1. **CAN frame (SOF)**

**SOF:**The Start of Frame is a 'dominant 0' to tell the other nodes that a CAN node intends to talk

**ID:**The ID is the frame identifier - lower values have higher priority

**RTR:** The Remote Transmission Request indicates whether a node sends data or requests dedicated data from another node

**Control:**The Control contains the Identifier Extension Bit (IDE) which is a 'dominant 0' for 11-bit. It also contains the 4 bit Data Length Code (DLC) that specifies the length of the data bytes to be transmitted (0 to 8 bytes)

**Data:** The Data contains the data bytes aka payload, which includes CAN signals that can be extracted and decoded for information

**CRC:**The Cyclic Redundancy Check is used to ensure data integrity

**ACK:**The ACK slot indicates if the node has acknowledged and received the data correctly

**EOF:** The EOF marks the end of the CAN frame

CAN devices send data across the CAN network in packets called frames. A CAN frame consists of the following sections.

CAN Frame -- an entire CAN transmission: arbitration ID, data bytes, acknowledge bit, and so on. Frames also are referred to as messages.

https://ni.scene7.com/is/image/ni/bd378304157-1?scl=1

Figure 2.  The standard CAN frame format.

**SOF (start-of-frame) bit** – indicates the beginning of a message with a dominant (logic 0) bit.

**Arbitration ID** – identifies the message and indicates the message's priority. Frames come in two formats -- standard, which uses an 11-bit arbitration ID, and extended, which uses a 29-bit arbitration ID.

**IDE (identifier extension) bit** – allows differentiation between standard and extended frames.

**RTR (remote transmission request) bit** – serves to differentiate a remote frame from a data frame. A dominant (logic 0) RTR bit indicates a data frame. A recessive (logic 1) RTR bit indicates a remote frame.

**DLC (data length code)** – indicates the number of bytes the data field contains.

**Data Field** – contains 0 to 8 bytes of data.

**CRC (cyclic redundancy check)** – contains 15-bit cyclic redundancy check code and a recessive delimiter bit. The CRC field is used for error detection.

**ACK (ACKnowledgement) slot** – any CAN controller that correctly receives the message sends an ACK bit at the end of the message. The transmitting node checks for the presence of the ACK bit on the bus and reattempts transmission if no acknowledge is detected. National Instruments Series 2 CAN interfaces have the capability of listen-only mode. Herein, the transmission of an ACK bit by the monitoring hardware is suppressed to prevent it from affecting the behavior of the bus.

**CAN Signal** – an individual piece of data contained within the CAN frame data field. You also can refer to CAN signals as channels. Because the data field can contain up to 8 bytes of data, a single CAN frame can contain 0 to 64 individual signals (for 64 channels, they would all be binary).

In the following image, there are six channels contained in the data field of a single CAN frame. Each signal contains 8 bits of data.

1. **What is importance of Acknowledge**
2. **Do you have any ideal about UDS**

Unified Diagnostics Service

+ Requesting for Data

+ Writing some data

+ Running test, getting result back

+ Flashing a program

+ Clearing memory

+ Setting a schedule

UDS: Collection of diagnostics services

What are available services

Format of request/Response message

What must be timing parameters

Service handling by Tester and ECU

Service Identifier 0x00 -> 0x3E

1. **What is importance of Diagnostics in vehicle**

Detect a failure in cars would be tedious task as it is a complex machine

When any problem is happening with a human and he is going to the hospital to do the diagnostics and gettinga solution, like this for a vehicle alos it needs to be diagnostic as to inform the human about his problem such as:

+ We may wish to see data stored within the system – such as trouble code – or zsome form of the identifications

+ We may wish to see live data – such as the engine or vehicle speed

+ We may want to transfer a large amount of data – for example re-flashing a module (ECU)

+ We may wish to take direct control of module I/O – for an example disabling individual cylinder to identify the fault

+ We may wish ti run specific routines already in a module – such as some form of self-Calibration

+ We may wish to apply security locks to certain services or to allow the normal function of a system to be disturbed to vary degrees

1. **Can you list routines & services which you know**

****

1. **What is default, programming & extended session**

**default session** = ECU is in normal operation state, but it usually response only to limited set of diagnostic requests. From this state you can usually switch into:

**extended session** to perform some diagnostic services, to talk with ECU (Read/Write values, start/stop routines etc.) Here you expect that ECU give a more or less timely response on your requests. therefore you can also switch between sessions

**programming session** to flash ECU memory, to update ECU SW. While updating ECU usually do not answer on diagnostic requests. Therefore it's not usual to switch from programming session into extended session or even into default session as this switch is a diagnostic request from tester. At the end of an update ECU can perform some memory checks and it will do a reset. When update was successful it will restart in default session.

There can be also other diagnostic sessions for example: dealer session, OEM session...

1. **DTC – How you will test BusOff DTCs**

Transmitter error counter >255

Bit error occurs when ECU monitor a different bit than it transmitted, so If ECU sends a dominant bit and finds recessive on the bus or sends a recessive bit and finds dominant on the bus 🡪 Bit error occurred and transmitted counter increased

* Shorting CAN high and CAN low

1. **How will you confirm whether your ECU entered into sleep**
2. **Do you have experience in testing**
3. **How will you write TCs**
4. **System TCs**

System test cases cover all the scenarios & use cases and also it covers functional, non- functional, user interface, security-related test cases. The test cases are written in the same way as they are written for functional testing.

1. Boundary value verification

Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.

* So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called "boundary testing".
* The basic idea in boundary value testing is to select input variable values at their:

**Equivalence Partitioning**

**Equivalence Partitioning** or Equivalence Class Partitioning is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc. In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of small number of test cases.

* It divides the input data of software into different equivalence data classes.
* You can apply this technique, where there is a range in the input field.

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum
6. **Different Unit test & System test**
7. **How you write TCs use MC-DC test technical**
8. **Different in CANoe& CANalyser**
9. **ASIL\_A, B,C,D**

# **What is ASIL?**

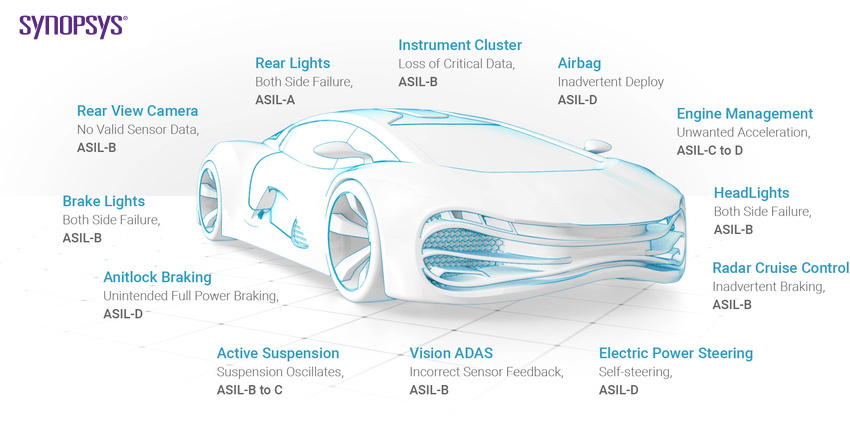
## Definition

ASIL refers to Automotive Safety Integrity Level. It is a risk classification system defined by the ISO 26262 standard for the functional safety of road vehicles.

The standard defines functional safety as “the absence of unreasonable risk due to hazards caused by malfunctioning behavior of electrical or electronic systems.” ASILs establish safety requirements―based on the probability and acceptability of harm―for automotive components to be compliant with [ISO 26262](https://www.iso.org/standards.html).

There are four ASILs identified by ISO 26262―A, B, C, and D. ASIL A represents the lowest degree and ASIL D represents the highest degree of automotive hazard.

Systems like airbags, anti-lock brakes, and power steering require an ASIL-D grade―the highest rigor applied to safety assurance―because the risks associated with their failure are the highest. On the other end of the safety spectrum, components like rear lights require only an ASIL-A grade. Head lights and brake lights generally would be ASIL-B while cruise control would generally be ASIL-C.



## How do ASILs work?

ASILs are established by performing hazard analysis and risk assessment. For each electronic component in a vehicle, engineers measure three specific variables:

* Severity (the type of injuries to the driver and passengers)
* Exposure (how often the vehicle is exposed to the hazard)
* Controllability (how much the driver can do to prevent the injury)

Each of these variables is broken down into sub-classes. Severity has four classes ranging from “no injuries” (S0) to “life-threatening/fatal injuries” (S3). Exposure has five classes covering the “incredibly unlikely” (E0) to the “highly probable” (E4). Controllability has four classes ranging from “controllable in general” (C0) to “uncontrollable” (C3).

All variables and sub-classifications are analyzed and combined to determine the required ASIL. For example, a combination of the highest hazards (S3 + E4 + C3) would result in an ASIL D classification.

## What are the challenges of ASILs?

Determining an ASIL involves many variables and requires engineers to make assumptions. For example, even if a component is hypothetically “uncontrollable” (C3) and likely to cause “life-threatening/fatal injuries” (S3) if it malfunctions, it could still be classified as ASIL A (low risk) simply because there’s a low probability of exposure (E1) to the hazard.

ASIL definitions are informative rather than prescriptive, so they leave room for interpretation. A lot of room. ASIL vocabulary relies on adverbs (usually, likely, probably, unlikely). Does “usually” avoiding injury mean 60% of the time or 90% of the time? Is the probability of exposure to black ice the same in Tahiti as it is in Canada? And what about traffic density? Rush hour in Los Angeles vs. late morning on an empty stretch of road in the Australian Outback?

Simply put, ASIL classification depends on context and interpretation.

## How are ASILs evolving?

Given the guesswork involved in determining ASILS, the [Society of Automotive Engineers (SAE)](https://www.sae.org/) drafted J2980, “Considerations for ISO 26262 ASIL Hazard Classification” in 2015. These guidelines provide more explicit guidance for assessing Exposure, Severity, and Controllability for a given hazard. J2980 continues to evolve―the SAE published a revision in 2018.

With the evolution of the self-driving car, ISO 26262 will need to revisit the definition of “Controllability,” which currently pertains to the human driver. As the standard reads now, the absence of a human driver means that Controllability will always be C3, the extreme of “uncontrollable.” The other variables of Severity (injury) and Exposure (probability) will no doubt require re-examination as well.

## What are the benefits of ASILs?

ISO 26262 is a goal-based standard that’s all about “preventing harm.” Despite their challenges, ASIL classifications are intended to “prevent harm” and help us achieve the highest safety rating possible for myriad automotive components across a long and often disjointed supply chain.

Key benefits include:

* Establishing safety requirements to mitigate risks to acceptable levels
* Managing and tracking safety requirements
* Ensuring that standardized safety procedures have been followed in the final product

## How does Synopsys help you meet ASIL requirements?

The Synopsys [DesignWare IP](https://www.synopsys.com/designware-ip/ip-market-segments/automotive.html) portfolio with safety packages is ASIL B and D ready, ISO 26262 certified, and designed for use in safety-critical applications. Our ASIL-certified IP also accelerates SoC development for applications like [advanced driver assistance systems (ADAS)](https://www.synopsys.com/automotive/what-is-adas.html).

Our [safety packages](https://www.synopsys.com/designware-ip/ip-market-segments/automotive/functional-safety.html) consist of failure modes effects and diagnostics analysis (FMEDA) reports, safety manuals, and certification reports to accelerate safety assessments and help you reach your target ASILs. Using DesignWare IP also reduces supply chain risk and accelerates the entire process of achieving SoC-level functional safety (from requirements specification to design, implementation, integration, verification, validation, and configuration).

1. CAPL
2. Autosar Framework
3. System that you are testing
4. System recovery
5. Component testing

Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. It's also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.

1. Blackbox and whitebox **What is Black Box testing?**

In Black-box testing, a tester doesn't have any information about the internal working of the software system. Black box testing is a high level of testing that focuses on the behavior of the software. It involves testing from an external or end-user perspective. Black box testing can be applied to virtually every level of software testing: unit, integration, system, and acceptance.

## What is White Box testing?

White-box testing is a testing technique which checks the internal functioning of the system. In this method, testing is based on coverage of code statements, branches, paths or conditions. White-Box testing is considered as low-level testing. It is also called glass box, transparent box, clear box or code base testing. The white-box Testing method assumes that the path of the logic in a unit or program is known.

## KEY DIFFERENCE

* In Black Box, testing is done without the knowledge of the internal structure of program or application whereas in White Box, testing is done with knowledge of the internal structure of program.
* Black Box test doesn’t require programming knowledge whereas the White Box test requires programming knowledge.
* Black Box testing has the main goal to test the behavior of the software whereas White Box testing has the main goal to test the internal operation of the system.
* Black Box testing is focused on external or end-user perspective whereas White Box testing is focused on code structure, conditions, paths and branches.
* Black Box test provides low granularity reports whereas the White Box test provides high granularity reports.
* Black Box testing is a not time-consuming process whereas White Box testing is a time-consuming process.