Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 3-2
Dual CAN Networking
(Controller Area Network, Dual CAN Networking)

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Hands-On Dual CAN Networking

Objectives

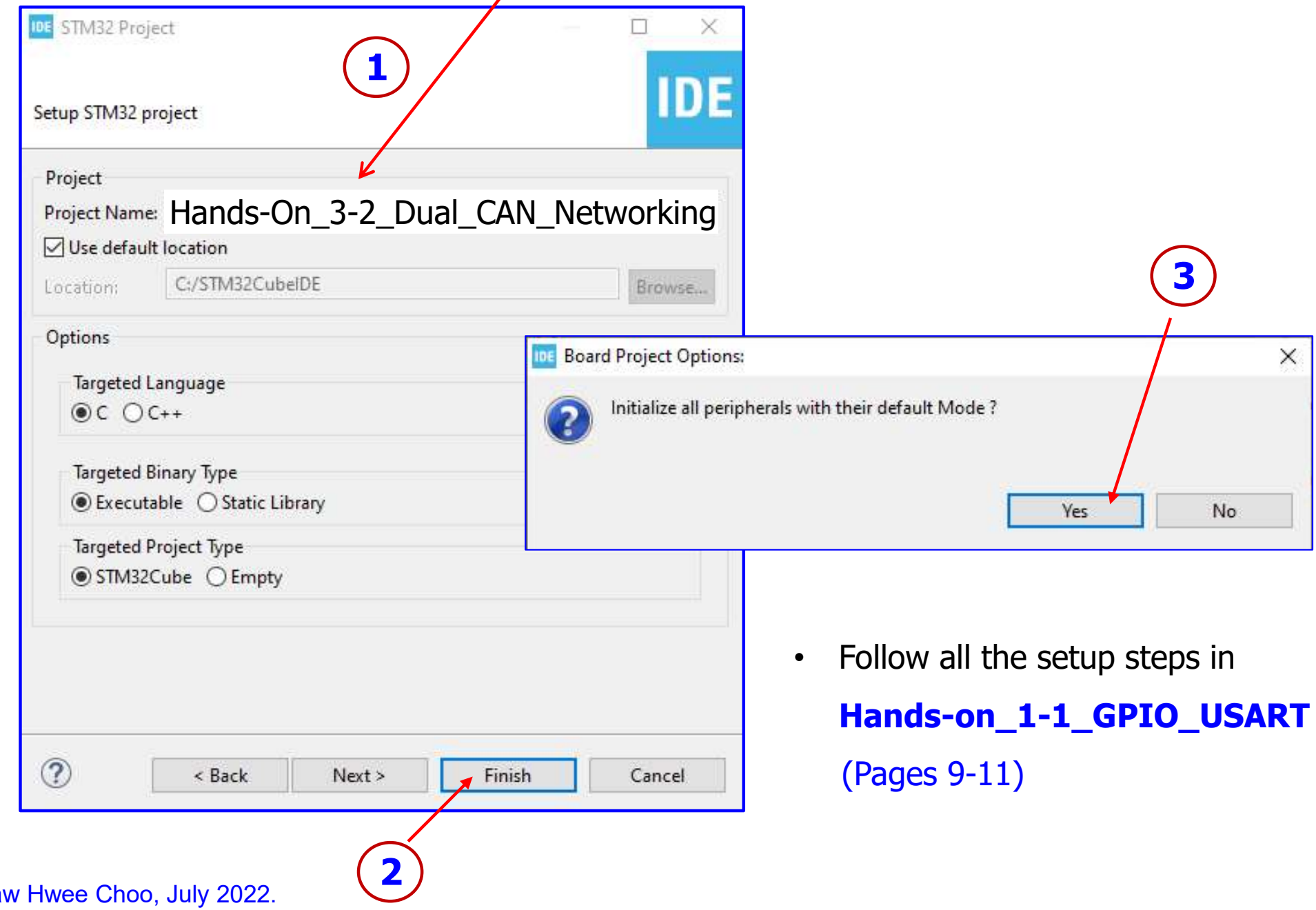
The aims of this session are to

- implement a STM32 (STM32CubeIDE) project
- set up a Dual CAN (Controller Area Network) application development system using STM32F767 microcontroller
- configure and program the dual CAN peripherals to send and receive CAN frames in normal mode
- test CAN programs using a CAN analyzer and TM terminal or RealTerm
- 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 Dual CAN Networking

Enter Project Name: Hands-On_3-2_Dual_CAN_Networking



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

Hands-On Dual CAN Networking

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

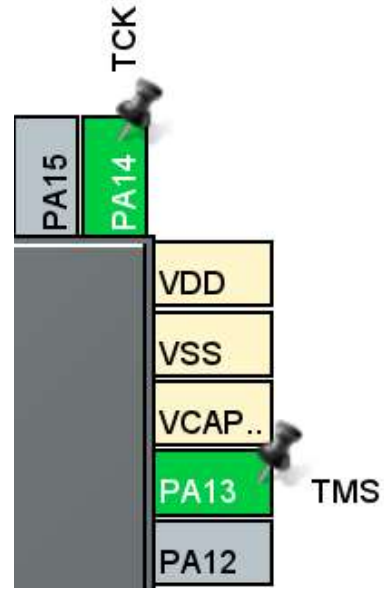
Mode

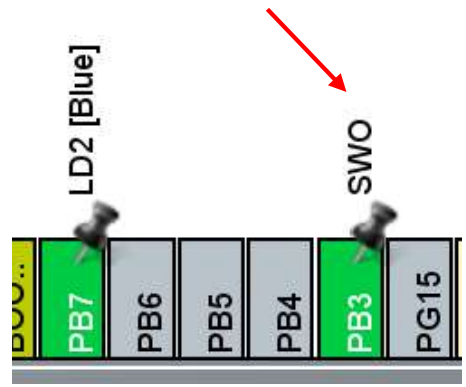
Debug: Serial Wire

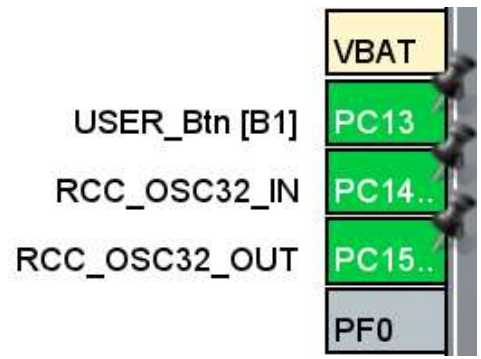
- Serial Wire (default)
- JTAG (4 pins)
- JTAG (5 pins)
- Trace Asynchronous Sw
- JTAG with Trace Synchro(1 bit)
- JTAG with Trace Synchro(2 bits)
- JTAG with Trace Synchro(4 bits)
- Trace Synchro Sw (1 bit)

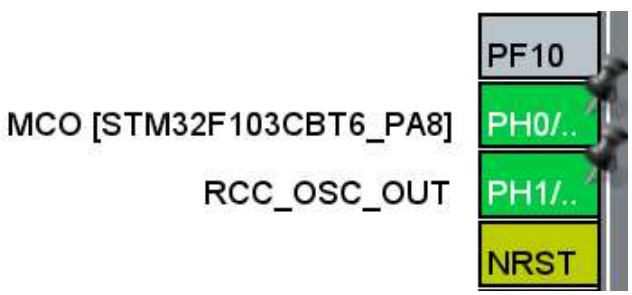
System Wake-Up 4

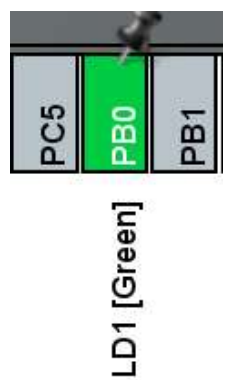
Timebase Source: SysTick

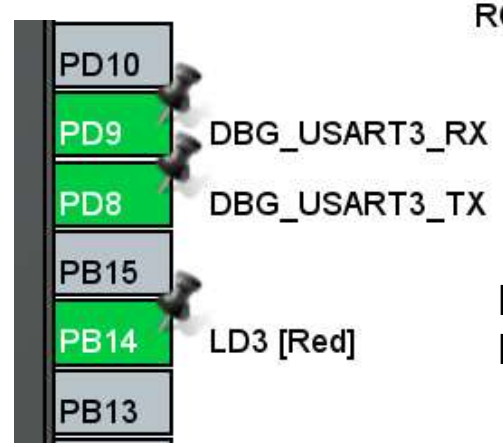












PD9: DBG_USART3_RX
PD8: DBG_USART3_TX

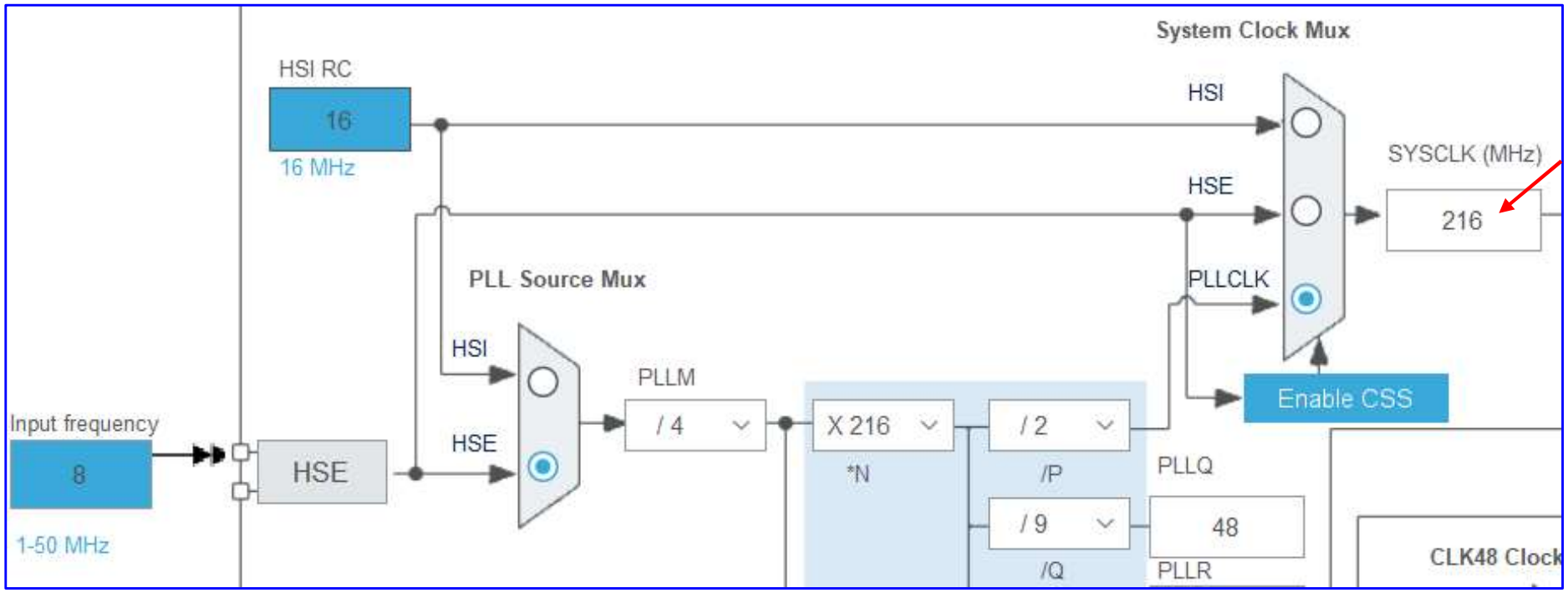
TCK (Test Clock)
TMS (Test Mode Select)
SWO (Single/Serial Wire Output)

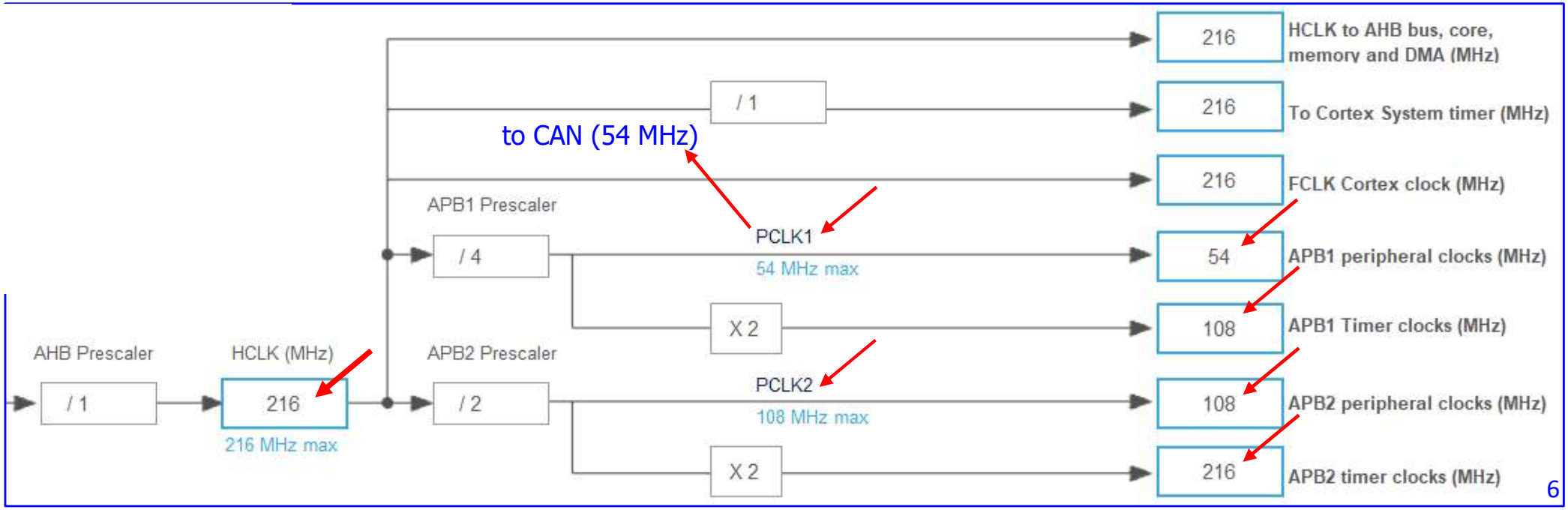
After re-set the unused pins

Hands-On CAN Loop Back

Hands-On Dual CAN Networking

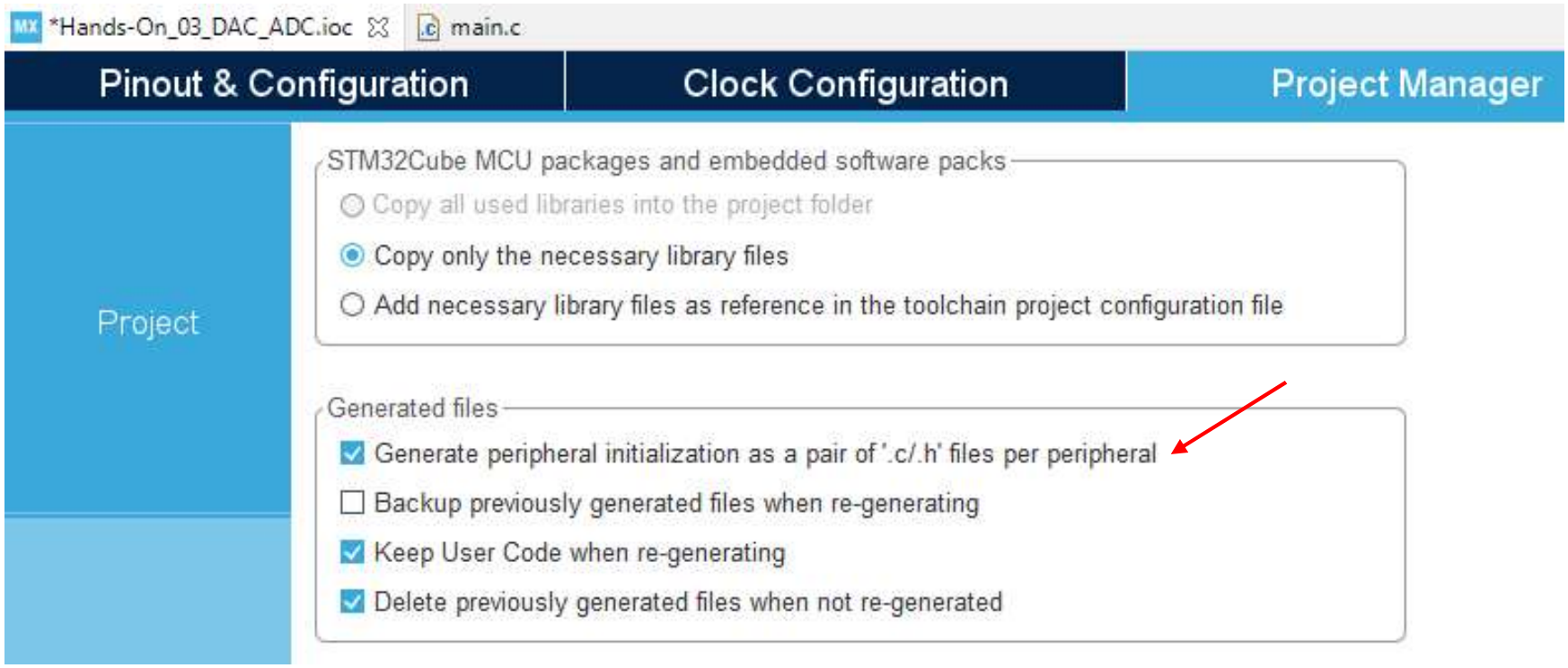
Clock Configuration: Use maximum frequency for clock settings





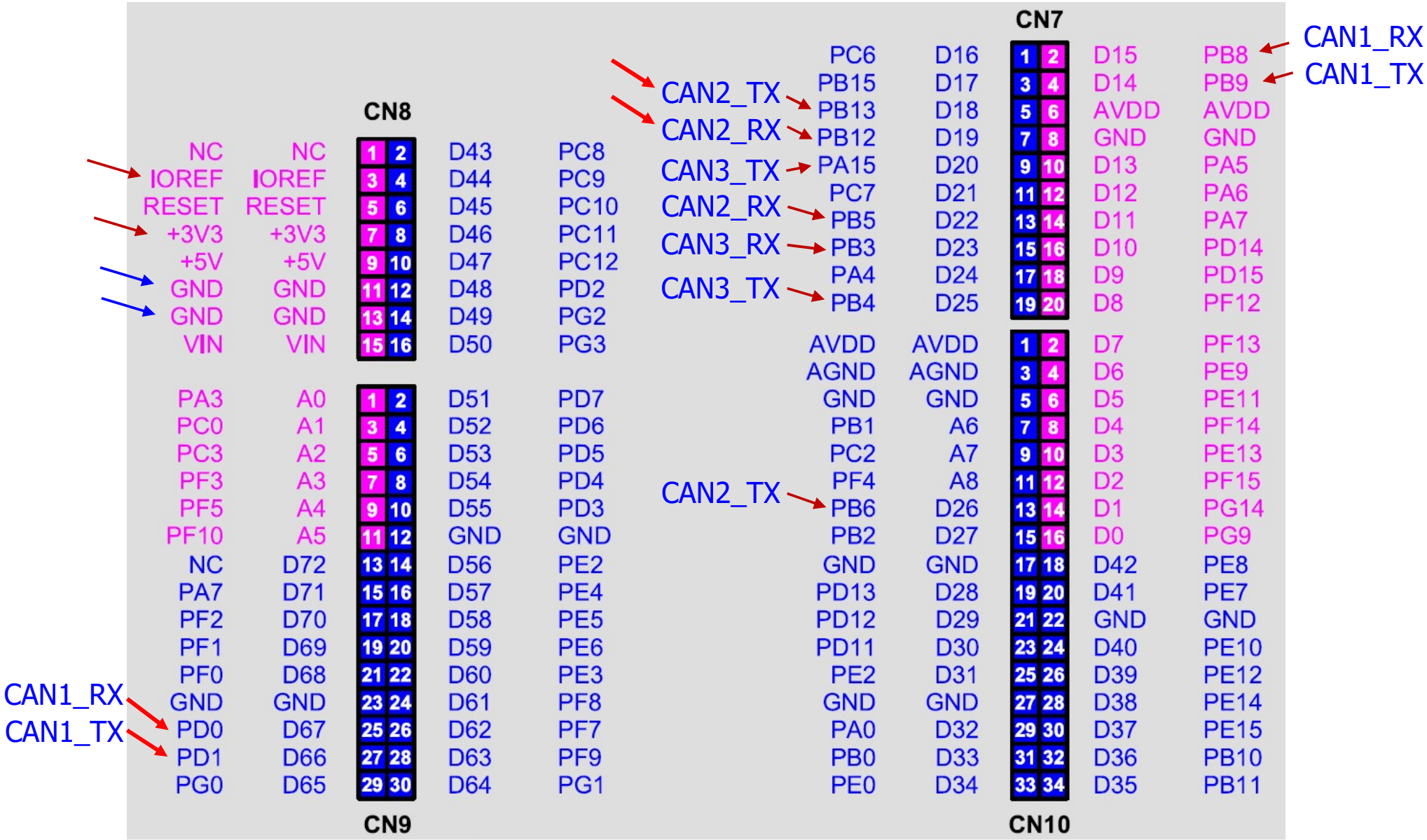
Hands-On Dual CAN Networking

- 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



Hands-On Dual CAN Networking

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



CAN1 RX : PD0
CAN1 TX : PD1

CAN2 TX : PB13
CAN2 RX : PB12

Hands-On Dual CAN Networking

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

Categories | A->Z

Connectivity

CAN1
CAN2
CAN3
ETH
FMC
I2C1
I2C2
I2C3
I2C4
MDIOS
QUADSPI
SDMMC1
SDMMC2
SPI1
SPI2
SPI3
SPI4
SPI5
SPI6
UART4
UART5
UART7
UART8
USART1
USART2
USART3
USART6

CAN1 Mode and Configuration

Mode

Activated

For Baud Rate = 1000 kbps

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

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	Normal

CAN1_TX
CAN1_RX
PD1
PD0

Note: Operating mode is **Normal**

Hands-On Dual CAN Networking

CAN Configuration: select CAN2, Activated, enter values (6, 6, 2, 4) as shown

Categories | A->Z

Connectivity

- CAN1
- CAN2
- CAN3
- ETH
- FMC
- I2C1
- I2C2
- I2C3
- I2C4
- MDIOS
- QUADSPI
- SDMMC1
- SDMMC2
- SPI1
- SPI2
- SPI3
- SPI4
- SPI5
- SPI6
- UART4
- UART5
- UART7
- UART8
- USART1
- USART2
- USART3
- USART6

CAN2 Mode and Configuration

Mode

Activated

For Baud Rate = 1000 kbps

Configuration

Reset Configuration

NVIC Settings | GPIO Settings

Parameter Settings | User Constants

Configure the below parameters :

Search (Ctrl+F)

Bit Timings Parameters

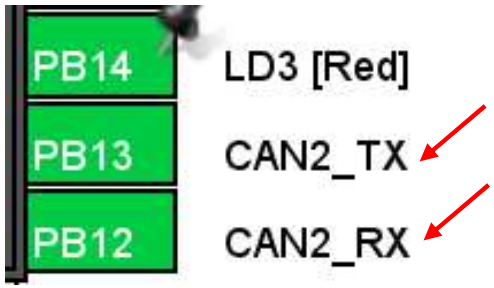
Prescaler (for Time Quantum)	6
Time Quantum	111.111111111111111 ns
Time Quanta in Bit Segment 1	6 Times
Time Quanta in Bit Segment 2	2 Times
Time for one Bit	1000 ns
Baud Rate	1000000 bit/s
ReSynchronization Jump Width	4 Times

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	Normal



Note: Operating mode is **Normal**

Hands-On Dual CAN Networking

CAN1 & CAN2 Configurations: GPIO Settings

Analog
Timers
Connect...
CAN1
CAN2
CAN3
ETH

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | GPIO Settings

Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin Name	Signal on Pin	GPIO out...	GPIO mode	GPIO Pull-up/Pull-down	Maximum out...	Fast ...	User ...	Modif...
PD0	CAN1_RX	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	n/a		☐
PD1	CAN1_TX	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	n/a		☐

Analog
Timers
Connect...
CAN1
CAN2
CAN3
ETH
FMC

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | GPIO Settings

Search Signals

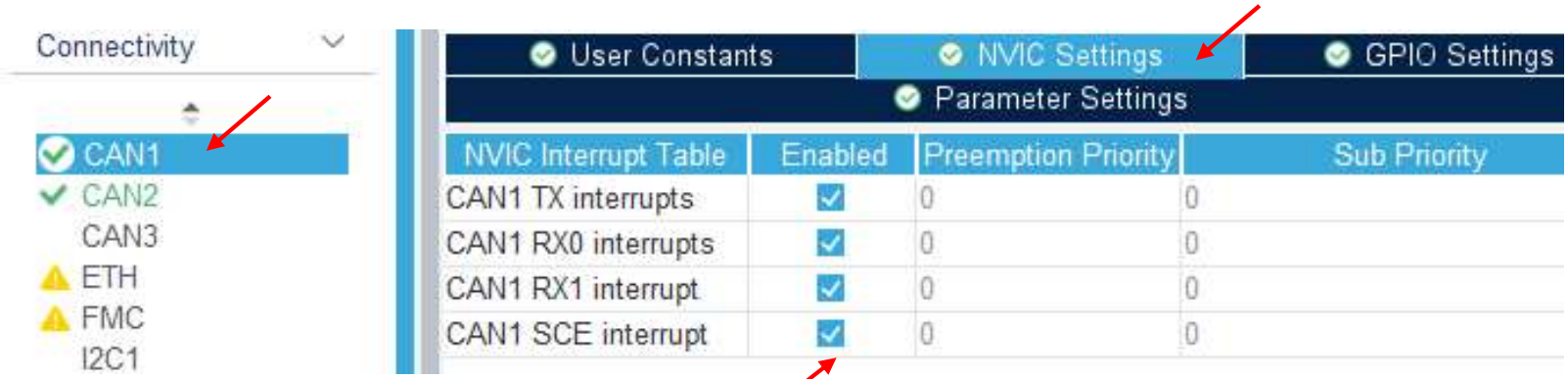
Search (Ctrl+F)

Show only Modified Pins

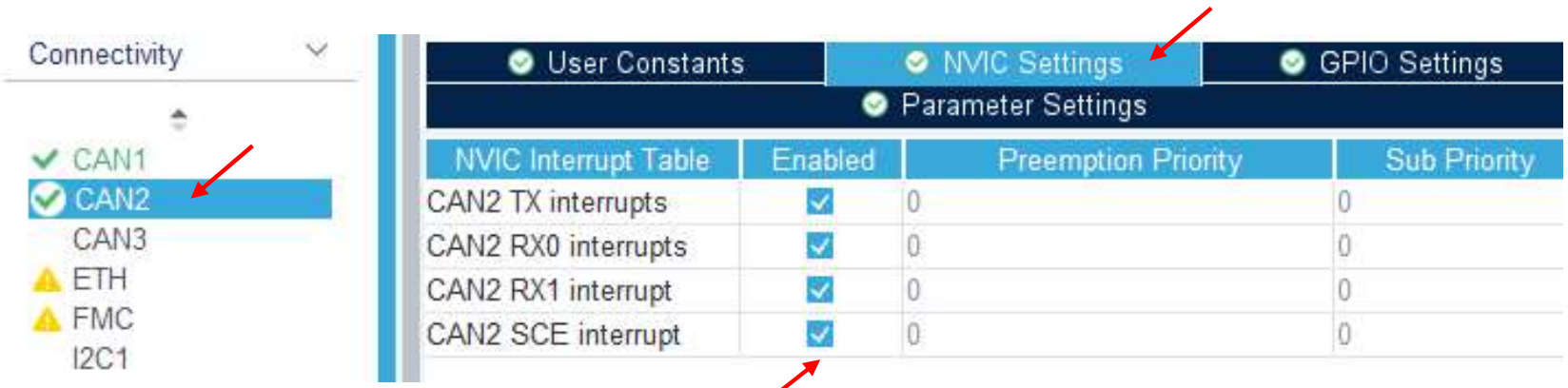
Pin Name	Signal on Pin	GPIO out...	GPIO mode	GPIO Pull-up/Pull-down	Maximum out...	Fast ...	Use...	Mo...
PB13	CAN2_TX	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	n/a		☐
PB12	CAN2_RX	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	n/a		☐

Hands-On Dual CAN Networking

CAN1 & CAN2 Configurations: NVIC Settings



Enable all interrupts



Enable all interrupts

SCE = Status Change Error

Check NVIC

Hands-On Dual CAN Networking

Categories | A->Z

System Core

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

Analog

Timers

Connectivity

- ✔ CAN1
- ✔ CAN2
- CAN3
- ⚠ ETH
- ⚠ FMC
- I2C1
- I2C2
- I2C3
- I2C4
- MDIOS
- QUADSPI

NVIC Mode and Configuration

Configuration

NVIC | Code generation

Search | Show available interrupts | ☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
CAN1 TX interrupts	☑	0	0
CAN1 RX0 interrupts	☑	0	0
CAN1 RX1 interrupt	☑	0	0
CAN1 SCE interrupt	☑	0	0
USART3 global interrupt	☐	0	0
EXTI line[15:10] interrupts	☑	0	0
CAN2 TX interrupts	☑	0	0
CAN2 RX0 interrupts	☑	0	0
CAN2 RX1 interrupt	☑	0	0
CAN2 SCE interrupt	☑	0	0
FPU global interrupt	☐	0	0

Check Interrupts for CAN1 and CAN2

Hands-On Dual CAN Networking

STM32_CS397 - Hands-On_3-2_Dual_CAN_Networking/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Hands-On_3-2_Dual_CAN_Networking
  - Binaries
  - Includes
  - Core
    - Inc
      - can.h
      - gpio.h
      - main.h
      - stm32f7xx_hal_conf.h
      - stm32f7xx_it.h
      - usart.h
    - Src
      - can.c
      - gpio.c
      - main.c
      - stm32f7xx_hal_msp.c
      - stm32f7xx_it.c
      - syscalls.c
      - sysmem.c
      - system_stm32f7xx.c
      - usart.c
    - Startup
  - Drivers
    - CMSIS
    - STM32F7xx_HAL_Driver
      - Inc
      - Src
      - License.md
      - LICENSE.txt
  - Debug
  - Hands-On_3-2_Dual_CAN_Networking.ioc
  - Hands-On_3-2_Dual_CAN_Networking.pdf
  - Hands-On_3-2_Dual_CAN_Networking.txt
  - STM32F767ZITX_FLASH.ld
  - STM32F767ZITX_RAM.ld

Hands-On_3-2_Dual_CAN_Networking.ioc | main.c

```
13   * in the root directory of this software component.
14   * If no LICENSE file comes with this software, it is provided AS-IS.
15   *
16   ******************************************************************************
17   */
18 /* USER CODE END Header */
19 /* Includes ------------------------------------------------------------------*/
20 #include "main.h"
21 #include "can.h"
22 #include "usart.h"
23 #include "gpio.h"
24
25 /* Private includes ----------------------------------------------------------*/
26 /* USER CODE BEGIN Includes */
27
28 /* USER CODE END Includes */
29
30 /* Private typedef -----------------------------------------------------------*/
31 /* USER CODE BEGIN PTD */
32
33 /* USER CODE END PTD */
34
35 /* Private define ------------------------------------------------------------*/
36 /* USER CODE BEGIN PD */
37 /* USER CODE END PD */
```

Save All, Generate Code,

Generate Report, and Build

Problems | Tasks | Console | Properties

CDT Build Console [Hands-On_3-2_Dual_CAN_Networking]

```
arm-none-eabi-objdump -h -S  Hands-On_3-2_Dual_CAN_Networking.elf  > "Hands-On_3-2_Dual_C
   text    data     bss     dec     hex filename
  13780      20    1788   15588    3ce4 Hands-On_3-2_Dual_CAN_Networking.elf
Finished building: default.size.stdout

Finished building: Hands-On_3-2_Dual_CAN_Networking.list


17:10:57 Build Finished. 0 errors, 0 warnings. (took 2s.81ms)
```

Generated **can.c** (1/4)

```c
#include "can.h"

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

CAN_HandleTypeDef hcan1;
CAN_HandleTypeDef hcan2;

/* CAN1 init function */
void MX_CAN1_Init(void)
{
  hcan1.Instance = CAN1;
  hcan1.Init.Prescaler = 6;
  hcan1.Init.Mode = CAN_MODE_NORMAL;
  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();
  }
}
```

```c
/* CAN2 init function */
void MX_CAN2_Init(void)
{
  hcan2.Instance = CAN2;
  hcan2.Init.Prescaler = 6;
  hcan2.Init.Mode = CAN_MODE_NORMAL;
  hcan2.Init.SyncJumpWidth = CAN_SJW_4TQ;
  hcan2.Init.TimeSeg1 = CAN_BS1_6TQ;
  hcan2.Init.TimeSeg2 = CAN_BS2_2TQ;
  hcan2.Init.TimeTriggeredMode = DISABLE;
  hcan2.Init.AutoBusOff = DISABLE;
  hcan2.Init.AutoWakeUp = DISABLE;
  hcan2.Init.AutoRetransmission = ENABLE;
  hcan2.Init.ReceiveFifoLocked = DISABLE;
  hcan2.Init.TransmitFifoPriority = DISABLE;
  if (HAL_CAN_Init(&hcan2) != HAL_OK)
  {
    Error_Handler();
  }
}
```

Hands-On Dual CAN Networking

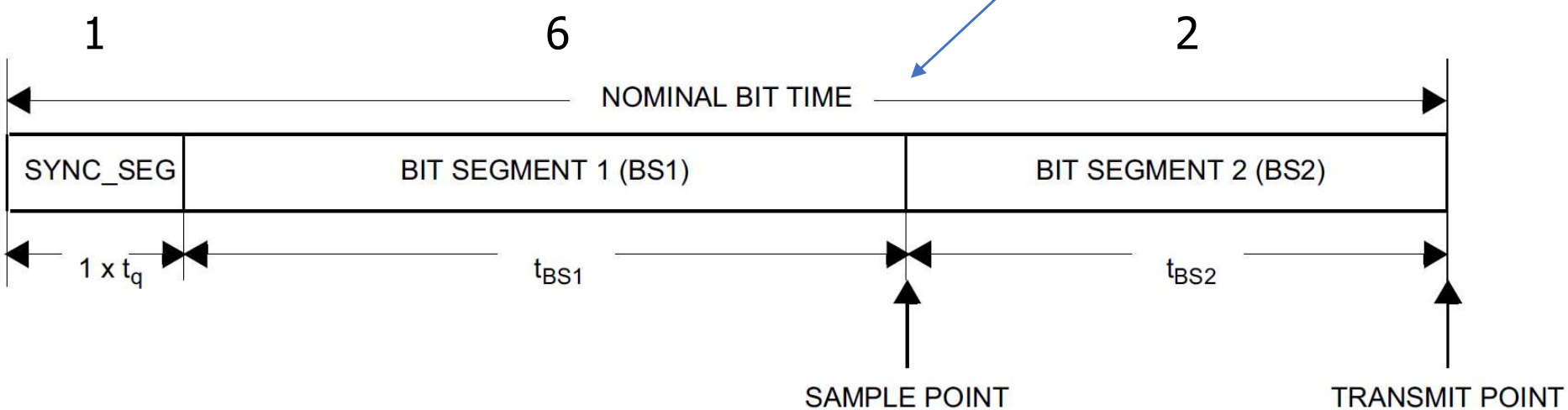
```
hcan1.Init.Prescaler = 6;
hcan1.Init.Mode = CAN_MODE_NORMAL;
hcan1.Init.SyncJumpWidth = CAN_SJW_4TQ;
hcan1.Init.TimeSeg1 = CAN_BS1_6TQ;
hcan1.Init.TimeSeg2 = CAN_BS2_2TQ;
```

at **75%**

$t_{BS1} = (3 \times t_{BS2}) - t_q$

$t_{BS2} = ((8 \text{ to } 25)/4) \times t_q$

$$\frac{7}{9} \times 100\% = 77.78\%$$



PCLK1= 54 MHz, Prescaler = 6 -> 54/6 = 9 MHz

Time Quantum, Tq = (6/54) us = (1/9) us = 111.111111 ns

Time for one Bit = Nominal Bit Time = 9 x Tq = 999.999999 ns = 1 us

Baud Rate = 1 / (Time for one Bit) = 1000 kbps or 1 Mbit/s

-> **Nominal bit time = 1 / baud rate = 1 us = 9 x Tq**

Generated **can.c** (2/4)

```
static uint32_t HAL_RCC_CAN1_CLK_ENABLED=0;

void HAL_CAN_MspInit(CAN_HandleTypeDef* canHandle)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  if(canHandle->Instance==CAN1)
  {
  /* USER CODE BEGIN CAN1_MspInit 0 */

  /* USER CODE END CAN1_MspInit 0 */
    /* CAN1 clock enable */
    HAL_RCC_CAN1_CLK_ENABLED++;
    if(HAL_RCC_CAN1_CLK_ENABLED==1){
      __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);

    /* CAN1 interrupt Init */
    HAL_NVIC_SetPriority(CAN1_TX_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN1_TX_IRQn);
    HAL_NVIC_SetPriority(CAN1_RX0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN1_RX0_IRQn);
    HAL_NVIC_SetPriority(CAN1_RX1_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN1_RX1_IRQn);
    HAL_NVIC_SetPriority(CAN1_SCE_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN1_SCE_IRQn);
  /* USER CODE BEGIN CAN1_MspInit 1 */

  /* USER CODE END CAN1_MspInit 1 */
  }
  else if(canHandle->Instance==CAN2)
  {
  /* USER CODE BEGIN CAN2_MspInit 0 */

  /* USER CODE END CAN2_MspInit 0 */
```

SCE = Status Change Error

Generated **can.c** (3/4)

```
    /* CAN2 clock enable */
    __HAL_RCC_CAN2_CLK_ENABLE();
    HAL_RCC_CAN1_CLK_ENABLED++;
    if(HAL_RCC_CAN1_CLK_ENABLED==1){
      __HAL_RCC_CAN1_CLK_ENABLE();
    }

    __HAL_RCC_GPIOB_CLK_ENABLE();
    /**CAN2 GPIO Configuration
    PB12     ------> CAN2_RX
    PB13     ------> CAN2_TX
    */
    GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13;
    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_CAN2;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* CAN2 interrupt Init */
    HAL_NVIC_SetPriority(CAN2_TX_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN2_TX_IRQn);
    HAL_NVIC_SetPriority(CAN2_RX0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN2_RX0_IRQn);
    HAL_NVIC_SetPriority(CAN2_RX1_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN2_RX1_IRQn);
    HAL_NVIC_SetPriority(CAN2_SCE_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN2_SCE_IRQn);
  /* USER CODE BEGIN CAN2_MspInit 1 */

  /* USER CODE END CAN2_MspInit 1 */
  }
}
```

SCE = Status Change Error

Generated **can.c** (4/4)

```
void HAL_CAN_MspDeInit(CAN_HandleTypeDef* canHandle)
{
  if(canHandle->Instance==CAN1)
  {
  /* USER CODE BEGIN CAN1_MspDeInit 0 */

  /* USER CODE END CAN1_MspDeInit 0 */
    /* Peripheral clock disable */
    HAL_RCC_CAN1_CLK_ENABLED--;
    if(HAL_RCC_CAN1_CLK_ENABLED==0){
      __HAL_RCC_CAN1_CLK_DISABLE();
    }

    /**CAN1 GPIO Configuration
    PD0     ------> CAN1_RX
    PD1     ------> CAN1_TX
    */
    HAL_GPIO_DeInit(GPIOD, GPIO_PIN_0|GPIO_PIN_1);

    /* CAN1 interrupt Deinit */
    HAL_NVIC_DisableIRQ(CAN1_TX_IRQn);
    HAL_NVIC_DisableIRQ(CAN1_RX0_IRQn);
    HAL_NVIC_DisableIRQ(CAN1_RX1_IRQn);
    HAL_NVIC_DisableIRQ(CAN1_SCE_IRQn);
  /* USER CODE BEGIN CAN1_MspDeInit 1 */

  /* USER CODE END CAN1_MspDeInit 1 */
  }
```

SCE = Status Change Error

```
  else if(canHandle->Instance==CAN2)
  {
  /* USER CODE BEGIN CAN2_MspDeInit 0 */

  /* USER CODE END CAN2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_CAN2_CLK_DISABLE();
    HAL_RCC_CAN1_CLK_ENABLED--;
    if(HAL_RCC_CAN1_CLK_ENABLED==0){
      __HAL_RCC_CAN1_CLK_DISABLE();
    }

    /**CAN2 GPIO Configuration
    PB12     ------> CAN2_RX
    PB13     ------> CAN2_TX
    */
    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_12|GPIO_PIN_13);

    /* CAN2 interrupt Deinit */
    HAL_NVIC_DisableIRQ(CAN2_TX_IRQn);
    HAL_NVIC_DisableIRQ(CAN2_RX0_IRQn);
    HAL_NVIC_DisableIRQ(CAN2_RX1_IRQn);
    HAL_NVIC_DisableIRQ(CAN2_SCE_IRQn);
  /* USER CODE BEGIN CAN2_MspDeInit 1 */

  /* USER CODE END CAN2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

Generated **can.h**

```
/* 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;
extern CAN_HandleTypeDef hcan2;

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

void MX_CAN1_Init(void);
void MX_CAN2_Init(void);

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

#ifdef __cplusplus
}
#endif

#endif /*__CAN_H__ */
```

IRQHandler functions generated in **stm32f7xx_it.c**

```c
/**
  * @brief This function handles CAN1 RX0 interrupts.
  */
void CAN1_RX0_IRQHandler(void)
{
  /* USER CODE BEGIN CAN1_RX0_IRQn 0 */

  /* USER CODE END CAN1_RX0_IRQn 0 */
  HAL_CAN_IRQHandler(&hcan1);
  /* USER CODE BEGIN CAN1_RX0_IRQn 1 */

  /* USER CODE END CAN1_RX0_IRQn 1 */
}
```

Open Declaration

```c
void HAL_CAN_IRQHandler(CAN_HandleTypeDef *hcan)
```

in

stm32f7xx_hal_can.c

```c
__weak void HAL_CAN_RxFifo0MsgPendingCallback
                    (CAN_HandleTypeDef *hcan)
```

```c
/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */

  /* USER CODE END EXTI15_10_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}
```

Open Declaration

stm32f7xx_hal_gpio.c

```c
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
```

Hands-On Dual CAN Networking

Understand **stm32f7xx_hal_can.h** and **stm32f7xx_hal_can.c, examples …**

```c
/** @brief  CAN init structure
definition   */
typedef struct
{
  uint32_t Prescaler;
  uint32_t Mode;
  uint32_t SyncJumpWidth;
  uint32_t TimeSeg1;
  uint32_t TimeSeg2;
  FunctionalState TimeTriggeredMode;
  FunctionalState AutoBusOff;
  FunctionalState AutoWakeUp;
  FunctionalState AutoRetransmission;
  FunctionalState ReceiveFifoLocked;
  FunctionalState TransmitFifoPriority;
} CAN_InitTypeDef;
```

```c
/** @brief  CAN filter configuration
structure definition */
typedef struct
{
  uint32_t FilterIdHigh;
  uint32_t FilterIdLow;
  uint32_t FilterMaskIdHigh;
  uint32_t FilterMaskIdLow
  uint32_t FilterFIFOAssignment;
  uint32_t FilterBank;
  uint32_t FilterMode
  uint32_t FilterScale;
  uint32_t FilterActivation;
  uint32_t SlaveStartFilterBank;
 } CAN_FilterTypeDef;
```

```c
/** @brief  CAN Tx message header
structure definition   */
typedef struct
{
  uint32_t StdId;
  uint32_t ExtId;
  uint32_t IDE;
  uint32_t RTR;
  uint32_t DLC;
  FunctionalState TransmitGlobalTime;
} CAN_TxHeaderTypeDef;
```

```c
/** @brief  CAN Rx message header
structure definition  */
typedef struct
{
  uint32_t StdId;
  uint32_t ExtId;
  uint32_t IDE;
  uint32_t RTR
  uint32_t DLC
  uint32_t Timestamp;
  uint32_t FilterMatchIndex
} CAN_RxHeaderTypeDef;
```

CAN_HandleTypeDef

HAL_CAN_CallbackIDTypeDef

Hands-On Dual CAN Networking

Understand **stm32f7xx_hal_can.h** and **stm32f7xx_hal_can.c, examples …**

```
typedef struct __CAN_HandleTypeDef
{
  CAN_TypeDef                 *Instance;   /*!< Register base address */
  CAN_InitTypeDef             Init;        /*!< CAN required parameters */
  __IO HAL_CAN_StateTypeDef   State;       /*!< CAN communication state */
  __IO uint32_t               ErrorCode;   /*!< CAN Error code.
                            This parameter can be a value of @ref CAN_Error_Code */
#if USE_HAL_CAN_REGISTER_CALLBACKS == 1
  void (* TxMailbox0CompleteCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* TxMailbox1CompleteCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* TxMailbox2CompleteCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* TxMailbox0AbortCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* TxMailbox1AbortCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* TxMailbox2AbortCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* RxFifo0MsgPendingCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* RxFifo0FullCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* RxFifo1MsgPendingCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* RxFifo1FullCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* SleepCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* WakeUpFromRxMsgCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* ErrorCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* MspInitCallback)(struct __CAN_HandleTypeDef *hcan);
  void (* MspDeInitCallback)(struct __CAN_HandleTypeDef *hcan);
#endif /* (USE_HAL_CAN_REGISTER_CALLBACKS) */
} CAN_HandleTypeDef;
```

Understand **stm32f7xx_hal_can.h** and **stm32f7xx_hal_can.c, examples …**

```
/** @defgroup CAN_filter_mode CAN Filter Mode */
#define CAN_FILTERMODE_IDMASK       (0x00000000U)  /*!< Identifier mask mode */
#define CAN_FILTERMODE_IDLIST       (0x00000001U)  /*!< Identifier list mode */

/** @defgroup CAN_filter_scale CAN Filter Scale  */
#define CAN_FILTERSCALE_16BIT       (0x00000000U)  /*!< Two 16-bit filters */
#define CAN_FILTERSCALE_32BIT       (0x00000001U)  /*!< One 32-bit filter  */

/** @defgroup CAN_filter_activation CAN Filter Activation  */
#define CAN_FILTER_DISABLE          (0x00000000U)  /*!< Disable filter */
#define CAN_FILTER_ENABLE           (0x00000001U)  /*!< Enable filter  */

/** @defgroup CAN_filter_FIFO CAN Filter FIFO  */
#define CAN_FILTER_FIFO0            (0x00000000U)  /*!< Filter FIFO 0 assignment for filter x */
#define CAN_FILTER_FIFO1            (0x00000001U)  /*!< Filter FIFO 1 assignment for filter x */

/** @defgroup CAN_identifier_type CAN Identifier Type  */
#define CAN_ID_STD                  (0x00000000U)  /*!< Standard Id */
#define CAN_ID_EXT                  (0x00000004U)  /*!< Extended Id */

/** @defgroup CAN_remote_transmission_request CAN Remote Transmission Request  */
#define CAN_RTR_DATA                (0x00000000U)  /*!< Data frame   */
#define CAN_RTR_REMOTE              (0x00000002U)  /*!< Remote frame */

/** @defgroup CAN_receive_FIFO_number CAN Receive FIFO Number  */
#define CAN_RX_FIFO0                (0x00000000U)  /*!< CAN receive FIFO 0 */
#define CAN_RX_FIFO1                (0x00000001U)  /*!< CAN receive FIFO 1 */

/** @defgroup CAN_Tx_Mailboxes CAN Tx Mailboxes  */
#define CAN_TX_MAILBOX0             (0x00000001U)  /*!< Tx Mailbox 0  */
#define CAN_TX_MAILBOX1             (0x00000002U)  /*!< Tx Mailbox 1  */
#define CAN_TX_MAILBOX2             (0x00000004U)  /*!< Tx Mailbox 2  */
```

Hands-On Dual CAN Networking

Understand **stm32f7xx_hal_can.h** and **stm32f7xx_hal_can.c, examples …**

Interrupt enable register

FIFO message pending Interrupt enable

```
/** @defgroup CAN_Interrupts CAN Interrupts */

/* Transmit Interrupt */
#define CAN_IT_TX_MAILBOX_EMPTY     ((uint32_t)CAN_IER_TMEIE)   /*!< Transmit mailbox empty interrupt */

/* Receive Interrupts */
#define CAN_IT_RX_FIFO0_MSG_PENDING ((uint32_t)CAN_IER_FMPIE0)  /*!< FIFO 0 message pending interrupt */
#define CAN_IT_RX_FIFO0_FULL        ((uint32_t)CAN_IER_FFIE0)   /*!< FIFO 0 full interrupt            */
#define CAN_IT_RX_FIFO0_OVERRUN     ((uint32_t)CAN_IER_FOVIE0)  /*!< FIFO 0 overrun interrupt         */
#define CAN_IT_RX_FIFO1_MSG_PENDING ((uint32_t)CAN_IER_FMPIE1)  /*!< FIFO 1 message pending interrupt */
#define CAN_IT_RX_FIFO1_FULL        ((uint32_t)CAN_IER_FFIE1)   /*!< FIFO 1 full interrupt            */
#define CAN_IT_RX_FIFO1_OVERRUN     ((uint32_t)CAN_IER_FOVIE1)  /*!< FIFO 1 overrun interrupt         */

/* Operating Mode Interrupts */
#define CAN_IT_WAKEUP               ((uint32_t)CAN_IER_WKUIE)   /*!< Wake-up interrupt                */
#define CAN_IT_SLEEP_ACK            ((uint32_t)CAN_IER_SLKIE)   /*!< Sleep acknowledge interrupt      */

/* Error Interrupts */
#define CAN_IT_ERROR_WARNING        ((uint32_t)CAN_IER_EWGIE)   /*!< Error warning interrupt          */
#define CAN_IT_ERROR_PASSIVE        ((uint32_t)CAN_IER_EPVIE)   /*!< Error passive interrupt          */
#define CAN_IT_BUSOFF               ((uint32_t)CAN_IER_BOFIE)   /*!< Bus-off interrupt                */
#define CAN_IT_LAST_ERROR_CODE      ((uint32_t)CAN_IER_LECIE)   /*!< Last error code interrupt        */
#define CAN_IT_ERROR                ((uint32_t)CAN_IER_ERRIE)   /*!< Error Interrupt                  */
/**
```

Understand **stm32f7xx_hal_can.h** and **stm32f7xx_hal_can.c, examples …**

```
/* Callback functions */
void HAL_CAN_TxMailbox0CompleteCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_TxMailbox1CompleteCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_TxMailbox2CompleteCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_TxMailbox0AbortCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_TxMailbox1AbortCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_TxMailbox2AbortCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_RxFifo0FullCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_RxFifo1MsgPendingCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_RxFifo1FullCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_SleepCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_WakeUpFromRxMsgCallback(CAN_HandleTypeDef *hcan);
void HAL_CAN_ErrorCallback(CAN_HandleTypeDef *hcan);

/* Interrupts management */
HAL_StatusTypeDef HAL_CAN_ActivateNotification(CAN_HandleTypeDef *hcan, uint32_t ActiveITs);
HAL_StatusTypeDef HAL_CAN_DeactivateNotification(CAN_HandleTypeDef *hcan, uint32_t InactiveITs);
void HAL_CAN_IRQHandler(CAN_HandleTypeDef *hcan);
```

Understand **stm32f7xx_hal_can.h** and **stm32f7xx_hal_can.c, examples …**

```
/* Control functions /
HAL_StatusTypeDef HAL_CAN_Start(CAN_HandleTypeDef *hcan);
HAL_StatusTypeDef HAL_CAN_Stop(CAN_HandleTypeDef *hcan);
HAL_StatusTypeDef HAL_CAN_RequestSleep(CAN_HandleTypeDef *hcan);
HAL_StatusTypeDef HAL_CAN_WakeUp(CAN_HandleTypeDef *hcan);
uint32_t HAL_CAN_IsSleepActive(CAN_HandleTypeDef *hcan);
HAL_StatusTypeDef HAL_CAN_AddTxMessage(CAN_HandleTypeDef *hcan,
                CAN_TxHeaderTypeDef *pHeader, uint8_t aData[], uint32_t *pTxMailbox);
HAL_StatusTypeDef HAL_CAN_AbortTxRequest(CAN_HandleTypeDef *hcan, uint32_t TxMailboxes);
uint32_t HAL_CAN_GetTxMailboxesFreeLevel(CAN_HandleTypeDef *hcan);
uint32_t HAL_CAN_IsTxMessagePending(CAN_HandleTypeDef *hcan, uint32_t TxMailboxes);
uint32_t HAL_CAN_GetTxTimestamp(CAN_HandleTypeDef *hcan, uint32_t TxMailbox);
HAL_StatusTypeDef HAL_CAN_GetRxMessage(CAN_HandleTypeDef *hcan, uint32_t RxFifo,
                                        CAN_RxHeaderTypeDef *pHeader, uint8_t aData[]);
uint32_t HAL_CAN_GetRxFifoFillLevel(CAN_HandleTypeDef *hcan, uint32_t RxFifo);
```

Add Code to **can.c** (1/8)

```c
/* USER CODE BEGIN 0 */

CAN_TxHeaderTypeDef   TxHeader1;
CAN_RxHeaderTypeDef   RxHeader1;

CAN_TxHeaderTypeDef   TxHeader2;
CAN_RxHeaderTypeDef   RxHeader2;

uint8_t     TxData1[8] = {0};
uint8_t     RxData1[8] = {0};
uint32_t    TxMailbox1 = 0;

uint32_t    RxStdId1 = 0x0000;
uint32_t    RxStdId2 = 0x0000;
uint32_t    RxExtId1 = 0x0000;
uint32_t    RxExtId2 = 0x0000;

uint8_t     TxData2[8] = {0};
uint8_t     RxData2[8] = {0};
uint32_t    TxMailbox2 = 0;

uint8_t ubKeyNumber = 0x0;

/* USER CODE END 0 */
```

```c
/* USER CODE BEGIN 1 */

/* Configure the CAN1 */
void CAN1_Config(void)
{
  CAN_FilterTypeDef  sFilterConfig1;

  /* #1 Configure the CAN Filter */
  sFilterConfig1.FilterBank = 0;
  sFilterConfig1.FilterMode = CAN_FILTERMODE_IDMASK;
  sFilterConfig1.FilterScale = CAN_FILTERSCALE_32BIT;
  sFilterConfig1.FilterIdHigh = 0x0000;
  sFilterConfig1.FilterIdLow  = 0x0000;
  sFilterConfig1.FilterMaskIdHigh = 0x0000;
  sFilterConfig1.FilterMaskIdLow  = 0x0000;
  sFilterConfig1.FilterFIFOAssignment = CAN_RX_FIFO0;
  sFilterConfig1.FilterActivation = ENABLE;
  sFilterConfig1.SlaveStartFilterBank = 14;

  if (HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig1) != HAL_OK)
  {
    /* Filter configuration Error */
    Error_Handler();
  }
```

Add Code to **can.c** (2/8)

```
/* #1 Configure the CAN Filter */
sFilterConfig1.FilterBank = 1;
sFilterConfig1.FilterMode = CAN_FILTERMODE_IDMASK;
sFilterConfig1.FilterScale = CAN_FILTERSCALE_32BIT;
sFilterConfig1.FilterIdHigh = 0x0000;
sFilterConfig1.FilterIdLow  = 0x0000;
sFilterConfig1.FilterMaskIdHigh = 0x0000;
sFilterConfig1.FilterMaskIdLow  = 0x0000;
sFilterConfig1.FilterFIFOAssignment = CAN_RX_FIFO0;
sFilterConfig1.FilterActivation = ENABLE;
sFilterConfig1.SlaveStartFilterBank = 14;

if (HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig1) != HAL_OK)
{
  /* Filter configuration Error */
  Error_Handler();
}

/* #1 Configure the CAN Filter */
  sFilterConfig1.FilterBank = 3;
  sFilterConfig1.FilterMode = CAN_FILTERMODE_IDMASK;
  sFilterConfig1.FilterScale = CAN_FILTERSCALE_32BIT;
  sFilterConfig1.FilterIdHigh = 0x0000;
  sFilterConfig1.FilterIdLow  = 0x0000;
  sFilterConfig1.FilterMaskIdHigh = 0x0000;
  sFilterConfig1.FilterMaskIdLow  = 0x0000;
  sFilterConfig1.FilterFIFOAssignment = CAN_RX_FIFO0;
  sFilterConfig1.FilterActivation = ENABLE;
  sFilterConfig1.SlaveStartFilterBank = 14;

  if (HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig1) != HAL_OK)
  {
    /* Filter configuration Error */
    Error_Handler();
  }
```

Add Code to **can.c** (3/8)

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

  /* #3 Activate CAN RX notification */
  if (HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING) != HAL_OK)
  {
    /* Notification Error */
    Error_Handler();
  }

//   if (HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO1_MSG_PENDING) != HAL_OK)
//   {
//     /* Notification Error */
//     Error_Handler();
//   }

  /* #4 Configure Transmission process */
  TxHeader1.StdId = 0x321;
  TxHeader1.ExtId = 0x01FFFF;
  TxHeader1.IDE = CAN_ID_STD; // CAN_ID_EXT;
  TxHeader1.RTR = CAN_RTR_DATA;
  TxHeader1.DLC = 3;
  TxHeader1.TransmitGlobalTime = DISABLE;
}
```

Add Code to **can.c** (4/8)

```
/* Configure the CAN2 */
void CAN2_Config(void)
{
  CAN_FilterTypeDef  sFilterConfig2;

  /* #1 Configure the CAN Filter */
  sFilterConfig2.FilterBank = 14;
  sFilterConfig2.FilterMode = CAN_FILTERMODE_IDMASK;
  sFilterConfig2.FilterScale = CAN_FILTERSCALE_32BIT;
  sFilterConfig2.FilterIdHigh = 0x0000;
  sFilterConfig2.FilterIdLow  = 0x0000;
  sFilterConfig2.FilterMaskIdHigh = 0x0000;
  sFilterConfig2.FilterMaskIdLow  = 0x0000;
  sFilterConfig2.FilterFIFOAssignment = CAN_RX_FIFO0;
  sFilterConfig2.FilterActivation = ENABLE;
  sFilterConfig2.SlaveStartFilterBank = 14;

  if (HAL_CAN_ConfigFilter(&hcan2, &sFilterConfig2) != HAL_OK)
  {
    /* Filter configuration Error */
    Error_Handler();
  }

  /* #1 Configure the CAN Filter */
  sFilterConfig2.FilterBank = 15;
  sFilterConfig2.FilterMode = CAN_FILTERMODE_IDMASK;
  sFilterConfig2.FilterScale = CAN_FILTERSCALE_32BIT;
  sFilterConfig2.FilterIdHigh = 0x0654 << 5;
  sFilterConfig2.FilterIdLow  = 0x0000;
  sFilterConfig2.FilterMaskIdHigh = 0x0234 << 5;
  sFilterConfig2.FilterMaskIdLow  = 0x0000;
  sFilterConfig2.FilterFIFOAssignment = CAN_RX_FIFO0;
  sFilterConfig2.FilterActivation = ENABLE;
```

Add Code to **can.c** (5/8)

```
sFilterConfig2.SlaveStartFilterBank = 14;

if (HAL_CAN_ConfigFilter(&hcan2, &sFilterConfig2) != HAL_OK)
{
  /* Filter configuration Error */
  Error_Handler();
}

/* #1 Configure the CAN Filter */
sFilterConfig2.FilterBank = 16;
sFilterConfig2.FilterMode = CAN_FILTERMODE_IDMASK;
sFilterConfig2.FilterScale = CAN_FILTERSCALE_32BIT;
sFilterConfig2.FilterIdHigh = 0x0654 << 5;
sFilterConfig2.FilterIdLow  = 0x0000;
sFilterConfig2.FilterMaskIdHigh = 0x0111 << 5;
sFilterConfig2.FilterMaskIdLow  = 0x0000;
sFilterConfig2.FilterFIFOAssignment = CAN_RX_FIFO0;
sFilterConfig2.FilterActivation = ENABLE;
sFilterConfig2.SlaveStartFilterBank = 14;

if (HAL_CAN_ConfigFilter(&hcan2, &sFilterConfig2) != HAL_OK)
{
  /* Filter configuration Error */
  Error_Handler();
}

/* #2 Start the CAN peripheral */
if (HAL_CAN_Start(&hcan2) != HAL_OK)
{
  /* Start Error */
  Error_Handler();
}
```

Add Code to **can.c** (6/8)

```
  /* #3 Activate CAN RX notification */
  if (HAL_CAN_ActivateNotification(&hcan2,  CAN_IT_RX_FIFO0_MSG_PENDING) != HAL_OK)
  {
    /* Notification Error */
    Error_Handler();
  }

  /* #4 Configure Transmission process */
  TxHeader2.StdId = 0x654;
  TxHeader2.ExtId = 0x01EEEE;
  TxHeader2.RTR = CAN_RTR_DATA;
  TxHeader2.IDE = CAN_ID_STD; // CAN_ID_EXT
  TxHeader2.DLC = 3;
  TxHeader2.TransmitGlobalTime = DISABLE;
}
```

Add Code to **can.c** (7/8)

```
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    if (hcan->Instance == CAN1)
    {
        /* Get RX message */
        if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader1, RxData1) != HAL_OK)
        {
            Error_Handler();
        }
        RxStdId1 = RxHeader1.StdId;
        RxExtId1 = RxHeader1.ExtId;
    }
    if (hcan->Instance == CAN2)
    {
        /* Get RX message */
        if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader2, RxData2) != HAL_OK)
        {
            Error_Handler();
        }
        RxStdId2 = RxHeader2.StdId;
        RxExtId2 = RxHeader2.ExtId;
  }
}
```

Add Code to **can.c** (8/8)

```
void LED_Display(uint8_t LedStatus)
{
  /* Turn OFF all LEDs */
  HAL_GPIO_WritePin(GPIOB, LD1_Pin|LD2_Pin|LD3_Pin, GPIO_PIN_RESET);

  switch(LedStatus)
  {
    case(1):
      /* Turn ON LED1 */
      HAL_GPIO_WritePin(GPIOB, LD1_Pin, GPIO_PIN_SET);
      break;

    case(2):
      /* Turn ON LED2 */
      HAL_GPIO_WritePin(GPIOB, LD2_Pin, GPIO_PIN_SET);
      break;

    case(3):
      /* Turn ON LED3 */
      HAL_GPIO_WritePin(GPIOB, LD3_Pin, GPIO_PIN_SET);
      break;

      default:
      break;
  }
}

/* USER CODE END 1 */
```

Add Code **can.h**

```
/* USER CODE BEGIN Private defines */

extern CAN_TxHeaderTypeDef   TxHeader1;
extern CAN_RxHeaderTypeDef   RxHeader1;

extern CAN_TxHeaderTypeDef   TxHeader2;
extern CAN_RxHeaderTypeDef   RxHeader2;

extern uint8_t     TxData1[8];
extern uint8_t     RxData1[8];
extern uint32_t    TxMailbox1;

extern uint32_t    RxStdId1;
extern uint32_t    RxStdId2;
extern uint32_t    RxExtId1;
extern uint32_t    RxExtId2;

extern uint8_t     TxData2[8];
extern uint8_t     RxData2[8];
extern uint32_t    TxMailbox2;

extern uint8_t     ubKeyNumber;

/* USER CODE END Private defines */

/* USER CODE BEGIN Prototypes */

void CAN1_Config(void);
void CAN2_Config(void);
void LED_Display(uint8_t LedStatus);

/* USER CODE END Prototypes */
```

Add Code to **main.c** (1/3)

```c
/* main.c */
/* Includes */
#include "main.h"
#include "can.h"
#include "usart.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();
```

```c
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART3_UART_Init();
MX_CAN1_Init();
MX_CAN2_Init();
/* USER CODE BEGIN 2 */
CAN1_Config();
CAN2_Config();
ubKeyNumber = 0x01;
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
```

Add Code to **main.c** (2/3)

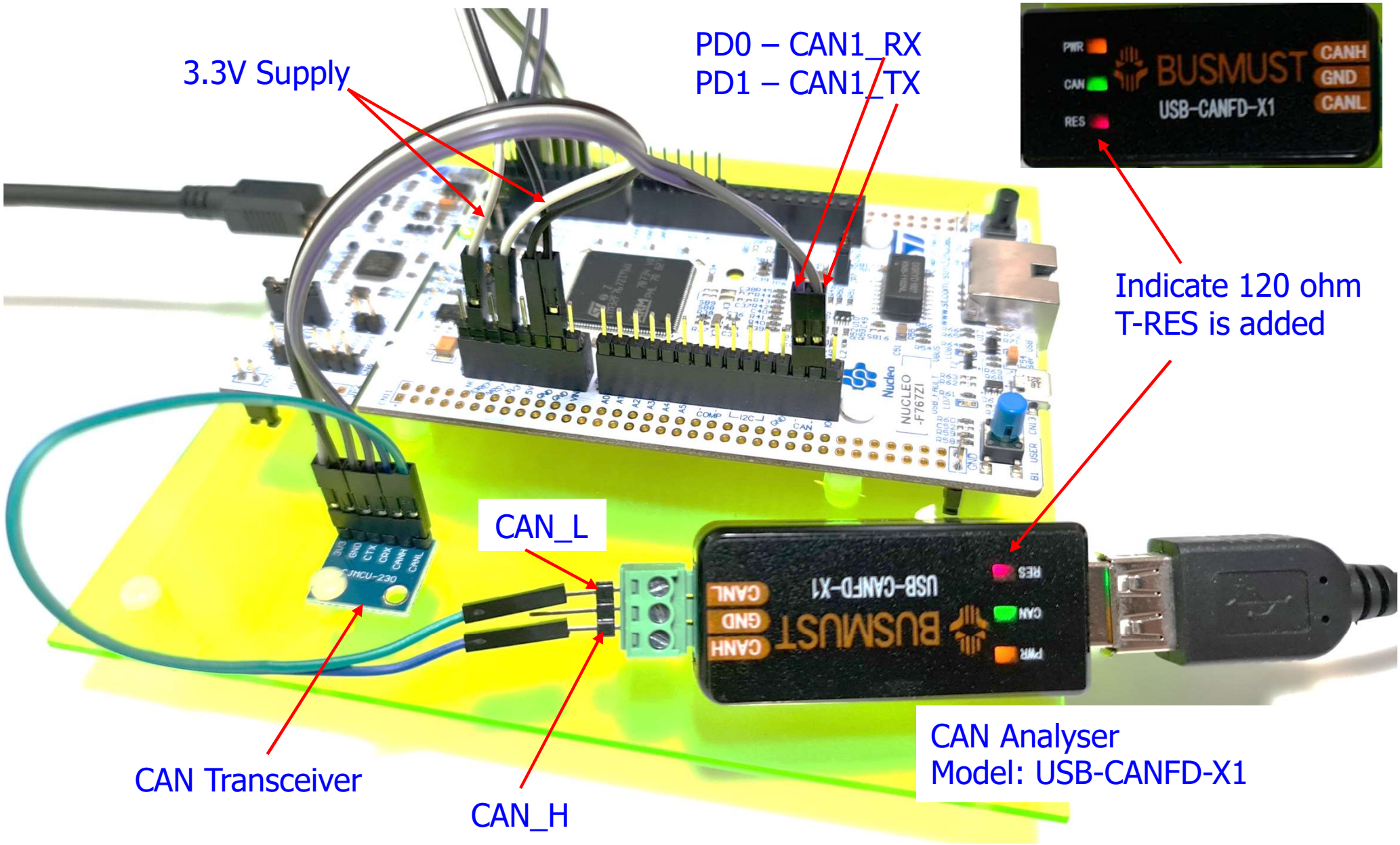

```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
     /* Set up data for transmit */
     TxData1[0] = ubKeyNumber;
     TxData1[1] = 0x77;
     TxData1[2] = 0xBB;
     TxData2[0] = ubKeyNumber;
     TxData2[1] = 0xAA;
     TxData2[2] = 0x55;
     /* Start the Transmission process */
     if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader1, TxData1, &TxMailbox1) != HAL_OK)
     {
         /* Transmission request Error */
         Error_Handler();
     }
     if (HAL_CAN_AddTxMessage(&hcan2, &TxHeader2, TxData2, &TxMailbox2) != HAL_OK)
     {
         /* Transmission request Error */
         Error_Handler();
     }
     HAL_Delay(500);
     LED_Display(ubKeyNumber);
     printf("CAN1: TX: 0x%X 0x%X 0x%X   RX: 0x%X 0x%X 0x%X  StdID: 0x%lX     ", TxData1[0],
             TxData1[1], TxData1[2], RxData1[0], RxData1[1], RxData1[2], RxStdId1);
     printf("CAN2: TX: 0x%X 0x%X 0x%X   RX: 0x%X 0x%X 0x%X  StdID: 0x%lX \n\r", TxData2[0],
             TxData2[1], TxData2[2], RxData2[0], RxData2[1], RxData2[2], RxStdId2);

    ubKeyNumber = ubKeyNumber + 1;
    if (ubKeyNumber >= 0x4) ubKeyNumber = 0x01;
    HAL_Delay(500);
   /* USER CODE END WHILE */
```

Add Code to **main.c** (3/3)

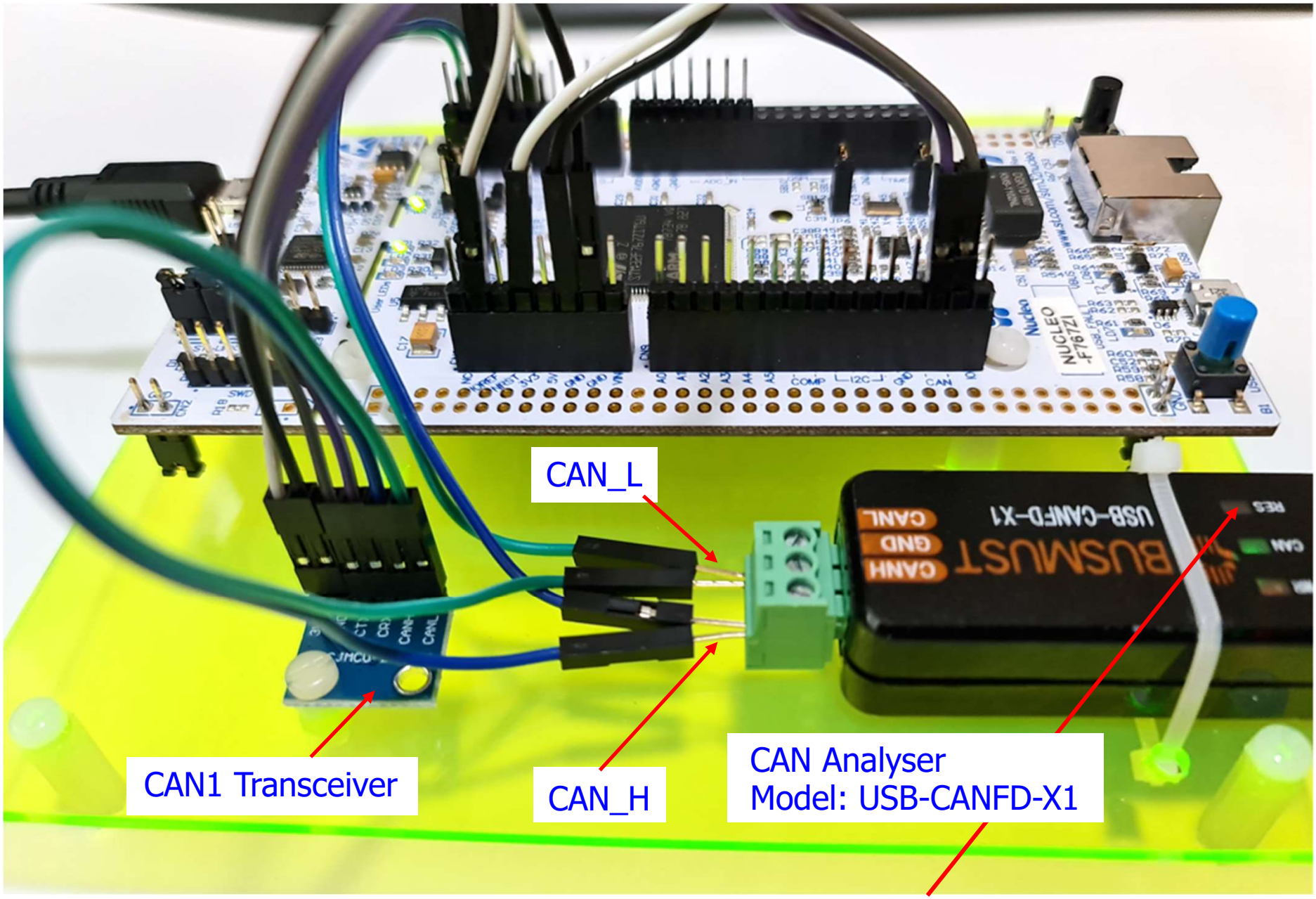
```c
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
    }
}

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 Dual CAN Networking

For single CAN: Connect CAN Analyzer to Nucleo-F767ZI via a CAN Transceiver

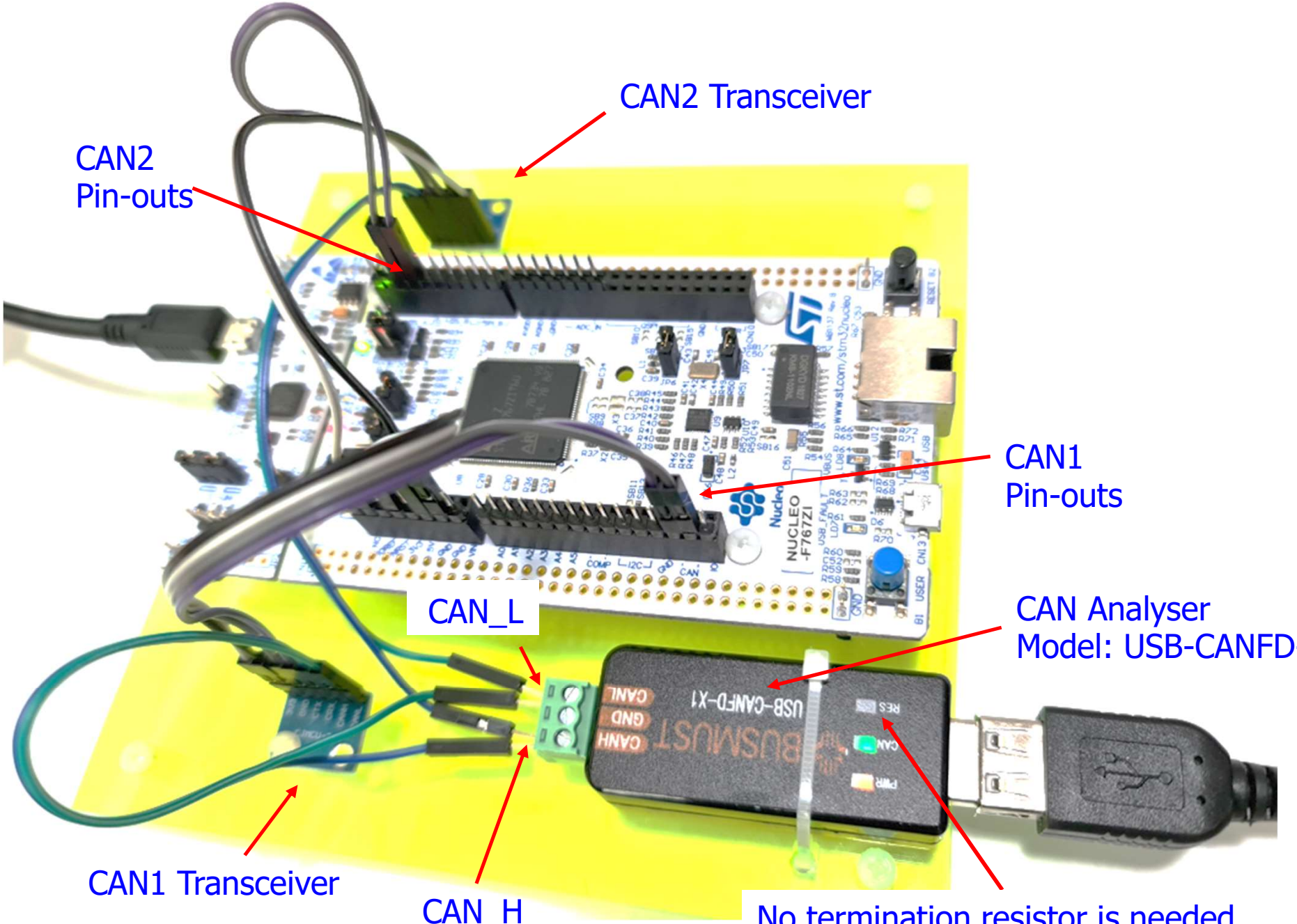


Setup for Dual CAN Networking



Hands-On Dual CAN Networking

Setup for Dual CAN Networking

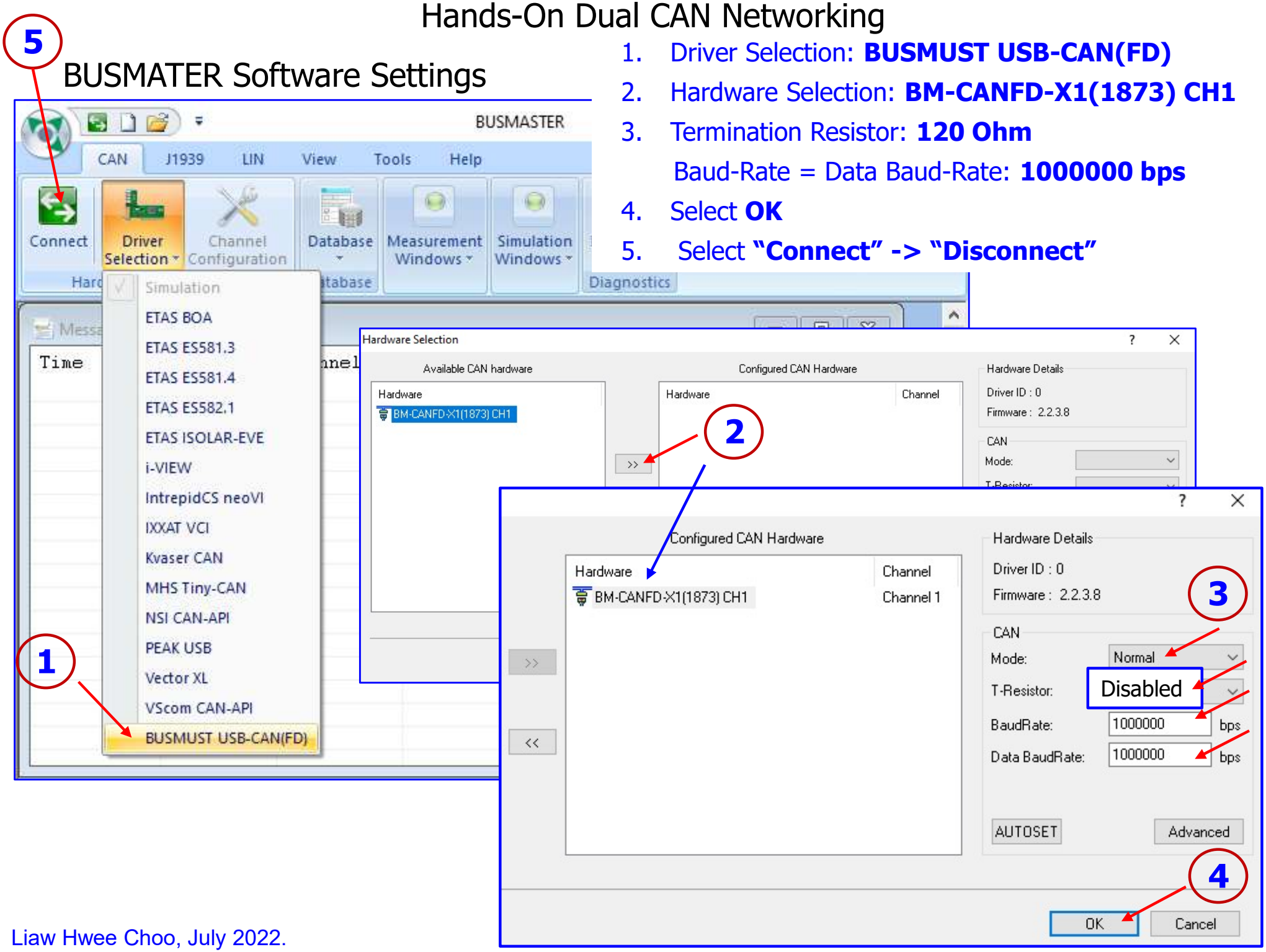


Hands-On Dual CAN Networking

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"**



Hands-On Dual CAN Networking

Set-up CAN Analyzer to Transmit Data

Select Transmit Window

BUSMASTER

CAN | J1939 | LIN | View | Tools | Help

Disconnect | Driver Selection | Channel Configuration | Database | Network Statistics | Signal Graph | Filters | Message Window | Signal Watch | Logging | Transmit Window | Node Simulation | Replay | Waveform Messages | Test Automation Executor

Hardware Configuration | Database | Measurement Windows | Simulation Windows

Configure Transmission Messages - CAN

Tx Frame List

Message Name	Frame Id	Channel	Data Length	Message Type	RTR		Repetition (ms)		Key
0x333	0x333	1	2	Std	☐	☑	1000	☐	a
0x321	0x321	1	2	Std	☐	☑	1000	☐	a
0x321	0x321	1	2	Std	☐	☑	1000	☐	a
0x345	0x345	1	2	Std	☐	☑	1000	☐	a
[Add Message]									

Double-click to add message (Frame ID)

Press "Send Message" button to transmit selected message.

Send Message | Delete | Delete All

Data Byte View (HEX)

Index	00	01	02	03	04	05	06	07
000	CA	FE						

Double-click to change data values

Data for 0x333 : CA FE
0x321 : 12 34
0x321 : AA CC
0x345 : 53 67

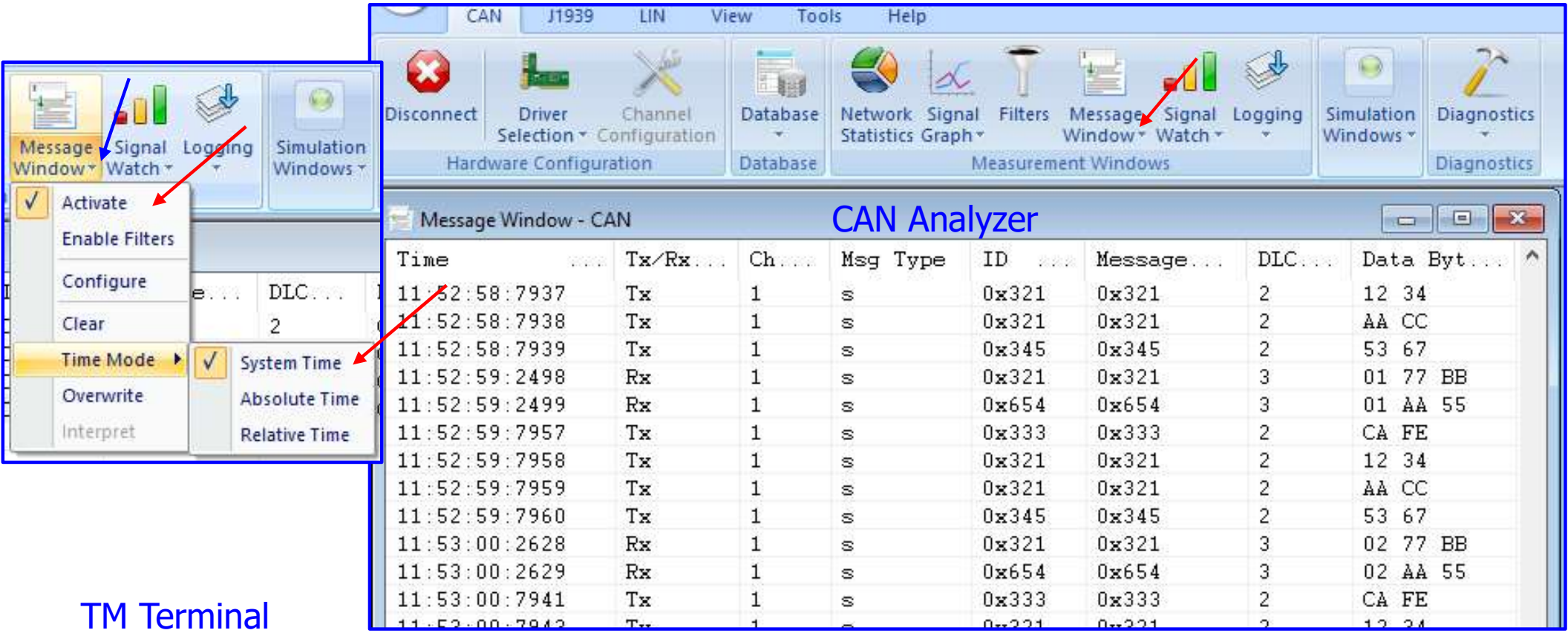
Signal Details

Signal Name	Raw Value	Physical Value	Unit

	7	6	5	4	3	2	1	0
0	1	1	0	0	1	0	1	0
1	1	1	1	1	1	1	1	0
2								

Hands-On Dual CAN Networking

Test and Understand the Dual CAN Networking (TX and RX) implementation



TM Terminal