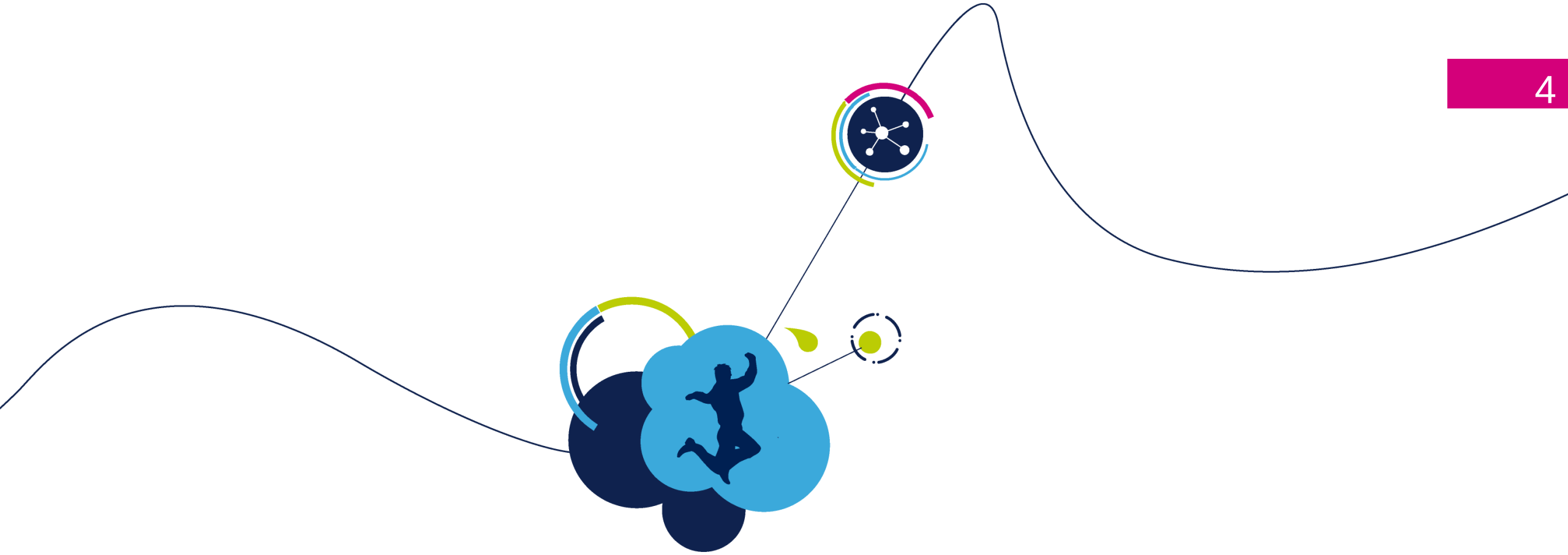


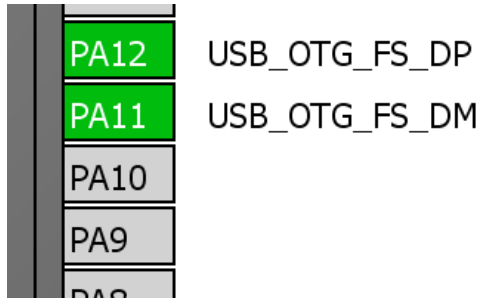
USB VCP Device

USB VCP Device

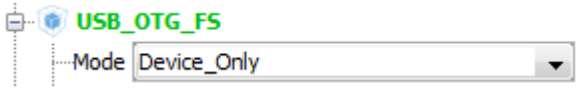
- Communication protocol emulating COM port
  - Standard for data communication between PC and embedded devices
- Widely spread – lot of VCP terminal applications for PC
- Drawbacks:
  - Additional endpoint usage, additional layer (COM port emulation)
  - Lack of native Plug & Play support

USB VCP Device

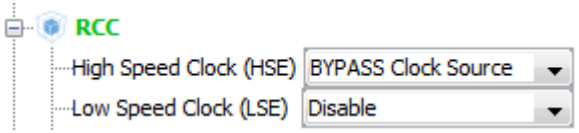
- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F4 > STM32F446 > LQFP144 > STM32F446ZETx



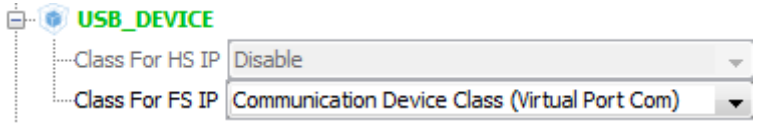
- Select USB FS OTG in device mode



- Select HSE clock
  - (Bypass HSE from STlink)

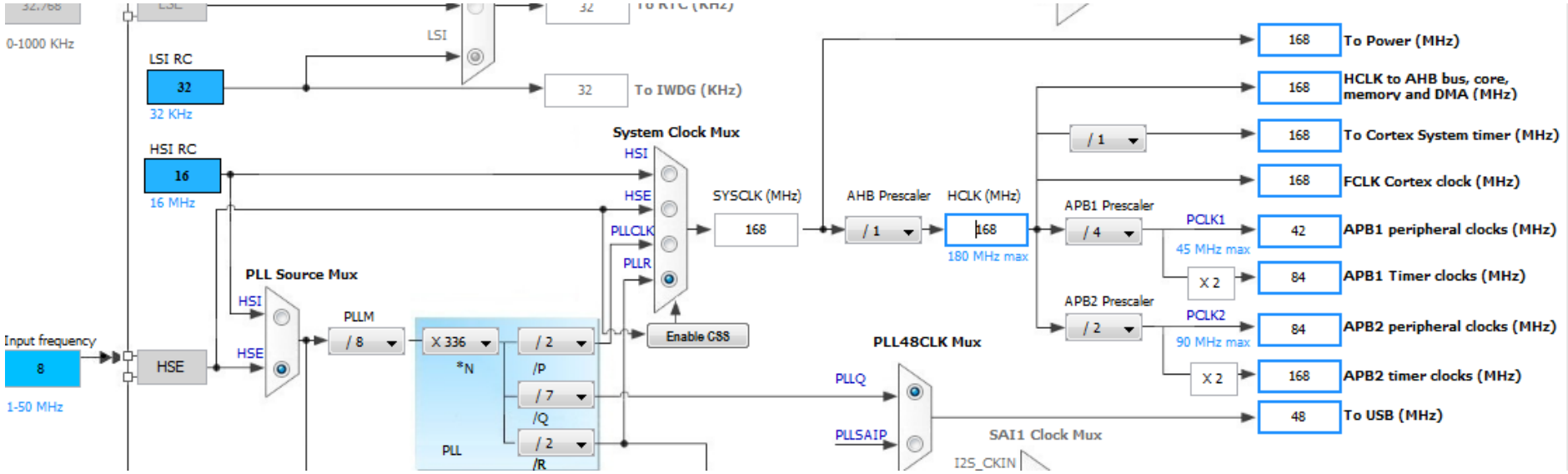


- Select CDC class in MiddleWares



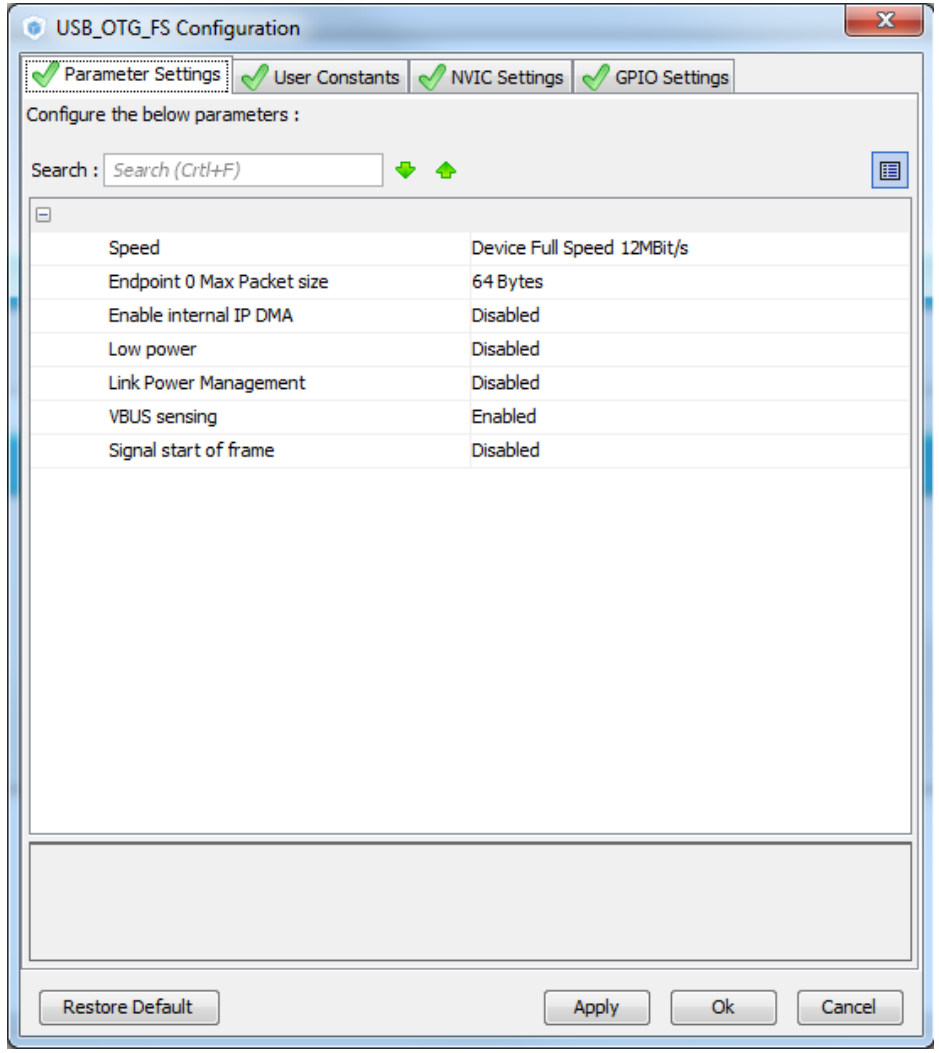
USB VCP Device

- Configure RCC clocks
  - Set 8 MHz HSE as PLL input
  - Set HCLK frequency 168 MHz
  - PLL parameters will be computed automatically



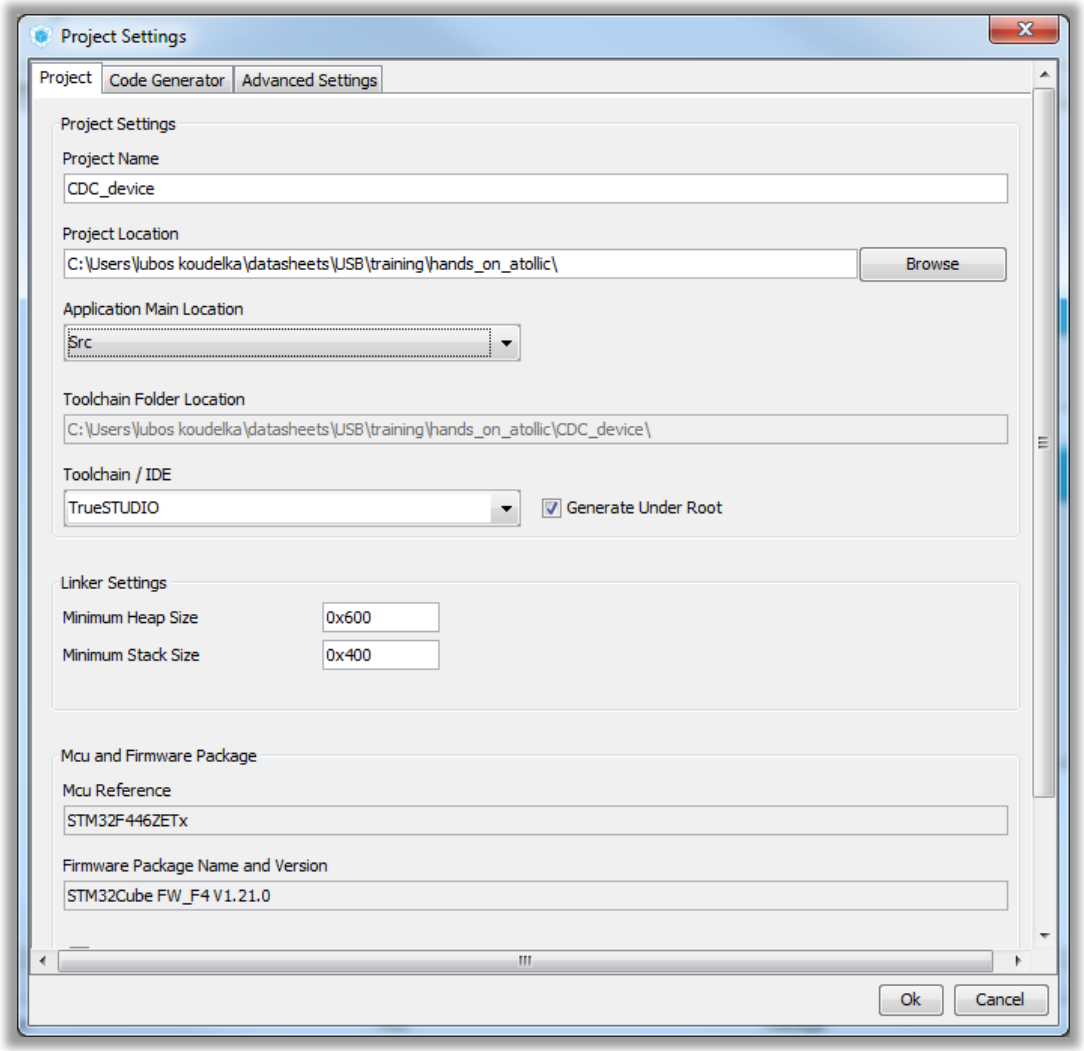
USB VCP Device

- USB OTG_FS configuration
  - Use default configuration
  - VBUS sensing is **disabled**



USB VCP Device

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Linker Settings
  - Increase heap size to 0x600
  - Default value 0x200 is not enough for VCP example
- Now we can Generate Code
  - Menu > Project > Generate Code



USB VCP Device

- How to send and receive data over VCP?
- Function which handle VCP operation are in generated file usbd_cdc_if.c
- Callback from control interface which allow to send COM port parameters

```
static int8_t CDC_Control_FS  (uint8_t cmd, uint8_t* pbuf, uint16_t length)
```

USB VCP Device

- Receive callback function
- In case you wand to receive more bytes you must call *USBD_CDC_ReceivePacket(&hUsbDeviceFS);*
- Otherwise the USB will not accept any data until you call this function

```
static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  return (USBD_OK);
  /* USER CODE END 6 */
}
```

USB VCP Device

- The Windows terminals using CDC commands to set correct line coding
  - But they also want to read this coding back
  - For this purpose we need to handle this actions
- This actions are done through function:

```
static int8_t CDC_Control_FS  (uint8_t cmd, uint8_t* pbuf, uint16_t length)
```

- We create a structure holding VCP parameters

```
/* USER CODE BEGIN PRIVATE_VARIABLES */
USBD_CDC_LineCodingTypeDef LineCoding = {
    115200, /* baud rate */
    0x00,   /* stop bits-1 */
    0x00,   /* parity - none */
    0x08    /* nb. of bits 8 */
  };
/* USER CODE END PRIVATE_VARIABLES */
```

USB VCP Device

- This part in CDC_Control_FS handling the storing and sending line coding information

```
case CDC_SET_LINE_CODING:
            LineCoding.bitrate    = (uint32_t)(pbuf[0] | (pbuf[1] << 8) |\
                                  (pbuf[2] << 16) | (pbuf[3] << 24));
            LineCoding.format     = pbuf[4];
            LineCoding.paritytype = pbuf[5];
            LineCoding.datatype   = pbuf[6];
    break;
  case CDC_GET_LINE_CODING:
            pbuf[0] = (uint8_t)(LineCoding.bitrate);
            pbuf[1] = (uint8_t)(LineCoding.bitrate >> 8);
            pbuf[2] = (uint8_t)(LineCoding.bitrate >> 16);
            pbuf[3] = (uint8_t)(LineCoding.bitrate >> 24);
            pbuf[4] = LineCoding.format;
            pbuf[5] = LineCoding.paritytype;
            pbuf[6] = LineCoding.datatype;
    break;
```

- Now communication with PC will be functional (Windows 7 demands line coding handling)

USB VCP Device

- Create simple loopback – device will echo received message to host

```
static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  CDC_Transmit_FS(Buf,*Len);
  return (USBD_OK);
  /* USER CODE END 6 */
}
```

USB VCP Device

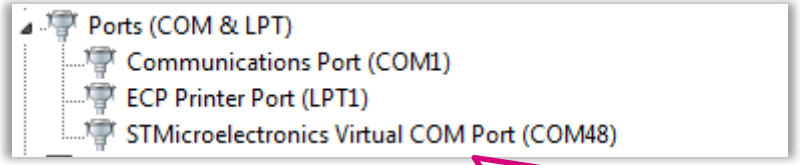
- User functions in usbd_cdc_if.c are missing the callback for CDC transmit

- Function USBD_CDC_DATAIn in usbd_cdc.c is called on complete IN transfer

```
static uint8_t  USBD_CDC_DataIn (USBD_HandleTypeDef *pdev, uint8_t epnum)
{
  USBD_CDC_HandleTypeDef   *hcdc = (USBD_CDC_HandleTypeDef*) pdev->pClassData;
  if(pdev->pClassData != NULL)
  {
    hcdc->TxState = 0;
    return USBD_OK;
  }
  else
  {
    return USBD_FAIL;
  }
}
```

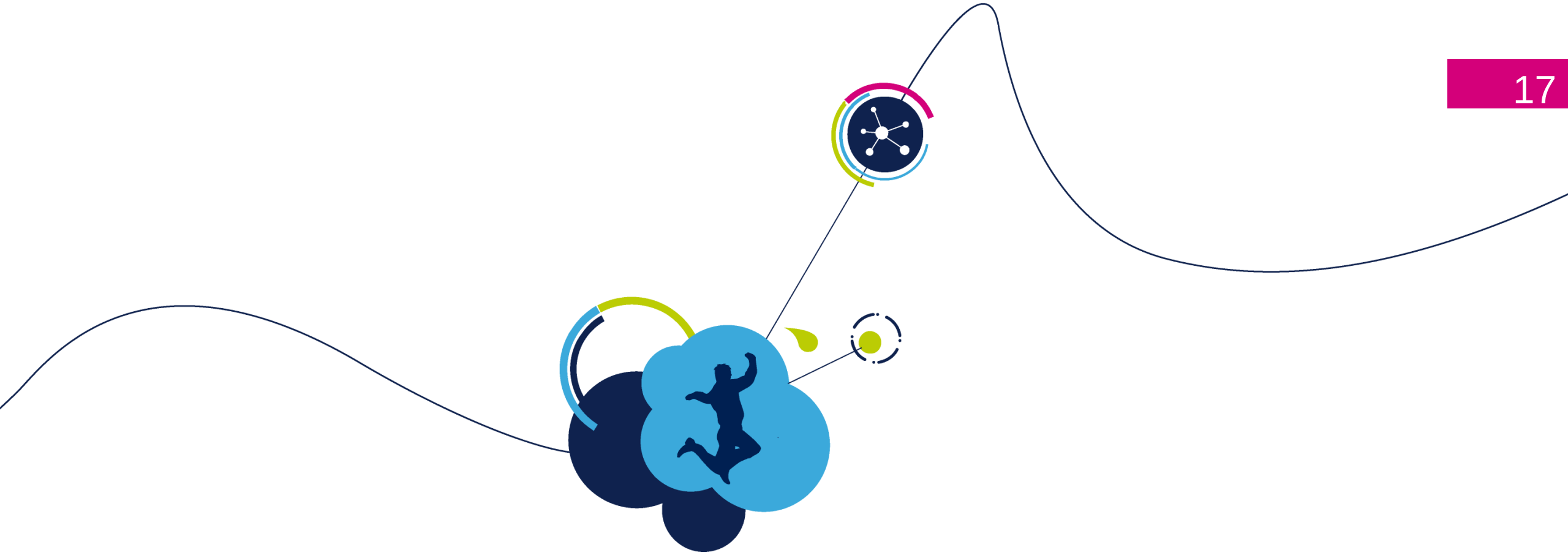
USB VCP Device

- Because Windows can select for VCP very high com port number you need the terminal where you can select the com number
- For example: http://realterm.sourceforge.net/
- If the USB is connected to PC it must be displayed in Device Manager



VCP with assigned port number

- In case you have no driver for VCP download it from ST webpages.



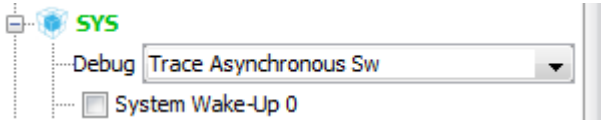
USB VCP SWD debug output

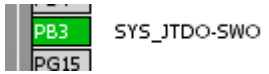
USB VCP SWD debug output

- In this lab will be described how to add debug option through ST-link
- This scenario can be used in any kind of project
- ST-link connection is mandatory

USB VCP SWD debug output

- Use the project from previous lab – USB VCP Device
- In CubeMX enable SW in Debug options





- And regenerate code
  - No need to do any other changes

USB VCP SWD debug output

- In usbd_cdc_if.c add functions and include to use debug output

```
/* USER CODE BEGIN INCLUDE */
#include <stdio.h>

/* USER CODE END INCLUDE */
```

```
/* USER CODE BEGIN PRIVATE_FUNCTIONS_DECLARATION */

int Debug_write(uint8_t *ptr, uint16_t len)
{
uint16_t i;
for (i = 0; i < len; i++)
{
   ITM_SendChar( *ptr++ );
}
return i;
}
/* USER CODE END PRIVATE_FUNCTIONS_DECLARATION */
```

USB VCP SWD debug output

- In usbd_cdc_if.c add functions and include to use debug output

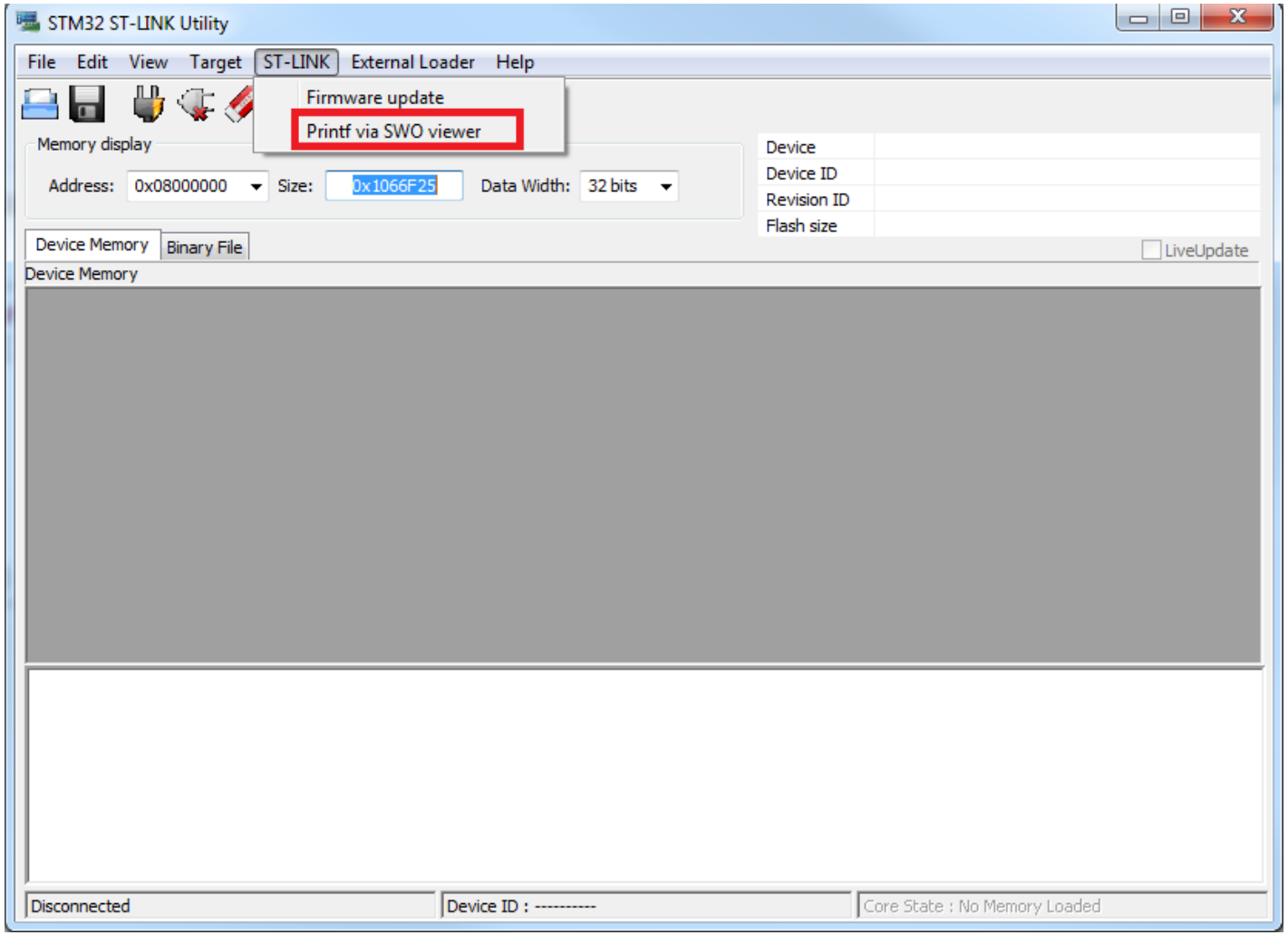
```
static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  …
  Debug_write("Message received\n",(uint16_t)17);

  return (USBD_OK);
  /* USER CODE END 6 */
}
```

- Now debug message will be sent each time a packet is received with CDC_Receive_FS function

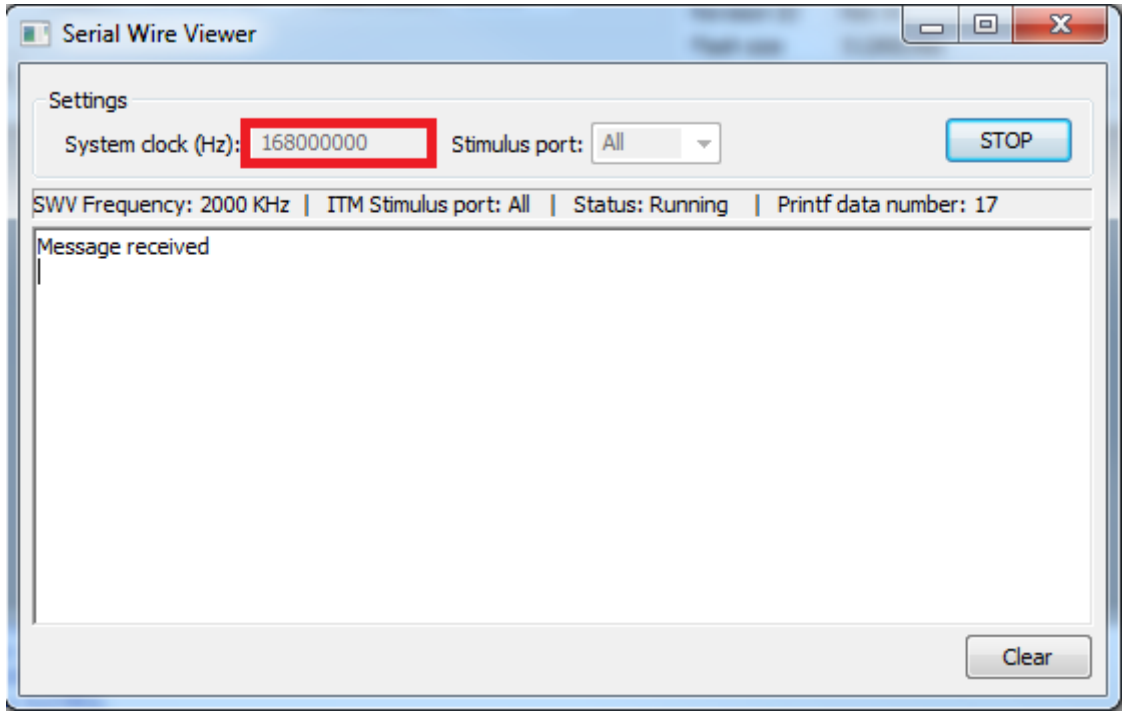
USB VCP SWD debug output

- Debug messages from SWO can be displayed in IDE or using ST-LINK utility



USB VCP SWD debug output

- Fill system clock setting according to the project setting and press START button



- Debug message are received by ST-LINK utility terminal

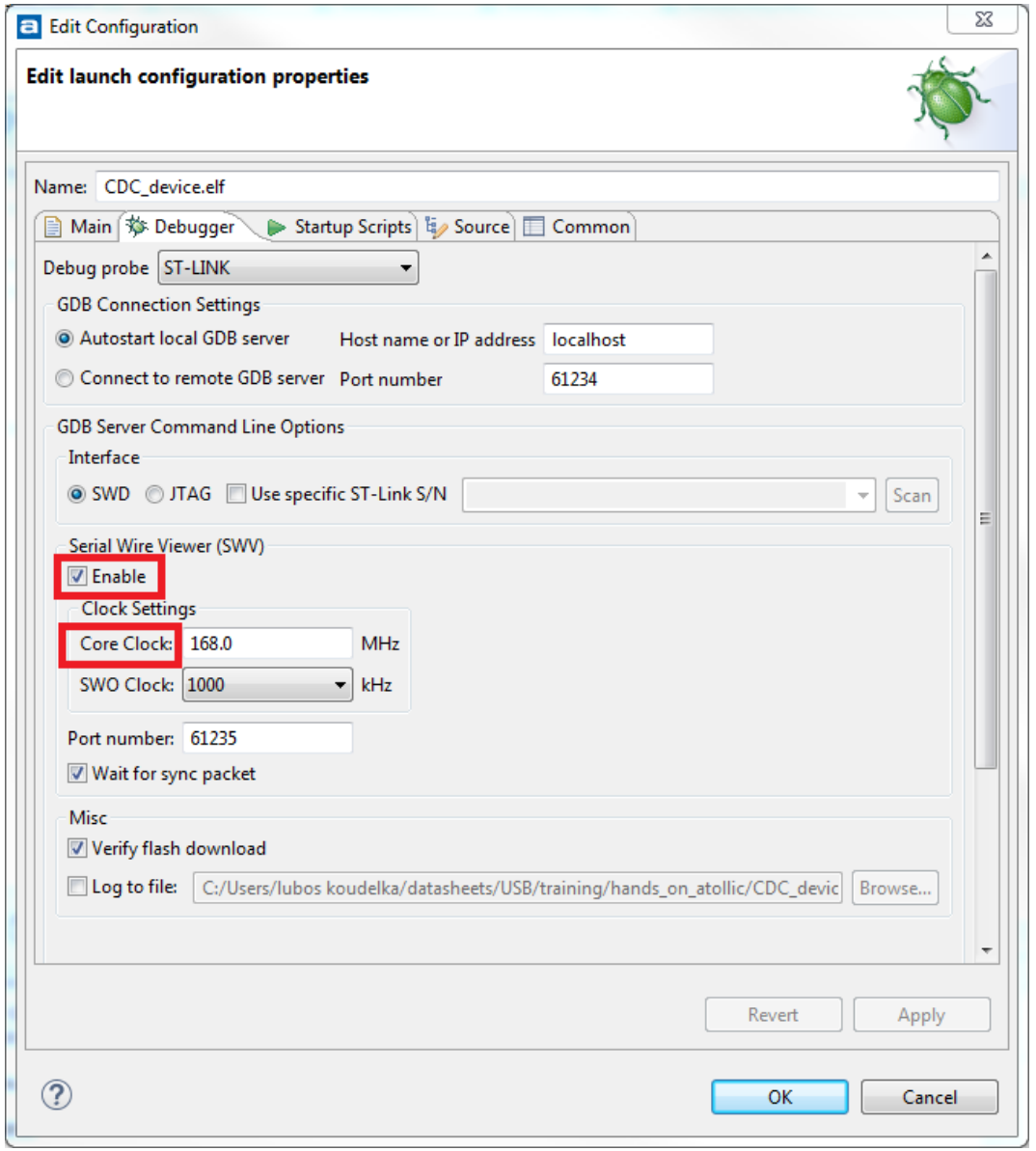
USB debug output

- There is possibility to activate debug messages inside USB library
  - Mainly host libraries have this functionality implemented
- For activation of this feature USBD_DEBUG_LEVEL inside usbx_conf.h (usbd or usbh) need to be raised to level grater than 0

```
#define USBD_DEBUG_LEVEL     3
```

Atollic SWO output activation

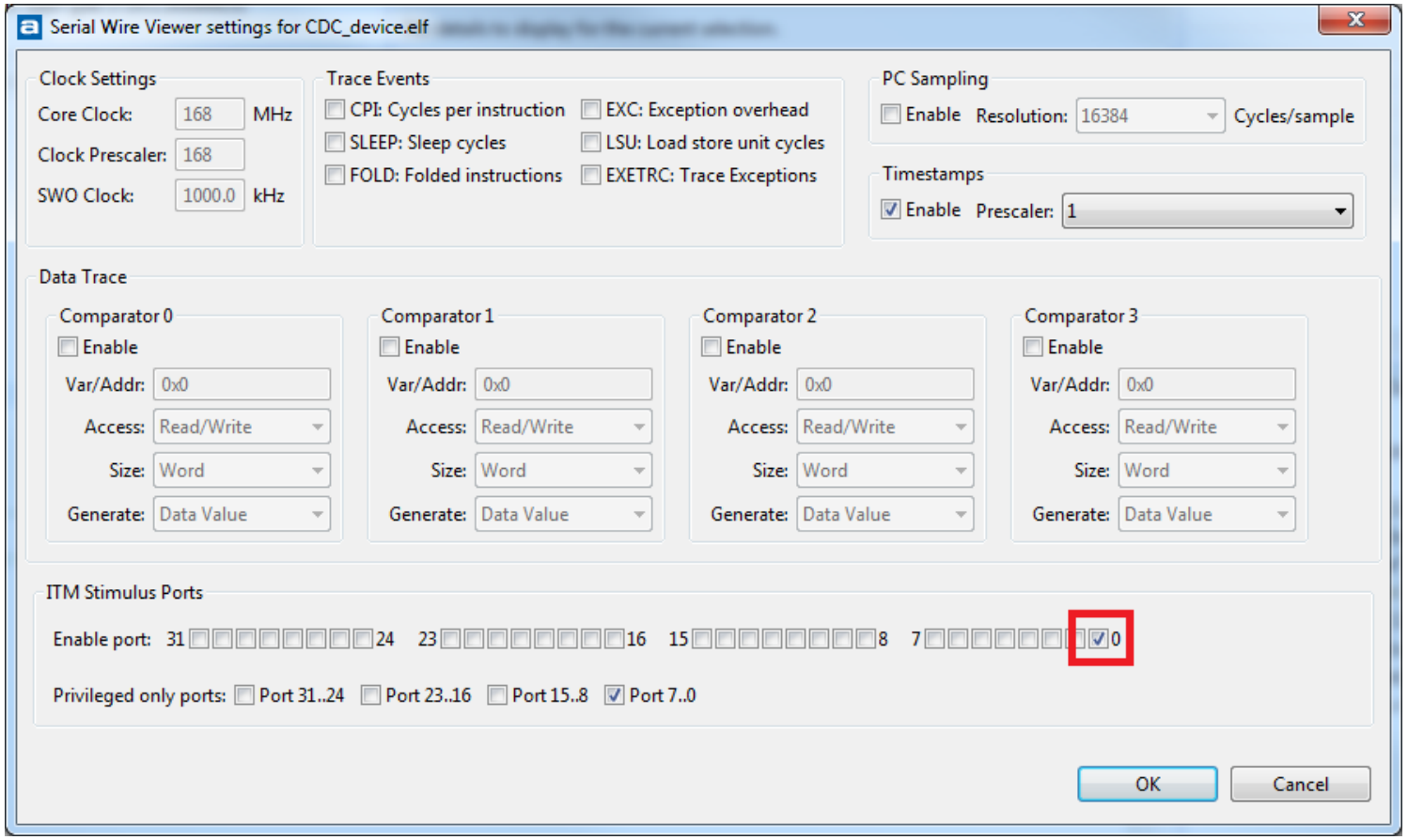
- Enable Serial Wire Viewer inside project configuration
  - Got to Project -> Properties -> Run/Debug Settings -> Launch configuration Edit… -> Debugger
- Core clock have to be set according to project clock settings

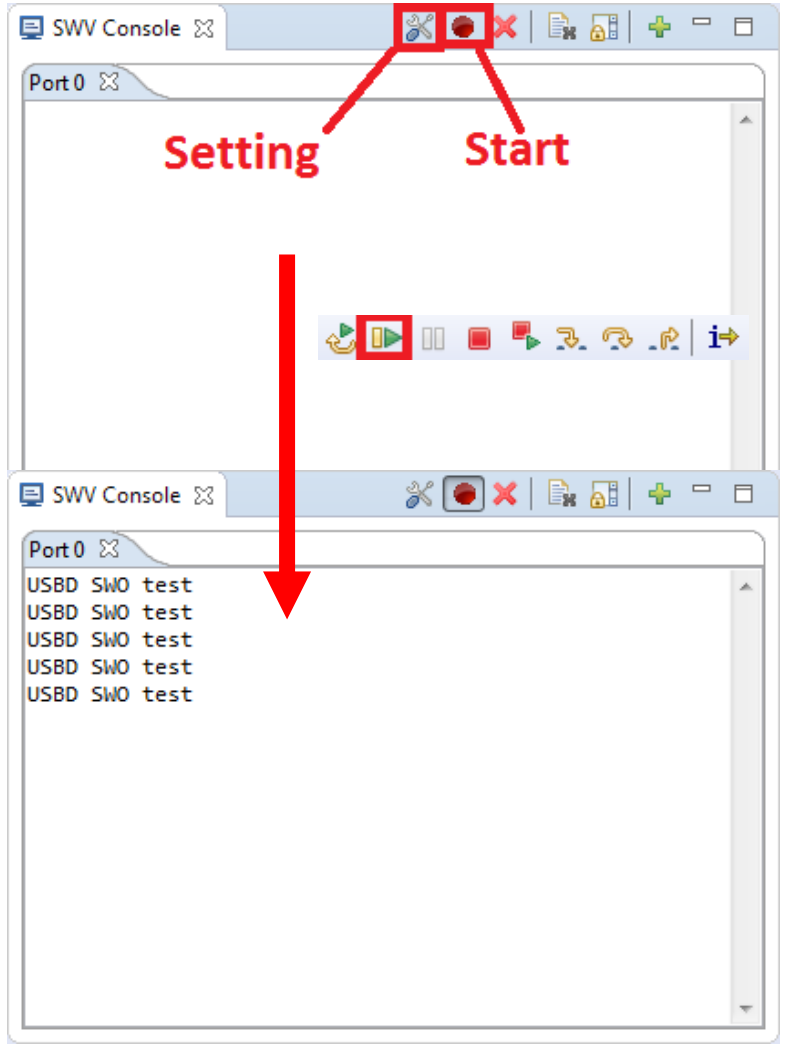


life.augmented

Atollic SWO output activation

- Now enter debugger, set and start SWV Console, then run the code





life.augmented