Inter-processor communication

Time 40 min

life.augmented

Lab Inter-processor communication

Lab Objective

- Illustrate the IPCC A7<-> M4 intercommunication
- Present STM32 eco-system M4 (System Workbench for STM32 IDE)
- Present Linux framework for IPCC & M4 firmware features
- Compile firmware for ARM Cortex-M4
- Load and start the firmware

Linux command on the host

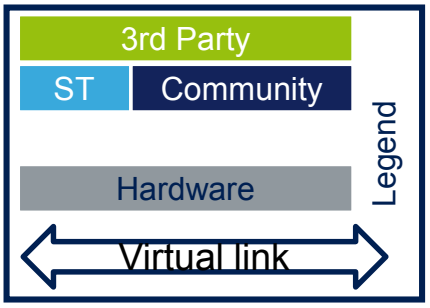
```
root@stm32mp1:/#
```

Linux command on the target

theory

practice

Lab Inter-processor communication



Virtual UART over RPMsg (I2C variant exists too)

application
/dev/ttyRPMSG0
stm32_rpmsg_tty
rpmsg
rproc_core
stm32_ipcc
OpenSTLinux Distribution
Cortex A
Cortex-M
TTY RPMSG channel
MCUSRAM
RX & TX Vrings
Buffers
IPCC
application
virtual_hal_uart
OpenAMP
HAL_IPCC

life.augmented

Lab Inter-processor communication

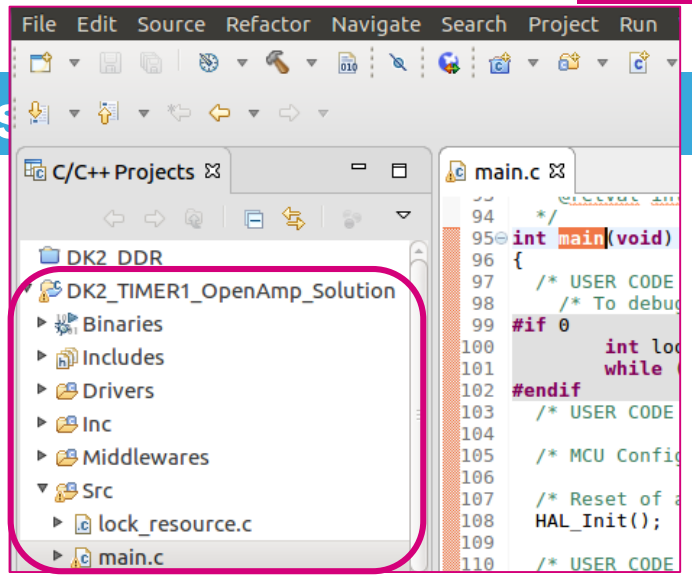
ARM Cortex-M4 firmware

- Firmware initializes the Timer 1 assigned to Cortex-M4
- The orange LED is toggled in an interrupt every 1 s
- The Timer 1 assigned to the Cortex-M4 is stopped/restarted by messages received over OpenSTLinux virtual UART
- Messages received by Cortex-M4 are echoed back to OpenSTLinux over the virtual UART

Lab Inter-processor communication

M4 firmware source code, simple APIs

- Upon receipt M4 sends back the message to OpenSTLinux
- Stop/restart messages can be used to control Timer 1




```
MX_IPCC_Init();
MX_OPENAMP_Init(RPMSG_REMOTE, NULL);

huart0.rvdev =  &rvdev;
log_info("Virtual UART0 OpenAMP-rpmsg channel creation\r\n");
if (VIRT_UART_Init(&huart0) != VIRT_UART_OK) {
```

```
void VIRT_UART0_RxCpltCallback(VIRT_UART_HandleTypeDef *huart)
{

    log_info("Msg received on VIRTUAL UART0 channel:  %s \n\r", (char *) huart->pRxBuffPtr);

    /* copy received msg in a variable to sent it back to master processor in main infinite
    VirtUart0ChannelRxSize = huart->RxXferSize < 100? huart->RxXferSize : 99;
    memcpy(VirtUart0ChannelBuffRx, huart->pRxBuffPtr, VirtUart0ChannelRxSize);
    VirtUart0RxMsg = SET;
}
```

```
while (1)
{

  OPENAMP_check_for_message();

  /* USER CODE END WHILE */
  if (VirtUart0RxMsg) {
    VirtUart0RxMsg = RESET;

    if (!strncmp((char *)VirtUart0ChannelBuffRx,"stop",4)){
        HAL_TIM_Base_Stop_IT(&TimHandle);
    }
    else if (!strncmp((char *)VirtUart0ChannelBuffRx,"restart",7)){
        HAL_TIM_Base_Start_IT(&TimHandle);
    }

    VIRT_UART_Transmit(&huart0, VirtUart0ChannelBuffRx, VirtUart0ChannelRxSize);
  }
```

life.augmented

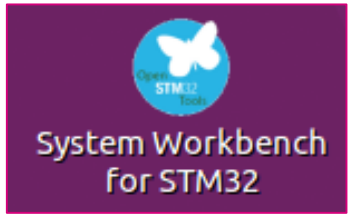
Lab Inter-processor communication

Open DK2_TIMER1_OpenAmp_Solution SW4STM32 Project

1) Open a new terminal console 
2) Unzip the lab material on the desktop directory

```
unzip $HOME/Desktop/InputLabMaterial/Lab-M4/STM32Cube_FW_MP1_handsOn.zip -d $HOME/Desktop/
```

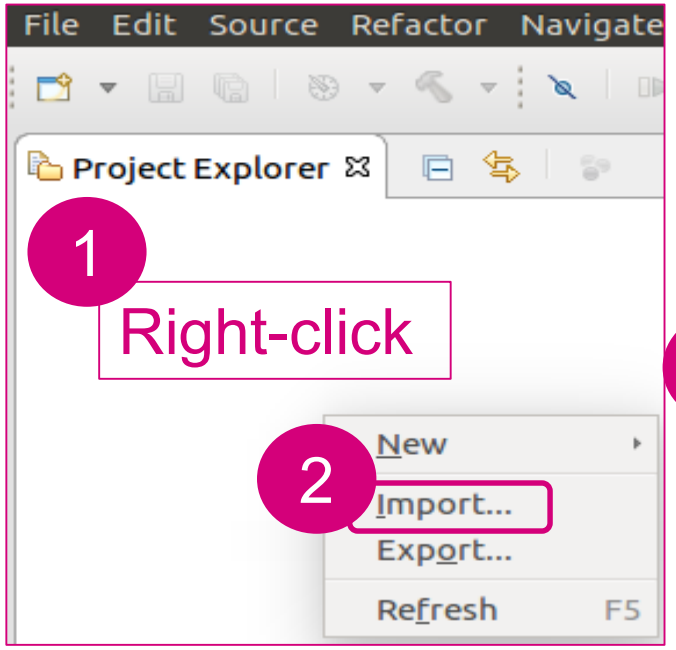
2) Launch System Workbench for STM32, double-click on the symbol

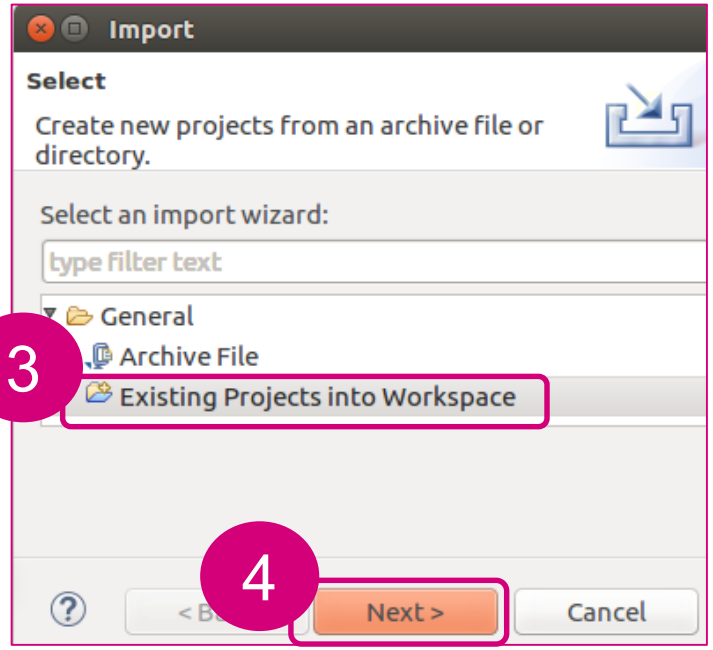


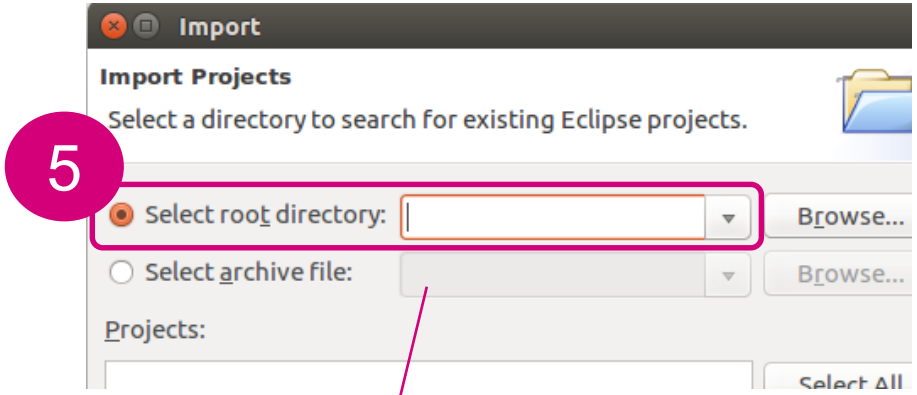
Lab Inter-processor communication

Open DK2_TIMER1_OpenAmp_Solution *SW4STM32* Project

- Import a prepared project into System Workbench for STM32







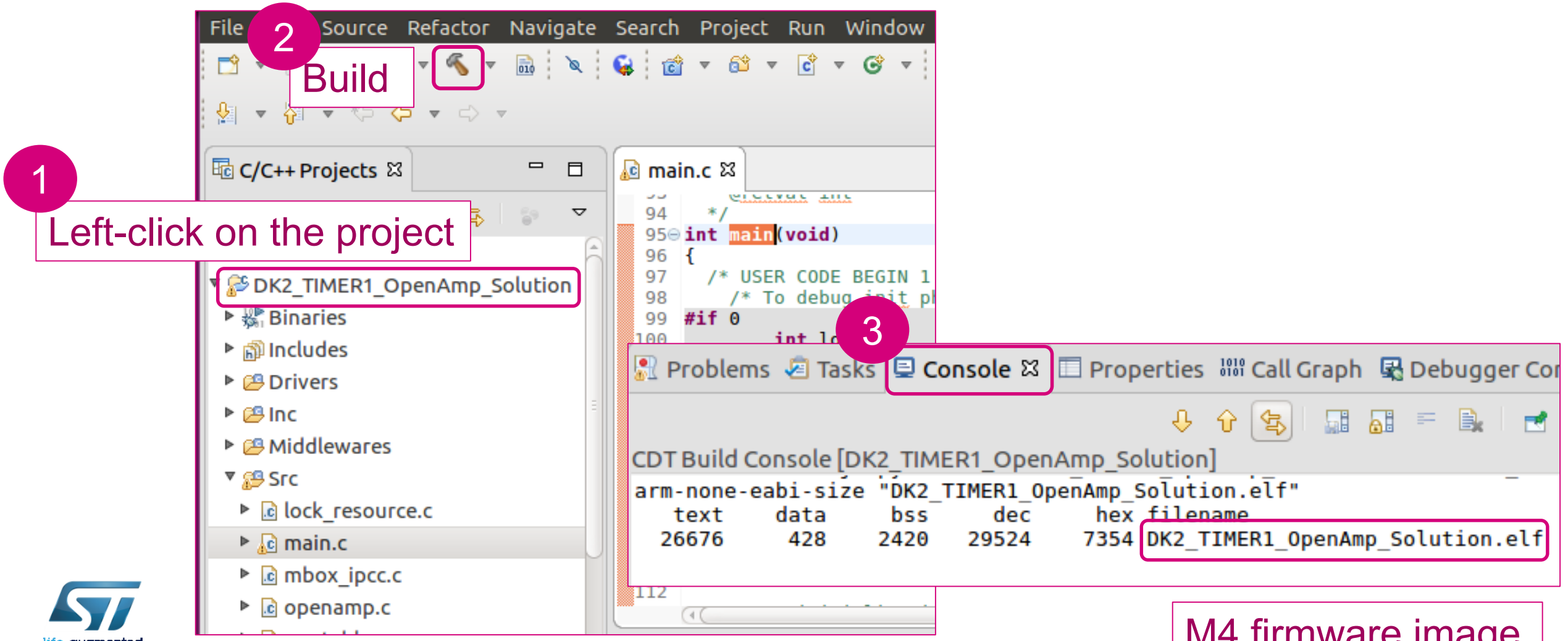
- After the 5th step press Finish

$HOME/Desktop/DK2_TIMER1_OpenAmp_Solution

life.augmented

Lab Inter-processor communication

Build firmware for ARM Cortex-M4



M4 firmware image

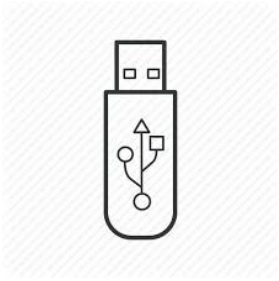
life.augmented

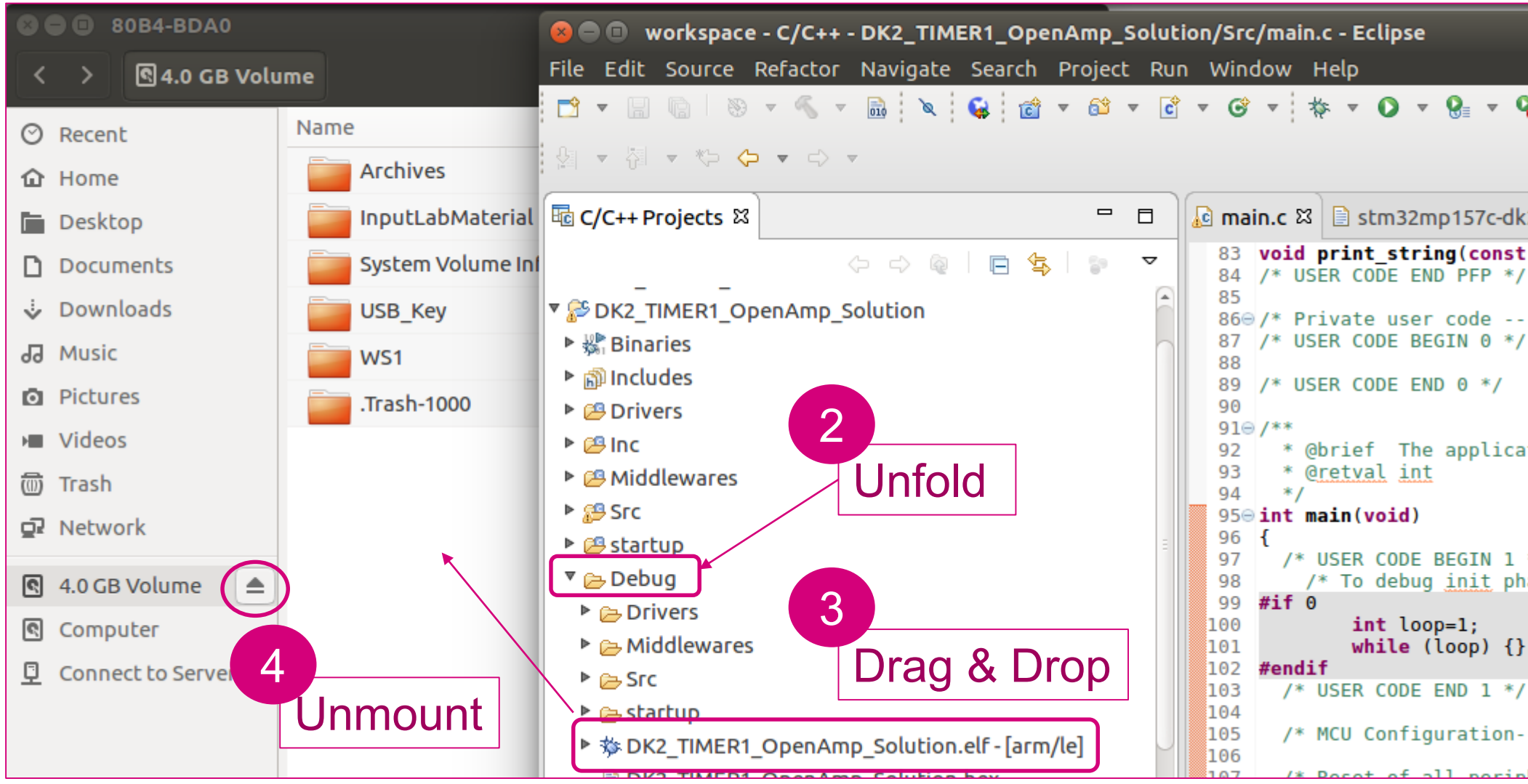
Lab Inter-processor communication

Copy firmware to a USB key

1

- Plug a USB key on Linux host

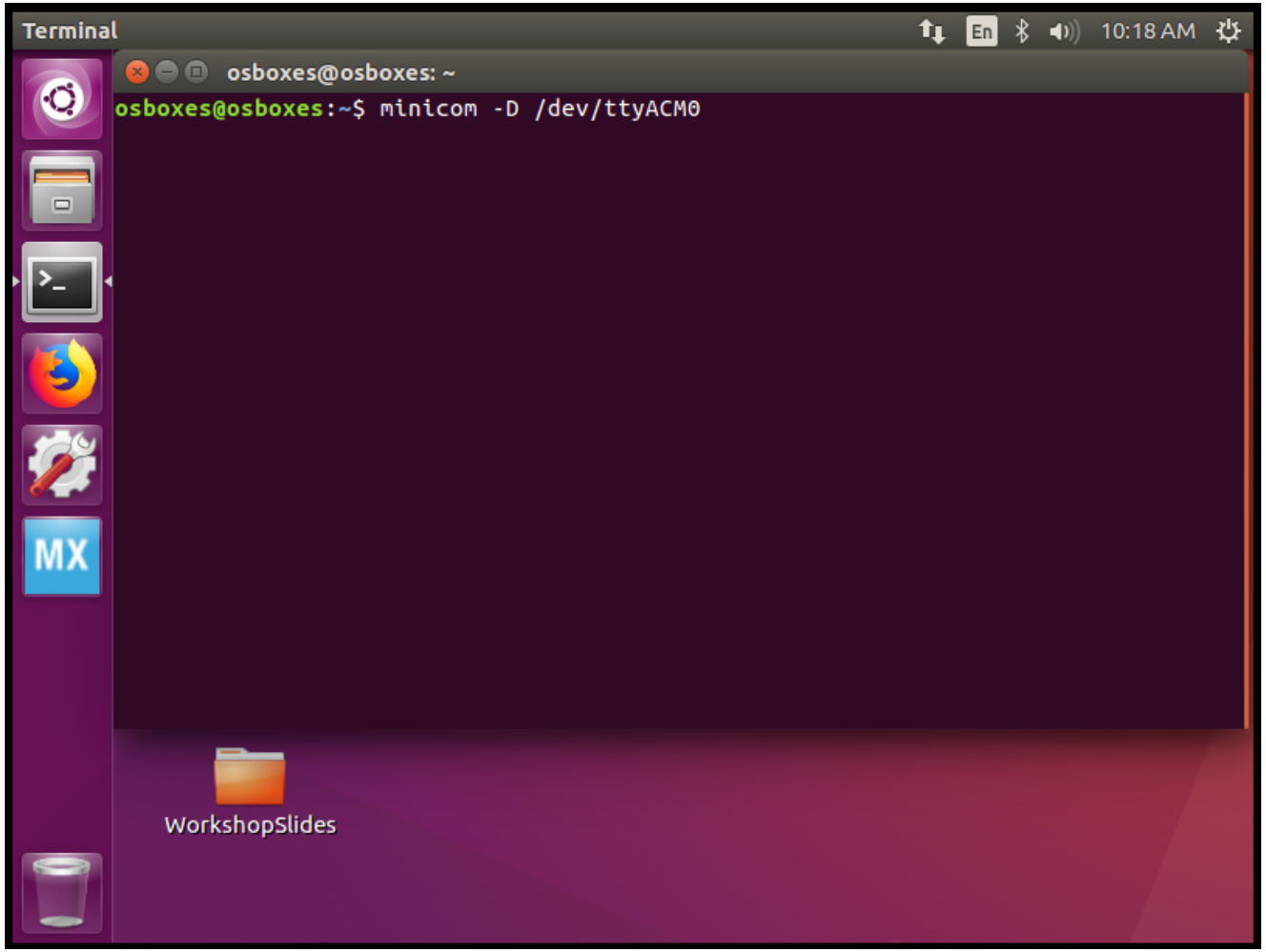




life.augmented

Lab Inter-processor communication

Start minicom



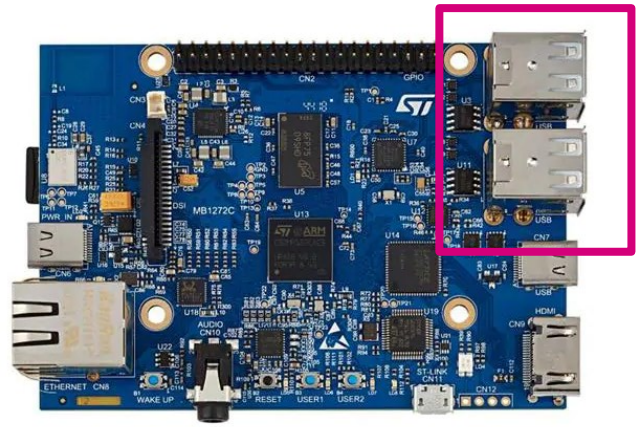
- Start a minicom terminal in order to establish a serial connection between Linux host and the target

```
minicom -D /dev/ttyACM0
```

Lab Inter-processor communication

Plug in the USB stick

1) Plug the USB stick into one of the 4 USB host ports on the discovery board
2) Observe the log in the minicom terminal indicating that the USB stick has been recognized



```
root@stm32mp1:/# [   64.100038] usb 2-1.4: new high-speed USB device number 3 using ehci-platform
[   64.257216] usb-storage 2-1.4:1.0: USB Mass Storage device detected
[   64.263116] scsi host0: usb-storage 2-1.4:1.0
[   65.271348] scsi 0:0:0:0: Direct-Access     SanDisk  Cruzer           1.26 PQ: 0 ANSI: 5
[   65.288637] sd 0:0:0:0: [sda] 7821312 512-byte logical blocks: (4.00 GB/3.73 GiB)
[   65.294885] sd 0:0:0:0: Attached scsi generic sg0 type 0
[   65.306175] sd 0:0:0:0: [sda] Write Protect is off
[   65.311439] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or
FUA
[   65.345645]  sda: sda1
[   65.360110] sd 0:0:0:0: [sda] Attached SCSI removable disk

root@stm32mp1:/#
```

life.augmented

Lab Inter-processor communication

Mount the USB stick on the Linux host

1) Mount the USB stick file system on the Linux host

```
root@stm32mp1:/#
```

```
mount -t vfat /dev/sda1 /mnt -v
```

Lab Inter-processor communication

Copy the binary file to the local directory

1) Mount the USB stick file system on the Linux host

```
root@stm32mp1:/#
```

```
mount -t vfat /dev/sda1 /mnt -v
```

2) Copy from the USB stick DK2_TIMER1_OpenAmp_Solution.elf on the target

```
root@stm32mp1:/#
```

```
cp /mnt/DK2_TIMER1_OpenAmp_Solution.elf /lib/firmware
```

Lab Inter-processor communication

Sync

1) Mount the USB stick file system on the Linux host

```
root@stm32mp1:/#
```

```
mount -t vfat /dev/sda1 /mnt -v
```

2) Copy from the USB stick DK2_TIMER1_OpenAmp_Solution.elf on the target

```
root@stm32mp1:/#
```

```
cp /mnt/DK2_TIMER1_OpenAmp_Solution.elf /lib/firmware
```

3) Sync

```
root@stm32mp1:/#
```

```
sync
```

Lab Inter-processor communication

Load firmware on RAM

1) Specify the name of the firmware to load

```
root@stm32mp1:/#
```

```
echo DK2_TIMER1_OpenAmp_Solution.elf > /sys/class/remoteproc/remoteproc0/firmware
```

2) Load and start the firmware

```
root@stm32mp1:/#
```

```
echo start  > /sys/class/remoteproc/remoteproc0/state
```

- Getting the orange LED blinking every 1 s = firmware start success

Lab Inter-processor communication

Load firmware on RAM

- To know if the firmware is loaded and running on M4 (offline/running)

```
root@stm32mp1:/#
```

```
cat /sys/class/remoteproc/remoteproc0/state
```

- To stop the firmware

```
root@stm32mp1:/#
```

```
echo stop > /sys/class/remoteproc/remoteproc0/state
```

- To start the firmware again

```
root@stm32mp1:/#
```

```
echo start  > /sys/class/remoteproc/remoteproc0/state
```

Lab Inter-processor communication

Control Timer 1 assigned to Cortex-M4 from the OpenSTLinux side

1) Configuration to see on A7 side the messages sent by M4 firmware fw:

```
root@stm32mp1:/#
```

```
stty -onlcr -echo -F /dev/ttyRPMSG0
cat /dev/ttyRPMSG0 &
```

2) Send a “Hello Virtual UART0” message from A7 to M4, which sends it back

```
root@stm32mp1:/#
```

```
echo "Hello Virtual UART0" >/dev/ttyRPMSG0
```

```
root@stm32mp1:~#  stty -onlcr -echo -F /dev/ttyRPMSG0
root@stm32mp1:~#     cat /dev/ttyRPMSG0 &
[1] 14067
root@stm32mp1:~# echo "Hello Virtual UART0" >/dev/ttyRPMSG0
root@stm32mp1:~# Msg received on VIRTUAL UART0 channel

Hello Virtual UART0

root@stm32mp1:~# echo "stop" > /dev/ttyRPMSG0
root@stm32mp1:~# Msg received on VIRTUAL UART0 channel

Led Blink Stopped in state OFF


root@stm32mp1:~# echo "restart" > /dev/ttyRPMSG0
root@stm32mp1:~# Led BMsg received on VIRTUAL UART0 channel

Led Blink Started

restart
```

3) To stop/restart Timer 1 (the orange LED stops/restarts blinking)

```
root@stm32mp1:/#
```

```
echo "stop" > /dev/ttyRPMSG0
echo "restart" > /dev/ttyRPMSG0
```

life.augmented

Thank you