Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 2-2
CAN Loop Back (Controller Area Network, Loop Back)

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Liaw Hwee Choo, July 2022.

Hands-On CAN Loop Back

Objectives

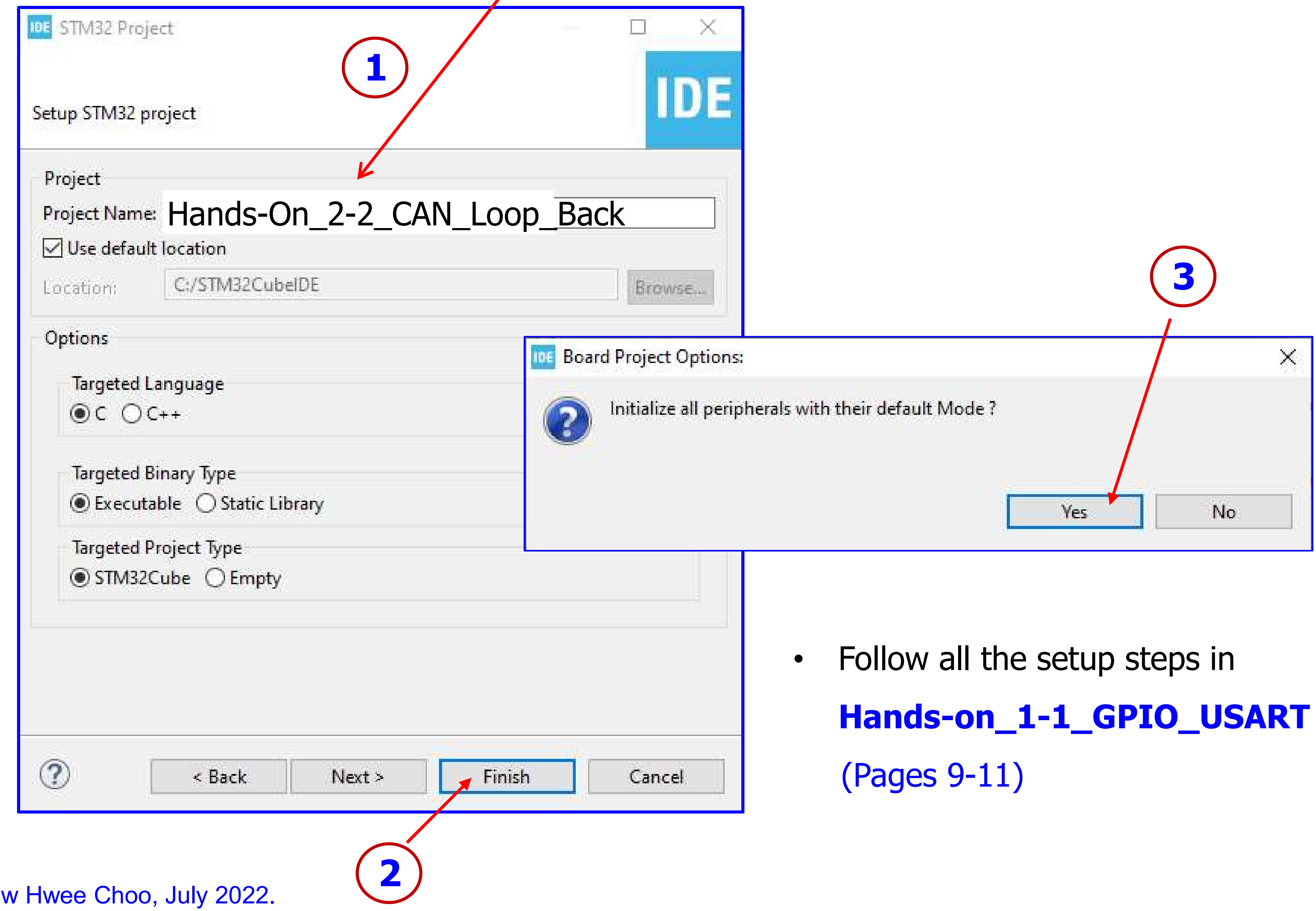
The aims of this session are to

- implement a STM32 (STM32CubeIDE) project
- set up a CAN (Controller Area Network) application development system using STM32F767 microcontroller
- develop a CAN application and program it to perform Loop Back
- test CAN program using a CAN analyzer
- build-up the development knowledge of CAN applications

  - Run STM32CubeIDE
  - Select workspace: C:\STM32_CS397
  - File -> Close All Editors
  - Start a New STM32 Project
  - Select the Nucleo-F767ZI Board

Hands-On CAN Loop Back

Enter Project Name: Hands-On_2-2_CAN_Loop_Back



STM32 Project

Setup STM32 project

Project

Project Name: Hands-On_2-2_CAN_Loop_Back

☑ Use default location

Location: C:/STM32CubeIDE

Options

Targeted Language

● C ○ C++

Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

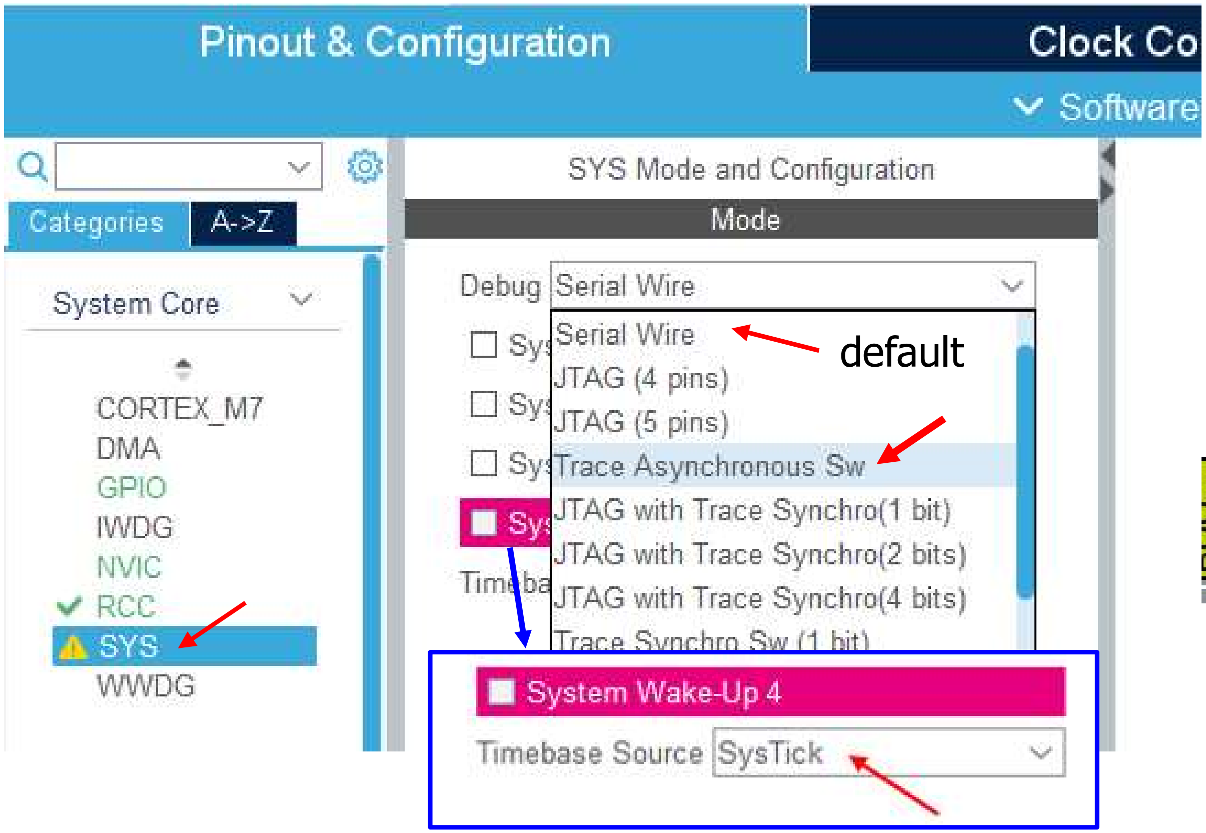
Board Project Options:

Initialize all peripherals with their default Mode ?

- Follow all the setup steps in **Hands-on_1-1_GPIO_USART** (Pages 9-11)

Hands-On CAN Loop Back

Select: System Core -> SYS -> Debug: Trace Asynchronous Sw



Pinout & Configuration

Clock Co

Software

Categories | A->Z

System Core

- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- ✓ RCC
- ⚠ SYS
- WWDG

SYS Mode and Configuration

Mode

Debug: Serial Wire

- Serial Wire ← default
- JTAG (4 pins)
- JTAG (5 pins)
- Trace Asynchronous Sw
- JTAG with Trace Synchro(1 bit)
- JTAG with Trace Synchro(2 bits)
- JTAG with Trace Synchro(4 bits)
- Trace Synchro Sw (1 bit)

☐ Sys

☐ Sys

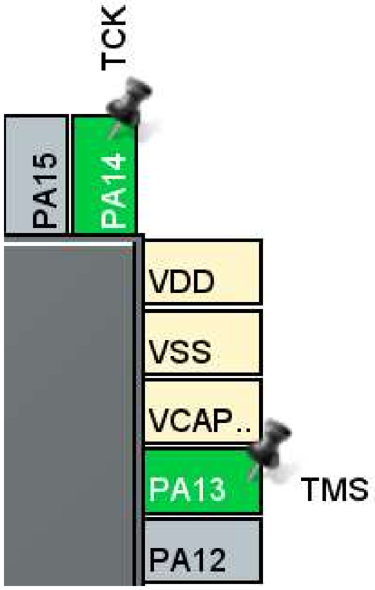
☐ Sys

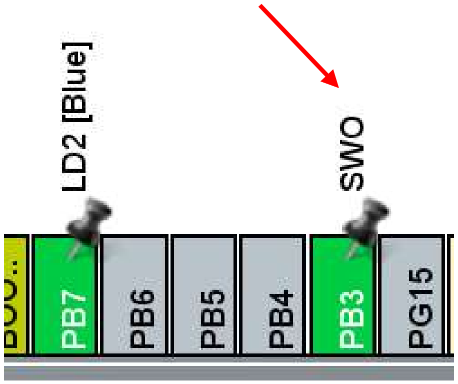
☐ Sys

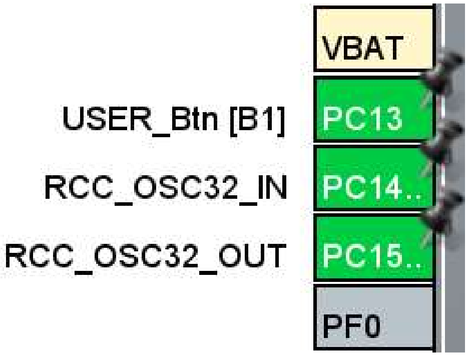
Timeba

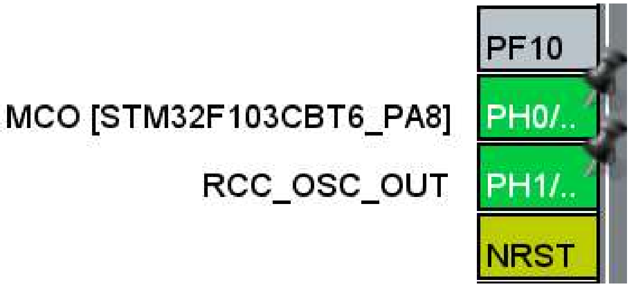
☐ System Wake-Up 4

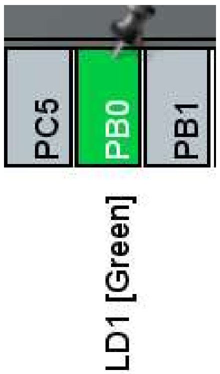
Timebase Source: SysTick

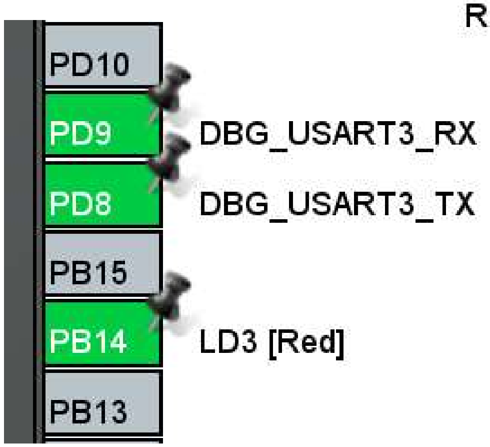










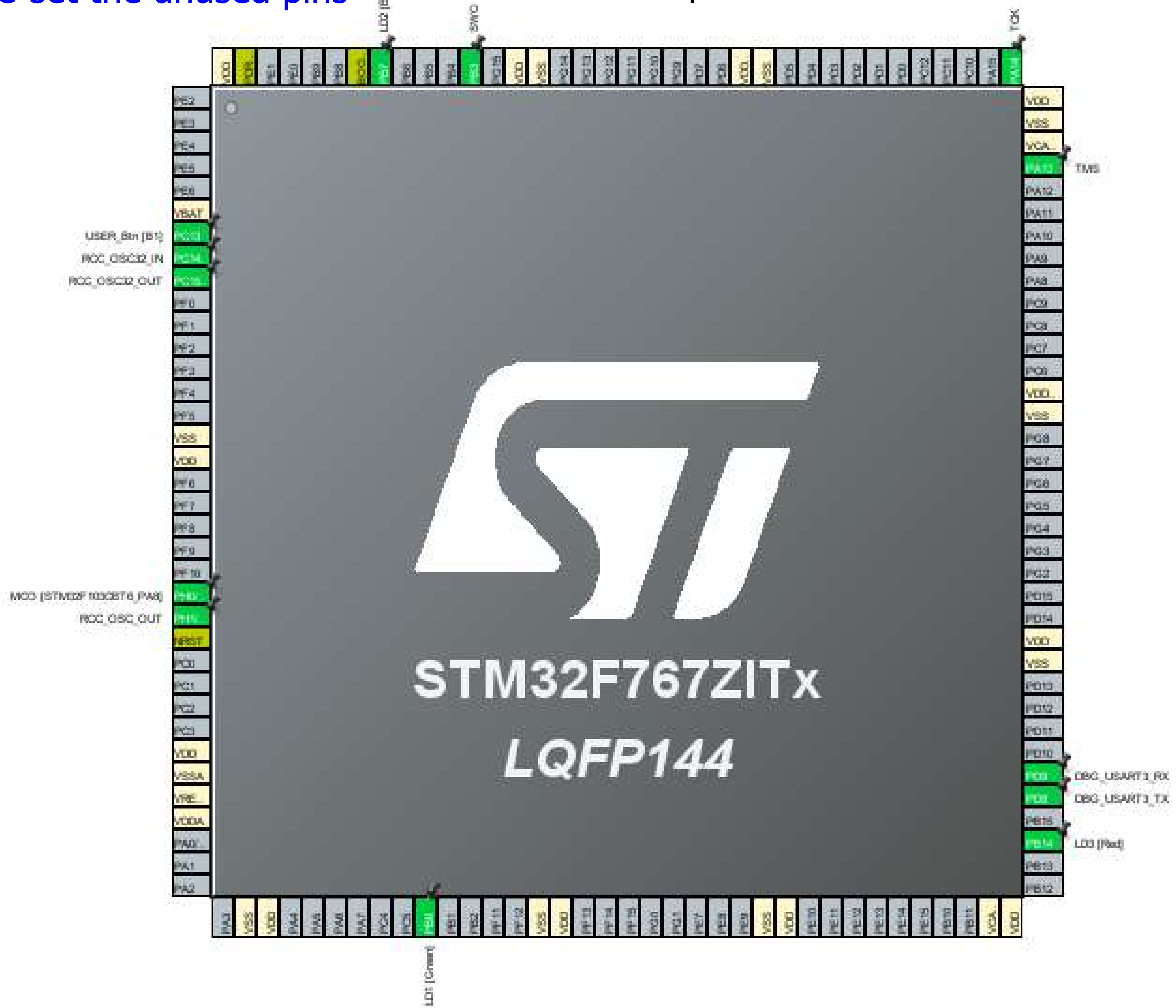


PD9: DBG_USART3_RX
PD8: DBG_USART3_TX

TCK (Test Clock)
TMS (Test Mode Select)
SWO (Single/Serial Wire Output)

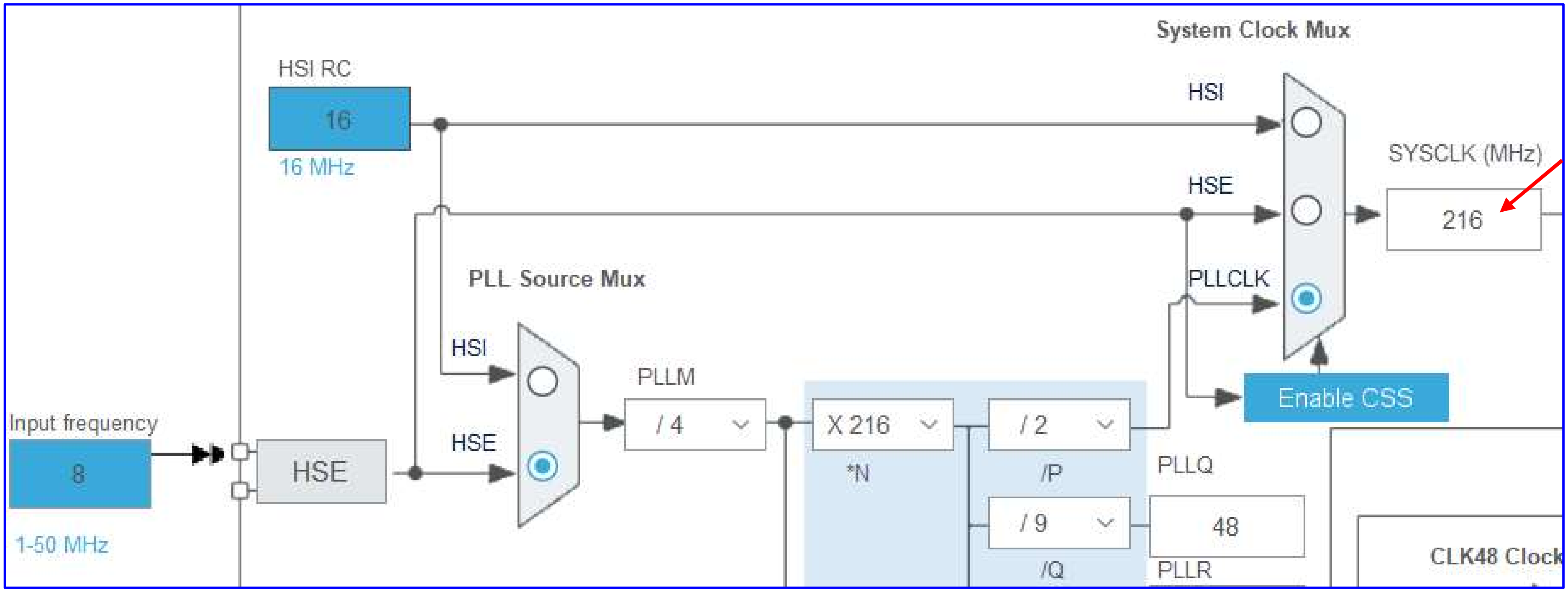
After re-set the unused pins

Hands-On CAN Loop Back



Hands-On CAN Loop Back

Clock Configuration: Use maximum frequency for clock settings



System Clock Mux

HSI RC

16

16 MHz

HSI

HSE

SYSCLK (MHz)

216

PLL Source Mux

PLLCLK

HSI

HSE

Enable CSS

PLLM

/ 4

X 216

*N

/ 2

/P

/ 9

/Q

PLLQ

48

PLLR

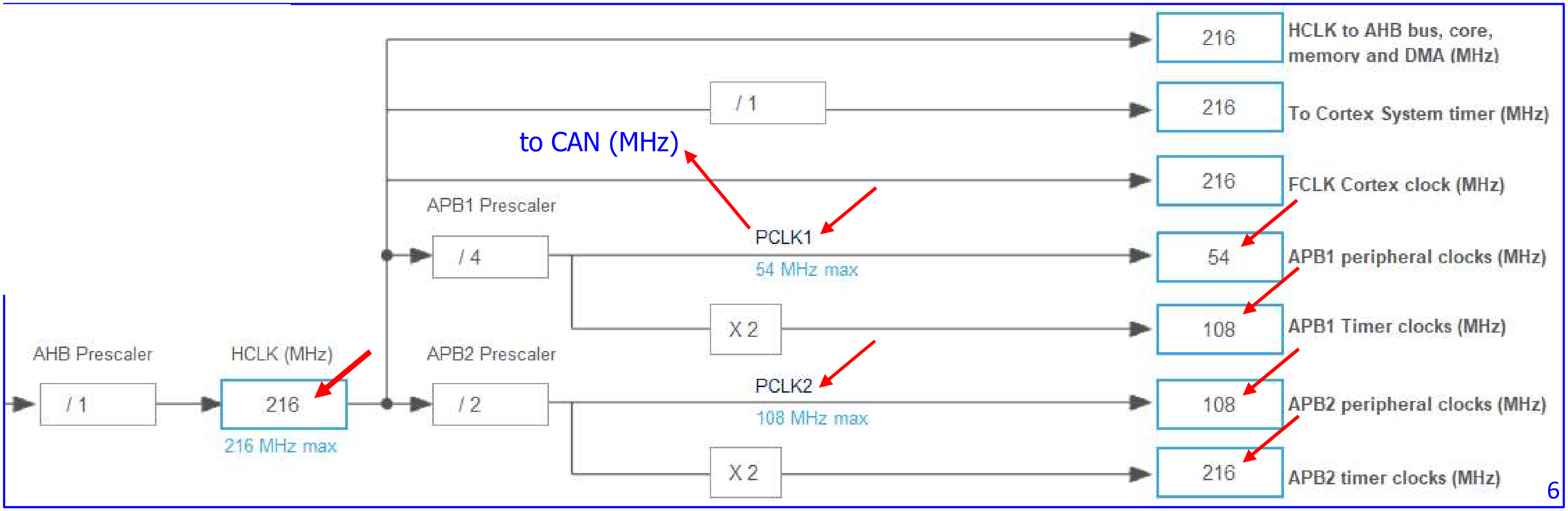
Input frequency

8

1-50 MHz

HSE

CLK48 Clock



to CAN (MHz)

AHB Prescaler

/ 1

HCLK (MHz)

216

216 MHz max

APB1 Prescaler

/ 4

PCLK1

54 MHz max

APB2 Prescaler

/ 2

PCLK2

108 MHz max

/ 1

X 2

X 2

216 HCLK to AHB bus, core, memory and DMA (MHz)

216 To Cortex System timer (MHz)

216 FCLK Cortex clock (MHz)

54 APB1 peripheral clocks (MHz)

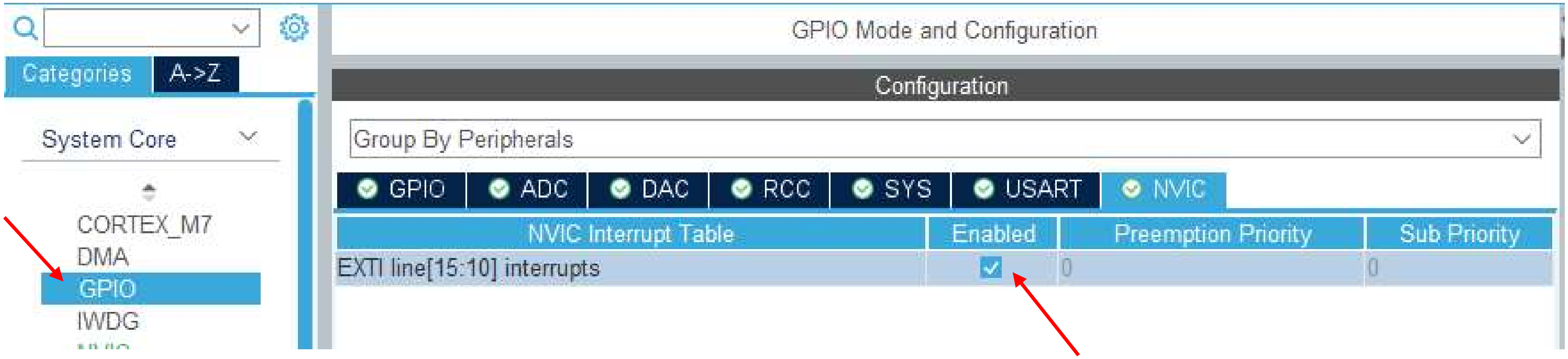
108 APB1 Timer clocks (MHz)

108 APB2 peripheral clocks (MHz)

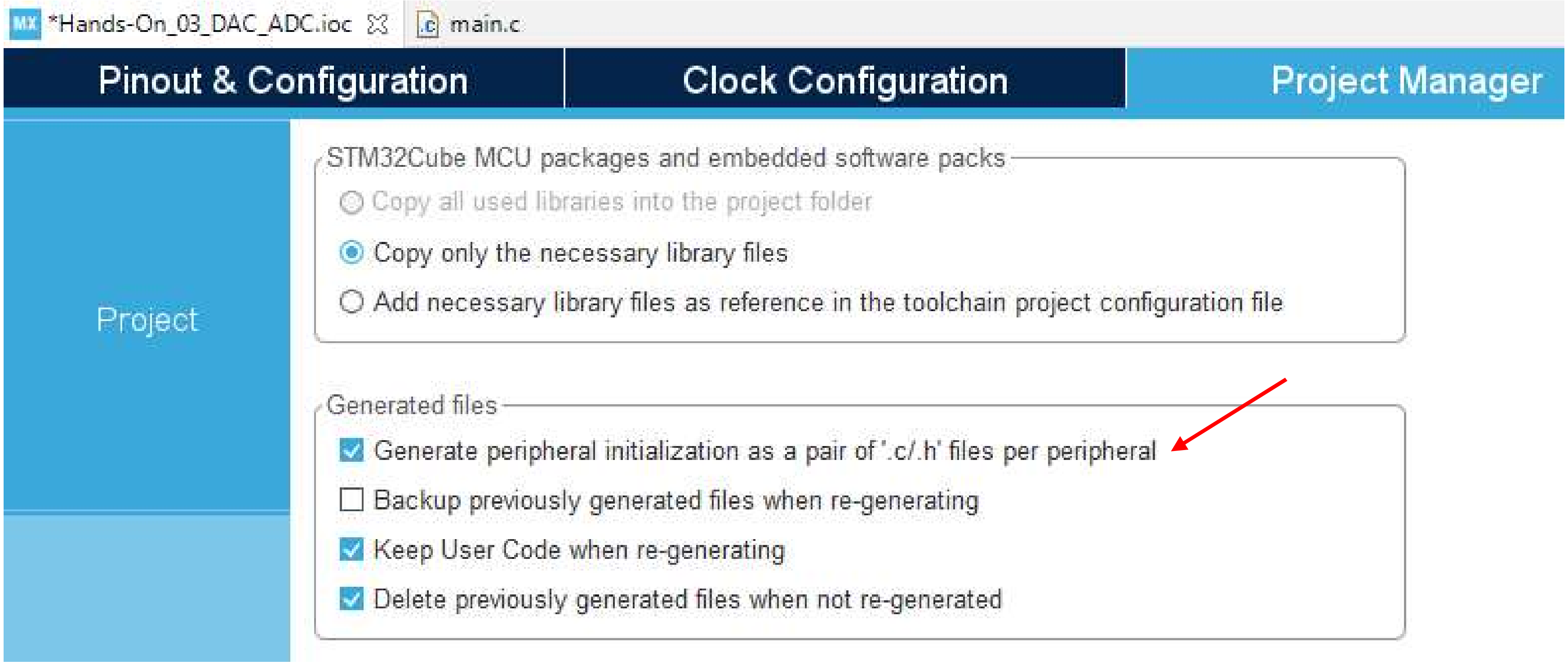
216 APB2 timer clocks (MHz)

Hands-On CAN Loop Back

- Keep default settings for LD1 [Green], LD2 [Blue], LD3 [Red], USER_Btn [B1], & USART3
- Enable Interrupt for EXTI line[15:10] for USER_Btn [B1]
- Set Project Manager – Generate … a pair of '.c/.h' files per peripheral



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line[15:10] interrupts	☑	0	0



STM32Cube MCU packages and embedded software packs

○ Copy all used libraries into the project folder

● Copy only the necessary library files

○ Add necessary library files as reference in the toolchain project configuration file

Generated files

☑ Generate peripheral initialization as a pair of '.c/.h' files per peripheral

□ Backup previously generated files when re-generating

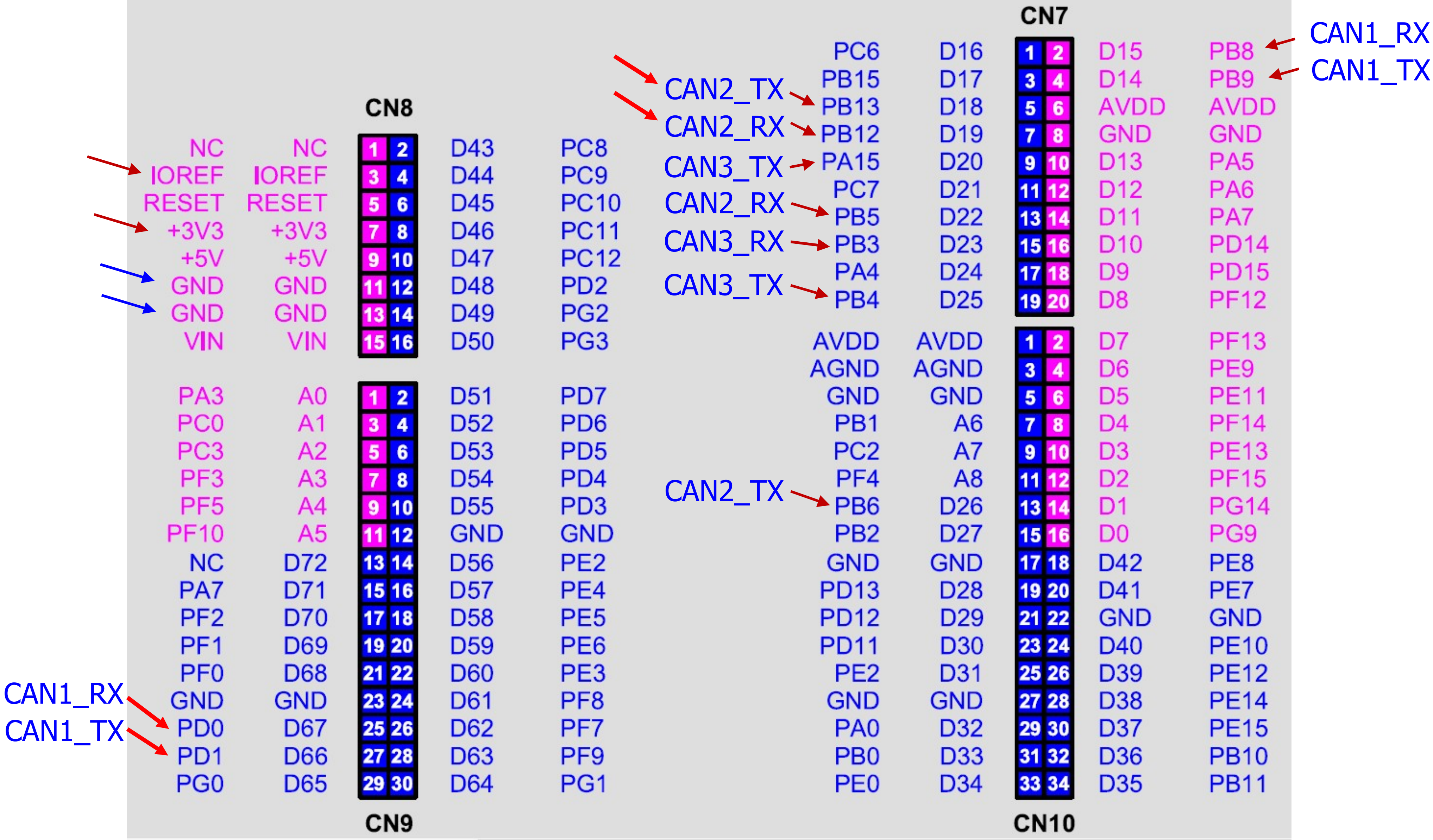
☑ Keep User Code when re-generating

☑ Delete previously generated files when not re-generated

Liaw Hwee Choo, July 2022.

Hands-On CAN Loop Back

Pinout for Controller Area Network (CAN) on ST Zio Connectors



CN8

NC	NC	1	2	D43	PC8
IOREF	IOREF	3	4	D44	PC9
RESET	RESET	5	6	D45	PC10
+3V3	+3V3	7	8	D46	PC11
+5V	+5V	9	10	D47	PC12
GND	GND	11	12	D48	PD2
GND	GND	13	14	D49	PG2
VIN	VIN	15	16	D50	PG3

	PA3	A0	1	2	D51	PD7
	PC0	A1	3	4	D52	PD6
	PC3	A2	5	6	D53	PD5
	PF3	A3	7	8	D54	PD4
	PF5	A4	9	10	D55	PD3
	PF10	A5	11	12	GND	GND
	NC	D72	13	14	D56	PE2
	PA7	D71	15	16	D57	PE4
	PF2	D70	17	18	D58	PE5
	PF1	D69	19	20	D59	PE6
	PF0	D68	21	22	D60	PE3
	GND	GND	23	24	D61	PF8
CAN1_RX	PD0	D67	25	26	D62	PF7
CAN1_TX	PD1	D66	27	28	D63	PF9
	PG0	D65	29	30	D64	PG1

CN9

CN7

	PC6	D16	1	2	D15	PB8	CAN1_RX
	PB15	D17	3	4	D14	PB9	CAN1_TX
CAN2_TX	PB13	D18	5	6	AVDD	AVDD	
CAN2_RX	PB12	D19	7	8	GND	GND	
CAN3_TX	PA15	D20	9	10	D13	PA5	
	PC7	D21	11	12	D12	PA6	
CAN2_RX	PB5	D22	13	14	D11	PA7	
CAN3_RX	PB3	D23	15	16	D10	PD14	
	PA4	D24	17	18	D9	PD15	
CAN3_TX	PB4	D25	19	20	D8	PF12	

	AVDD	AVDD	1	2	D7	PF13
	AGND	AGND	3	4	D6	PE9
	GND	GND	5	6	D5	PE11
	PB1	A6	7	8	D4	PF14
	PC2	A7	9	10	D3	PE13
	PF4	A8	11	12	D2	PF15
CAN2_TX	PB6	D26	13	14	D1	PG14
	PB2	D27	15	16	D0	PG9
	GND	GND	17	18	D42	PE8
	PD13	D28	19	20	D41	PE7
	PD12	D29	21	22	GND	GND
	PD11	D30	23	24	D40	PE10
	PE2	D31	25	26	D39	PE12
	GND	GND	27	28	D38	PE14
	PA0	D32	29	30	D37	PE15
	PB0	D33	31	32	D36	PB10
	PE0	D34	33	34	D35	PB11

CN10

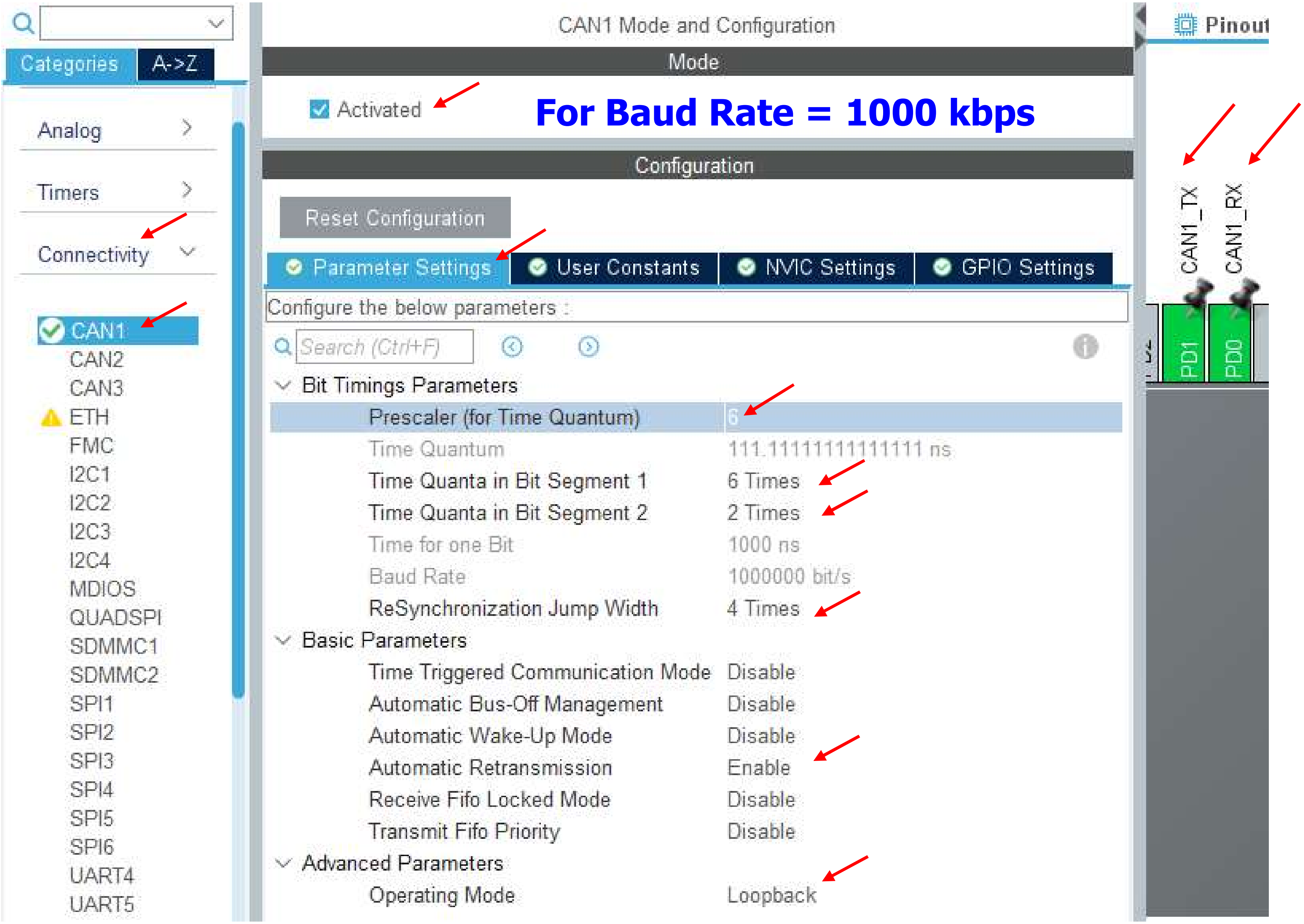
CAN1 RX : PD0
CAN1 TX : PD1

CAN2 TX : PB13
CAN2 RX : PB12

Hands-On CAN Loop Back

CAN Configuration: select CAN1, Activated, enter values (6, 6, 2, 4) as shown

Enter this value last



For Baud Rate = 1000 kbps

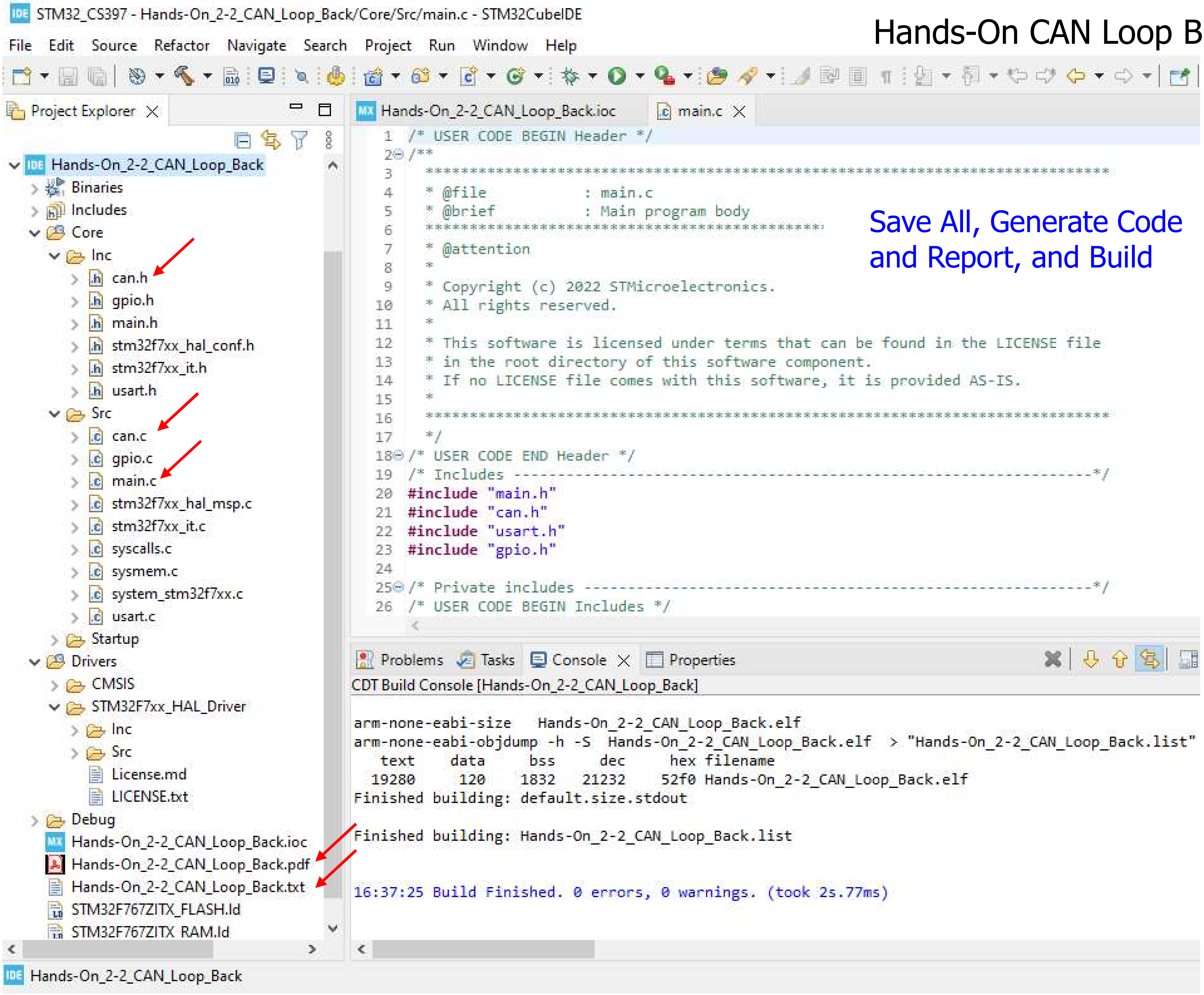
☑ Activated

Configure the below parameters :

Bit Timings Parameters	
Prescaler (for Time Quantum)	6
Time Quantum	111.11111111111111 ns
Time Quanta in Bit Segment 1	6 Times
Time Quanta in Bit Segment 2	2 Times
Time for one Bit	1000 ns
Baud Rate	1000000 bit/s
ReSynchronization Jump Width	4 Times
Basic Parameters	
Time Triggered Communication Mode	Disable
Automatic Bus-Off Management	Disable
Automatic Wake-Up Mode	Disable
Automatic Retransmission	Enable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable
Advanced Parameters	
Operating Mode	Loopback

Note: Operating mode is **Loopback**

Hands-On CAN Loop Back



```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2022 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "can.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
```

CDT Build Console [Hands-On_2-2_CAN_Loop_Back]

```
arm-none-eabi-size   Hands-On_2-2_CAN_Loop_Back.elf
arm-none-eabi-objdump -h -S  Hands-On_2-2_CAN_Loop_Back.elf  > "Hands-On_2-2_CAN_Loop_Back.list"
   text    data     bss     dec     hex filename
  19280     120    1832   21232    52f0 Hands-On_2-2_CAN_Loop_Back.elf
Finished building: default.size.stdout

Finished building: Hands-On_2-2_CAN_Loop_Back.list


16:37:25 Build Finished. 0 errors, 0 warnings. (took 2s.77ms)
```

Save All, Generate Code and Report, and Build

Hands-On CAN Loop Back

Generated **can.c**

```
/* can.c */

/* Includes */
#include "can.h"

/* USER CODE BEGIN 0 */
/* USER CODE END 0 */

CAN_HandleTypeDef hcan1;

/* CAN1 init function */
void MX_CAN1_Init(void)
{
  hcan1.Instance = CAN1;
  hcan1.Init.Prescaler = 6;
  hcan1.Init.Mode = CAN_MODE_LOOPBACK;
  hcan1.Init.SyncJumpWidth = CAN_SJW_4TQ;
  hcan1.Init.TimeSeg1 = CAN_BS1_6TQ;
  hcan1.Init.TimeSeg2 = CAN_BS2_2TQ;
  hcan1.Init.TimeTriggeredMode = DISABLE;
  hcan1.Init.AutoBusOff = DISABLE;
  hcan1.Init.AutoWakeUp = DISABLE;
  hcan1.Init.AutoRetransmission = ENABLE;
  hcan1.Init.ReceiveFifoLocked = DISABLE;
  hcan1.Init.TransmitFifoPriority = DISABLE;
  if (HAL_CAN_Init(&hcan1) != HAL_OK)
  {
    Error_Handler();
  }
}
```

```
void HAL_CAN_MspInit(CAN_HandleTypeDef* canHandle)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(canHandle->Instance==CAN1)
  {
    /* CAN1 clock enable */
    __HAL_RCC_CAN1_CLK_ENABLE();

    __HAL_RCC_GPIOD_CLK_ENABLE();
    /**CAN1 GPIO Configuration
    PD0     ------> CAN1_RX
    PD1     ------> CAN1_TX
    */
    GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF9_CAN1;
    HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
  }
}

void HAL_CAN_MspDeInit(CAN_HandleTypeDef* canHandle)
{

  if(canHandle->Instance==CAN1)
  {
     /* Peripheral clock disable */
      __HAL_RCC_CAN1_CLK_DISABLE();

     HAL_GPIO_DeInit(GPIOD, GPIO_PIN_0|GPIO_PIN_1);
   }
}
```

Hands-On CAN Loop Back

Generated **can.h**

```c
/* can.h */
/* Define to prevent recursive inclusion */
#ifndef __CAN_H__
#define __CAN_H__

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes */
#include "main.h"

/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

extern CAN_HandleTypeDef hcan1;

/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */

void MX_CAN1_Init(void);

/* USER CODE BEGIN Prototypes */
/* USER CODE END Prototypes */

#ifdef __cplusplus
}
#endif

#endif /*__CAN_H__ */
```

Hands-On CAN Loop Back

Add Code to **can.c**

```c
/* can.c */
/* USER CODE BEGIN 0 */
CAN_TxHeaderTypeDef   TxHeader;
CAN_RxHeaderTypeDef   RxHeader;
uint8_t               TxData[8] = {0};
uint8_t               RxData[8] = {0};
uint32_t              TxMailbox;
/* USER CODE END 0 */

/* USER CODE BEGIN 1 */
HAL_StatusTypeDef CAN_Polling_LoopBack(void)
{
  CAN_FilterTypeDef  sFilterConfig;

  /* #1 Configure the CAN Filter */
  sFilterConfig.FilterBank = 0;
  sFilterConfig.FilterMode = CAN_FILTERMODE_IDMASK;
  sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
  sFilterConfig.FilterIdHigh = 0x0000;
  sFilterConfig.FilterIdLow = 0x0000;
  sFilterConfig.FilterMaskIdHigh = 0x0000;
  sFilterConfig.FilterMaskIdLow = 0x0000;
  sFilterConfig.FilterFIFOAssignment = CAN_RX_FIFO0;
  sFilterConfig.FilterActivation = ENABLE;
  sFilterConfig.SlaveStartFilterBank = 14;

  if(HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig) != HAL_OK)
  {
    /* Filter configuration Error */
    Error_Handler();
  }
```

```c
 /* #2 Start the CAN peripheral */
  if (HAL_CAN_Start(&hcan1) != HAL_OK)
  {
    /* Start Error */
    Error_Handler();
  }

  /* #3 Start the Transmission process */
  TxHeader.StdId = 0x11;
  TxHeader.RTR = CAN_RTR_DATA;
  TxHeader.IDE = CAN_ID_STD;
  TxHeader.DLC = 2;
  TxHeader.TransmitGlobalTime = DISABLE;
  TxData[0] = 0xCA;
  TxData[1] = 0xFE;

  /* Request transmission */
  if(HAL_CAN_AddTxMessage(&hcan1, &TxHeader,
              TxData, &TxMailbox) != HAL_OK)
  {
    /* Transmission request Error */
    Error_Handler();
  }
```

Hands-On CAN Loop Back

Add Code to **can.c** and **can.h**

```c
 /* Wait transmission complete */
  while(HAL_CAN_GetTxMailboxesFreeLevel(&hcan1) != 3) {}

  /* #4 Start the Reception process */
  if(HAL_CAN_GetRxFifoFillLevel(&hcan1, CAN_RX_FIFO0) != 1)
  {
    /* Reception Missing */
    Error_Handler();
  }

  if(HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &RxHeader, RxData) != HAL_OK)
  {
    /* Reception Error */
    Error_Handler();
  }

  if((RxHeader.StdId != 0x11)         ||
     (RxHeader.RTR != CAN_RTR_DATA)   ||
     (RxHeader.IDE != CAN_ID_STD)     ||
     (RxHeader.DLC != 2)              ||
     ((RxData[0]<<8 | RxData[1]) != 0xCAFE))
  {
    /* Rx message Error */
    return HAL_ERROR;
  }
  return HAL_OK; /* Test Passed */
}
/* USER CODE END 1 */
```

```c
/* can.h */
/* USER CODE BEGIN Private defines */
extern CAN_TxHeaderTypeDef   TxHeader;
extern CAN_RxHeaderTypeDef   RxHeader;
extern uint8_t               TxData[8];
extern uint8_t               RxData[8];
extern uint32_t              TxMailbox;
/* USER CODE END Private defines */

/* USER CODE BEGIN Prototypes */
HAL_StatusTypeDef CAN_Polling_LoopBack(void);
/* USER CODE END Prototypes */
```

Hands-On CAN Loop Back

Add Code to **main.c**

```
/* main.c */
/* Includes */
#include "main.h"
#include "can.h"
#include "eth.h"
#include "usart.h"
#include "usb_otg.h"
#include "gpio.h"

/* Private includes */
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

/* Private function prototypes */
void SystemClock_Config(void);

int main(void)
{
  /* MCU Configuration */
  /* Reset of all peripherals,
     Initializes . . . */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();
```

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_CAN1_Init();
  MX_USART3_UART_Init();
  /* USER CODE BEGIN 2 */
  if(CAN_Polling_LoopBack() == HAL_OK)
  {
      /* OK: Turn on LED1 */
      HAL_GPIO_WritePin(GPIOB, LD1_Pin, GPIO_PIN_SET);
  }
  else
  {
      /* Not OK: Turn on LED2 */
      HAL_GPIO_WritePin(GPIOB, LD2_Pin, GPIO_PIN_SET);
  }
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
     printf("Tx1 Tx2 Rx1 Rx2 = 0x%X  0x%X  0x%X  0x%X
     \r\n", TxData[0], TxData[1], RxData[0], RxData[1]);

     HAL_Delay(1000);
     /* USER CODE END WHILE */
     /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

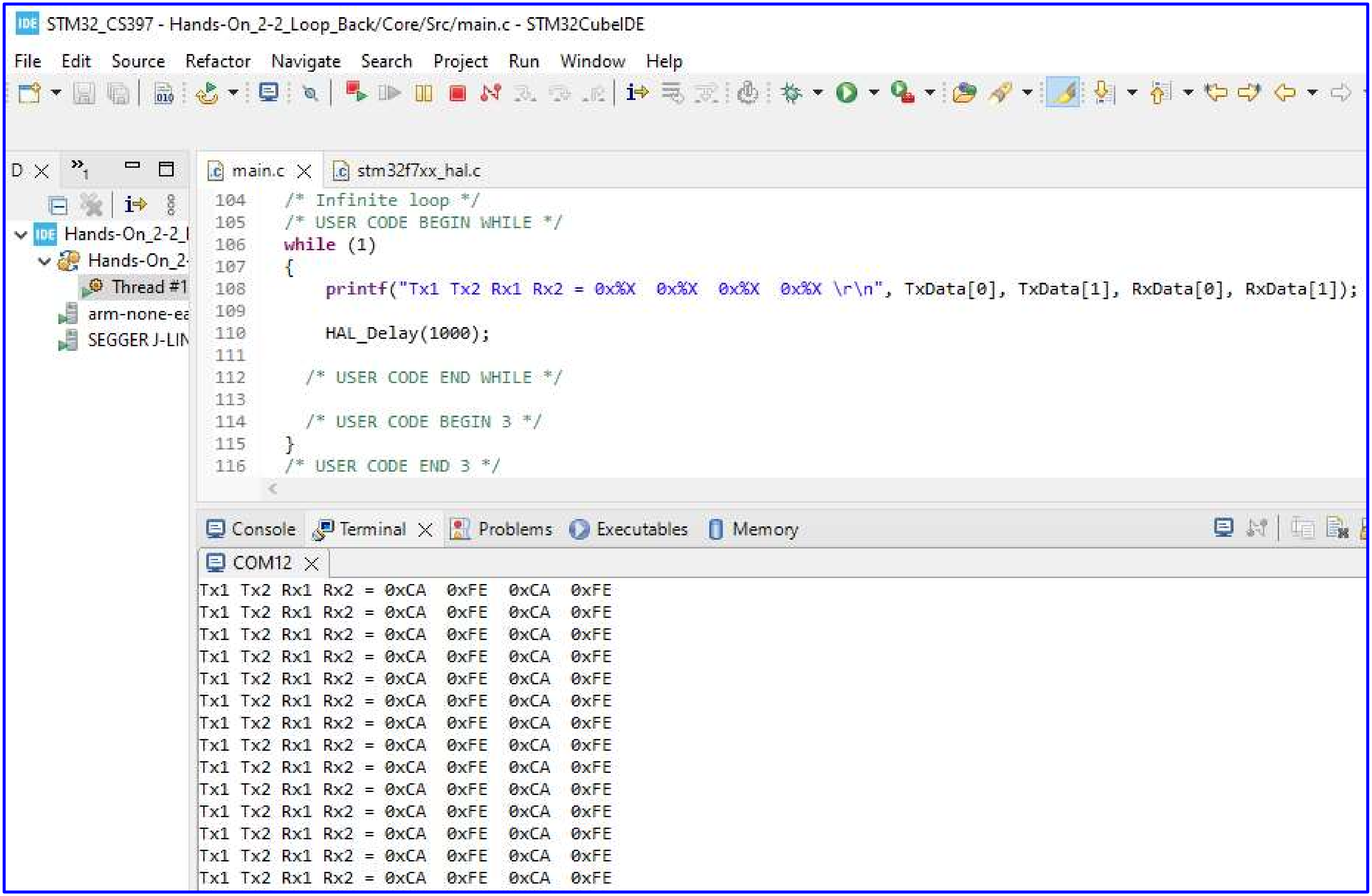
Hands-On CAN Loop Back

Add Code to **main.c**, USER CODE 4

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
    }
}

int __io_putchar(int ch)
{
    uint8_t c[1];
    c[0] = ch & 0x00FF;
    HAL_UART_Transmit(&huart3, &*c, 1, 10);
    return ch;
}

int _write(int file, char *ptr, int len)
{
    int DataIdx;
    for(DataIdx= 0; DataIdx< len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}
/* USER CODE END 4 */
```

Hands-On CAN Loop Back

Study and understand the implemented program

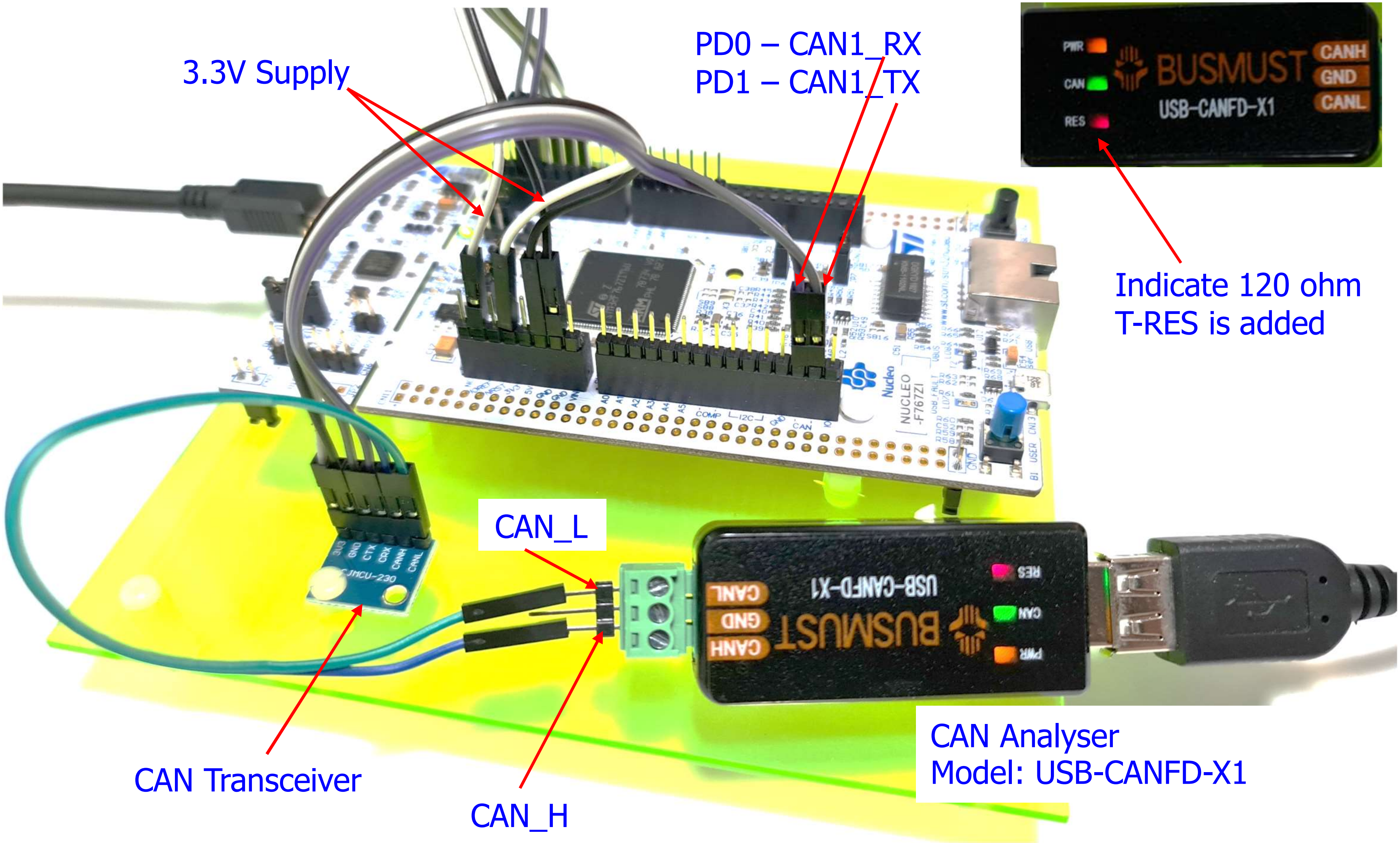


```c
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    printf("Tx1 Tx2 Rx1 Rx2 = 0x%X  0x%X  0x%X  0x%X \r\n", TxData[0], TxData[1], RxData[0], RxData[1]);

    HAL_Delay(1000);

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

```
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
Tx1 Tx2 Rx1 Rx2 = 0xCA  0xFE  0xCA  0xFE
```

Hands-On CAN Loop Back

Connect CAN Analyzer to Nucleo-F767ZI via CAN Transceiver



3.3V Supply

PD0 – CAN1_RX
PD1 – CAN1_TX

Indicate 120 ohm
T-RES is added

CAN_L

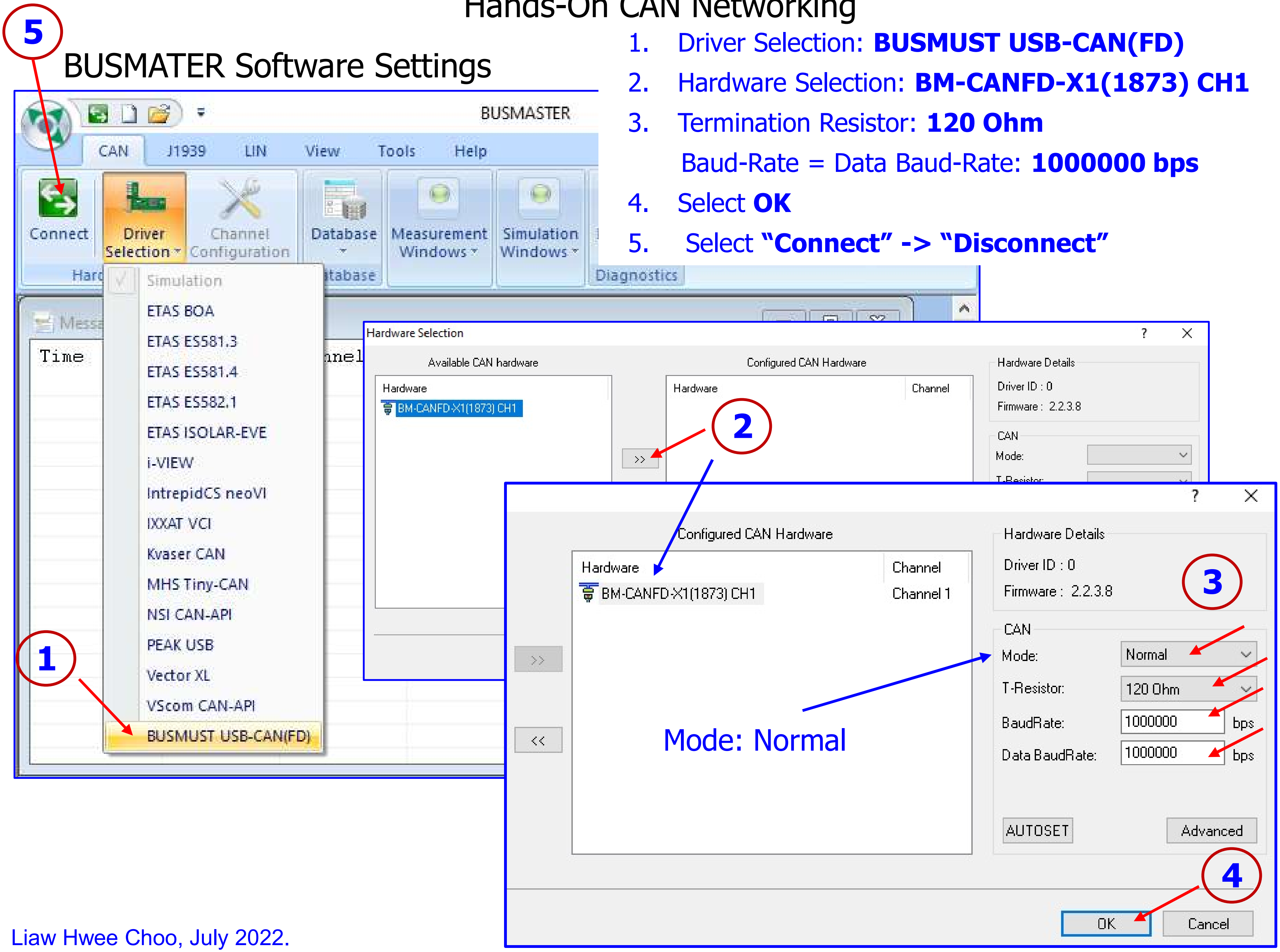
CAN Transceiver

CAN_H

CAN Analyser
Model: USB-CANFD-X1

BUSMATER Software Settings

1. Driver Selection: **BUSMUST USB-CAN(FD)**
2. Hardware Selection: **BM-CANFD-X1(1873) CH1**
3. Termination Resistor: **120 Ohm**
   Baud-Rate = Data Baud-Rate: **1000000 bps**
4. Select **OK**
5. Select **"Connect" -> "Disconnect"**



Configured CAN Hardware

Hardware	Channel
BM-CANFD-X1(1873) CH1	Channel 1

Hardware Details

Driver ID : 0

Firmware : 2.2.3.8

CAN

Mode: Normal

T-Resistor: 120 Ohm

BaudRate: 1000000 bps

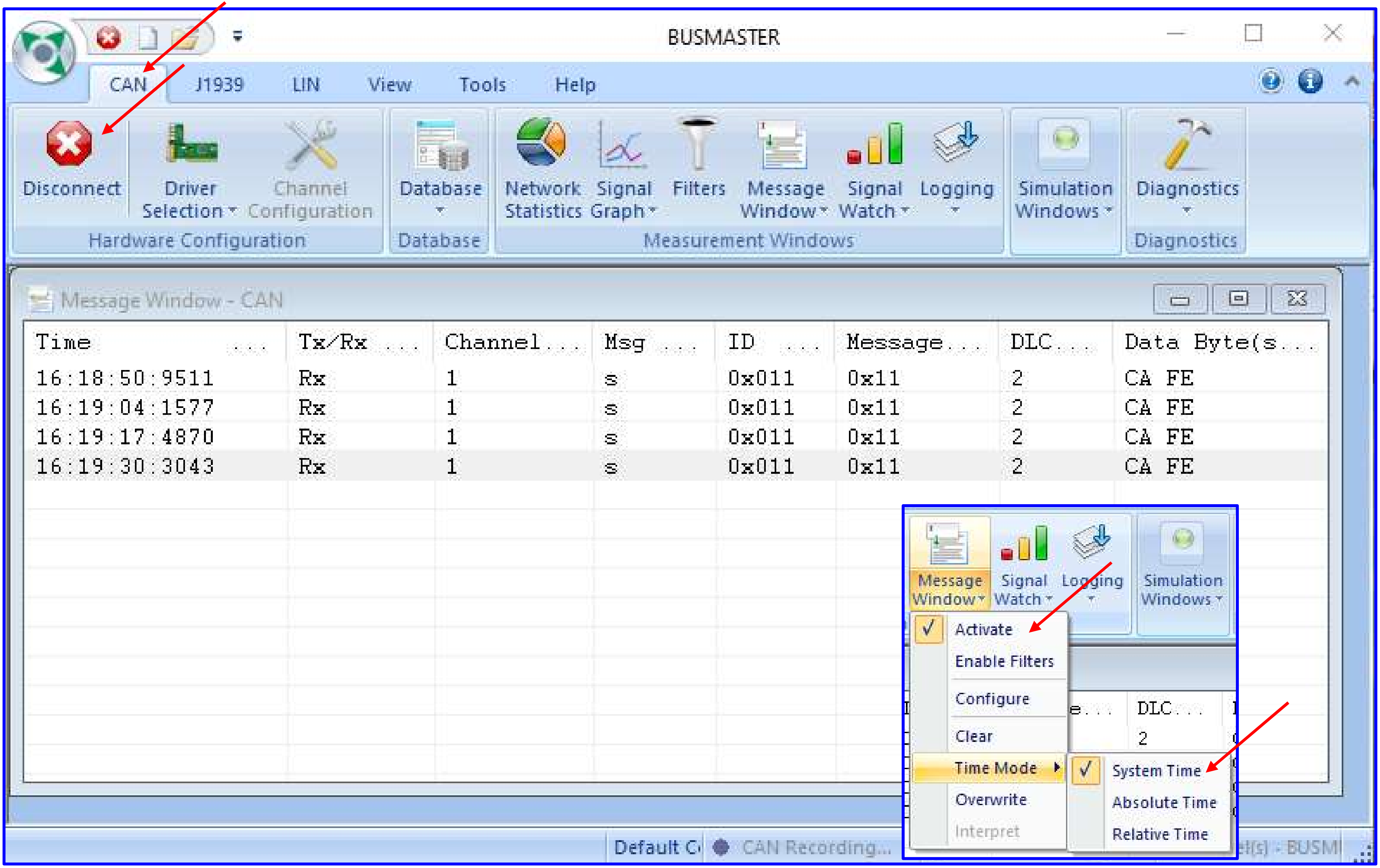
Data BaudRate: 1000000 bps

Mode: Normal

Liaw Hwee Choo, July 2022.

Hands-On CAN Loop Back

Results from CAN Analyzer (Run the program a few times)



Time	Tx/Rx	Channel	Msg	ID	Message	DLC	Data Byte(s...
16:18:50:9511	Rx	1	s	0x011	0x11	2	CA FE
16:19:04:1577	Rx	1	s	0x011	0x11	2	CA FE
16:19:17:4870	Rx	1	s	0x011	0x11	2	CA FE
16:19:30:3043	Rx	1	s	0x011	0x11	2	CA FE

- End -