Code understand

- PCC-> PCCn[PCC_PORTC_INDEX] = PCC_PCCn_CGC_MASK;
- PCC-> PCCn[PCC_PORTD_INDEX] = PCC_PCCn_CGC_MASK;
- PCC : peripheral clock controller -> exist for accuracy for peripheral
- - We will use PORTC and PORTD. (PCC PORTC Register, PCC PORTD Register)
- CGC : Clock Gate Control -> access to the module's registers.
- 1b - Clock enabled. The current clock selection and divider options are locked and cannot be modified.

Code understand

- PTC->PDDR &= ~(1<<PTC12);
- PDDR : Port Data Direction Register
- -The PDDR configures the individual port pins for input or output.
- PDD - Port Data Direction
- Input(0) : configured as general-purpose input -> set this
- Output(1) : configured as general-purpose output

Code understand

- PORTC->PCR[12] = PORT_PCR_MUX(1) |PORT_PCR_PFE_MASK;
- PCR :Pin Control Register n
- PORT_PCR_MUX(1)
- - Pin Mux Control -> select the pins role
- - 001 : set to GPIO
- PORT_PCR_PFE_MASK
- - PFE : Passive Filter Enable
- - 1 : passive input filter is enabled

Code understand

- PTD->PDDR |= 1<<PTD0;
- PORTD->PCR[0] = PORT_PCR_MUX(1);
- PDD – Port Data Direction
- Input(0) : configured as general-purpose input
- Output(1) : configured as general-purpose output -> set this;
- PORT_PCR_MUX(1)
- - Pin Mux Control -> select the pins role
- - 001 : set to GPIO
- if (PTC->PDIR & (1<<PTC12)) {
- PTD-> PCOR |= 1<<PTD0;
- }

Code understand

- PTC->PDIR & (1<<PTC12)
- PDIR : Port Data Input Register -> capture the logic level driven in each general-purpose input pin.
- PDI : Port Data Input
- 0b – Pin logic level is logic 0, or is not configured for use by digital function. -> What is the digital function?
- 1b – Pin logic level is logic 1.
- PTD-> PCOR |= 1<<PTD0;
- PCOR : Port Clear Output Register (PCOR) -> configures whether to clear the fields of PDOR.
- 0b – Corresponding bit in PDORn does not change.
- 1b – Corresponding bit in PDORn is cleared to logic 0.

Code understand

- else {
- PTD-> PSOR |= 1<<PTD0;
- }
- PSOR : Port Set Output Register -> set the fields of the PDOR.
- 0b – Corresponding bit in PDORn does not change.
- 1b – Corresponding bit in PDORn is set to logic 1.

Code Assignment

```
#include "device_registers.h"

#define PTD0 0

#define PTC13 13


void WDOG_disable (void)

{

  WDOG->CNT=0xD928C520;     /* Unlock watchdog        */

  WDOG->TOVAL=0x0000FFFF;   /* Maximum timeout value     */

  WDOG->CS = 0x00002100;    /* Disable watchdog        */

}
```

Code Assignment

```
int main(void)
{
    int counter = 0;

    WDOG_disable();/* Disable Watchdog in case it is not done in startup code */
    PCC-> PCCn[PCC_PORTC_INDEX] = PCC_PCCn_CGC_MASK;/* Enable clocks to peripherals (PORT modules) */
    PCC-> PCCn[PCC_PORTD_INDEX] = PCC_PCCn_CGC_MASK;/* Enable clock to PORT C*/
                        /* Enable clock to PORT D*/

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

    PTD->PDDR |= 1<<PTD0;       /* Port D0: Data Direction= output */
    PORTD->PCR[0] = PORT_PCR_MUX(1); /* Port D0: MUX = GPIO */
```

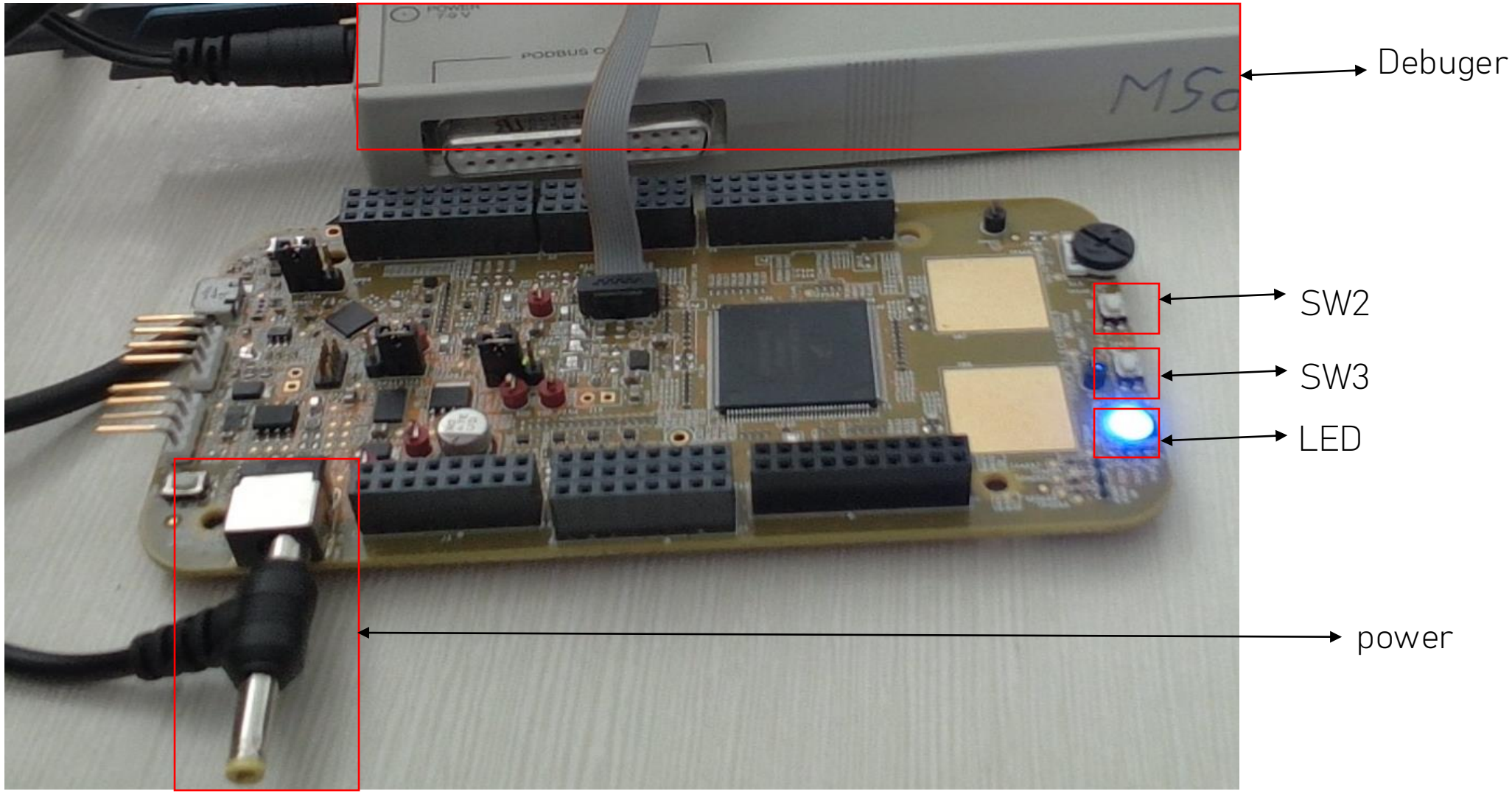
Code Assignment

```
for(;;)
	{
	  if (PTC->PDIR & (1<<PTC13)) { //push switch
	    PTD-> PCOR |= 1<<PTD0; // clear, led off
	  }
	  else {                        /* -If BTN0 was not pushed*/
	    PTD-> PSOR |= 1<<PTD0; // set1, led on
	  }
	  counter++;
	}
}
```

Debug step

1) change code in S32DS
2) Build project
3) open trace32
4) run script -> Desktop
5) Load Elf
6) CPU->In target reset
7) CPU Register reset
8) Run
9) pause
10) While coming out of trace 32->CPU in target reset

Debug result



AUTOSAR

2 types - AUTOSAR, Non-AUTOSAR

What is AUTOSAR? - **AUTomotive Open System Architecture**

- **Software standard platform.**

Benefit of use AUTOSAR

1. Reusability 2. Development cost 3. Time saving

2 types of AUTOSAR

1. Classic AUTOSAR 2. Adaptive AUTOSAR

-> We use Classic AUTOSAR.

- Same architecture enables use of ECU in different vehicles.

AUTOSAR Layer

3-layer = BSW, RTE, Application Layer

BSW: Basic software

RTE: Runtime environment

Application layer

RTE - mid-line. It connects Application layer & BSW.

Application layer - It is smillar to front-end. It is directly related to what we see.

AUTOSAR Layer

BSW: Basic software

1. MCAL 2. ECAL 3. Service layer

MCAL (Microcontroller Abstraction Layer) - It provides drivers to control pins in MCU. (ADC, CAN, SPI, etc...). Software works independently not rely on the hardware.

ECAL (ECU Abstraction Layer) - Delivers higher software layers ECU specific drivers. It is independent of mcu and dependent on ECU hardware

Service layer – It works like os. It is Task manager. It orders what to do first.