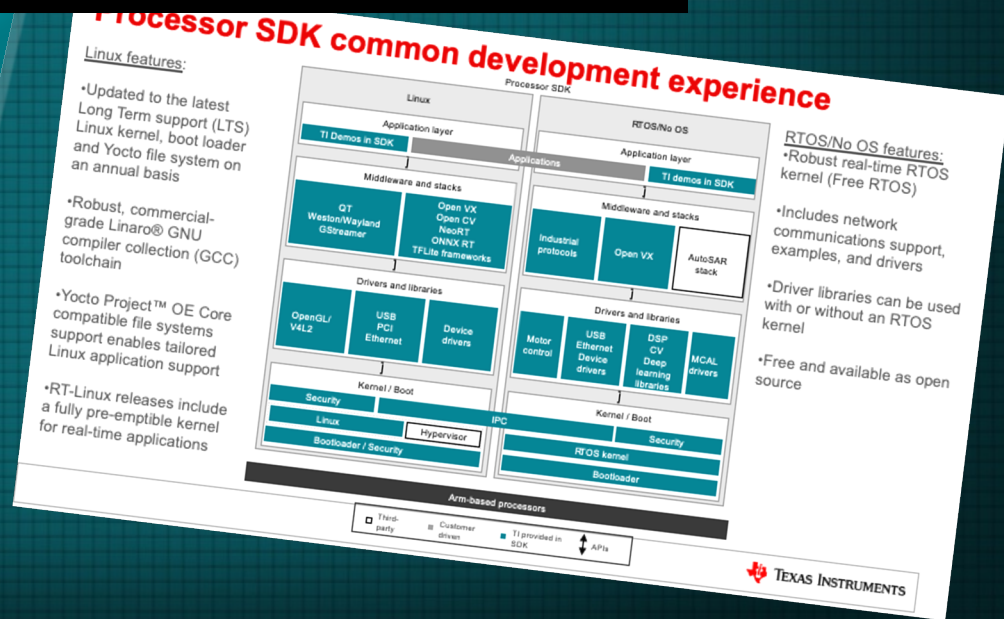


AM64X SOFTWARE DEVELOPMENT OVERVIEW

RUNNING ALL
CORES ON THE
DEVICE FOR
FULL
ENTITLEMENT



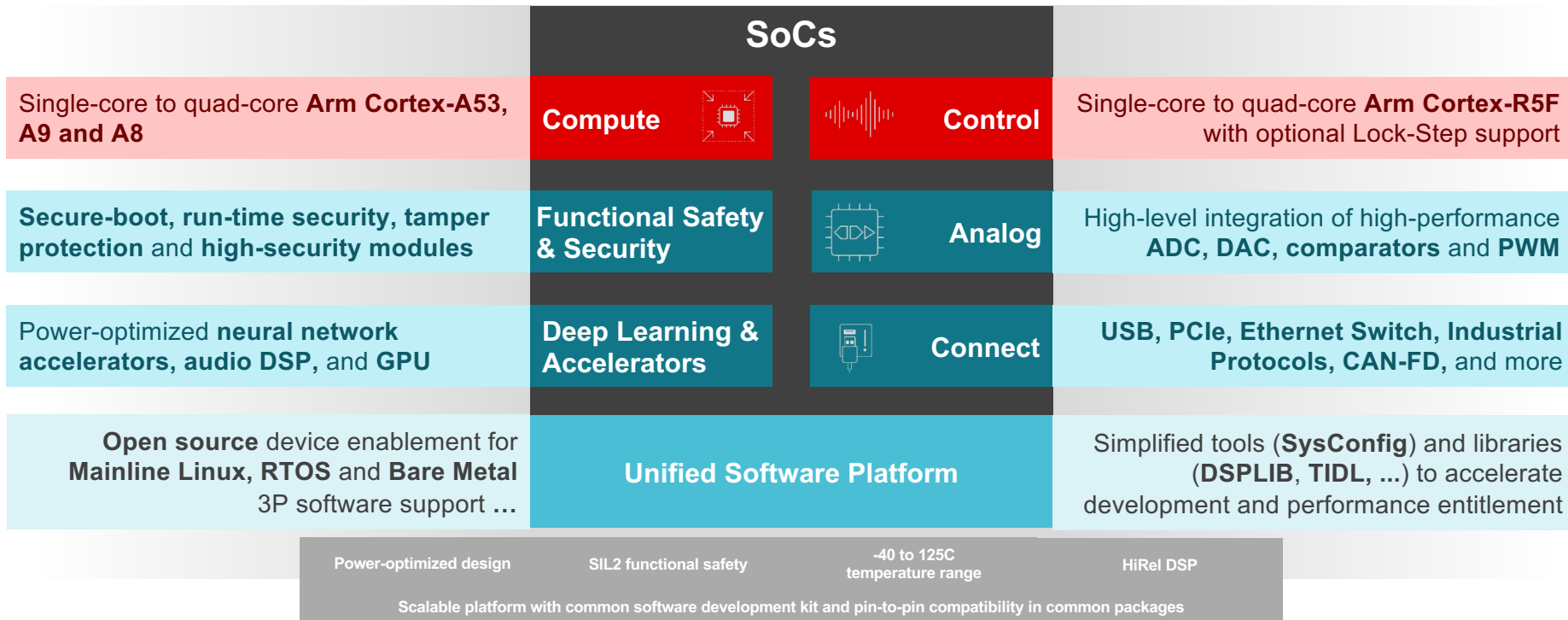
Agenda

- Short Overview
- MCU+ SDK Development
- Demo
- Linux Development
- Summary

Sitara overview



Scalable, cost-optimized portfolio with accelerators, analog integration, robust connectivity, security and functional safety designed for industrial markets



AM64x (17mm x 17mm) Cortex®-A53 based processors

• Cores & Memory

- Dual Cortex-A53 up to 1GHz
- Dual or Quad Cortex-R5F up to 800MHz
- >2MB on-chip SRAM
- ECC on all critical memories
- 16b LPDDR4/DDR4 controller with inline ECC

• Functional safety features

- 400MHz Cortex-M4F subsystem has **freedom from interference** to enable usage as a safety monitor
 - Dedicated Peripherals I2C, SPI, UART & GPIO
 - Tightly coupled memory of 256KB
- Diagnostic tool kit for entire SoC voltage, temp, clock, ECC monitors and Error signaling

• 2xPRU-ICSS-Gb

- Enables up to 2x Gb industrial Ethernet protocols
- 1x industrial Ethernet protocol + motor control current and position feedback

• Peripheral / IO Highlight

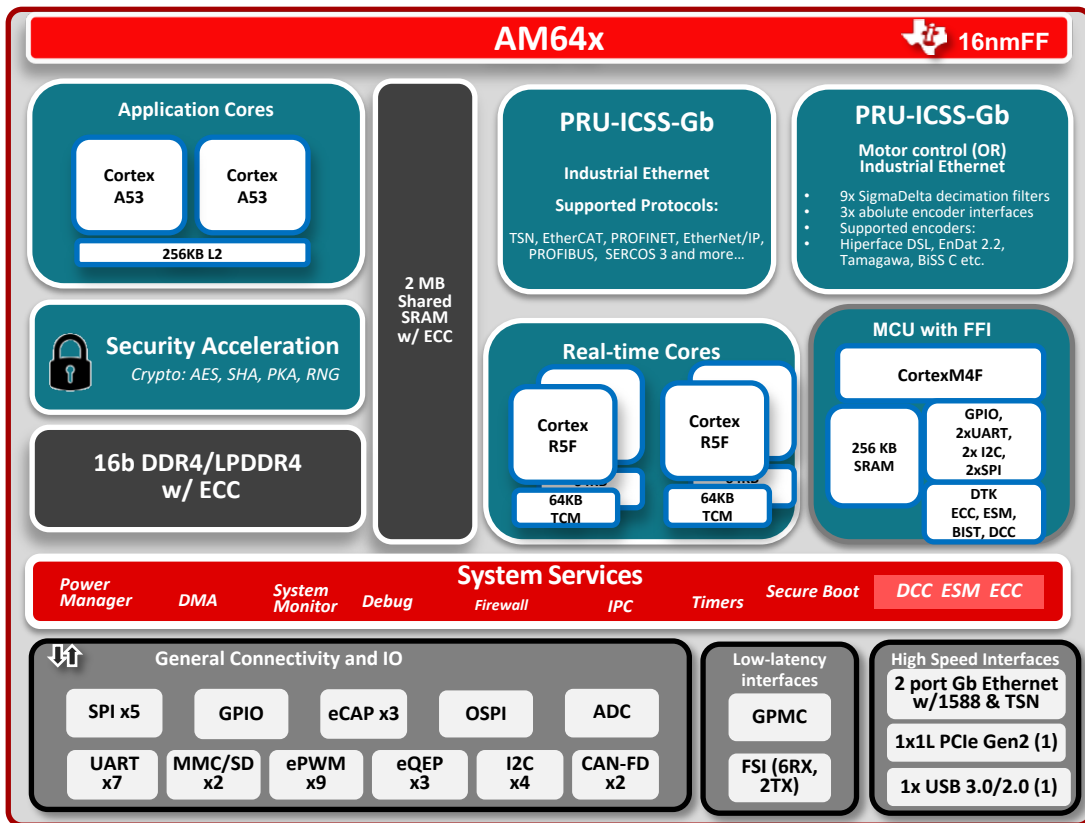
- GPMC (32b parallel bus) and FSI (serial connection for use with TI's C2000 MCUs) offer low-latency interfaces to motor control front-end
- PCIe Gen2, USB3.0/2.0, and 2-port Gb Ethernet Switch CPSW provide high-speed (Gbps) connectivity options
- RS485 support on UART
- Octal/Quad-SPI with execution-in-place support

• Integrated analog

- 8-channel, 12-bit ADC with 4 Msps
- Simplified power solution, Integrated Voltage Monitors

• Package

- 17.2 x 17.2mm, 0.8mm ball pitch



(1) PCIe and USB 3.0 share the same SERDES

AM243x (17mm x 17mm) Cortex®-R5F based processors

• Cores & Memory

- Dual Cortex-A53 up to 1GHz
- Dual or Quad Cortex-R5F up to 800MHz
- >2MB on-chip SRAM
- ECC on all critical memories
- 16b LPDDR4/DDR4 controller with inline ECC

• Functional safety features

- 400MHz Cortex-M4F subsystem has **freedom from interference** to enable usage as a safety monitor
 - Dedicated Peripherals I2C, SPI, UART & GPIO
 - Tightly coupled memory of 256KB
- Diagnostic tool kit for entire SoC voltage, temp, clock, ECC monitors and Error signaling

• 2xPRU-ICSS-Gb

- Enables up to 2x Gb industrial Ethernet protocols
- 1x industrial Ethernet protocol + motor control current and position feedback

• Peripheral / IO Highlight

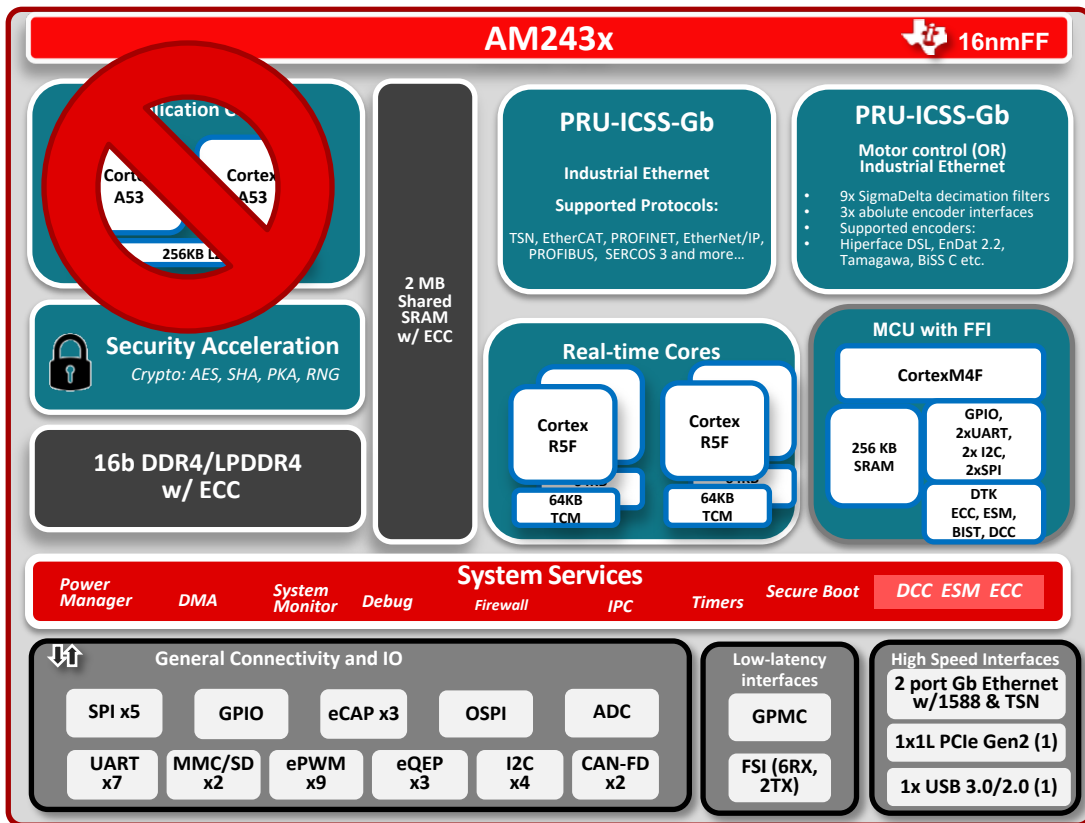
- GPMC (32b parallel bus) and FSI (serial connection for use with TI's C2000 MCUs) offer low-latency interfaces to motor control front-end
- PCIe Gen2, USB3.0/2.0, and 2-port Gb Ethernet Switch CPSW provide high-speed (Gbps) connectivity options
- RS485 support on UART
- Octal/Quad-SPI with execution-in-place support

• Integrated analog

- 8-channel, 12-bit ADC with 4 Msps
- Simplified power solution, Integrated Voltage Monitors

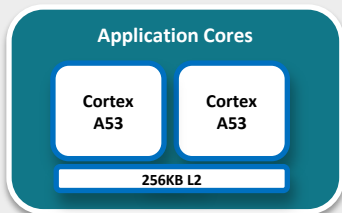
• Package

- 17.2 x 17.2mm, 0.8mm ball pitch



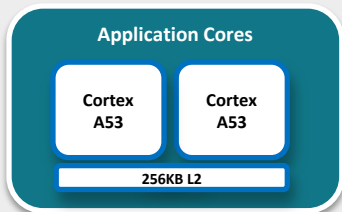
(1) PCIe and USB 3.0 share the same SERDES

Processor Details

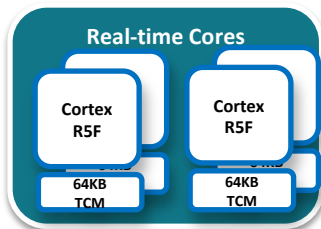


- Dual / Single Arm® Cortex®-A53 (only on AM64x devices)
 - Up to 1 GHz, ARMv8-A instructions set
 - Dual core cluster with shared 256KB L2. Each core has 32KB L1 I\$ and D\$
 - AArch64 for 64b support and new architecture features
 - Backward compatible with code for previous Arm processors (AArch32)
 - Integrated Neon™ processing engine and VFPv4 compatible hardware
 - Hardware virtualization support

Processor Details

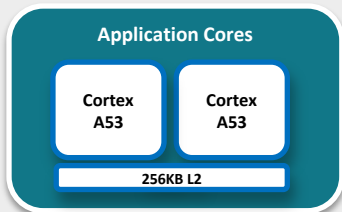


- Dual / Single Arm® Cortex®-A53 (only on AM64x devices)
 - Up to 1 GHz, ARMv8-A instructions set
 - Dual core cluster with shared 256KB L2. Each core has 32KB L1 I\$ and D\$
 - AArch64 for 64b support and new architecture features
 - Backward compatible with code for previous Arm processors (AArch32)
 - Integrated Neon™ processing engine and VFPv4 compatible hardware
 - Hardware virtualization support

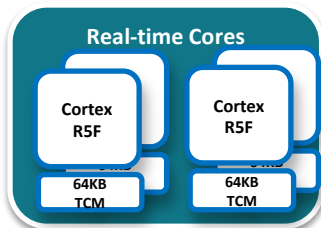


- 2x- or 4x- Arm® Cortex®- R5Fs (AM64x and AM243x)
 - Up to 800MHz, ARMv7-R instruction set
 - 2x dual clusters with total of 256KB TCM. Each core has 32KB I\$ and 32KB D\$
 - No Lock-step operation, only split mode. Optimized for real time operations
 - Integrated Neon™ processing engine and VFPv3 compatible hardware
 - Multi-processing extensions for multiprocessing functionality
 - Vectored Interrupt Manager (VIM)

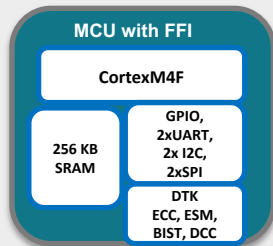
Processor Details



- Dual / Single Arm® Cortex®-A53 (only on AM64x devices)
 - Up to 1 GHz, ARMv8-A instructions set
 - Dual core cluster with shared 256KB L2. Each core has 32KB L1 I\$ and D\$
 - AArch64 for 64b support and new architecture features
 - Backward compatible with code for previous Arm processors (AArch32)
 - Integrated Neon™ processing engine and VFPv4 compatible hardware
 - Hardware virtualization support



- 2x- or 4x- Arm® Cortex®- R5Fs (AM64x and AM243x)
 - Up to 800MHz, ARMv7-R instruction set
 - 2x dual clusters with total of 256KB TCM. Each core has 32KB I\$ and 32KB D\$
 - No Lock-step operation, only split mode. Optimized for real time operations
 - Integrated Neon™ processing engine and VFPv3 compatible hardware
 - Multi-processing extensions for multiprocessing functionality
 - Vectored Interrupt Manager (VIM)



- 1x- Arm® Cortex®- M4F (AM64x and AM243x)
 - Up to 400MHz, ARMv7-R instruction set
 - Integrated Full precision floating point unit (FPU)
 - 256KB of local SRAM, 192KB I-code and 64KB D-code
 - Ability to execute code from unified memory (I-code/D-code) or external memory via System bus. SoC integration includes Region Address Translation (RAT) to enable contiguous memory
 - Nested Vector Interrupt Controller (NVIC)

Processor Details

PRU-ICSS-Gb

Industrial Ethernet

Supported Protocols:

TSN, EtherCAT, PROFINET, EtherNet/IP, PROFIBUS, SERCOS 3 and more...

PRU-ICSS-Gb

Motor control (OR) Industrial Ethernet

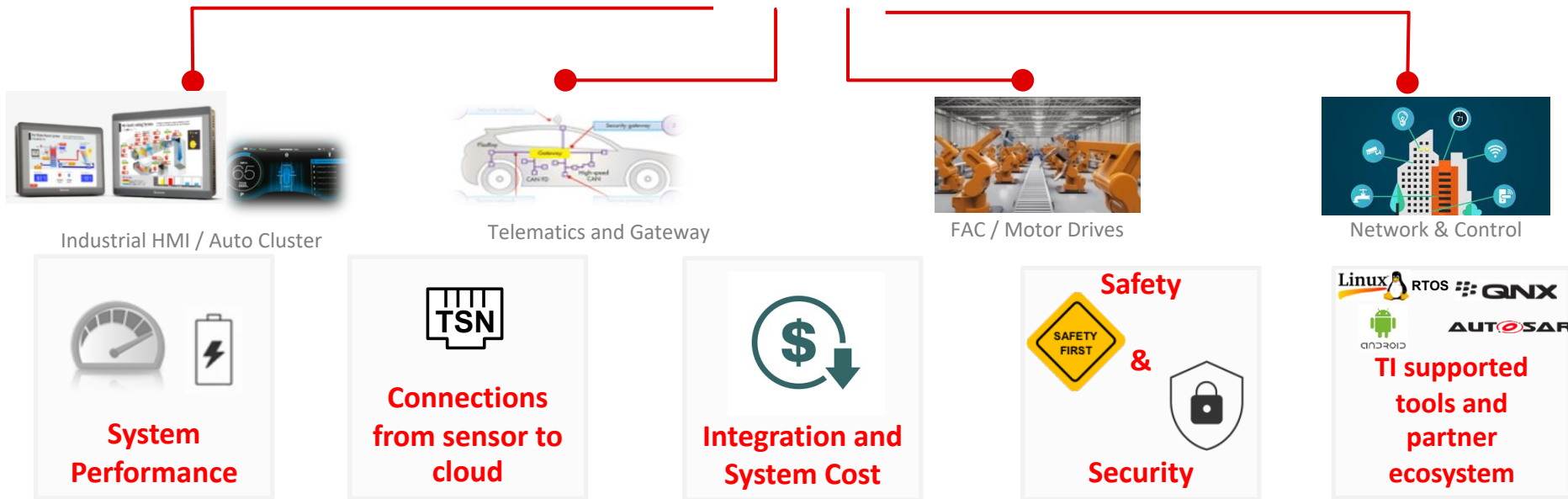
- 9x SigmaDelta decimation filters
- 3x absolute encoder interfaces
- Supported encoders:
 - Hiperface DSL, EnDat 2.2, Tamagawa, BiSS C etc.

- 2x PRU_ICSSG – Industrial Communication Subsystems (AM64x and AM243x)
 - 2x general-purpose PRU cores with 12KB program RAM, 2 KB data RAM per CPU, MAC, CRC16/32 HW accelerator
 - 2x auxiliary Real-time transfer units with 8KB program RAM, 2KB data RAM, MAC, CRC16/32 HW accelerator
 - 2x transmit real-time transfer units with 6KB program RAM, 2KB data RAM, MAC, CRC16/32 HW accelerator
 - 64KB shared RAM
 - Two MDIO ports
 - Two PRU_ICSSG support 2x (each) RGMII or MII_RT. One PRU_ICSSG supports 2x SGMII, RGMII, or MII_RT
 - Two Industrial Ethernet Peripheral and industrial Ethernet timers (IEP)
 - 1 x 16550-compatible UART
 - Capable of supporting master and/or slave modes of protocols such as:
 - TSN, Profinet IRT, Ethernet/IP with DLR, Profibus, EtherCAT, POWERLINK, Sercos 3, Hiperface DSL, BiSS C, EnDat 2.2, HSR/PRP, and more
 - Capable of supporting operation as standard Gb Ethernet
 - SD interface upgraded to support Manchester encoding
 - Load sharing of PRU's with concurrent Sigma Delta and EnDAT interface
 - Reset isolation interface

Sitara built around a unified software platform



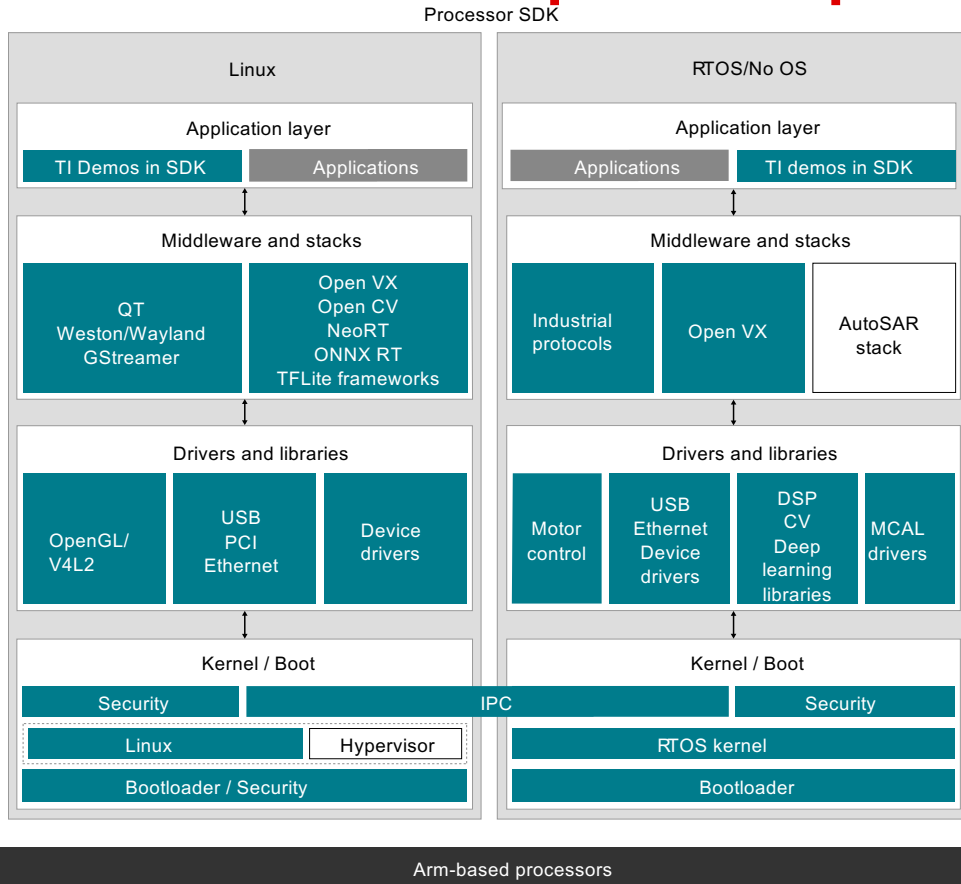
100% code compatibility



Processor SDK common development experience

Linux features:

- Updated to the latest Long Term support (LTS) Linux kernel, boot loader and Yocto file system on an annual basis
- Robust, commercial-grade ARM® GNU compiler collection (GCC) toolchain
- Yocto Project™ OE Core compatible file systems support enables tailored Linux application support
- RT-Linux releases include a fully pre-emptible kernel for real-time applications
- AM64x only



RTOS/No OS features:

- Robust real-time RTOS kernel (Free RTOS)
- Includes network communications support, examples, and drivers
- Driver libraries can be used with or without an RTOS kernel
- Free and available as open source
- Available for AM64x and AM243x

Software Enablement

Application Cores

Cortex
A53

Cortex
A53

- Processor SDK for Linux
- Other HLOS commercial offerings
- Bootloader: U-Boot
- Load other cores via remoteproc

Real-time Cores

Cortex
R5F

Cortex
R5F

- MCU+ SDK
- No RTOS and RTOS capability, CCS IDE
- Bootloader: SBL or load via Linux

MCU with FFI

CortexM4F

- MCU+ SDK
- No RTOS and RTOS capability, CCS IDE
- Bootloader: SBL or load via Linux

PRU-ICSS-Gb

PRU

PRU-ICSS-Gb

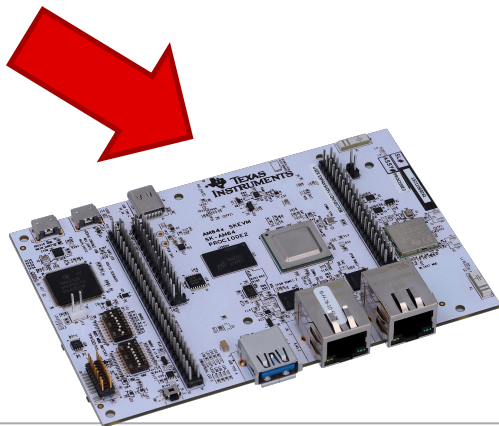
PRU

- PRU Software Support Package
- No RTOS, bare metal code only, CCS IDE
- Bootloader: SBL or load via Linux

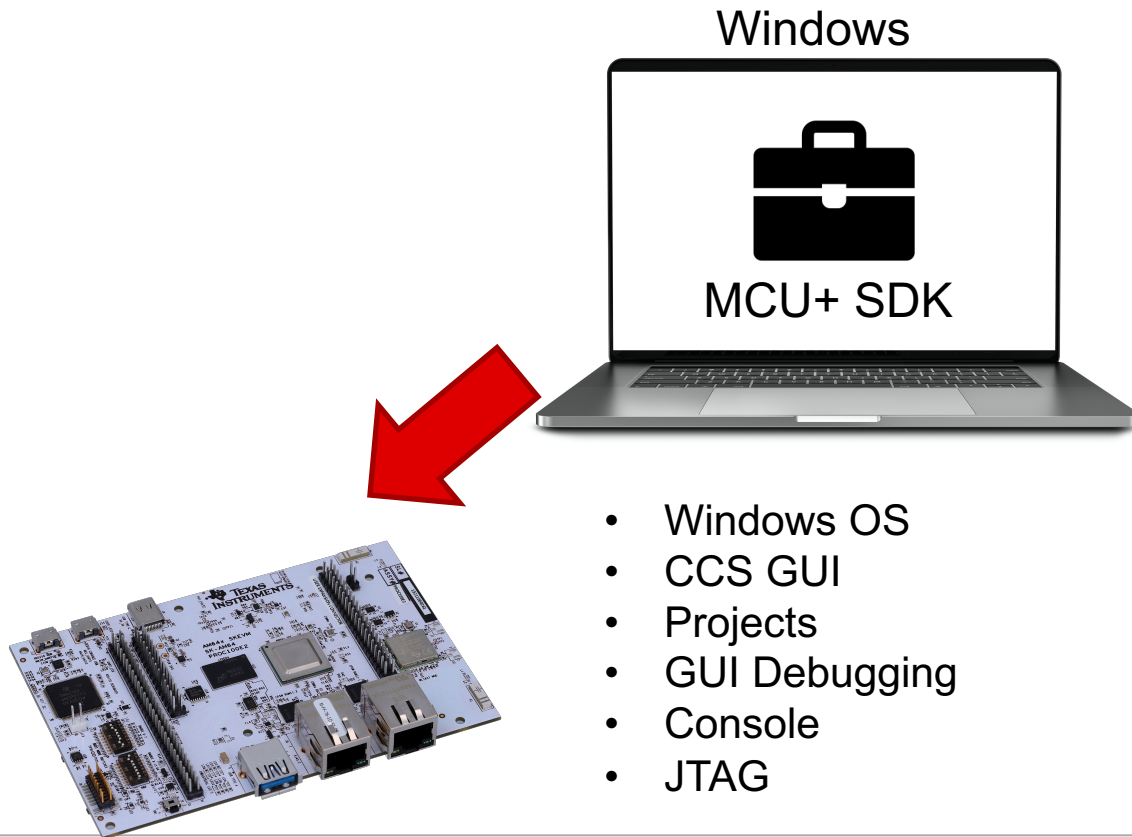
Linux (Kernel) Development Overview



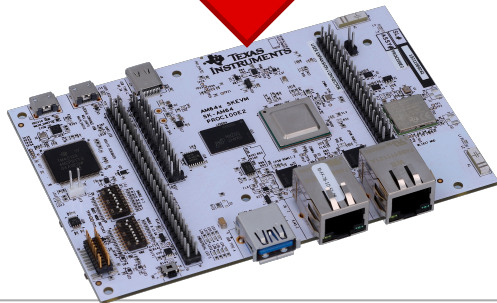
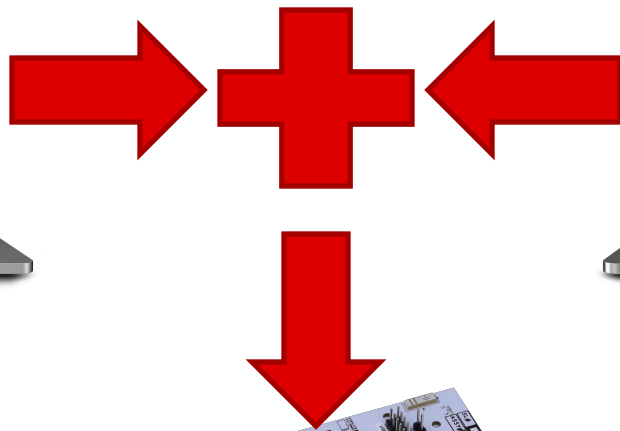
- Linux OS (Ubuntu)
- Largely Command Line
- make files
- kgdb/gdb
- printf()
- TFTP/NFS



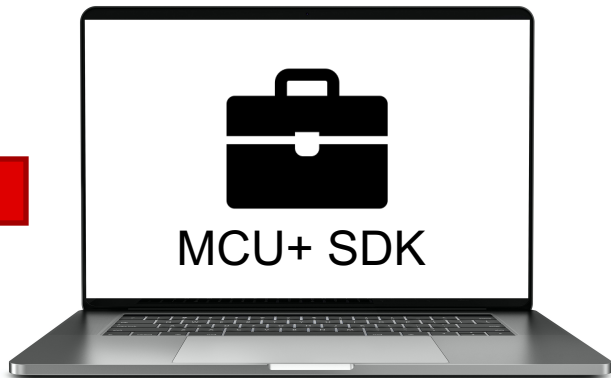
no RTOS/RTOS Development Overview



Software Development Summary



Windows



- Linux OS (Ubuntu)
- Largely Command Line
- make files
- kgdb/gdb
- printf()
- TFTP/NFS

- Windows OS
- CCS GUI
- Projects
- GUI Debugging
- Console
- JTAG

MCU+ SDK | Benefits for the end user



MCU Simplicity

- Simple drivers with GUI configuration tool
- 110+ examples to run and debug with CCS
- Multi-Core Bootloader examples included



MCU Optimized

- Low Latency Drivers
- Low Memory Usage
- No-OS or FreeRTOS



Tools

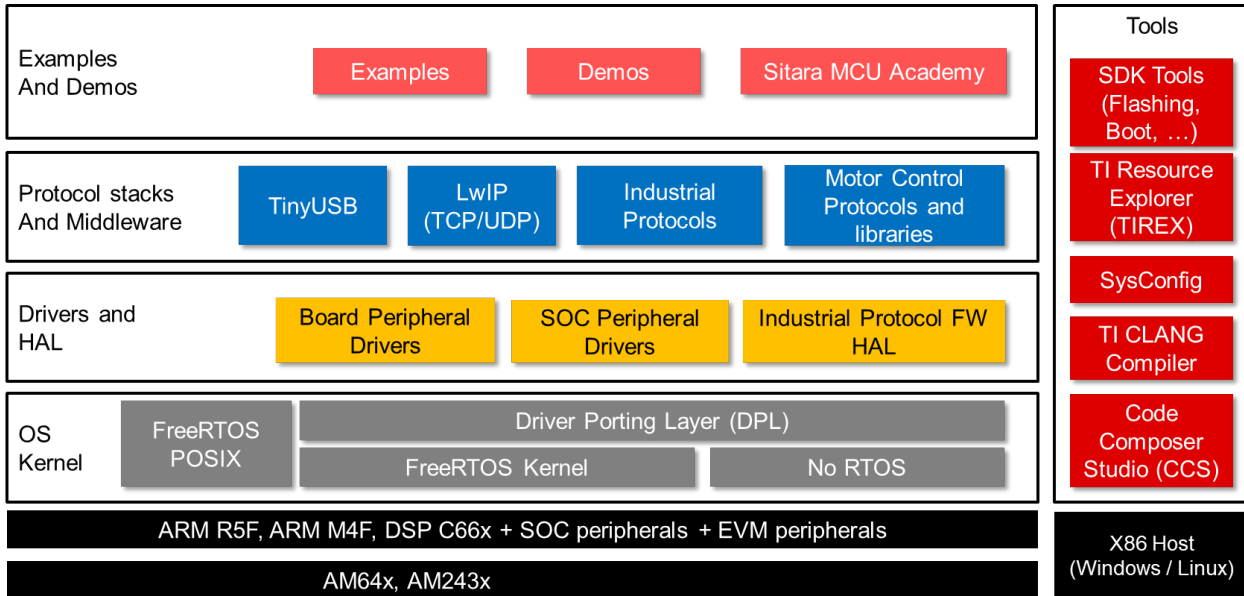
- SysConfig Tool
- MCU+ Academy & TI Resource Explorer
- TI ARM CLANG compiler
- CCS + FreeRTOS live debugging
- Board flashing tools



Libraries

- TinyUSB, LwIP, TSN
- Industrial Networking Stacks Integrated
- Motor Position Encoders and Control Algorithms
- Inter-Processor Communication (IPC)

Sitara MCU | MCU+ SDK Overview



- Simplified and easy to use MCU+ SDK for MCU+ applications on R5F, M4F and C66
- Open Source OS and middleware stacks – FreeRTOS, LwIP, tinyUSB
- Simplified, low memory, low latency optimized drivers
- SysConfig for easy system configuration like pinmux, clock, driver setup
- Pre-integrated industrial protocols and motor control protocols
- Lots of examples and a step by step “MCU Academy” to quickly get started
- Interfacing with other OS like Linux (on A53) to expand to more applications and end equipments

OS Environment | FreeRTOS

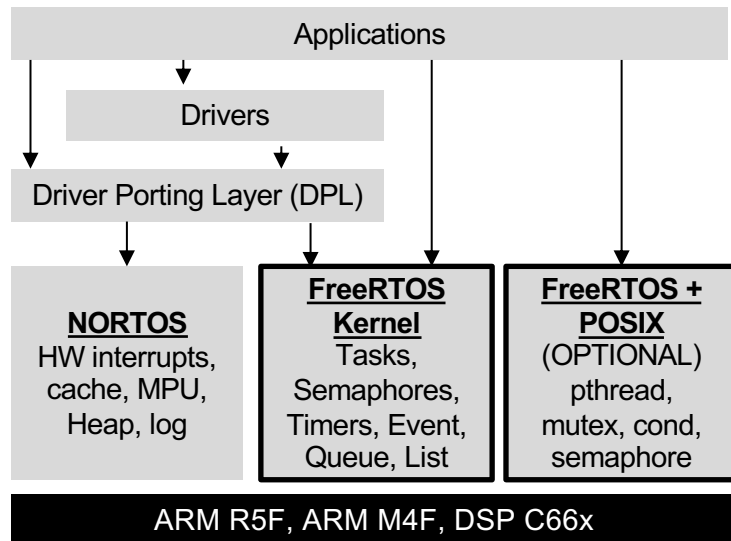


Overview

- Well established RTOS with 15+ years of deployments and partnerships with leading semiconductor vendors
- MIT open source license, allows customer to deploy in production and also protect their IP

FreeRTOS Kernel [\[LINK\]](#)

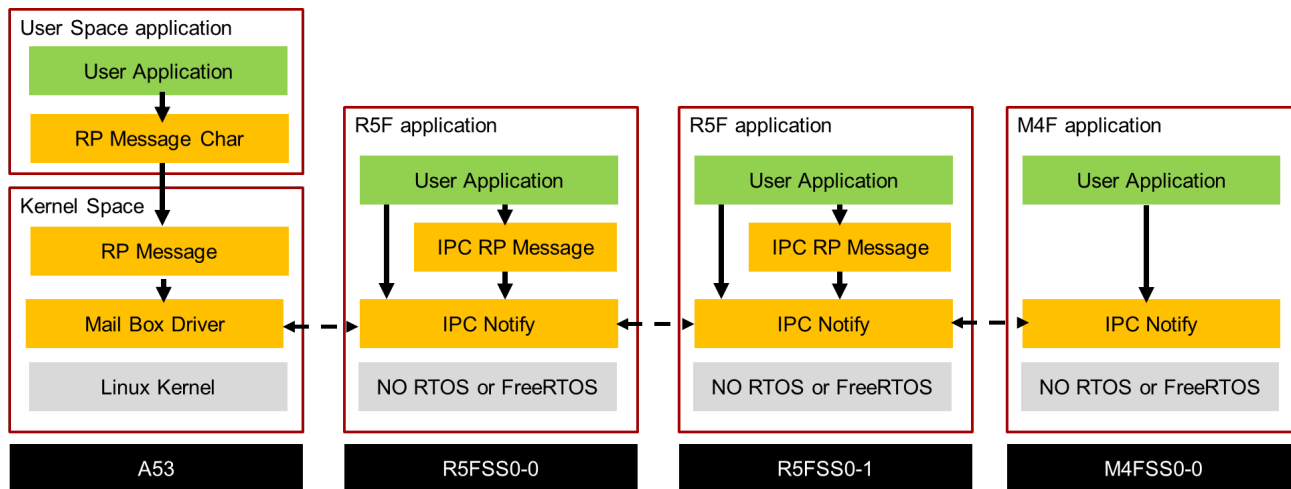
- Primary RTOS for MCU+ SDK on R5F, M4F, C66 CPUs
- Pre-integrated with all device drivers, middleware (LwIP, TinyUSB), industrial protocols
- RTOS aware debugging and state viewer integrated with CCS IDE
- Deterministic and **sub-micro second** task switch and interrupt latency
- **< 20 KB** RTOS Kernel size
- Driver porting layer (DPL) allows switching to NO RTOS and/or other RTOS, like SafeRTOS
- Optional POSIX threading layer to allow application level portability



Middleware | Inter Processor Communication (IPC)

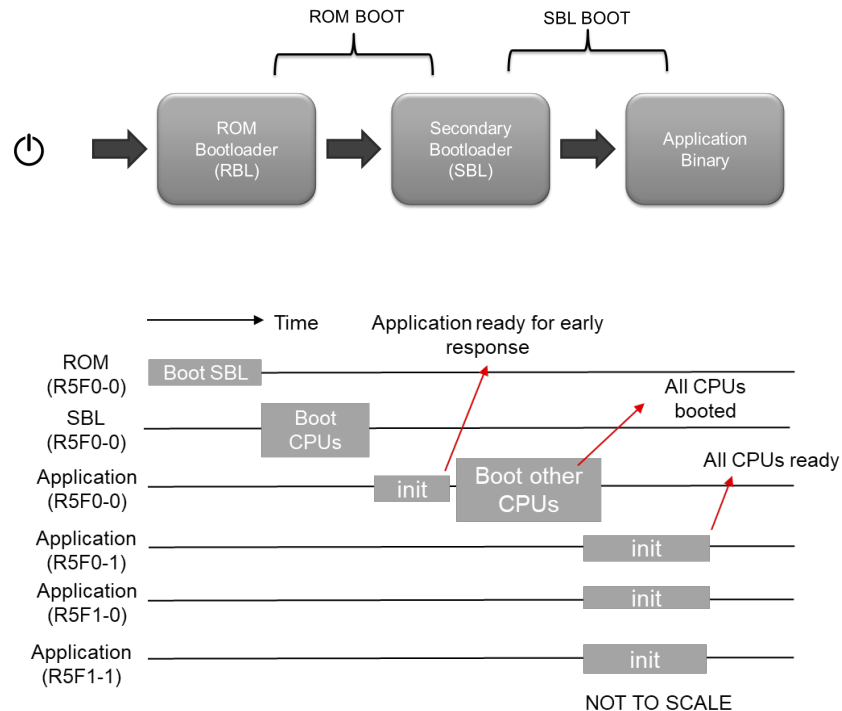
- Inter - Processor Communication enables multiple different CPUs on the SOC to collaborate to realize the system use-case
- **IPC Notify**
 - Low level API, < 1us latency, ~ 6KB code size
 - Allows to interrupt other CPUs using SOC HW mechanisms
- **IPC RPMMessage**
 - Higher level abstracted API, < 5us latency, ~12KB code size
 - Allows to exchange message packets with logical endpoints
 - Used to talk to Linux when present on the SOC

- **Spinlock**
 - HW mechanism to implement mutual exclusion across multiple CPUs
- **Shared memory**
 - Shared memory architecture allows to keep data in shared memory and only pass pointer to data in IPC message, avoids memory copy overheads.



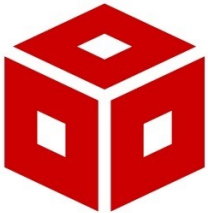
Boot | Boot Flow and Bootloader

- SOC ROM boots a secondary bootloader (SBL). SBL included in MCU+ SDK
- Boot modes supported by SBL
 - UART
 - OSPI
 - MMC/SD – COMING SOON !!
- Features
 - Ability to boot all MCU CPUs on the SOC
 - Optionally, do multi-stage booting, to reduce boot time for “early response” functions
 - Post build tools to convert compiler generated applications into format suitable for flashing and booting
 - Flashing tool to flash application binaries over UART



Sitara MCU | Debug and trace tools

- Code composer studio from TI (**free download**)
 - Eclipse based IDE
 - C6x, R5F, M4F – debug and trace via JTAG
 - single step, breakpoint, watch point, disassembly
 - FreeRTOS aware (Real-time Object View - ROV)
 - Powerful scripting via Debug Server Scripting (DSS)
 - Access to system memory and peripheral registers through Debug Access Port
 - Multicore debugging
 - Multiprocessor debugging
- TRACE32 from Lauterbach
 - Support C6x, ARMv7 (R5F, M4F)
 - OS aware debugging – FreeRTOS, AutoSAR
 - Powerful script language
 - Easy high-level and assembler debugging
 - Support for CoreSight components like Debug Access Port, Trace Funnel, Trace Port Interface Unit, Embedded Trace Buffer, Cross Trigger Interface, Cross Trigger Matrix, System Trace Port, Trace Memory Controller
 - Real-time access to system memory and peripheral registers through Debug Access Port without halting the core
 - Multicore debugging
 - Multiprocessor debugging
 - Safety tool kit and certifications making them useable for ISO 26262 and DO-178C



<http://www.ti.com/tool/download/CCSTUDIO>



www.lauterbach.com/
Search for CHIP = TI

workspace_v10 - hello_world_am64x-evm-r5fs0-0_freertos-ti-arm-clang-example.syscfg - Code Composer Studio

File Edit View Project Run Tools Scripts Window Help

Debug

AM64x_XDS110.ccxml [Code Composer Studio - Device]Texas Instruments XDS110 USB Debug Probe_0/CortexTexas Instruments XDS110 USB Debug Probe_0/CortexTexas Instruments XDS110 USB Debug Probe_0/MALhello_world_main(void *)() at hello_world.c:76 0xfretos_main(void *)() at main.c:52 0x700A2A68prvTaskExitError() at port.c:120 0x700A2F80 (next)Texas Instruments XDS110 USB Debug Probe_0/MAL

Project Explorer

Generated SourceBinariesIncludesDebugtargetConfigshello_world.clinker.cmdmain.cexample.syscfgmakefile_ccs_bootimage_genREADME.htmlsyscfg_c.rov.xs

Terminal

COM13
Starting NULL Bootloader ...

DMSC Firmware Version 21.1.1--v2021.01a (Terrific Lla
DMSC Firmware revision 0x15
DMSC ABI revision 3.1

INFO: Bootloader_runCpu:151: CPU r5f1-0 is initialized to 80000
INFO: Bootloader_runCpu:151: CPU r5f1-1 is initialized to 800000
INFO: Bootloader_runCpu:151: CPU m4f0-0 is initialized to 400000
INFO: Bootloader_loadSelfCpu:214: CPU r5f0-0 is initialized to 8
INFO: Bootloader_loadSelfCpu:214: CPU r5f0-1 is initialized to 8
INFO: Bootloader_runSelfCpu:235: All done, resetting self ...

Viewable Modules

Monitor OS Kernel System

OS Kernel	Task Instances (FreeRTOS)					
Handle	Name	Priority	BasePriority	State	StackBase	EstimatedFreeStack
0x70087608	IDLE	0	0	Ready	0x70085008	4096
0x7008768c	Tmr Svc	15	15	Blocked	0x70086c40	896
0x70087710	freertos_mai	15	15	Running	0x70080000	15872

main.c

main.c

main.c

main.c

test_freertos.c

ti_dpl_config.c

TimerP.c

HwiP_armv7m.c

example.syscfg

main.c

hello_world.c

_vectors() a...

Type Filter Text...

TI DRIVER PORTING LAYER (DP...
Clock 1/1
Debug Log 1/1
MPU ARMv7 4/16
RAT
TIMER
TI DRIVERS (14)
ADC
BOOTLOADER
EPWM
FSL_RX
FSL_TX
GPIO
I2C
IPC
MCAN
MCSPI

Clock
Instance
Input Clock Frequency (Hz)
Input Clock Source
Tick Period (usecs)

TIMER8

25000000

MCU_HFOSC0

1000

Updates Available

Updates are available for your software. Click to review and install updates.

View Terminal Output


Tools | Code Composer Studio IDE

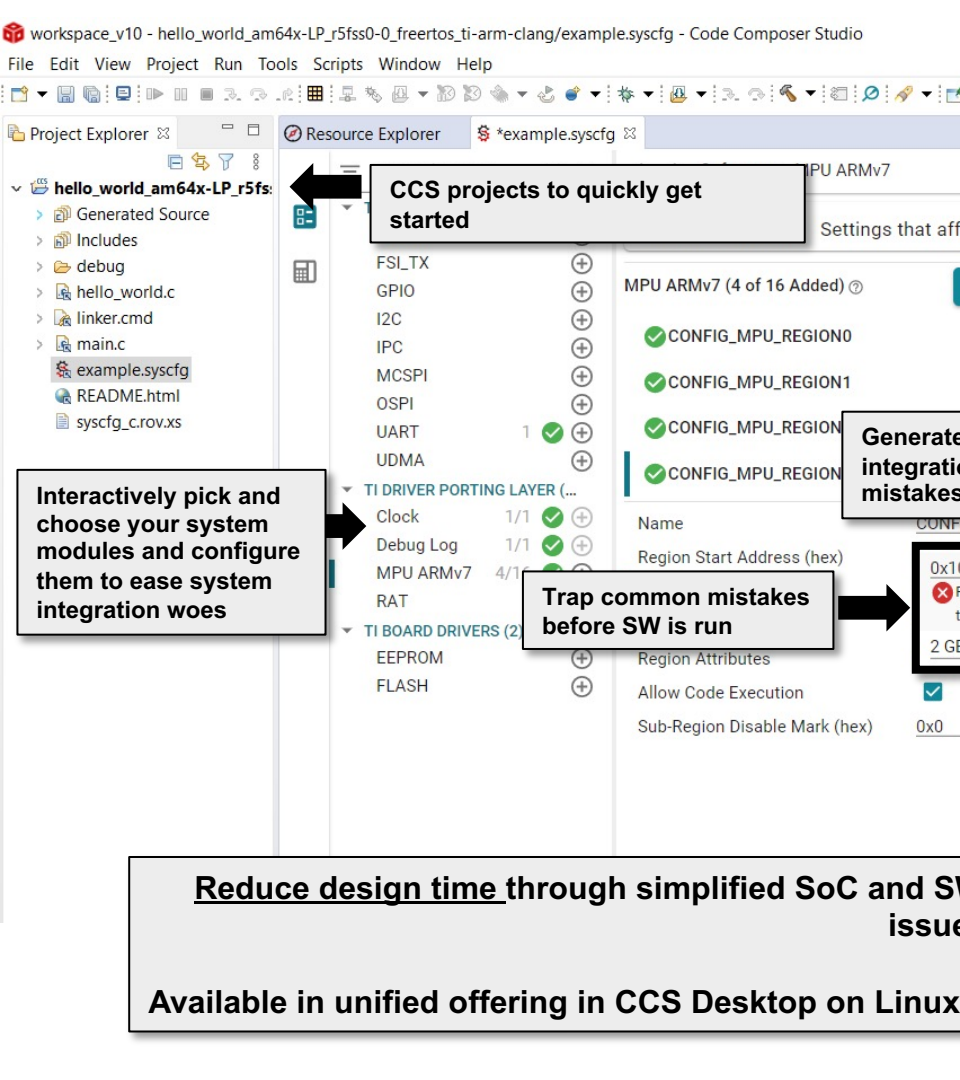
Debug Applications

Monitor System State using ROV

Application Projects, including Multi-CPU projects

Configure application using SysConfig

 TEXAS INSTRUMENTS



Tools | SysConfig

CCS projects to quickly get started

Interactively pick and choose your system modules and configure them to ease system integration woes

Generate, easy to read, system integration code and reduce mistakes

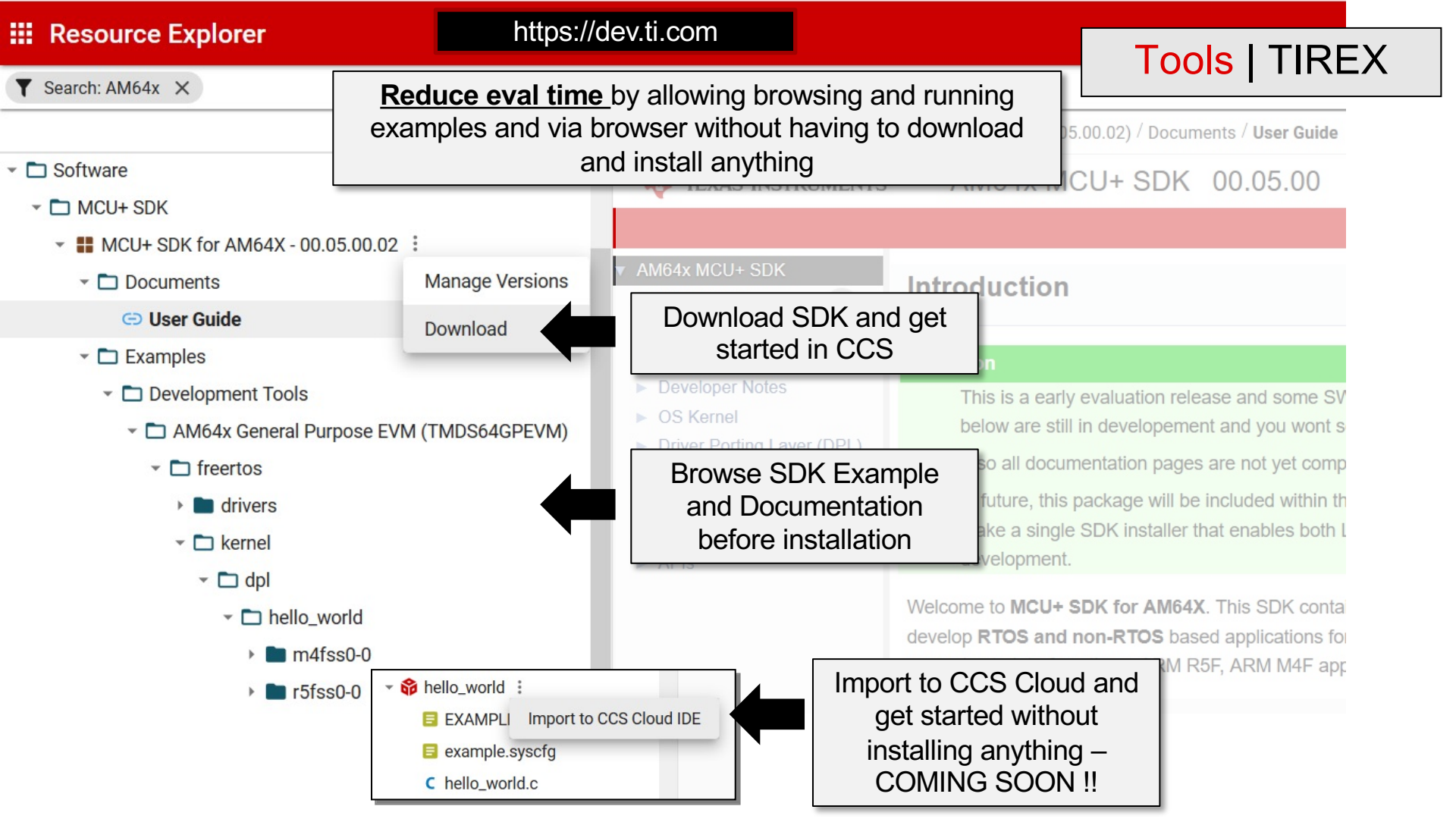
Trap common mistakes before SW is run

Reduce design time through simplified SoC and SW view in SysConfig to ease system integration issues

Available in unified offering in CCS Desktop on Linux/Windows and in the cloud on TI.com

Tools | SysConfig Features and Benefits

Feature	Benefit to end user
SW Driver Configuration	Quickly add a peripheral to your project, interactive config without needing to refer TRM or API guide. Only include features and modules that are needed, saving code size and complexity.
Integrated Pinmux Configuration	No need for separate pinmux tool, resolve pin conflicts interactively and without running on EVM
Clock enable and clock frequency setup depending on SW driver that is selected	Hides DMSC / SYSFW APIs call complexity from end user. Only modules that are needed get enabled.
MPU / MMU / RAT configuration	Easily set memory access controls for the CPU without needing to read TRM
Debug log configuration to select UART, CCS or shared memory output	Target logs to console of interest, quickly enable/disable log and log “zones”
IPC configuration	Tune shared memory usage and IPC type (low latency vs normal latency)
Board peripheral configuration (FLASH, EEPROM, LED, ETHPHY)	Pick and choose board level peripherals and quickly start using them in your project
Multi-core validation	Resolve resource conflicts including across multiple R5F and M4F before running on EVM



Tools | MCU+ Academy

Easy-to-use training modules for Sitara MCU developers

- AM243x EVM Quick Start Guide
- SDK Setup Guide with LED Blink Example
- Learn MCU+ SDK Fundamentals
 - Projects and files structure
 - SysConfig Tool
 - Adding Drivers
 - Linker file, Multicore example
 - Debugging, Flashing tools
- Learn Advanced Topics
 - Benchmarking (CMSIS with Cycle Counter)
 - Multicore communication
 - Direct Memory Access

Welcome to MCU+ Academy

The MCU+ Academy is a great resource for developers to learn about the Sitara™ MCU Platform.

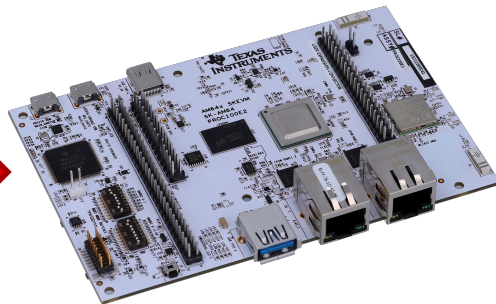
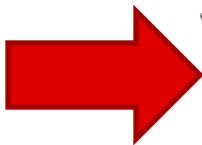
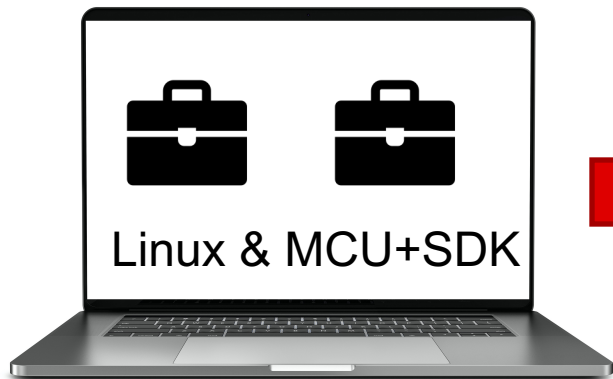
R5F development using CCS

- Start from template CCS projects in AM64x MCU+ SDK:
 - <mcu_plus_sdk_am64x_x_x_x_x>\examples\empty\am64x-evm
- Make sure to go through the [EVM set up](#) in the [Documentation](#)
- [MCU+ Academy](#):
 - Made for AM243x but applicable to AM64x in principle
 - Files that are specific for AM243x should be renamed for AM64x, e.g., library names referenced in part 4 should replace “am243x” with “am64x”.

MCU+ SDK Fundamentals

- Part 1: Introduction to MCU+ SDK
- Part 2: Understanding an MCU+ SDK Project
- Part 3: SysConfig Overview
- Part 4: Adding Drivers
- Part 5: Linker Command File Essentials
- Part 6: Debugging Basics
- Part 7: Understanding Multi-core Projects
- Part 8: Flashing

Demo



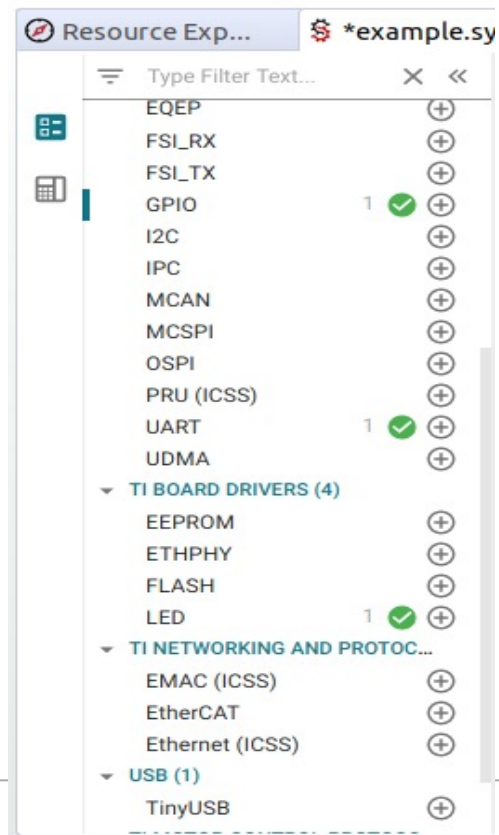
- Linux OS (Ubuntu)
- Largely Command Line
- make files
- kgdb/gdb
- printf()
- TFTP/NFS

- ~~Windows OS~~
- CCS GUI
- Projects
- GUI Debugging
- Console
- JTAG



MCU+ Differences for Linux

tos_ti-arm-clang/example.syscfg - Code



1

Add IPC

2

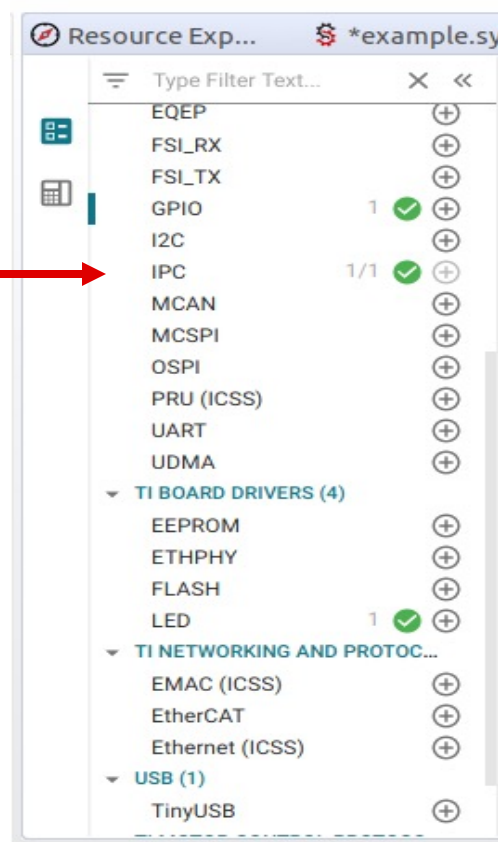
Remove or change UART

3

linker.cmd

Allocate IPC
Buffers

-0_nortos_ti-arm-clang/example.syscfg



Launching R5F application from Linux

- Steps needed to make a R5F application launchable from Linux:
 1. Add IPC to syscfg similarly to the benchmark demo:
`mcu_plus_sdk_am64x_x_x_x_x\examples\motor_control\benchmark_demo\am64x-evm\r5fss0-0_nortos\examples.syscfg`
 2. Make sure no resource conflict with Linux. For example, in the MCU+ Academy Part 4 project, UART should be removed from syscfg or a different UART instance than 0 should be used.
 3. Add .resource_table section to DDR in linker command file. Or simply reuse the linker command file from the benchmark demo:
`mcu_plus_sdk_am64x_x_x_x_x\examples\motor_control\benchmark_demo\am64x-evm\r5fss0-0_nortos\ti-arm-clang\linker.cmd`
 4. Build CCS project and strip all symbols from the .out file:
`C:/ti/ccs1040/ccs/tools/compiler/ti-cgt-armllvm_1.3.0.LTS/bin/tiarmstrip -o=<stripped binary file> <original .out file>`
 5. Copy the stripped .out file to Linux filesystem and set symbolic link
`/lib/firmware/am64-main-r5f0_0-fw, am64-main-r5f0_1-fw, etc.`

Processor SDK Linux Kernel

- Based off of [kernel.org Longterm maintenance](https://kernel.org/doc/longterm/)
 - Provides bug fixes for about 2 years from the community
- Move to new Longterm annually
 - Allows users to pick up new features and capabilities

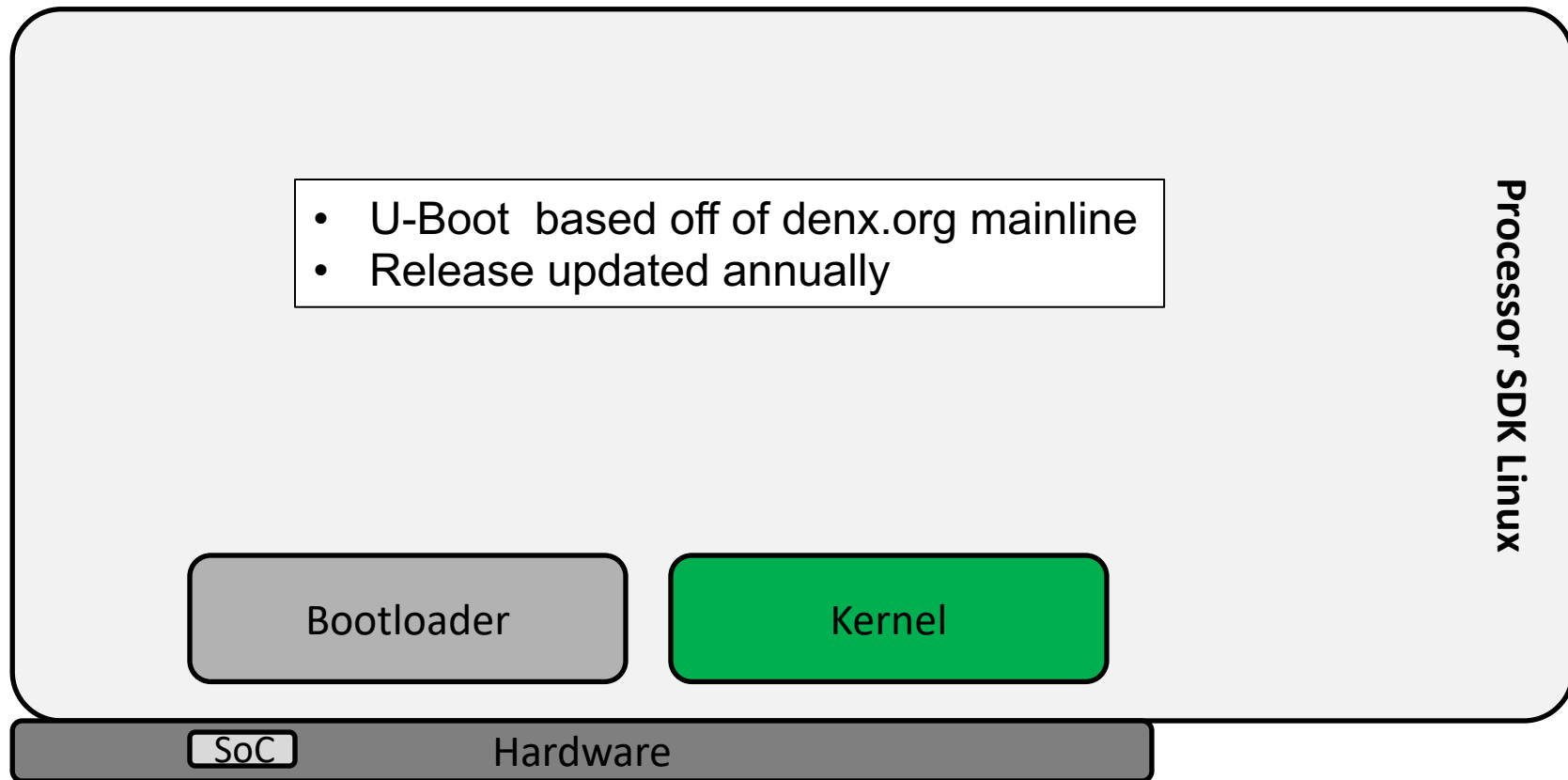
Processor SDK Linux

Kernel

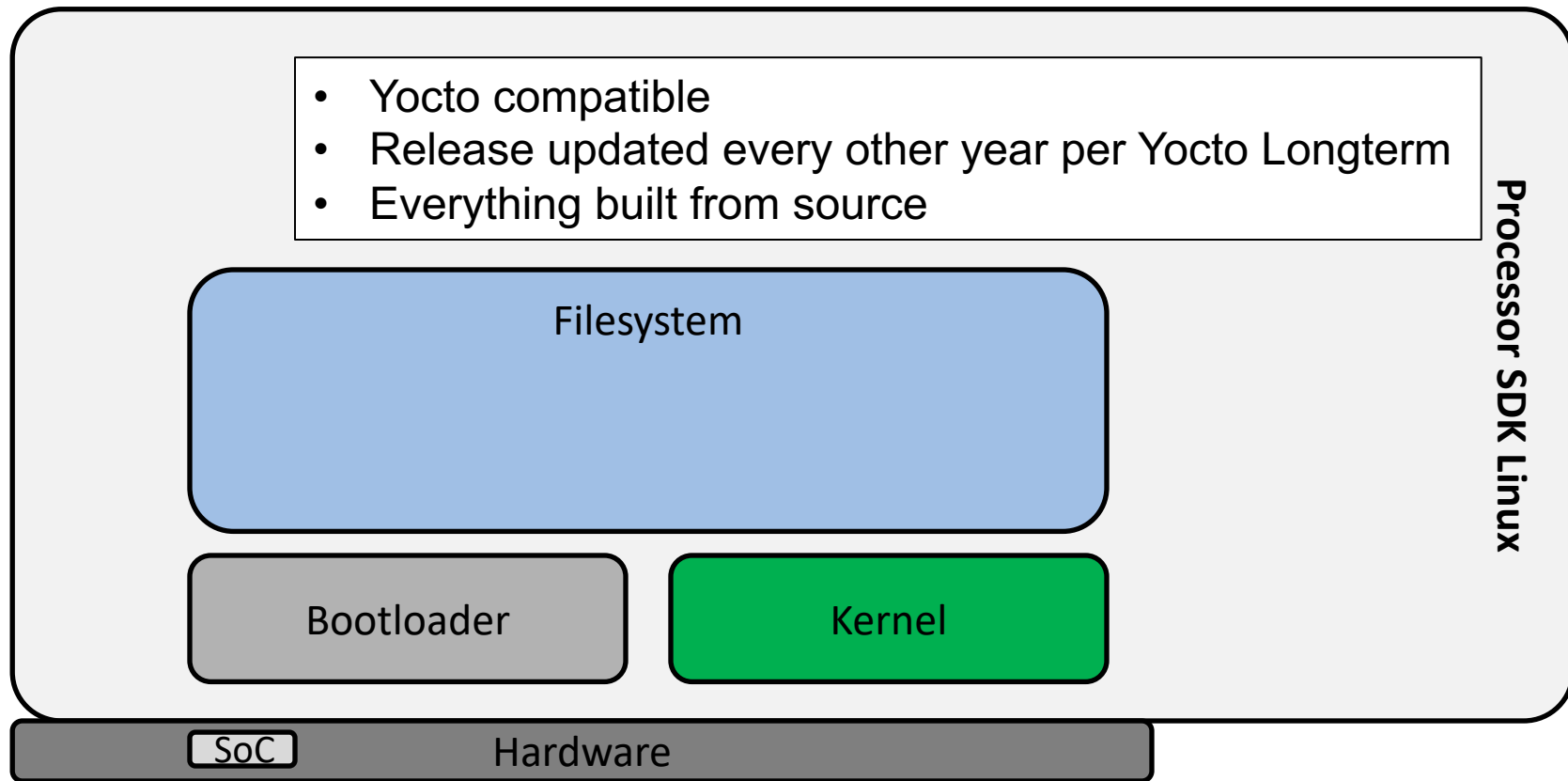
SoC

Hardware

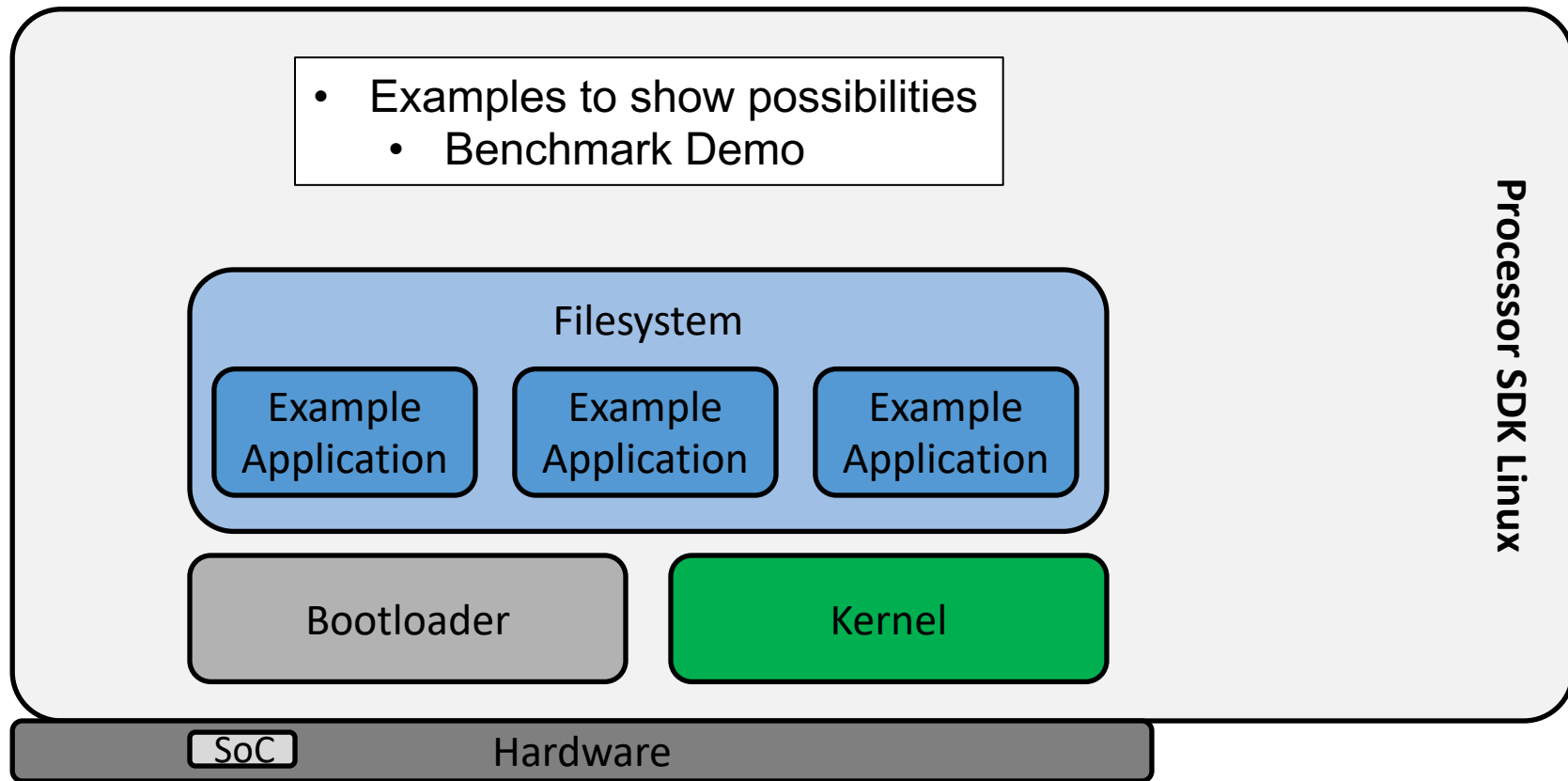
Processor SDK Linux Bootloader



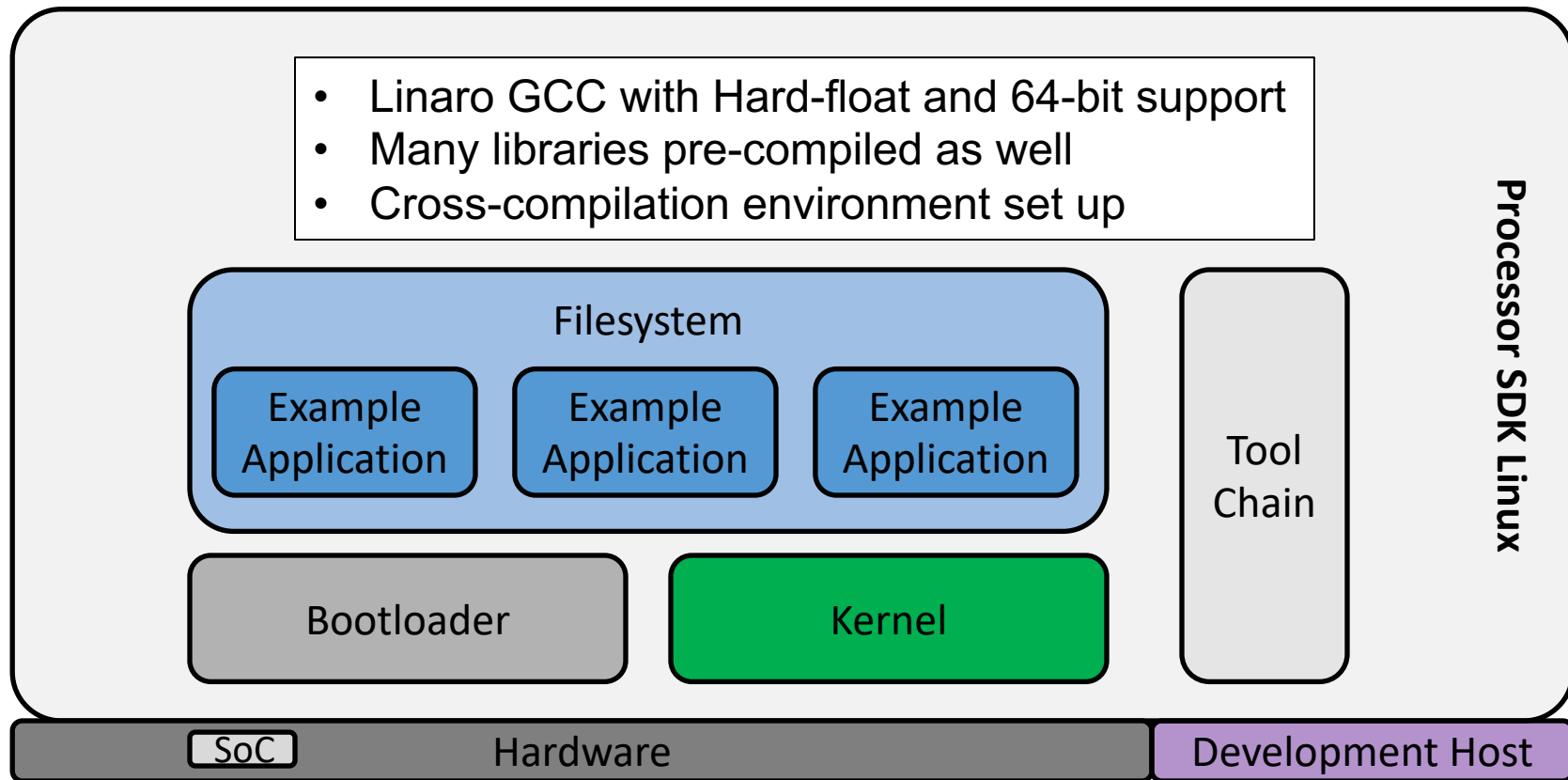
Processor SDK Linux Filesystem



Processor SDK Linux Example Applications

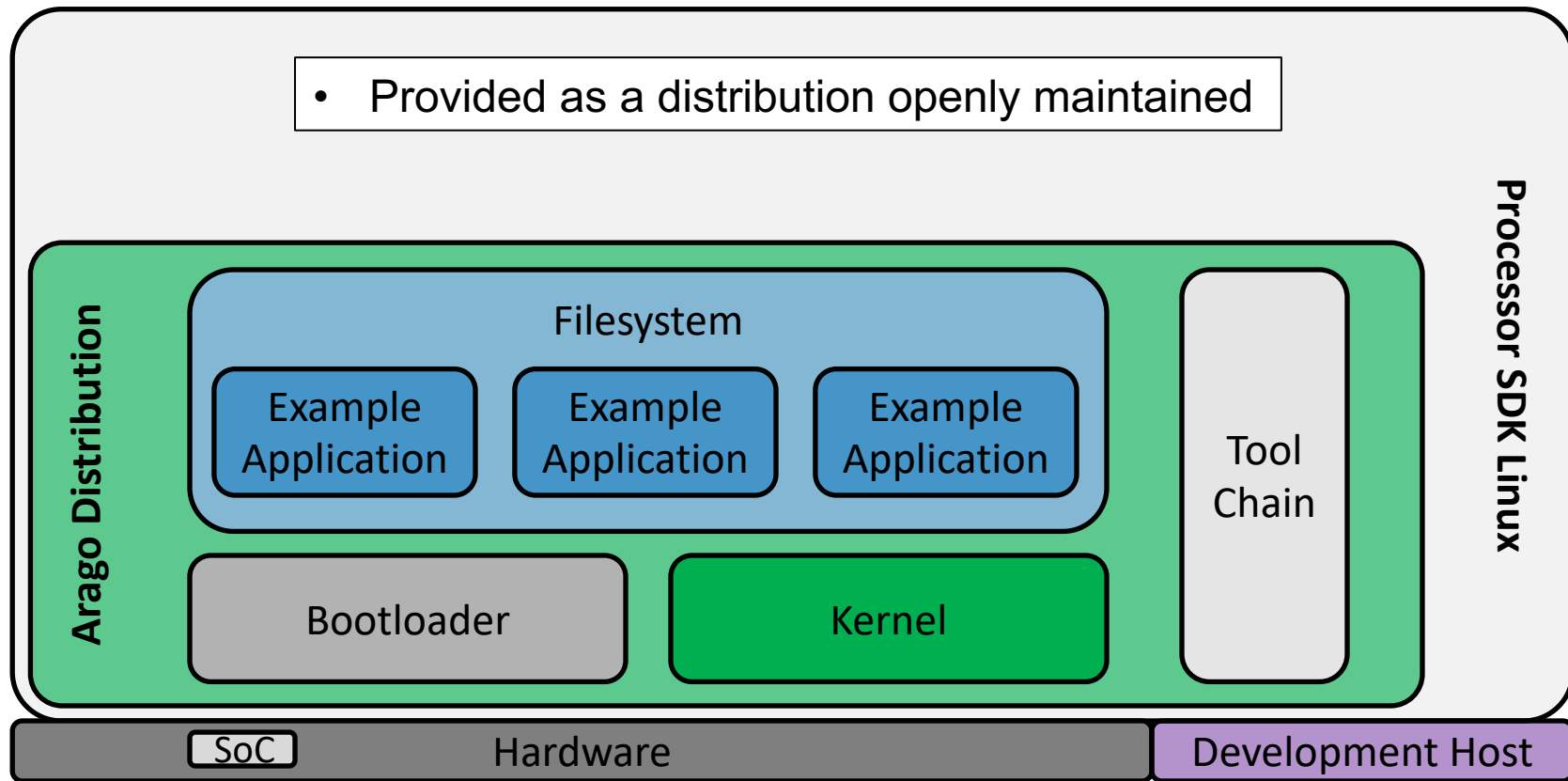


Processor SDK Linux Toolchain

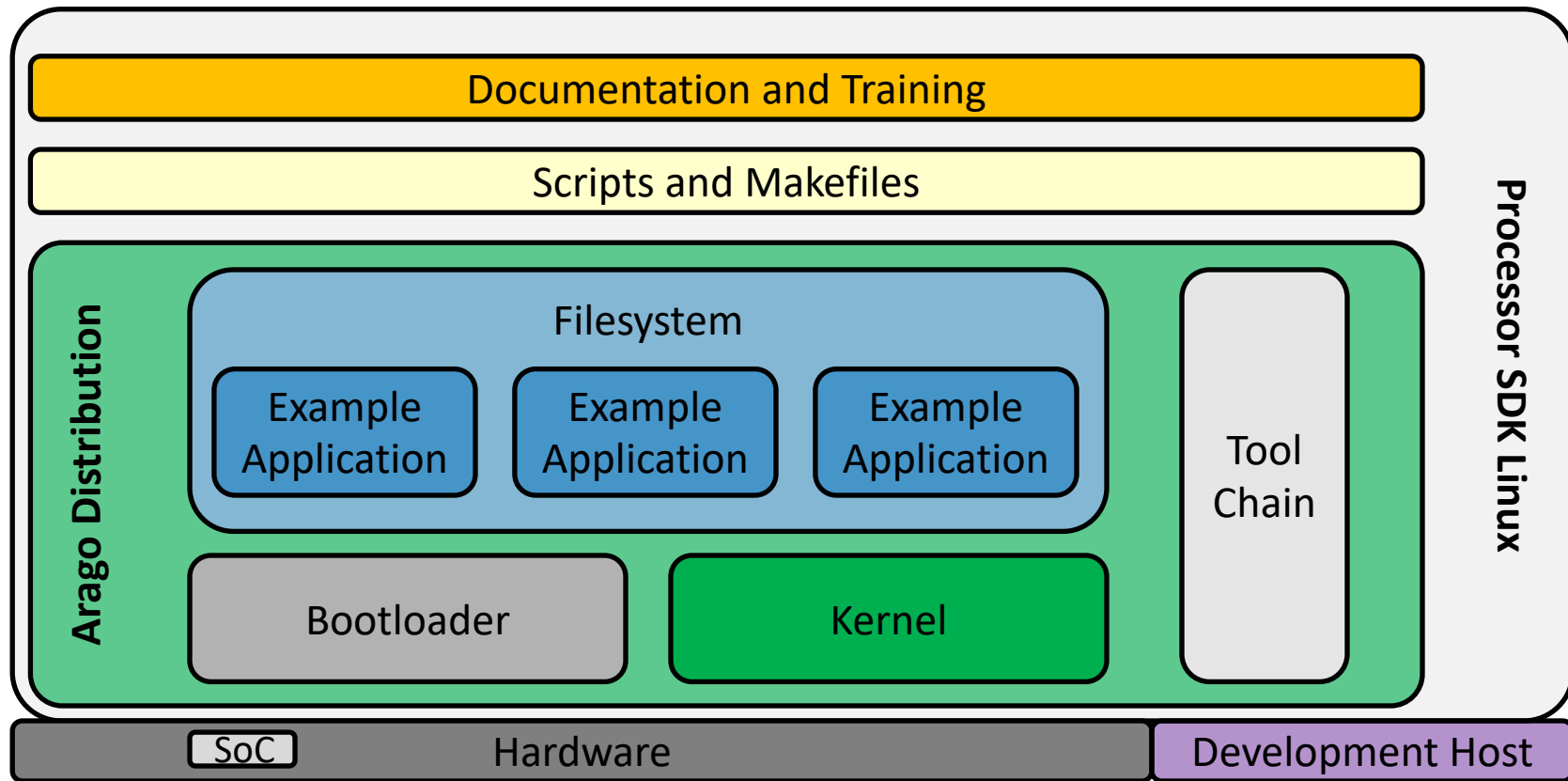


Processor SDK Linux Arago Distribution

- Provided as a distribution openly maintained



Processor SDK Linux Summary



Summary

- AM64x is a powerful, heterogeneous core processor to meet a variety of system needs
- SDKs and tools are provided to facilitate software development on all cores
- Training and examples are available to accelerate development

Future Webinars

- AM64x Multiple Industrial Protocols
 - October 19 at 10 AM CST and October 21 at 10 PM CST
- AM64x IPC
 - November 16 at 10 AM CST and November 18 at 10 PM CST
- More information, registration, and videos:
- <https://training.ti.com/process-monthly-webinar-series>

Backup

References

AM64x Linux Software Development Kit (SDK)

- <https://www.ti.com/tool/PROCESSOR-SDK-AM64X>

AM64x MCU+ Software Development Kit (SDK)

- <https://www.ti.com/tool/download/MCU-PLUS-SDK-AM64X>

AM64x Linux SDK User Guide

- <https://software-dl.ti.com/processor-sdk-linux/esd/AM64X/latest/exports/docs/devices/AM64X/index.html>

AM64x MCU+ SDK Documentation

- https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_00_00_21/exports/docs/api_guide_am64x/index.html

AM64x EVM:

- <https://www.ti.com/tool/TMDS64GPEVM>

AM64x SK board:

- <https://www.ti.com/tool/SK-AM64>