LED Blink

on NXP EVB With AUTOSAR

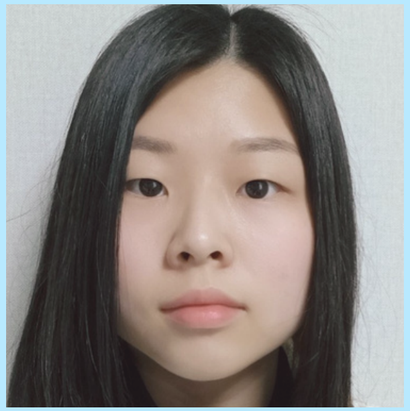
Group E

HL Mando

KYUNGPOOK NATIONAL UNIVERSITY 경북대학교

knu

TEAM MEMBERS



Sanghui Ko

Youngeun Seo

Yeongjin Jeong

Contents

Theory

Theory

Embedded system

It is integrated system that is formed as a combination of computer hardware and software for a specific function.

Hardware + Software + Firmware

Theory

Microcontroller (MCU)

- A small computer.
- It combines the functions of a central processing unit (CPU), memory, and input/output interfaces, all on a single chip.
- It’s programmable, then can be customized to perform specific tasks.

Microprocessor, RAM, ROM, I/O port

Theory

These MCU has instruction set.

Instruction set is a set of commands that a microcontroller can understand and execute.

ADD	SUB	MOVE	JUMP
add the first and the second elements	subtraction the first to the second elements	copy the first element to the second element	If it meet condition, program counter change to set counter

Theory

Register : A memory device that temporarily stores data necessary for the CPU to process requests.

Program counter register : A register that holds the address of the next instruction to be executed.

Status register : A register used to test for various conditions during an operation.

Theory

Memory Segments

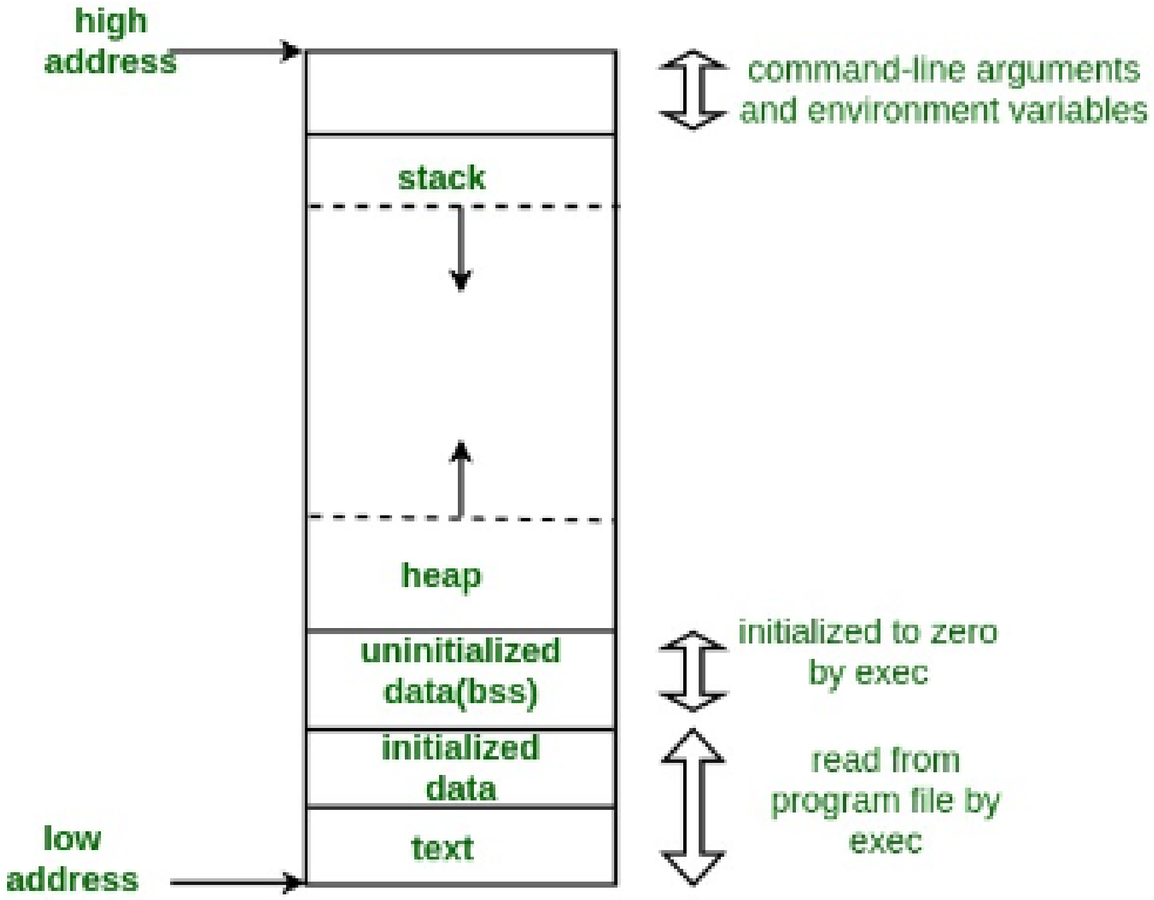
There are four segments :

stack
- automatic variables are stored, along with information that is saved each time a function is called

heap
- It save a dynamic memory (e.g. array, pointer)

data
- initialized data (e.g. int a = 10;)
- uninitialized data : save garbage value
  before program start executing, value is setting 0 by kernel
  (e.g. int a;)

text
- code segment
- It contains executable instructions



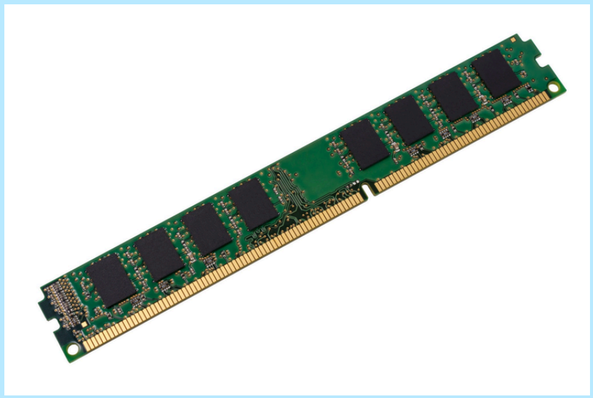
Theory

Two type of memory : volatile, non-volatile

volatile

Feature : If power is off, It lost storage data

- RAM (Random Access Memory)

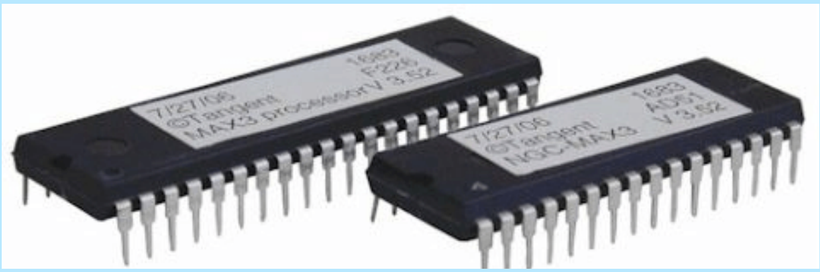


non-volatile

Feature : If power is off, It still store data

- ROM (Read Only Memory)

Example : PROM, EPROM, EEPROM, FLASH, D-FLASH



Theory

Pull-up resistor

- Originally, Voltage is set to logic level High(1).
- If switch is pushed, then It set to logic level Low(0).

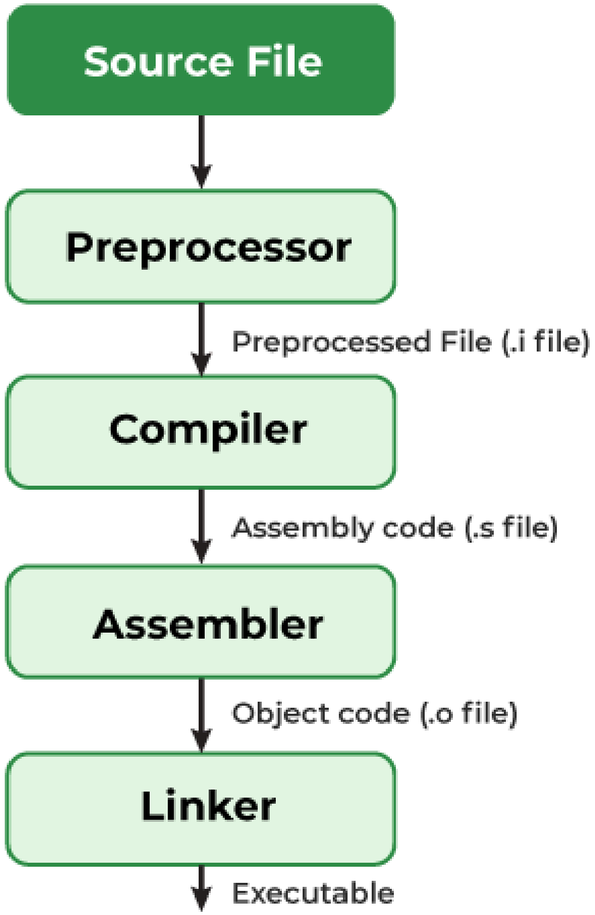
Pull-down resistor

- Originally, Voltage is set to logic level Low(0).
- If switch is pushed, then It set to logic level High(1).

Why do we set pull-up or pull-down resistor?

- When floating occurs, These resistor prevent error situation.

Theory



A compiler converts a C program into an executable file.

These procedure is called Compile.

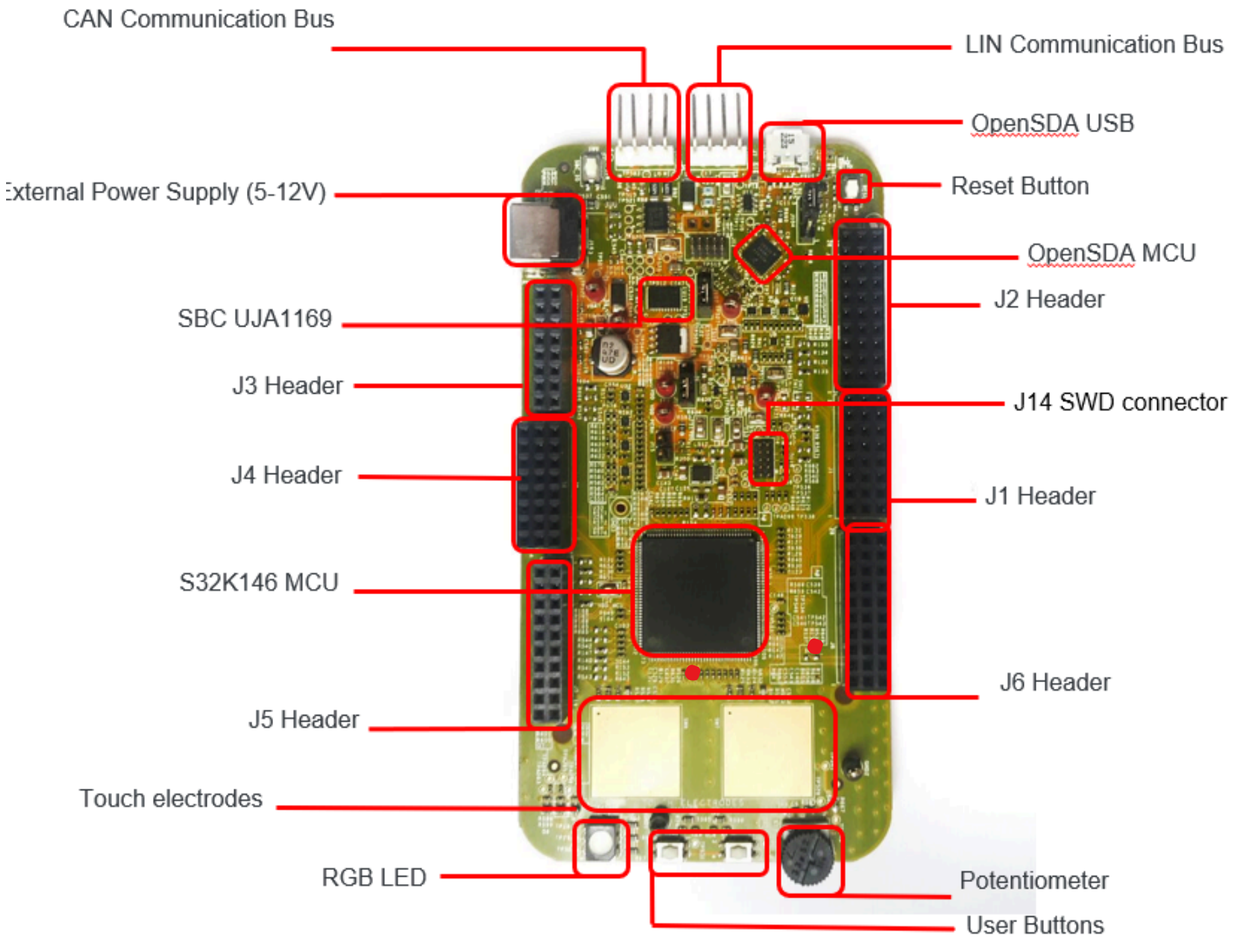
Removal of Comments, Expansion of Macros Expansion of the included files, Conditional compilation
Change into assembly-level instructions (ADD, SUB, MOV)
Change into machine level instructions that is computer-readable (0 or 1)
All the linking of function calls with their definitions

Theory

S32K146 : A microcontroller made by NXP.

Configuration

- core : based on the Arm® Cortex®-M series core
- memory
- Timer
- I/O
- 3 SPI
- 3 CAN
- 1 I2C
- 3 UART
- 2 ADC
- 1 DAC

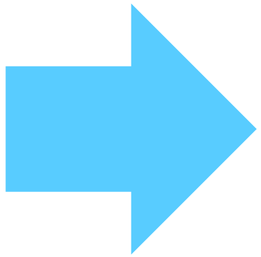


LED Blink
with Non Autosar

LED Blink with Non Autosar

Hello.c analysis

```
#include "device_registers.h"

#define PTD0 0
#define PTC12 12

void WDOG_disable (void)
{
 WDOG->CNT=0xD928C520;
 WDOG->TOVAL=0x0000FFFF;
 WDOG->CS = 0x00002100;
}

int main(void)
{
  int counter = 0;

  WDOG_disable();
  PCC-> PCCn[PCC_PORTC_INDEX] = PCC_PCCn_CGC_MASK;
  PCC-> PCCn[PCC_PORTD_INDEX] = PCC_PCCn_CGC_MASK;

  PTC->PDDR &= ~(1<<PTC12);
  PORTC->PCR[12] = PORT_PCR_MUX(1);

  PTD->PDDR |= 1<<PTD0;
  PORTD->PCR[0] = PORT_PCR_MUX(1);

  for(;;)
  {
    if (PTC->PDIR & (1<<PTC12)) {
        PTD-> PCOR |= 1<<PTD0;
     }
     else {
        PTD-> PSOR |= 1<<PTD0;
        }
     counter++;
  }
}
```



< Result >

Blue LED turns on.

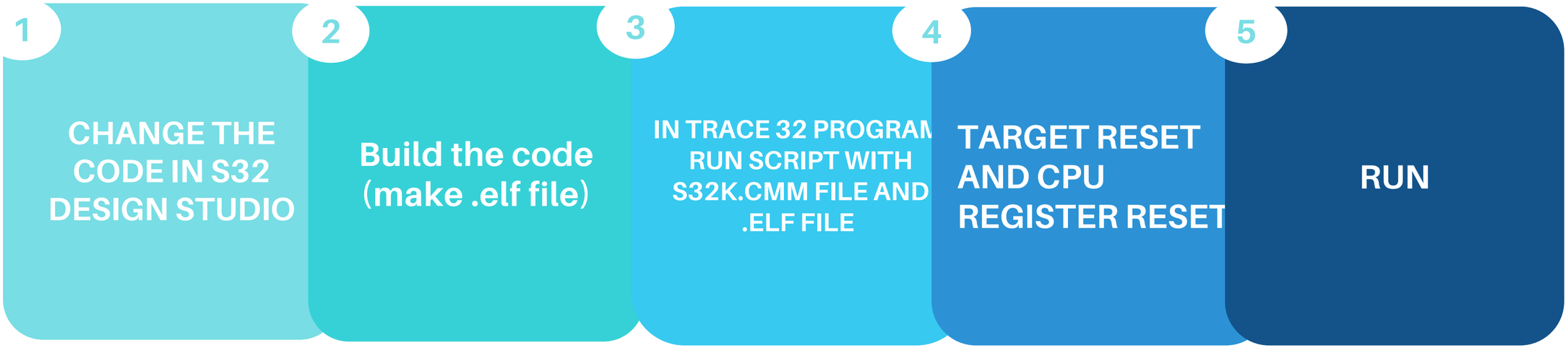
If switch is pushed, then LED turns off.

< Code Change >

component	Before	After
LED	Blue	RED
Switch	SW2	SW3

LED Blink with Non Autosar

Non-Autosar process



LED Blink with Non Autosar

Step 1. Change the code in S32 design studio

- **change define**

Base Code

```
#define PTD0 0   // BLUE LED
#define PTC12 12 // SW2
```

Changed Code

```
#define PTD16 16 // RED LED
#define PTC13 13 // SW3
```

LED Blink with Non Autosar

Step 1. Change the code in S32 design studio

- **change components (SW2 -> SW3, BLUE LED -> RED LED)**

Base Code

```
PTC->PDDR &= ~(1<<PTC12);
PORTC->PCR[12] = PORT_PCR_MUX(1) |
PORT_PCR_PFE_MASK

PTD->PDDR |= 1<<PTD0;
PORTD->PCR[0] = PORT_PCR_MUX(1);
```

Changed Code

```
PTC->PDDR &= ~(1<<PTC13);
/* Port C13:Data Direction is input */
PORTC->PCR[13]  =  PORT_PCR_MUX(1)  |
PORT_PCR_PFE_MASK
/* Port C 13: MUX = GPIO, input filter enable */

PTD->PDDR |= 1<<PTD16;
/* Port D16: Data Direction = output */
PORTD->PCR[16] = PORT_PCR_MUX(1);
/* Port D16: MUX = GPIO */
```

LED Blink with Non Autosar

Step 1. Change the code in S32 design studio

- **change “if statement”**

Base Code

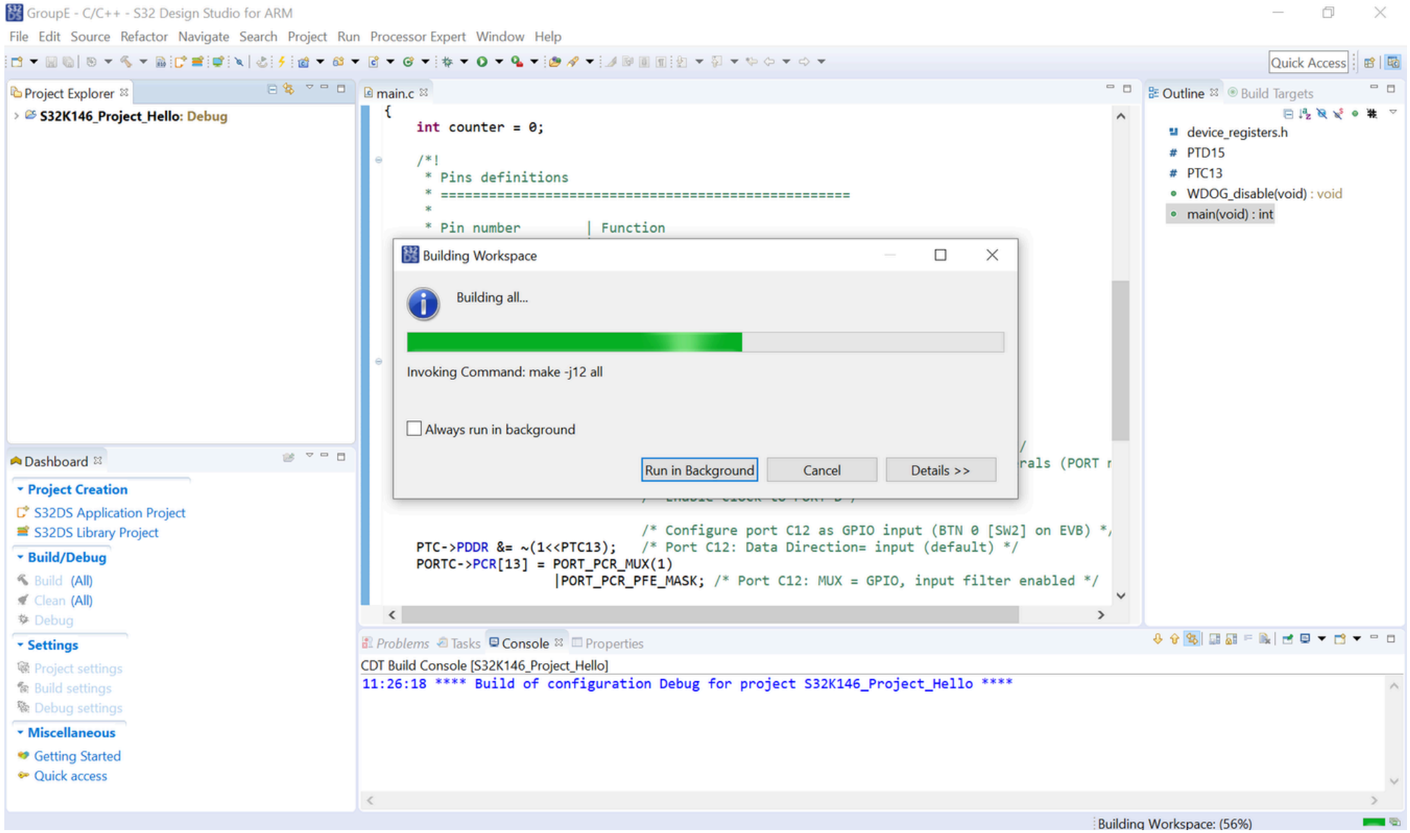
```
if (PTC->PDIR & (1<<PTC12)) {
    PTD-> PCOR |= 1<<PTD0;
}
else {
   PTD-> PSOR |= 1<<PTD0;
}
```

Changed Code

```
if (PTC->PDIR & (1<<PTC13)) {
/*push switch*/
    PTD-> PCOR |= 1<<PTD16;
/* clear, LED turn off */
 }
 else {
     PTD-> PSOR |= 1<<PTD16;
/* set register 1, LED turn on */
 }
```

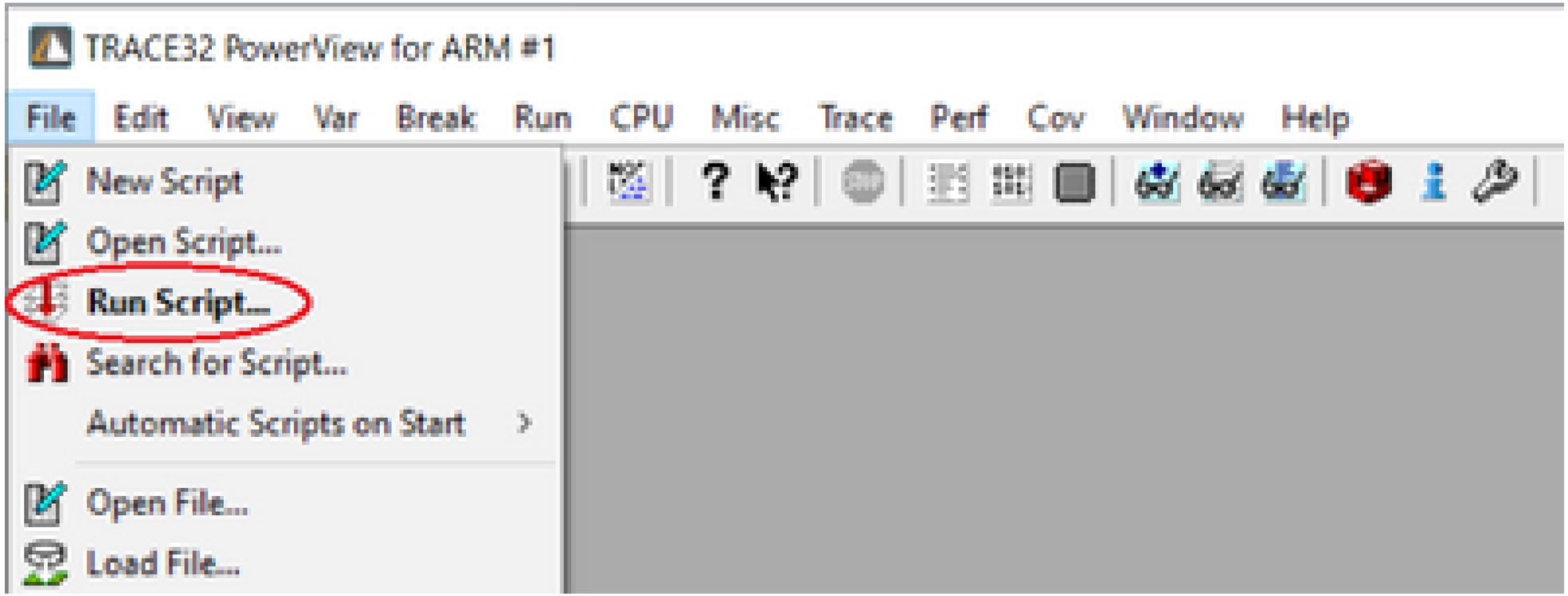
LED Blink with Non Autosar

Step 2. Build the code



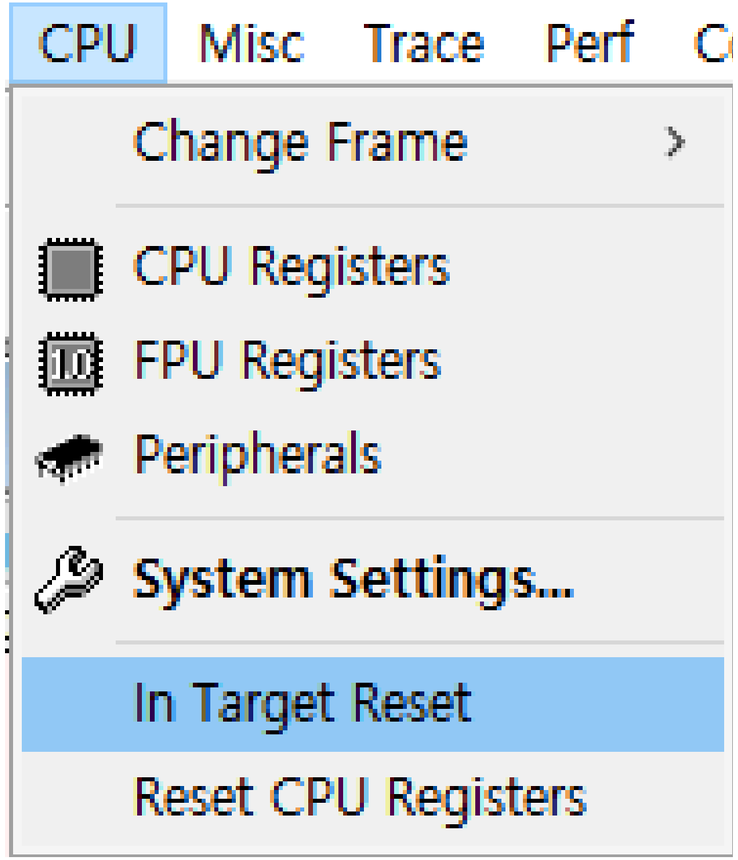
LED Blink with Non Autosar

Step 3. Run script with s32k.cmm file and .elf file in Trace 32 Program



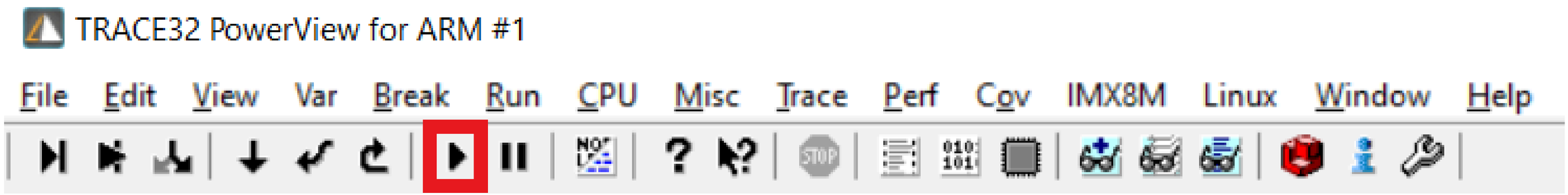
LED Blink with Non Autosar

Step 4. Click to target reset and cpu register reset button

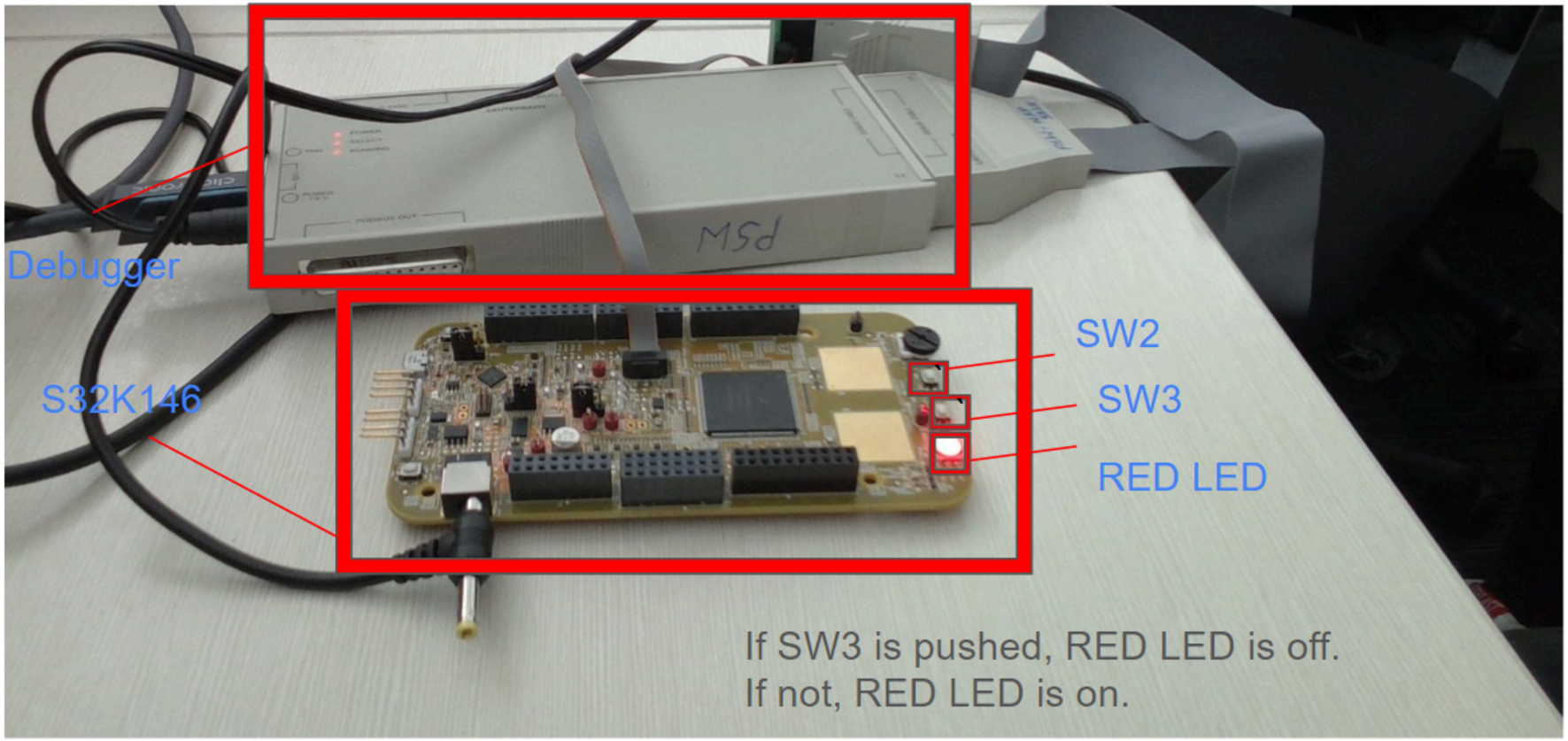


LED Blink with Non Autosar

Step 5. RUN



LED Blink with Non Autosar



Result

LED Blink
with Autosar

LED Blink with Autosar

Autosar : AUtomotive Open System ARchitecture

Autosar is a standardized software architecture for automotive system.

Autosar has benefits:

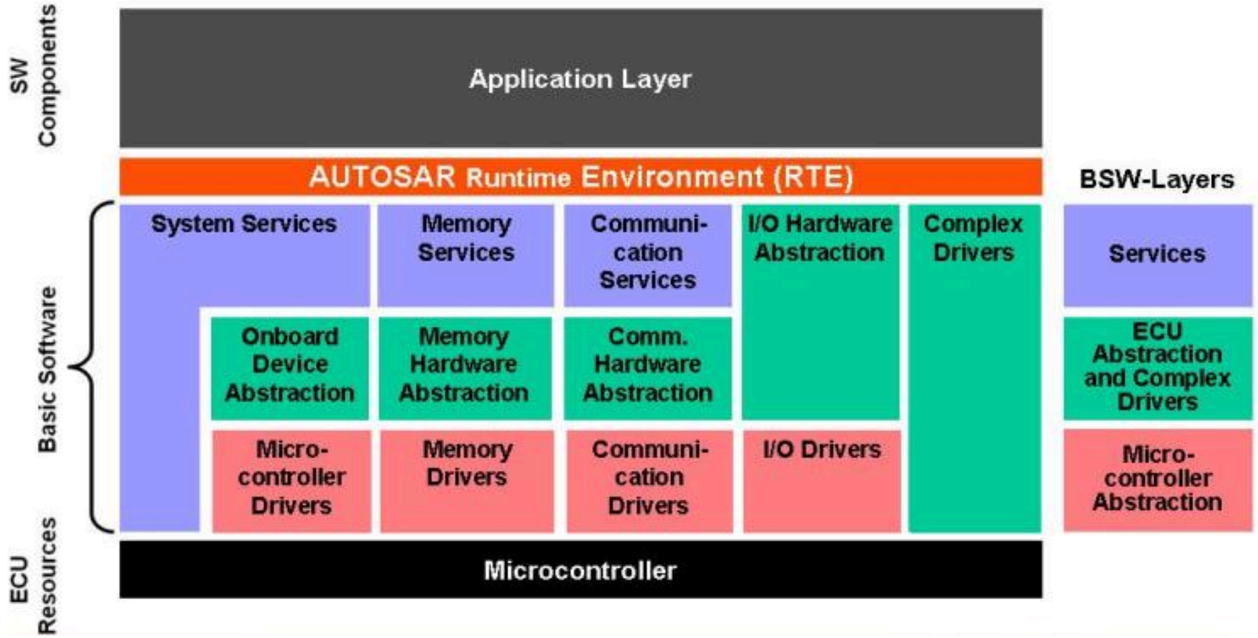
1. reusability
2. loosing development cost
3. time saving

Same architecture enable use of ECU in different vehicles

LED Blink with Autosar

BSW

- **MCAL** : Abstracts the hardware details of the microcontroller, allowing the upper layers of software to operate independently of the hardware.
- **ECU Abstraction** : Delivers higher software layers ECU specific drivers.
- **Services** : Consists of OS, network services, diagnostics, communication services and etc.

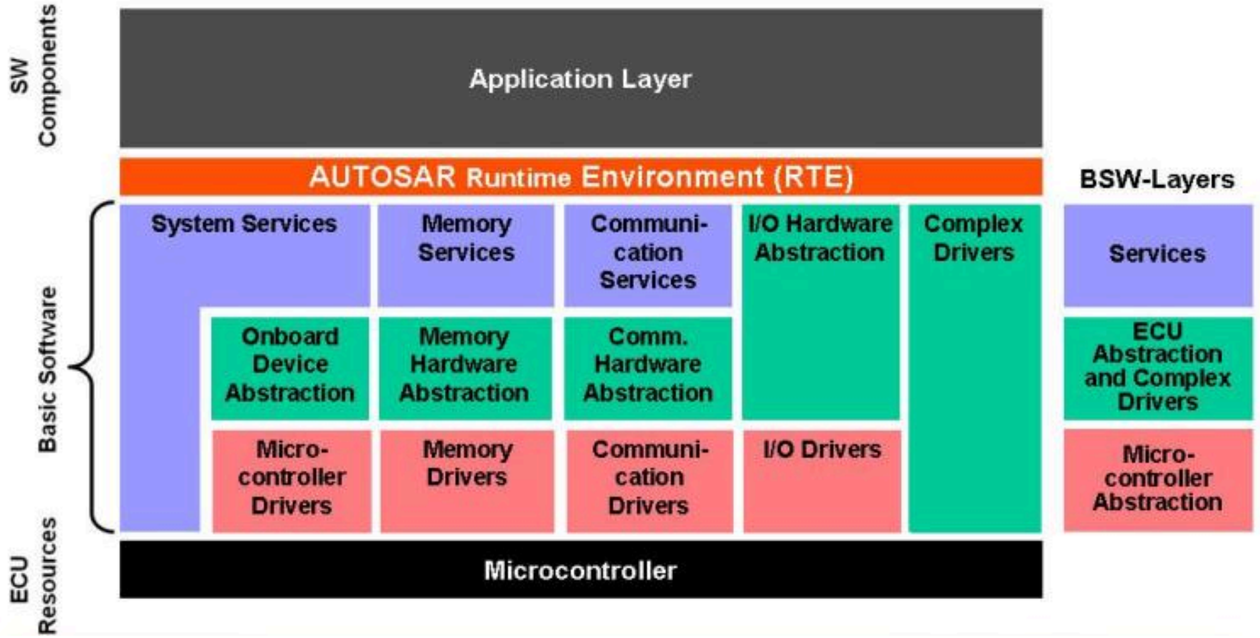


<Autosar architecture>

LED Blink with Autosar

RTE : Provides the interface between application software and the BSW, allowing applications to be developed and deployed independently.

Application Layer : Consists of software modules that implement specific functions of the vehicles.



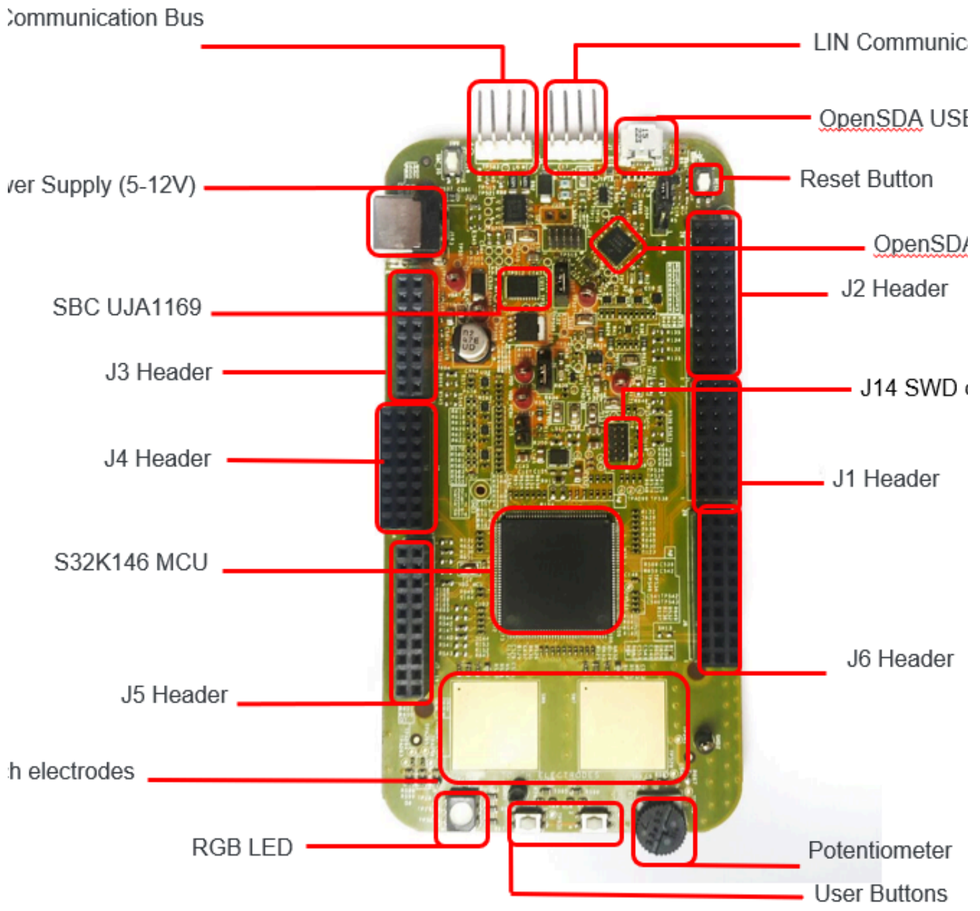
<Autosar architecture>

LED Blink with Autosar

PORT : Group of pins

There are 5 Ports : A, B, C, D, E

PORT A : 0 ~ 31
PORT B : 32 ~ 63
PORT C : 64 ~ 95
PORT D : 96 ~ 127
PORT E : 128 ~ 159



LED Blink with Autosar

PORT : Group of pins

PTC12 - SW2 (64 + 12 = 76)
PTC13 - SW3 (64 + 13 = 77)

PTD 0 - Blue LED (96)
PTD 15 - Red LED (96 + 15 = 111)
PTD 16 - Green LED (96 + 16 = 112)

LED Blink with Autosar

DIO : Digital Input Output

digital input : Read digital signal from external device or sensor.
(example : switch, sensor)

digital output : Drive digital signal from external device or actuator
(example : LED)

DIO is essential capability of handling digital signal, both for input and output.

LED Blink with Autosar

EB tresos

EB tresos is an AUTOSAR tool.

We choose the configuration, and then this tool generates code automatically (configuration of BSW module).

This tool creates files (.c, .h, etc.) under the output folder.

LED Blink with Autosar

SDLC : Software Development Life Cycle

SDLC are various process or methods that are chosen for project development depending on the objectives and goals of the project.

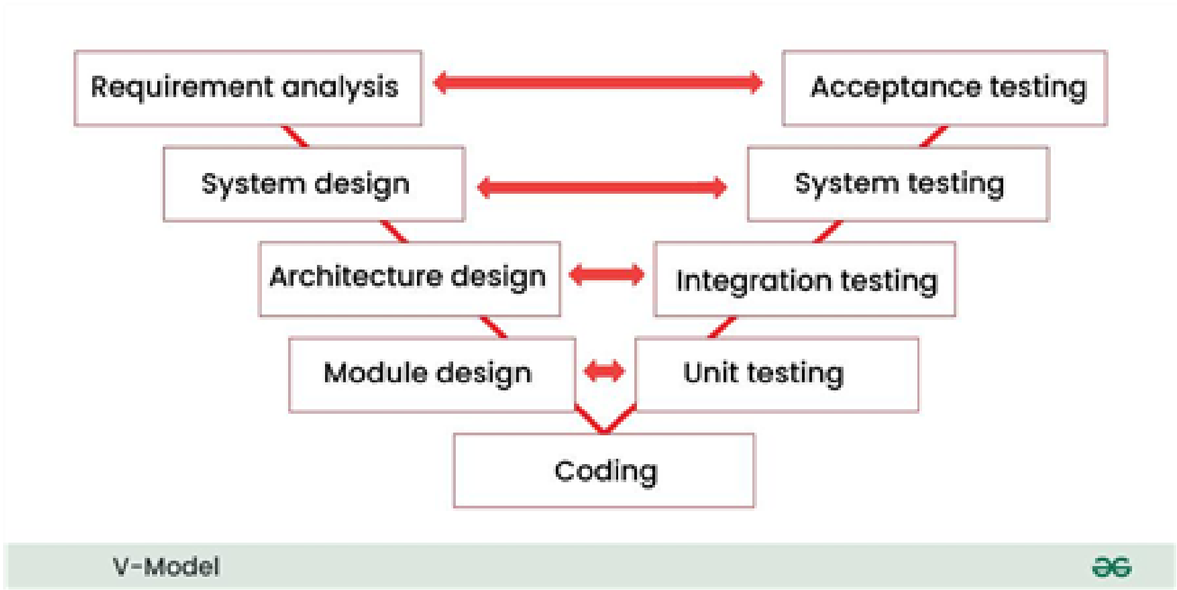
List of SDLC:

- Waterfall Model
- Spiral Model
- V - model (We used this model)
- Prototype model
- Agile model

LED Blink with Autosar

V model

- Verification Phase will be on one side, Validation Phase will be on the other side that is both the activities run simultaneously.
- Both of them are connected to each other in V-Shape through coding phase.
- Small and medium scale developments can be easily completed using it.



LED Blink with Autosar

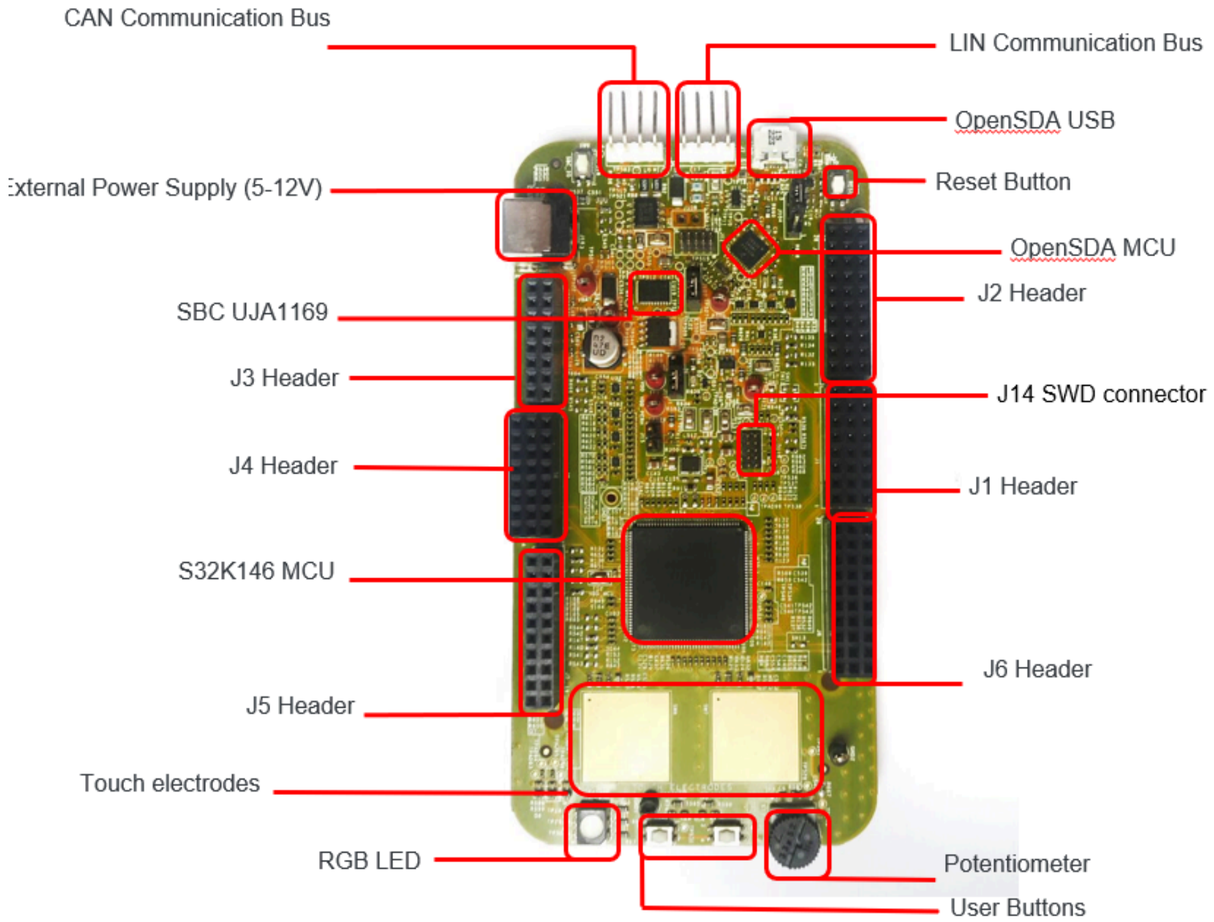
LED Blink with Autosar

1. Requirement

1) When switch is pushed, LED is On.
2) When switch is released, LED is Off.
3) Switch is 3. (PTC 13)
4) LED is red. (PTD 15)
5) Red LED is input.
6) Switch is output.

LED Blink with Autosar

1. Requirement



S32K146 - 144pins LQFP

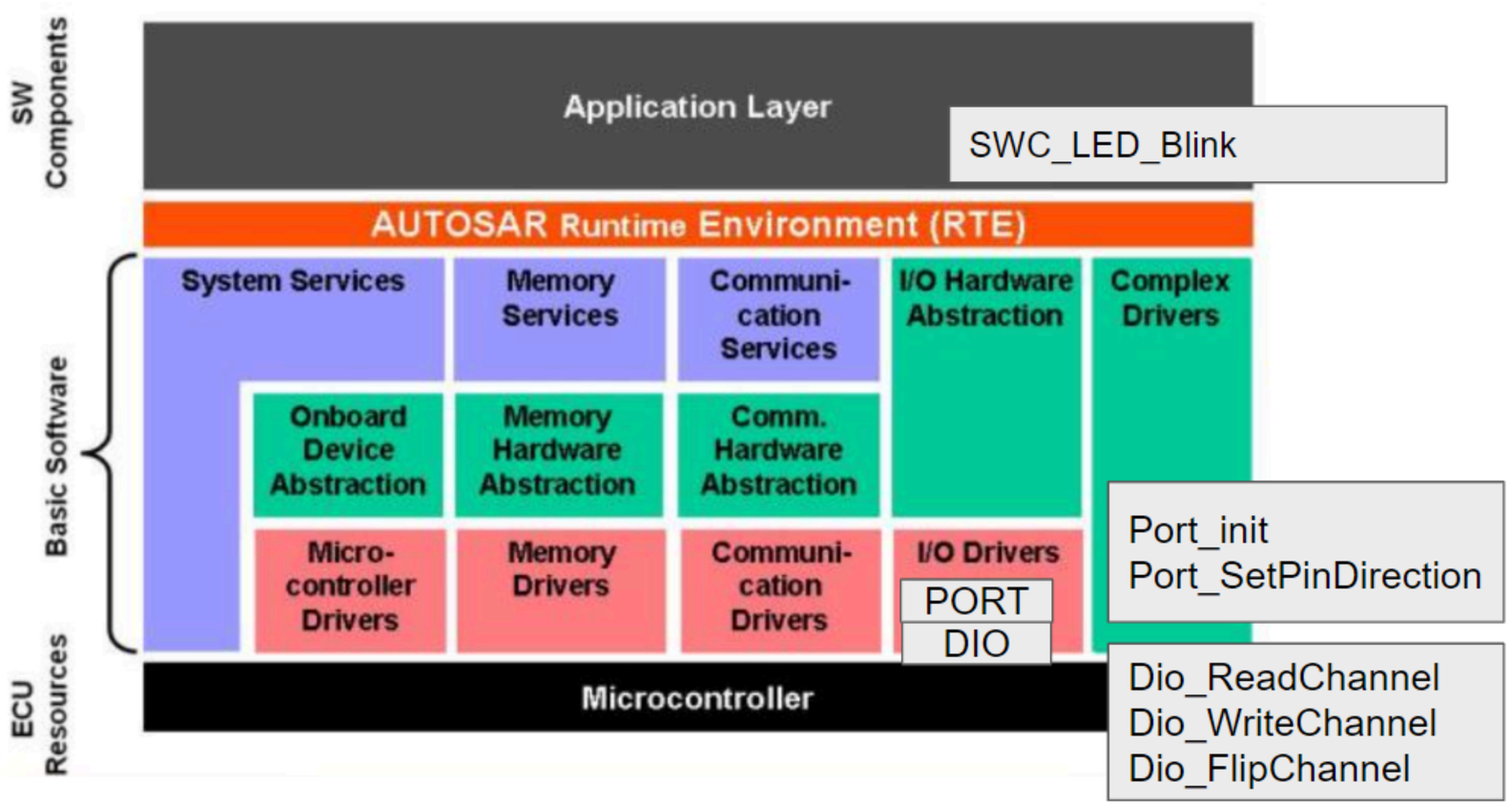
LED Blink with Autosar

1. Requirement

Schmetic

LED Blink with Autosar

1. Requirement



LED Blink with Autosar

2. Designing

Software functions - Port

8.3.1 Port_Init

[SWS_Port_00140] Definition of API function Port_Init ⌈

Service Name	Port_Init	

Syntax	`void Port_Init (`
	`  const Port_ConfigType* ConfigPtr`
	`)`	

Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the Port Driver module.	
Available via	Port.h	

8.3.2 Port_SetPinDirection

[SWS_Port_00141] Definition of API function Port_SetPinDirection ⌈

Service Name	Port_SetPinDirection	

Syntax	`void Port_SetPinDirection (`
	`  Port_PinType Pin,`
	`  Port_PinDirectionType Direction`
	`)`	

Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Pin	Port Pin ID number
	Direction	Port Pin Direction
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the port pin direction	
Available via	Port.h	

LED Blink with Autosar

2. Designing

Software functions - Dio

8.3.2 Dio_WriteChannel

[SWS_Dio_00134] Definition of API function Dio_WriteChannel ⌈

Service Name	Dio_WriteChannel	

Syntax	`void Dio_WriteChannel (`
	`  Dio_ChannelType ChannelId,`
	`  Dio_LevelType Level`
	`)`	

Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
	Level	Value to be written
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to set a level of a channel.	
Available via	Dio.h	

8.3.1 Dio_ReadChannel

[SWS_Dio_00133] Definition of API function Dio_ReadChannel ⌈

Service Name	Dio_ReadChannel	

Syntax	`Dio_LevelType Dio_ReadChannel (`
	`  Dio_ChannelType ChannelId`
	`)`	

Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Dio_LevelType	`STD_HIGH` The physical level of the corresponding Pin is `STD_HIGH` `STD_LOW` The physical level of the corresponding Pin is `STD_LOW`
Description	Returns the value of the specified DIO channel.	
Available via	Dio.h	

LED Blink with Autosar

2. Designing

Software functions - Dio

9.3.8 Dio_FlipChannel

[SWS_Dio_00190][

Service Name	Dio_FlipChannel	
Syntax	`Dio_LevelType Dio_FlipChannel ( Dio_ChannelType ChannelId )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Dio_LevelType	STD_HIGH: The physical level of the corresponding Pin is STD_HIGH. STD_LOW: The physical level of the corresponding Pin is STD_LOW.
Description	Service to flip (change from 1 to 0 or from 0 to 1) the level of a channel and return the level of the channel after flip.	
Available via	Dio.h	

LED Blink with Autosar

2. Designing

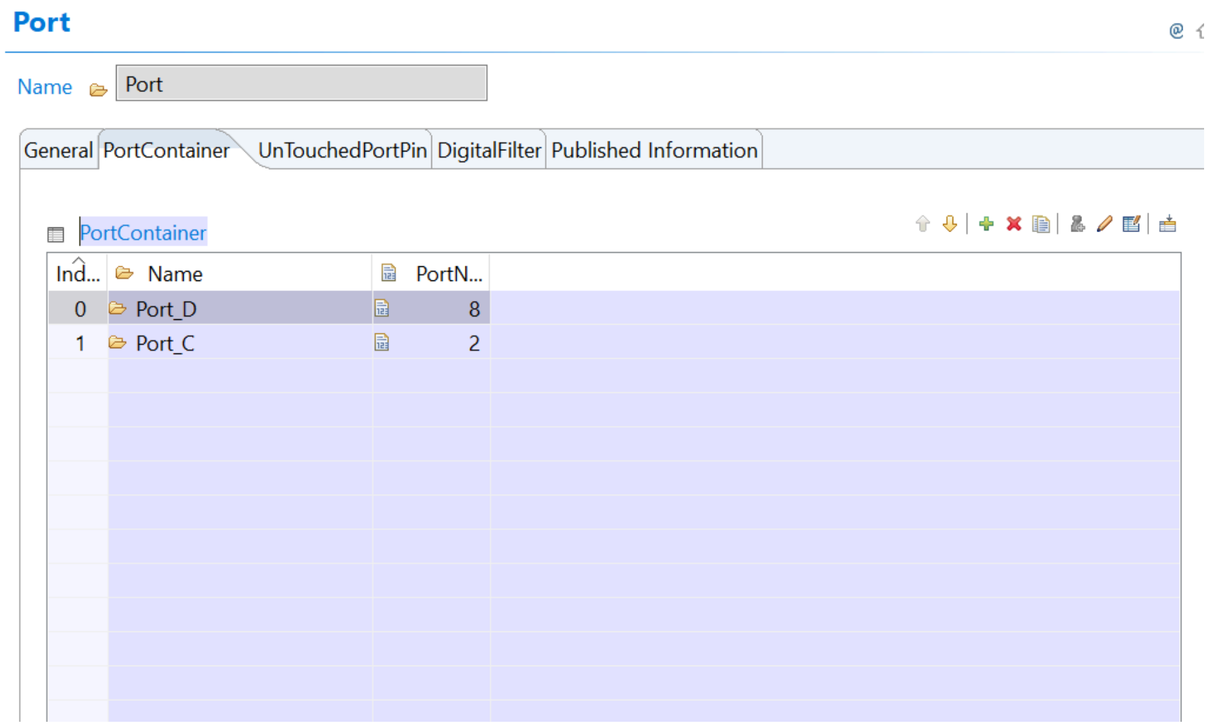
Software functions - Interface

I/O	PORT number	component
input	PTC 13	SW3
output	PTD 15	RED Led

LED Blink with Autosar

2. Designing

EB tresos configuration - Port Setting (Port Container)



Port D(8) :

- JTAG_TDI
- JTAG_TDO
- JTAG_TCK
- JTAG_TMS
- Reset_b
- Red_LED
- Blue_LED
- Green_LED

Port C(2) :

- SW2
- SW3

LED Blink with Autosar

JTAG (Joint Test Action)

- It is a method of transmitting output data or receiving input data in a serial communication method for digital input/output of a specific node in a digital circuit.
- One example is debugging equipment used in the development of the embedded system

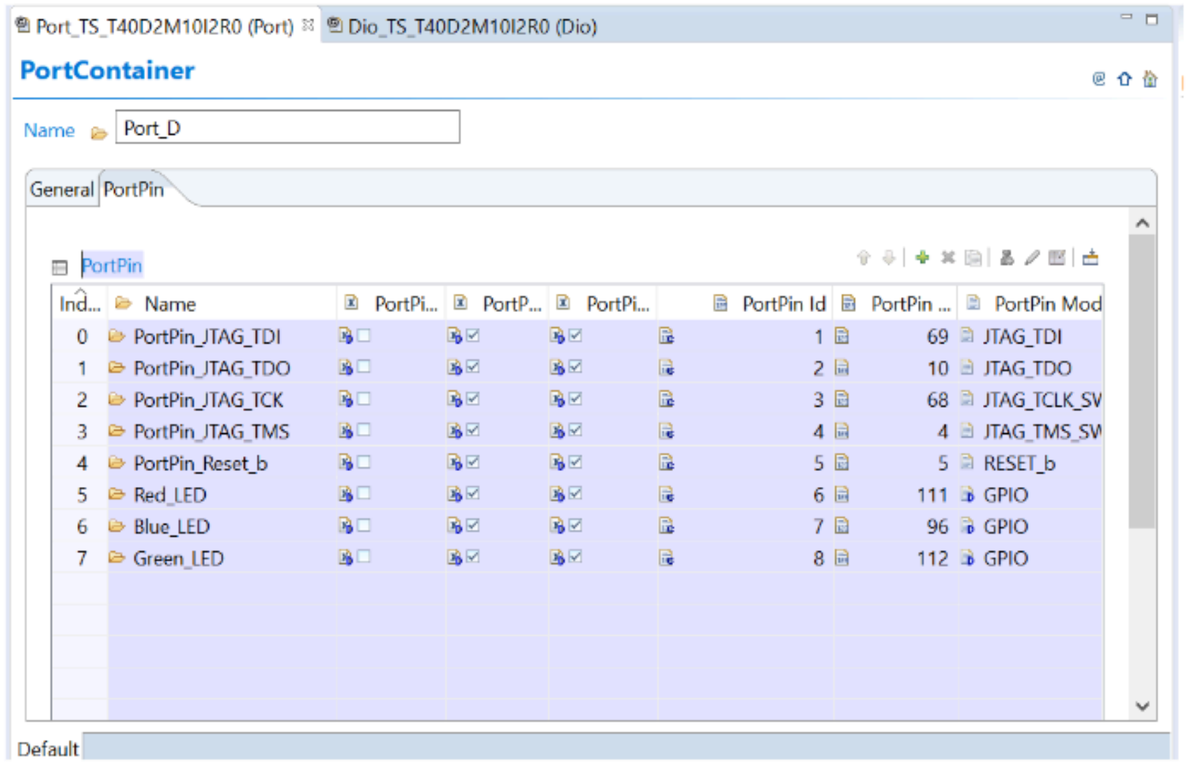
JTAG have 4 pins

- TDI : input pin that receives serial test instructions and data.
- TDO : output pin that transits serial output for test instructions and data.
- TCK : input pin used to synchronize the test logic and control register access.
- TMS : input pin used to sequence the IEEE 1149.1-2001 test control state machine.

LED Blink with Autosar

2. Designing

EB tresos configuration - Port Setting (Port D)



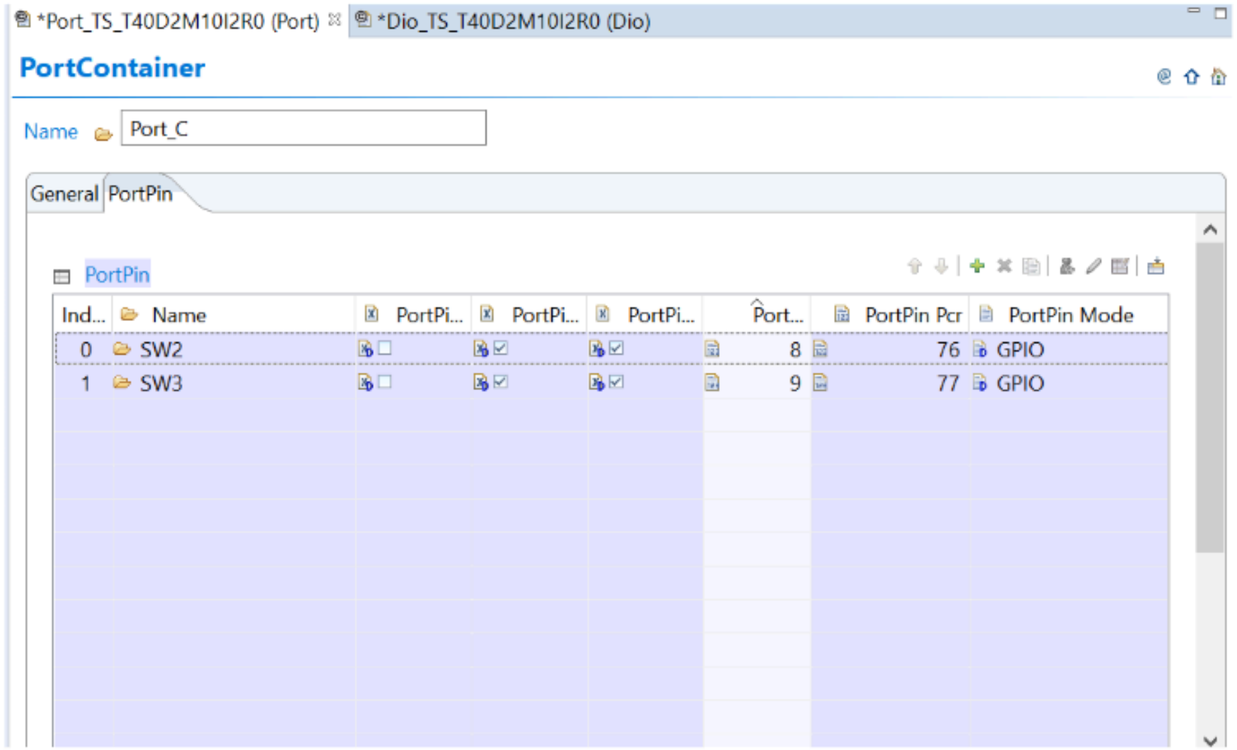
JTAG : for debugging

PTD 15 (Red_LED) : 96 + 15 = 111
PTD 0 (Blue_LED) : 96
PTD 16 (Green_LED) : 96 + 16 = 112

LED Blink with Autosar

2. Designing

EB tresos configuration - Port Setting (Port C))



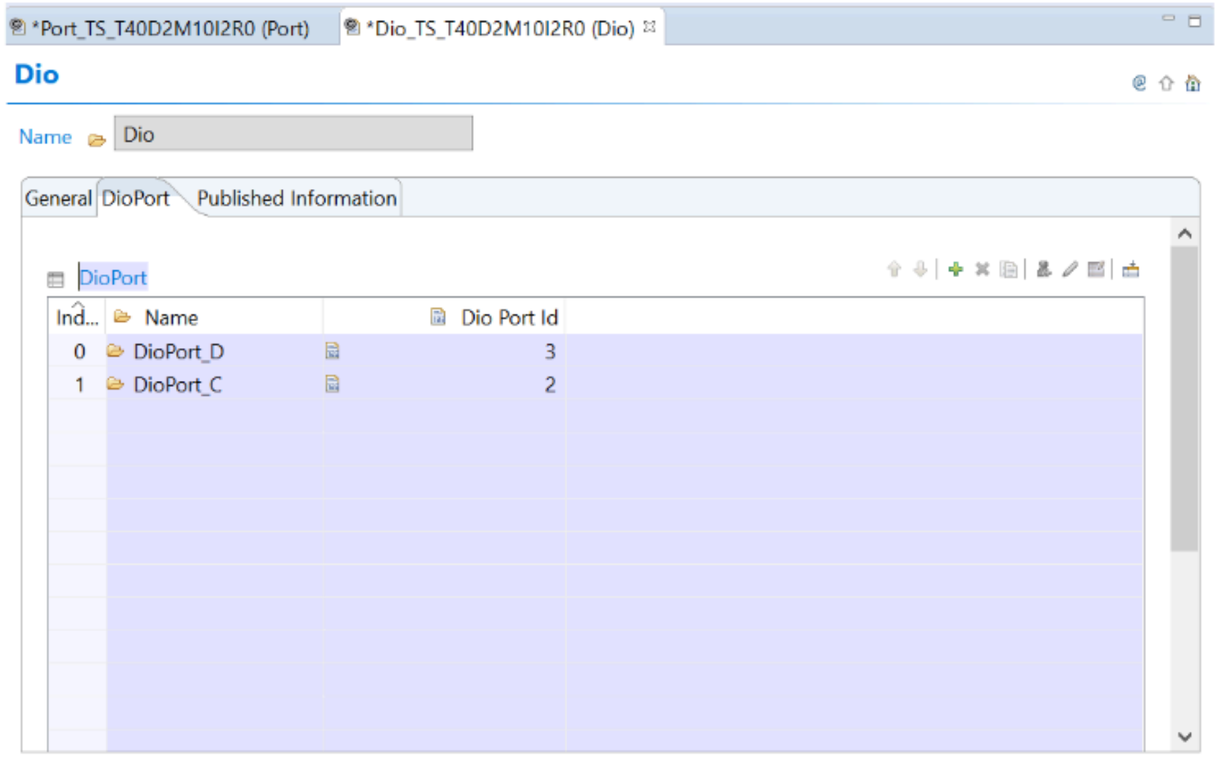
PTC 12(SW2) : 64 + 12 = 76

PTC 13(SW3) : 64 + 13 = 77

LED Blink with Autosar

2. Designing

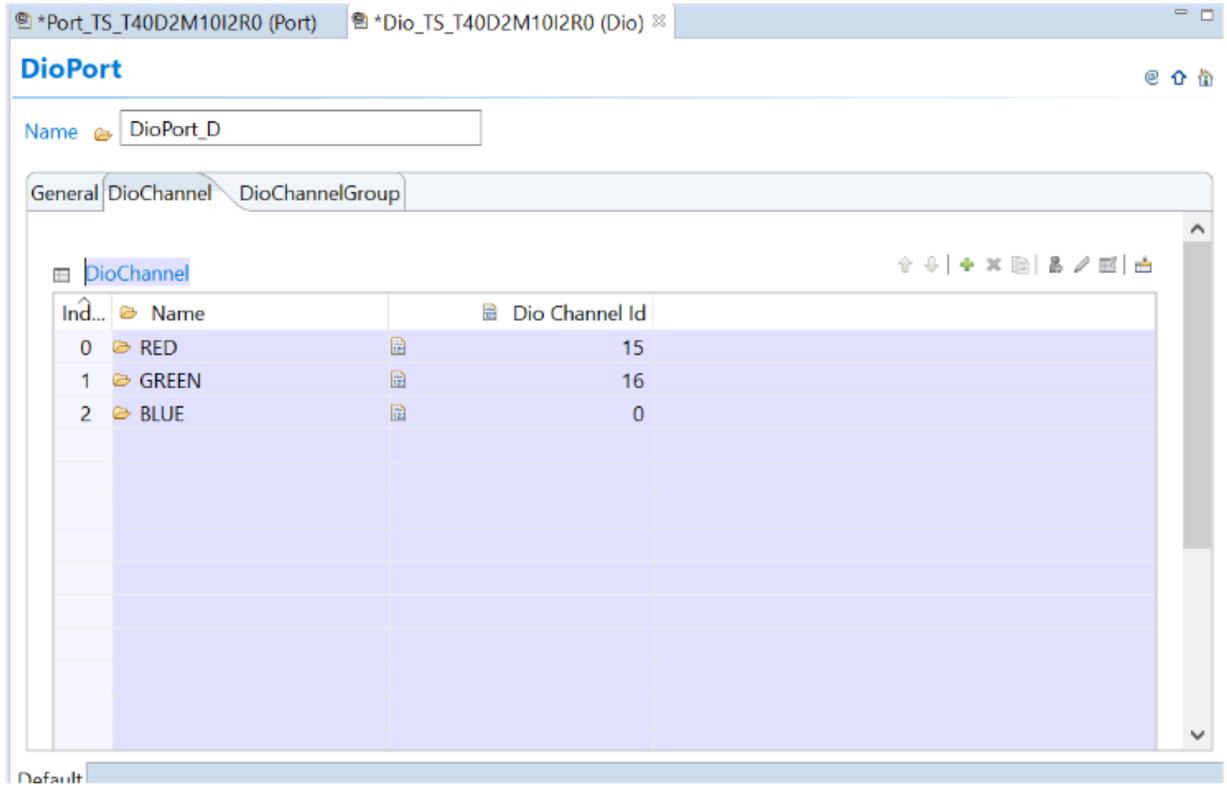
EB tresos configuration - Dio Setting (DioPort)



LED Blink with Autosar

2. Designing

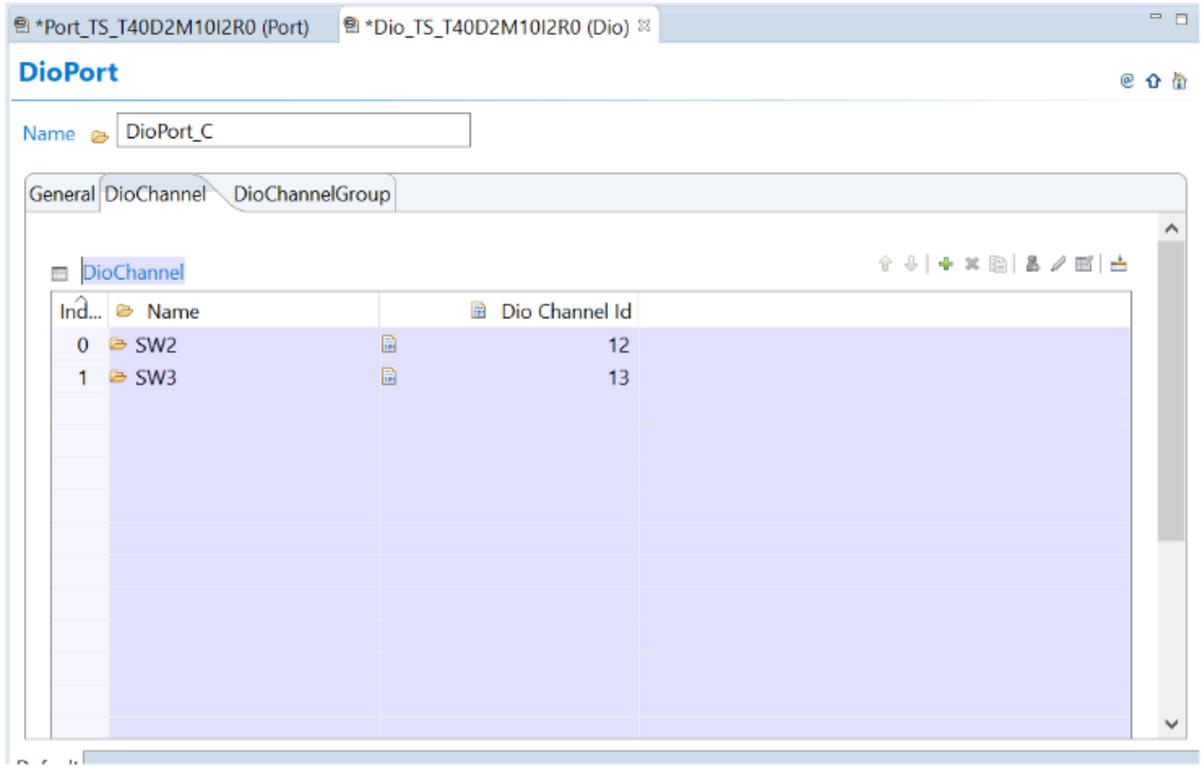
EB tresos configuration - Dio Setting (DioPort D)



LED Blink with Autosar

2. Designing

EB tresos configuration - Dio Setting (DioPort C)



LED Blink with Autosar

3. Coding

Step

1) Setting PORT and DIO driver in EB tresos
   PTD15 (RED LED) is calculated as 96 + 15 = 111.
   PTC13 (SW3) is calculated as 64 + 13 = 77.

2) Export .arxml file

3) Make SWC (behavior, runnable, event, etc.) in Artop.

4) Import .arxml file (modified)

5) Verify project and generate project.

6) Open VScode program and change SWC runnable (functions).

7) Go to util directory and Open launch file.
   type “make clean” : remove all except generate folder
   type “make -j” : generate depend files and executable (.elf file)

8) Run in TRACE32

LED Blink with Autosar

SWC : Software component

SWCs are reusable building blocks of AUTOSAR software that encapsulate algorithms and communicate with their environment through ports

Parts : Runnable, port, interface, connector

Runnable : function to implement the functionality of the Software component

Port : point to connect a SW-C with other SW-C or services provided by/via the RTE
(R-Port, P-Port, PR-Port)

Interface : specification of the communication via a Port
(Sender-Receiver interface, server-client interface, etc.)

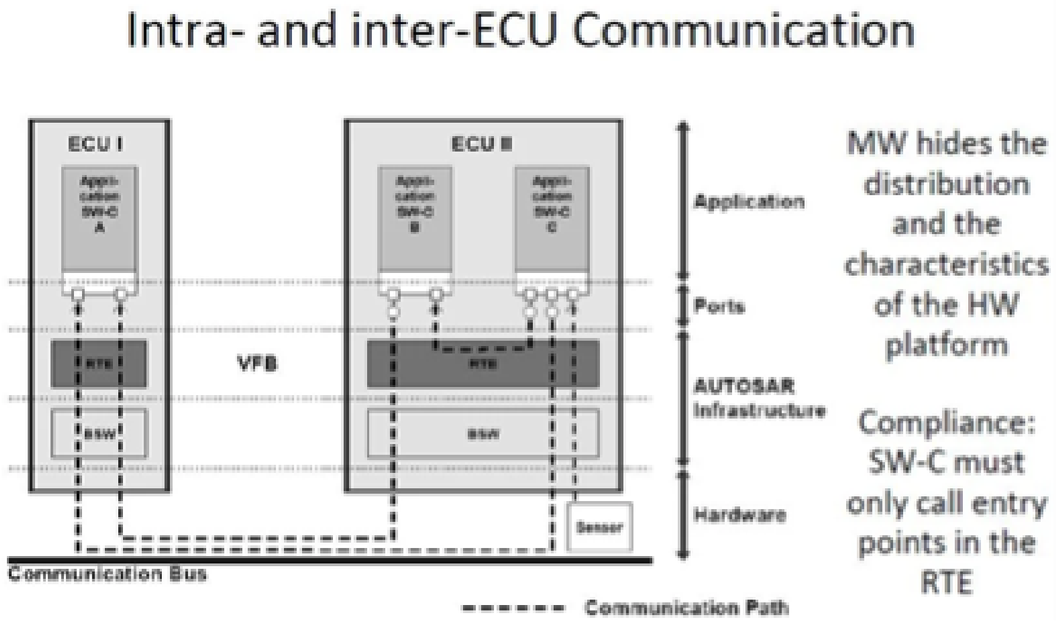
Connector : connect SW-C Ports with other SW-C Ports or service Ports from BSW

Software components are located in Application layer.

LED Blink with Autosar

Software component communication by RTE
- intra-ECU : inside one ECU
- inter-ECU : between different ECUs

Steps for Inter-ECU communication between ECU1 and ECU2
: ECU1 -> RTE -> BSW -> Communication Bus (CAN, Flexray, etc) -> BSW -> RTE -> ECU2



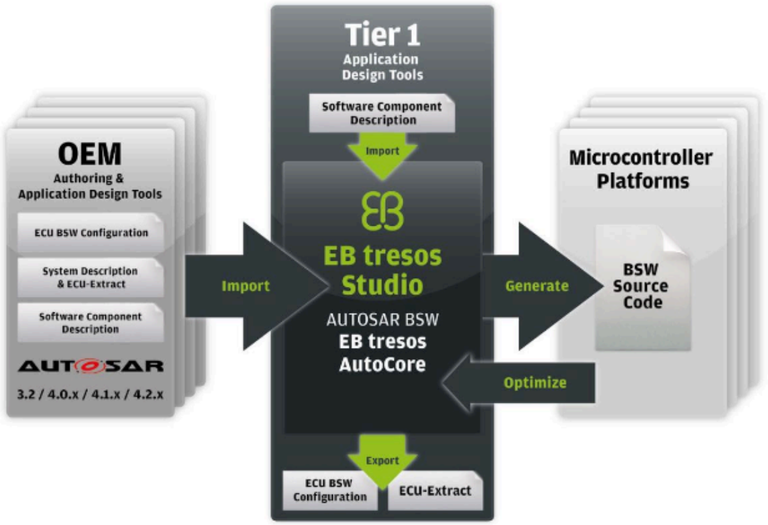
LED Blink with Autosar

OEM: Enterprise that give a product request.

OEM make System Description and ECU Extract, then give these information to tier 1 supplier.

System Description : overall system architecture and requirements.
ECU Extract : detailed specifications for each ECU(Steering, Break, Suspension, etc.) within the system.

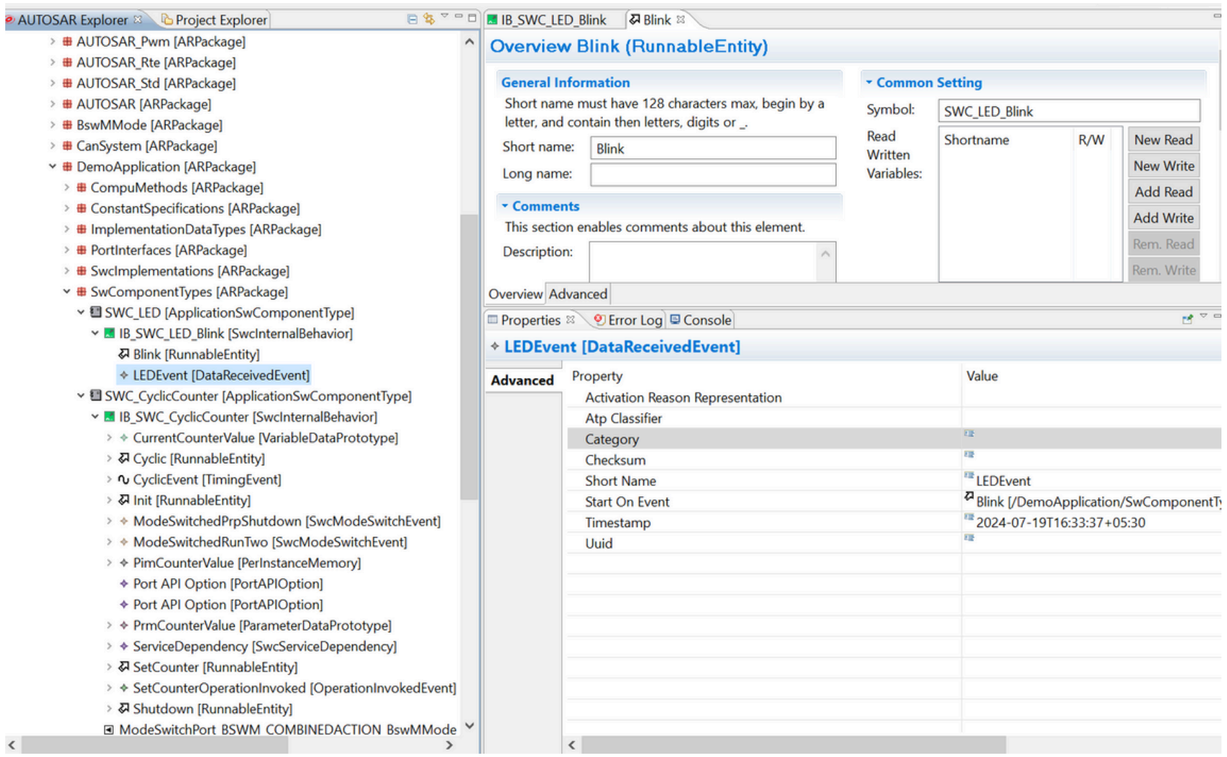
Tier 1 supplier (like HL mando) make ECU Configuration and Software Component according to the specifications provided by the OEM.



LED Blink with Autosar

3. Coding

Make SWC (behavior, runnable, event, etc.) in Artop



LED Blink with Autosar

3. Coding

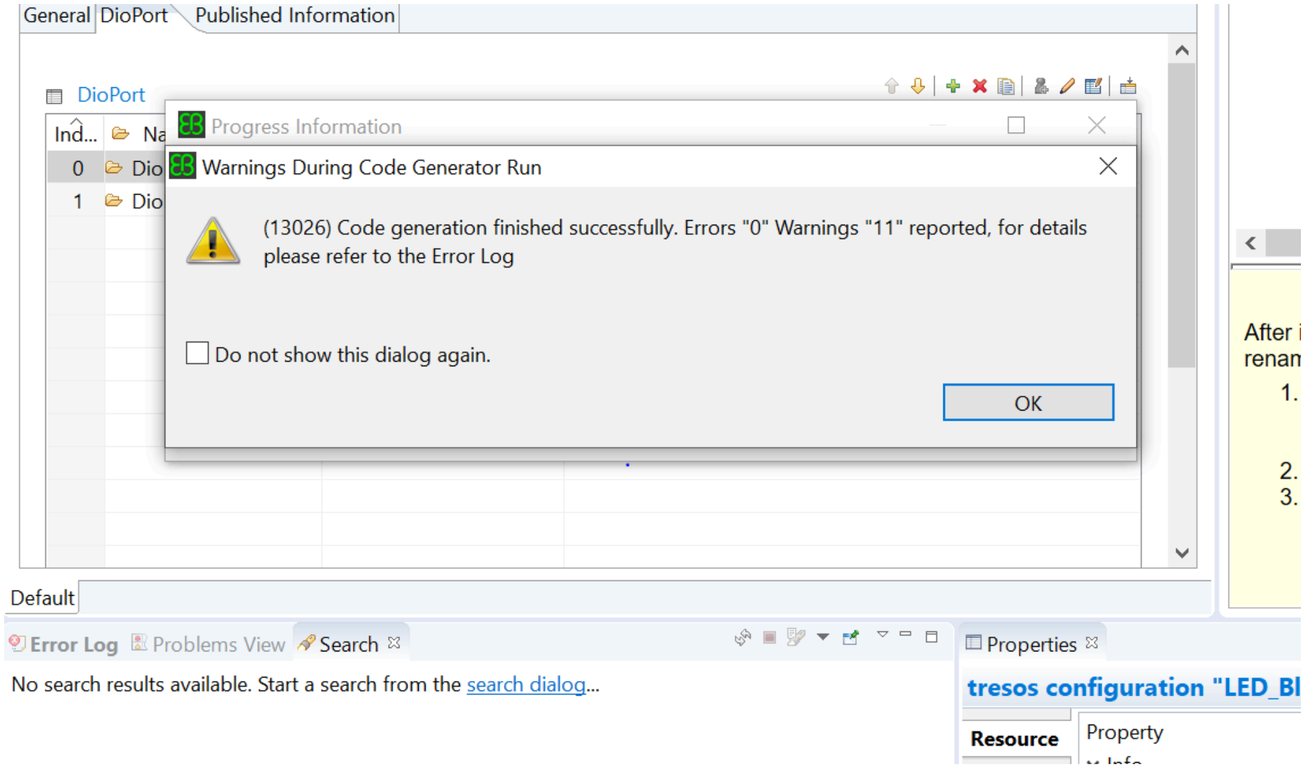
Open VScode program and change SWC runnable (functions)

```
#define SWC_LED_Blink_START_SEC_CODE
#include <SWC_LED_Blink_MemMap.h>


static int Abc;
int count = 0;
FUNC(void, RTE_CODE) SWC_LED_Blink_Blinking (void)
{
  if(count == 0){
    Abc = Dio_FlipChannel(DioConf_DioChannel_Red_LED);
  }
  else if(count == 1){
    Abc = Dio_FlipChannel(DioConf_DioChannel_Red_LED);
  }
  else if(count == 2) {
    Abc = Dio_FlipChannel(DioConf_DioChannel_Green_LED);
  }
  else if(count == 3) {
    Abc = Dio_FlipChannel(DioConf_DioChannel_Green_LED);
  }
  else if(count == 4) {
    Abc = Dio_FlipChannel(DioConf_DioChannel_Blue_LED);
  }
  else{
    Abc = Dio_FlipChannel(DioConf_DioChannel_Blue_LED);
    count = -1;
```

LED Blink with Autosar

3. Coding

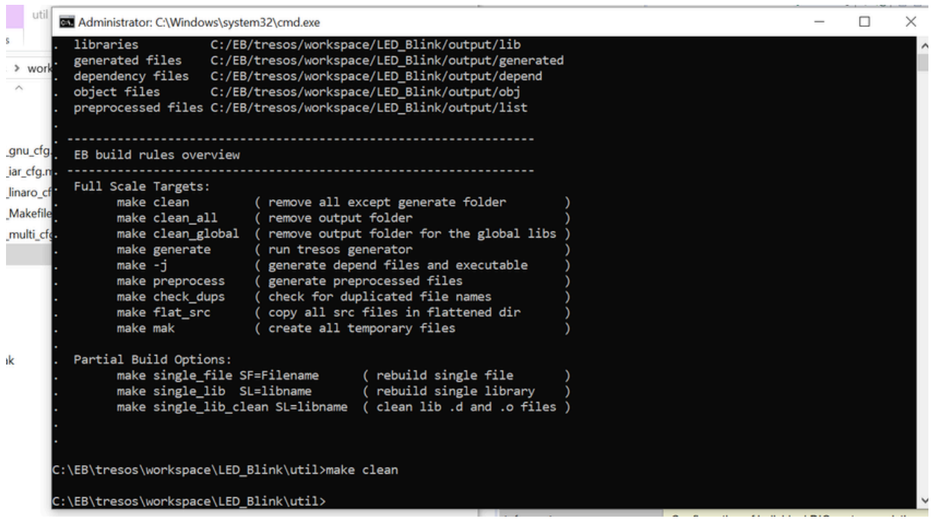
Verify project and generate project.



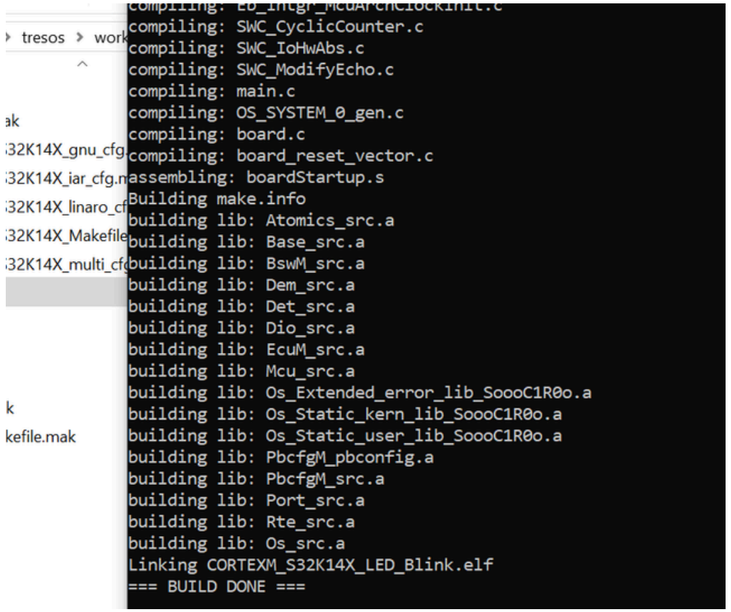
LED Blink with Autosar

3. Coding

Go to util directory and Open launch file.



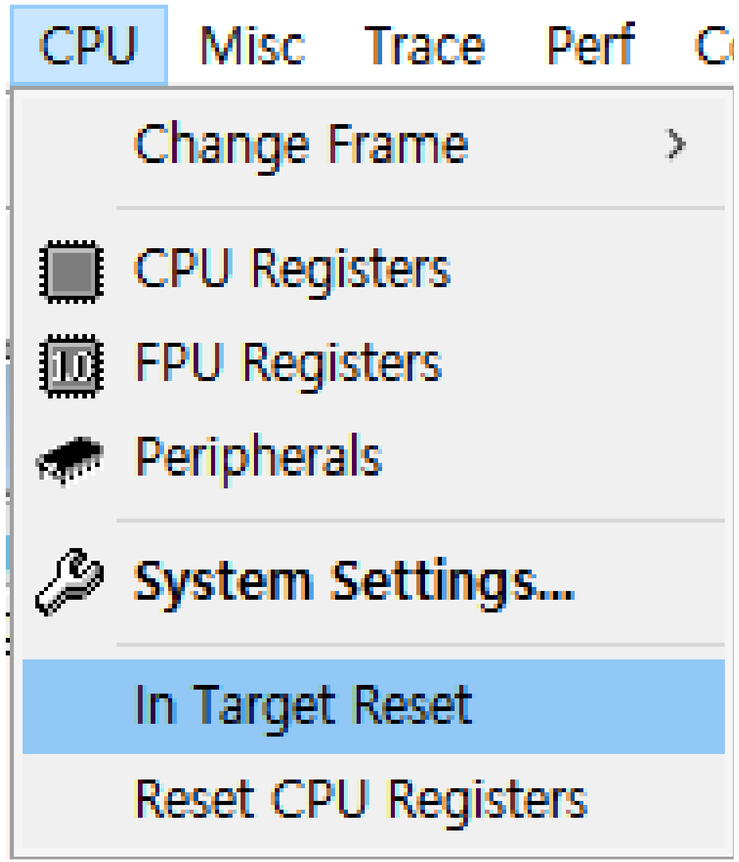
type “make clean”

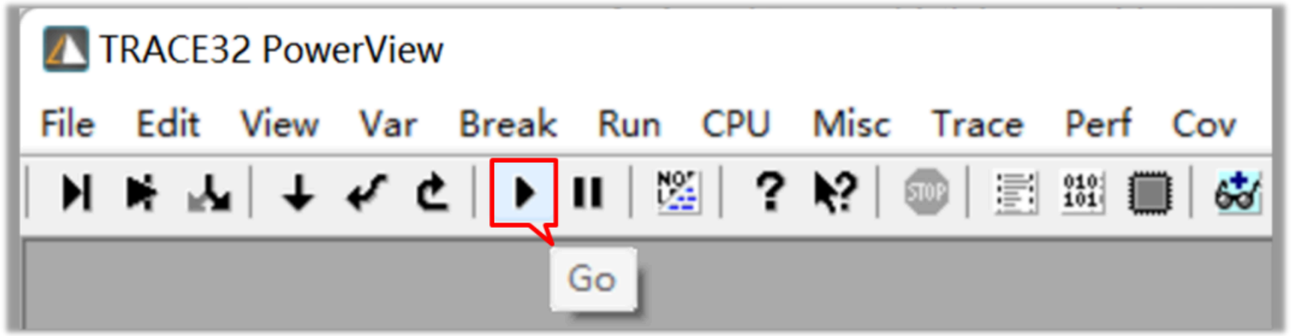


type “make -j”

LED Blink with Autosar

4. Testing





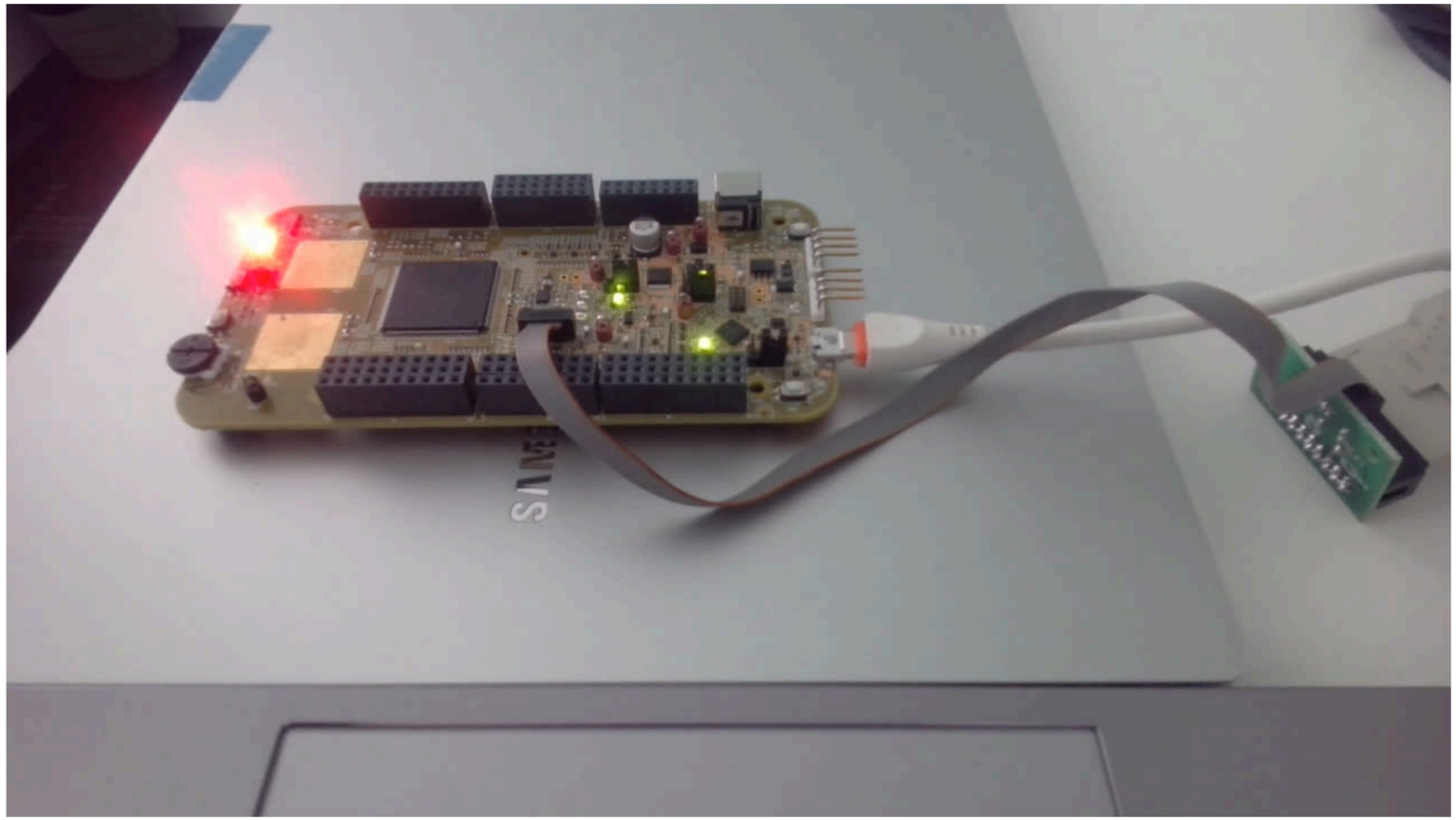
LED Blink with Autosar

4. Testing

Tester name	YeongJin Jeong
Actual Start Date	2024-7-22
Actual End Date	2024-7-22

Test case ID	Precondition	Test Procedure	Expected Result	Actual Result	Test result(Pass/ Fail)
TC_DIO_001	ECU working normal and no hardware faults present	Set 5 second timer	Red LED is on, off, Green LED is on, off, and Blue LED is on, off every 5 second.	Red LED is on, off, Green LED is on, off, and Blue LED is on, off every 5 second	Pass

LED Blink with Autosar



Result

Q&A