Project Configurator & Gecko Platform - 20Q4 FAE Training

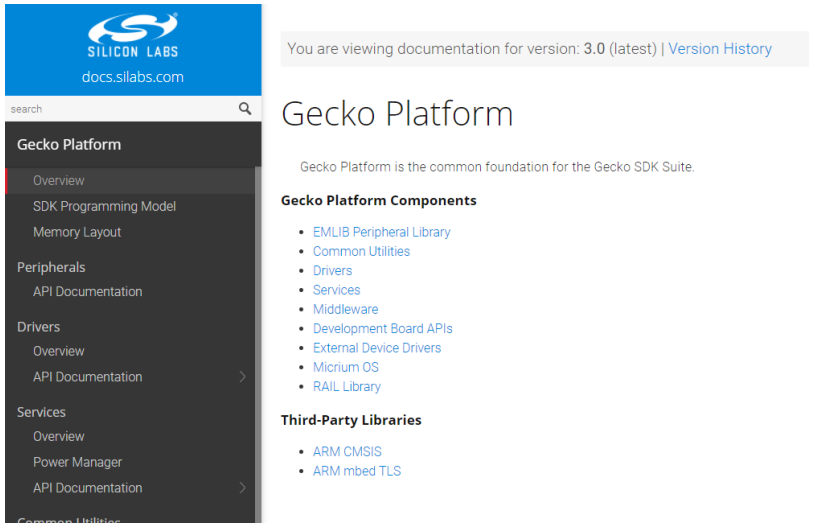
PLATFORM AND MCU APPS | NOVEMBER 2020

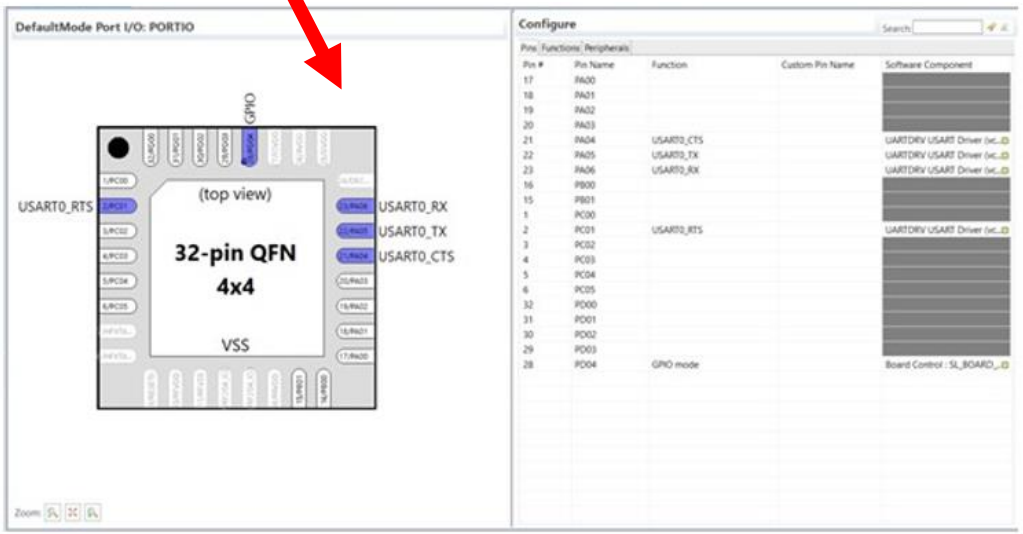
Introduction and Agenda

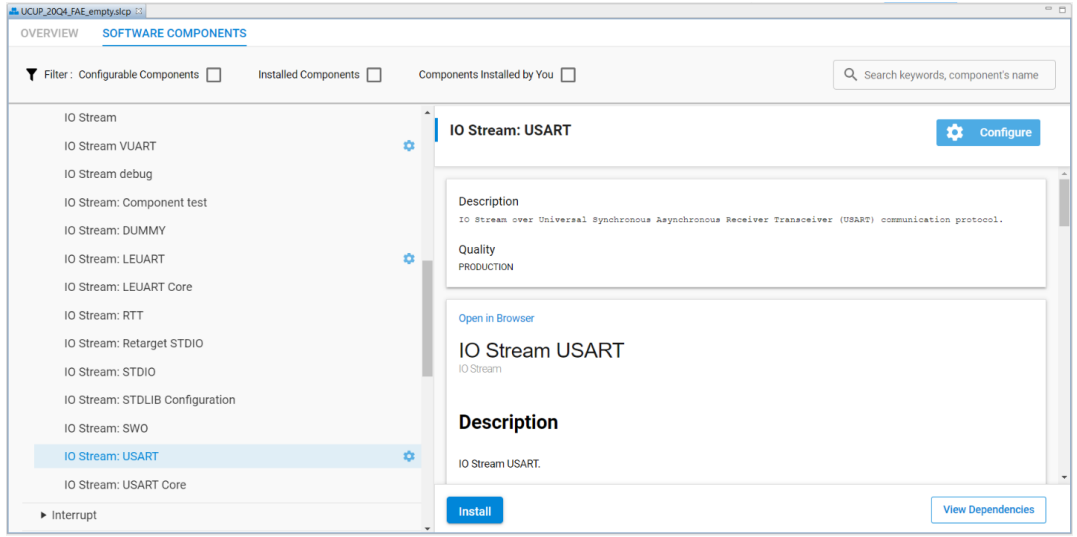
- Definition of Project Configurator and Gecko Platform
- Silicon Labs Software Architecture Overview and Context
- Project Configurator/Gecko Platform Vocabulary and Terms
- Project Configurator Overview and GUI Walkthrough
- Gecko Platform Overview
- Gecko Platform Programming Model
- Platform Services Overview
- Platform Services API Walkthrough
- Special Considerations

Project Configurator and Gecko Platform Definition

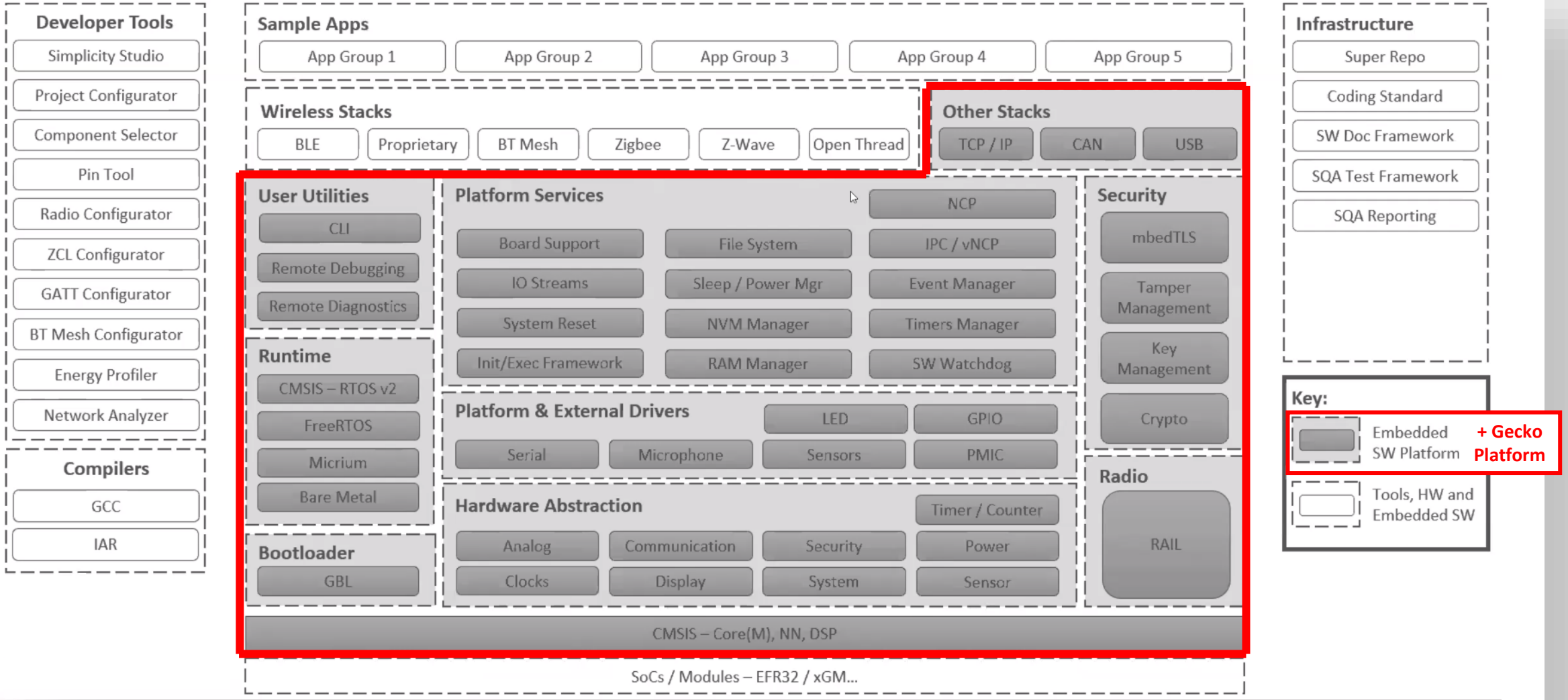
- Development Architecture:
  - Gecko Platform
    - APIs for use across range of Silicon Labs devices and stacks
  - Project Configurator
    - GUI Tool for management of software platform modules
    - Pin Tool







Software Architecture Overview

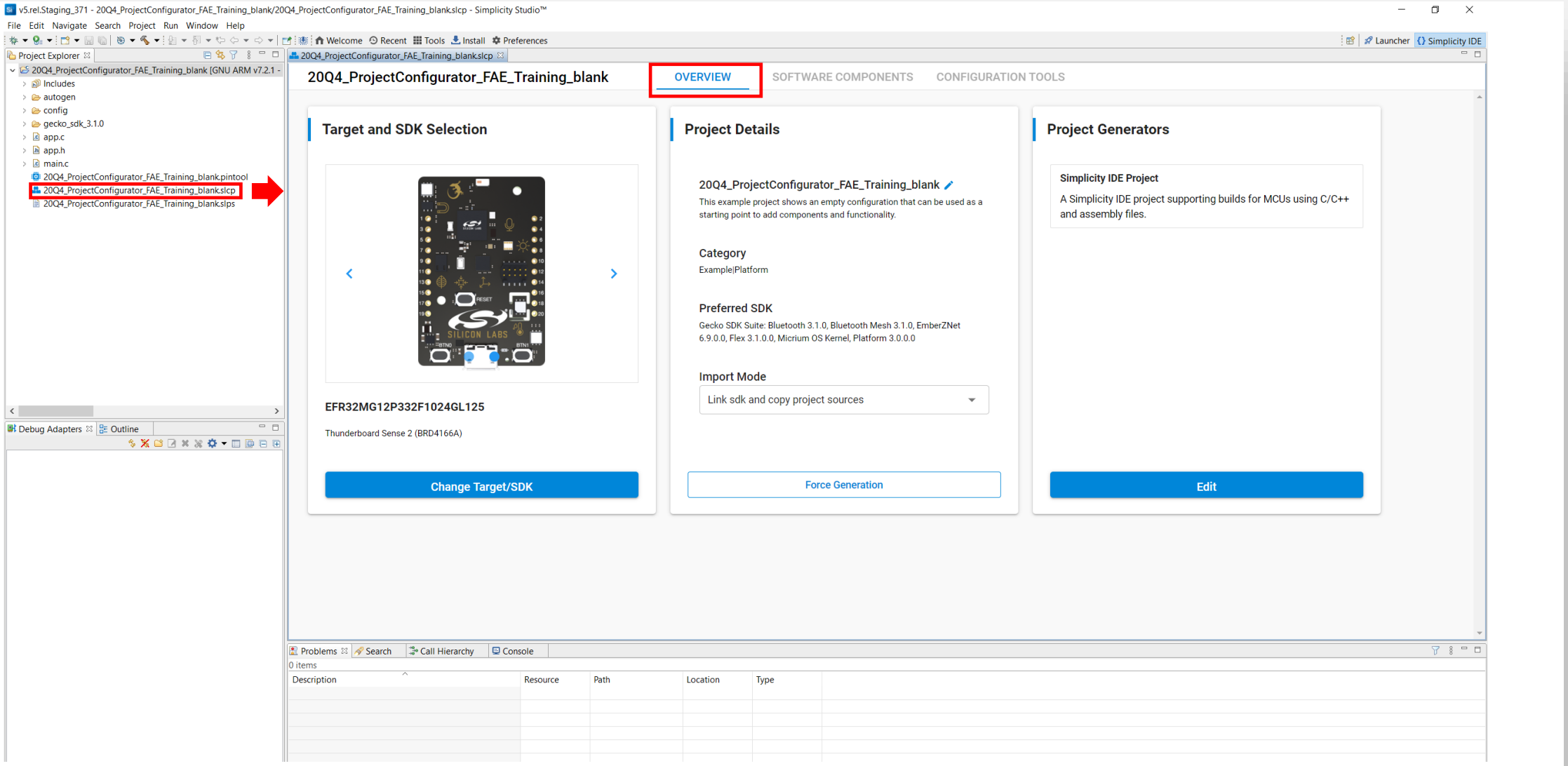


Project Configurator and Gecko Platform Vocabulary

- Project Configurator – A GUI tool to install and uninstall project components
- Component Editor – GUI element used to configure component parameters
- Pin Tool – GUI element used to configure device peripheral and hardware assignments
- Bluetooth GATT Configurator – GUI element used to customize Bluetooth GATT database
- Radio Configurator – GUI element used to define and manage proprietary radio protocols and channel groups
- Software Component – modules that can be installed via Project Configurator, including stacks, RTOS, services, drivers, etc.
- Instance – A unique copy of a software component; supported by only some components
- <project>.slcp – A component-based project file
- <component>.slcc – a component definition file

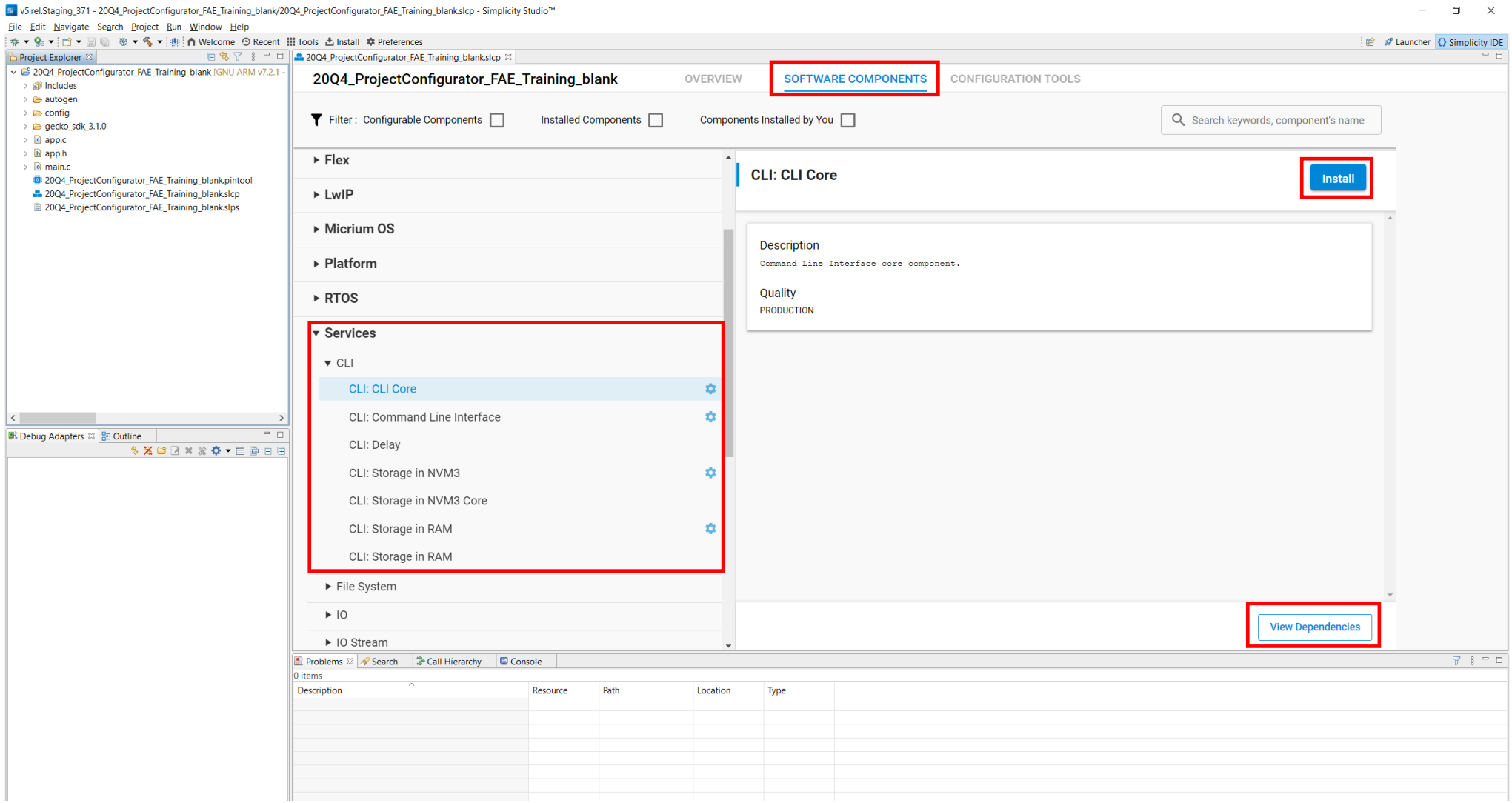
Project Configurator

Project Configurator – OVERVIEW Tab



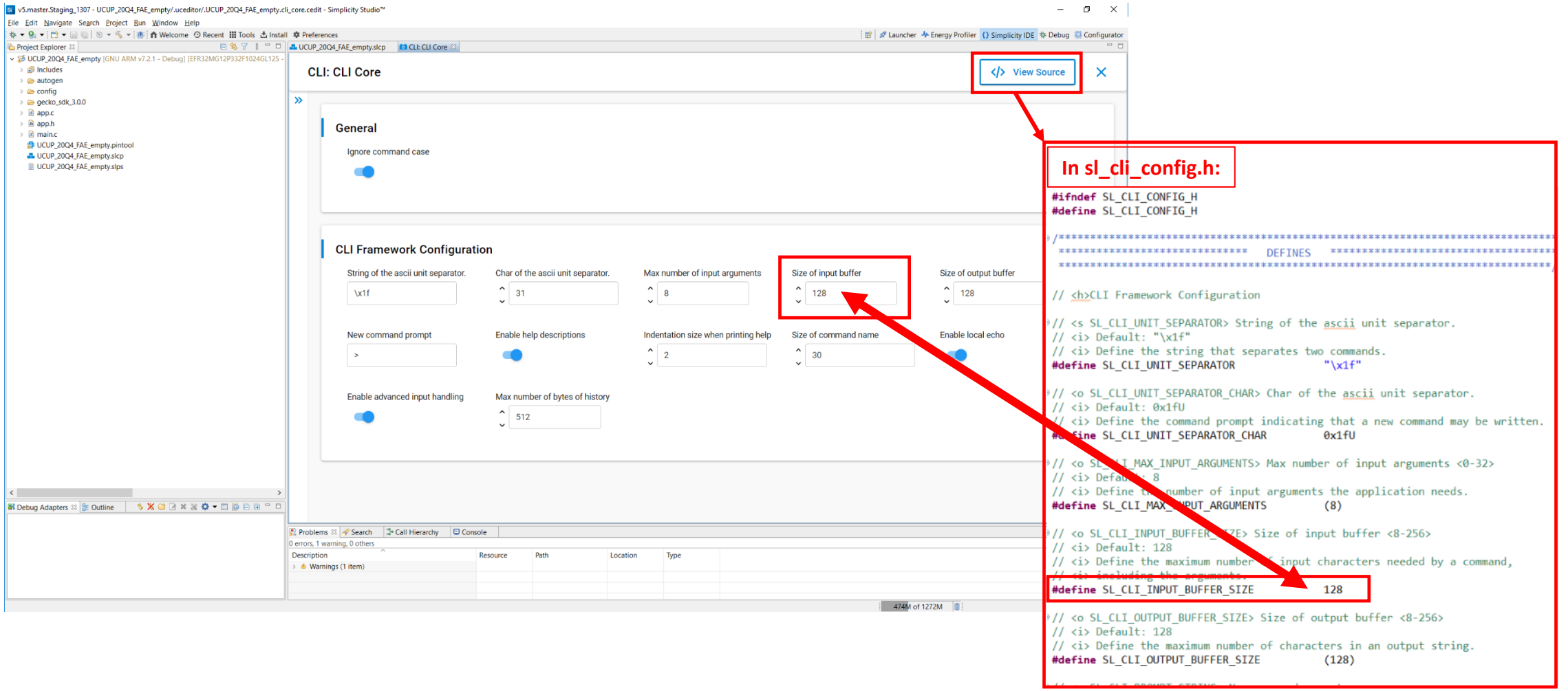
- https://docs.silabs.com/simplicity-studio-5-users-guide/1.0/developing-for-32bit-devices/developing-with-project-configurator/

Project Configurator – SOFTWARE COMPONENTS Tab



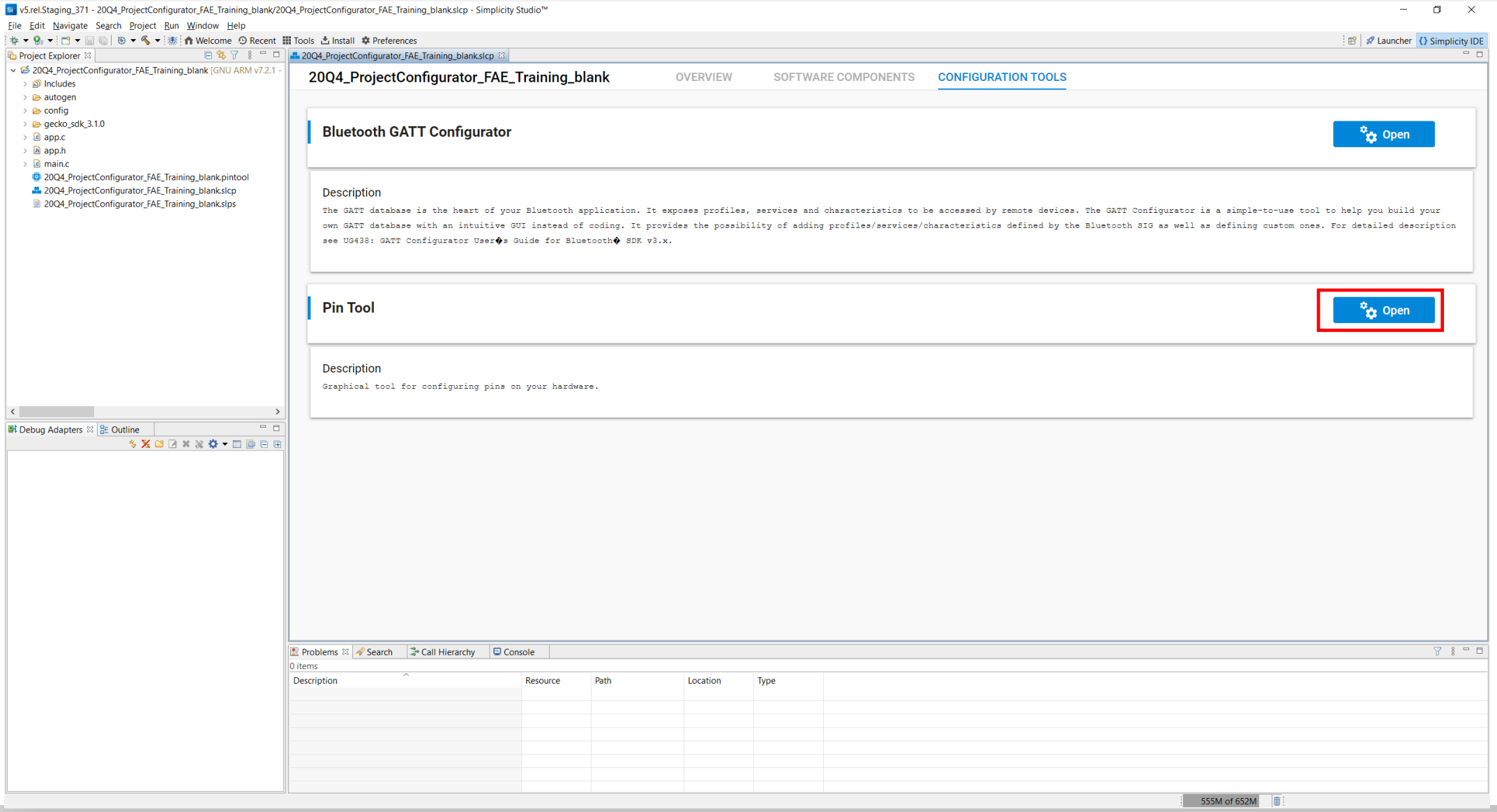
- https://docs.silabs.com/simplicity-studio-5-users-guide/1.0/developing-for-32bit-devices/developing-with-project-configurator/#software-components-tab

Project Configurator – Component Editor

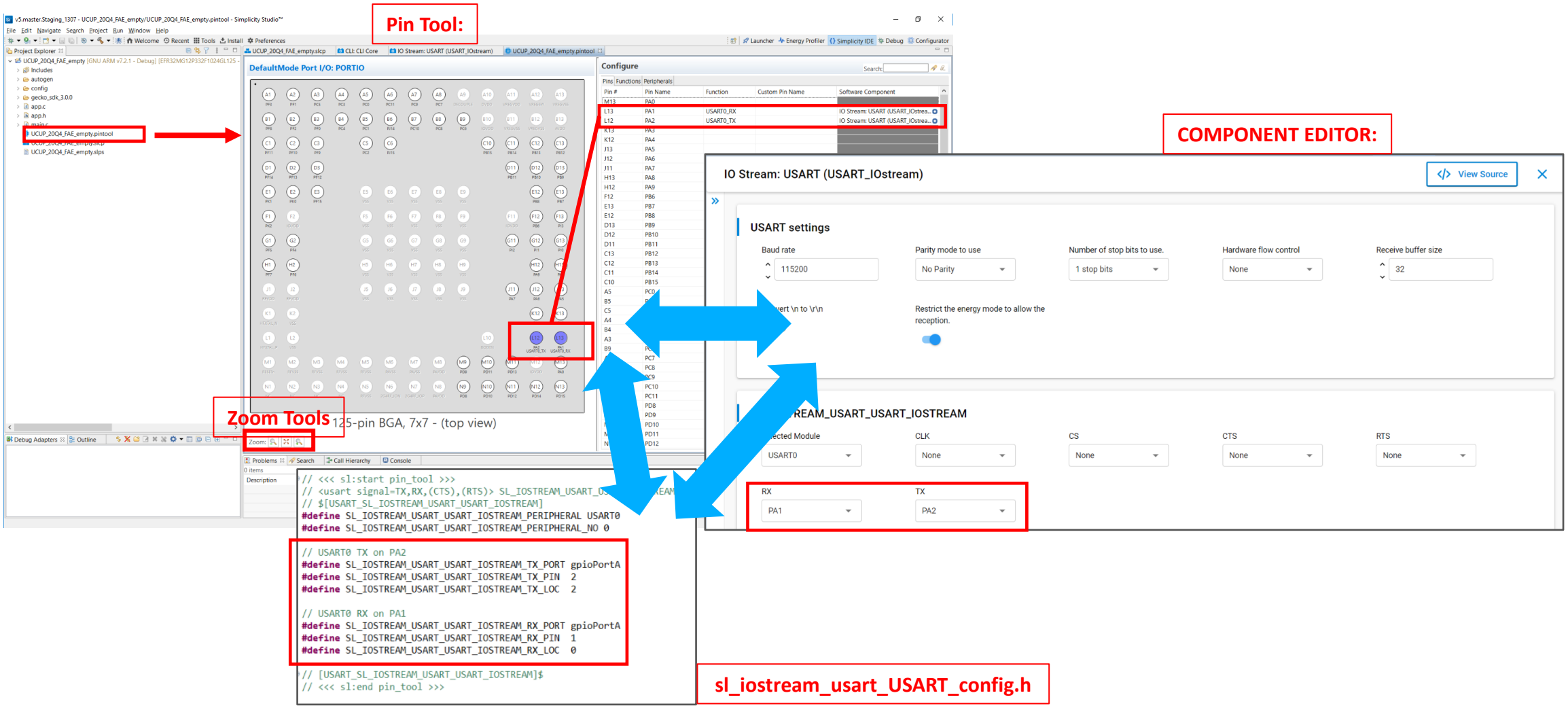


- https://docs.silabs.com/simplicity-studio-5-users-guide/1.0/developing-for-32bit-devices/developing-with-project-configurator/#component-editor

Project Configurator – Configuration Tools



Project Configurator – Pin Tool



- https://docs.silabs.com/simplicity-studio-5-users-guide/1.0/developing-for-32bit-devices/developing-with-project-configurator/pin-tool/

Gecko Platform

https://docs.silabs.com/gecko-platform/latest/index

Gecko Platform

Gecko Platform is the common foundation for the Gecko SDK Suite.

Gecko Platform Components

- EMLIB Peripheral Library
- Common Utilities
- Drivers
- Services
- Middleware
- Development Board APIs
- External Device Drivers
- Micrium OS
- RAIL Library

Third-Party Libraries

- ARM CMSIS
- ARM mbed TLS

Programming Model Overview

- New programming model with Simplicity Studio v5
- Creates a project structure to facilitate easy addition and removal of software modules
- Core concepts:
  - Hardware Initialization
  - Software Initialization
  - Periodic Action Processing
- Project structure provides insertion points for autogenerated and application (custom) code
- Current wireless support: Bluetooth and Proprietary SDKs only
- Supports bare metal and kernel (RTOS) application
  - Easy to switch between the two
- Integrated power management (using Power Manager)
- Documentation: https://docs.silabs.com/gecko-platform/latest/sdk-programming-model

Programming Model Structure – Bare metal

In main.c:

```c
int main(void)
{
  // Initialize Silicon Labs device, system, service(s) and protocol stack(s).
  // Note that if the kernel is present, processing task(s) will be created by
  // this call.
  sl_system_init();

  // Initialize the application. For example, create periodic timer(s) or
  // task(s) if the kernel is present.
  app_init();

#if defined(SL_CATALOG_KERNEL_PRESENT)
  // Start the kernel. Task(s) created in app_init() will start running.
  sl_system_kernel_start();
#else // SL_CATALOG_KERNEL_PRESENT
  while (1) {
    // Do not remove this call: Silicon Labs components process action routine
    // must be called from the super loop.
    sl_system_process_action();

    // Application process.
    app_process_action();

#if defined(SL_CATALOG_POWER_MANAGER_PRESENT)
    // Let the CPU go to sleep if the system allows it.
    sl_power_manager_sleep();
#endif
  }
#endif // SL_CATALOG_KERNEL_PRESENT
}
```

In sl_system_init.c:

```c
void sl_system_init(void)
{
  sl_platform_init();
  sl_driver_init();
  sl_service_init();
  sl_stack_init();
  sl_internal_app_init();
}
```

- Initializes autogenerated (Gecko Platform) software components

In sl_system_process_action.c:

```c
void sl_system_process_action(void)
{
  sl_platform_process_action();
  sl_service_process_action();
  sl_stack_process_action();
  sl_internal_app_process_action();
}
```

- Processes autogenerated (Gecko Platform) firmware actions

In app.c:

```c
/***************************************************************************//**
 * Initialize application.
 ******************************************************************************/
void app_init(void)
{
  // Place application initialization code here
}

/***************************************************************************//**
 * App ticking function.
 ******************************************************************************/
void app_process_action(void)
{
  // Place application loop processing code here
}
```

- Initializes user-added (custom) firmware elements
- Processes user-added (custom) firmware actions

Programming Model Structure – Kernel

In main.c:

```
int main(void)
{
  // Initialize Silicon Labs device, system, service(s) and protocol stack(s).
  // Note that if the kernel is present, processing task(s) will be created by
  // this call.
  sl_system_init();

  // Initialize the application. For example, create periodic timer(s) or
  // task(s) if the kernel is present.
  app_init();

#if defined(SL_CATALOG_KERNEL_PRESENT)
  // Start the kernel. Task(s) created in app_init() will start running.
  sl_system_kernel_start();
#else // SL_CATALOG_KERNEL_PRESENT
  while (1) {
    // Do not remove this call: Silicon Labs components process action routine
    // must be called from the super loop.
    sl_system_process_action();

    // Application process.
    app_process_action();

#if defined(SL_CATALOG_POWER_MANAGER_PRESENT)
    // Let the CPU go to sleep if the system allows it.
    sl_power_manager_sleep();
#endif
  }
#endif // SL_CATALOG_KERNEL_PRESENT
}
```

In sl_system_init.c:

```
void sl_system_init(void)
{
  sl_platform_init();
  sl_driver_init();
  sl_service_init();
  sl_stack_init();
  sl_internal_app_init();
}
```

In app.c:

```
/***************************************************************************//**
 * Initialize application.
 ******************************************************************************/
void app_init(void)
{
  // Place application initialization code here
}
```

- Starts the kernel running
- Processing of system and app actions handled by tasks:
  - Tasks created in sl_system_init() and app_init() will execute according to kernel scheduler

Gecko Platform Software Modules

- Gecko Platform is composed of a broad range of software modules
- High Level APIs
  - RAIL, Micrium, Drivers
- Low Level APIs
  - Common Utilities, EMLIB

- https://docs.silabs.com/gecko-platform/latest/index

Gecko Platform

Gecko Platform is the common foundation for the Gecko SDK Suite.

Gecko Platform Components

- EMLIB Peripheral Library
- Common Utilities
- Drivers
- Services
- Middleware
- Development Board APIs
- External Device Drivers
- Micrium OS
- RAIL Library

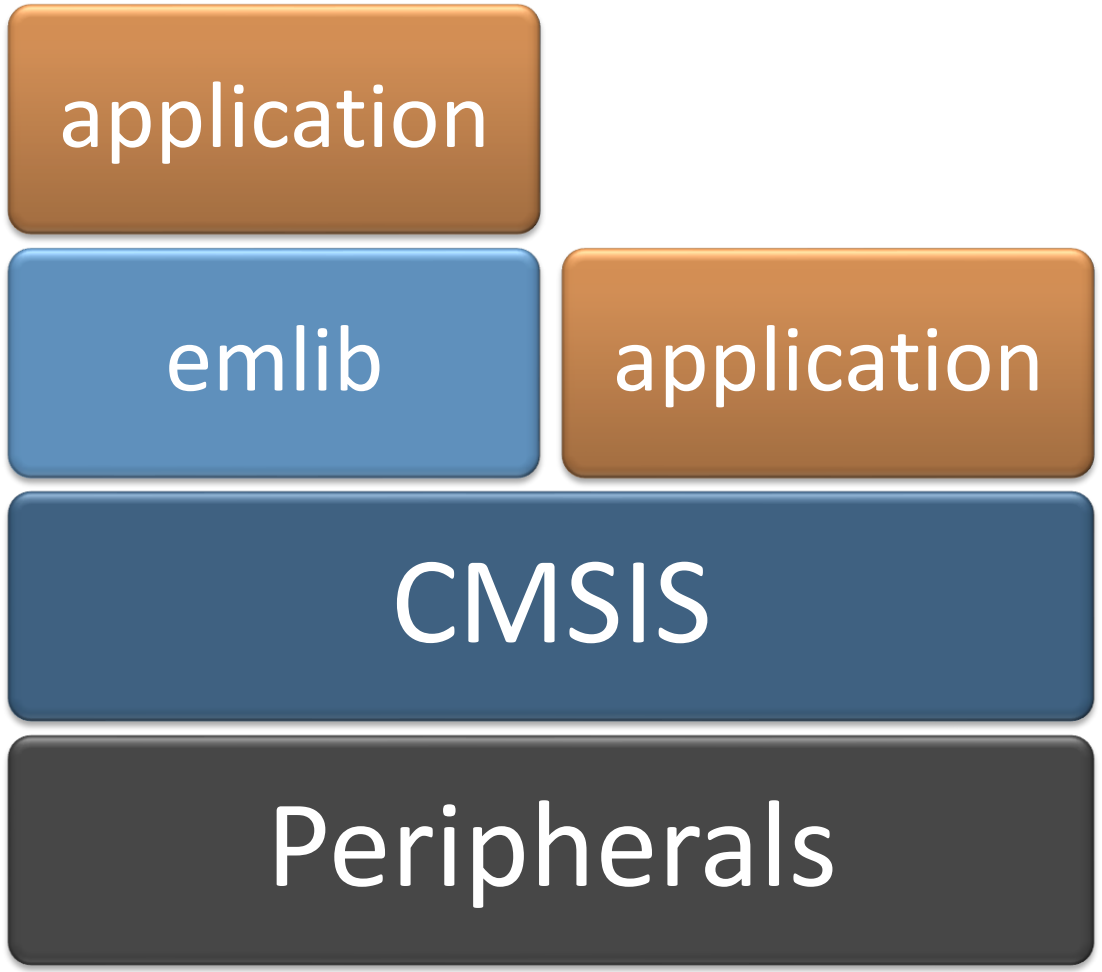
Third-Party Libraries

- ARM CMSIS
- ARM mbed TLS

Gecko Platform – EMLIB Peripheral Library

- Low-level peripheral support library
  - Abstracts direct register manipulation
- Provides unified API for all Silicon Labs 32-bit MCU and Wireless MCUs (EFM32, EZR32, EFR32)
  - API is device specific – see documentation
  - Provides extensive API coverage for most peripherals and device features
- Implements Datasheet and Errata requirements and workarounds
  - Enforces device-specific configuration requirements

- https://docs.silabs.com/gecko-platform/latest/emlib/api
- https://github.com/SiliconLabs/peripheral_examples



Gecko Platform – Common Utilities

Utility Module	Description	API Documentation link
Atomic Operations	Atomic operations	https://docs.silabs.com/gecko-platform/latest/common/api/group-atomic
Enumerations	Enumerations with stable binary representation	https://docs.silabs.com/gecko-platform/latest/common/api/group-enum
Linker	Functions to extract locations of linker sections	https://docs.silabs.com/gecko-platform/latest/common/api/group-linker
Singly-Linked List	Singly-linked list	https://docs.silabs.com/gecko-platform/latest/common/api/group-slist
Standard I/O	Standard I/O	https://docs.silabs.com/gecko-platform/latest/common/api/group-stdio
Status Codes	Status codes	https://docs.silabs.com/gecko-platform/latest/common/api/group-status
String	String functions	https://docs.silabs.com/gecko-platform/latest/common/api/group-string

Gecko Platform – Drivers

Driver	Description	API Documentation link
Button API	Generic Button API (Simple Button Driver)	https://docs.silabs.com/gecko-platform/latest/driver/api/group-button
DMADRV	Direct Memory Access Driver	https://docs.silabs.com/gecko-platform/latest/driver/api/group-dmadrv
ECODE	Error and Status Codes	https://docs.silabs.com/gecko-platform/latest/driver/api/group-ecode
GPIOINT	GPIOINT General Purpose Input/Output Interrupt dispatcher	https://docs.silabs.com/gecko-platform/latest/driver/api/group-gpioint
I2C Simple Polled Master	I2C Simple Polled Master driver	https://docs.silabs.com/gecko-platform/latest/driver/api/group-i2cspm
LED API + Simple RGBW PWM LED Driver	Generic LED API (Simple LED Driver + Simple Red/Green/Blue/White PWM LED Driver)	https://docs.silabs.com/gecko-platform/latest/driver/api/group-led
NVM3	NVM3 Non-Volatile Memory Data Management driver	https://docs.silabs.com/gecko-platform/latest/driver/api/group-nvm3
PWM	PWM driver (supports configurable frequency, duty cycle, and polarity; supports multiple instances)	https://docs.silabs.com/gecko-platform/latest/driver/api/group-pwm
RTCDRV	Real-time Clock Driver (DEPRECATED – Use Sleep Timer Service)	https://docs.silabs.com/gecko-platform/latest/driver/api/group-rtcdrv
SLEEP	Sleep Management Driver (DEPRECATED – Use Sleep Timer Service)	https://docs.silabs.com/gecko-platform/latest/driver/api/group-sleep
SPIDRV	Serial Peripheral Interface Driver	https://docs.silabs.com/gecko-platform/latest/driver/api/group-spidrv
TEMPDRV	Temperature Driver	https://docs.silabs.com/gecko-platform/latest/driver/api/group-tempdrv
UARTDRV	Universal Asynchronous Receiver/Transmitter Driver	https://docs.silabs.com/gecko-platform/latest/driver/api/group-uartdrv
USTIMER	Microsecond Delay Timer Driver	https://docs.silabs.com/gecko-platform/latest/driver/api/group-ustimer

Gecko Platform – Services

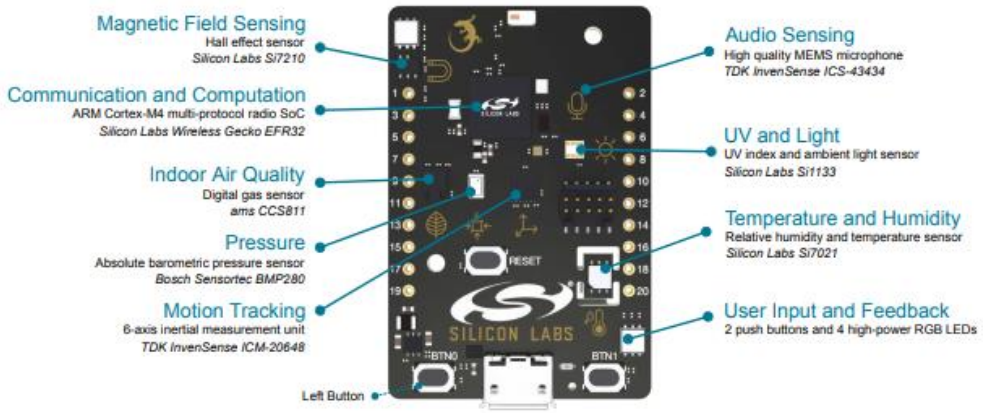
Service	Description	Documentation Link
Power Manager	Manages device energy modes	https://docs.silabs.com/gecko-platform/latest/service/power_manager/overview
CLI	Command Line Interface (CLI) + CLI Storage	https://docs.silabs.com/gecko-platform/latest/service/api/group-cli
Device Initialization	Device Initialization (EMU, CMU)	https://docs.silabs.com/gecko-platform/latest/service/api/group-device-init
IO Stream	Input/Output stream service	https://docs.silabs.com/gecko-platform/latest/service/api/group-iostream
Legacy HAL	Legacy HAL API	https://docs.silabs.com/gecko-platform/latest/service/api/group-legacyhal
Simple MPU	Simple utilities to disable execution on certain memory regions	https://docs.silabs.com/gecko-platform/latest/service/api/group-mpu
Microsecond Delay	Microsecond delay function	https://docs.silabs.com/gecko-platform/latest/service/api/group-udelay
Secure Element Manager	Provides thread-safe APIs for the Secure Element's mailbox interface	https://docs.silabs.com/gecko-platform/latest/service/api/group-sl-se-manager
Sleep Timer	Provides software timers, delays, timekeeping and date functionalities using a low-frequency real-time clock peripheral	https://docs.silabs.com/gecko-platform/latest/service/api/group-sleeptimer
System Initialization and Action Processing	System Initialization and Action Processing	https://docs.silabs.com/gecko-platform/latest/service/api/group-system
Token Manager	Routines for working with tokens	https://docs.silabs.com/gecko-platform/latest/service/api/group-token-manager

Gecko Platform – Middleware

Middleware Module	Description	Documentation Link
CSLIB Capacitive Sensing Library	Capacitive sensing firmware library for Silicon Labs MCUs	https://docs.silabs.com/gecko-platform/latest/middleware/api/group-cslib-group
GLIB - Graphics Library	Silicon Labs Graphics Library, includes DMD - Dot Matrix Display (Hardware abstraction layer for dot matrix displays), GLIB BMP (bitmap support), and GLIB Colors (predefined colors)	https://docs.silabs.com/gecko-platform/latest/middleware/api/group-glib
USB Stacks	Gecko USB stacks, including USB Device (Gecko USB device protocol stack), USB Host (Gecko USB host protocol stack), and USB Common (common parts for both host and device USB stacks)	https://docs.silabs.com/gecko-platform/latest/middleware/api/group-usb
USBXpress	USBXpress interface library	https://docs.silabs.com/gecko-platform/latest/middleware/api/group-usbxpress

Gecko Platform – Development Board APIs

Board API Modules	Description	Documentation Link
Board Control	Functions to control Silicon Labs board features	https://docs.silabs.com/gecko-platform/latest/hardware-board/api/group-board-control
Board Initialization	Initialization of Silicon Labs board features	https://docs.silabs.com/gecko-platform/latest/hardware-board/api/group-board-init
Thunderboard Sense 2 Support	Board support functions for Thunderboard Sense 2 (BRD4166A)	https://docs.silabs.com/gecko-platform/latest/hardware-board/api/group-brd4166a-support

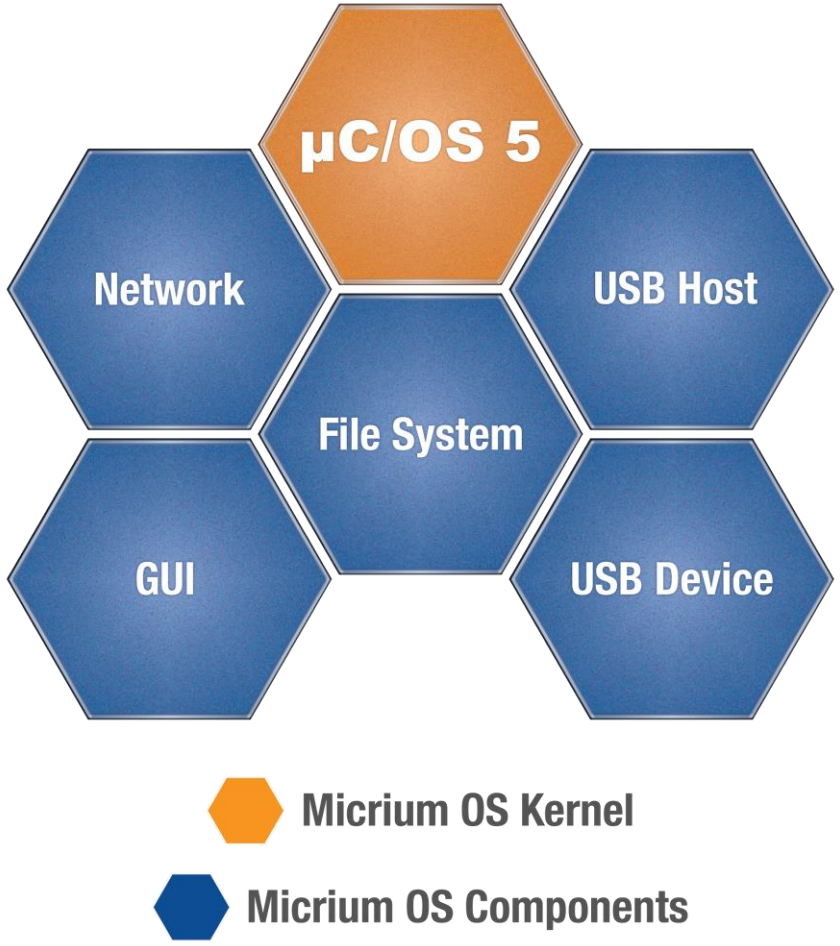


Gecko Platform – External Device Drivers

External Device Driver	Description	Documentation Link
BMP280 - Barometric Pressure Sensor	Driver for the Bosch Sensortec BMP280 barometric pressure sensor	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-bmp280
CCS811 - Gas Sensor	Driver for the Cambridge CMOS Sensors CCS811 gas and indoor air quality sensor	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-ccs811
EFP - Energy Friendly PMIC	EFP (Energy Friendly PMIC) driver	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-sl-efp
ICM20648 - Motion Sensor	Driver for the Invensense ICM20648 6-axis motion sensor	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-icm20648
IMU - Inertial Measurement Unit	Inertial Measurement Unit driver	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-imu
MEMLCD - Memory LCD	Memory LCD interface	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-memlcd
MX25 SPI Flash Shutdown	Provide a function to put the MX25 SPI flash into deep power down mode to reduce power consumption	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-mx25-flash-shutdown
Microphone	Sound level driver for PDM and I2S microphones	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-mic
Si1133 - Light and UV Sensor	Driver for the Silicon Labs Si1133 ambient light and UV sensor	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-si1133
Si70xx - RHT Sensor	Silicon Labs Si7006/13/20/21 Relative Humidity and Temperature Sensor I2C driver	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-si70xx
Si7210 - Magnetic Hall Effect sensor	Driver for the Silicon Labs Si7210 Hall effect sensor	https://docs.silabs.com/gecko-platform/latest/hardware-driver/api/group-si7210

Gecko Platform – Micrium OS

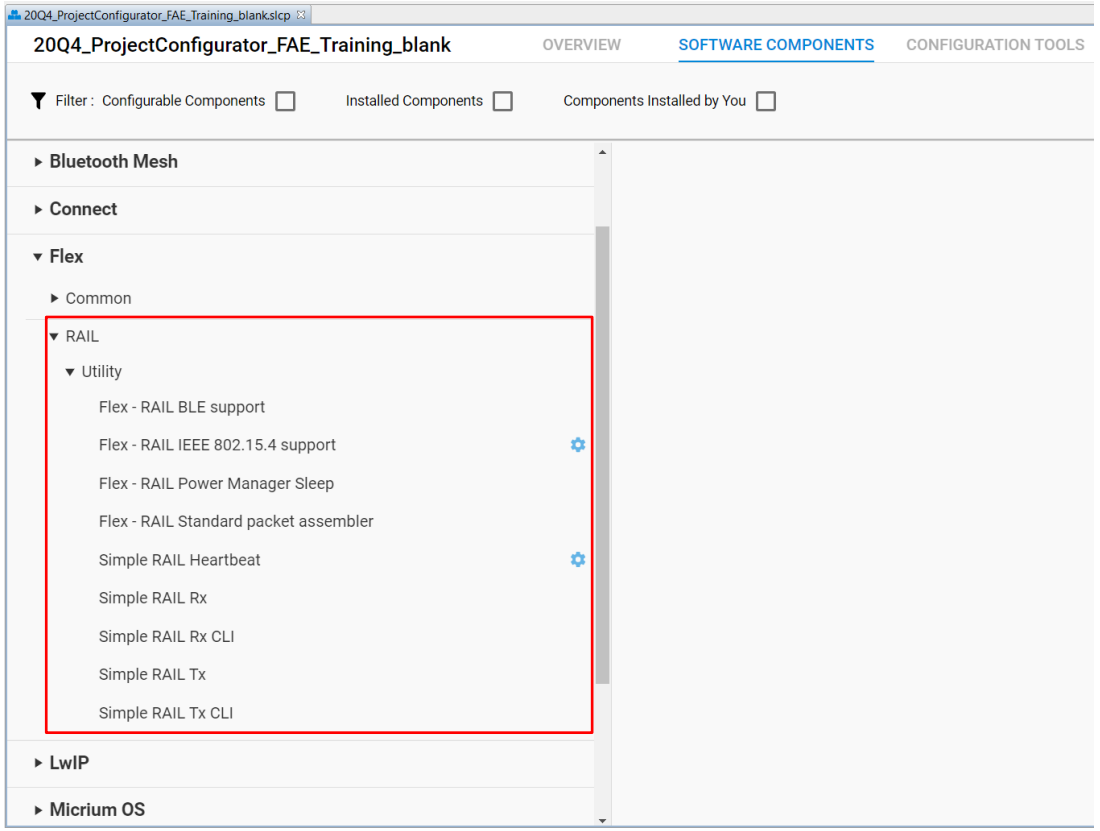
- A full-featured embedded operating system
  - Tightly integrates all components of the operating system together
- Can be added as a **Software Component** using Project Configurator in Simplicity Studio v5
- Documentation: https://doc.micrium.com/display/OSUM50900



Gecko Platform – RAIL Library

- Abstracts the register level of Silicon labs wireless radio devices
- Provides API to simplify standards-based and proprietary radio development
- Underlies the Silicon Labs wireless stacks as a low-level radio HAL
- Useful for creating Proprietary radio protocols

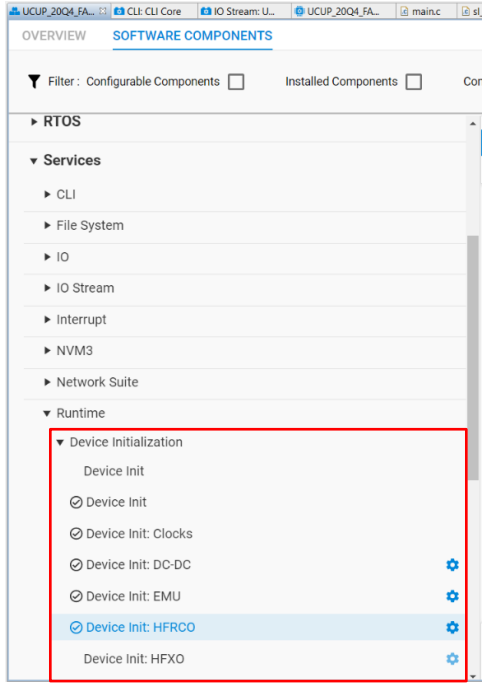
- https://docs.silabs.com/rail/2.9/



Platform Services Detail

Device Initialization

- Device Init component adds autogen files and function calls to project:
  - Errata fixes (CHIP_Init();)
  - EMU/DCDC
  - Clocks and Oscillators
- Configurable in **Component Editor**
- Components in [Platform] > [Services] > [Runtime]
- API Documentation: https://docs.silabs.com/gecko-platform/latest/service/api/group-device-init



In main.c:

```
int main(void)
{
  // Initialize Silicon Labs device, system, se
  // Note that if the kernel is present, proces
  // this call.
  sl_system_init();

  // Initialize the application. For example, c
  // task(s) if the kernel is present.
  app_init();

#if defined(SL_CATALOG_KERNEL_PRESENT)
  // Start the kernel. Task(s) created in app_i
  sl_system_kernel_start();
#else // SL_CATALOG_KERNEL_PRESENT
  while (1) {
    // Do not remove this call: Silicon Labs co
    // must be called from the super loop.
    sl_system_process_action();

    // Application process.
    app_process_action();

#if defined(SL_CATALOG_POWER_MANAGER_PRESENT)
    // Let the CPU go to sleep if the system al
    sl_power_manager_sleep();
#endif
  }
#endif // SL_CATALOG_KERNEL_PRESENT
}
```

In sl_system_init.c:

```
void sl_system_init(void)
{
  sl_platform_init();
  sl_driver_init();
  sl_service_init();
  sl_stack_init();
  sl_internal_app_init();
}
```

In sl_event_handler.c:

```
void sl_platform_init(void)
{
  CHIP_Init();
  sl_device_init_dcdc();
  sl_device_init_hfrco();
  sl_device_init_clocks();
  sl_device_init_emu();
}
```

System Init and Action Processing

- System Init
  - Via sl_system_init() function
  - Bare metal: Initialize Silicon Labs device, system, service(s) and protocol stack(s)
  - Kernel mode: same as bare metal **plus** create system action processing tasks
- System Action Processing
  - Bare metal: Via sl_system_process_action() function
  - Kernel mode will perform action processing via tasks created in sl_system_init() subroutines after call to sl_system_kernel_start()
- API Documentation: https://docs.silabs.com/gecko-platform/latest/service/api/group-system

In sl_system_init.c:

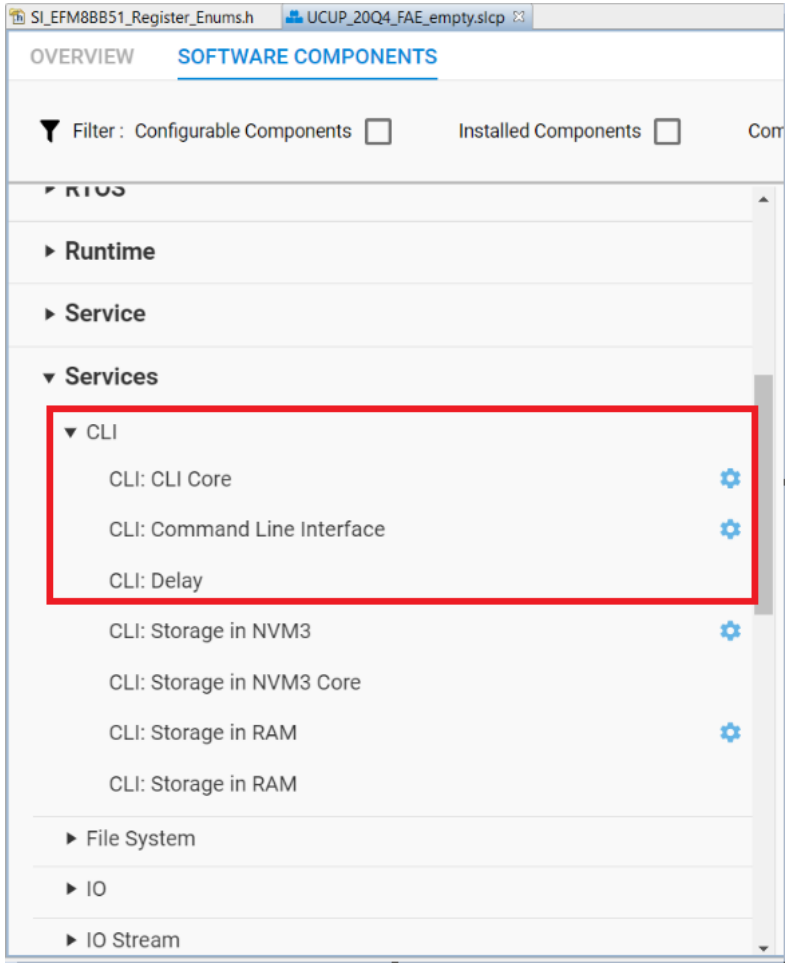
```
void sl_system_init(void)
{
  sl_platform_init();
  sl_driver_init();
  sl_service_init();
  sl_stack_init();
  sl_internal_app_init();
}
```

In sl_system_process_action.c:

```
void sl_system_process_action(void)
{
  sl_platform_process_action();
  sl_service_process_action();
  sl_stack_process_action();
  sl_internal_app_process_action();
}
```

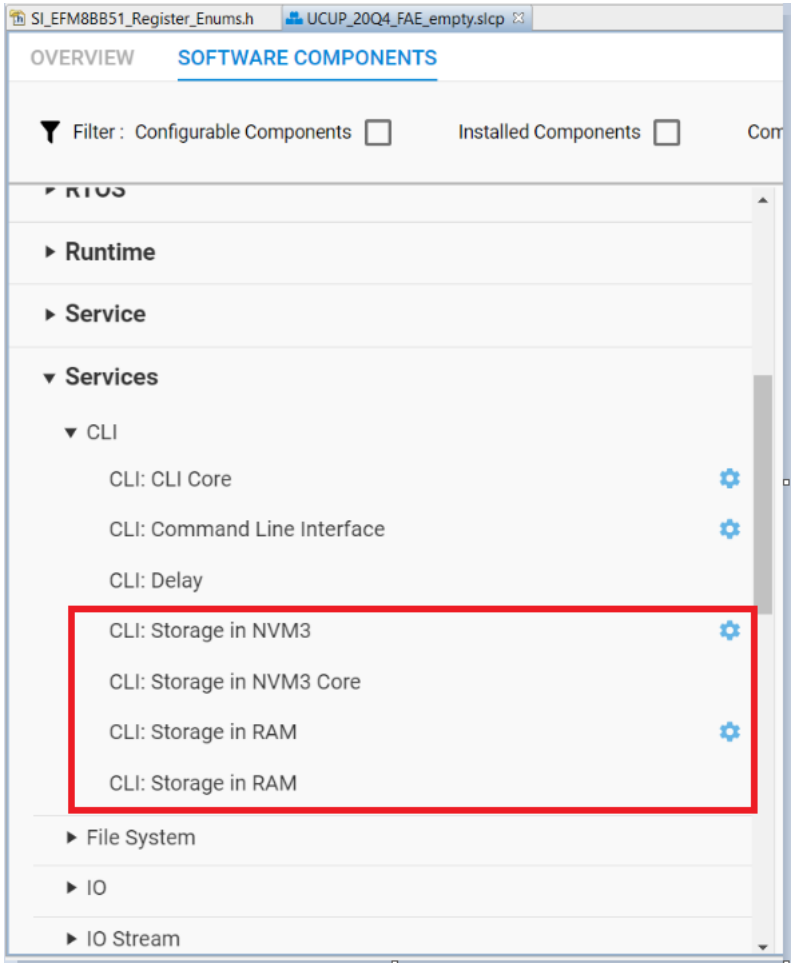
CLI

- Simplifies integration of command line interface to embedded programs
  - Receive keyboard input, parse commands/arguments, call handler function
- Uses IO Stream component for I/O
- Features:
  - Command groups
  - Help
  - Auto-complete
  - Cursor movement
  - Command history
  - Compile or run time command definition
- Command decoding and validation
- Supports multiple instances
  - CLI instances will use separate IO Stream instances
- Implementation different for bare-metal vs kernel (RTOS):
  - Bare-metal: process-action function to poll IO stream (sl_system_process_action())
  - Kernel: CLI will create task (input handler()) to poll IO stream
- API documentation: https://docs.silabs.com/gecko-platform/latest/service/api/group-cli



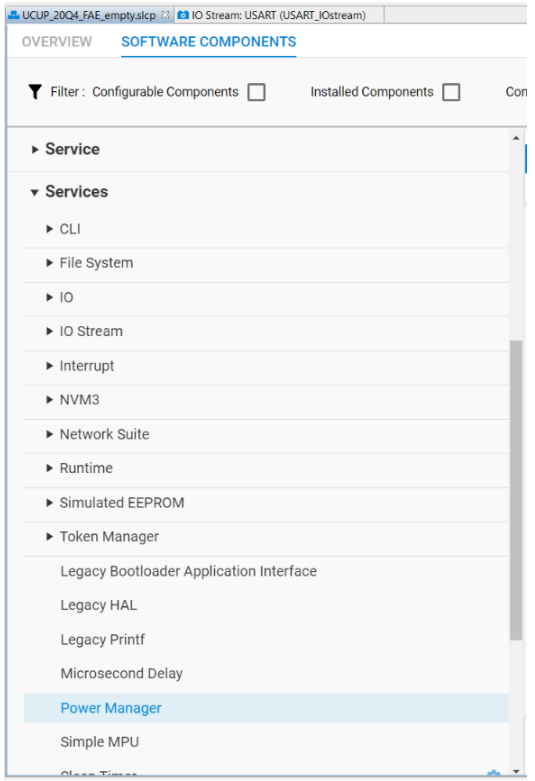
CLI Storage

- CLI storage is an added feature to the CLI component
- Support for CLI commands stored in and executed from different memory regions:
  - RAM
  - NVM3
- Stored commands can be queued and then executed at once
  - Useful in testing or for time critical command sequences
- API documentation:
  - RAM: https://docs.silabs.com/gecko-platform/latest/service/api/group-cli-storage-ram
  - NVM3: https://docs.silabs.com/gecko-platform/latest/service/api/group-cli-storage-nvm3



Power Manager

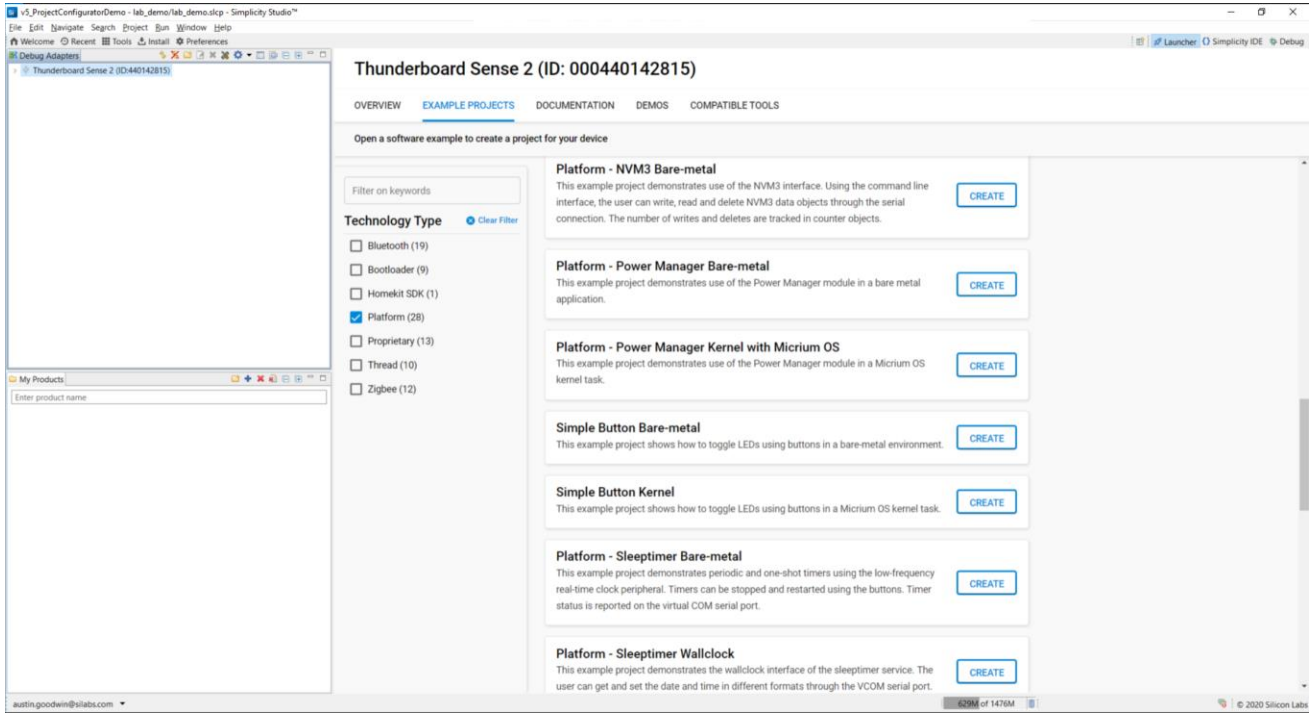
- Manages interaction between software (components) and hardware (CPU/EMU/CMU/energy modes)
- Interaction governed by two interfaces: requirements and notifications
- Requirements: indicates the lowest energy mode the system can go to while performing certain operations
  - Registered by software modules when performing actions
  - Removed by software modules after action complete
- Notifications: allows software module to be notified of any energy mode transition
- Entering low power mode - Kernel vs Bare metal:
  - Kernel – power mode transitions handled by RTOS
  - Bare metal – application required to call sl_power_manager_sleep() to query different software components in project and determine lowest possible energy mode
- Overview: https://docs.silabs.com/gecko-platform/latest/service/power_manager/overview
- API documentation: https://docs.silabs.com/gecko-platform/latest/service/api/group-power-manager



Energy Mode	Description
EM0	System is fully operational. All peripherals are available. The CPU is executing code.
EM1	The CPU is in sleep mode and is not executing any code.
EM2	High frequency clock sources are shut down. Peripherals that require a high frequency clock are unavailable or have limited functionalities.
EM3	Low frequency clock sources are shut down. Peripherals that require a low frequency clock are unavailable.

Platform Examples

- Examples of Platform Component usage
- Available in GSDK 3.1 and later (create directly from Simplicity Studio v5)
- These are generally high-level examples that demonstrate use cases for platform components



Special Considerations and Takeaways

- New Programming Model:
  - High level project architecture level configuration
  - Important to understand structure when building/modifying applications in Simplicity Studio v5
  - Closely ties Project Configurator to Platform Components
- Project Configurator does not generate emlib level initialization or peripheral code, but does include initialization code defined by and contained in installed software components
- Extensive documentation available at docs.silabs.com
- Extensive examples available:
  - Platform Examples (in Simplicity Studio v5)
  - MCU Peripheral Examples (https://github.com/SiliconLabs/peripheral_examples)

Thank you!