Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 3-1
CAN Networking (Controller Area Network, Networking)

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Hands-On CAN Networking

Objectives

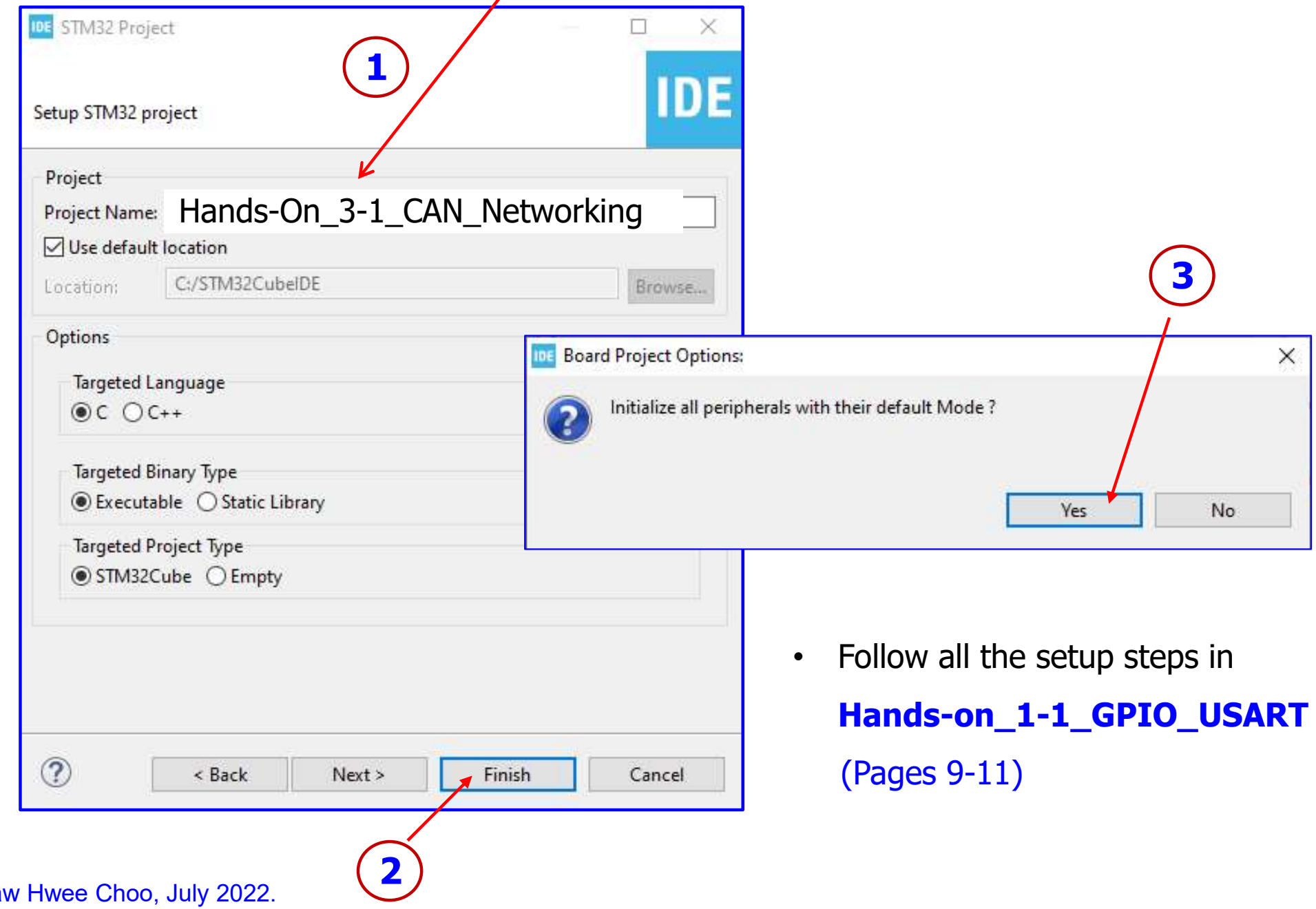
The aims of this session are to

- implement a STM32 (STM32CubeIDE) project
- set up a CAN (Controller Area Network) application development system using STM32F767 microcontroller
- develop a CAN networking (TX and RX) application
- 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 Networking

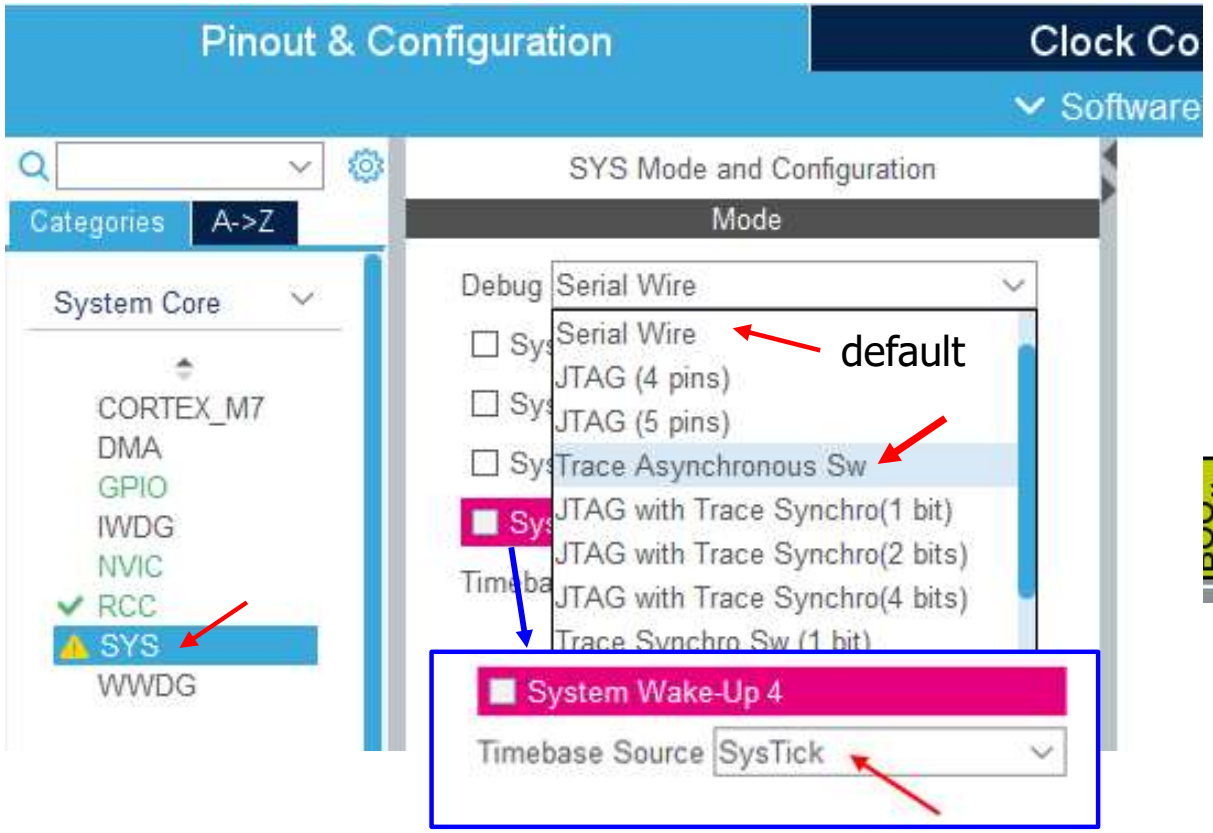
Enter Project Name: Hands-On_3-1_CAN_Networking

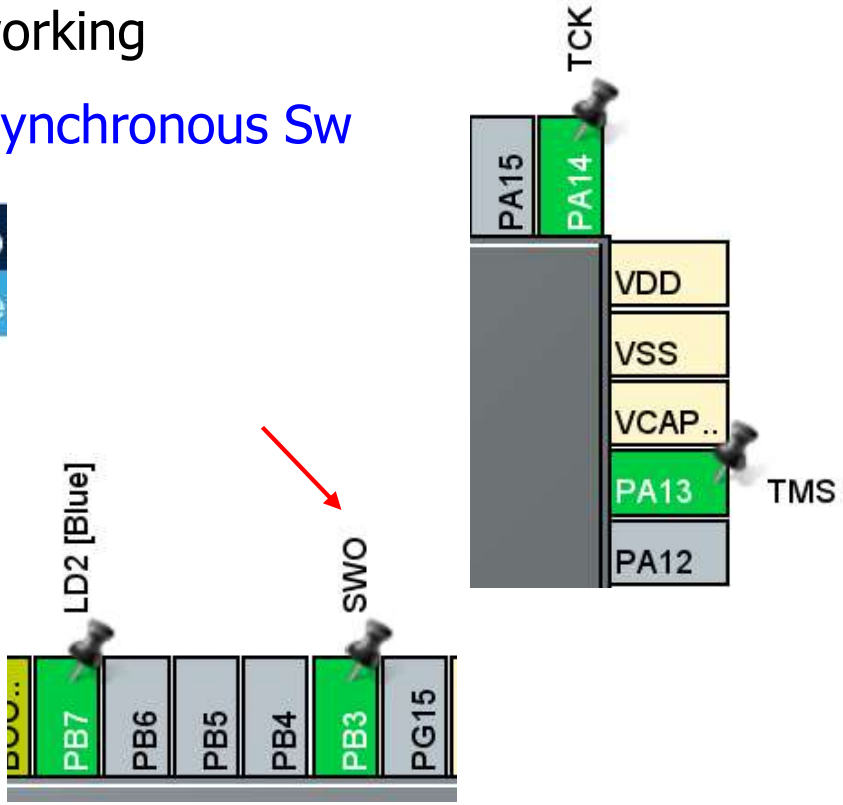


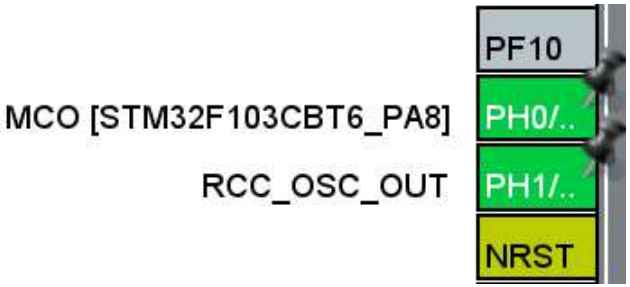
- Follow all the setup steps in **Hands-on_1-1_GPIO_USART** (Pages 9-11)

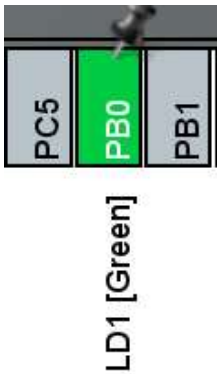
Hands-On CAN Networking

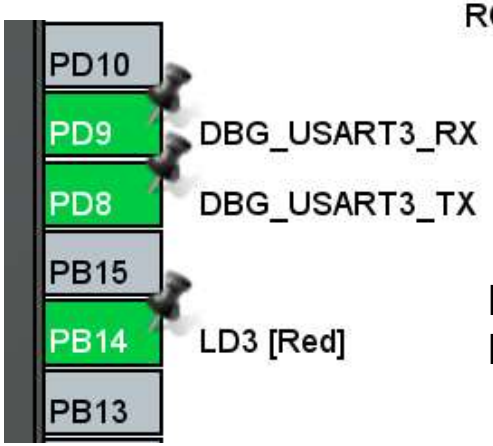
Select: System Core -> SYS -> Debug: Trace Asynchronous Sw

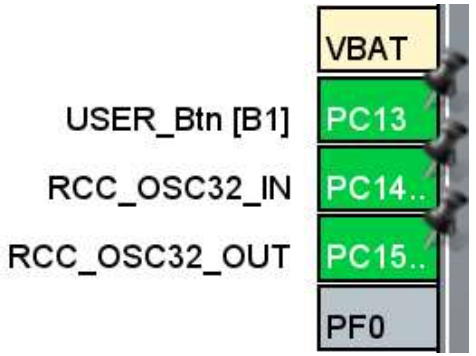












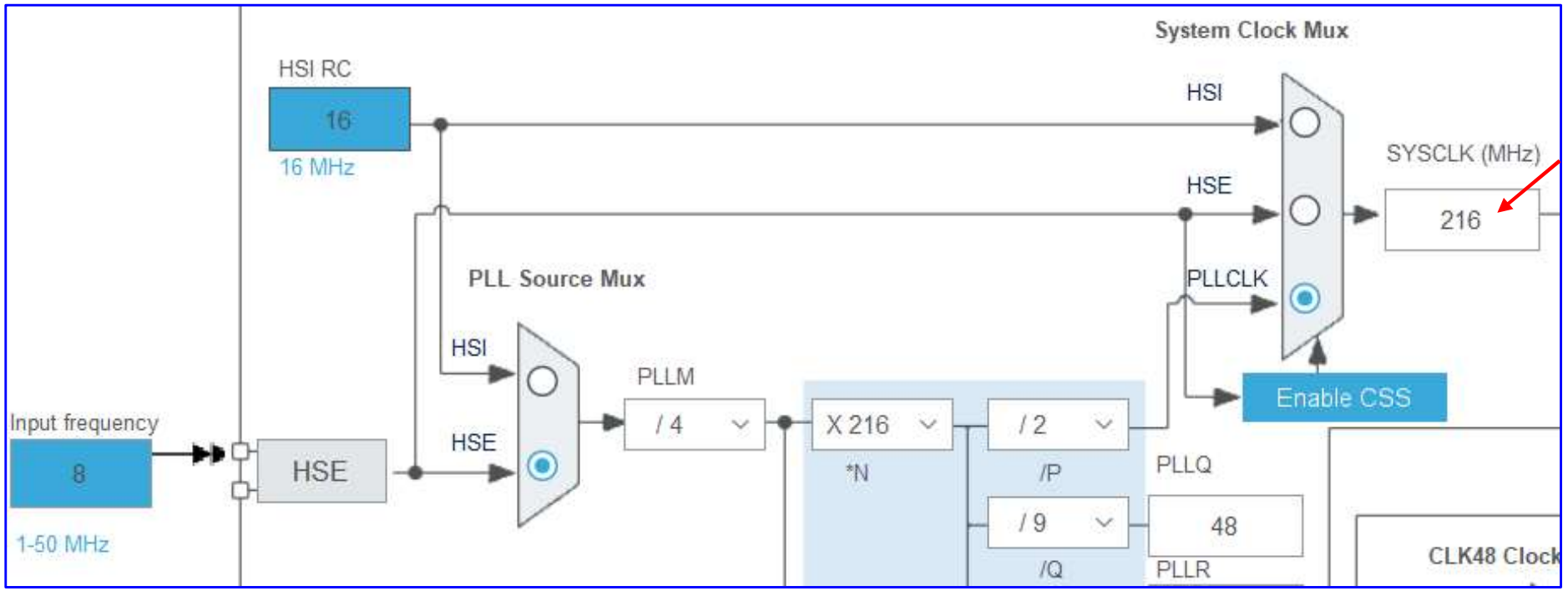
PD9: DBG_USART3_RX
PD8: DBG_USART3_TX

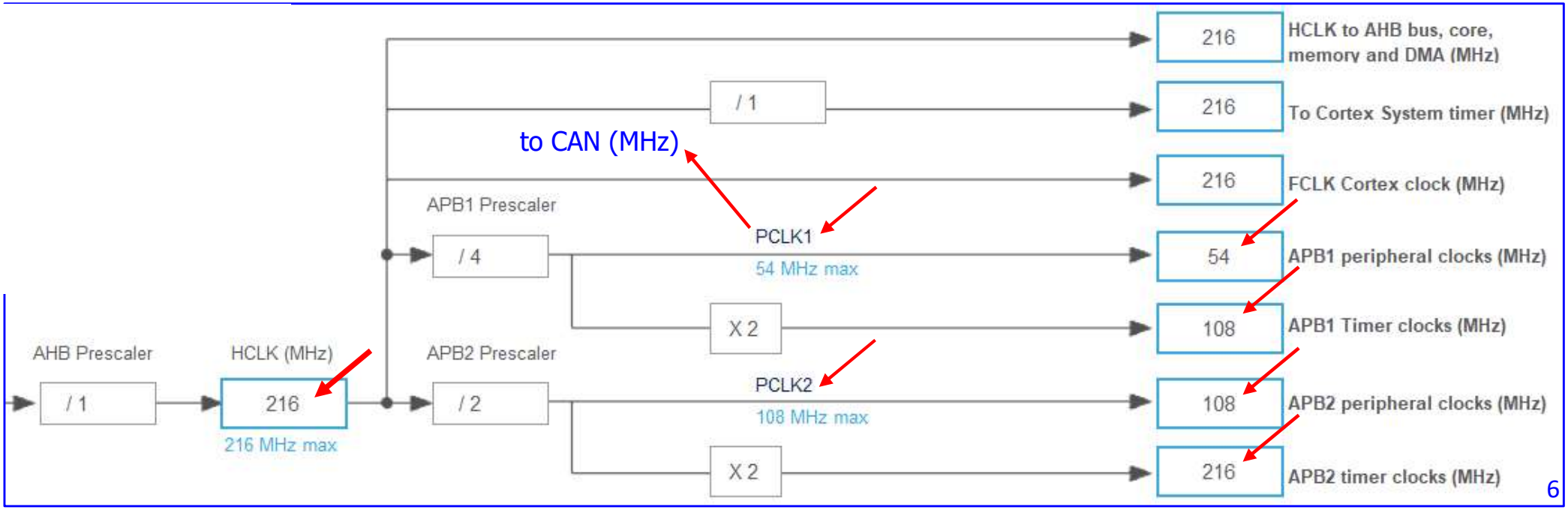
TCK (Test Clock)
TMS (Test Mode Select)
SWO (Single/Serial Wire Output)

After re-set the unused pins

Hands-On CAN Loop Back

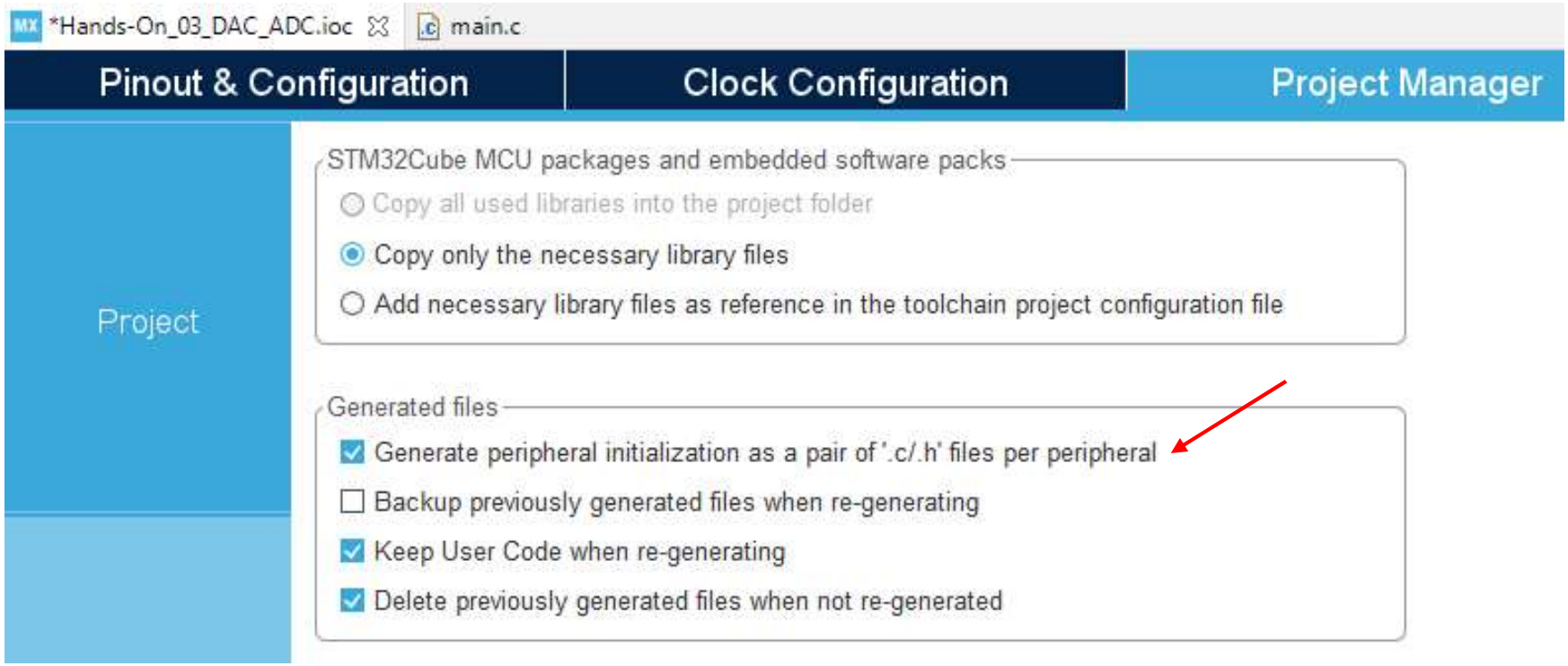
Clock Configuration: Use maximum frequency for clock settings





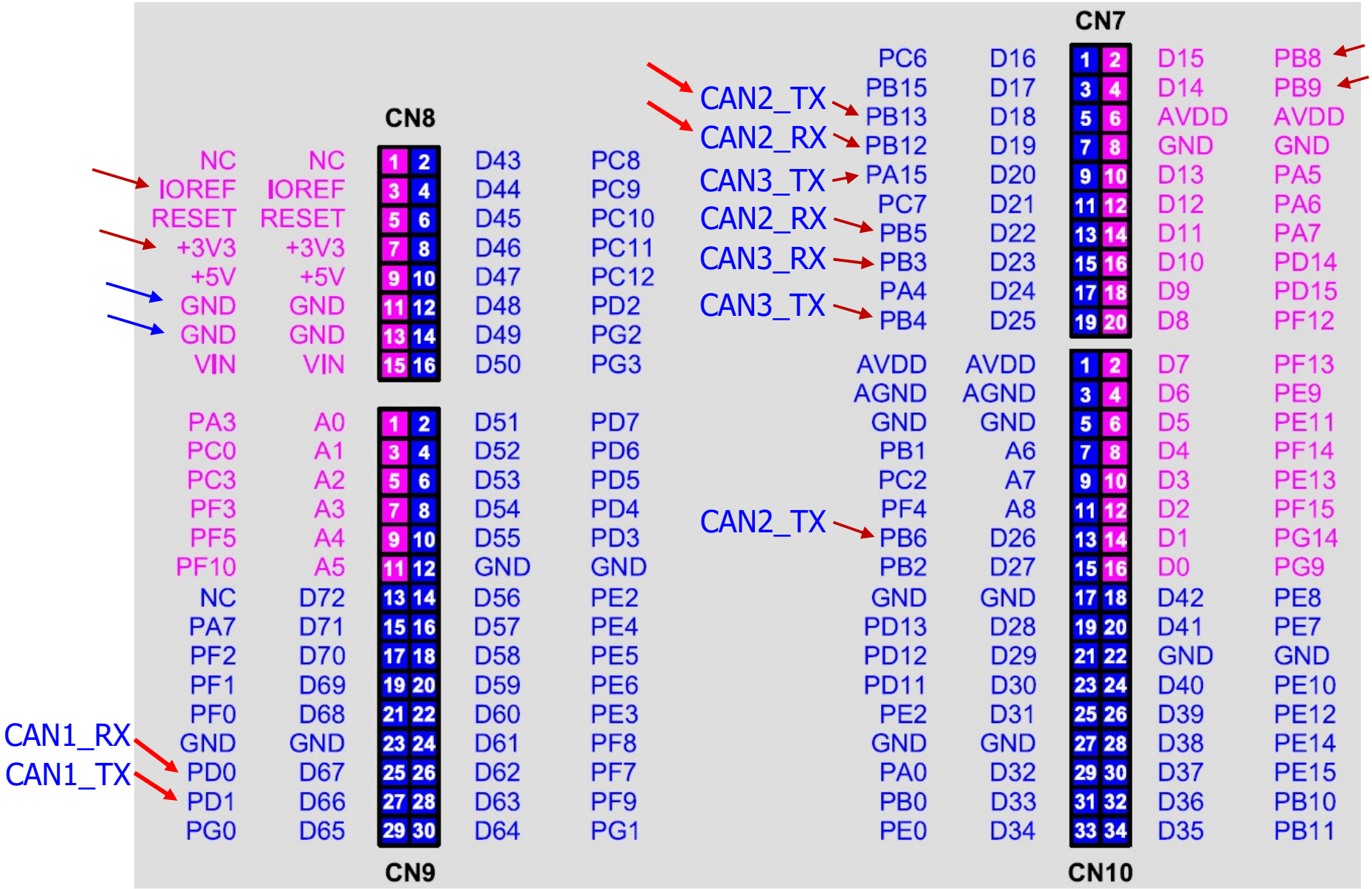
Hands-On CAN Networking

- 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 CAN Networking

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

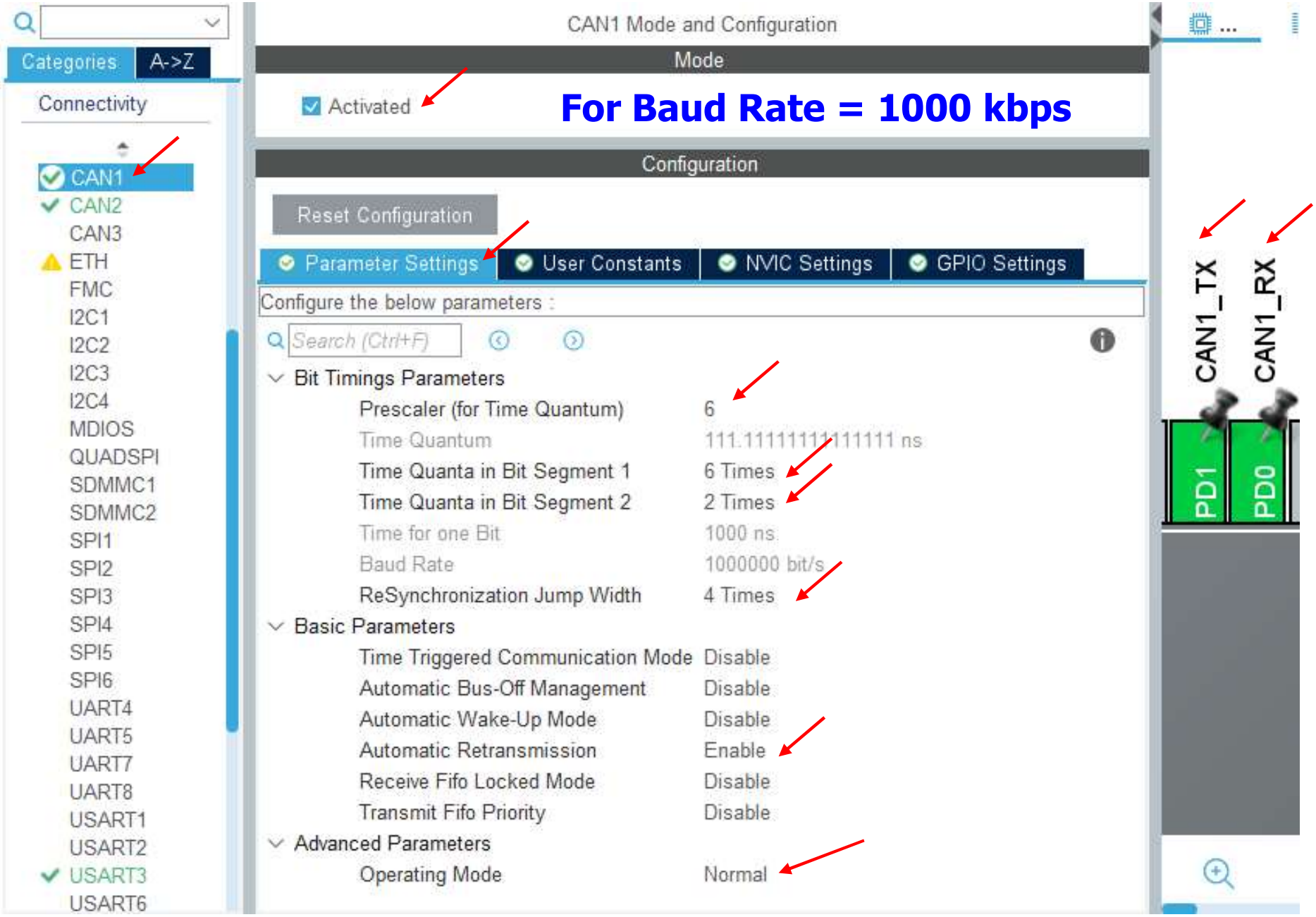


CAN1 RX : PD0
CAN1 TX : PD1

CAN2 TX : PB13
CAN2 RX : PB12

Hands-On CAN Networking

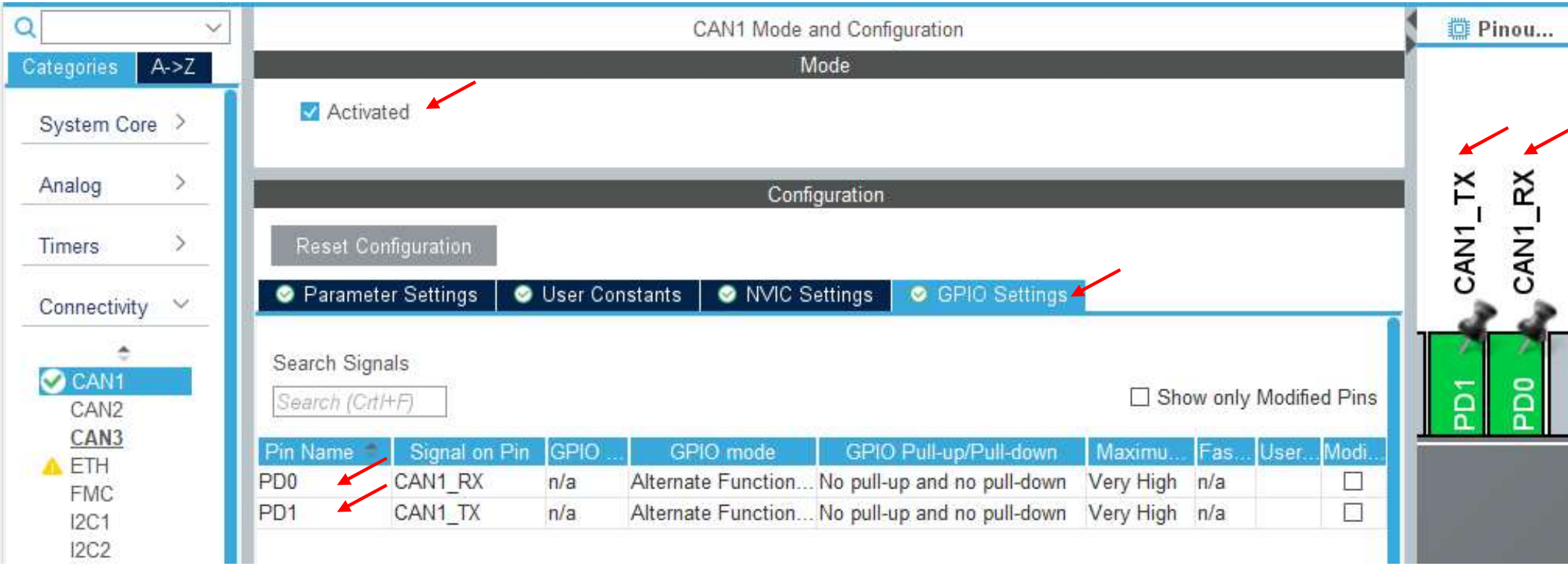
CAN Configuration: select CAN1, Activated, enter values (6, 6, 2, 4) as shown

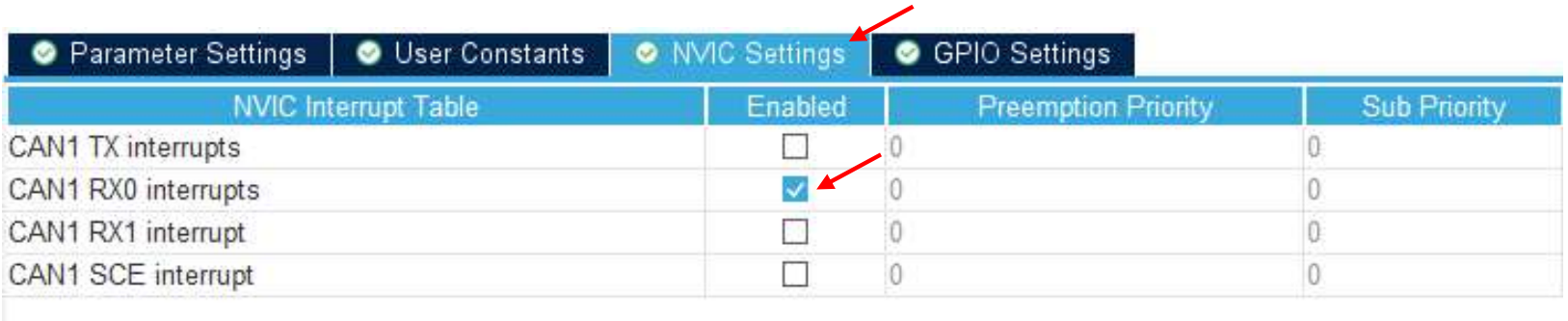


Note: Operating mode is **Normal**

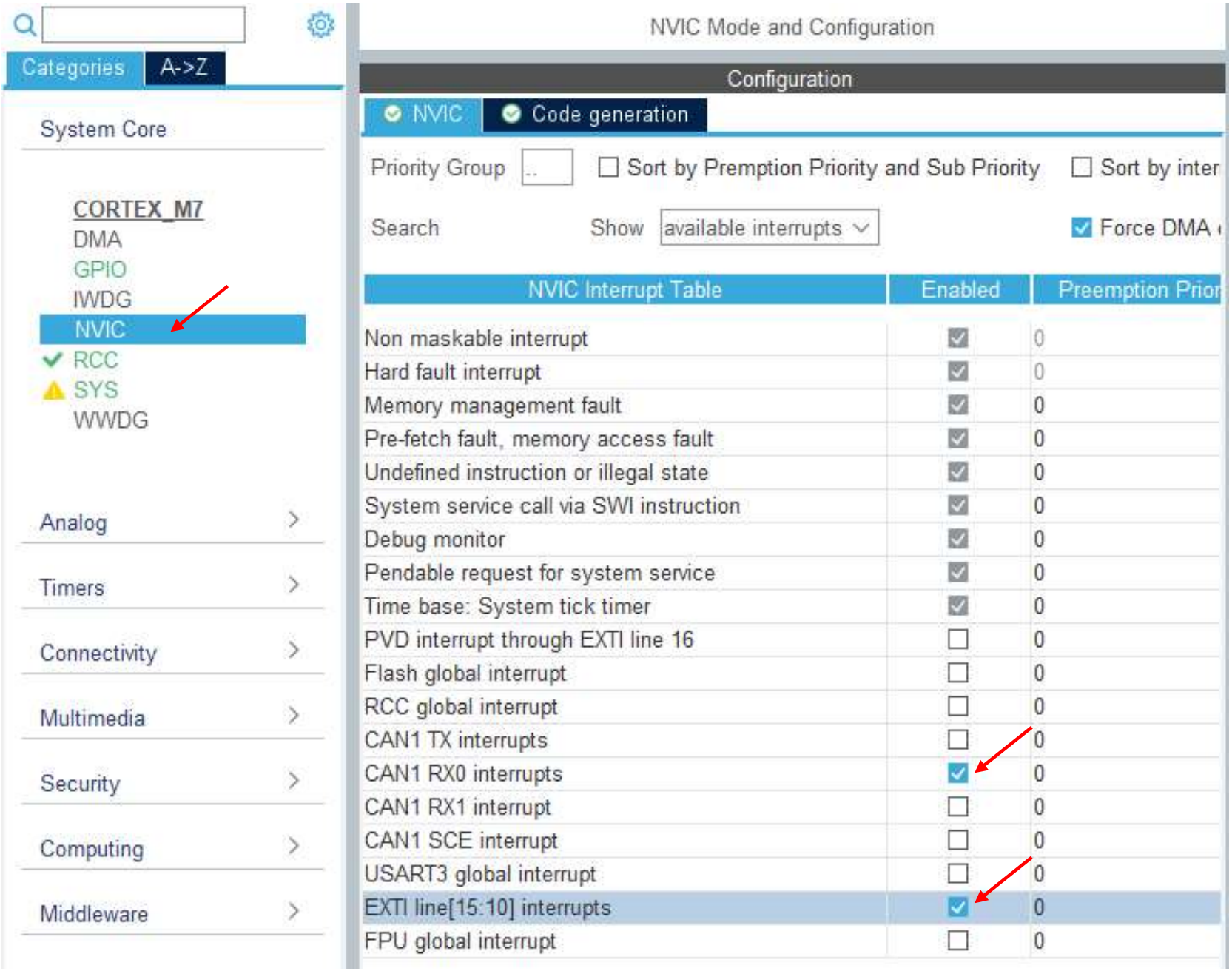
Hands-On CAN Networking

CAN Configuration: GPIO and NVIC Settings

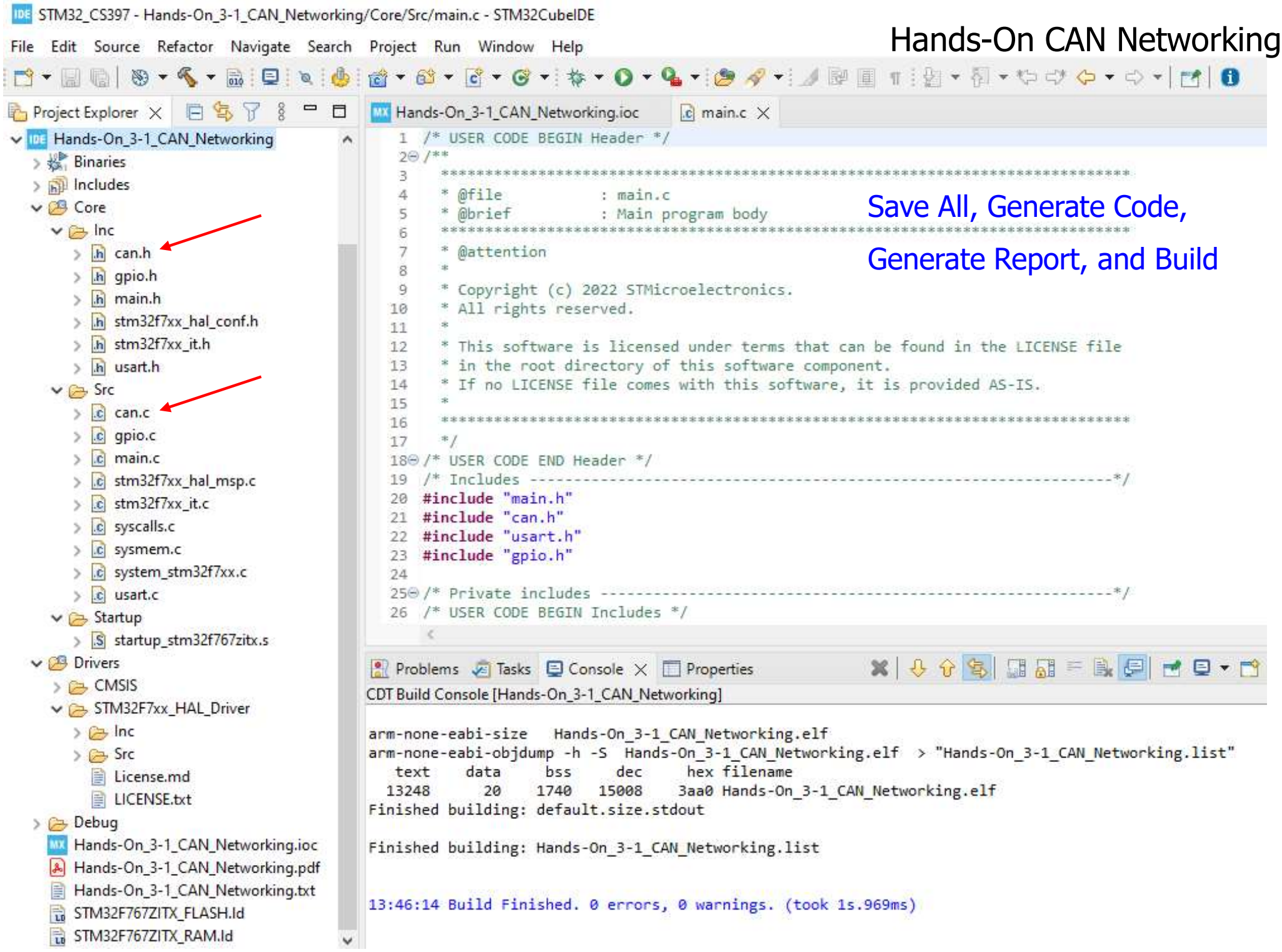




Check NVIC


Categories
A->Z
System Core
CORTEX_M7
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Multimedia
Security
Computing
Middleware
NVIC Mode and Configuration
Configuration
NVIC
Code generation
Priority Group
Sort by Premption Priority and Sub Priority
Sort by inter
Search
Show
available interrupts
Force DMA
NVIC Interrupt Table
Enabled
Preemption Prior
Non maskable interrupt 0
Hard fault interrupt 0
Memory management fault 0
Pre-fetch fault, memory access fault 0
Undefined instruction or illegal state 0
System service call via SWI instruction 0
Debug monitor 0
Pendable request for system service 0
Time base: System tick timer 0
PVD interrupt through EXTI line 16 0
Flash global interrupt 0
RCC global interrupt 0
CAN1 TX interrupts 0
CAN1 RX0 interrupts 0
CAN1 RX1 interrupt 0
CAN1 SCE interrupt 0
USART3 global interrupt 0
EXTI line[15:10] interrupts 0
FPU global interrupt 0

Hands-On CAN Networking



Save All, Generate Code,

Generate Report, and Build

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_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();
  }
}
```

```
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);
    /* CAN1 interrupt Init */
    HAL_NVIC_SetPriority(CAN1_RX0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(CAN1_RX0_IRQn);
  }
}

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);
    /* CAN1 interrupt Deinit */
    HAL_NVIC_DisableIRQ(CAN1_RX0_IRQn);
  }
}
```

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;

/* 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__ */
```

IRQHandler functions generated in **stm32f7xx_it.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

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

in

stm32f7xx_hal_can.c

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

```
/**
  * @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_can.h

```
/* Receive Interrupts */
#define CAN_IT_RX_FIFO0_MSG_PENDING
          ((uint32_t)CAN_IER_FMPIE0)
```

Interrupt enable register

FIFO message pending Interrupt enable

stm32f7xx_hal_gpio.c

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

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

```
/* 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 */
void CAN_Config(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();
  }

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

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

  /* #4 Configure Transmission process */
  TxHeader.StdId = 0x321;
  TxHeader.ExtId = 0x01FFFF;
  TxHeader.IDE = CAN_ID_STD;
  TxHeader.RTR = CAN_RTR_DATA;
  TxHeader.DLC = 2;
  TxHeader.TransmitGlobalTime = DISABLE;
}
```

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

```
/* Rx Fifo 0 message pending callback */
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
  /* Get RX message */
  if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader, RxData) != HAL_OK)
  {
    /* Reception Error */
    Error_Handler();
  }
}
```

Add Code to **can.c** and **can.h** (3/3)

```
/* Turns ON/OFF the selected LED */
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 */
```

```
/* 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 */
void CAN_Config(void);
void LED_Display(uint8_t LedStatus);
/* USER CODE END Prototypes */
```

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 */
  CAN_Config();
  TxData[0] = 0;
  TxData[1] = 0xAA;
  /* USER CODE END 2 */

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

Add Code to **main.c**

```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
     /* Set the data to be transmitted */
     TxData[0] = TxData[0] + 1;
     TxData[1] = TxData[1] + 1;
     LED_Display(TxData[0]);

     printf("Tx1 Tx2: 0x%X  0x%X,  StdID DLC Rx1 Rx2: 0x%lX  0x%lX  0x%X  0x%X \n\r",
          TxData[0], TxData[1], RxHeader.StdId, RxHeader.DLC, RxData[0], RxData[1] );
     /* Start the Transmission process */
     if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox) != HAL_OK)
     {
         /* Transmission request Error */
         Error_Handler();
     }
     if (TxData[0] >= 3)
     {
         TxData[0] = 0;
         TxData[1] = 0xAA;
     }
     HAL_Delay(1000);

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

Add Code to **main.c** between /* USER CODE BEGIN 4 */ and /* USER CODE END 4 */

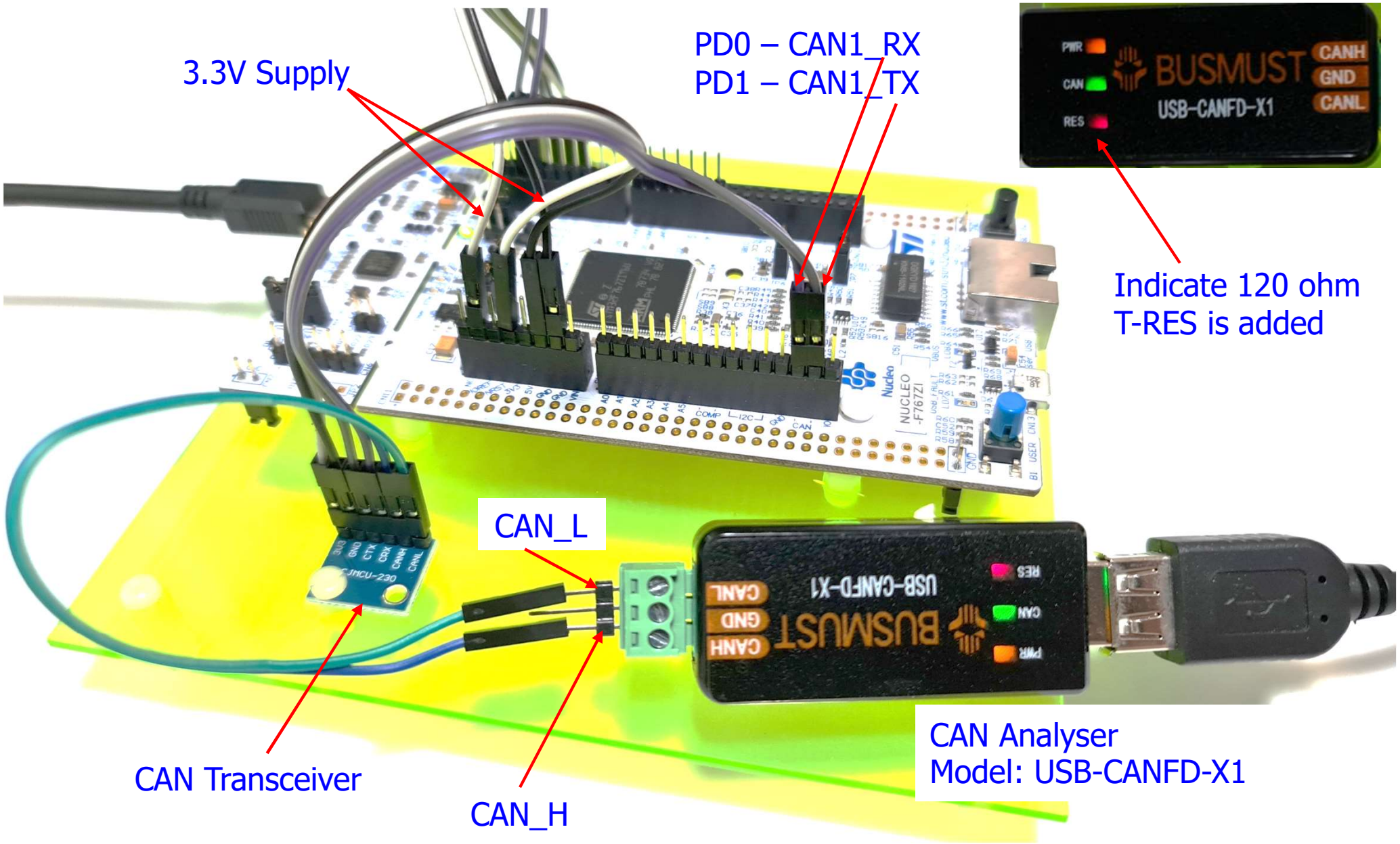
```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
    }
}

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 Networking

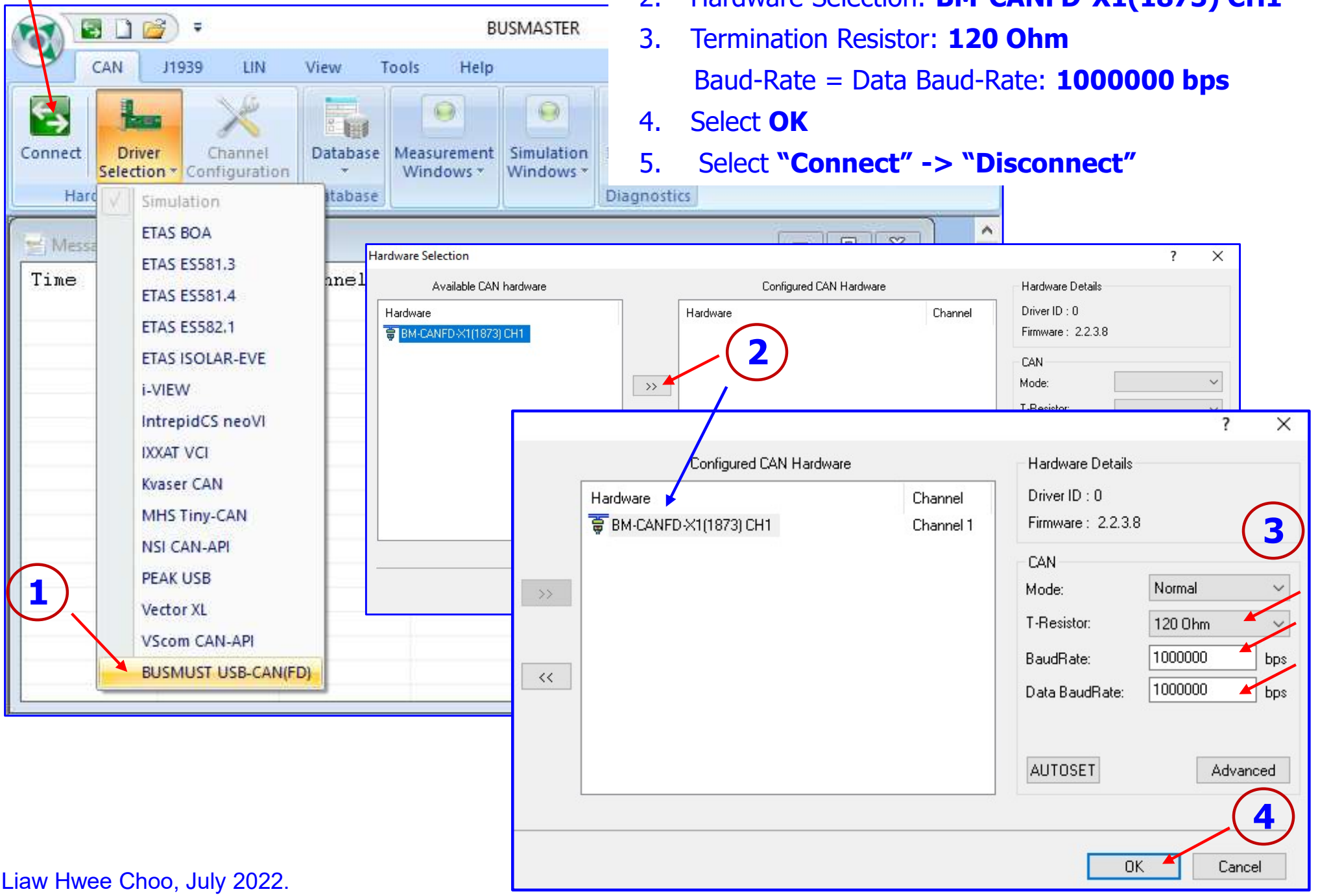
Connect CAN Analyzer to Nucleo-F767ZI via CAN Transceiver



Hands-On CAN Networking

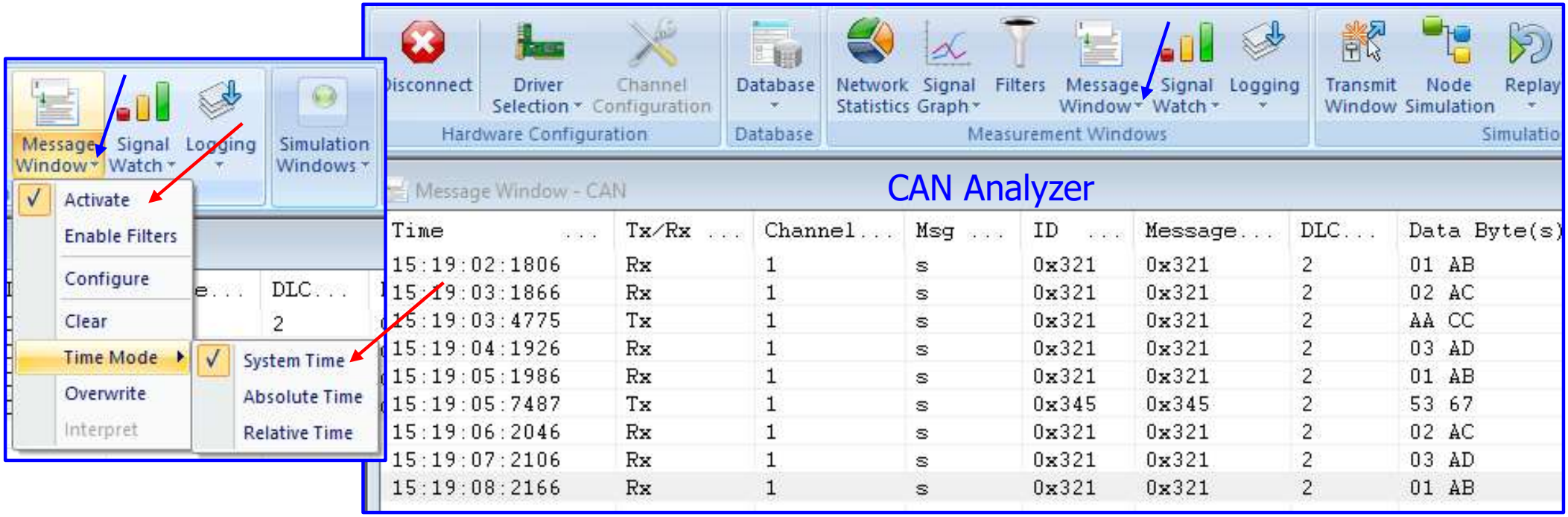
BUSMATER Software Settings

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"**

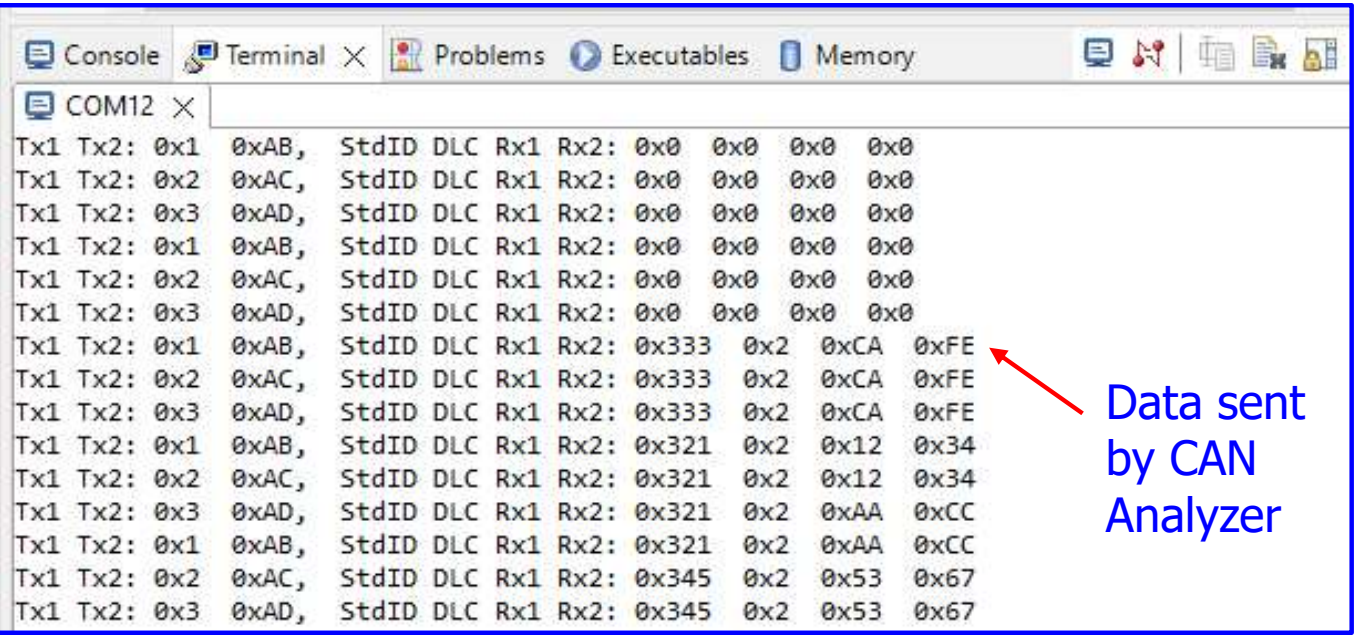


Hands-On CAN Networking

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

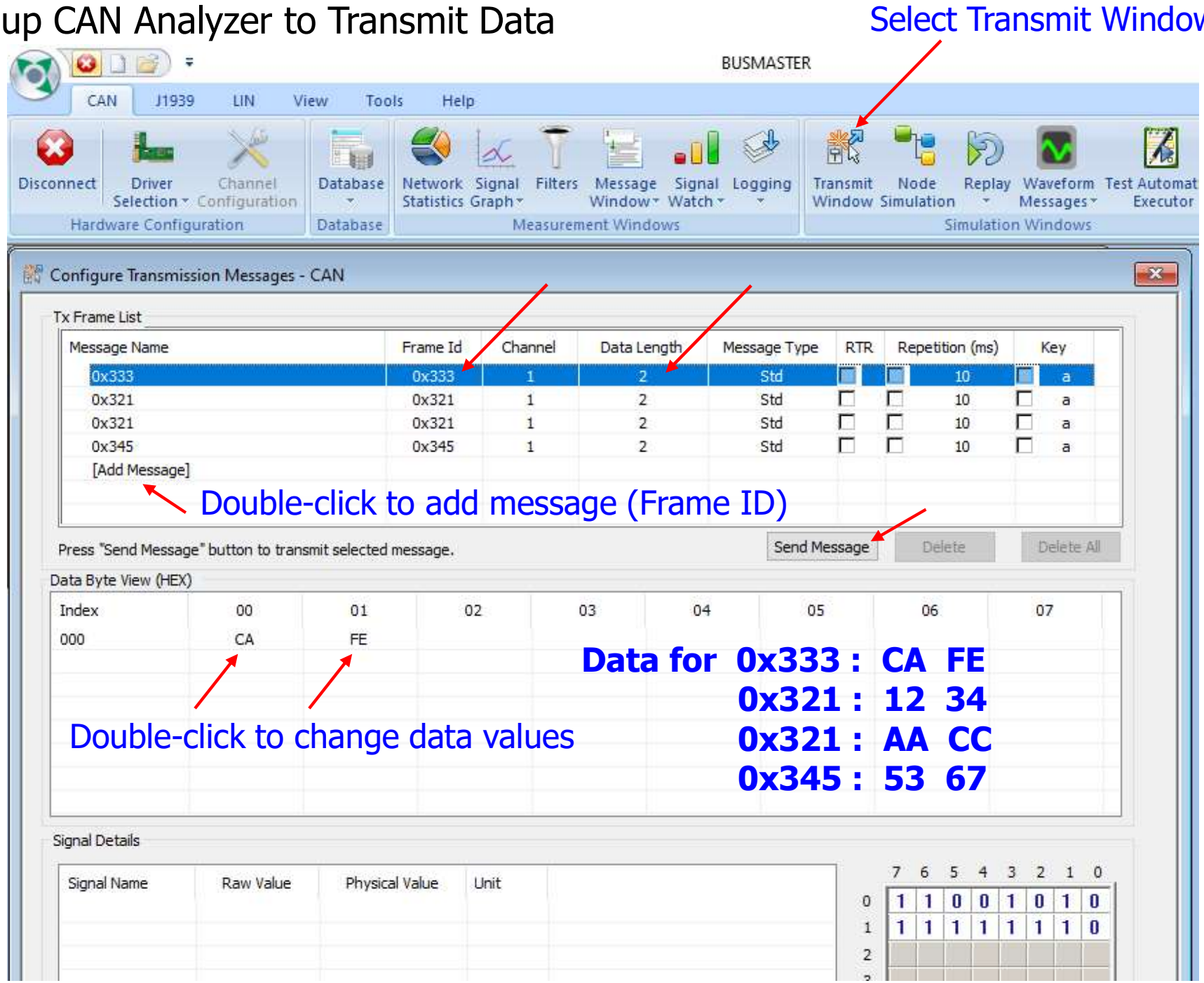


TM Terminal



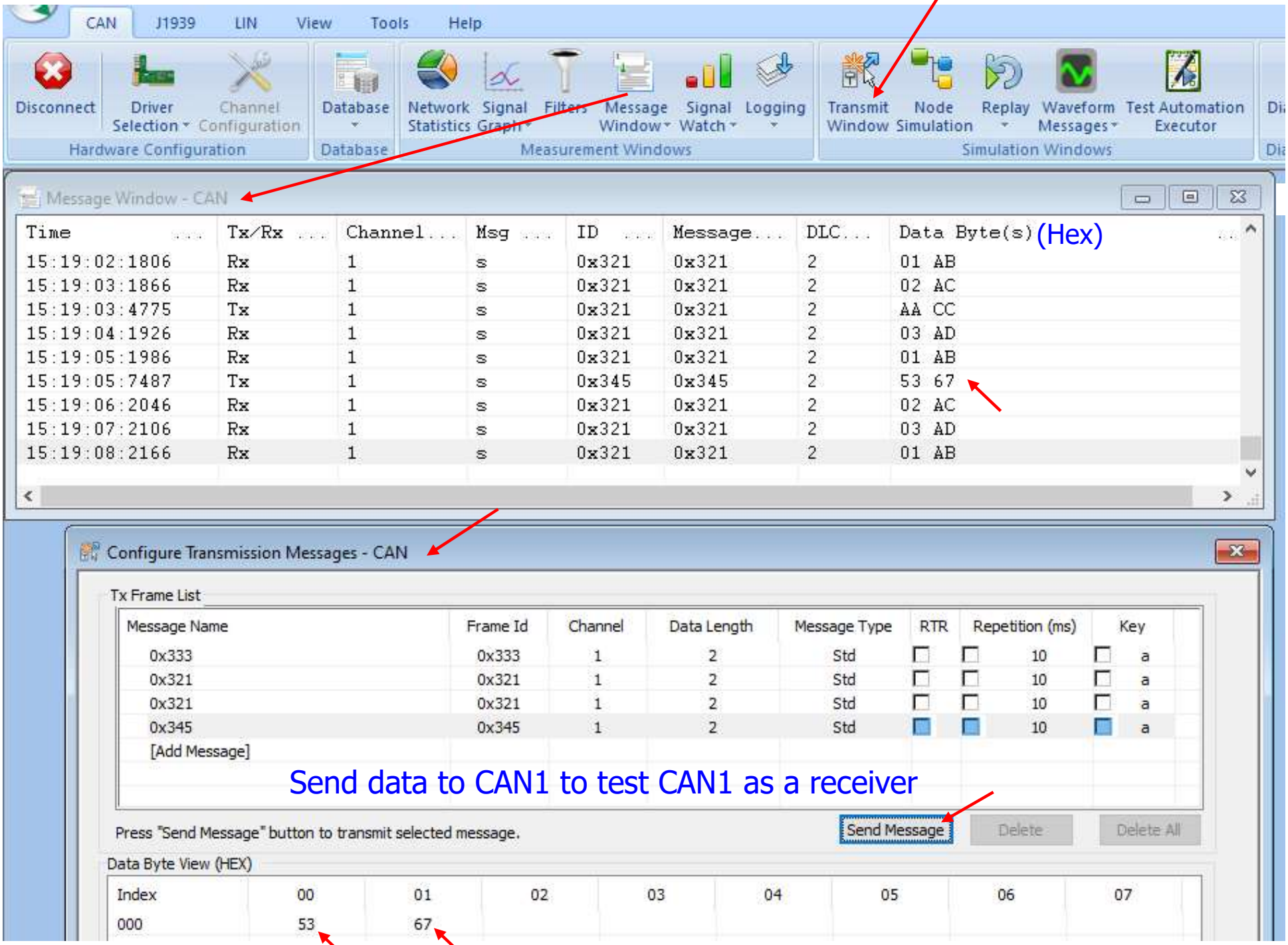
Data sent by CAN Analyzer

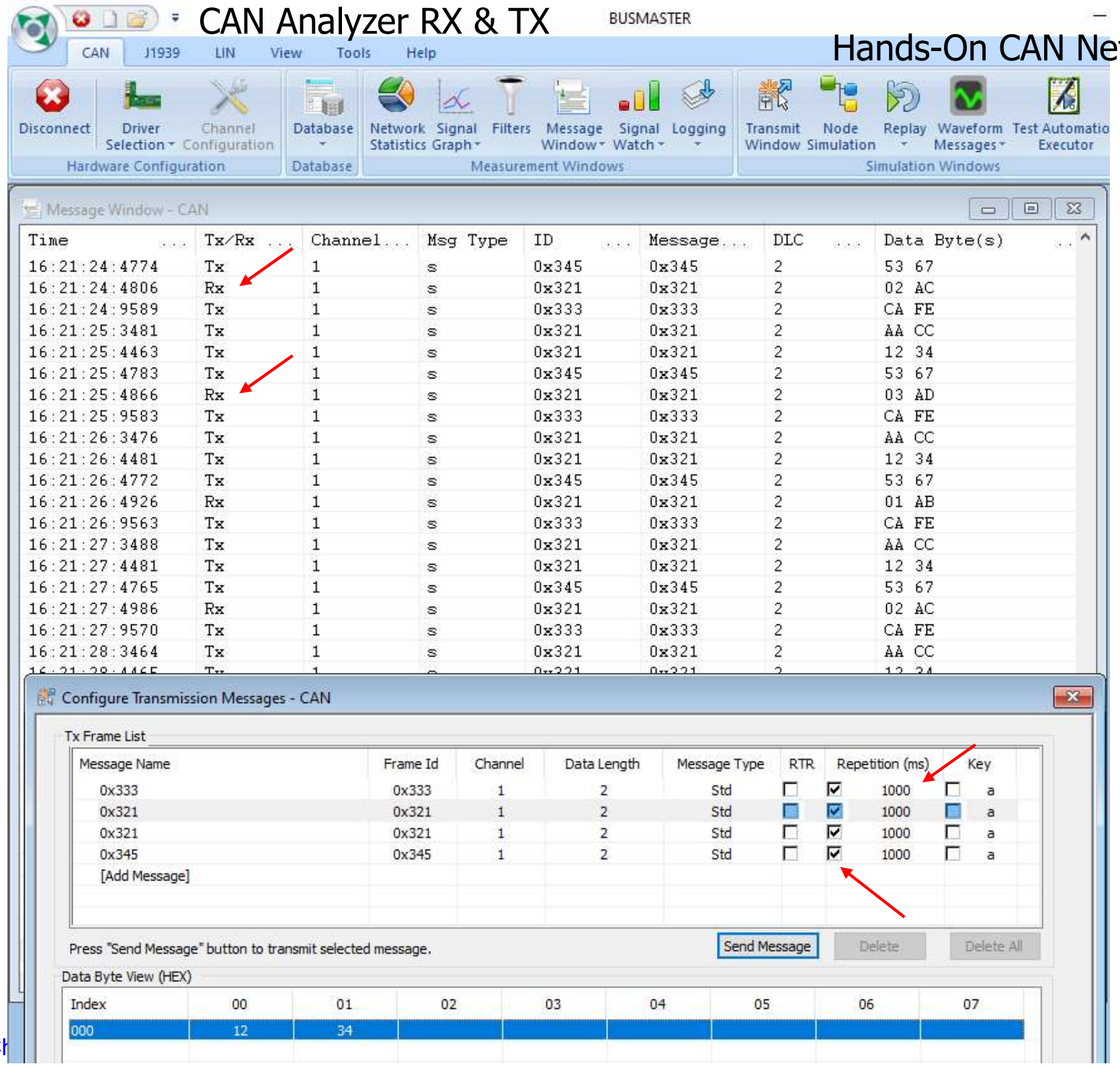
Set-up CAN Analyzer to Transmit Data



CAN Analyzer RX & TX

Hands-On CAN Networking




BUSMASTER
CAN
J1939
LIN
View
Tools
Help
Disconnect
Driver Selection
Channel Configuration
Hardware Configuration
Database
Database
Network Statistics
Signal Graph
Filters
Message Window
Signal Watch
Logging
Measurement Windows
Transmit Window
Node Simulation
Replay
Waveform Messages
Test Automatio Executor
Simulation Windows
Message Window - CAN
Time
Tx/Rx
Channel
Msg Type
ID
Message
DLC
Data Byte(s)
16:21:24:4774 Tx 1 s 0x345 0x345 2 53 67
16:21:24:4806 Rx 1 s 0x321 0x321 2 02 AC
16:21:24:9589 Tx 1 s 0x333 0x333 2 CA FE
16:21:25:3481 Tx 1 s 0x321 0x321 2 AA CC
16:21:25:4463 Tx 1 s 0x321 0x321 2 12 34
16:21:25:4783 Tx 1 s 0x345 0x345 2 53 67
16:21:25:4866 Rx 1 s 0x321 0x321 2 03 AD
16:21:25:9583 Tx 1 s 0x333 0x333 2 CA FE
16:21:26:3476 Tx 1 s 0x321 0x321 2 AA CC
16:21:26:4481 Tx 1 s 0x321 0x321 2 12 34
16:21:26:4772 Tx 1 s 0x345 0x345 2 53 67
16:21:26:4926 Rx 1 s 0x321 0x321 2 01 AB
16:21:26:9563 Tx 1 s 0x333 0x333 2 CA FE
16:21:27:3488 Tx 1 s 0x321 0x321 2 AA CC
16:21:27:4481 Tx 1 s 0x321 0x321 2 12 34
16:21:27:4765 Tx 1 s 0x345 0x345 2 53 67
16:21:27:4986 Rx 1 s 0x321 0x321 2 02 AC
16:21:27:9570 Tx 1 s 0x333 0x333 2 CA FE
16:21:28:3464 Tx 1 s 0x321 0x321 2 AA CC
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]
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 12 34

Consider modifying **can.c** & **main.c**

```c
/* can.h */

/* USER CODE BEGIN 0 */
#include <string.h>
CAN_TxHeaderTypeDef   TxHeader;
CAN_RxHeaderTypeDef   RxHeader;
uint8_t               TxData[8] = {0};
uint8_t               RxData[8] = {0};
uint32_t              TxMailbox;
/* USER CODE END 0 */
```

```c
/* can.c */

/* Rx Fifo 0 message pending callback */
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    RxHeader.StdId = 0x0;
    RxHeader.ExtId = 0x0;
    memset(RxData, 0, sizeof(RxData));

    /* Get RX message */
    if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader, RxData) != HAL_OK)
    {
        /* Reception Error */
        Error_Handler();
    }
}
```

```c
/* main.c */

printf("Tx1 Tx2: 0x%X  0x%X,  StdID ExtID DLC Rx1 Rx2: 0x%lX / 0x%lX : 0x%lX  0x%X  0x%X \n\r",
           TxData[0], TxData[1], RxHeader.StdId, RxHeader.ExtId, RxHeader.DLC, RxData[0], RxData[1] );
```