Introduction To OSEK OS

Sarea Alhariri

Introduction To OSEK OS

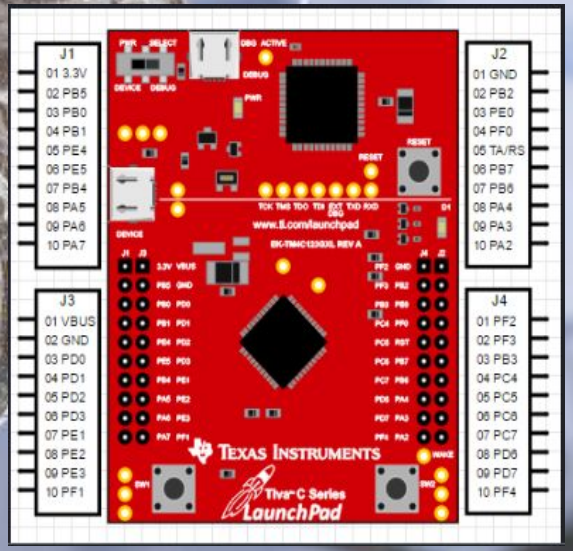
Prerequisites:

- **Basics of C programming**
- **Simple background in MCU programming**
- **Simple background in RTOS concepts**

Introduction To OSEK OS

Hardware & Tools:

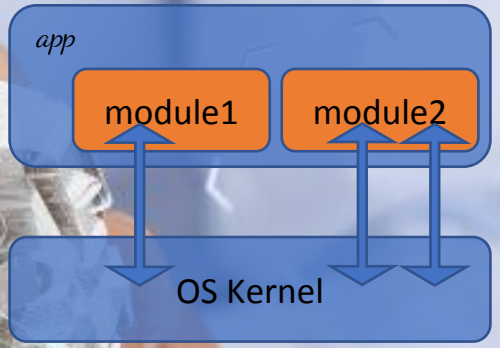
- Keil Uvision 5
- Tiva C Launchpad
- VSCode or Notepad++ for code editing

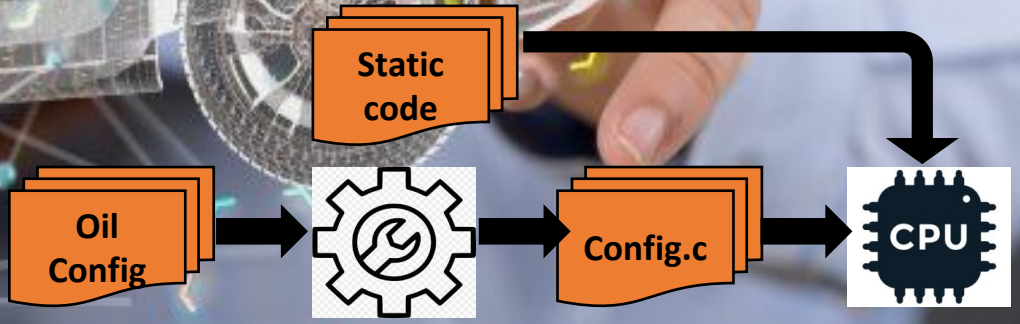


Introduction To OSEK OS

What is OSEK?

- small, scalable RTOS.
- Standard interfaces between the app and the OS.
- Different conformance classes, various scheduling mechanisms
- Statically scaled/configured at the system generation time



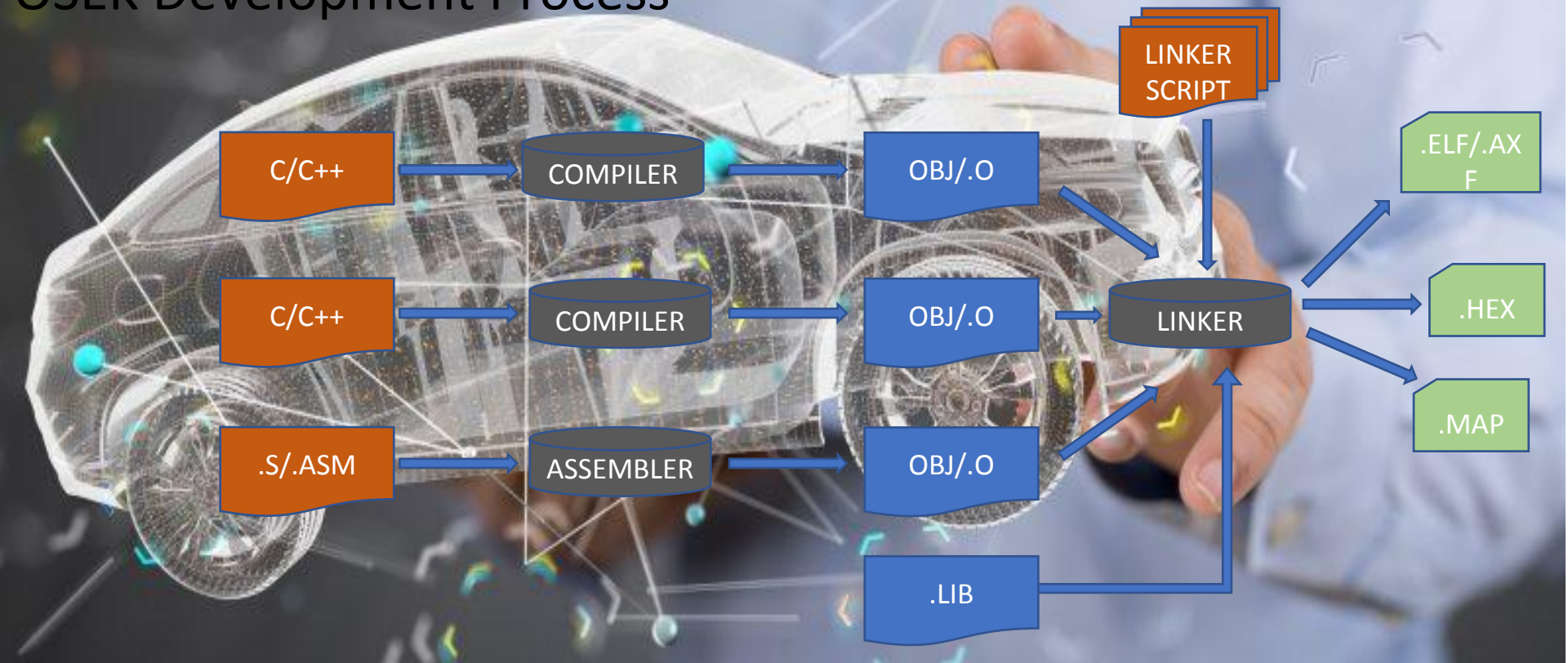


Introduction To OSEK OS

OSEK Features

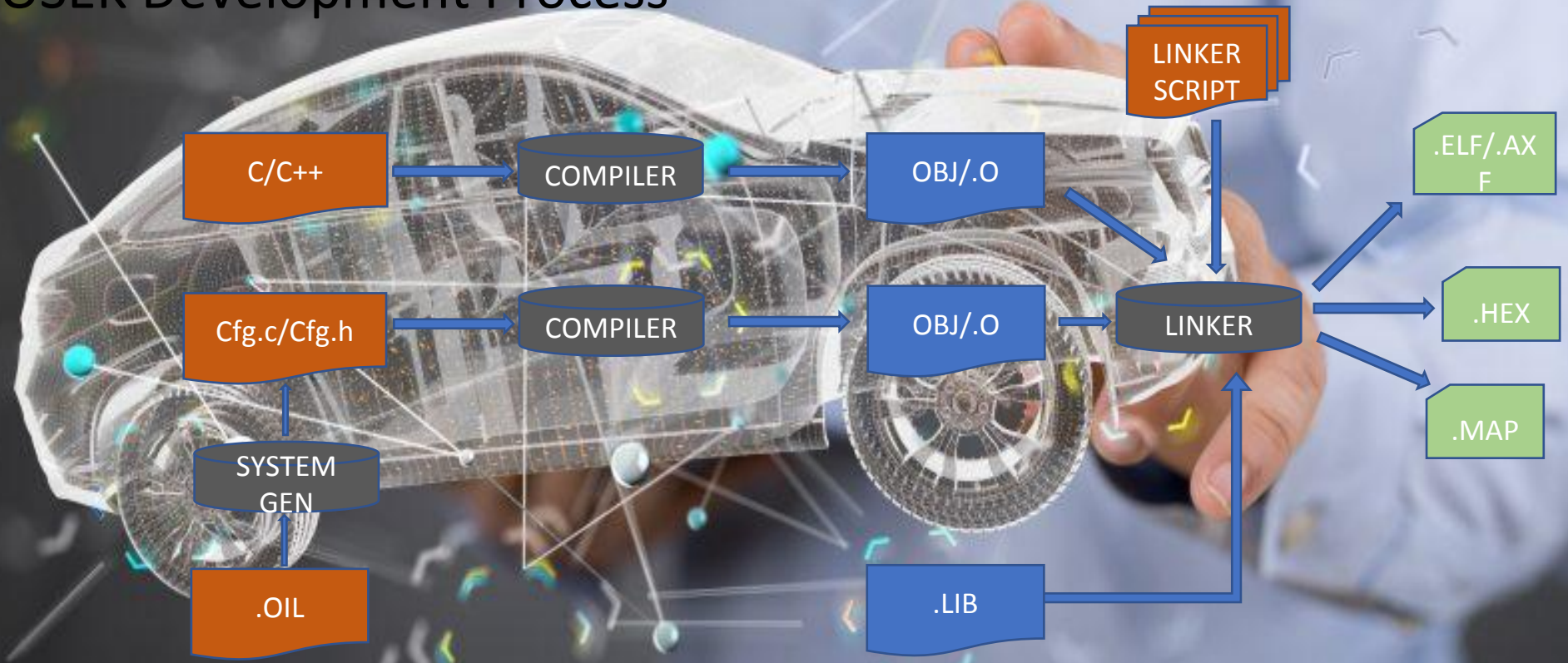
Introduction To OSEK OS

OSEK Development Process



Introduction To OSEK OS

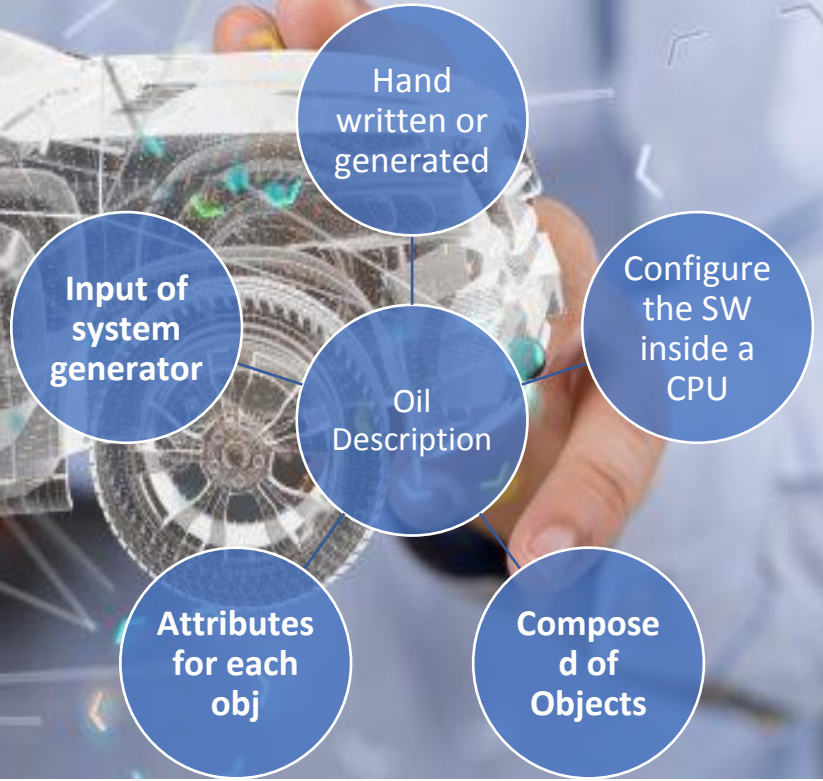
OSEK Development Process



Introduction To OSEK OS

OSEK Implementation Language (OIL)

```
CPU CORTEX-M4 {
        OS OSEKOS {
                STATUS = EXTENDED ;
                ...
        };
        TASK MyTask{
                PRIORITY = 45;
                SCHEDULE = NON;
                ...
        }
        RESOURCE ResourceExample{
                RESOURCEPROPERTY = STANDARD;
        }
        ...
        ...
}
```



Introduction To OSEK OS

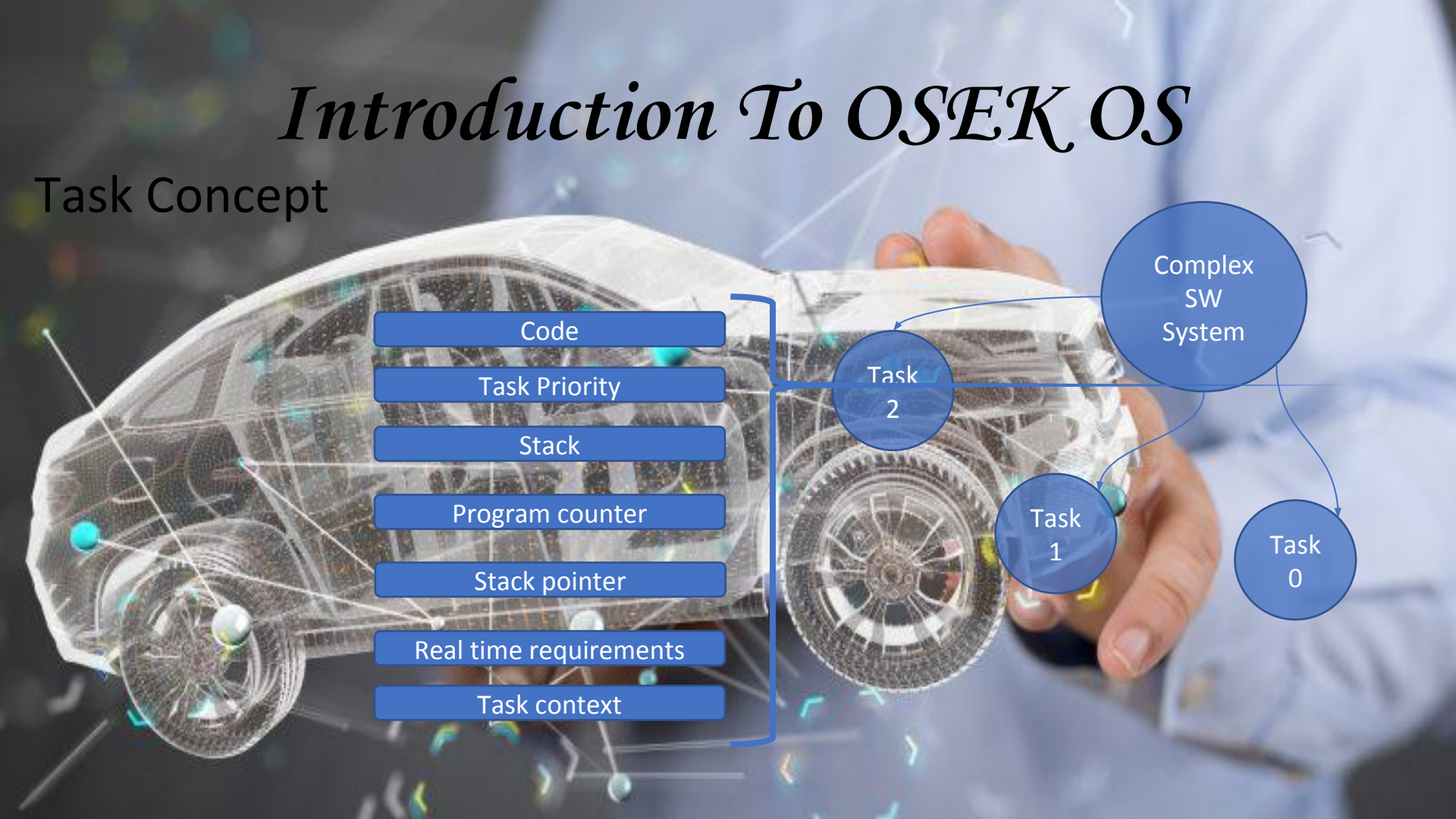
OSEK Implementation Language (OIL)

```
CPU CORTEX-M4 {
    OS OSEKOS {
      STATUS = EXTENDED ;
            STARTUPHOOK = FALSE;
      ERRORHOOK = FALSE;
      STACKOVERFLOWHOOK = TRUE;
      SHUTDOWNHOOK = FALSE;
      PRETASKHOOK = TRUE;
      POSTTASKHOOK = TRUE;
      USEGETSERVICEID = FALSE;
      USEPARAMETERACCESS = FALSE;
      USERESSCHEDULER = FALSE;
      SYSTEMTICKMS = 10
    };
```

```
TASK TaskA {
      PRIORITY = 10;
      SCHEDULE = FULL;
      ACTIVATION = 1;
      AUTOSTART = TRUE {APPMODE = AppMode1;};
      STACKSIZE = 50;
```

Introduction To OSEK OS

Task Concept



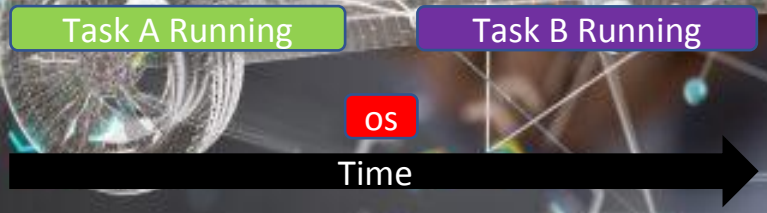
Introduction To OSEK OS

OSEK Task Management

TASK ShapeUpdate{

PRIORITY = 10;

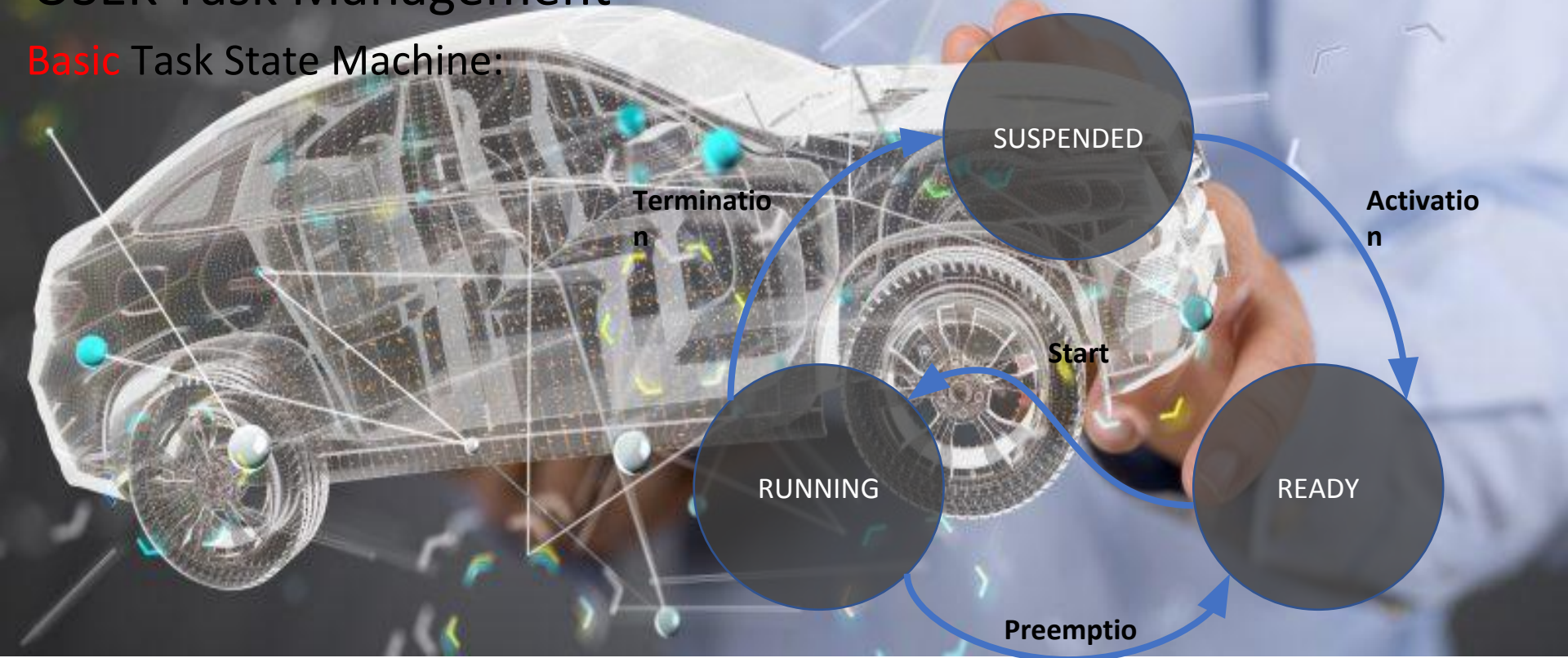
SCHEDULE = FULL/NON;



Introduction To OSEK OS

OSEK Task Management

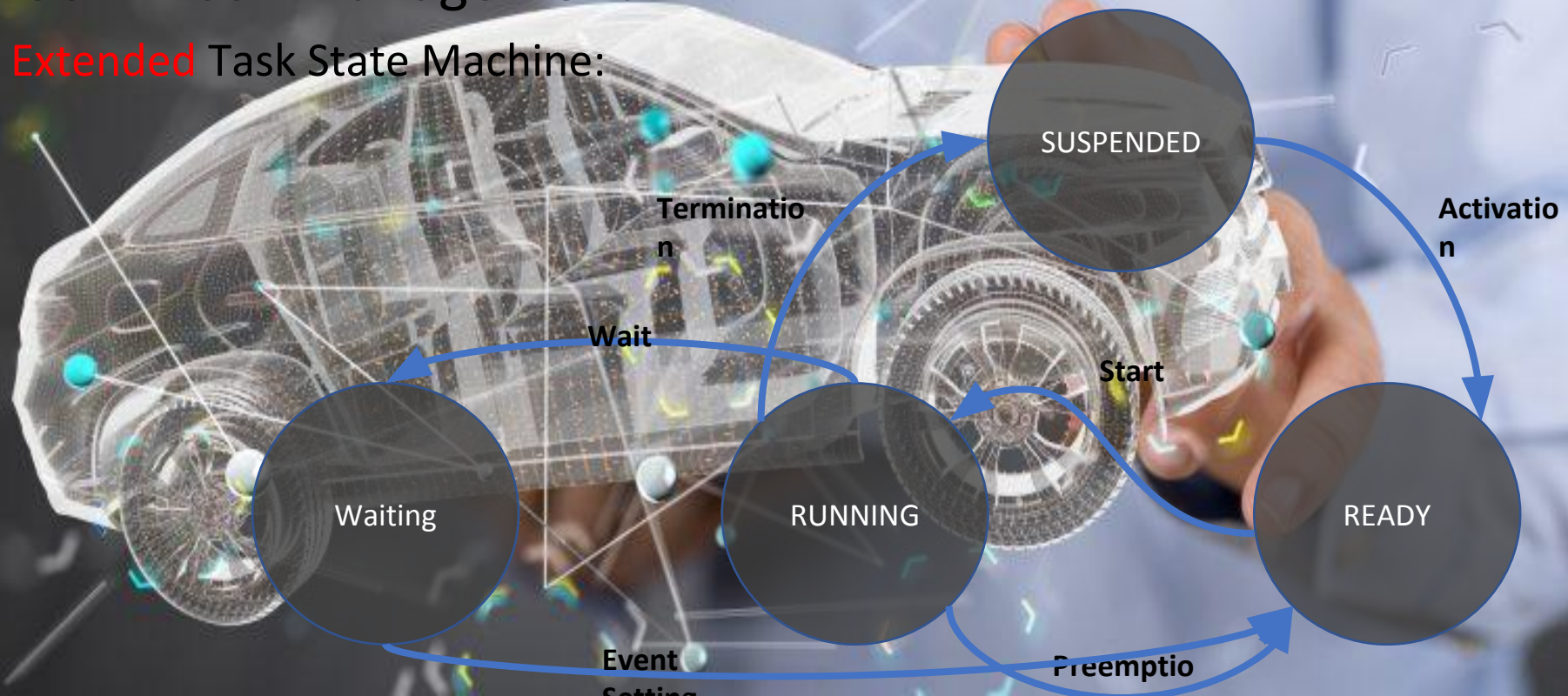
Basic Task State Machine:



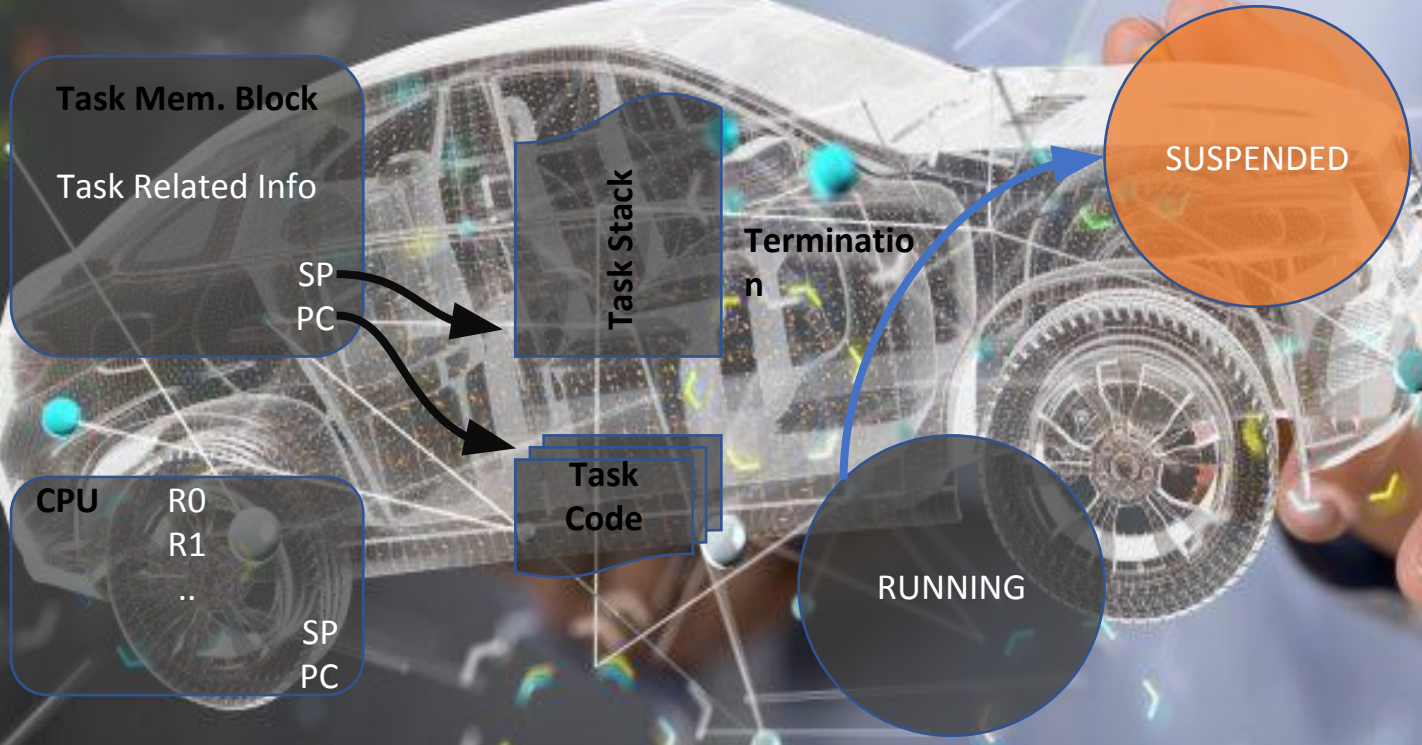
Introduction To OSEK OS

OSEK Task Management

Extended Task State Machine:



Introduction To OSEK OS



Introduction To OSEK OS

Introduction To OSEK OS

Introduction To OSEK OS

Conformance Classes

- 4 conformance classes to satisfy different requirements
- The conformance class is determined based on the following:
  - Multiple activation request of task
  - Task types
  - Number of tasks per priority level

Introduction To OSEK OS

Conformance Classes

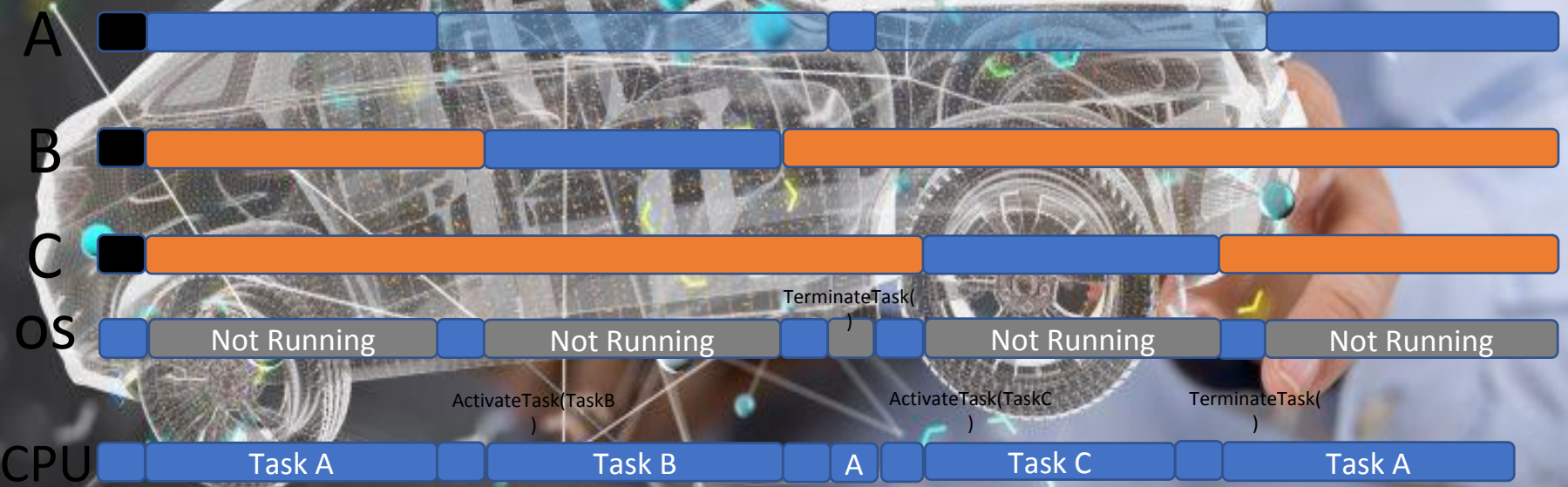
ECC 1:

- **Basic / extended tasks are supported**
- **One task per priority Level**
- **No multiple activation**

	Basic task only	Basic& extended task
One Task/Priority No multiple activation	BCC 1	ECC 1
Many Task/Priority Multiple activation	BCC 2	ECC 2

Introduction To OSEK OS

Task Management Ex1



Introduction To OSEK OS

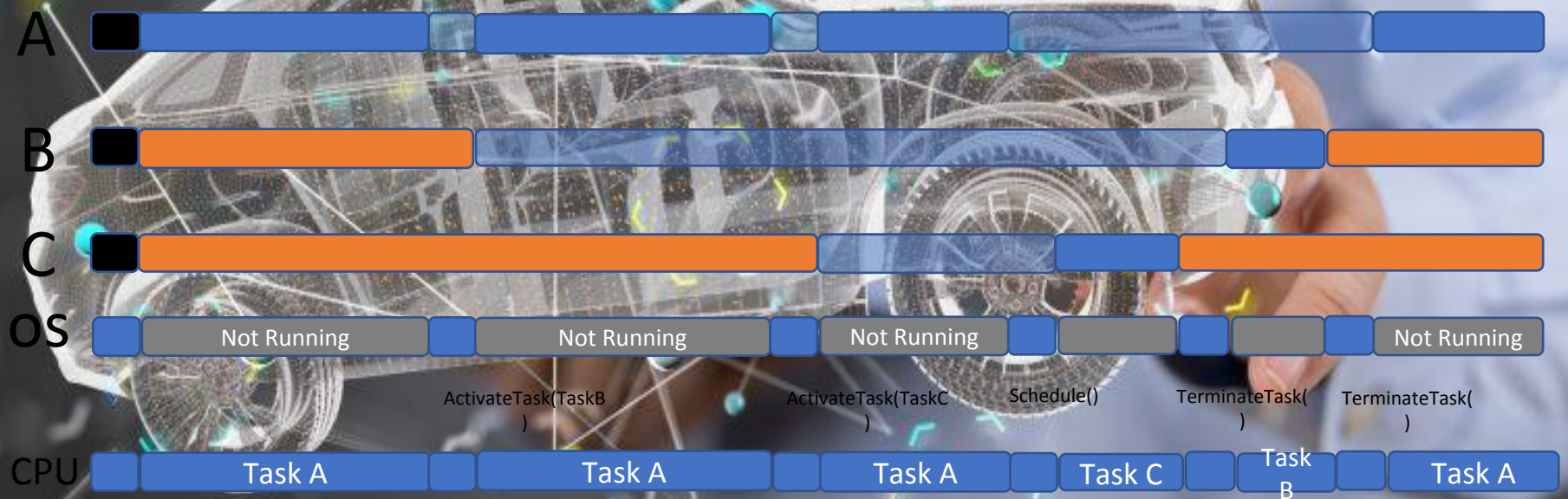
Task Management Ex1

- Oil description understand
- Code understanding
- Run/debug

Introduction To OSEK OS

Task Management Ex2

Preemptible /non preemptible Task : OSEK provides “preemptability” as a task attribute



Introduction To OSEK OS

Task Management Ex2

Oil description understand

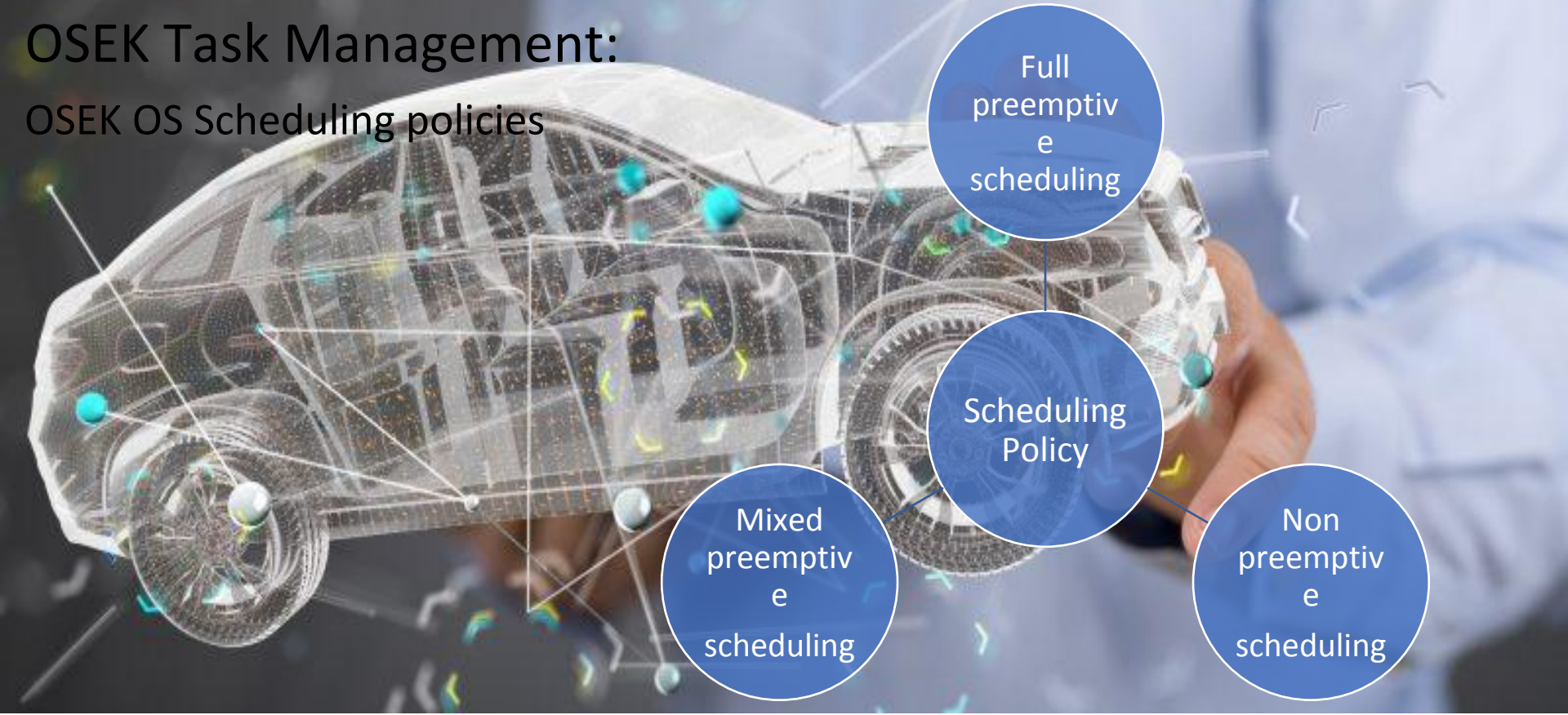
Code understanding

Run/debug

Introduction To OSEK OS

OSEK Task Management:

OSEK OS Scheduling policies



Introduction To OSEK OS

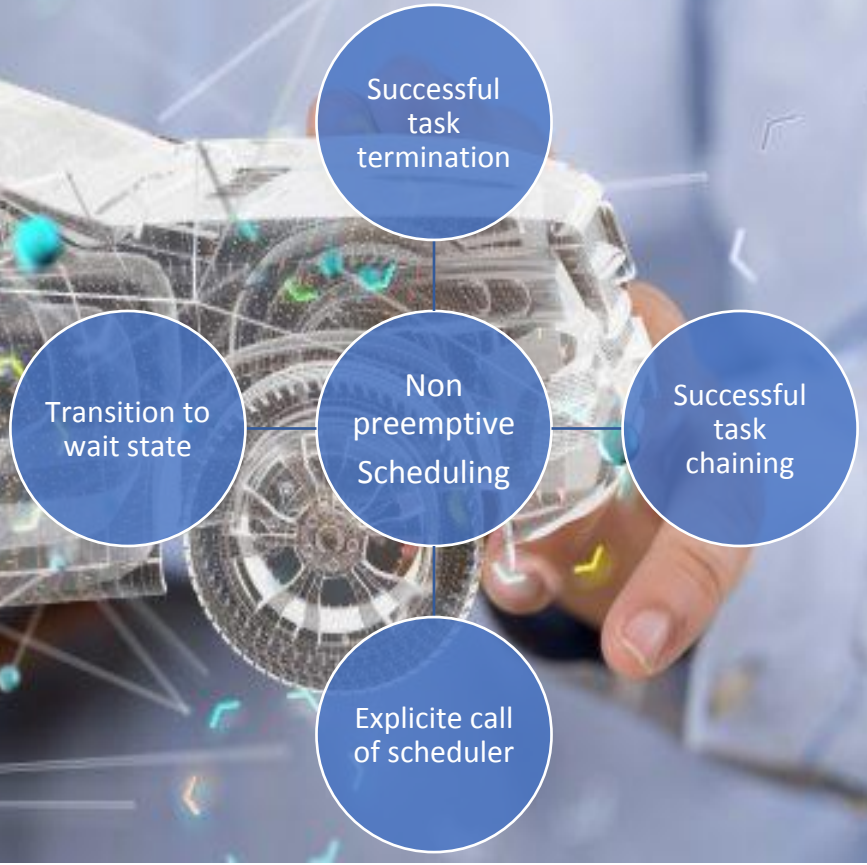
OSEK Task Management:

Full preemptive scheduling

Introduction To OSEK OS

OSEK Task Management:

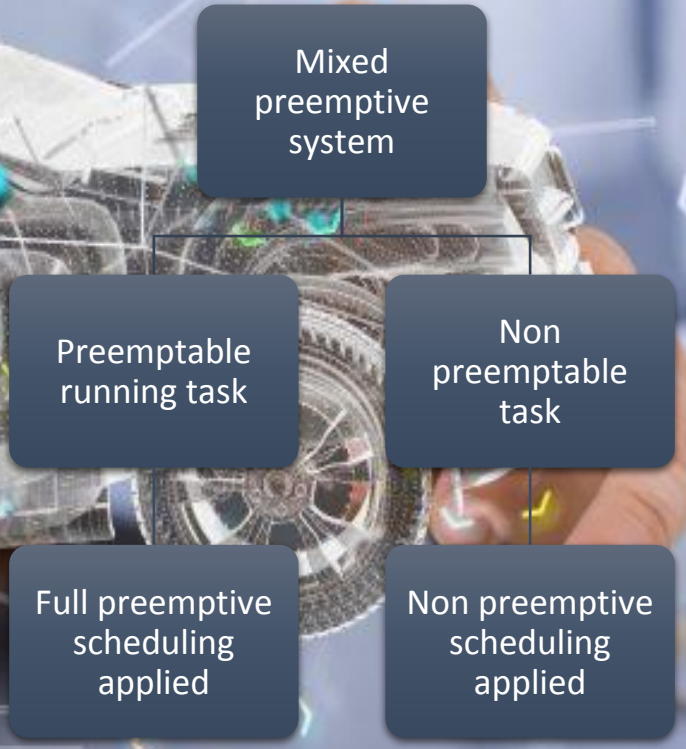
Non preemptive scheduling



Introduction To OSEK OS

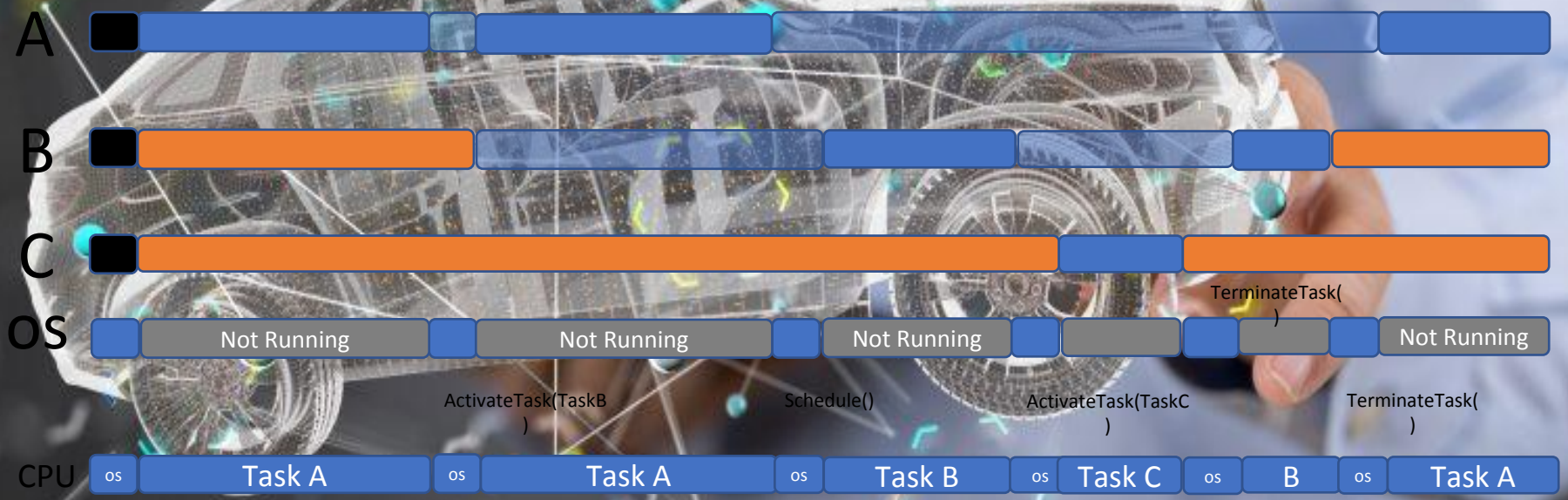
OSEK Task Management:

Mixed preemptive scheduling



Introduction To OSEK OS

Task Management Ex3



Introduction To OSEK OS

Task Management Ex3

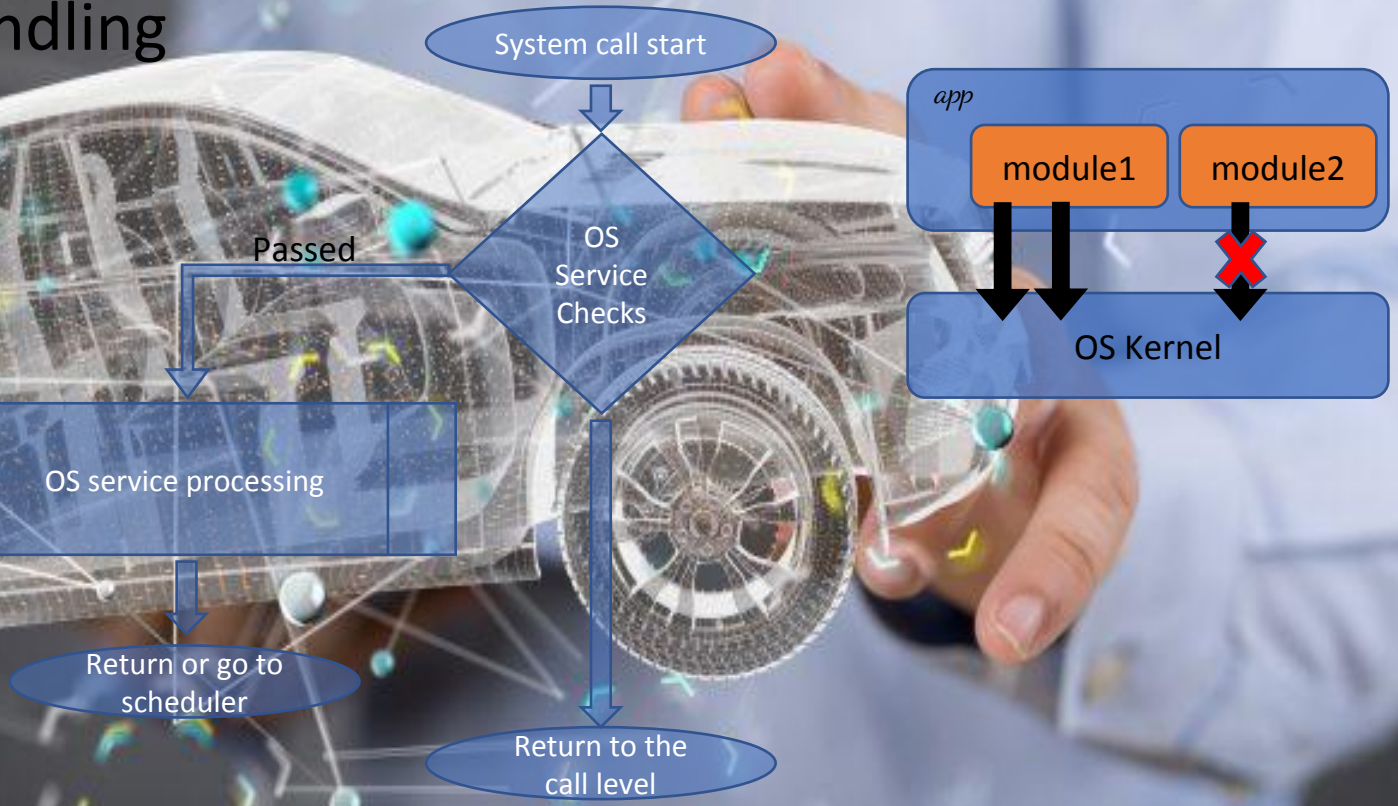
Oil description understand

Code understanding

Run/debug

Introduction To OSEK OS

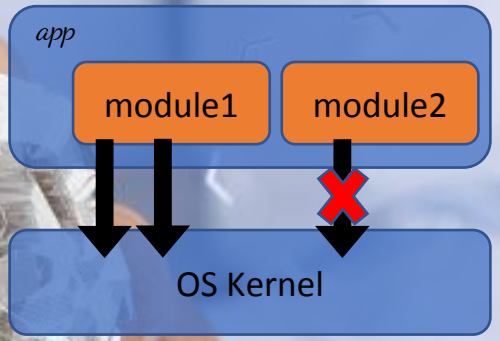
OSEK Error Handling



Introduction To OSEK OS

OSEK Error Handling

```
OS SOSEK {
        STATUS = EXTENDED;
            ...
            ..
```



Introduction To OSEK OS

OSEK Task Management:

Task management services:

StatusType ActivateTask(TaskType Task)

- E_OS_LIMIT
- E_OS_ID
- E_OK

Introduction To OSEK OS

OSEK Task Management:

Task management services:

StatusType TerminateTask(void)

E_OS_RESOURCE

E_OS_CALLLEVEL

Introduction To OSEK OS

OSEK Task Management:

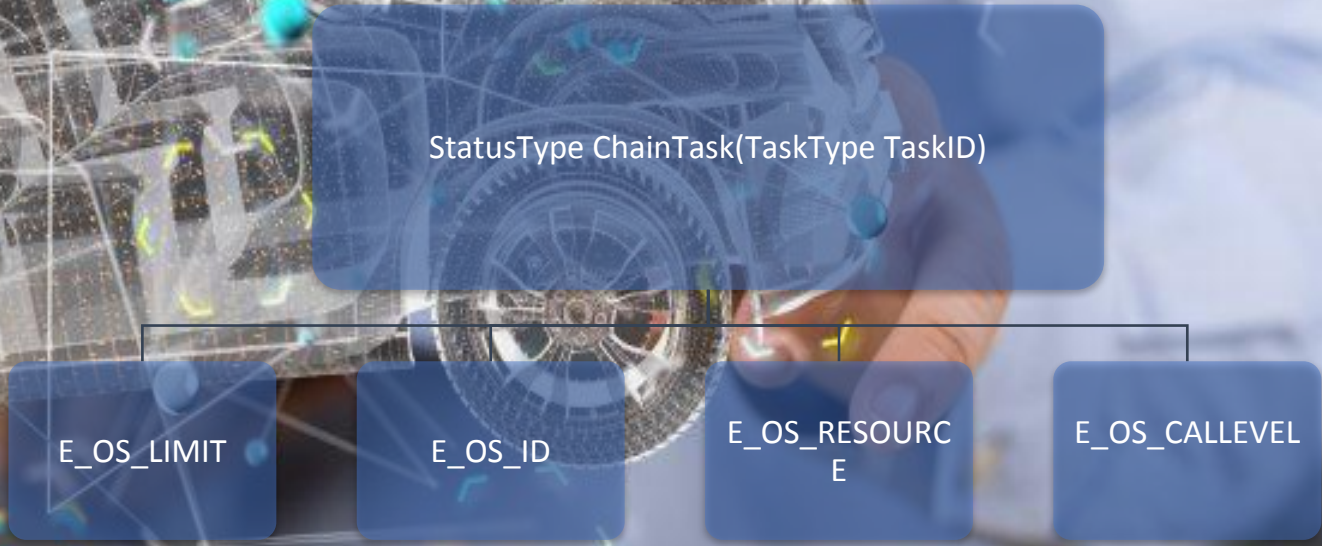
Task management services:

- Task can only terminate itself
- Task should terminate itself at the end of its code
- Ending the task without a call to TerminateTask or ChainTask **is not allowed**

Introduction To OSEK OS

OSEK Task Management:

Task management services:



Introduction To OSEK OS

OSEK Task Management:

Task management services:

StatusType Schedule(void)

- E_OS_CALLEVEL
- E_OK
- E_OS_RESOURCE

Introduction To OSEK OS

OSEK Task Management:

Task management services:

- Checks for the highest priority ready task
- The system return to the caller only when ALL higher priority tasks are done
- Schedule() service has no effect on the preemptable tasks

Introduction To OSEK OS

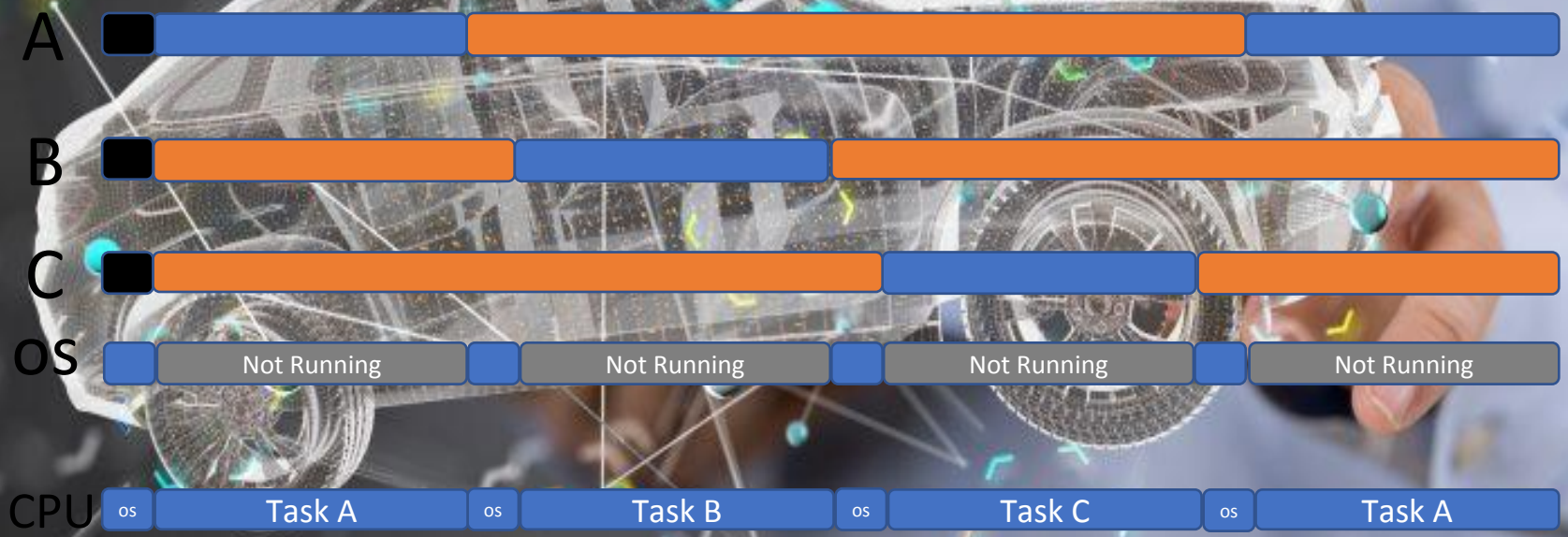
OSEK Task Management:

Task management services:

- StatusType GetTaskID(TaskRefType TaskID)
- StatusType GetTaskState(TaskType TaskID, TaskStateRefType State)
- DeclareTask(TaskIdentifier)
- TASK(TaskID)

Introduction To OSEK OS

Task Management Ex4



Introduction To OSEK OS

Task Management Ex4

Oil description understand

Code understanding

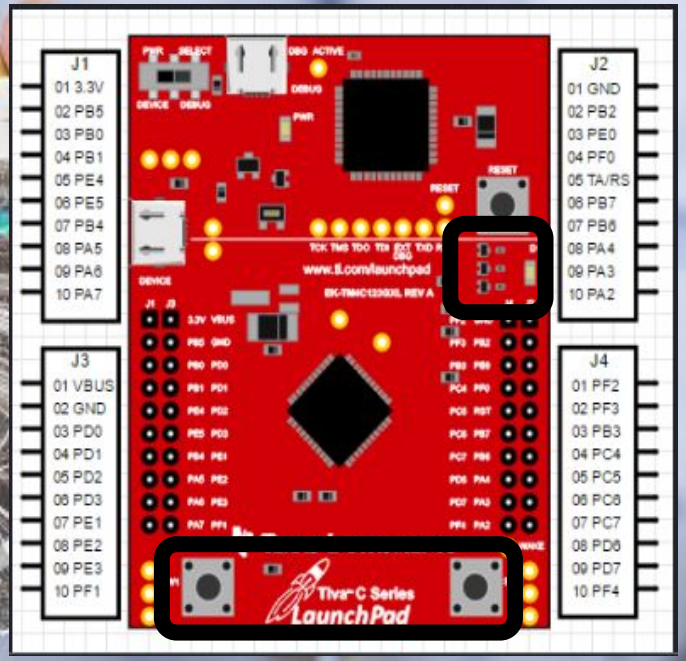
Run/debug

Introduction To OSEK OS

Task Management Ex5

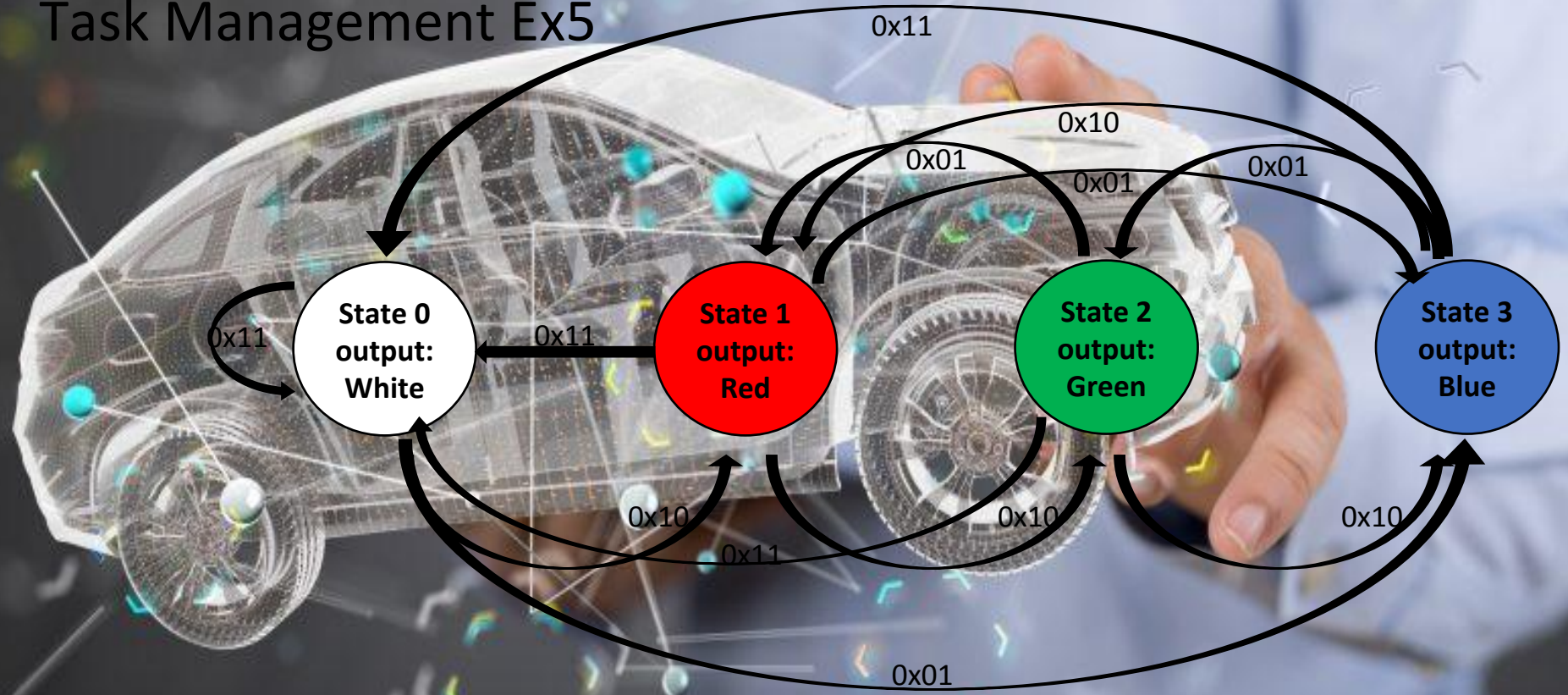
LED simple state machine:

- Input: 2 onboard switches
- Output: 3 LEDs
- Possible state: 4 States



Introduction To OSEK OS

Task Management Ex5



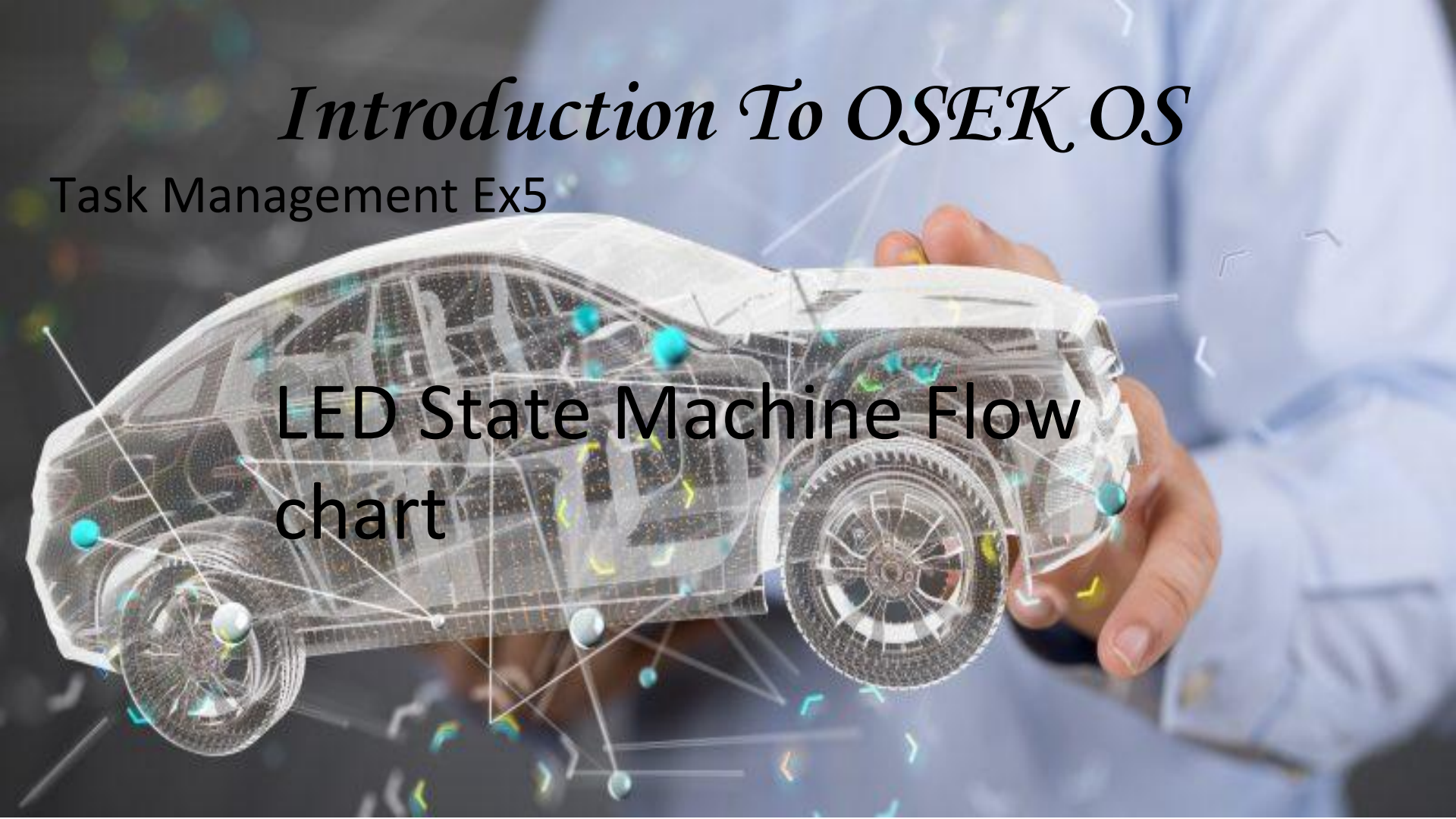
Introduction To OSEK OS

Task Management Ex5

SW 1	Not Pressed	Pressed	Pressed	Pressed	Not Pressed	Not Pressed	Not Pressed	Not Pressed	Not Pressed	Not Pressed
SW 2	Not Pressed	Not Pressed	Not Pressed	Not Pressed	Not Pressed	Pressed	Pressed	Pressed	Pressed	Not Pressed
State										

Introduction To OSEK OS

Task Management Ex5



LED State Machine Flow chart

Introduction To OSEK OS

Task Management Ex5

Oil description understand

Code understanding

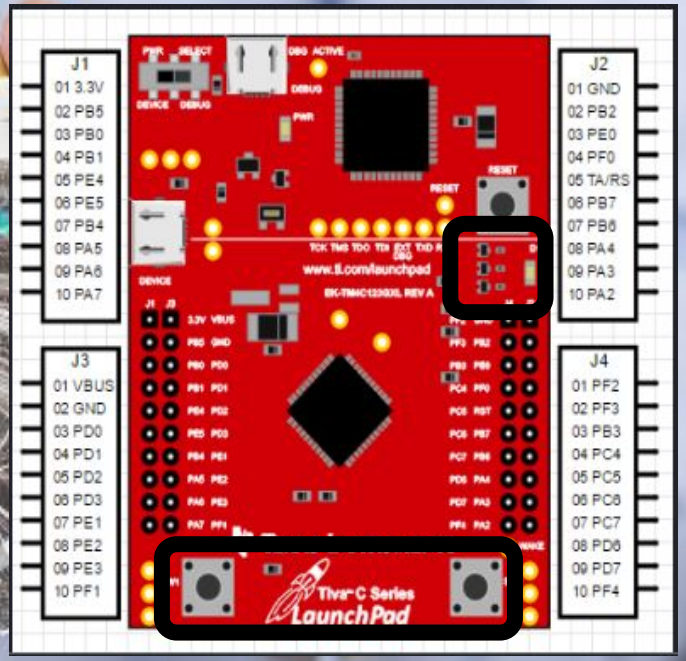
Run/debug

Introduction To OSEK OS

Task Management Assignment

LED sequence state machine:

- Input: 2 onboard switches
- Output: 3 LEDs
- Four Tasks shall be implemented:
  - Task_Init
  - Task_BasicState
  - Task_RGB
  - Task_BGR



Introduction To OSEK OS

Task Management Ex5

LED Sequence State Machine Flow chart

Introduction To OSEK OS

Task Management Assignment 2

High level problem statement

- Alarm should be triggered if at least one of the doors is not closed
- Alarm should not be triggered if both of the doors are closed

Assumptions

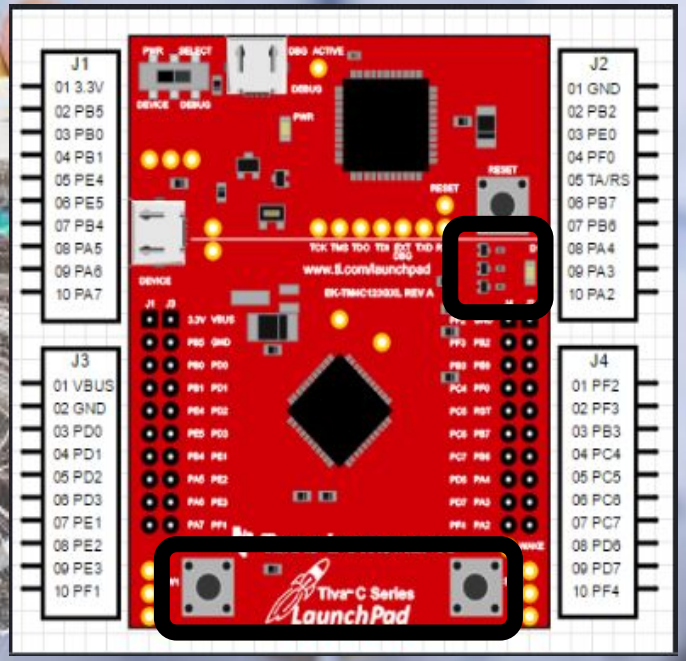
- The system is composed of 2 doors and 1 alarm
- No timing requirements

Introduction To OSEK OS

Task Management Assignment 2

Simplify the problem

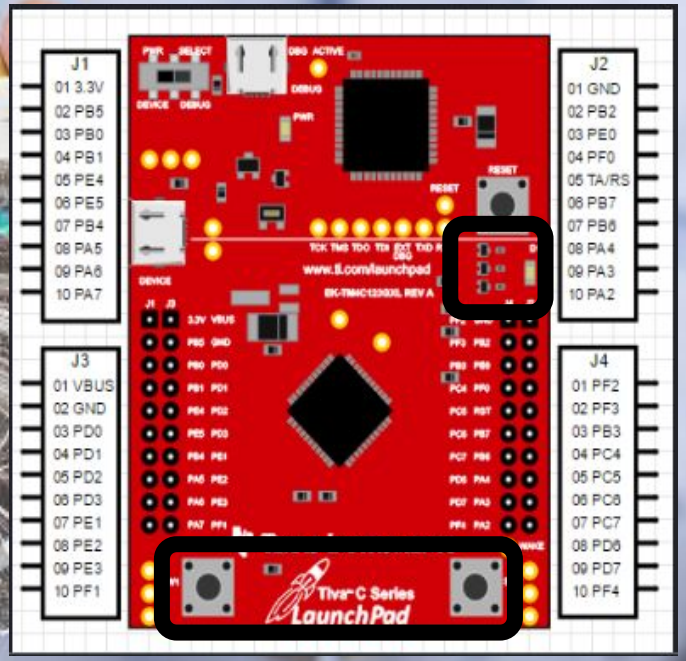
- We will assume the sensors and alarm are on the same ECU
- We will use the on board switches to represent the door sensors
- We will use the on board red LED to represent the alarm activation



Introduction To OSEK OS

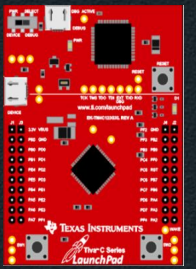
Task Management Assignment 2

- Write your own oil description file
- Use a different threads to achieve the whole functionality
- Try to re-use from the previous examples(oil/code.. etc.).



Intro to OSEK RTOS I

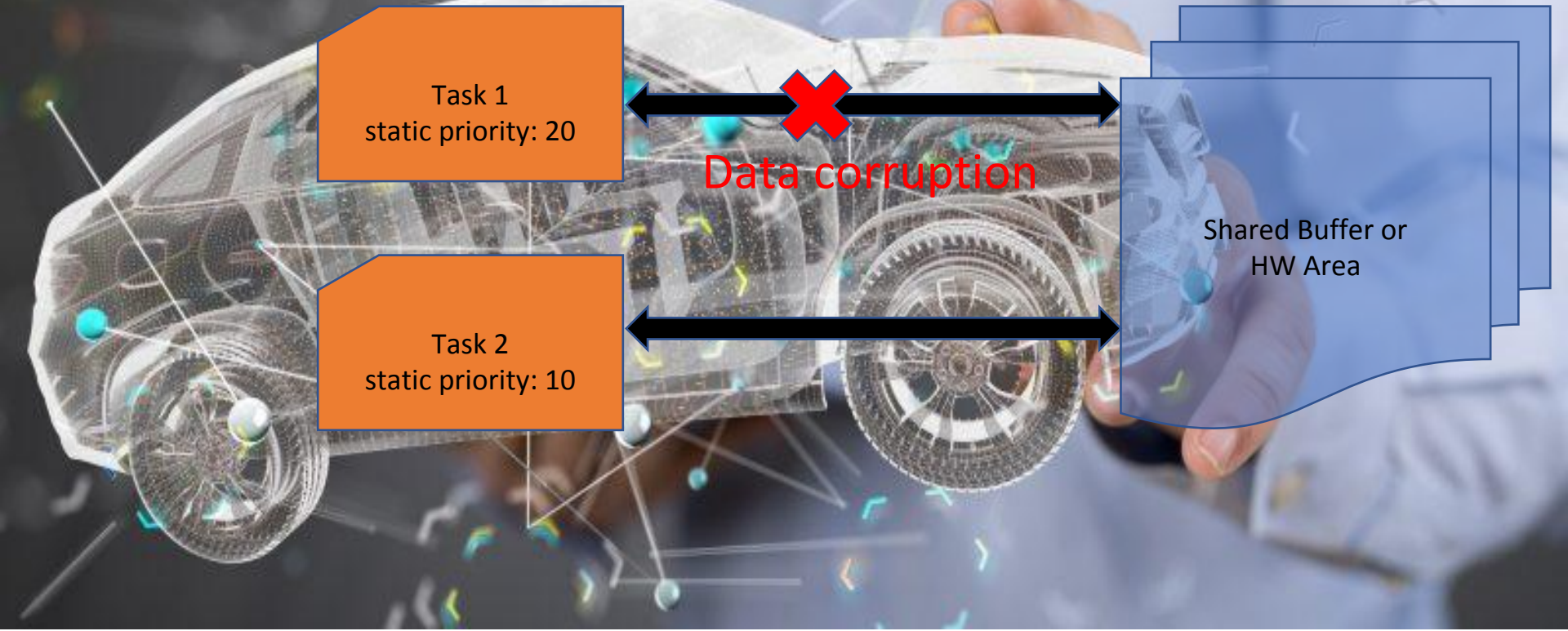
```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
            continue;
        } else if (iN == (iLength - 3) ) {
            continue;
```



OSEK Generator

Introduction To OSEK OS

Multithreading and concurrent access



Introduction To OSEK OS

Multithreading And Racing

0x20000020	00	00	00	0B

R0	0x0B
R1	0x20000020

```
; …
LDR  R0 , [R1]
ADD  R0, R0, #1
STR  R0, [R1]
; …
```

0x20000020	00	00	00	0B

RMW_Task CPU Context
static priority: 10

CntInitTask CPU context
static priority: 20

```
uint32_t g_u32_Counter = 0U;
TASK(RMW_Task )
{
    // …
    // increments by 1
    g_u32_Counter+=1U;
    //…
}
TASK(CntInitTask)
{
    // …
    if(/*some condition*/)
    {
      /* re init the counter */
      g_u32_Counter = 0U;
    }
    //…
}
```

Introduction To OSEK OS

Multithreading And Racing

0x20000020	00	00	00	F1

R0	0xF1
R1	0x20000020

```
; …
LDRB  R0 , [R1]
ORR   R0, R0, #1
STRB  R0, [R1]
; …
```

0x20000020	00	00	00	F1

Bit0_Task CPU Context
static priority: 10

Bit1_Task CPU context
static priority: 20

```
uint8_t EnableMask= 0x00U;
TASK(Bit0_Task )
{
    // …
    // EnableMask is 0xF0
    //Enable bit 0
    EnableMask |= 0x01
    //…
}

TASK(Bit1_Task)
{
    // …
    if(/*some condition*/)
    {
      //Enable bit 1
    EnableMask |= 0x02;
    }
    //…
}
```

Correct EnableMask	***0xF3***	***0b1111 0011***
Actual EnableMask	***0xF1***	***0b1111 0001***

Introduction To OSEK OS

Multithreading And Data Inconsistency

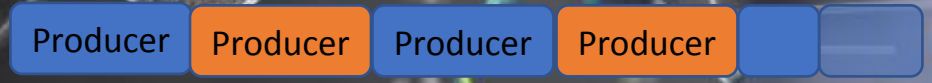
Producer Task
static priority: 10
Period: 10 ms

Time Stamp	99929.2123
ID	Client _0x01
Data[0]	0x3A
Data[1]	0xB0
…	…
Data[n]	…

Consumer Task
static priority: 15

- Time stamp: 9960.323
- ID: 0x04
- Data[0..n] = 0x2F, 0x43, 0x8E…etc.

- Time stamp: 9970.397
- ID: 0x02
- Data[0..n] = 0x2F, 0x43, 0x8E…etc.



Introduction To OSEK OS

Resource Example

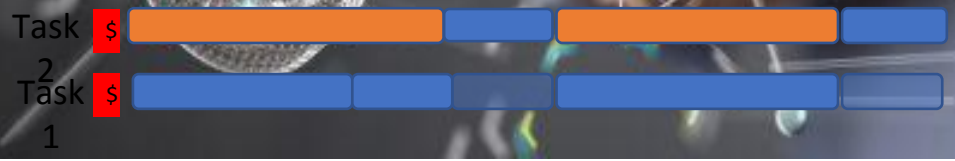
- Oil description understand
- Code understanding
- Run/debug

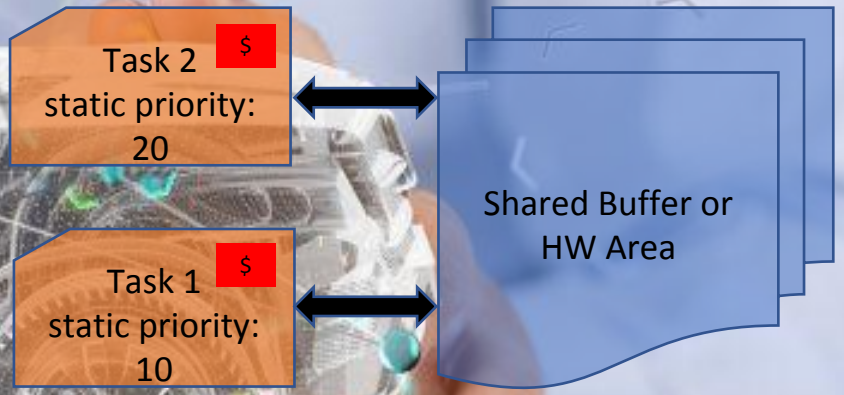
Introduction To OSEK OS

Semaphore, Mutex or Resource

```
uint32_t g_u32_Counter = 0U;
TASK(Task1)
{
        // ...
        //GetSemaphoreAPI()

        g_u32_Counter+=1U;
        //..
        // Accessing the shared
        // Resource

        //GiveSemaphoreAPI()
}
```

```
TASK(Task2)
{
        // ...
        //GetSemaphoreAPI()
        //..
        g_u32_Counter = 0U;
        //..
        // Accessing the shared
        // Resource

        //GiveSemaphoreAPI()
}
```



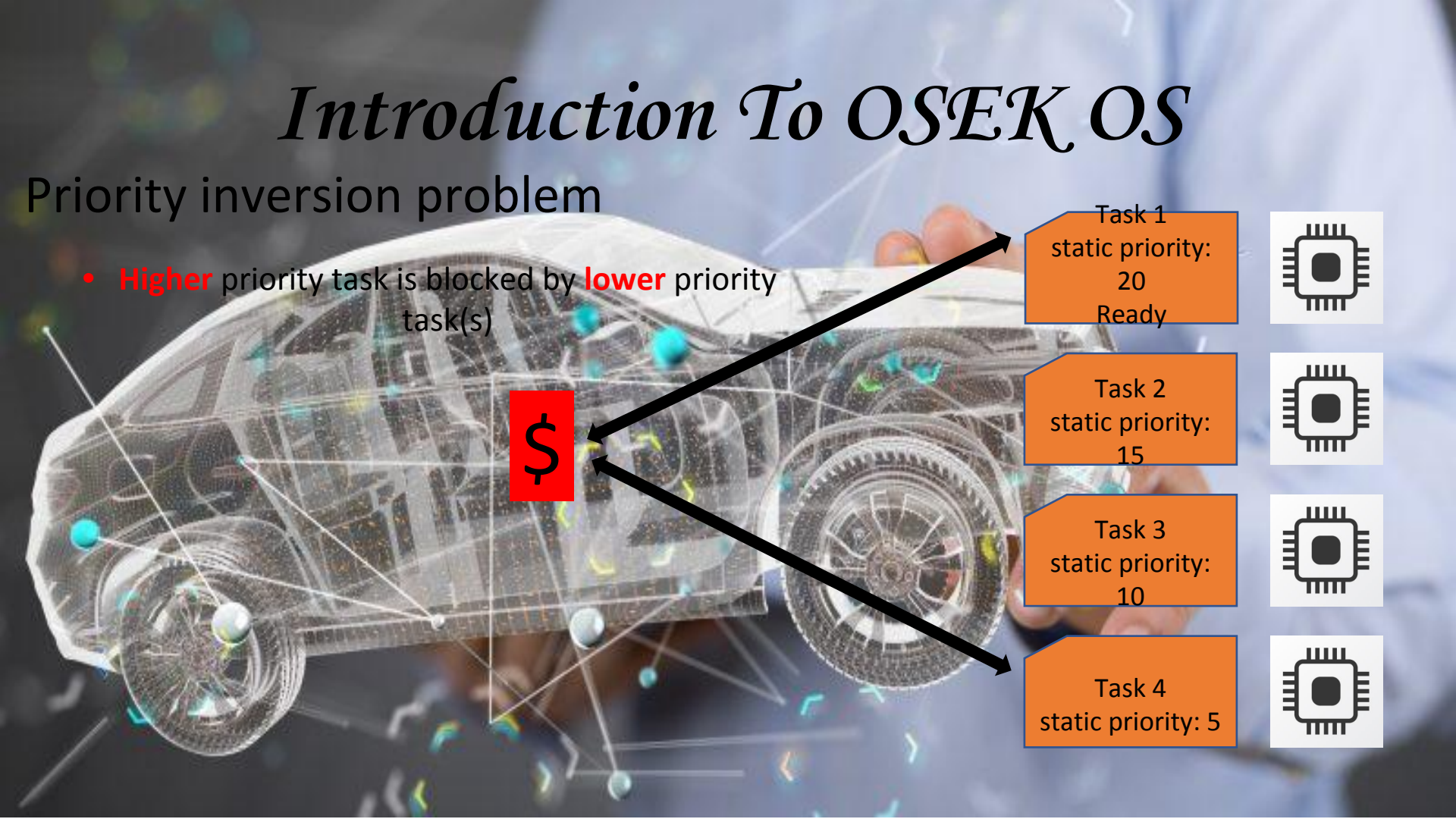


Only one task at a time

Introduction To OSEK OS

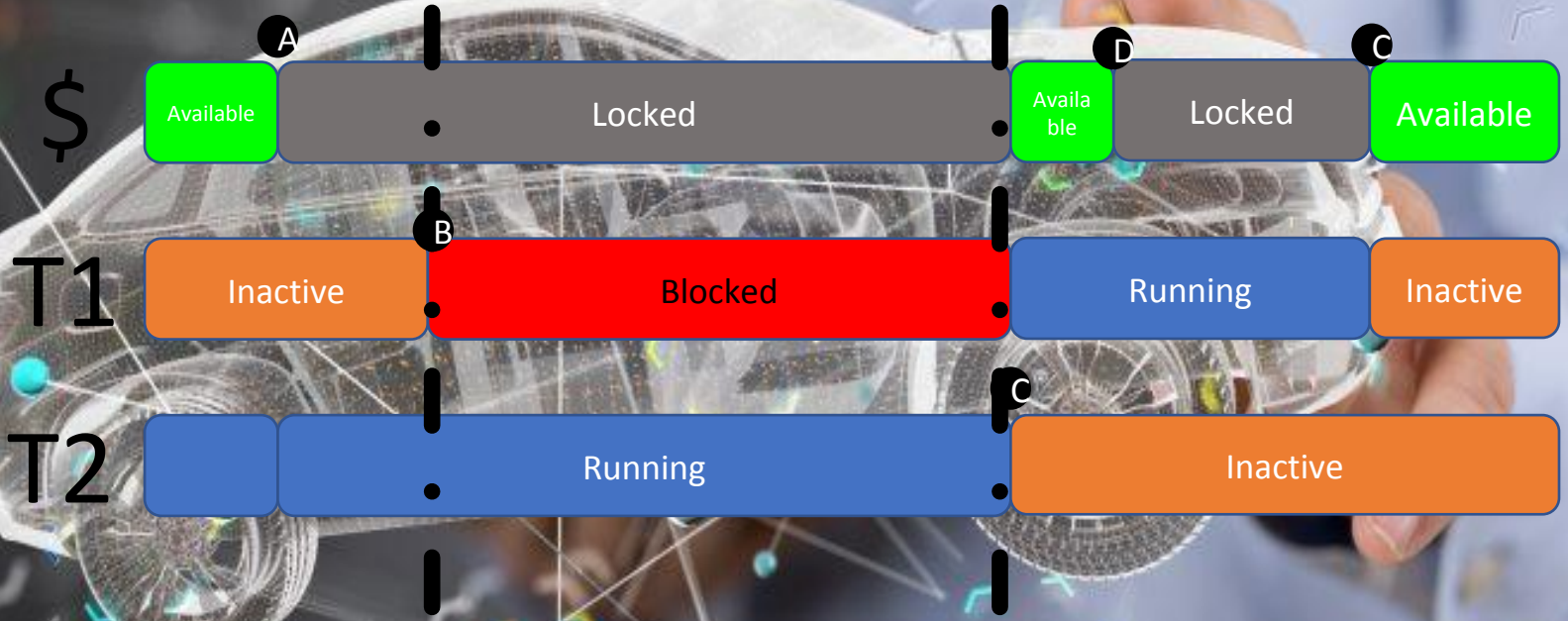
Priority inversion problem

- **Higher** priority task is blocked by **lower** priority task(s)



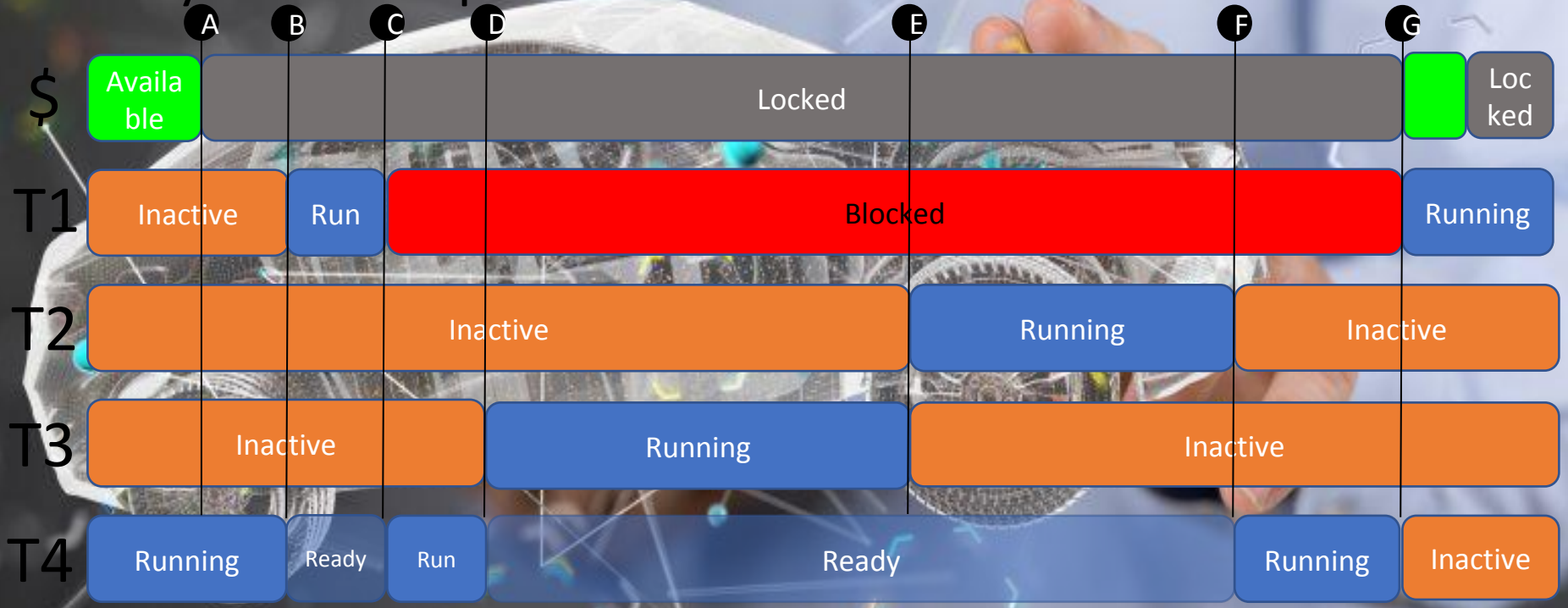
Introduction To OSEK OS

Priority inversion problem



Introduction To OSEK OS

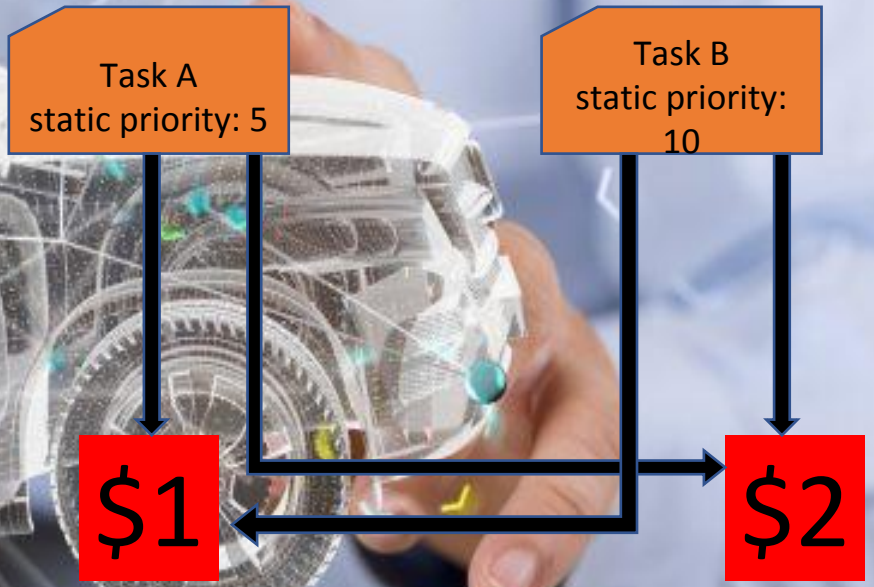
Priority inversion problem



Introduction To OSEK OS

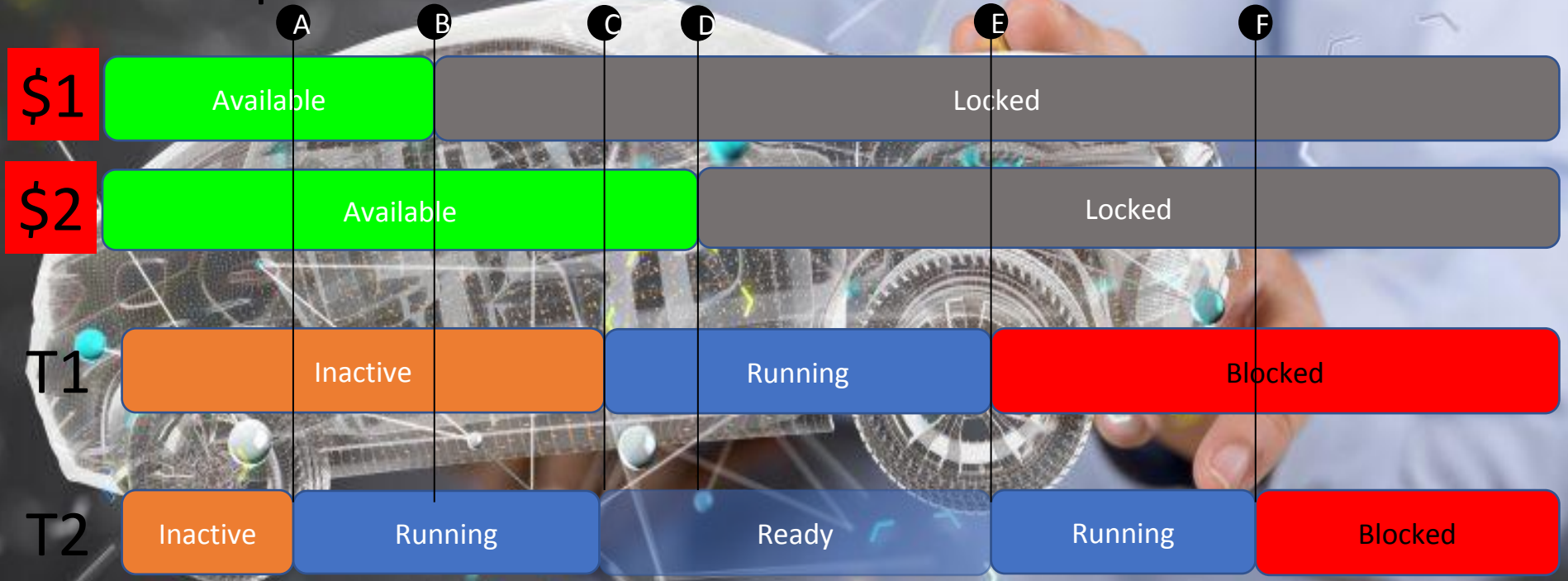
Deadlock problem

- **Task stopping & System freeze**
- **Impossible conditions**



Introduction To OSEK OS

Deadlock problem

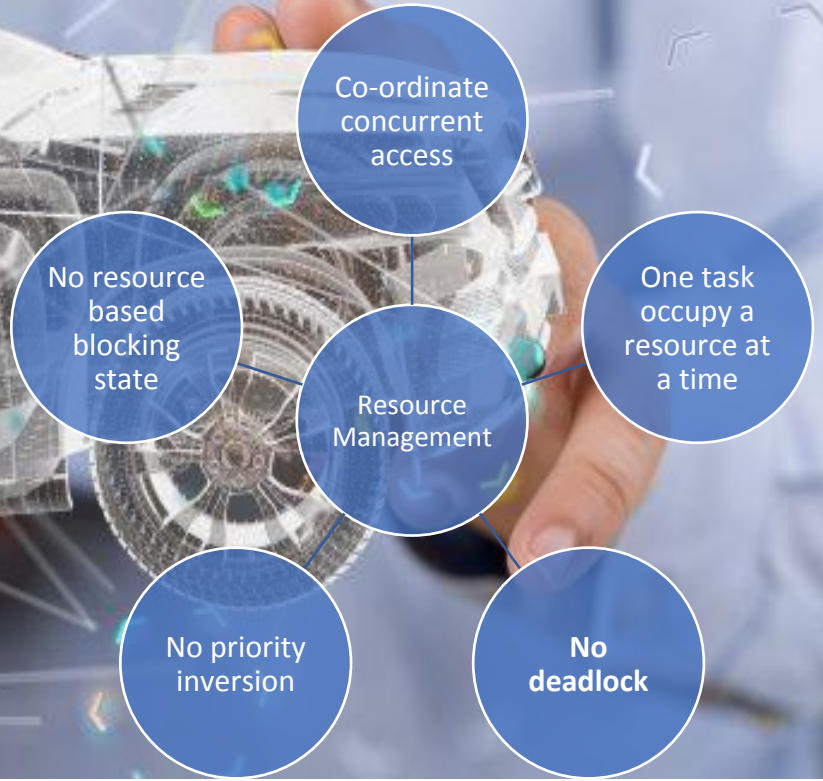


Introduction To OSEK OS

OSEK resource management

Priority Ceiling Pattern:

To avoid deadlocks and priority inversion



Introduction To OSEK OS

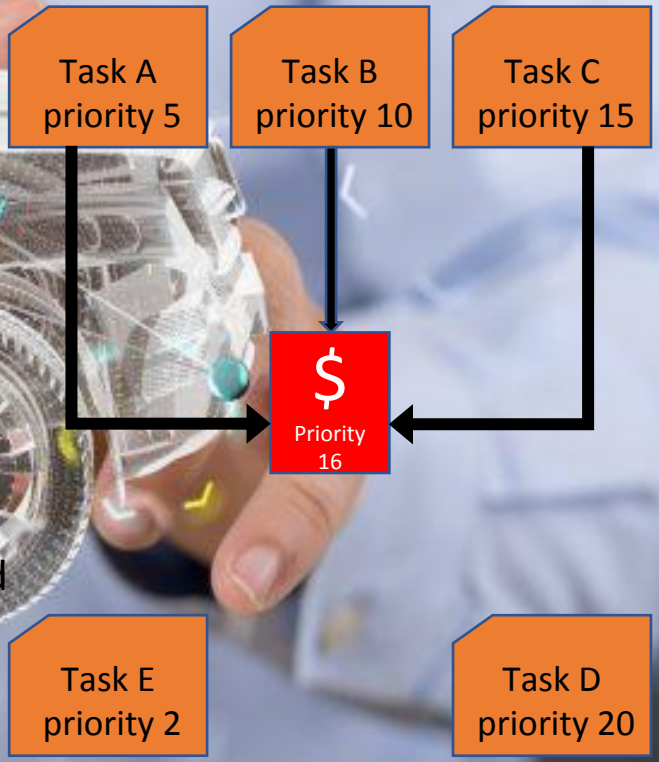
OSEK priority ceiling protocol

- ***At generation/compilation time***

1. Every $ □ Priority level (the ceiling priority)
2. Priority level > max user task priority
3. Priority level < min non user task priority (if higher than the highest user)

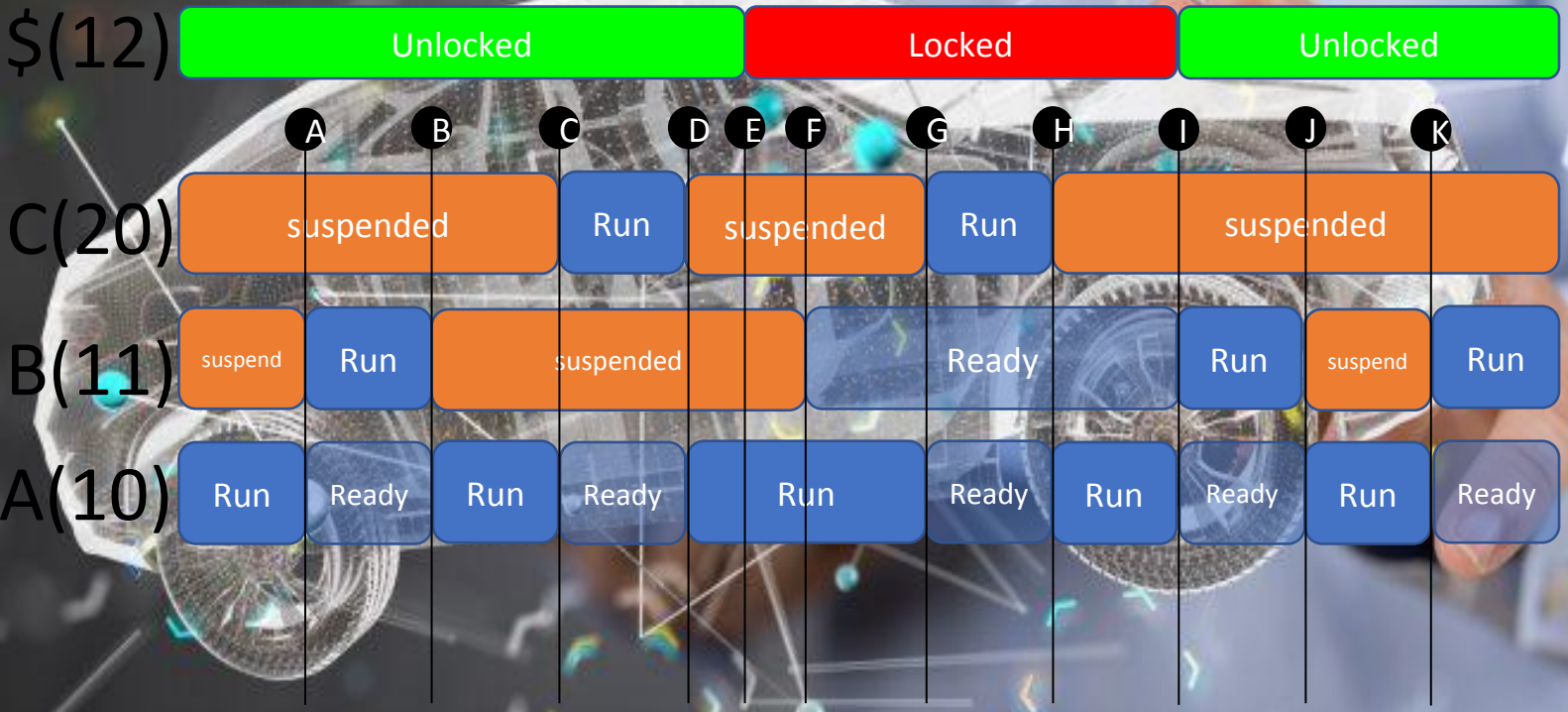
- **Run time behavior**

1. Requiring a resource □ execute at the ceiling priority level
2. Releasing a resource □ back to the nominal priority level
   □ Scheduling point if preemptable thread



Introduction To OSEK OS

Resource Management Example 1



Introduction To OSEK OS

Resource Example

- Oil description understand
- Code understanding
- Run/debug

Introduction To OSEK OS

OSEK resources APIs

StatusType GetResource (ResourceType <ResID>)

1. Entering the critical section
2. The Ceiling priority will be applied
3. The used error codes:
   1. E_OK
   2. E_OS_ID
   3. E_OS_ACCESS

StatusType ReleaseResource (ResourceType <ResID>)

1. Leave the critical section
2. The static priority will be applied
3. The used error codes:
   1. E_OK
   2. E_OS_ID
   3. E_OS_NOFUNC
   4. E_OS_ACCESS

Introduction To OSEK OS

OSEK resources APIs

Usage restriction while holding resource

- *TerminateTask -> E_OS_RESOURCE*
- *ChainTask -> E_OS_RESOURCE*
- *Schedule -> E_OS_RESOURCE*
- *WaitEvent -> E_OS_RESOURCE*

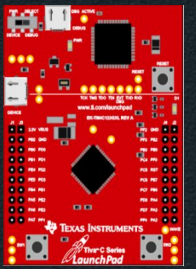
Introduction To OSEK OS

Resource Management Example 2

- Oil description understand
- Code understanding
- Run/debug

Intro to OSEK RTOS I

```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
            continue;
        } else if (iN == (iLength - 3) ) {
            continue;
```



OSEK Generator