Computer System

❖ It is a machine designed to retrieve, process and store numerical data

❖ It can be a general-purpose or an embedded computer system

  ➢ But the basic principles of operation and the underlying architectures are the same

❖ It is composed of hardware and software

❖ The hardware executes the software

❖ The software controls the operation and functionality of the hardware

❖ The architecture of the softwares can be different

  ➢ E.g. AUTOSAR, Linux based general-purpose systems and etc.

  ➢ But the first layer in the software architectures is the

   **H**ardware **A**bstraction **L**ayer (HAL)
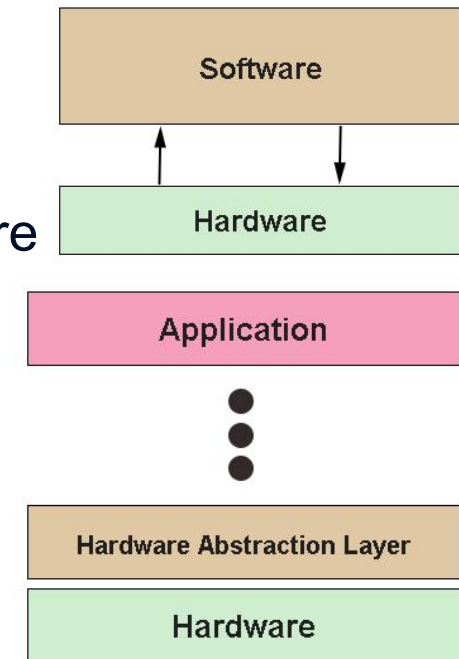
Hardware Abstraction Layer

❖ **HAL** is the first layer in the software architecture of a computer system

❖ It makes softwares hardware independent

❖ It increases portability and testability of softwares

❖ It is the only layer which directly interacts with the hardware

❖ It can for example contain

➢ Board Support Package of a microcontroller

➢ Firmware, device drivers, BIOS, bootloader and etc.

❖ A device **driver** is a software that controls and interacts directly with the device and provides some functionalities for the upper layers by some interfaces.
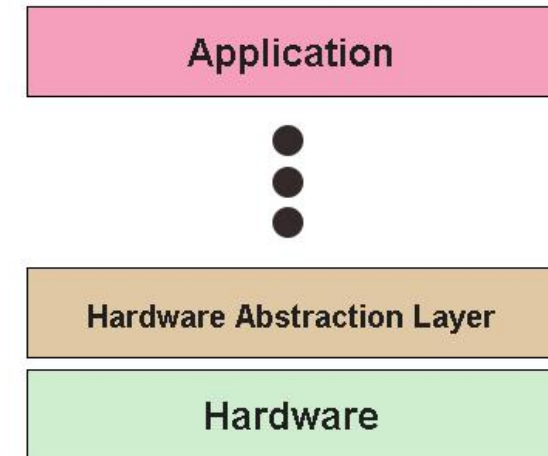
Interface Layer

❖ **Between the HAL and the Application layer, generally there is an interface layer**

  ➢ Provides services and abstractions for the applications

  ➢ Depending on the application, it has its own architecture

  ➢ In complex systems an OS is used in this layer

  ➢ E.g. AUTOSAR, Linux and etc.

Application Layer

Harware/Application Interface Layer

Hardware Abstraction Layer
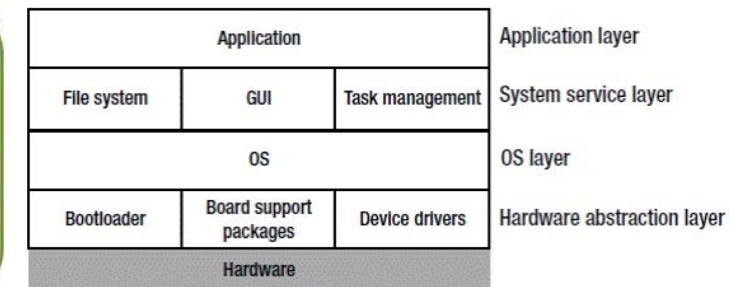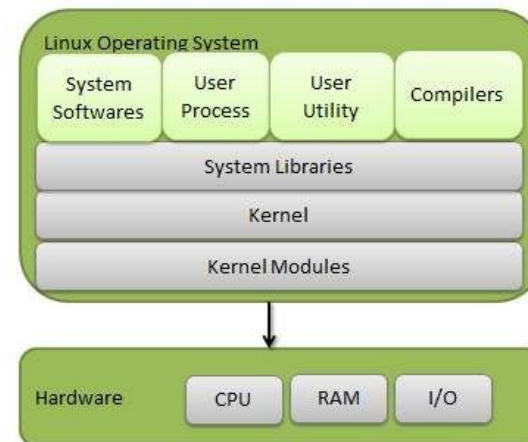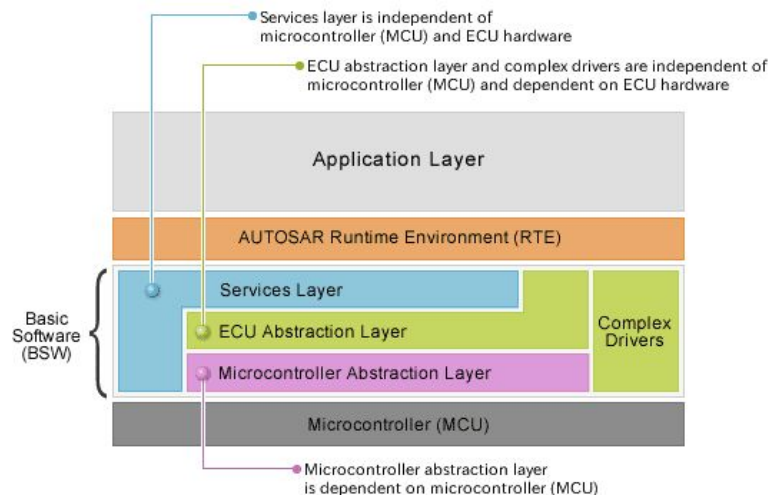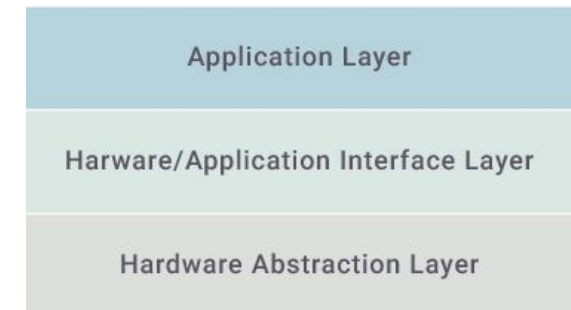
Typical Software Architecture
of an embedded system

Yrkes Akademin
Vi hjälper dig att lyckas!

Basic Computer System Architecture

❖ Three components connected to a bus system

➢ Processor

■ The computing and controlling part of a computer system which processes the data

➢ Memory

■ Is used by the processor to retrieve and store data

➢ I/O Devices

■ Are used by the processor to communicate with the external world

❖ Processor

➢ A processor is an electronic device capable of processing data based on an instruction set

➢ There are two major instruction set architecture (ISA) for processors

■ Complex Instruction Set Computer (**CISC**) processors; e.g. Intel x86 processors

■ Reduced Instruction Set Computer (**RISC**) processors; e.g. ARM processors like Cortex-M4
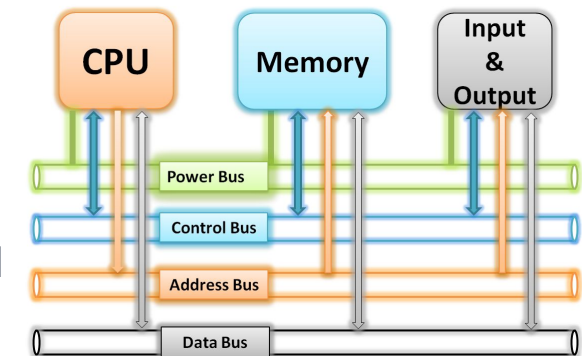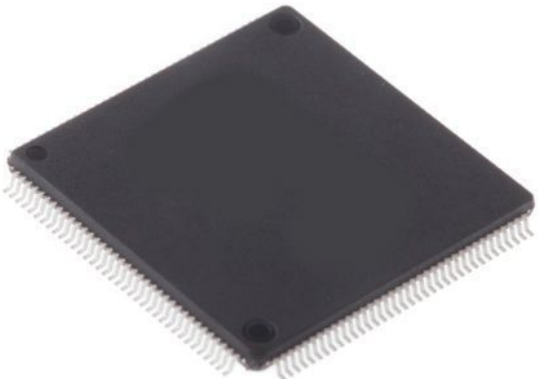
Computer System Architecture

Microprocessor vs. Microcontroller	
Microprocessor	**Microcontroller**
A computational unit. Nothing else	A computer system;Microprocessor, Memory, Storage and I/O and etc.
General purpose, non specific task	Specialized and used in embedded systems
High clock speed (> 1.5GHz)	Moderate clock speed (40-600 MHz)
No memory	On chip memory (Both RAM & ROM)
External memory controller	In circuit memory controller
No HW interfaces	Embedded HW interfaces
Requires a lot of support components	Can work almost alone
Runs an OS	Usually no OS
64 bit architecture	8, 16 or 32 bit architecture

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Computer System Architecture

Microprocessor vs. Microcontroller	
Microprocessor - Ex. Intel Core i9 - 9980XE	Microcontroller - Ex. Kinetis K66F (MK66FX1M)
18 cores - No I/O - Memory Controller - < 128Gb - Up to 4,5GHz - CISC - Complex Instruction Set Computing 64 bit architecture	- An ARM Cortex-M4 core, HW-RNG, HW-CRC < 100 I/O pins, inkl. CAN, SPI, I2C, UART, PWM, etc. 256 kb RAM, 1 Mb Flash 180 MHz - RISC - Reduced Instruction Set Computing 32 bit architecture

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Volatile and Non-volatile Memories

❖ Memory

➢ Generally there are two types of memory; often a mix is used within a single system

■ Non-volatile memory

● Requires no power to keep its content

■ Volatile memory

● Requires power to keep its content

2.7-3.6V 256Kb SPI Serial SRAM
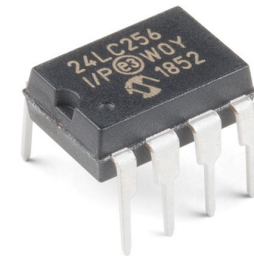
I2C EEPROM - 256k Bit (24LC256)

❖ RAM (Random Access Memory)

➢ Is the main and working memory in a computer system

➢ The processor may easily write/read data to/from it

➢ Generally there are two types of RAM: static RAM (SRAM) and dynamic RAM (DRAM)

➢ Is generally volatile; loses its contents when the system loses power

■ There are special nonvolatile RAMs that integrate a battery-backup system

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Volatile and Non-volatile Memories

❖ Cache memory

➢ Caches are implemented using high-speed memories

➢ Many processors have instruction and/or data caches to improve the performance

➢ Caches are often, not always, internal to the processors

➢ Caches are volatile
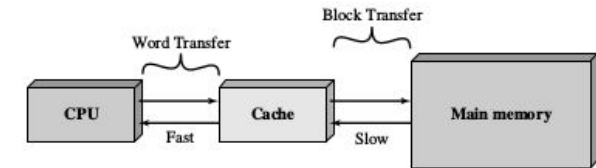
❖ Non-volatile memories

➢ Are used to store codes, data, configuration and calibration values

■ ROM (Read-Only Memory)

■ EPROM (Erasable Programmable Read-Only Memory)

■ EEPROM (Electrically Erasable Programmable Read-Only Memory)

■ Flash Memory

Volatile and Non-volatile Memories

- ❖ ROM (Read-Only Memory)
  - ➢ Is a large array of diodes
  - ➢ The unwritten bit state for a ROM is all 1s (0xFF)
  - ➢ A ROM burner is used to load data into a ROM
    - ■ By burning some diodes using passing a sufficiently large current through them
  - ➢ Can be burned only once.

- ❖ EPROM (Erasable Programmable Read-Only Memory)
  - ➢ It is an array of floating-gate transistors individually can be programmed
    - ■ By supplying higher voltages than those normally used in digital circuits
  - ➢ An EPROM can be erased by exposing it to strong ultraviolet light source
  - ➢ An EPROM programmer is used to program an EPROM
    - ■ The chip must be removed from the circuit to be erased

How an EPROM Works

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Volatile and Non-volatile Memories

❖ EEPROM (Electrically Erasable Programmable Read-Only Memory)

➢ Are organized as arrays of floating-gate transistors. Usually are byte-programmable

➢ Can be programmed and erased in-circuit, by applying special programming signals.

➢ Is slow and Its capacity is significantly smaller than standard ROM (few kilobytes)

➢ Is typically used by microcontrollers to store system parameters and calibration values

➢ Has a limited life for erasing and reprogramming (100,000 - 1,000,000 times)

❖ Flash Memory

➢ Is the newest ROM technology

➢ Is a type of EEPROM designed for high speed and high density

➢ Is organized as sectors (typically 512 bytes or larger)

➢ Number of write cycles are limited (often more than 10,000)

➢ Many microcontrollers have both: flash memory for the code , and an EEPROM for parameters

Volatile and Non-Volatile Memories - Exercise

❖ On Teensy 3.5 there is a 4 kB EEPROM.

❖ In Arduino you can use **EEPROM.h library** to read and write data to Teensy's EEPROM

❖ **Exercise 17**: Use the EEPROM.h library to develop an eeprom driver module in C

➢ The module shall have

■ An **eeprom_driver_init** function to initialize the module

● The client shall be able to pass a boolean to this function if he/she wants to clear the EEPROM.

■ An **eeprom_driver_write** function to write any data type to the EEPROM.

■ An **eeprom_driver_read** function to read any data type from the EEPROM.

➢ Use TDD and follow the dual targeting strategy to develop and test your module

➢ Use the link-time faking technique and a layer over Arduino to make the test doubles

➢ *You know that pointers to any data type can be converted to **uint8_t** pointers.*

I/O Devices - Sensors

❖ I/O in an embedded system

➢ GPIOs, ADCs, DACs and communication interfaces are used to provide

■ **Input** data from peripheral devices, for example **sensors**, for the system

■ **Output** data in order to communicate to or control peripheral devices like **actuators**
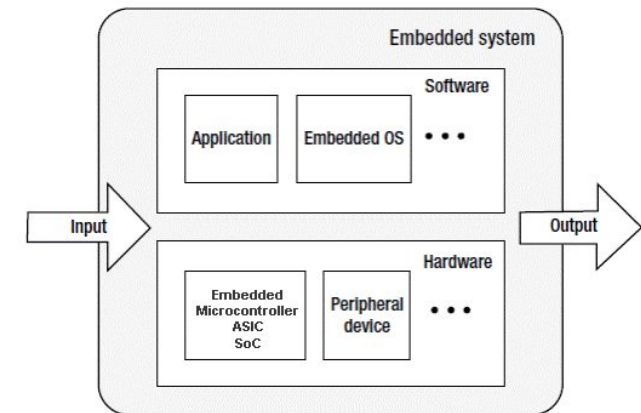
❖ There are two types of sensor

➢ Analog sensor

■ Gives analog signal and measures almost anything!!

■ Signals can be voltage, current, resistance, or etc.

■ E.g. TMP36 sensor, PN-12110215TC-8 Level Sensor and etc.

➢ Digital Sensor

■ Gives digital signal using for example digital pins, I2C or SPI bus and etc.

■ Usually is more complex sensor. E.g. MCP9808 Temperature Sensor

I/O Devices - Actuators

❖ Actuators

➤ Hardware devices which make something move or actuate

➤ Convert an electrical control signal into physical action; e.g. an electric motor

➤ Different types of actuator: electrical, magnetic, hydraulic actuators, pneumatic and etc.

❖ In a microcontroller I/O pins are arranged in ports mapped to the memory

➤ 8 or more pins together forming a byte, word or dword

➤ I/O pins can be used as serial or parallel Input/Output signals

➤ Normally an I/O signal is serial, meaning 1 bit at a time.

➤ Parallel I/O means we transfer several bits at a time, e.g. 8-bit

■ I/O ports can be written or read as an entity

■ Faster than serial but hard to synchronize

How a Stepper Motor works?
What is a Stepper Motor?
Stepper Motors with Arduino
Big Stepper Motors with Arduino

How a Servo Motor Works?
Control Servos using Arduino
Using Servo Motors with Arduino
Understanding and Using Servos
Controlling a Servo with Arduino

Yrkes Akademin
Vi hjälper dig att lyckas!

I/O Ports on Teensy 3.5

❖ On Teensy some registers are used to read/write from/to ports

- ➤ **PORTx_PCRn** is used to configure pin **n** in port **x**

    - ■ E.g. PORTD_PCR0 = (1 << 8); // Pin 0 in port D is a GPIO pin

- ➤ **GPIOx_PDDR** is Data Direction Register for port x

    - ■ 0 for input and 1 for output; e.g. GPIOD_PDDR = 0xFF;

- ➤ **GPIOx_PDOR** is used to set a logical level of output in port x.

    - ■ 0 for LOW and 1 for HIGH; e.g. GPIOD_PDOR = 0xFF;

- ➤ **GPIOx_PDIR** is where pin status is read for port x

    - ■ A 1 indicates a high level and a 0 indicates low level on input pins in port.

- ➤ *Look at the manual*

Yrkes Akademin
Vi hjälper dig att lyckas!

I/O Ports - Example

❖ Using TDD and the dual targeting strategy develop a single-instance module (leds_driver)

according the requirements below:

➢ Use a GPIO port on Teensy 3.5

➢ The driver controls 8 two-state LEDs

➢ LEDs are memory-mapped to a byte (8 bits) at an address to be determined

➢ A 1 in a bit position lights the corresponding LED; 0 turns it off

➢ The LSB corresponds to LED 1 and the MSB corresponds to LED 8

➢ At power-on, the LEDs are on. They must be turned off by the software

➢ The user of the driver can query the state of any LED

➢ The driver can turn on or off any individual LED without affecting the others

➢ The driver can turn all LEDs on or off with a single interface call

➢ The code coverage shall be 100%

Yrkes
Akademin
Vi hjälper dig att lyckas!

Device Driver - Exercise 18

❖ As you know in Arduino there is **random()** function to generate a **pseudo-random** number.

   ➢ If you look at the implementation of this function you see that a simple software algorithm has been implemented and the generated number is not a good random number and it is possible to guess it.

❖ In the microcontroller on Teensy 3.5 there is a **Random Number Generator Accelerator** unit.

   ➢ For the details of the **RNGA**, read chapter 34 of the manual

   ➢ Develop a driver module(**rnga**) for the RNGA according to the requirements below:

   ➢ The driver shall have a function to initialize the RNGA.

      ■ *Note that the RNGA uses System Clock Gating Control Register 6 (**SIM_SCGC6**).*

   ➢ The driver shall have a function to start the RNGA

   ➢ The driver shall have a function to seed the RNGA with an entropy

   ➢ The driver shall have a function to generate a random integer number in the range of [min., max.]

   ➢ The driver shall have a function to stop the RNGA.

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Interrupts

❖ There are two ways to handle an event

➢ Polling; checking for the condition periodically or continuously

➢ Interrupt: the processor executes a special piece of code if the event occurs

■ This piece of code is called interrupt handler or interrupt service routine

❖ Interrupt

➢ Is a hardware feature and many interrupts are available for conditions like

■ Pins changing, data received, timers overflowing, errors and etc.

➢ Every interrupt has a flag bit

■ It is set by hardware when the interrupt trigger condition occurs

■ Normally it is automatically reset when the interrupt service routine is called

■ The flag bit is set even if interrupts are not used

● Polling can read the flag bit to check if the event has occurred

Yrkes
Akademin
Vi hjälper dig att lyckas!

Interrupts ...

❖ Every interrupt also has a mask bit to enable/disable interrupts individually

❖ Also a global interrupt enable/disable bit for all interrupts that have their mask bits set

❖ Prioritized interrupts are possible; a value 0 - 255. **0** for the highest and **255** to the lowest priority

❖ Note that not all devices are interrupt-safe. E.g. an SD card.

❖ Nested interrupts are possible, but should be avoided by

➢ Clearing the global interrupt flag in the beginning of interrupt service routines and

➢ Setting the global interrupt flag in the end of the interrupt service routines

❖ Keep all interrupt service routines short and simple

➢ Interrupts change the timing of programs

❖ Shared variables must be declared as volatile and be protected with cli() and sei()

❖ **Example**: Connect a push button to a digital pin on Teensy and activate the internal pull-up resistor for the pin. Debounce the button using an interrupt.

Direct Memory Access (DMA)

❖ IO devices are very slower than the processor

❖ If we want to transfer a large block of IO data byte by byte

using the the processor, for each byte it should read the byte

from IO, load it into a register and then store it in a location

of RAM. It is clear that this process is an overload on the CPU and it is not efficient.

❖ DMA (**D**irect **M**emory **A**ccess) is a special device which

is used to transfer large blocks of data between two

sections of memory, or between memory and I/O devices.

❖ A DMA Controller, performs high-speed transfers

between memory and I/O devices.


IO Access Without DMA

IO Access Using DMA
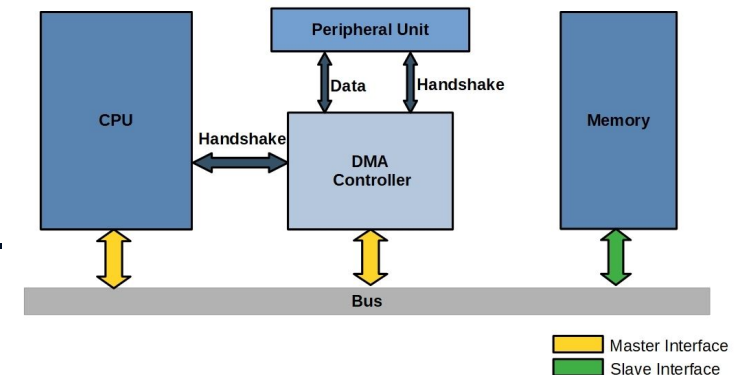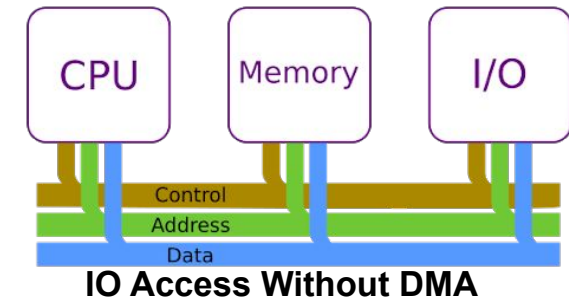
Direct Memory Access (DMA)

❖ A DMA bypasses the processor by setting up a channel between the I/O device and the memory, and data is read from the I/O device and written into memory without needing to involve the processor to execute code to transfer data byte by byte (or word by word). And during data transferring by the DMA, the CPU is free to do something else and in this way the DMA improves the performance dramatically.

❖ When the CPU wants to transfer a large block of data between different memory locations or between memory and I/O devices, it makes a request to the DMA by specifying the source, destination, and size of the data, as well as other parameters, and for a short time the DMA takes control of the bus system (data and address buses) and the DMA transfers the data. When the data transfering is completed, the DMA releases control of the buses and generates an interrupt.
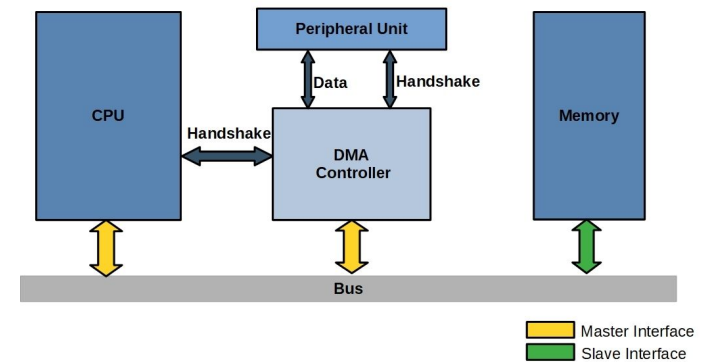
Debugging

❖ Debugging is the process of finding and resolving problems within a computer program

❖ Debugging of embedded softwares is not like debugging of softwares running on a PC

❖ To debug embedded softwares generally there are two methods

➢ Using special tools and protocols

■ A debugger software, an adapter like J-Link and standards like JTAG and SWD

■ JTAG (Joint Test Action Group) is an industry standard (IEEE 1149.1) for programming, debugging and testing ICs

■ SWD (Serial Wire Debug) is used by ARM for programming and debugging ARM microcontrollers

➢ Using the interfaces and I/O of the microcontroller

■ E.g. A serial communication port, some LEDs connected to GPIOs and etc.

➢ Debugging can change the execution timing because of the overhead it adds to the code

Yrkes
Akademin
Vi hjälper dig att lyckas!
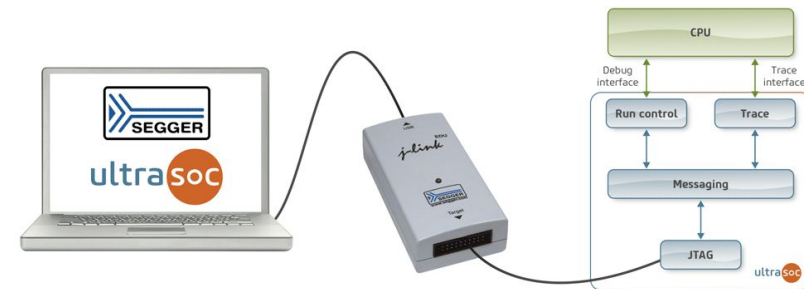
Calibration

❖ Calibration is the process of understanding of how a device behaves

➢ Measurement of a device against a reference and establish a relation between the measured value and the reference value

❖ In an application, calibration is tuning of its behavior

❖ A variant of calibration is option handling

➢ Option handling often is used to enable or switch between features

❖ There are two main types of calibration

➢ Static: tuning is done statically by using predefined values

➢ Dynamic(adaptation): tuning is done by the system itself and it can learn how to behave

■ Instead of setting static limits, the system can learn the limits, either by itself or triggered during service or manufacturing. E.g. A sensor or an actuator limits

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Calibration

❖ Why calibration?
  ➢ It handles configurations and adapt the functionality without changing the software
    ■ It means less releases, less variants of the software and the software can be reused
  ➢ It allows us to do the adaptation of the software late in the project, without affecting the software development and quality assurance
  ➢ It reduces the costs
❖ Modern engine management systems have about 40.000 calibration params to be tuned
❖ By calibration, generally a software can handle
  ➢ Different variants and configuration, like EU or US specifications
  ➢ Tuning of parameters - Defining behaviour in different conditions
  ➢ Options handling - Manufacturing configurations. E.g. MT or AT

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Calibration

❖ Calibration Methodologies

➢ Manual calibration

➢ Script based

➢ Testbed automation

■ Running the system in all conditions

➢ Model based calibration

■ Using simulation and statistical methods to find the optimal calibration

❖ Common standards in the automotive industry

➢ **ASAM MCD-1** describes how to read and exchange measurement and calibration data with the system

■ E.g. **CCP** defines a CAN-specific calibration protocol between a calibration tool and an ECU.

➢ **ASAM MCD-2** describes different protocols that can be used during development

■ E.g. **MC** defines the description format of the internal ECU variables used in measurement and calibration

YA Yrkes Akademin
Vi hjälper dig att lyckas!

Calibration

❖ Types of calibration values

➢ Engineering values; e.g. static limits

➢ Raw values; e.g. select variant of a feature

➢ Strings; e.g. software identifiers

➢ Curves: limit values depending on 1 input condition; e.g. temperature dependent limits

➢ Map; limit values depending on 2 or more input condition

■ An engine torque depending on both the engine speed and the active gear

❖ How to store calibration data?

➢ **Flash Memory** is suitable for data that is not changed

■ Flash memory does not allow to write to the memory in reading mode

■ In some cases the PROGMEM modifier can also be used

➢ **EEPROM** is used for adaptation or parameters written to the device during runtime
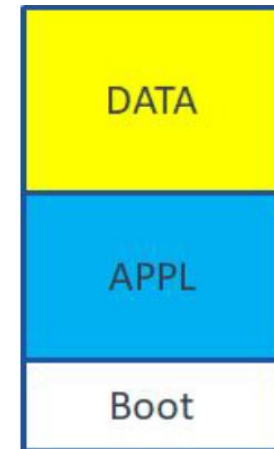
Computer System Components and Architecture

❖ Some useful links

➢ [Computer Organization and Architecture Tutorials](#)

➢ [Embedded Systems Tutorial](#)

➢ [Typical Software Architecture](#)

➢ [Operating System - Linux](#)

➢ [Embedded Systems Architecture, Device Drivers](#)

➢ [Typical Software Architecture](#)

➢ [ROM, EPROM, and EEPROM Technology](#)

➢ [Direct Memory Access (DMA) Introduction](#)

➢ [Interrupts in Arduino](#)

➢ [Arduino Attachinterrupt](#)

➢ [Using Interrupts on Arduino](#)

➢ [Processor Interrupts with Arduino](#)