Aufgabe 1

Is an ECU an embedded system? Explain your answer.

Answer (2 point): An electronic control unit (ECU) is an embedded system that controls one or more of the electrical subsystems in a vehicle. Sensors, actuators and ECUs are often one unit in mechatronic systems.

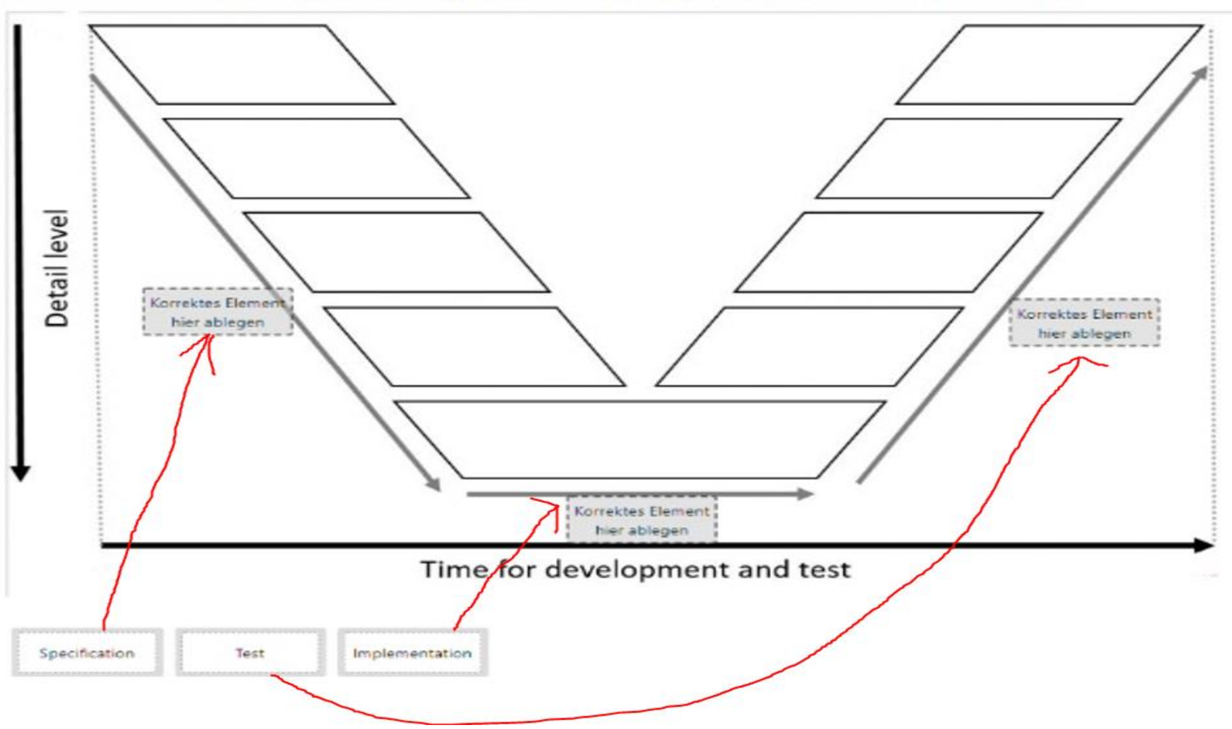
Is it necessary to test ECUs in the automotive domain? Explain your answer briefly.

Answer (2 point): ECUs should be tested mandatorily. ECU integrates functionalities for calculation, analysis of sensor data and controlling actuators.

White box testing should be performed in calculation and on sensed data provided by the dedicated sensor for the ECU. Black box test should be performed in the actuator controlling against the requirement.

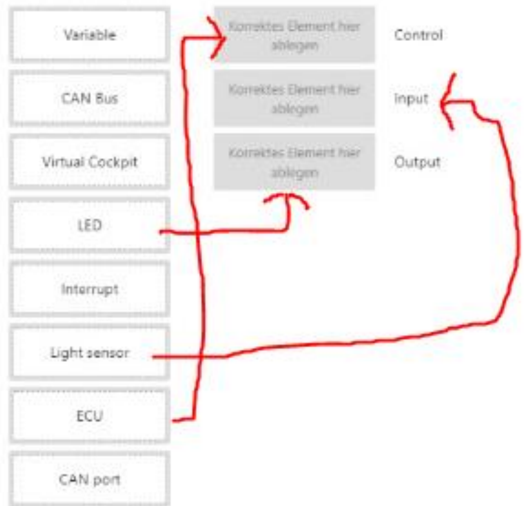
Aufgabe 3

Put the given words to the correct position in the picture, representing the V-Model. Use Drag & Drop to put the words to the correct position.



Aufgabe 4

From the knowledge gained during Unit 1, Please match the below elements to their correct assignment. Use Drag & Drop to put the words to the correct position.



Answer(1.5marks):

Control = ECU   Input = Light Sensor   Output = LED

Aufgabe 5

Which of the following is true about Asynchronous communication ?
(Please select at most 4 answers. otherwise selecting more answers will result in 0 for this question)

- [ ] If communication clock is higher than bit rate, then clock division is required
- [ ] If communication clock is higher than bit rate, then clock division is not required due to master clock
- [ ] Re-synchronisation phase is not needed
- [x] Absence of a centralised master clock
- [x] Re-synchronisation phase is necessary
- [ ] Communication speed is managed through a master clock
- [x] Communication bit rate is same for all nodes

Tanvir

Aufgabe 6

Through an automotive perpective, mark the following attributes as advantage/disadvantage of a bus system compared to point-to-point communication approach

	Advantage	Disadvantage
Cabling	☐	X
Reliability	X	☐
Message packet length	☐	X
Weight	☐	X
Real time properties	X	☐
Cost effectiveness	X	☐

Aufgabe 7

Punkte: 3 Keine Antwort

With your understanding of CAN arbitration process, determine the correct sending order from the following message-IDs, when multiple nodes in a CAN-Network are trying to send simultaneously. Move the from the left list to correct position in the right list.

Message ID List:

0x 0AC
0x 0FF
0x 01E
0x 100
0x 009
0x 00C

2 0x00C	1
5 0x0FF	2
3 0x01E	3
4 0x0AC	4
6 0x100	5
1 0x009	6

Aufgabe 8 Punkte: 2 Keine Antwort

The following is a snapshop of a message register that respresents an 8-digit message ID of a CAN-Message

ID bit 7	ID bit 6	ID bit 5	ID bit 4	ID bit 3	ID bit 2	ID bit 1	ID bit 0
MSB							LSB

Calculate the masking (MASK) and acceptance register (ACCEPT) values to approve only CAN messages with ID 229_{10} ($0E5_{16}$) and 197_{10} ($0C5_{16}$). (Numbers are given in decimal and numbers in brackets are given in Hexadecimal format) (Please answer the question in Hexadecimal/Binary/Decimal formats only)

Mask: ________

Accept: ________

Answer(2marks):

229=11100101 (OE5) **197=11000101 (0C5)**

In binary answer is

Mask: 11011111 (**ODF**)

Accept: 11x00101 => 11**1**00101 (0E5) **or** 11**0**00101 (**0C5**)

Aufgabe 9 Punkte: 4 Keine Antwort

Following is a function for sending a 16 bit long unsigned counter value encapsulated in a can message with id 0x10. The function is called at a time interval of every 500 ms and should send an incremented value of "counter" variable by 30 at every time interval. Upon reaching a value that equals 1000 or more, the counter should be reset to zero. The variable must be split in two parts of 8 bits each, in order to accommodate it in a can message. The first data byte of the can message should contain the first 8 least significant bits of the aforementioned variable "counter". And the second data byte should contain 8 most significant bits. Analyse the following piece of code to mark and select the correction option for the problems/errors against the above specified requirements in the below function:

```
void counterUpdateSend()
{
static uint16 counter = 0; // variable declaration for counter
counter = counter + 30; // counter increment
if (counter = 1000)
{ counter = 0; }
CAN_0.BUF[8].MSG_ID.B.STD_ID = 10; // CAN message id
CAN_0.BUF[8].CS.B.LENGTH = 1; // CAN message data length
CAN_0.BUF[8].DATA.B[0] = counter % 128; // obtaining first 8 least significant bits from the variable
CAN_0.BUF[8].DATA.B[0] = counter / 256; // obtaining the 8 most significant bits
CAN_0.BUF[8].CS.B.CODE = 0xC; // buffer code for transmitting a message
}
```

Answer (4 marks):

C X X X X X C

Aufgabe 10

Following is an 8-bit MASK and ACCEPT configuration. Please select which of the IDs can be accepted or not using this configuration on an ECU node connected to a CAN-Bus.

Mask : 0xF1
ACCEPT: 0x01

	Will be Rejected	Will be Accepted
0x10	✗	
0x01		✓
0x03		✓
0x13	✗	
0x0F		✓
0xF3	✗	
0xFA	✗	
0x09		✓

- Aufgabe 11
- 3. AUTOSAR
  - Aufgabe 12
  - Aufgabe 13
  - Aufgabe 14
  - Aufgabe 15
  - Aufgabe 16
  - Aufgabe 17
- 4. Test of ECUs
  - Aufgabe 18
  - Aufgabe 19
  - Aufgabe 20
  - Aufgabe 21
  - Aufgabe 22
  - Aufgabe 23
- 5. Programming
  - Aufgabe 24

89 Minuten 12 Sekunden

```
else
PIT_StopTimer(1);

}
}


void PIT_CHANNEL1(void) /* timer 1 interrupt function */
{
SIU.GPDO[59].R = 0;
}
```

- [x] Timer is set to 100 ms
- [x] Timer is never started
- [x] LED 6 will turn ON only once, depending on switch input
- [x] LED can remain permanently OFF
- [ ] LED can remain permanently ON
- [x] Interrupt function will be only triggered when timer is started
- [ ] LED 6 will blink every 100 ms
- [ ] Interrupt function will be always triggered until program stops
- [ ] Timer is set to 1 sec
- [ ] Timer can remain permanently started
- [ ] LED 6 is turned on before timer starts
- [ ] Timer is wrongly configured
- [ ] Switch has no influence on LED

Aufgabe 12

What are the main ideas of the standard "AUTOSAR" (Automotive Open Systemarchitecture)?

- [x] Function Orientated Approach
- [x] Hardwaren independent Application which enables Reuseability
- [ ] Less wires in car
- [ ] Increasing Complexity of Software

Aufgabe 13

Explain the concept of "Complex Device Drivers" in AUTOSAR. What is it? Where is it located in the AUTOSAR architecture diagram? Why is it needed?

Answer(2marks): Complex Device Driver (CDD) enables the integration of special-purpose functionality that is not specified in AUTOSAR. The interface to RTE must be AUTOSAR compliant. Direct hardware access.

In AUTOSAR architecture diagram, it is located in the Basic Software Components. It works as a stack between the ECU and the RTE.

CDD provides hardware compatibility that is not supported, simplifies the migration process from non-AUTOSAR to AUTOSAR ECUs, time critical functions, and functional safety.

Aufgabe 14

Which elements are used to create a software architecture (application) in AUTOSAR?

- ○ Micro Services, Kubernetes, Cloud
- ○ Webservice, FIBEX, SVN
- ○ Software components, Database, Registers
- ○ AI, Machine Learning, SVM
- ● Software components, Ports, Interfaces, Runnables
- ○ Container, Constructor, Loops
- ○ Function, Classes, , Registers, Interrupts

Aufgabe 15

What does VFB mean in the context of AUTOSAR? State its benefit and the software development stage in which it is used.

Answer (2): **Virtual Functional Bus – VFB:**

- Logical connection of software components from application with each other
- Before mapping of ports to concrete signals/bus system
- Before mapping of SWCs to ECUs (application as non distributed)

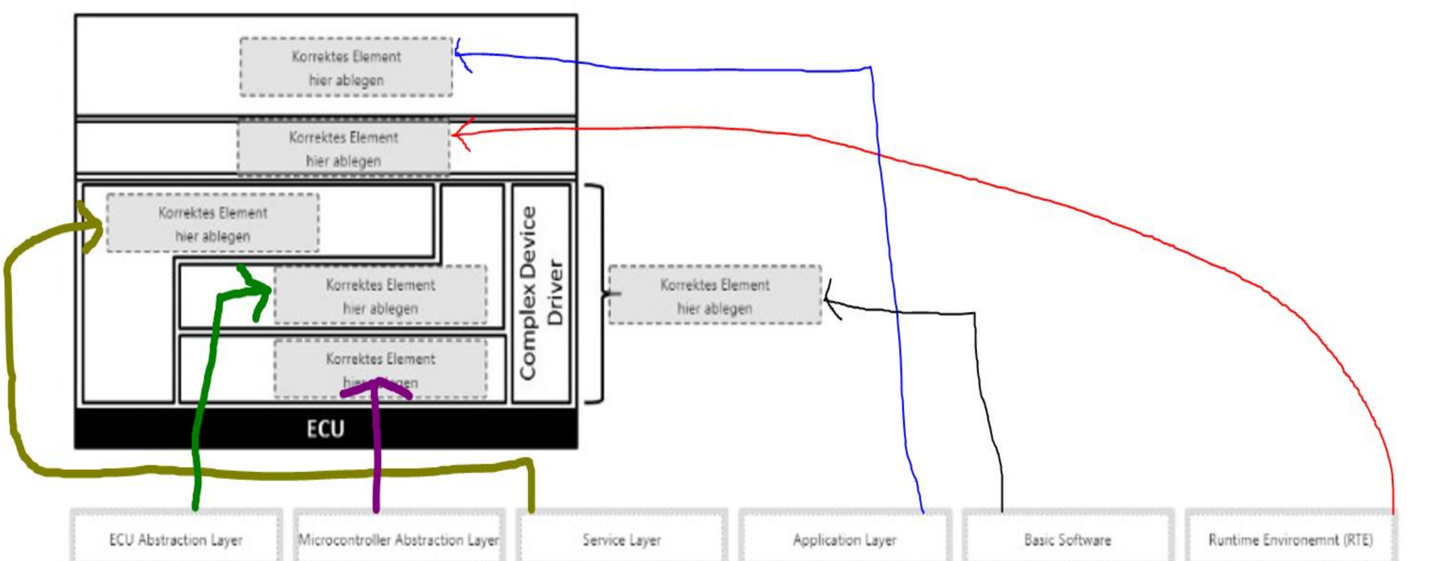
Benefits:

- Enables testing (without BSW) of software components – test not on target platform

Used in Implementation/Generation of application stage

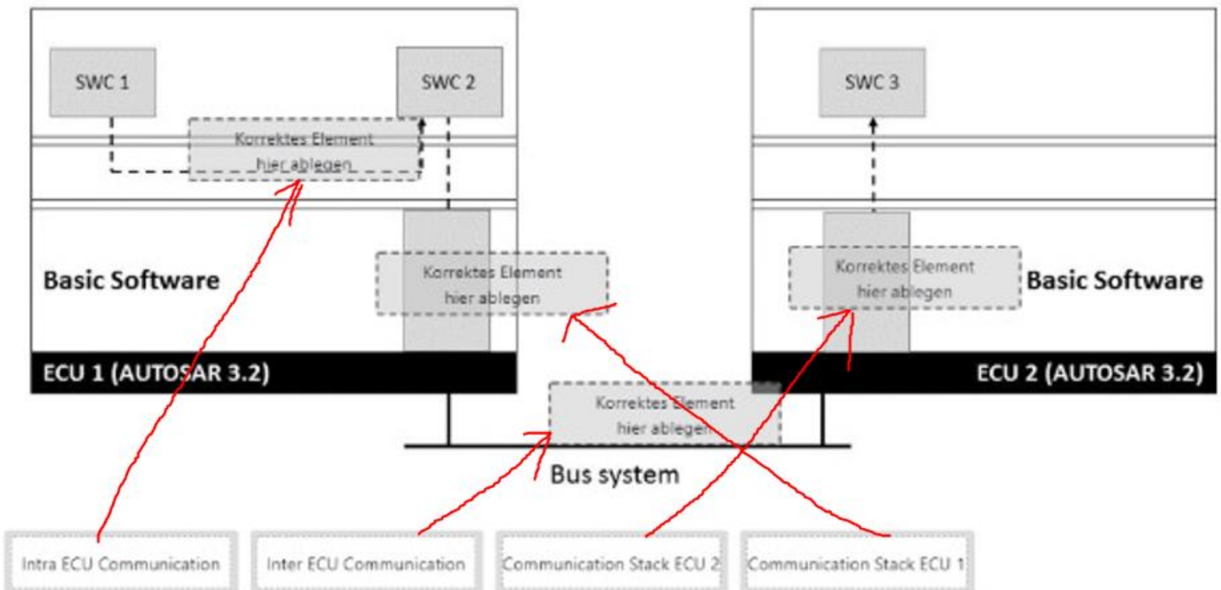
Aufgabe 16 Punkte: 3

Put the words to the correct position in the architecture picture of AUTOSAR.



Aufgabe 17

Put the given words to the correct position in the picture of two ECUs with AUTOSAR architecture.



Aufgabe 18

What is the main difference between static test and dynamic test?

Explain shortly, in which development phases static test and dynamic test are mostly used.

Aktuell geschriebene Wörter: 0

Answer(2marks):

Static test: Test object is not executed, but analyzed.

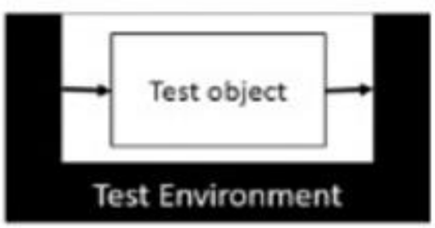
Dynamic test: Test object is executed(partly).

Static test mostly used during/at the end of implementation to save software quality and code compliance. Where dynamic test mostly used after implementation.

Aufgabe 19

The SuT (System under Test) represents the test object and test environment.

What does **PoC** and **PoO** in SuT stand for? What do they realize?



Answer(2marks):

In System under Test (SuT), Point of Control (PoC) realizes simulation of an object with test data and Point of Observation (PoO) realizes the output reading test object.

Aufgabe 20

The V-Model contains different phases of test. In the given picture, the four abstract test phases are named. Explain briefly the major differences of the phases.

Where do the test cases come from?

Aktuell geschriebene Wörter: 0

Answer(3marks):

Component Test:

- Directly after development, small units, classes or modules are tested, often white box test.

Integration Test

- Set of components is connected and tested

System Test:

- Test from customer point of view where integrated components are tested.

Acceptance Test:

- Considered as final test where test regards the user contract and user acceptance.

Aufgabe 21

Which sequence of the simulation types in the V-Model is correct (from top left to top right)?

- ○ MiL, HiL, SiL
- ✓ MiL, SiL, HiL
- ○ HiL, SiL, MiL

Aufgabe 22 Punkte: 5

Testing Park Pilot of a Car

The functionality of the Park Pilot System (PPS) of a car was developed and needs to be tested. The Park Pilot System supports the driver in parking. A button with two posible states (on, off) activates or deactivates the park pilot assist.

Features:

- The park pilot can be switched on by the driver. It can be switched on when the speed from the car is less than 10 km/h
- The park pilot will be deactivated when sensors are not working (sensor status)

Possibles values of the sensors:

button for park pilot : **0 - OFF, 1 - ON**
current speed: **< 10 km/h, >= 10 km/h**
sensor status: **0 - ERROR, 1 - WORKS**

Possibles States of Park Pilot System:

0 - OFF, 1 - ON

The test engineer has defined the following truth table based on the specification.

Button Park Pilot	Sensor Status	Current speed	Park Pilot Status
0	0	<10 km/h	0
0	1		0
1	0		0
1	1		1
0	0	>=10 km/h	0
0	1		0
1	0		0
1	1		0

Question a:
Which test strategy is represented by the given truth table (for example random test strategy)?

Answer (5 marks):

a)

It is sequential test strategy.

Aufgabe 23 Punkte: 6

Testing Seat Heating (Driver Seat) of a Car

The functionality of the Seat Heating of a car was developed and needs to be tested. The Seat Heating is used to heat the surface of the driver seat.
Our Seat Heating has a button, which switches the heating on or off. Our Seat Heating has a very simple functionality, it switches off, when a temperature of 60 degrees is reached (time in ON state is not considered).

Features:

- Seat Heating can be switched on or off by a button
- If Seat Heating is switched ON, the heater increases the temperature up to 60°C. After that the Seat Heating is switched OFF

button for Seat Heating: **0 - OFF, 1 - ON**
temperature sensor: **<= 60°C, > 60°C**

Question a:
Create a (complete) truth table for the Seat Heating .

Question b:
Create a manual test scenario for the Seat Heating. Consider in the manual test sceanrio input values in which the Seat Heating is in ON state.

Hints:

- You can realize the truth table like that:

a | b | c
0 | 0 | 1
1 | 1 | 1

- Use the test constructs from unit 5.

Construct	Parameters	Example
Loop	<Condition>	LOOP (true)
If Then	<Condition>	If (x == 13) then y = 1
Wait	<Time> (in ms)	Wait (1000) // ms
Calculator		x = a + b
Bus Read	<Message ID> (according to DCB file)	a = Can.read(engine_speed)
Bus Write	<Message ID>, <value>	Can.write(engine_speed, 100)
User Interaction	<Text>	Dialog ("Put your hand on the sensor")
Variable		int x = 1
Test Cancelation		END

Answer(6marks) a:

varBtn|varTempSensor|varHeater

0|-|0

1|<= 60|1

1|> 60|0

Answer(6marks) b:

```
Start

Dialog ("Hi! Button is off and Heater is off")


varBtn=can.read(varBtn)

varTempSensor=can.read(varTempSensor)

varHeater =can.read(varHeater)


if(varBtn == 0 && varTempSensor == 0 && varHeater == 0)

{

        Success

}

else

{

        Failed

}


Dialog ("Hi! Temperature is less than 60, Button is on, and Heater is on")


varBtn =can.read(varBtn)

varTempSensor =can.read(varTempSensor)

varHeater =can.read(varHeater)
```

```
if(varBtn == 1 && varTempSensor<60 && varHeater == 1)
{
        Success
}
else
{
        Failed
}

Dialog ("Hi! Temperature is greater than 60, Button is on, and Heater is off")

varBtn =can.read(varBtn)
varTempSensor =can.read(varTempSensor)
varHeater =can.read(varHeater)

if(varBtn == 1 && varTempSensor>=60 && varHeater == 0)
{
        Success
}
else
{
        Failed
}
```

Aufgabe 24

Punkte: 10 Keine Antwort

Please write C-Code for the boards used in the practical to perform the following specified tasks:

- Configure the required hardware peripherals (pins, LEDs and switches)
- Send a CAN message with ID **36**$_{10}$ every 1000ms
- This message to be sent uses 3 data fields to send the following information:
  Data-byte[0]: Button 5 , Data-byte[1]: Button 6, Data-byte[2]: Light sensor value
  For the light sensor value, 8 least significant bits should be sent. The light sensor is connected to Feature "ANA IN1" shown in the table on next page.
- The board should also receive a can message with ID **37**$_{16}$
- The message to be received has 4 data fields. These data fields represent 4 LEDs. Please turn the LEDs "on" or "off" according to the data you get from the CAN message. The possible values for this representation are 0 and 1.
  Data-byte[X] contains status for LED X, where X ranges from 0 to 3.
- LED 5 blinks every second and LED 6 blinks at an interval of 200 ms.

Please use the provided information. It helps you to write the code.

Input/output

A button can be configured as digital input by writing "0x0100" to the corresponding PCR register. A pressed button or a switch on "action" results in a low signal ("0") on the pin. This signal can be checked by the input register of the corresponding pin ("SIU.GPDI[X].R", where X=corresponding PCR register number). Pins can be configured as output by writing a "0x200" to the corresponding PCR register. To set a digital output write "0" or "1" to the corresponding output register "SIU.GPDO[X].R". Analog inputs can be configured by writing a "0x2500" to the corresponding register. The converted values can be retrieved by reading the "ADC_0.CDR[2].B.CDATA" register.
You can find the PCR register numbers in the following description in Table 1.

Timer

No software interrupts need to be configured. There is only one timer available. Please use the function "*PIT_ConfigureTimer(int timerChannel, int period)*" to configure this timer. To start the timer use the function "*PIT_StartTimer(int timerChannel)*".

Parameter	Value
timerChannel	Timer channel to be configured
period	Period in milliseconds

Upon successful configuration of the timer interrupt, function "*PITCHANNEL0()*" will be automatically called. Implement this function for timer related tasks.

CAN

To reduce the complexity of the task you may configure the CAN driver just like this:

```
SIU.PCR[16].B.PA = 1;     /* TX CAN pin configuration */

SIU.PCR[17].B.PA = 1;     /* RX CAN pin configuration */
```

Furthermore for sending a CAN message you just need to set the register with the following constructs. It is possible to use some of them more than once. For example, the following code sends character "a" using the message ID 10.

CAN

To reduce the complexity of the task you may configure the CAN driver just like this:

```
SIU.PCR[16].B.PA = 1;     /* TX CAN pin configuration */

SIU.PCR[17].B.PA = 1;     /* RX CAN pin configuration */
```

Furthermore for sending a CAN message you just need to set the register with the following constructs. It is possible to use some of them more than once. For example, the following code sends character "a" using the message ID 10.

```
CAN_0.BUF[0].MSG_ID.B.STD_ID = 10; /* message identifier */

CAN_0.BUF[8].CS.B.LENGTH = 1;     /* message length */

CAN_0.BUF[8].DATA.B[0]  = 'a';   /* data byte 0 */

CAN_0.BUF[8].CS.B.CODE  = 0xC;   /* code for sending a message */
```

Implement the function "*CANRCV()*" to handle the received messages. This function is called automatically if a new CAN message is received. In this function you can read the following registers within the if-condition. "X" is replaceable.

```
CAN_0.BUF[0].ID.B.STD_ID     /* received message identifier */

CAN_0.BUF[0].DATA.B[X]      /* received data byte X */
```

Feature	CPU pin name	PCR register	Comments
LED7	PA11	11	Light when command is low
LED6	PD11	59	Light when command is low
LED5	PD13	61	Light when command is low
LED4	PD14	62	Light when command is low
LED3	PA12	12	Light when command is low
LED2	PA13	13	Light when command is low
LED1	PC10	42	Light when command is low
LED0	PA9	9	Light when command is low
BT6	PA0	0	On = low
BT5	PA1	1	On = low
ANA IN1	PC1	33	External analog input
ANA IN2	PC2	34	External analog input
ANA OUT	PC9	41	Analog output
POT	PE1	65	Potentiometer
TEMP	PE2	66	Temperature Sensor

Table 1 : Overview of PCR registers for I/O

Please use the following code skeleton for your answer in the text area below :

```
/*------------------------------Start of Code------------------------------*/
/* global variables */


/* pin configuration */
void configure(void) {
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].R = ;
SIU.PCR[ ].B.PA = ;
SIU.PCR[ ].B.PA = ;
}

/* main function */
void main(void) {


init(); /* board initialization */
configure(); /* pin configuration */

for(;;) {



}
}


/* timer interrupt function */
void PIT_CHANNEL0(void) {
```

Answer(6marks) a:

```
//global variables
int lightValue = ADC0.CDR[2].B.CDATA;

SIU.PCR[0].R = 0x0100; //BT5
SIU.PCR[1].R = 0x0100; //BT6

SIU.PCR[9].R = 0x0200; //LED0
SIU.PCR[42].R = 0x0200; //LED1
SIU.PCR[13].R = 0x0200; //LED2
SIU.PCR[12].R = 0x0200; //LED3
SIU.PCR[62].R = 0x0200; //LED4
SIU.PCR[61].R = 0x0200; //LED5
SIU.PCR[59].R = 0x0200; //LED6

SIU.PCR[33].R = 0x2500; //ANA IN 1

SIU.PCR[16].B.PA = 1 //Tx CAN pin config
SIU.PCR[17].B.PA = 1 //Rx CAN pin config

//Can buffer configuration
CAN_0.BUF[8].MSG_ID.B.STD_ID = 36;
CAN_0.BUF[8].CS.B.LENGTH = 3;
CAN_0.BUF[8].CS.B.CODE = 8;

//Configuration timers
PIT_ConfigureTimer(1,200);
```

```
PIT_ConfigureTimer(2,1000);
PIT_StartTimer(1);
PIT_StartTimer(2);

//inside for loop
if(CAN_0.BUF[0].ID.B.STD_ID == 0x37){
	SIU.GPDO[9].R = CAN_0.BUF[0].DATA.B[0];
	SIU.GPDO[42].R = CAN_0.BUF[0].DATA.B[1];
	SIU.GPDO[13].R = CAN_0.BUF[0].DATA.B[2];
	SIU.GPDO[12].R = CAN_0.BUF[0].DATA.B[3];
}

//after main
void PIT_Channel_1(void){
	SIU.GPDO[59].R = ~SIU.GPDO[59].R;
	PIT.CHANNEL[1].TLFG.R = 1;
}

void PIT_Channel_2(void){
	SIU.GPDO[61].R = ~SIU.GPDO[61].R;
	CAN_0.BUF[8].DATA.B[0] = SIU.GPDI[0].R;
	CAN_0.BUF[8].DATA.B[1] = SIU.GPDI[1].R;
	CAN_0.BUF[8].DATA.B[2] = lightValue & 0xFF;
	CAN_0.BUF[8].CS.B.CODE = 0xC;
	PIT.CHANNEL[1].TLFG.R = 1;
}
```

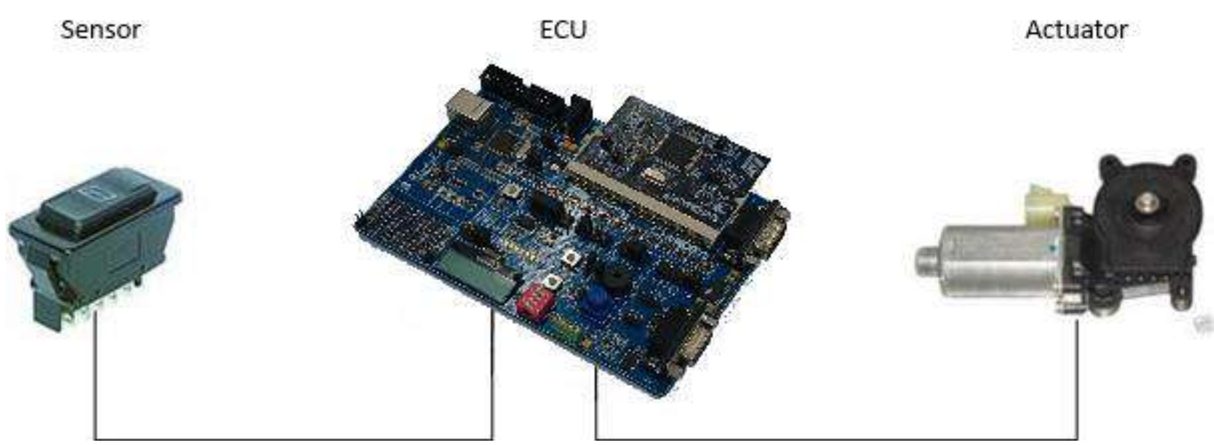
Don't forget to double check your answer before trusting on mine ;)

Good Luck !

1) What is ECU?

ECU integrates functionalities for calculation, analysis of sensor data and controlling actuators

2) Name this diagram .



1 ----------------- 1-------------------- 1---------------------

2------------------ 2------------------- 2--------------------

3) Is ECU an embedded system, explain?

yes, as embedded system are designed for specific task and ECU serves the same

4) Name 3 or 4 ECU in the car.

ABS, ESP, ACC, ES

5) Requirements of ECU?

Sensor, Actuator, Processor, Memory, Digital Hardware, and Interface

---

unit2:

6) Name 3 advantage of Can bus.

Reliability, Safety

low cost, space and weight

multiple use of signal

7) Structure of CAN message according to Can2.0A:

SOF[1] (1 Bit)	Message Identifier (11)	Control Bits (7)	Data Field (0 - 64)	CRC[2] (16)	ACK[3] (2)	EOF[4] (7)

8) The can bus uses the CSMA/CA principle. What does it mean . Explain?

Carrier sense multiple access with collision avoidance (CSMA/CA) , is a network multiple access method in which carrier sensing is used, but nodes attempt to avoid collisions by transmitting only when the channel is sensed to be "idle"

9) Explain about this code:

```
if (SIU....[])=0 : switch

{conf timer0 (1,10000;

}

else { conftimer0(0,1000);

}
```

10) Synchronous communication.

sampling time, synchronisation between transmitter and receiver during communication and time between two transmissions are defined
usually a common clock or
self synchronising code is used for synchronisation (master/slave relation necessary)

Unit 3:

11) What is the acceptance and mask value for 35 and 34?

Accept is with 0

Ignore is with 1

12) We want to send the word "HELLO" trough can-bus.

Data length? 5 bytes

Data ID? can be anything

13) Name 3 buses except can -bus.

LIN(<10kbps), MOST(>10Mbps), FlexRay(<10Mbps)

14) We have 2 can buses witch have different data rate. Can they communicate? Why?

No, because loss of data might happen

15) Arbitration example.

Unit 4:

16) For what Autosar stand for. What was the goal?

Automotive Open System Architecture

Modularity of software elements,

Scalability of functions,

Transferability,

Reusability of functions and

Standardized interfaces, to be able to guarantee exchange

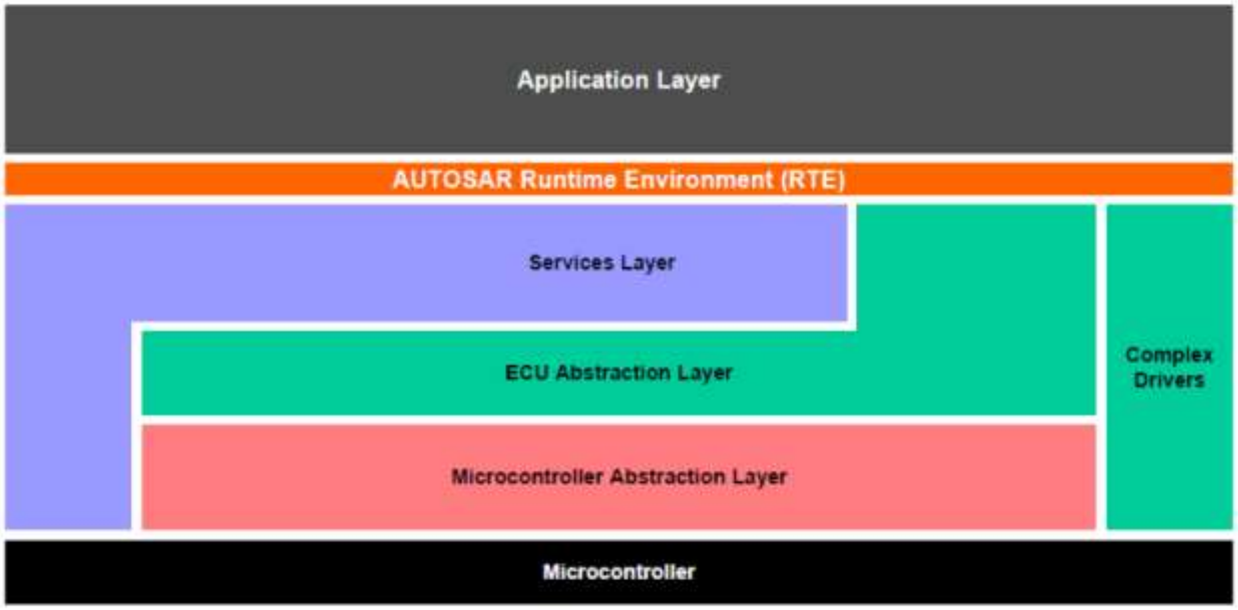
17) Name 3 Goal and 3 topics in Autosar.

A architecture

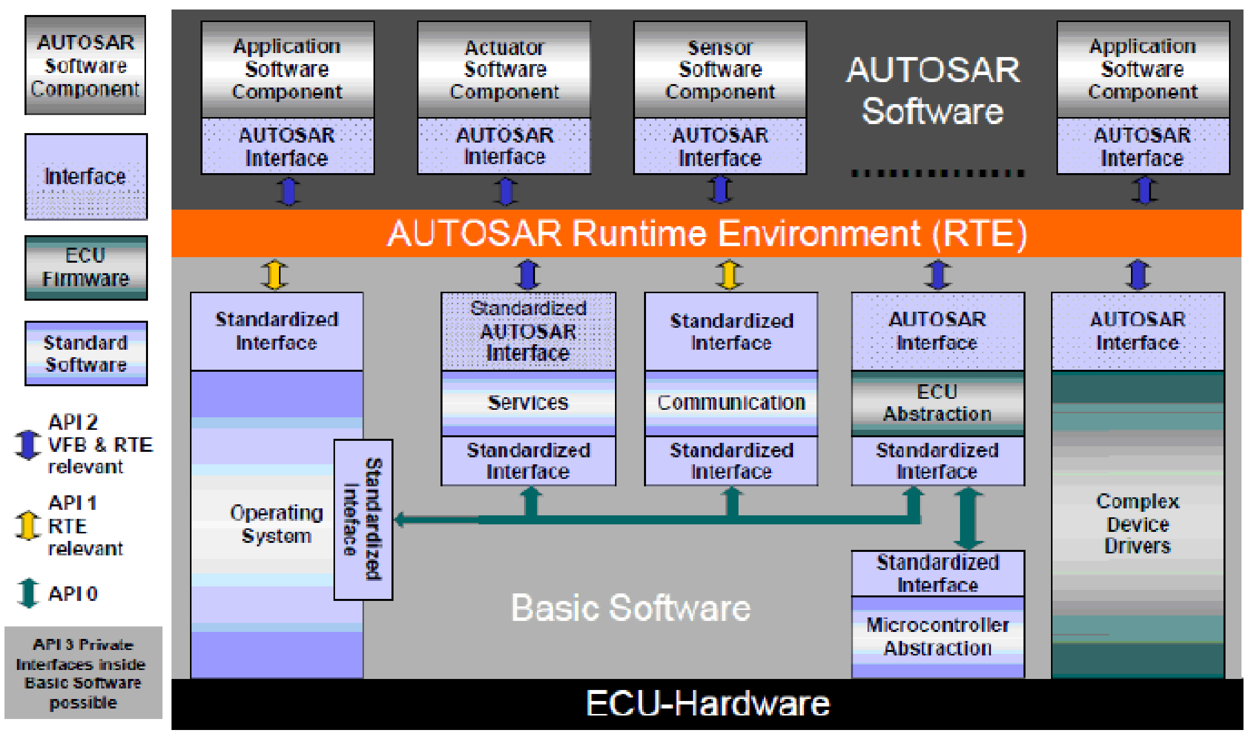
I interface

M methodology

18) Label the schematic display of an autosar control unit.



19) Label the vertical areas of the basic software of an AUTOSAR control unit.



20) Name communication between software components in Autosar.

~~VFB : Virtual Functional Bus~~

Unit 5:

21. Definition of verification and validation.

- **Definition:** Validation

  "The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements."

  IEEE Standard Glossary of Software Engineering Terminology. In: IEEE Std 610.12-1990 (1990)

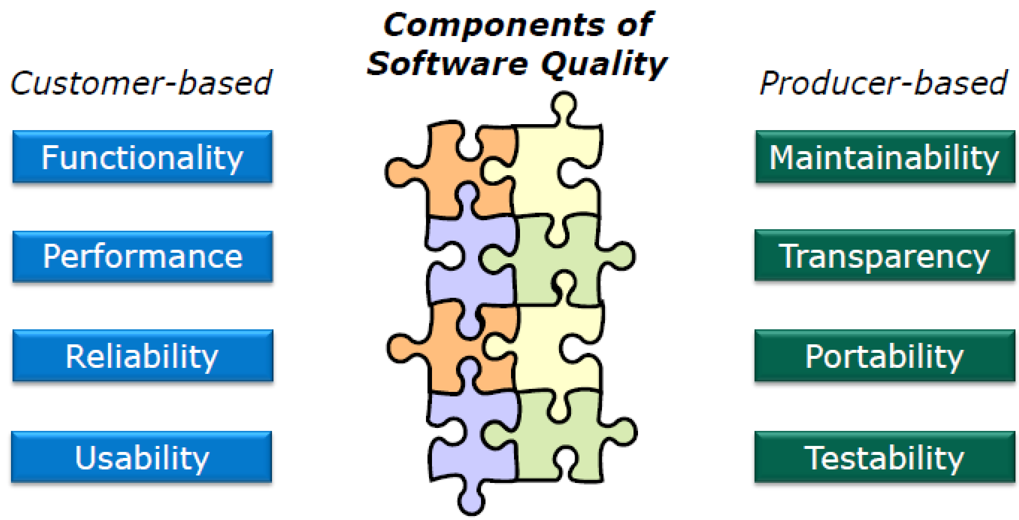
- **Definition:** Verification

  "The process of evaluating a system or component to determine whether the products of a given development phase satisfies the conditions imposed at the start of that phase."

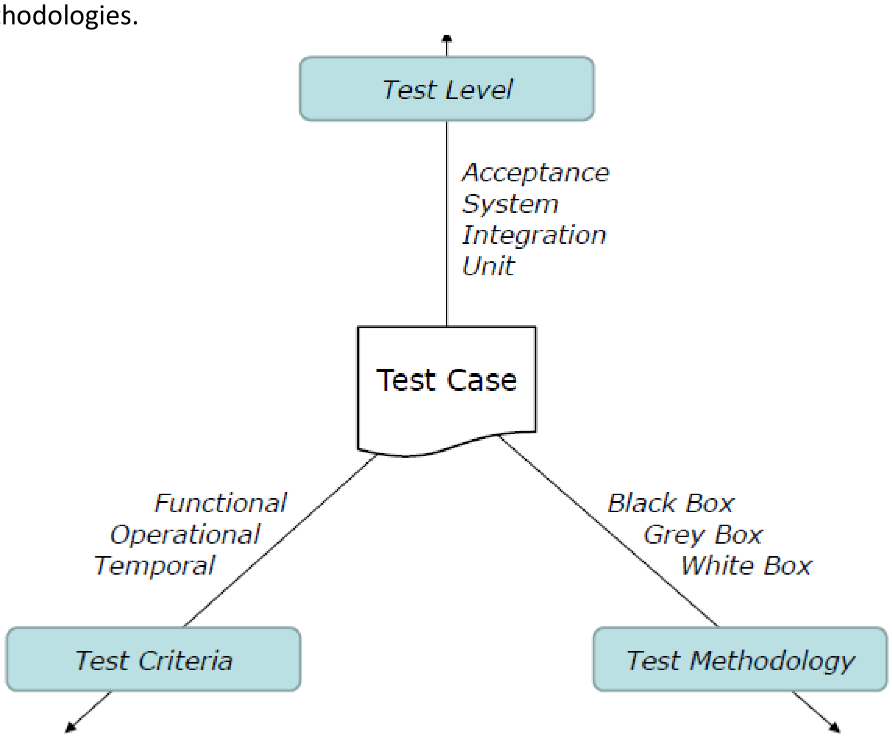
  IEEE Standard Glossary of Software Engineering Terminology. In: IEEE Std 610.12-1990 (1990)

22. Customer based and producer based quality.

- **Definition**
  "[Software quality is] capability of a software product to satisfy stated and implied needs when used under specified conditions."
  ISO/IEC 25000:2005 Software Engineering -- Software product Quality Requirements and Evaluation (SQuaRE)



23. Test methodologies.



24. Mil, Pil, Sil, Hil.

25. Static and dynamic test. (Static test = Test object not executed but analyzed. Types)

(Dynamic test = Test object executed partly. Types)

26. Debugging a code of blinking LED. (Variable not defined)

27. Lab 2 – Task 3

28. ECU communication code (from unit 3).