166/mastersproject/.venv/Scripts/python.exe (found suitable version "3.11.6", minimum required is "3.10") found components: Interpreter
-- Cache files will be written to: C:/Users/C75166/zephyrproject/zephyr/.cache
-- Zephyr version: 4.0.0-rc1 (C:/Users/C75166/zephyrproject/zephyr)
-- Found west (found suitable version "1.2.0", minimum required is "0.14.0")
-- Board: same54_xpro, qualifiers: same54p20a
-- ZEPHYR_TOOLCHAIN_VARIANT not set, trying to locate Zephyr SDK
-- Found host-tools: zephyr 0.16.8 (C:/Users/C75166/zephyr-sdk-0.16.8)
-- Found toolchain: zephyr 0.16.8 (C:/Users/C75166/zephyr-sdk-0.16.8)
-- Found Dtc: C:/ProgramData/chocolatey/bin/dtc.exe (found suitable version "1.5.0", minimum required is "1.4.6")
-- Found BOARD.dts: C:/Users/C75166/zephyrproject/zephyr/boards/atmel/sam0/same54_xpro/same54_xpro.dts
-- Generated zephyr.dts: C:/Users/C75166/zephyrproject/build/zephyr/zephyr.dts
```

```
-- Build files have been written to: C:/Users/C75166/zephyrproject/build
←[92m-- west build: building application
[1/117] Generating include/generated/zephyr/version.h
-- Zephyr version: 4.0.0-rc1 (C:/Users/C75166/zephyrproject/zephyr), build: v4.0.0-rc1-70-g5dc5fa5ee2ac
[117/117] Linking C executable zephyr\zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:       14812 B         1 MB      1.41%
             RAM:        4352 B       256 KB      1.66%
        IDT_LIST:           0 GB        32 KB      0.00%
Generating files from C:/Users/C75166/zephyrproject/build/zephyr/zephyr.elf for board: same54_xpro
```

SUCCESS!

MICROCHIP

Lab 1: Summary

- In this lab, you learned how to navigate the Zephyr OS project tree, build and flash sample code, and build a skeleton framework for a new project.
- Your project structure may look different if you choose, this is just one way to lay out a project

Zephyr OS
Kernel Services

Zephyr OS Kernel Services

- **Scheduling Services**
  - Threads, Idling, Interrupts, Semaphores, Mutexes, Events
- **Data Passing Services**
  - FIFO/LIFO, Message Queue, Mailbox, Pipe
- **Memory Management**
  - Heaps, Slabs, Blocks Allocator, Demand Paging, Virtual Memory
- **Timing**
  - Kernel Timing, Timers
- **Other**
  - Atomic Variables, Floating Point, Fatal Errors

MICROCHIP

Using Zephyr OS Threads

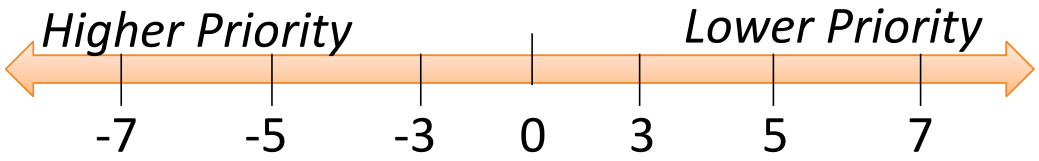
- **What is a Thread?**
  - "A *thread* is a kernel object that is used for application processing that is too lengthy or too complex to be performed by an ISR."

    Any number of threads can be defined by an application (limited only by available RAM). Each thread is referenced by a *thread id* that is assigned when the thread is spawned."
- **Initializing a thread:**

```
#define MY_STACK_SIZE 500
#define MY_PRIORITY 5

extern void my_entry_point(void *, void *, void *);

K_THREAD_STACK_DEFINE(my_stack_area, MY_STACK_SIZE);
struct k_thread my_thread_data;

k_tid_t my_tid = k_thread_create(&my_thread_data, my_stack_area,
                                 K_THREAD_STACK_SIZEOF(my_stack_area),
                                 my_entry_point,
                                 NULL, NULL, NULL,
                                 MY_PRIORITY, 0, K_NO_WAIT);
```

Using Zephyr OS Threads (continued)

- **Using Priority**
  - Integers, lower numbers are higher priority
  - *Avoid Priority Inversion!!!*



- **Passing in UserData (why?)**
  - `void my_entry_point(void *, void *, void *);`
- **RTOS Aware Delays/Yields**
  - `k_msleep();` `k_usleep();` `k_yield();`
- **Gracefully Ending a Task**
  - Release any shared resources (the scheduler won't do it for you)
  - `k_thread_abort();`

MICROCHIP

Communicating between tasks

- **Counting Semaphore**
  - Allows threads to “borrow” and “return” (take/give) a certain number of a specified resource
  - **count** indicates the number of times a semaphore is available to be taken
  - A **limit** indicates the maximum value of semaphore’s **count**
- **Message Queue**
  - Allows threads and ISR’s to send/receive fixed length items asynchronously
- **Mutex**
  - Allow threads to safely share access to a resource, restricting usage to one at a time
- **Event**
  - Inter-thread communication indicating something has occurred (timer, button press, analysis complete, shutdown, FAULT)
  - Many threads can listen for a given event

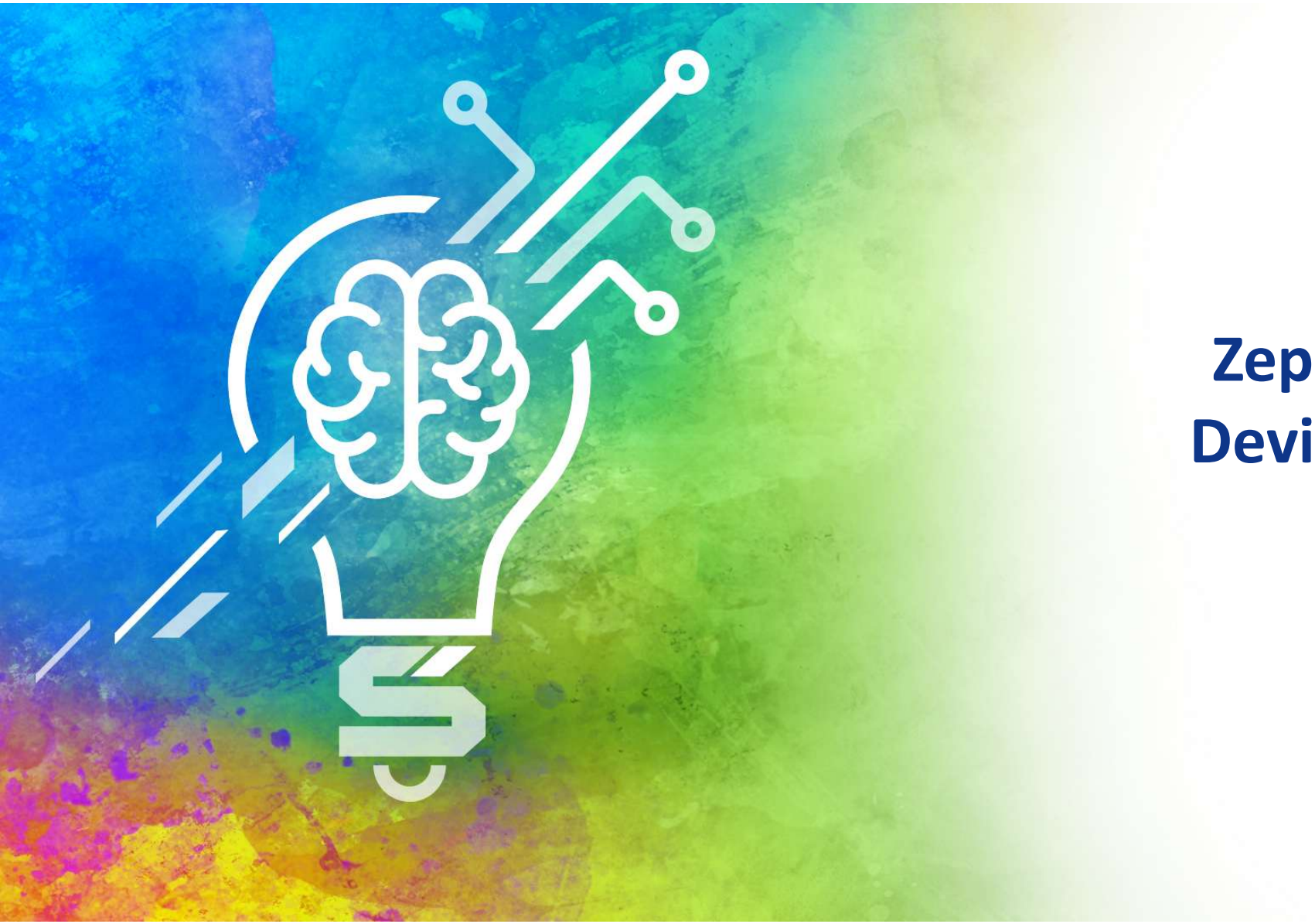
MICROCHIP

Lab 2: Creating Zephyr OS Tasks and Queues

- In this lab, you will create a producer task that delivers a count to a message queue and a consumer task that prints the count when it’s received
- Communications between tasks done via Message Queue

Lab 2: Summary

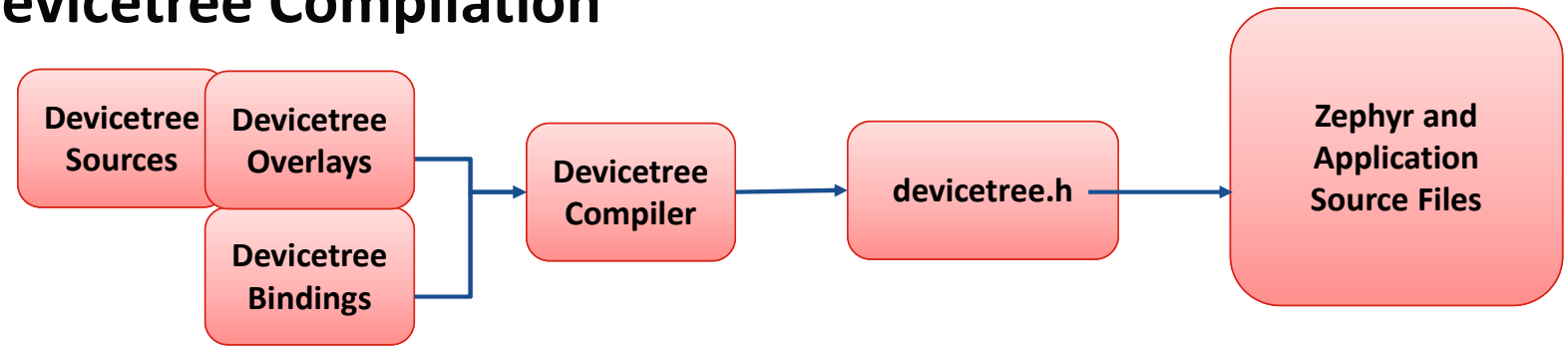
- In this lab, you created new project source (.c) and header (.h) files, updated your CMakeLists.txt file to include these new files, and built two tasks of similar priority with a Message Queue to communicate data between them.
- How else might you use one or more threads in your own projects to replace the cooperative multitasking loop?
- As you move forward using threads, try to become more “Event Based” rather than polled.



Zephyr OS
DeviceTree

Zephyr OS Devicetree

- **What is Devicetree?**
  - Used to describe the presence of hardware and how that hardware should be configured
  - Hierarchical – lower layers can be overlayed with more definitive descriptions
  - Allows for software to target multiple boards by calling “aliases” defines for each board
- **Devicetree Compilation**



MICROCHIP

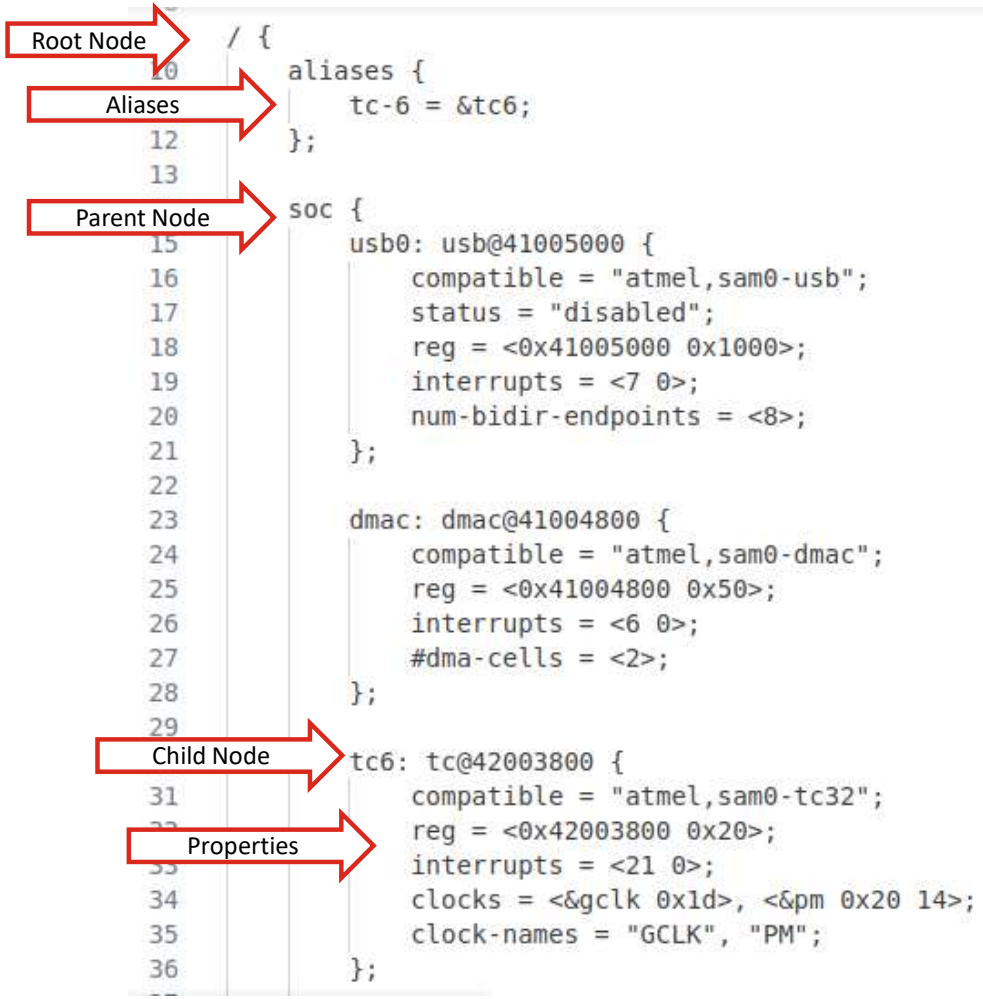
Zephyr OS - Parts of a Devicetree Source

- **Node – an object in a Devicetree source (.dtsi)**
  - / (root node) – collection of sub-nodes
  - Nested nodes: ex: /soc/sercom3/spiflash@0
- **Properties**
  - Describe the configuration of the current node
  - Properties can be integers, strings, phandles/arrays, booleans

  ex:

```
status= “okay”;
compatible=“atmel,sam0-spi”;
dpio = <0>;
cs-gpios=<&porta 13 GPIO_ACTIVE_LOW>, <&portb 6 GPIO_ACTIVE_LOW>;
requires-ulbp;
```

Zephyr OS – Sample Device Tree Source

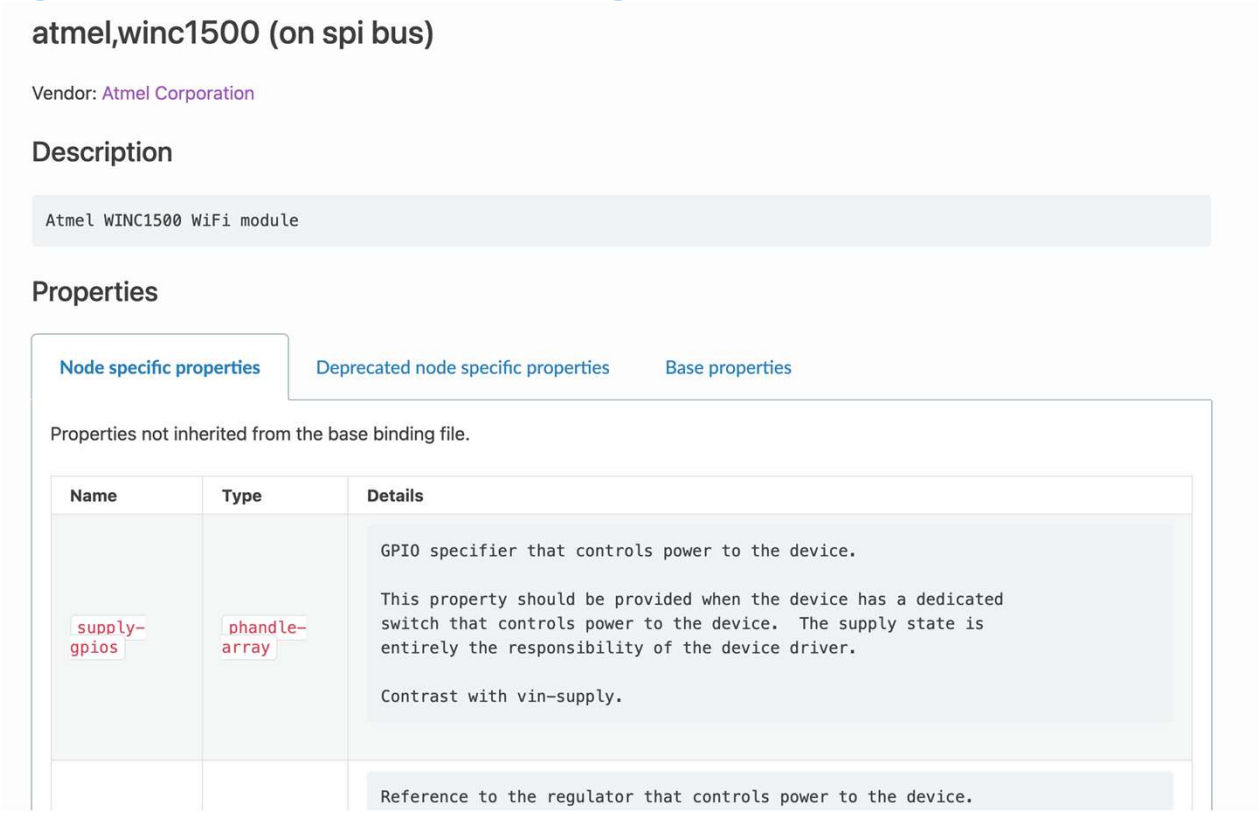



```
/ {
    aliases {
        tc-6 = &tc6;
    };

    soc {
        usb0: usb@41005000 {
            compatible = "atmel,sam0-usb";
            status = "disabled";
            reg = <0x41005000 0x1000>;
            interrupts = <7 0>;
            num-bidir-endpoints = <8>;
        };

        dmac: dmac@41004800 {
            compatible = "atmel,sam0-dmac";
            reg = <0x41004800 0x50>;
            interrupts = <6 0>;
            #dma-cells = <2>;
        };

        tc6: tc@42003800 {
            compatible = "atmel,sam0-tc32";
            reg = <0x42003800 0x20>;
            interrupts = <21 0>;
            clocks = <&gclk 0x1d>, <&pm 0x20 14>;
            clock-names = "GCLK", "PM";
        };
```


MICROCHIP

Zephyr OS Devicetree

Demo: Finding a Devicetree Binding API



atmel,winc1500 (on spi bus)

Vendor: Atmel Corporation

Description

Atmel WINC1500 WiFi module

Properties

Node specific properties | Deprecated node specific properties | Base properties

Properties not inherited from the base binding file.

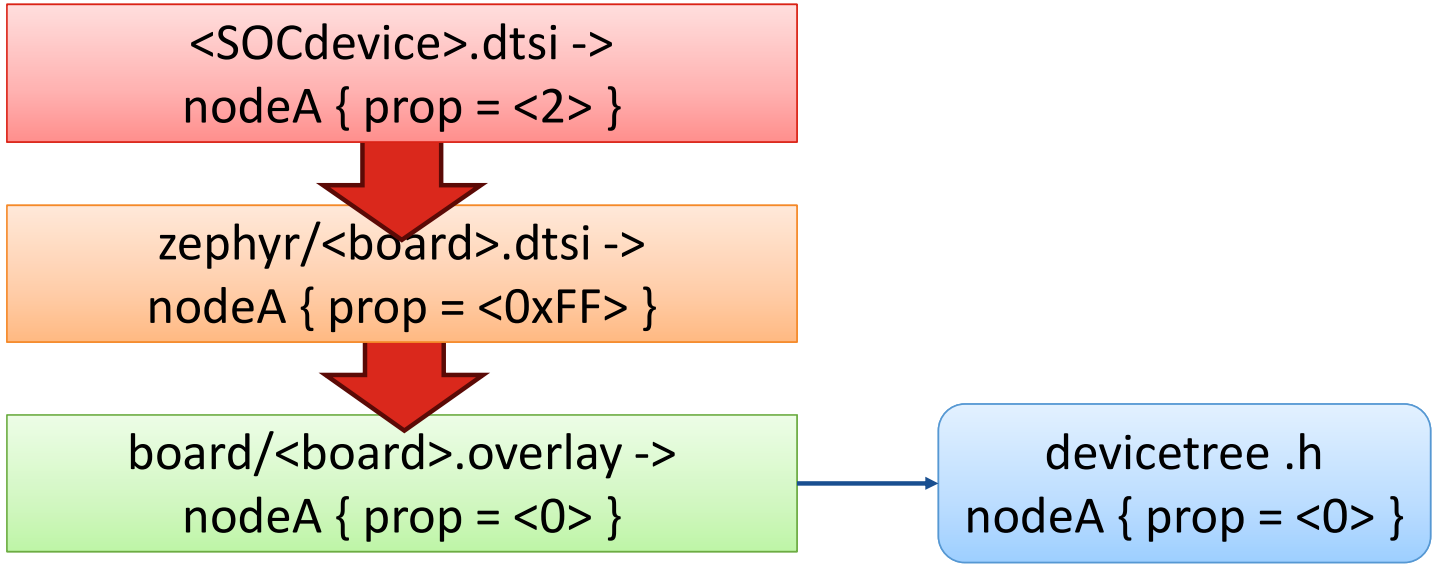
Name	Type	Details
supply-gpios	phandle-array	GPIO specifier that controls power to the device. This property should be provided when the device has a dedicated switch that controls power to the device. The supply state is entirely the responsibility of the device driver. Contrast with vin-supply.
		Reference to the regulator that controls power to the device.

- https://docs.zephyrproject.org/latest/build/dts/api/bindings.html

MICROCHIP

Zephyr OS Devicetree

- **What is an Overlay?**
- **Using an Overlay to extend or change a Device Configuration**



MICROCHIP

Zephyr OS - Using a Devicetree device in your code

- **Get a node identifier**

```
/* Option 1: by node label */
#define MY_SERIAL DT_NODELABEL(serial0)

/* Option 2: by alias */
#define MY_SERIAL DT_ALIAS(my_serial)

/* Option 3: by chosen node */
#define MY_SERIAL DT_CHOSEN(zephyr_console)

/* Option 4: by path */
#define MY_SERIAL DT_PATH(soc, serial_40002000)
```

- **Get a device handle**

```
const struct device *const uart_dev = DEVICE_DT_GET(MY_SERIAL);

if (!device_is_ready(uart_dev))
{
     /* Not ready, do not use */
     return -ENODEV;
}
```


MICROCHIP

Zephyr OS – Configuring prj.conf

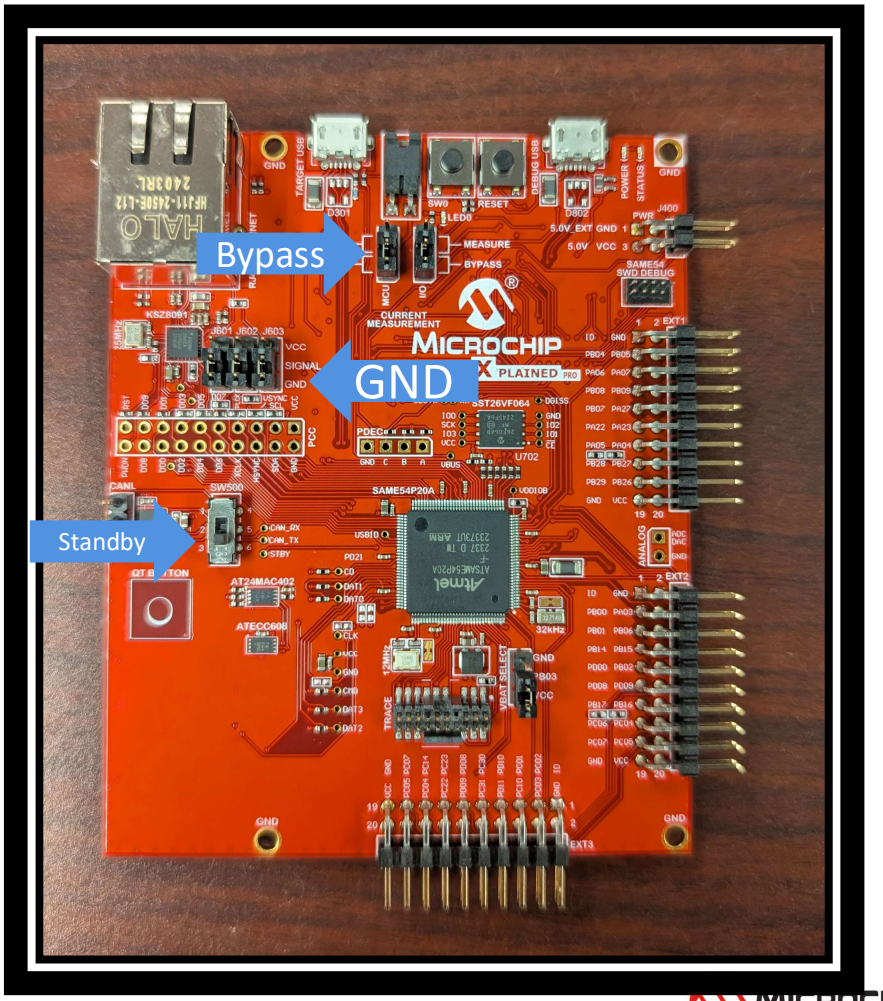
- **prj.conf holds a list of configuration defines**
  - CONFIG_GPIO=y
  - CONFIG_NET_TX_STACK_SIZE=2048
- **Compile Time Flags that Enable/Disable/Configure content**
- **Kconfig Search**
  - Kconfig Search — Zephyr Project Documentation
- **Editing prj.conf**
  - Text edit – prj.conf
  - west build -t menuedit – uses menuconfig
  - west build -t guiedit – opens a GUI window

MICROCHIP

Lab 3: Adding Wifi with Devicetree

- Use the “atmel,winc1500” devicetree binding to describe the configuration of the WINC1500 chip on EXT1

- Once connected, engineer will use wifi console application to scan area for available wifi, connect to an Access Point, and send a UDP packet to a listener on the network



Lab 3: Summary

- **Did you notice a problem with the Reset Line in our Devicetree Overlay?**

- **The console operation is managed in wifi_shell.c and net_shell.c. These serve as great code examples as you move from debug/test into programmatic networking products**

OS Services and Finding Help

Zephyr OS Services

- **Many more built in “goodies” to make development more efficient:**
  - Console/Shell
  - File Systems
  - State Machine Framework
  - Power Manager
  - Logging System
  - Task Watchdog
  - Modem (GNSS, Wifi, Cellular)
  - Unit Testing Framework (twister) ->

Zephyr OS – Unit Testing (twister)

- **Twister is the Test Suite for Unit tests**
  - Testing should be focused on “library” functions
  - NOT application code (no thread tests!)
- **Each TestSuite has CMakeLists.txt, prj.com, main.c, and testcase.yaml**
  - main.c holds test_main() (or associated macros)
  - #include any headers with code you intend to test
- **Update CMakeLists.txt and prj.conf to point to specific code to be tested**
  - Don’t include your applications main.c in CMakeLists.txt
- **Add test cases to tests/src/main.c (or similar)**
  - zassert_<type>(arguments);

MICROCHIP

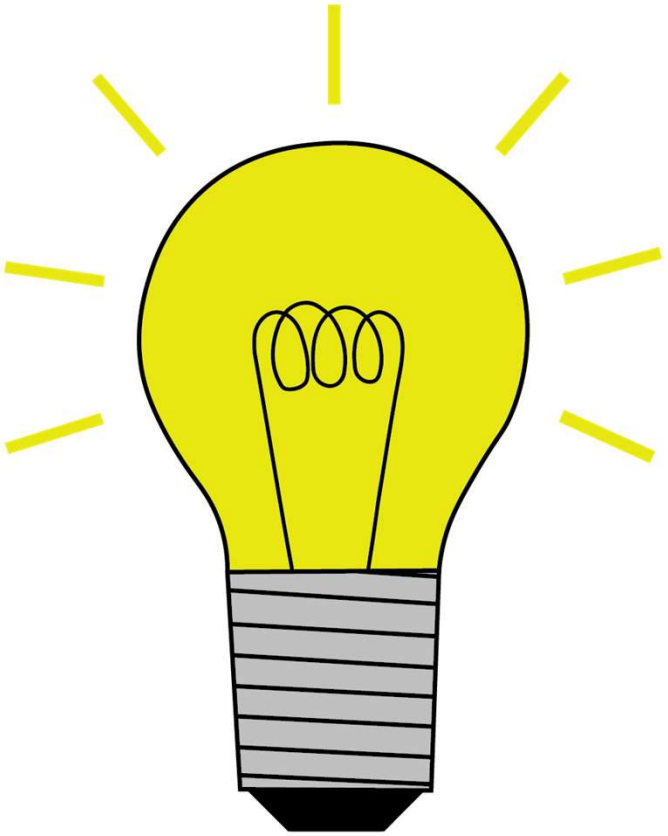
Zephyr OS – Unit Testing (continued)

- **Running**
  - python
  - same54
  - Can als
- **Results**
  - twister
- **View de**

```
*** Booting Zephyr OS build v4.0.0-rc1-70-g5dc5fa5ee2ac ***
Running TESTSUITE framework_tests
===================================================================
START - test_assert
 PASS - test_assert in 0.001 seconds
===================================================================
TESTSUITE framework_tests succeeded
Running TESTSUITE producer_tests
===================================================================
START - test_assert
 PASS - test_assert in 0.001 seconds
===================================================================
TESTSUITE producer_tests succeeded

------ TESTSUITE SUMMARY START ------

SUITE PASS - 100.00% [framework_tests]: pass = 1, fail = 0, skip = 0, total = 1 duration = 0.001 seconds
 - PASS - [framework_tests.test_assert] duration = 0.001 seconds

SUITE PASS - 100.00% [producer_tests]: pass = 1, fail = 0, skip = 0, total = 1 duration = 0.001 seconds
 - PASS - [producer_tests.test_assert] duration = 0.001 seconds

------ TESTSUITE SUMMARY END ------

===================================================================
RunID: a5c66babaf0414561e8bec0b9163b793
PROJECT EXECUTION SUCCESSFUL
```


MICROCHIP

Getting Support



- Official Zephyr OS Docs
- Discord Server
- Mailing Lists
- Wiki
- YouTube Training
- Social Media
- Me (or your MCHP ESE/CAE)

Thank You!

MICROCHIP