

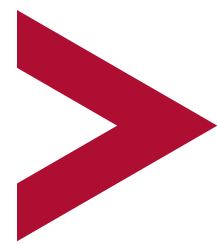
Winter
soldiers



AUTOSAR

Bootcamp

VECTOR



Cancerbero generation

Include: Certified Embedded Associate AUTOSAR Classic



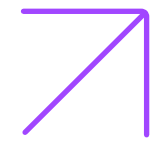
01

Editorial

If there's something that distinguishes our Bootcamp, it's that with each edition it improves, adding new topics, refreshing the old ones, learning from past lessons, and placing a strong emphasis on ensuring participants truly learn. To achieve this, we continue with our self-study approach, providing them only what's necessary to guide them but without limiting their creativity, while also helping them expand their capabilities.

The entire training focuses on four core concepts: **C programming, Microcontrollers, AUTOSAR, and Software Development**. Over the course of 6 months, you will consistently practice and continue learning these concepts, both individually and as part of a team.

We are staunch opponents of PowerPoint; 90% practical and 10% theoretical.



02

Badges

As a way to recognize your effort, but above all your level of knowledge, throughout the bootcamp you will have a series of badges that you must earn.



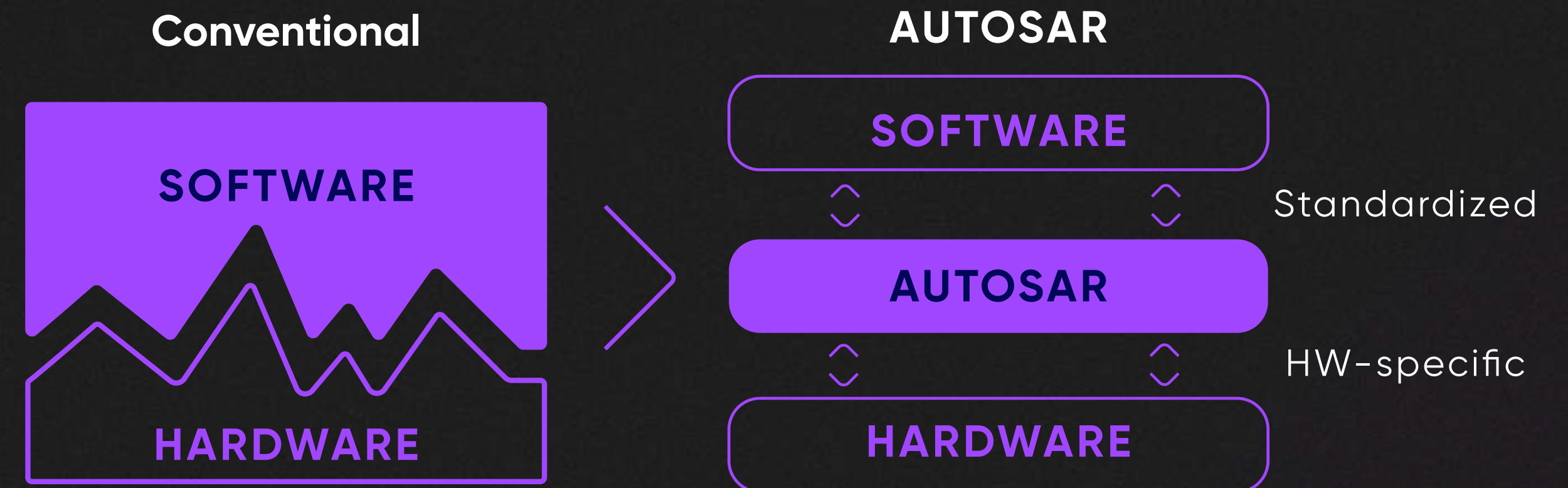


03 First phase

Enter AUTOSAR

If we're going to program microcontrollers in an automotive style, let's do it using AUTOSAR and use one specifically for this purpose: the NXP S32K144 with a Cortex M4F CPU capable of ASIL-B.

You will learn what AUTOSAR is and the MCAL layer. Additionally, you will work on configuring drivers such as Mcu, Port, Dio, Gpt, Adc, Pwm, and Det using the EB Tresos tool from Elektrobit, one of the most widely used tools in the industry.



Port y Dio

Introduction to EB Tresos and configuring your first BSW (Basic Software) driver. You will learn how general-purpose pins function and how to configure them. You'll also understand how AUTOSAR conceptualizes them and, most importantly, how to read AUTOSAR and microcontroller documentation to configure them in EB Tresos.

Mcu, Interrupts y Gpt

Setting up the microcontroller clocks with the Mcu driver, understanding and managing interrupts, using the CDD Platform driver for interrupt configuration in AUTOSAR. Using Hardware Timers to establish time periods. AUTOSAR manages a specification for using microcontroller timers exclusively for timing periods and notifications in our Gpt Driver.

Adc y Pwn

To your previous state machine, you will add two potentiometers that control the blinking of LEDs. You will read the potentiometers using ADC channels, which you will configure using EB Tresos according to the AUTOSAR standard and the microcontroller datasheet

Error detection

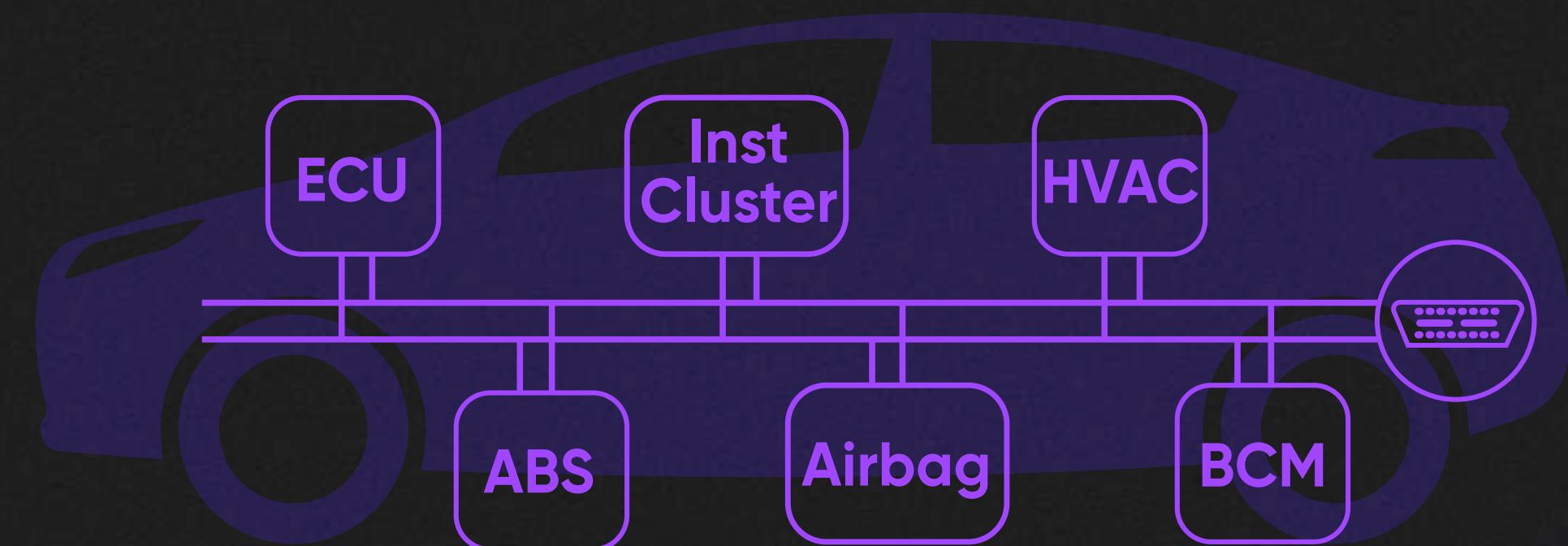
Configuring software units in AUTOSAR can be a complex task that easily leads to errors. To help detect errors, AUTOSAR defines the Development Error Tracer (DET) module, which we will configure to work with pseudo printf and display potential errors for us.



04 Second phase

Total CAN

Controller Area Network, commonly abbreviated as CAN, is the factor standard for communication between different computers within a vehicle, and understanding it is essential. In this section, we will focus exclusively on this protocol to expand our skills as software engineers. We will learn about concepts such as SDU (Service Data Unit), ICS (Internal Communication Service), and PDU (Protocol Data Unit) for message definition and management.



CAN-BUS

Controller Area Network

SPI

The SPI bus is not part of CAN itself, but it is part of the CAN Stack, a very simple serial protocol that AUTOSAR extends with concepts such as Channels, Jobs, and Sequences, along with levels of synchronous and asynchronous operation.

CAN Protocol

CAN bus, CAN frame structure, setting time quanta for data transfer calculation, use of filters for message reception, receive and transmit buffers. Using the FlexCAN peripheral of the S32K114, CAN in classic and FD mode, standard and extended IDs.

CanDrv

Configuring the CAN MCAL driver, AUTOSAR concepts for CAN such as Hardware Object Handlers (HOHs) for reception and transmission, configuration of HOHs and controllers. Understanding PDU (Protocol Data Unit) and SDU (Service Data Unit), CAN errors, and introduction to the AUTOSAR CAN Stack.

CanIf

Setting up a second CAN controller connected to an external configurable transceiver involves configuring the AUTOSAR CanTcrv driver.

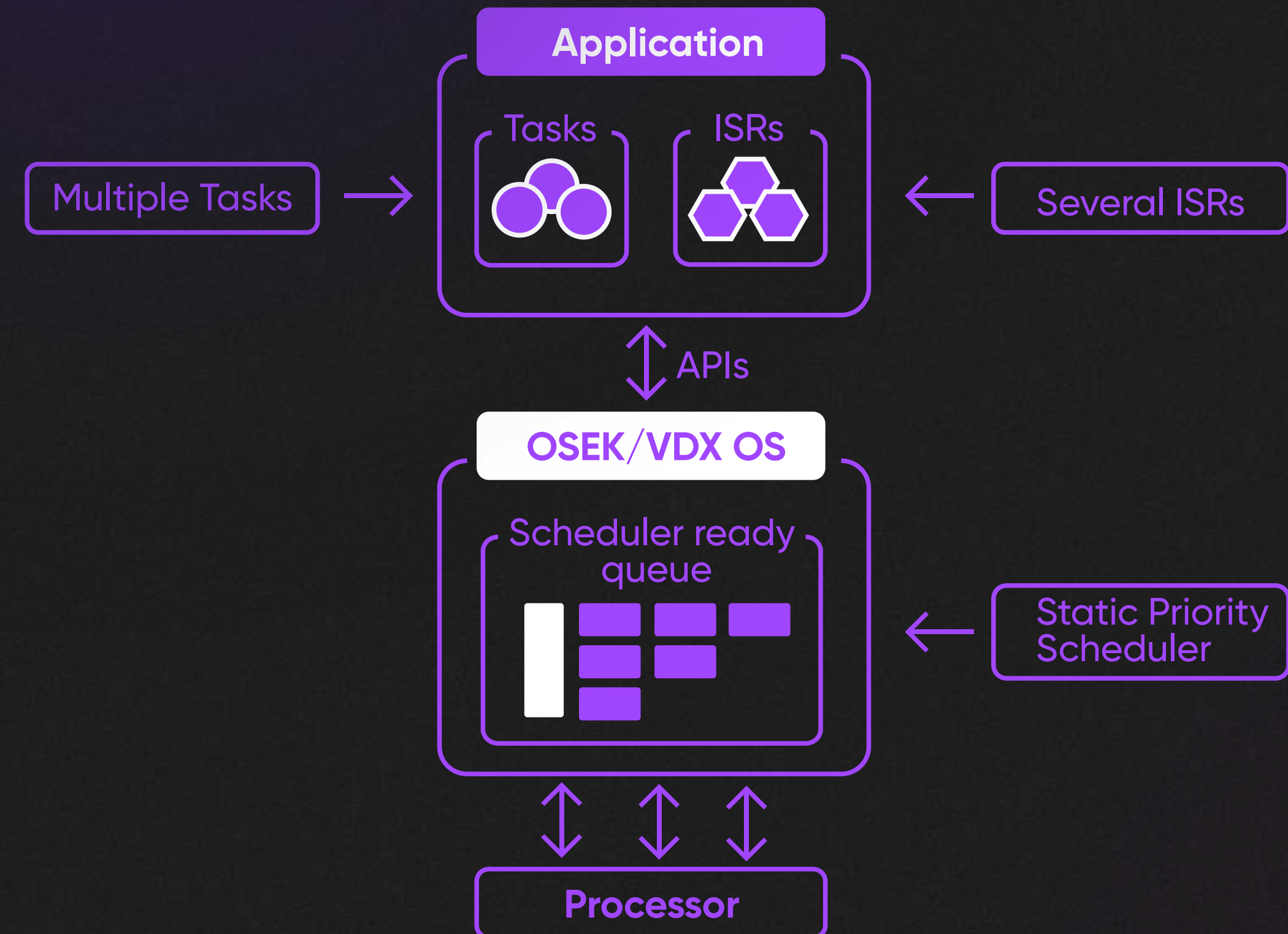
Defining PDUs in the CanIf layer and creating functions to handle the CAN transceiver with the Can Interface.



05 Third phase

OSEK/VDX

It's time to move beyond the MCAL layer and delve into the RTE and ASW layers to learn software concepts, components, ports, virtual buses, and how to structure our application with a strong AUTOSAR style. You will elevate your programming skills by adding new functionalities such as a clock, alarms, and Functional Safety (FuSa). On the software management side, you will create your first pull requests



Tasks and events

The foundation of any program with operating systems lies in tasks. We will learn to define them in the OSEK operating system and configure their attributes such as priority, number of activations, and task type. We will also cover the concept of events and how events help synchronize tasks.

Messages

Tasks can be considered as autonomous programs, but an application consists of multiple tasks interacting with each other. OSEK defines the concept of "message objects" for both reception and transmission between tasks. These message objects are also used as notification mechanisms to trigger events, call callbacks, and activate tasks.

Tick and alarms

Setting up the Tick to establish the time base for the entire system. OSEK defines the concept of timers as alarms, which are mechanisms used to trigger the execution of tasks, events, or callback functions. Additionally, we will see how these alarms can be modified at runtime.

Resource and Interrupt Handling

We will learn how to manage system resources efficiently to prevent data corruption issues. We'll also explore the OSEK Priority Ceiling protocol, a mechanism for exclusion, and understand how interrupt communication is established with tasks awaiting messages



06 Fourth phase

Certified Embedded Associate AUTOSAR Classic

VECTOR



A New Beginning with Vector Configurator and Vector Developer tools, this time you will have access to a virtual machine with full Vector virtual target for a period of 5 weeks, where you can follow self-training material and with them to access the possibility to get officially certified.

Gain an understanding of the Vector software delivery and how to start

Introduction to DaVinci Configurator Classic and DaVinci Developer Classic

Basics about AUTOSAR OS and communication

Fundamental configuration of the Mode Management and how to store non-volatile data

AUTOSAR fundamentals – cluster structure of the basic software (BSW), the fundamentals of the RTE, and basic application development

Basics about Diagnostics

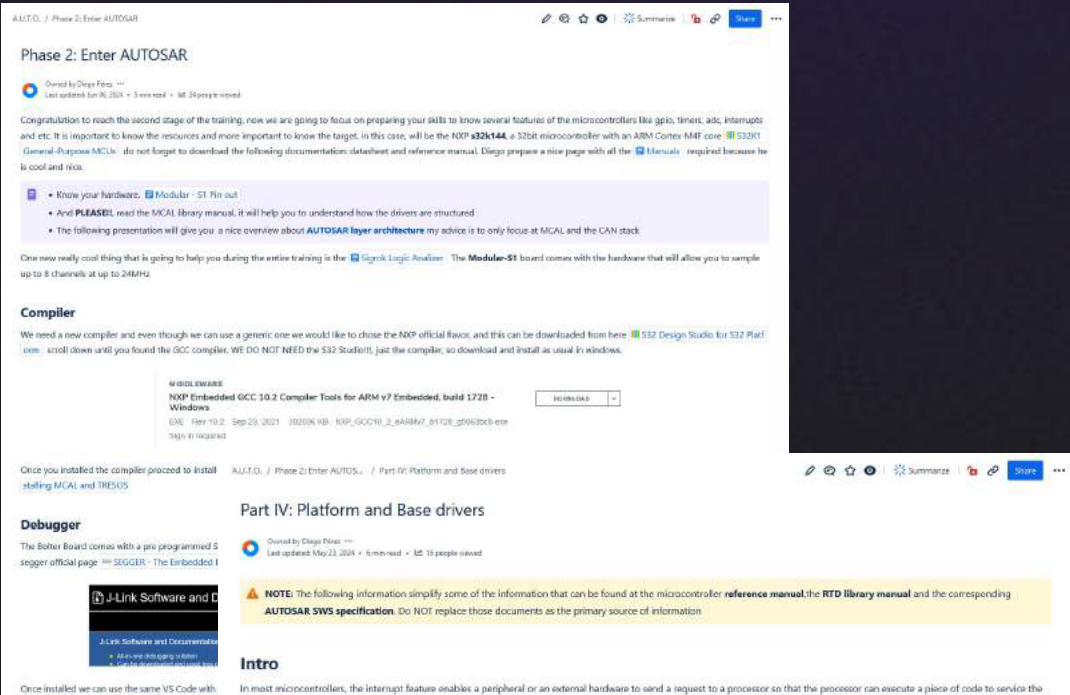
Overview of the Vector products and supporting services



07 Library

Throughout the bootcamp, you will have access to our library with over 100 articles, guides, and application notes to aid in your learning.

Additionally, 130 exercises, 60 examples, and a project with over 5000 lines of code will serve as a guide and training on everything you need to know and learn during the bootcamp.

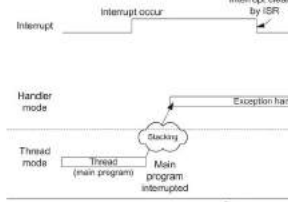


Intro

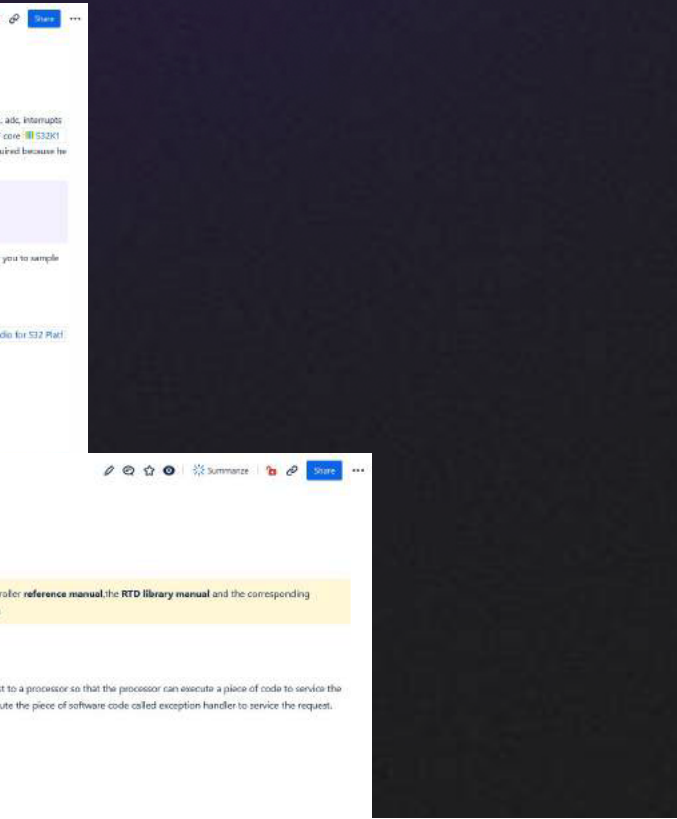
In most microcontrollers, the Interrupt feature enables a peripheral or an external hardware to send a request to a processor so that the processor can execute a piece of code to service the request. The process involves suspending the current executing task, or wake up from sleep mode, and execute the piece of software code called exception handler to service the request. After the request is serviced, the processor can then resume the previous interrupted code.

Interrupt Flow:

- The peripheral triggers an interrupt on the processor.
- The processor suspends the execution of the program flow, Thread Mode.
- The general purpose registers of the processor (including the Program Counter) are saved, Stacking.
- The processor executes the interrupt service routine (ISR) of the triggering peripheral, Handler Mode.
- The saved processor registers are retrieved and restored, Unstacking.
- The processor returns to resume the previous program flow.
- The processor knows exactly which instruction to resume as the previous Program Counter was saved prior.



NOTE: Thread mode is the processor resumes the main program while Handler mode is the processor



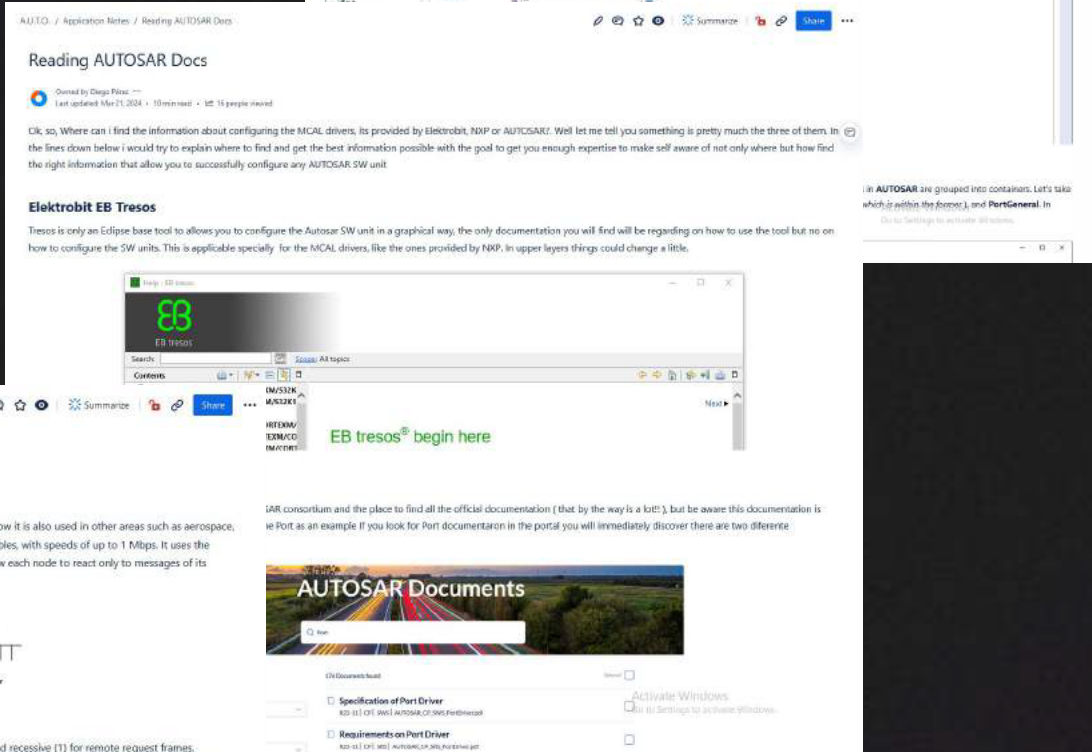
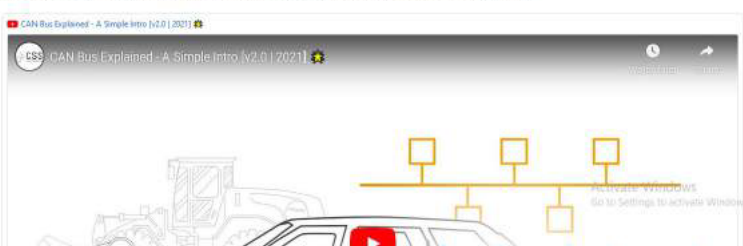
Phase 3: Total CAN

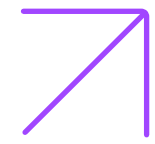
The CAN bus is a multi-master data transmission protocol based on messages and specifically designed for automotive applications, but now it is also used in other areas such as aerospace, industrial automation, and medical equipment. It allows different components to communicate on a network data bus using one or two cables, with speeds of up to 1 Mbps. It uses the concept of broadcasting, which means that all nodes can "listen" to all transmissions. However, the CAN controller has local filters that allow each node to react only to messages of its interest.

- A CAN 2.0A "standard CAN" data frame consists mainly of the following parts:
- **Arbitration Field:** This field consists of 11 bits for the message identifier and the RTR bit. The RTR bit is dominant (0) for data frames and recessive (1) for remote request frames.
 - **Control Field:** This field includes the Identifier Extension bit (IDE), which must be dominant (0) for 11-bit identifiers; a reserved bit for potential future protocol updates; and four bits for Data Length Code (DLC) that indicate the number of data bytes in the message, ranging from 0 to 8 bytes.
 - **Data Field:** This field contains 0 to 8 bytes of data.
 - **CRC Field:** This field consists of 15 bits for error checking and detection, serving as a code to verify the correct transmission of data. It also includes the CRC delimiter bit.
 - **ACK Field:** The transmitter sends a recessive (1), and any receiver sends dominant (0). If this bit is dominant, it indicates that at least one other device received the message. Otherwise, the transmitter re-transmits the message.

Bus CAN Training Videos

Instead of going into high level of details I'm going to place this little nice series of video from **CSS Electronics**, as a introductory

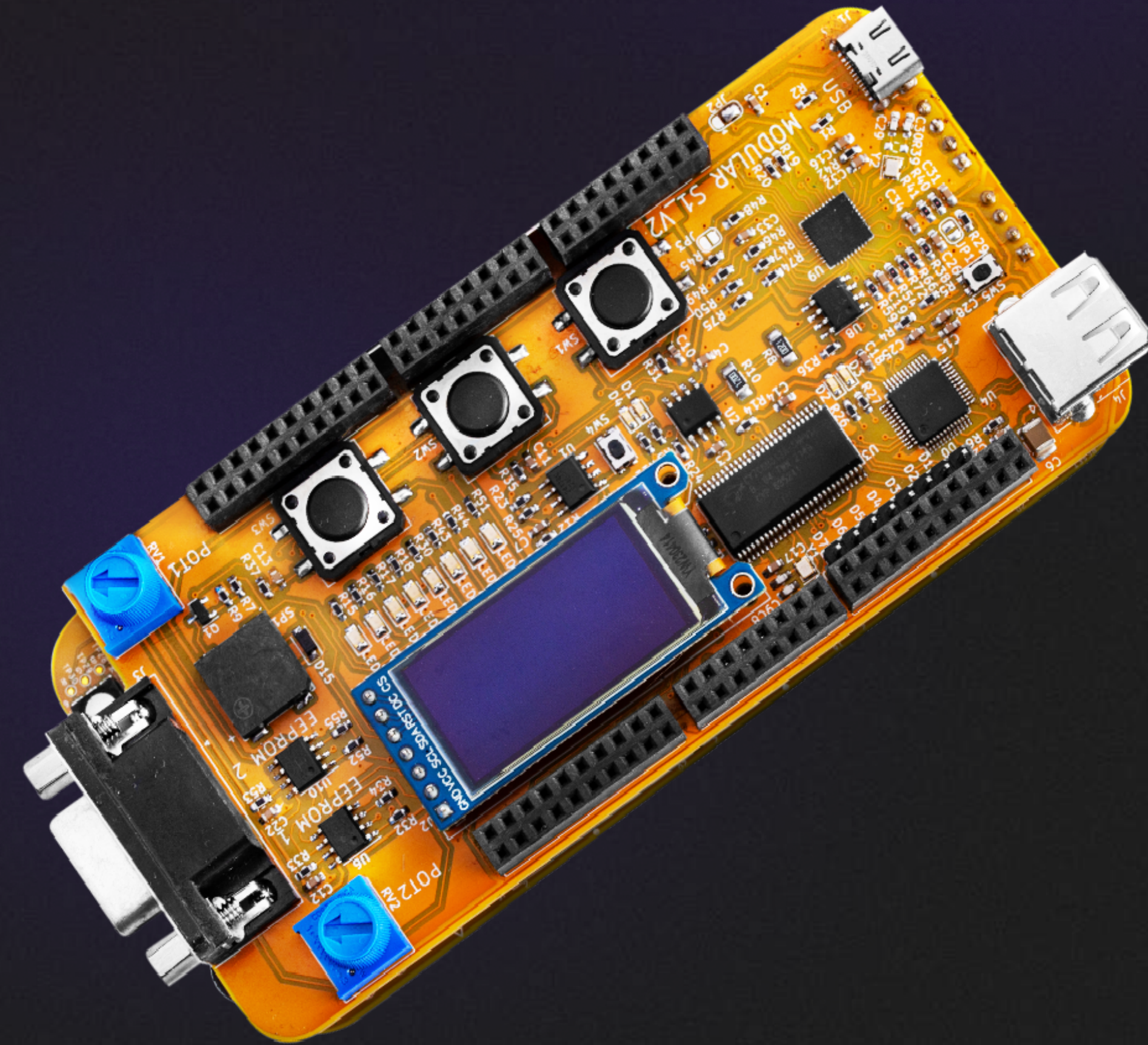




08 Hardware

Certified Embedded Associate AUTOSAR Classic.

As our de facto platform, we use the NXP S32K144-Q100 to which we add our Modular-S1, equipped with everything necessary to maximize your training.





09 Requirements



**Window
computer**



**Internet
connection**



**Conversational
english**



C programming



This version of our AUTOSAR Bootcamp has been targeted especially for professionals who are already part of the automotive industry and can only dedicate part-time, the classes and the self-learning material have been designed in such a way they can carry out all the activities in their spare time.



modular^{mx}