



life.augmented

STP32MP1 coprocessor development handson

Johnson Zhang

March 2021

1 Debug M4 in engineering mode

2 M4 production mode demo

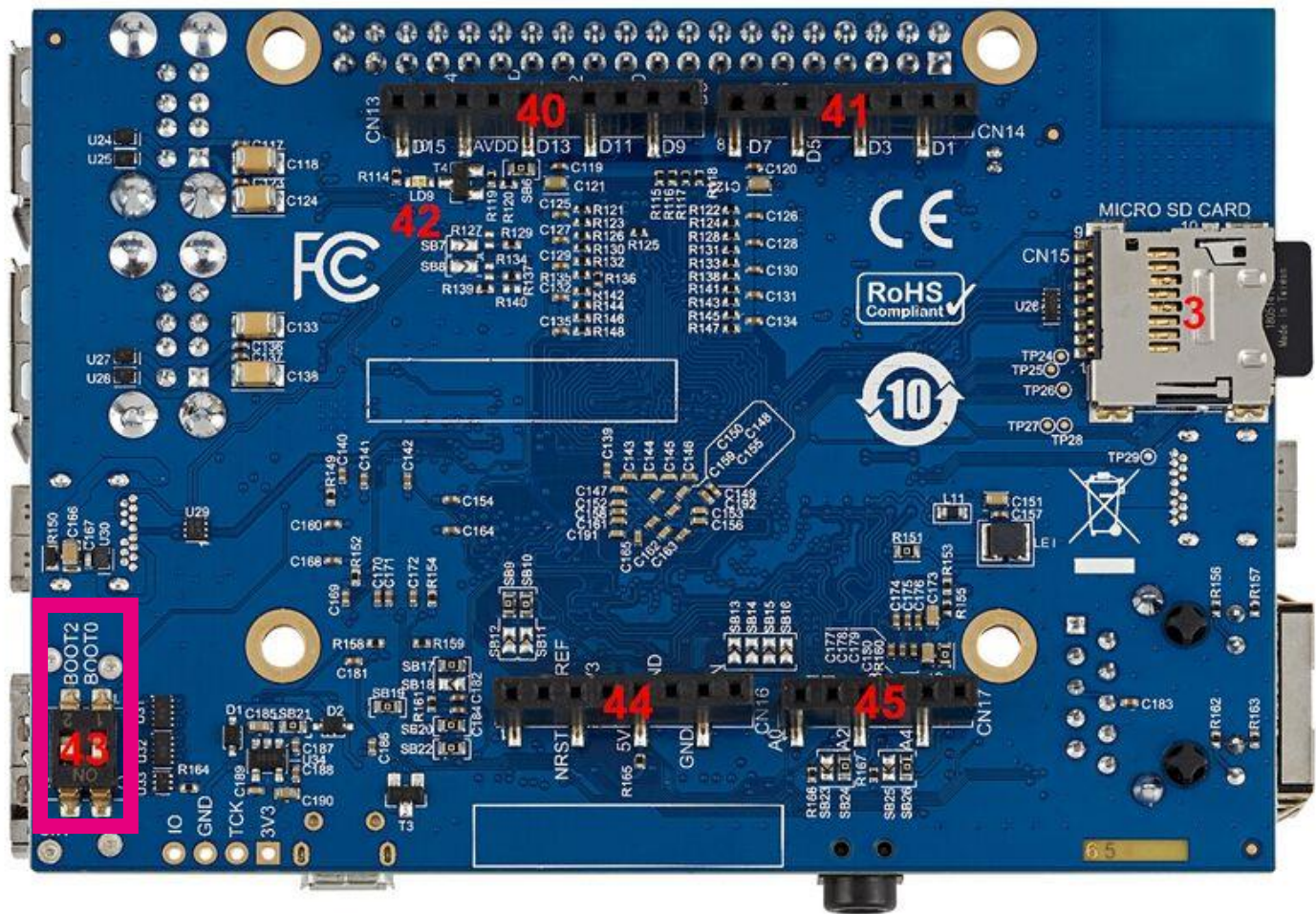
3 Inter-processor communication

Debug M4 in engineering mode



Engineering Mode

Set the board to Engineering mode



Boot mode	Boot 0	Boot 2
Forced USB boot for flashing	0	0
Not supported	ON	0
Engineering boot	0	ON
microSD card	ON	ON

Generating project with STM32CubeMX



Double click the STM32CubeMX icon on the desktop to open it

- Click on « Access to board selector »
- Filter on STM32MP1 (1)
- Then select your board (2) and click on Start project on the top right corner (3)
- Answer « Yes » to the popup « apply peripheral default settings »



The screenshot shows the 'New Project from a Board' window in STM32CubeMX. The 'Board Selector' tab is active. On the left, under 'Board Filters', the 'MCU Series' is set to 'STM32MP1'. A red box labeled '1' highlights the 'STM32MP1' checkbox in the list. On the right, the details for 'STM32MP157C-EV1' are shown. A red box labeled '2' highlights the 'Show view' button next to the board image in the 'Boards List' table. A red box labeled '3' highlights the 'Start Project' button in the top right corner.

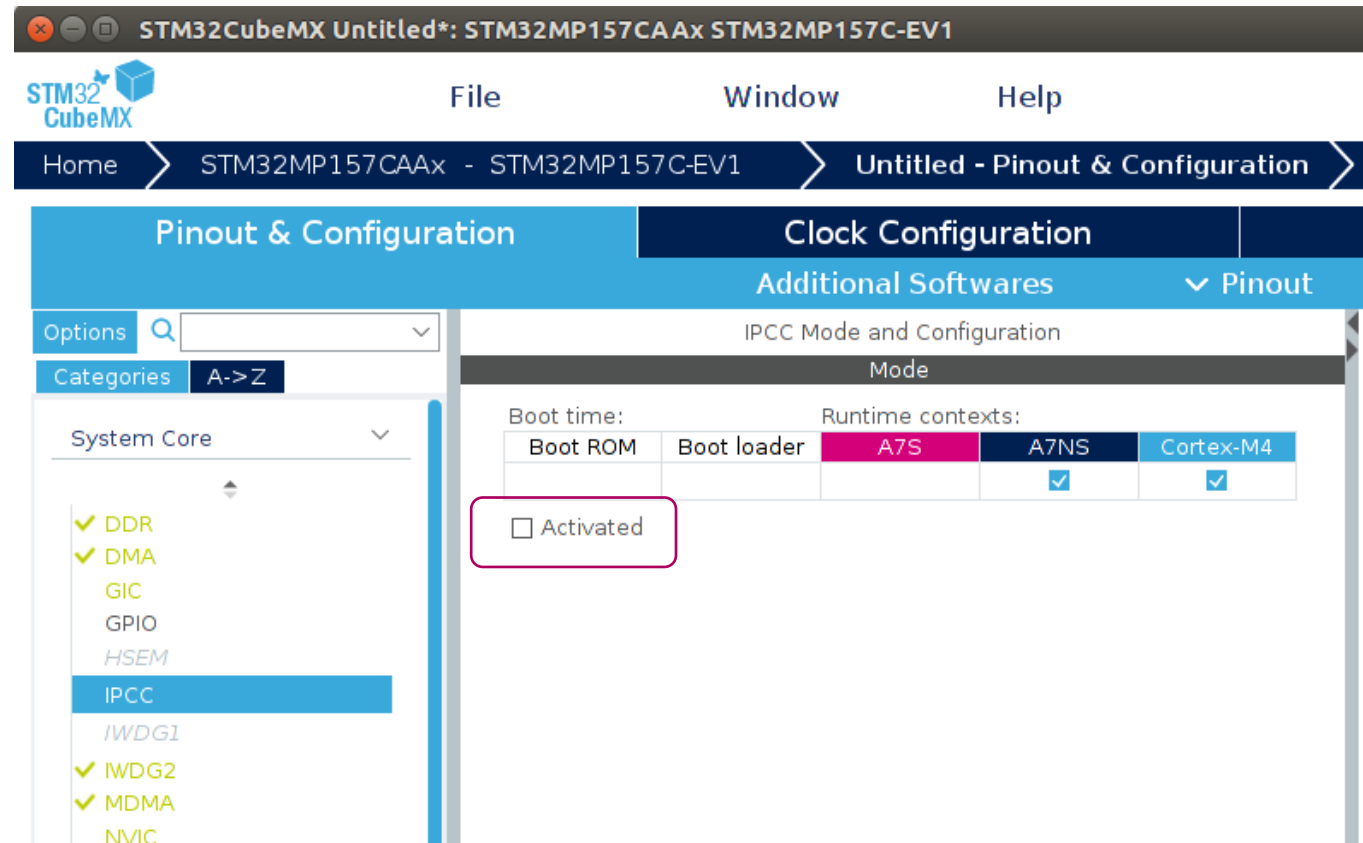
	Part No.	Type	Marketing Status	Unit Price (US\$)	Mounted Device	Kit Cont.	Included
☆	STM32MP157A-EV1	Evaluation Board		0.0	STM32MP157AAx		
☆	STM32MP157C-DK2	Discovery		0.0	STM32MP157CACx		
☆	STM32MP157C-EV1	Evaluation Board		0.0	STM32MP157CAx		

Engineering Mode



Disable OpenAMP and IPCC

Until « Activated » box in IPCC automatically disable OpenAMP in the same time.





Engineering Mode

Assign GPIO PH7 to M4.



STM32CubeMX Untitled*: STM32MP157CACx STM32MP157C-DK2

File Window Help

Home > STM32MP157CACx - STM32MP157C-DK2 > Untitled - Pinout & Configuration > GENERATE CODE

Pinout & Configuration Clock Configuration Project Manager Tools

Software Packs Pinout

GPIO Mode and Configuration

Configuration

Group By Peripherals

☒ SDMMC	☒ UART	☒ USART	☒ USB	☒ USBH
☒ I2C	☒ I2S	☒ PWR	☒ RCC	☒ RTC
☒ GPIO	☒ ADC	☒ DDR	☒ DSIHOST	☒ ETH
			☒ SAI	☒ HDMI

Search Signals
Search (Ctrl+F) ☐ Show only Modified Pins

Pin N...	Signal on...	GPIO mo...	GPIO Pull...	Maximu...	User Label	Modified
PH4	n/a	Output P...	No pull-u...	Low	WL_REG_...	☒
PH5	n/a	Input mo...	No pull-u...	n/a	BT_HOST...	☒
PH7	n/a	Output P...	No pull-u...	Low	LED_Y [L...	☒
PI11	n/a	Input mo...	No pull-u...	n/a	STUSB1...	☒
PZ6	n/a	Output P...	No pull-u...	Low	BT_REG_...	☒
PZ7	n/a	Output P...	No pull-u...	Low	BT DEV ...	☒

PH7 Configuration :

GPIO mode: Output Push Pull

GPIO Pull-up/Pull-down: No pull-up and no pull-down

Maximum output speed: Low

Pinout view System view

Enter User Label
Signal Unpinning
Pin Stacking
Pin Reserved

- ☐ Free
- ☐ Cortex-A7 secure
- ☐ Cortex-A7 non secure
- ☒ Cortex-M4

MCUs Selection Output DDR Interactive logs

Series	Lines	Mcu	Package	Required Peripherals
STM32MP1	STM32MP157	STM32MP157CACx	TFBGA361	None



Project Settings and generation

1. Fill project name
2. Set Project location (if default does not suit you)
3. Choose Toolchain/IDE
4. Generate Code

The screenshot displays the STM32CubeMX software interface. The title bar reads "STM32CubeMX Untitled*: STM32MP157CACx STM32MP157C-DK2". The menu bar includes "File", "Window", and "Help". The breadcrumb navigation shows "Home > STM32MP157CACx - STM32MP157C-DK2 > Untitled - Project Manager". A "GENERATE CODE" button is highlighted in the top right. The main interface has four tabs: "Pinout & Configuration", "Clock Configuration", "Project Manager" (selected), and "Tools". The "Project Manager" tab is divided into three sections: "Project", "Code Generator", and "Advanced Settings".

- Project Settings:**
 - Project Name:** "Engineer_mode_LED_example" (highlighted with a red box).
 - Project Location:** "/work/STM32CubeIDE/workspace_1.5.1/" (highlighted with a red box). A "Browse" button is next to it.
- Code Generator:**
 - Application Structure:** "Advanced" (dropdown menu). A checkbox "Do not generate the main()" is present.
 - Toolchain Folder Location:** "/work/STM32CubeIDE/workspace_1.5.1/Engineer_mode_LED_example/" (text field).
 - Toolchain / IDE:** "STM32CubeIDE" (dropdown menu, highlighted with a red box). A checkbox "Generate Under Root" is checked.
- Advanced Settings:**
 - Linker Settings:**
 - Minimum Heap Size:** "0x200" (text field).
 - Minimum Stack Size:** "0x400" (text field).
 - Mcu and Firmware Package:**
 - Mcu Reference:** "STM32MP157CACx" (text field).
 - Firmware Package Name and Version:** (empty text field).

At the bottom, there are tabs for "MCUs Selection", "Output", and "DDR Interactive logs". Below these is a table with the following data:

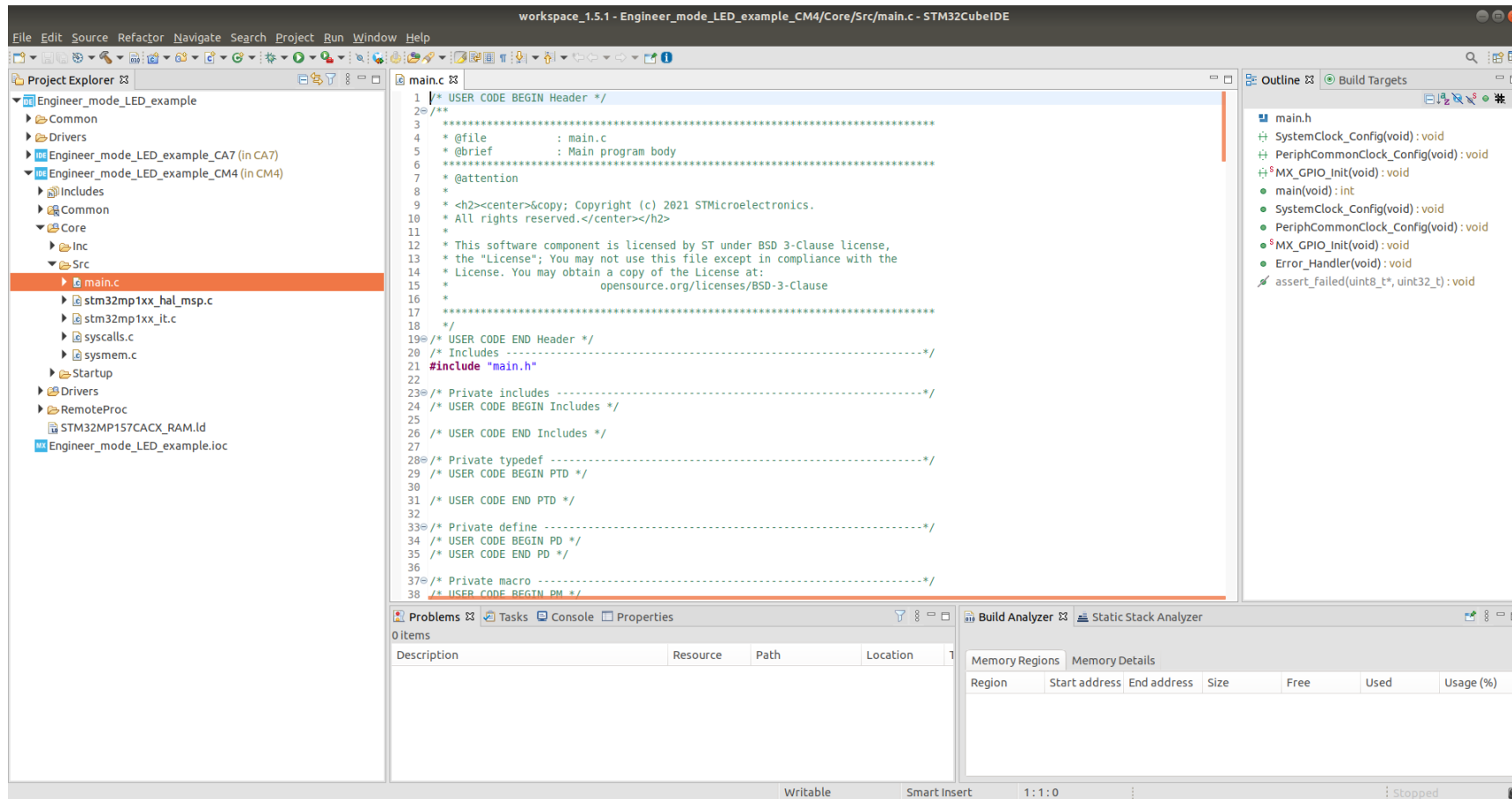
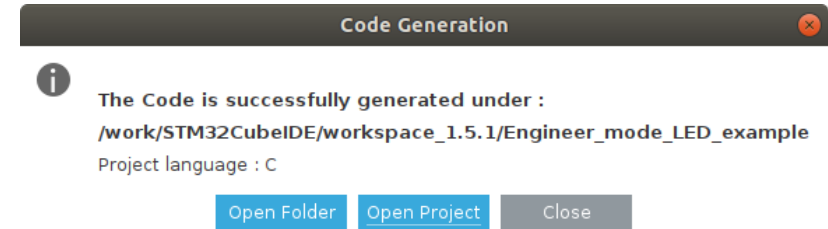
Series	Lines	Mcu	Package	Required Peripherals
STM32MP1	STM32MP157	STM32MP157CACx	TFBGA361	None

Engineering Mode



Open project in CubeIDE

Choose Open Project in the popup window,
Then the project will be opened in CubeIDE



 Add your own code to the user code region

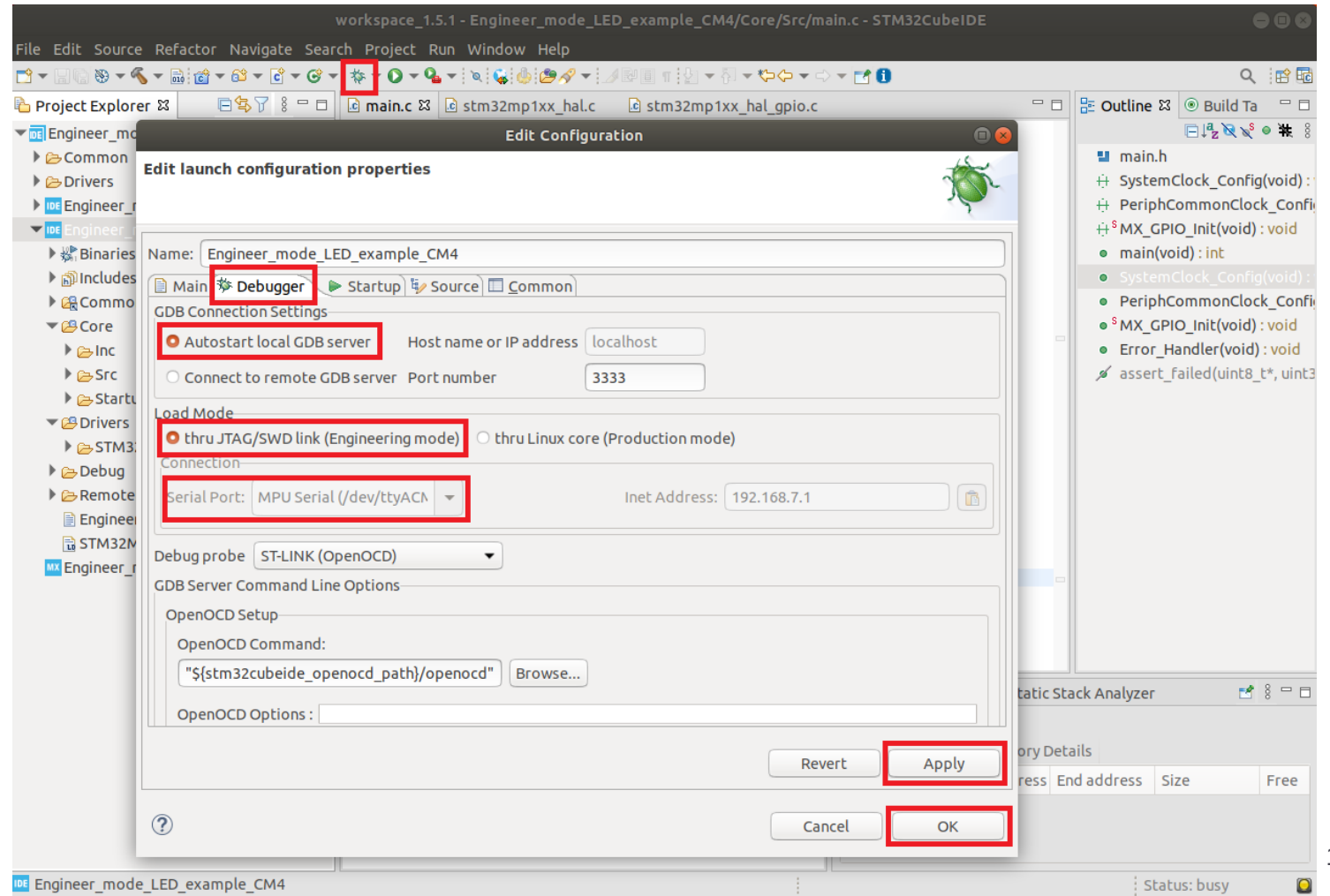
```
main.c  stm32mp1xx_hal.c  stm32mp1xx_hal_gpio.c
95
96  /* Initialize all configured peripherals */
97  MX_GPIO_Init();
98  /* USER CODE BEGIN 2 */
99
100 /* USER CODE END 2 */
101
102 /* Infinite loop */
103 /* USER CODE BEGIN WHILE */
104 while (1)
105 {
106     /* USER CODE END WHILE */
107
108     /* USER CODE BEGIN 3 */
109     HAL_GPIO_TogglePin(LED_Y_GPIO_Port, LED_Y_Pin);
110     HAL_Delay(250);
111 }
112 /* USER CODE END 3 */
113 }
114
115 /**
116  * @brief System Clock Configuration
117  * @retval None
118  */
```

Engineering Mode

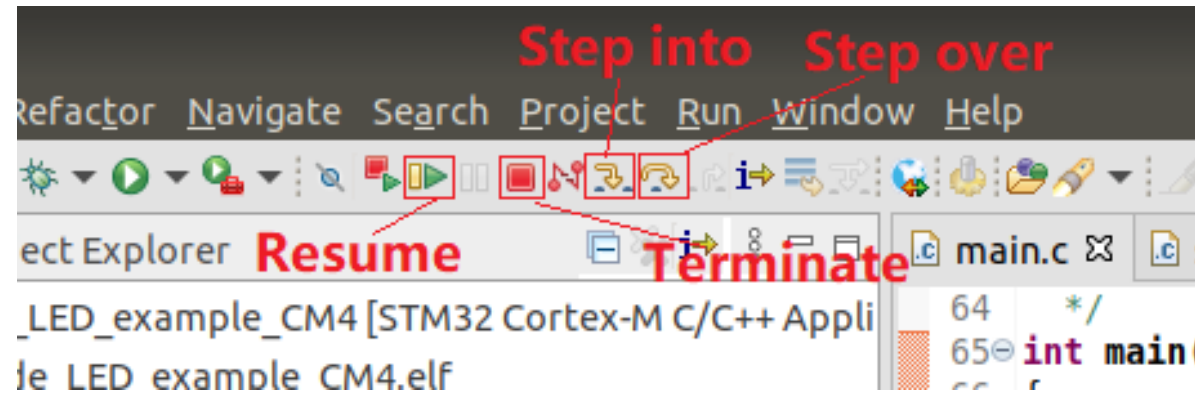


Debug configurations

1. Click the Debug tool on the toolbar to open the Debug configuration window
2. Configure the debug settings as below
3. Click Apply&OK to debug the code



 Exécute and Debug the code



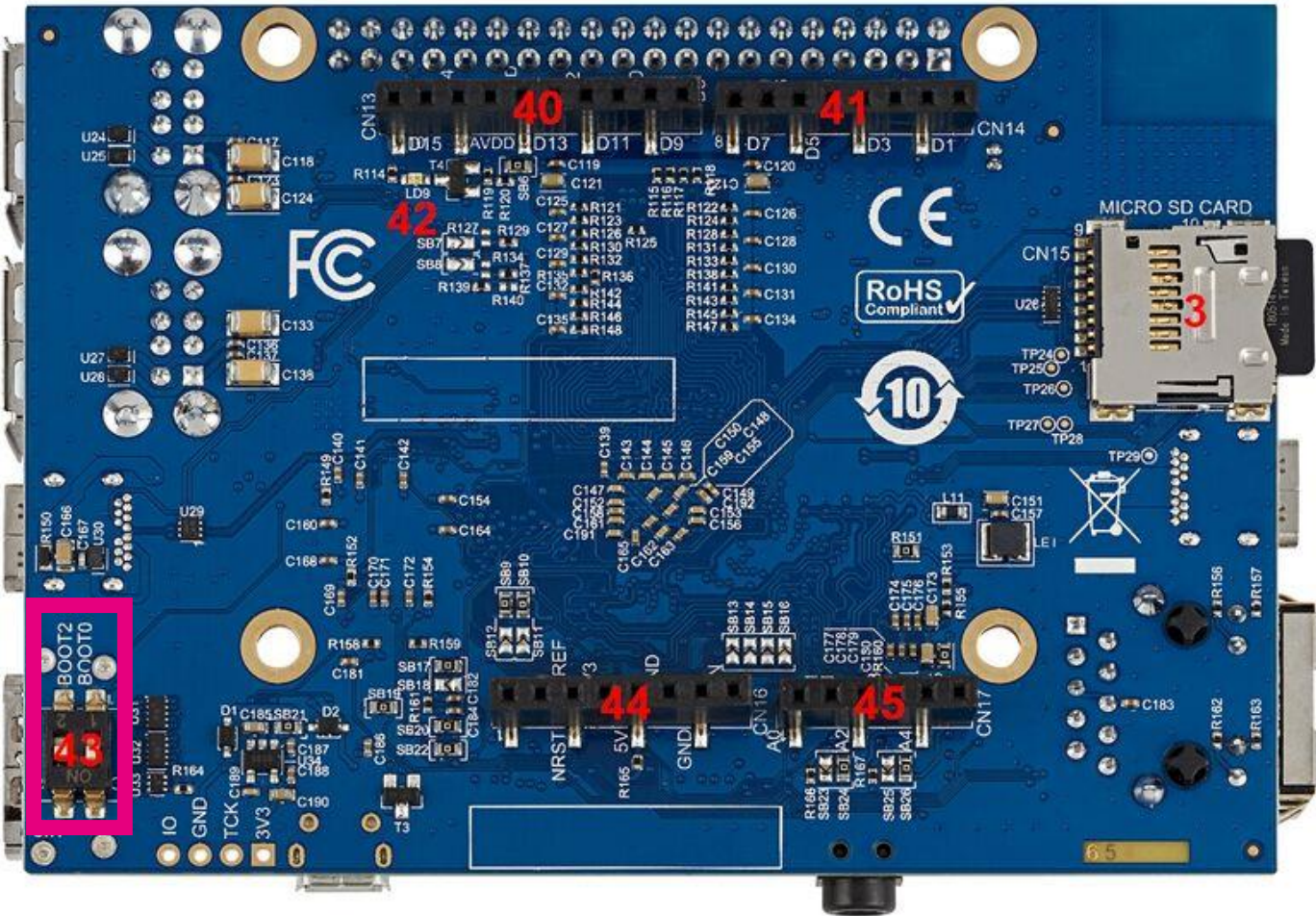
M4 production mode demo

Communicate between A7 and M4 through I2C



Production Mode

Set the board to Production mode



Boot mode	Boot 0	Boot 2
Forced USB boot for flashing	0	0
Not supported	ON	0
Engineering boot	0	ON
microSD card	ON	ON

Connect I2C2 to I2C5

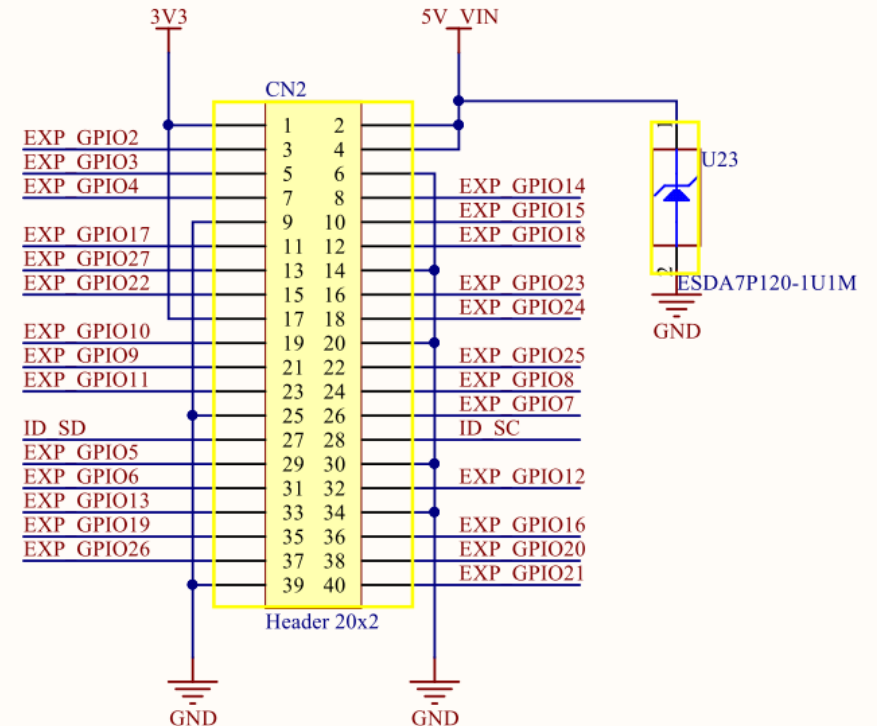
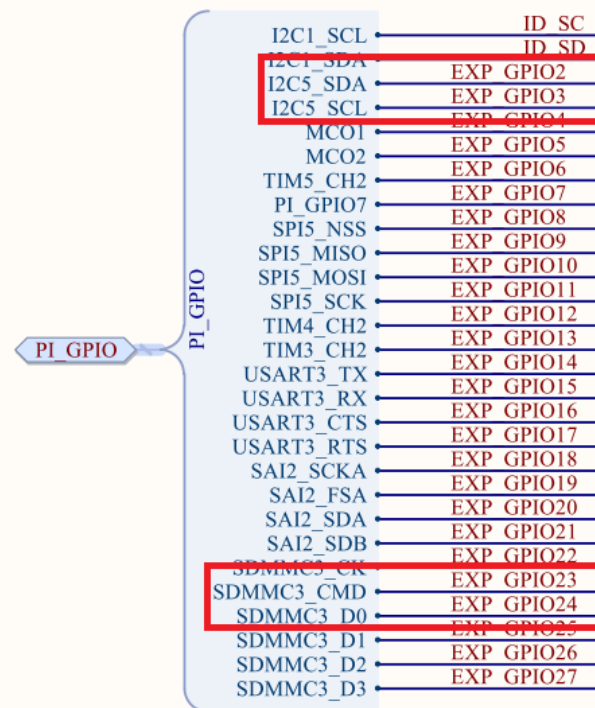
Find a pair of PINs for I2C2

Port F	PF0	-	-	-	-	I2C2_SDA
	PF1	-	-	-	-	I2C2_SCL

Find the PINs' location in CN2

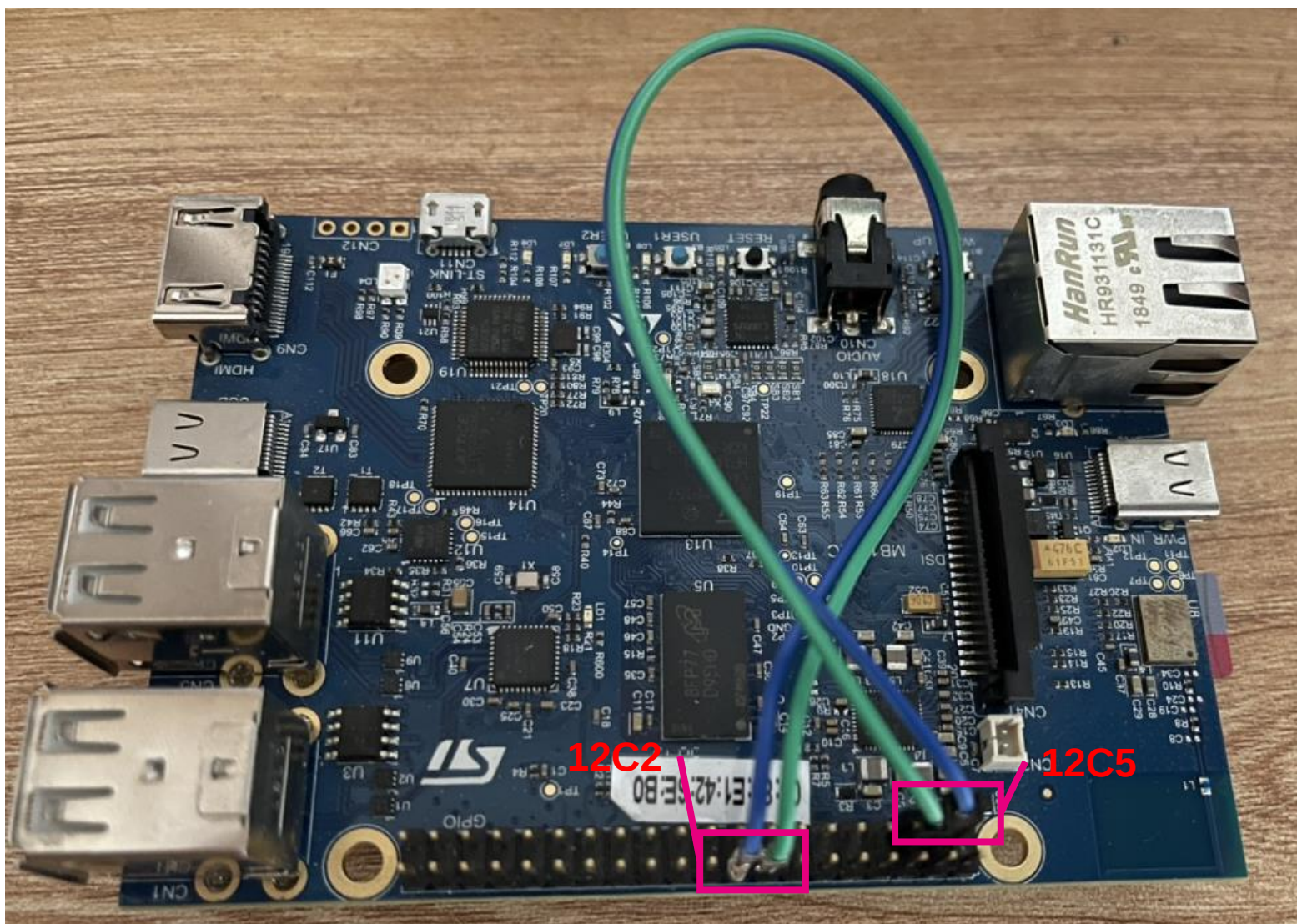
GPIO EXPANSION

SDMMC3_D0	D8	PF0
SDMMC3_CMD	A5	PF1



Connect I2C2 to I2C5

Connect I2C with I2C5



Kernel device tree modification

Assign PINs for I2C2:

```
diff --git a/arch/arm/boot/dts/stm32mp15-pinctrl.dtsi b/arch/arm/boot/dts/stm32mp15-pinctrl.dtsi
index 8035044b8c39..7437b98e0cf2 100644
--- a/arch/arm/boot/dts/stm32mp15-pinctrl.dtsi
+++ b/arch/arm/boot/dts/stm32mp15-pinctrl.dtsi
@@ -356,6 +356,23 @@
     };

+};
+
+    i2c2_pins_c: i2c2-4 {
+        pins {
+            pinmux = <STM32_PINMUX('F', 1, AF4)>, /* I2C2_SCL */
+                <STM32_PINMUX('F', 0, AF4)>; /* I2C2_SDA */
+            bias-disable;
+            drive-open-drain;
+            slew-rate = <0>;
+        };
+    };
+
+    i2c2_pins_sleep_c: i2c2-5 {
+        pins {
+            pinmux = <STM32_PINMUX('F', 1, ANALOG)>, /* I2C2_SCL */
+                <STM32_PINMUX('F', 0, ANALOG)>; /* I2C2_SDA */
+        };
+    };
+
+    i2c5_pins_a: i2c5-0 {
+        pins {
+            pinmux = <STM32_PINMUX('A', 11, AF4)>, /* I2C5_SCL */
```

Kernel device tree modification

Configure I2C2 parameters

```
diff --git a/arch/arm/boot/dts/stm32mp15xx-dkx.dtsi b/arch/arm/boot/dts/stm32mp15xx-dkx.dtsi
index 685a82161cc8..6a1ce5aa22e8 100644
--- a/arch/arm/boot/dts/stm32mp15xx-dkx.dtsi
+++ b/arch/arm/boot/dts/stm32mp15xx-dkx.dtsi
@@ -457,6 +457,19 @@
        status = "disabled";
    };

+&i2c2 {
+    pinctrl-names = "default", "sleep";
+    pinctrl-0 = <&i2c2_pins_c>;
+    pinctrl-1 = <&i2c2_pins_sleep_c>;
+    i2c-scl-rising-time-ns = <185>;
+    i2c-scl-falling-time-ns = <20>;
+    clock-frequency = <400000>;
+    /* spare dmas for other usage */
+    /delete-property/dmas;
+    /delete-property/dma-names;
+    status = "disabled";
+};
+
    &i2s2 {
        clocks = <&rcc SPI2>, <&rcc SPI2_K>, <&rcc PLL3_Q>, <&rcc PLL3_R>;
```

Kernel device tree modification

Enable I2C2 for A7 and I2C5 for M4:

```
diff --git a/arch/arm/boot/dts/stm32mp157c-dk2.dts b/arch/arm/boot/dts/stm32mp157c-dk2.dts
index ba1d15de2f2c..8c3d60d250c3 100644
--- a/arch/arm/boot/dts/stm32mp157c-dk2.dts
+++ b/arch/arm/boot/dts/stm32mp157c-dk2.dts
@@ -169,3 +169,13 @@
         vddio-supply = <&v3v3>;
     };
 };
+
+&i2c2 {
+    status = "okay";
+};
+
+&m4_i2c5 {
+    pinctrl-names = "default";
+    pinctrl-0 = <&m4_i2c5_pins_a>;
+    status = "okay";
+};
```

Rebuild the dtb and download to the board

1. Rebuild the dtb :

PC\$ cd /work/developer/linux

PC \$ makekernel dtb

2. Boot board and check I2C device on board:

Board# i2cdetect -l

3. Download the dtb to the board :

PC \$ flashkernel dtb

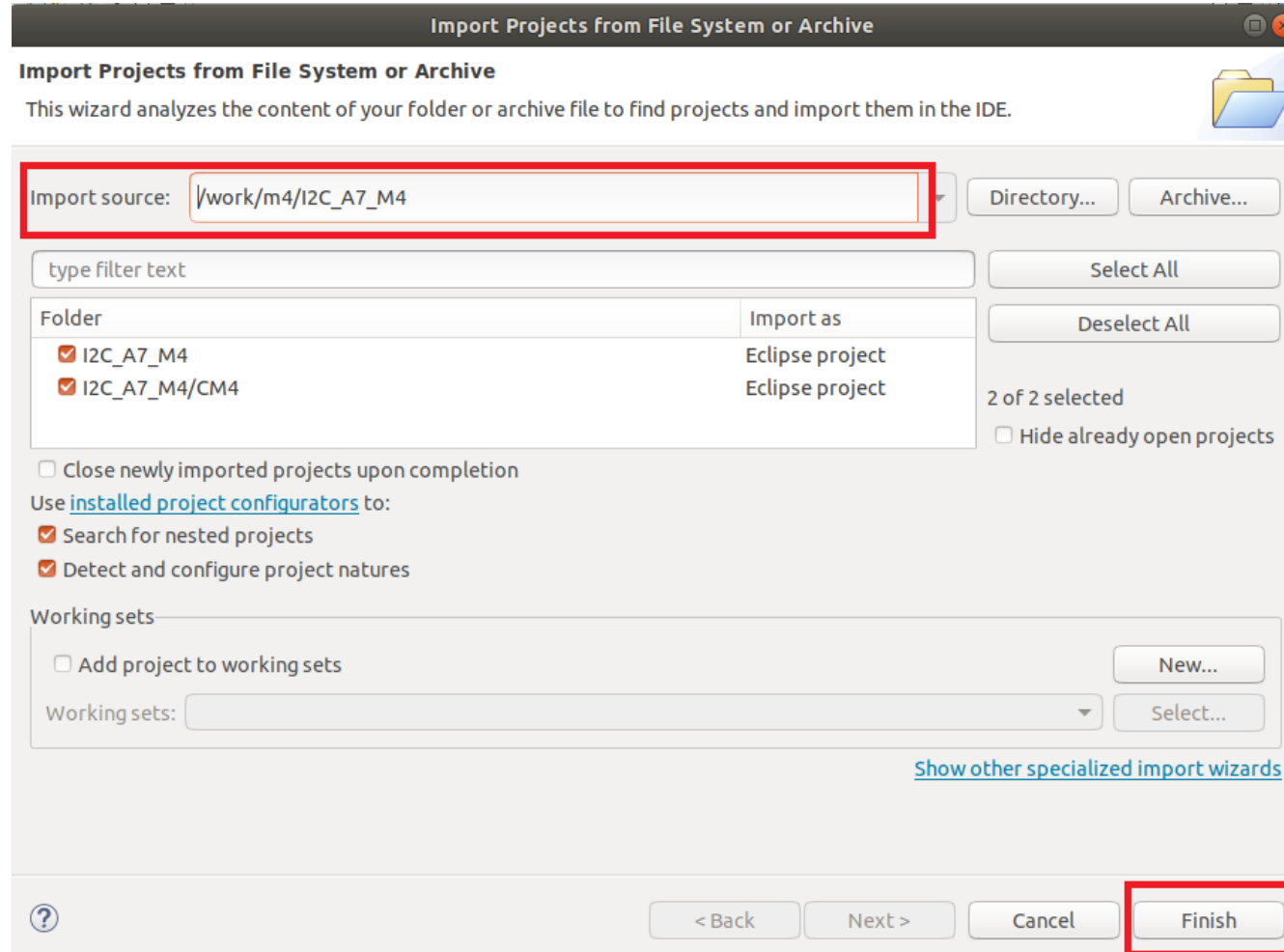
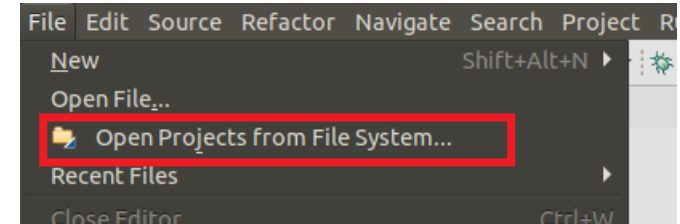
4. Reboot board and check I2C device on board again:

Board# i2cdetect -l

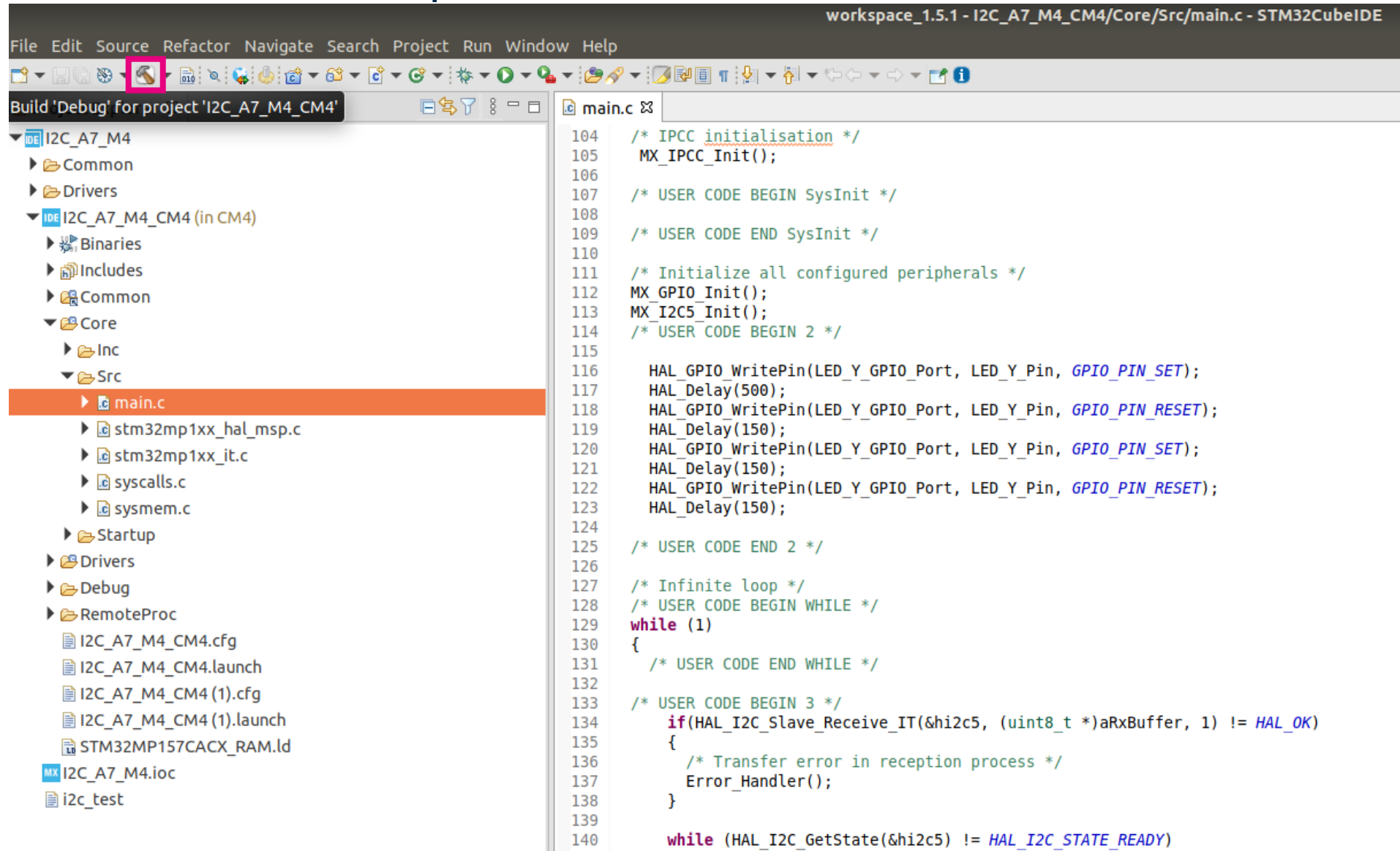
```
root@stm32mp1:~# i2cdetect -l
i2c-3    i2c                i2c-0-mux (chan_id 0)      I2C adapter
i2c-1    i2c                STM32F7 I2C(0x40013000)    I2C adapter
i2c-2    i2c                STM32F7 I2C(0x5c002000)    I2C adapter
i2c-0    i2c                STM32F7 I2C(0x40012000)    I2C adapter
```

Import the I2C_A7_M4 demo

1. Open CubeIDE, click “File”=>”Open Projects from File System”
2. Fill in the “Import source” textbox with “/work/m4/I2C_A7_M4”
3. Click “Finish”



Code modification & compilation



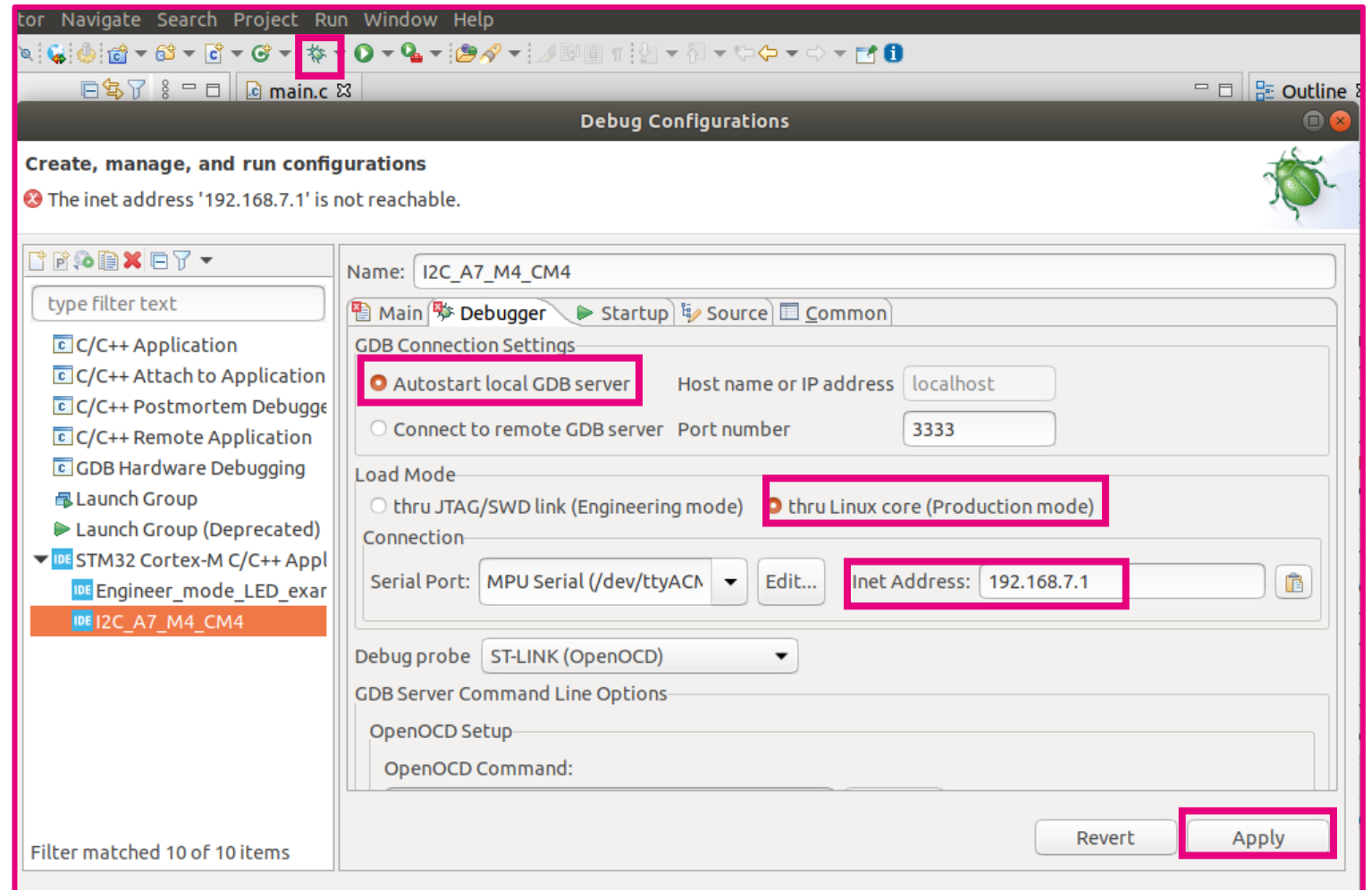
The screenshot displays the STM32CubeIDE workspace for a project named 'I2C_A7_M4_CM4'. The left sidebar shows the project structure, with the 'main.c' file selected under the 'Src' folder. The main editor window shows the code for 'main.c', which includes initialization functions for the IPCC, GPIO, and I2C5, followed by a while loop that checks the I2C5 state and handles errors. The code is written in C and uses the STM32 HAL library.

```
workspace_1.5.1 - I2C_A7_M4_CM4/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Build 'Debug' for project 'I2C_A7_M4_CM4'
I2C_A7_M4
├── Common
├── Drivers
├── I2C_A7_M4_CM4 (in CM4)
│   ├── Binaries
│   ├── Includes
│   ├── Common
│   └── Core
│       ├── Inc
│       └── Src
│           ├── main.c
│           ├── stm32mp1xx_hal_msp.c
│           ├── stm32mp1xx_it.c
│           ├── syscalls.c
│           └── systemem.c
├── Startup
├── Drivers
├── Debug
├── RemoteProc
│   ├── I2C_A7_M4_CM4.cfg
│   ├── I2C_A7_M4_CM4.launch
│   ├── I2C_A7_M4_CM4 (1).cfg
│   ├── I2C_A7_M4_CM4 (1).launch
│   └── STM32MP157CACX_RAM.ld
└── I2C_A7_M4.ioc
    └── i2c_test

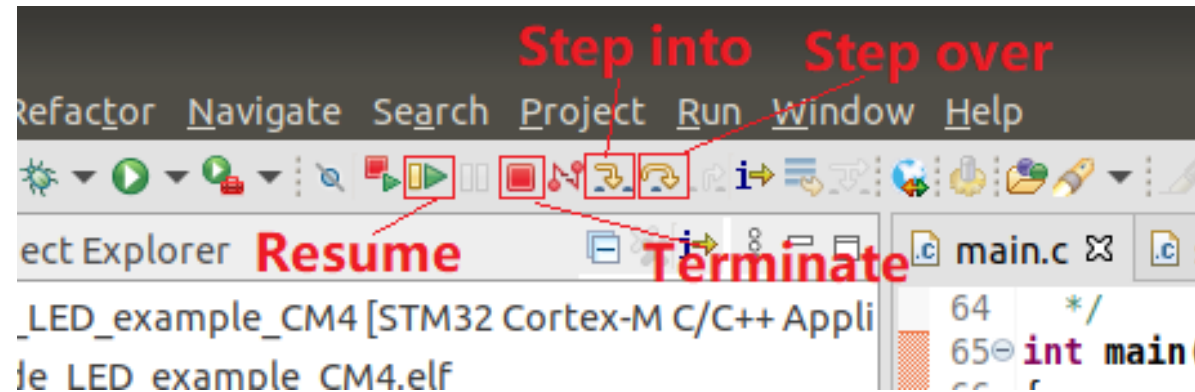
104  /* IPCC initialisation */
105  MX_IPCC_Init();
106
107  /* USER CODE BEGIN SysInit */
108
109  /* USER CODE END SysInit */
110
111  /* Initialize all configured peripherals */
112  MX_GPIO_Init();
113  MX_I2C5_Init();
114  /* USER CODE BEGIN 2 */
115
116  HAL_GPIO_WritePin(LED_Y_GPIO_Port, LED_Y_Pin, GPIO_PIN_SET);
117  HAL_Delay(500);
118  HAL_GPIO_WritePin(LED_Y_GPIO_Port, LED_Y_Pin, GPIO_PIN_RESET);
119  HAL_Delay(150);
120  HAL_GPIO_WritePin(LED_Y_GPIO_Port, LED_Y_Pin, GPIO_PIN_SET);
121  HAL_Delay(150);
122  HAL_GPIO_WritePin(LED_Y_GPIO_Port, LED_Y_Pin, GPIO_PIN_RESET);
123  HAL_Delay(150);
124
125  /* USER CODE END 2 */
126
127  /* Infinite loop */
128  /* USER CODE BEGIN WHILE */
129  while (1)
130  {
131      /* USER CODE END WHILE */
132
133      /* USER CODE BEGIN 3 */
134      if(HAL_I2C_Slave_Receive_IT(&hi2c5, (uint8_t *)aRxBuffer, 1) != HAL_OK)
135      {
136          /* Transfer error in reception process */
137          Error_Handler();
138      }
139
140      while (HAL_I2C_GetState(&hi2c5) != HAL_I2C_STATE_READY)
```

Debug configurations

1. Click the Debug tool on the toolbar to open the Debug configuration window
2. Configure the debug settings as below
3. Click Apply
4. Click the Debug tool to run debug



Exécute and Debug the code



Send the elf file to the board:

```
PC$ scp /work/m4/I2C_A7_M4/CM4/Debug/I2C_A7_M4_CM4.elf root@192.168.7.1:/lib/firmware/
```

Login into the board console:

```
PC$ ssh root@192.168.7.1
```

Stop M4 from running:

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

Set the M4 firmware name:

```
Board# echo I2C_A7_M4_CM4.elf > /sys/class/remoteproc/remoteproc0/firmware
```

Boot M4:

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

Test the demo

Send I2C message in linux console to M4:

```
root@stm32mp1:~# i2cset -f -y -a 1 0x7b 0x66 0x00
```

```
root@stm32mp1:~# i2cset -f -y -a 1 0x7b 0x66 0x05
```

```
root@stm32mp1:~# i2cset -f -y -a 1 0x7b 0x66 0x01
```

Check the board to see if the LED on/off/blink correctly.

Try to modify the code to change the phenomenon.

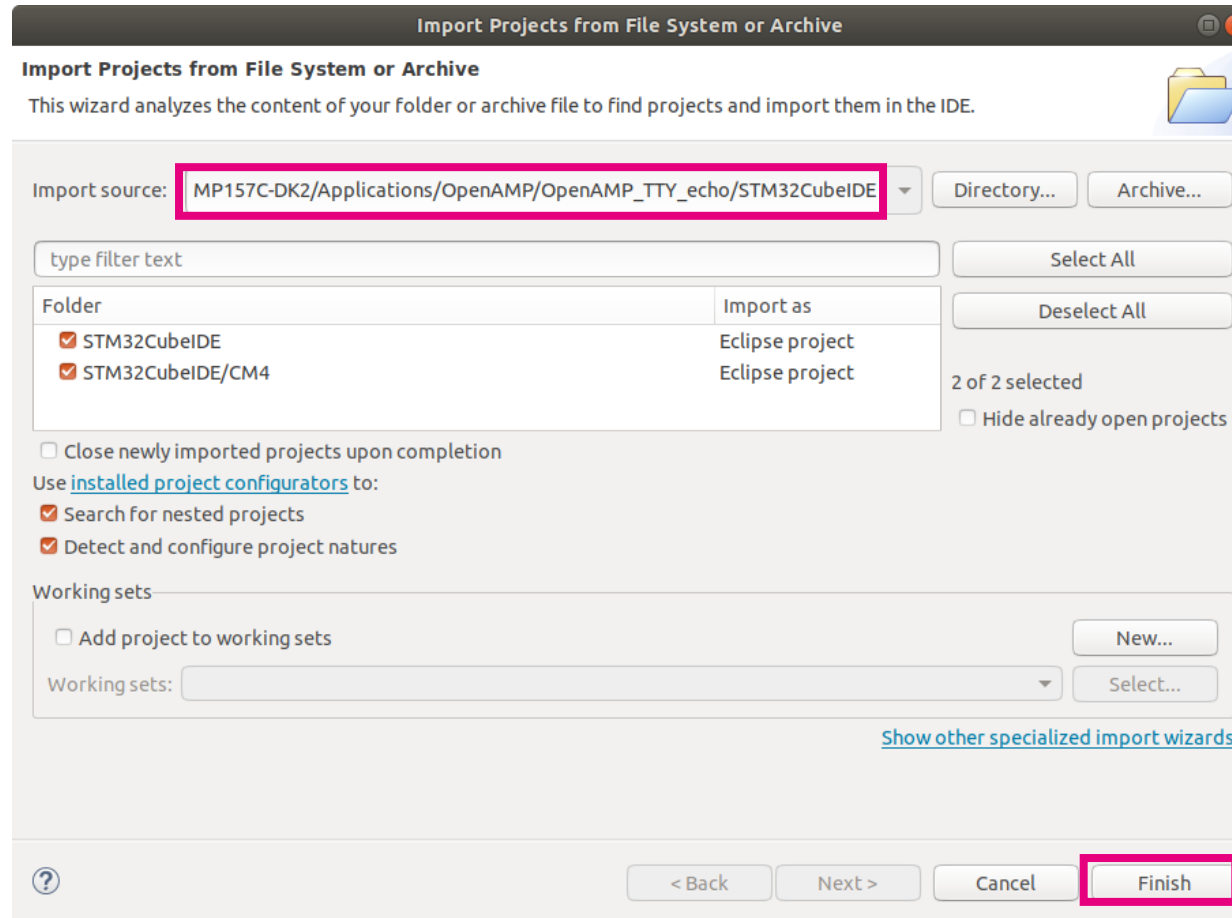
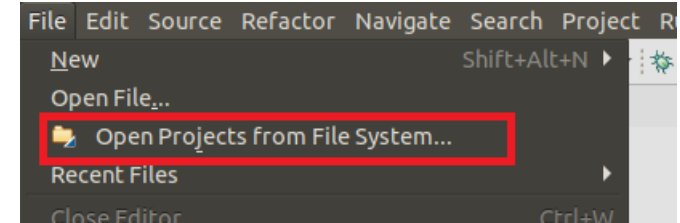
Inter-processor communication



Inter-processor communication

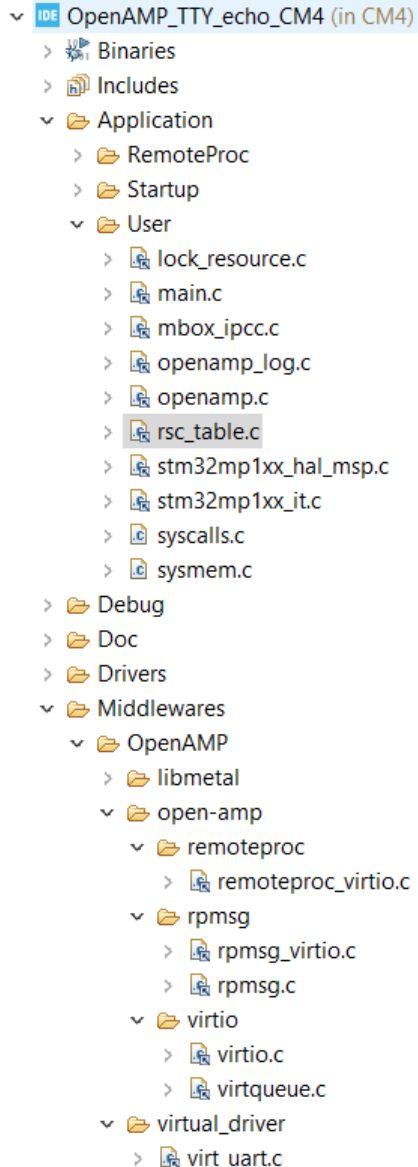
Import the OpenAMP_TTY_echo

1. Open CubeIDE, click “File”=>”Open Projects from File System”
2. Fill in the “Import source” textbox with
“/work/m4/STM32Cube_FW_MP1_V1.3.0/Projects/STM32MP157C-DK2/Applications/OpenAMP/OpenAMP_TTY_echo/STM32CubeIDE”
3. Click “Finish”



Inter-processor communication

Code modification



```
90 volatile struct shared_resource_table __resource __attribute__((used)) resource_table;
91 #else
92 CONST struct shared_resource_table __resource __attribute__((used)) resource_table = {
93 #endif
94 #elif defined(__ICCARM__)
95 __root CONST struct shared_resource_table resource_table @ ".resource_table" = {
96 #endif
97
98 #if defined(__ICCARM__) || defined (__CC_ARM) || defined (LINUX_RPROC_MASTER)
99     .version = 1,
100 #if defined (__LOG_TRACE_IO_)
101     .num = 2,
102 #else
103     .num = 1,
104 #endif
105     .reserved = {0, 0},
106     .offset = {
107         offsetof(struct shared_resource_table, vdev),
108         offsetof(struct shared_resource_table, cm_trace),
109     },
110
111     /* Virtio device entry */
112     .vdev = {
113         RSC_VDEV, VIRTIO_ID_RPMSG_, 0, RPMSG_IPU_C0_FEATURES, 0, 0, 0,
114         VRING_COUNT, {0, 0},
115     },
116
117     /* Vring rsc entry - part of vdev rsc entry */
118     .vring0 = {VRING_TX_ADDRESS, VRING_ALIGNMENT, VRING_NUM_BUFFS, VRING0_ID, 0},
119     .vring1 = {VRING_RX_ADDRESS, VRING_ALIGNMENT, VRING_NUM_BUFFS, VRING1_ID, 0},
120
121 #if defined (__LOG_TRACE_IO_)
122     .cm_trace = {
123         RSC_TRACE,
124         (uint32_t)system_log_buf, SYSTEM_TRACE_BUF_SZ, 0, "cm4_log",
125     },
126 }
```

Inter-processor communication

Load the firmware and start M4

Transfer the elf file to the board:

```
PC$ scp OpenAMP_TTY_echo_CM4.elf root@192.168.7.1:/lib/firmware/
```

Indicate the firmware name for remoteproc:

```
PC $ ssh root @192.168.7.1
```

```
Board# echo OpenAMP_TTY_echo_CM4.elf >/sys/class/remoteproc/remoteproc0/firmware
```

Execute M4 firmware:

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

Check the virtual TTY device:

```
Board# ls /dev/ttyRPMSG*
```

Send cmd by A7 and check the feedback from M4:

```
Board# cat /dev/ttyRPMSG0 &
```

```
Board# echo "test" > /dev/ttyRPMSG0
```

Exercises

In example 2, why send 0x7b 0x66 ? Change code to send 0x7b 0x33 to control LED. (Point:15)
Combined with example 1 and example 3, toggle LED after receiving virtual TTY data.(Point: 30)

Thank you