Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 2-1
CAN TX (Controller Area Network, Transmit Data)

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Hands-On CAN TX

Objectives

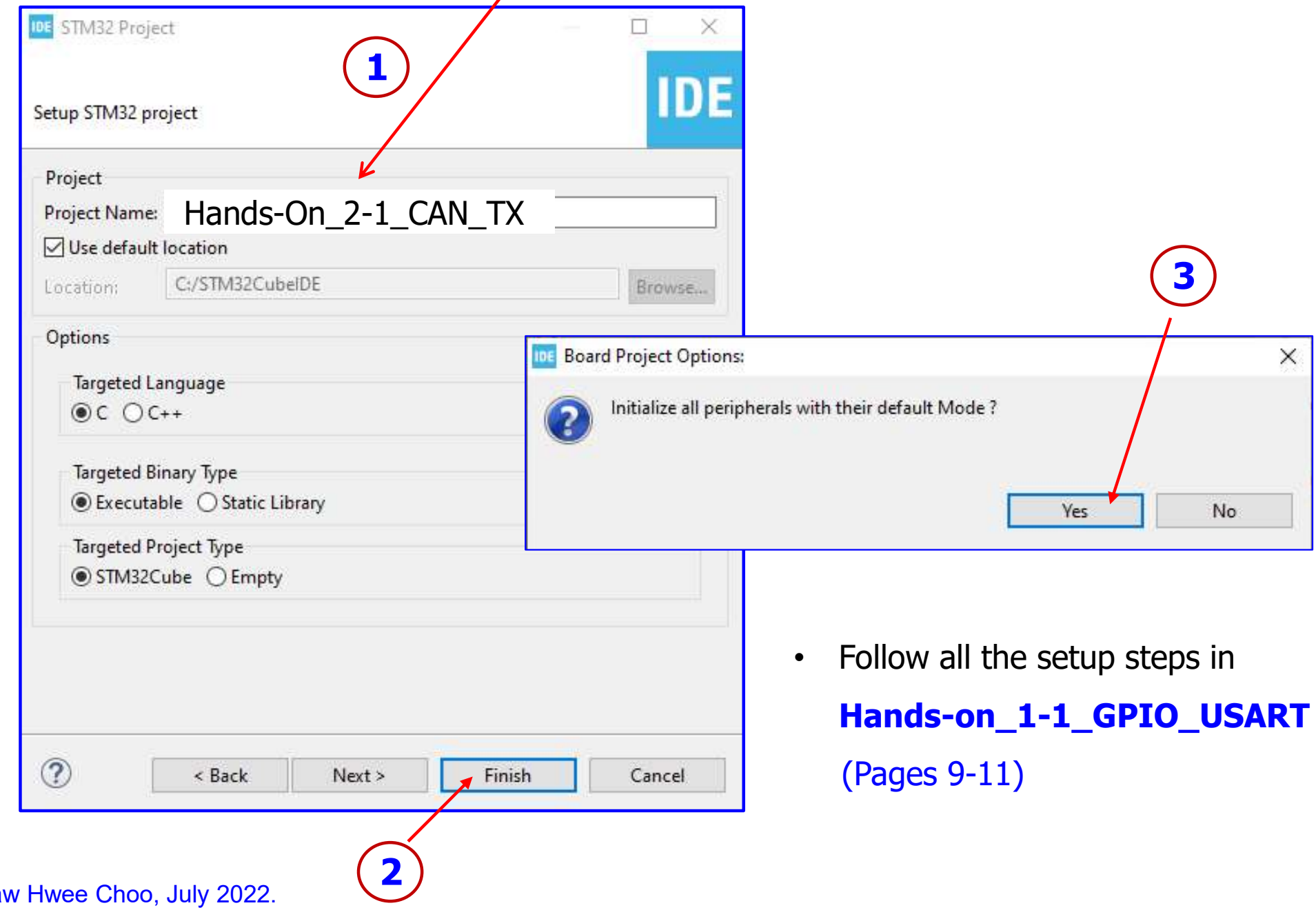
The aims of this session are to

- implement a STM32 (STM32CubeIDE) project
- develop a CAN (Controller Area Network) application using STM32F767ZI microcontroller
- program and test the CAN for transmitting data
- use of a CAN analyzer to evaluate the CAN communication
- 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 TX

Enter Project Name: Hands-On_2-1_CAN_TX



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

Hands-On CAN TX

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

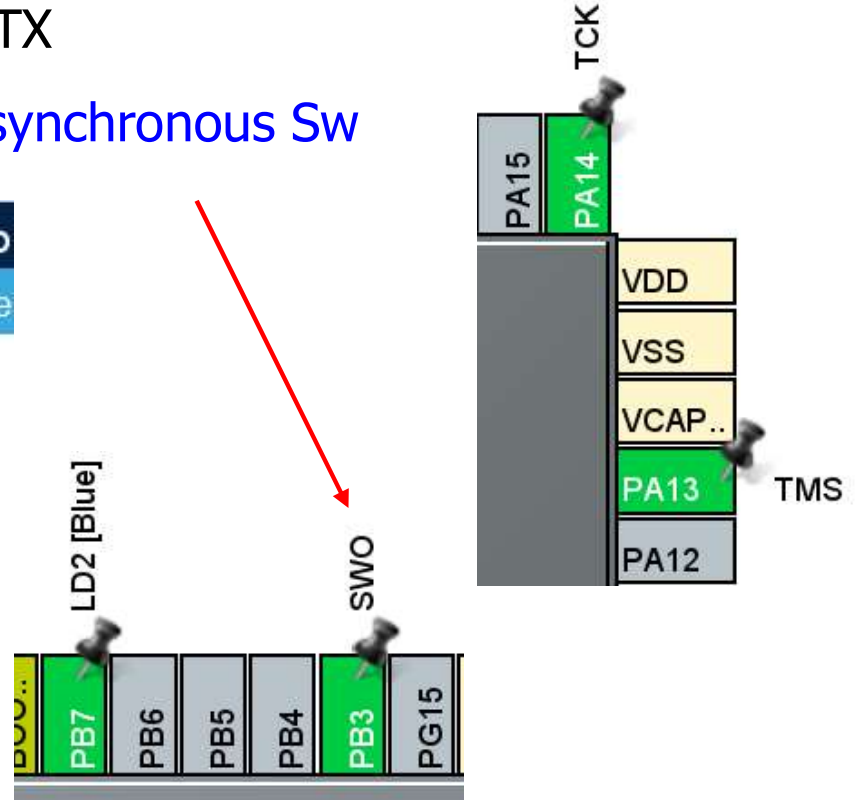
Mode

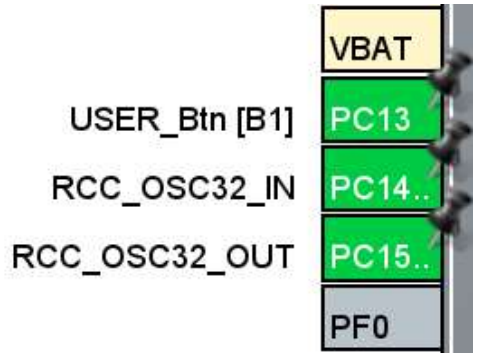
Debug: Serial Wire

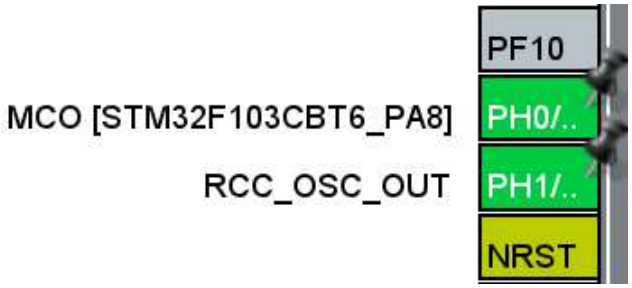
- Serial Wire (default)
- JTAG (4 pins)
- JTAG (5 pins)
- Trace Asynchronous Sw
- JTAG with Trace Synchro(1 bit)
- JTAG with Trace Synchro(2 bits)
- JTAG with Trace Synchro(4 bits)
- Trace Synchro Sw (1 bit)

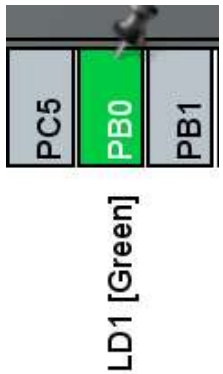
System Wake-Up 4

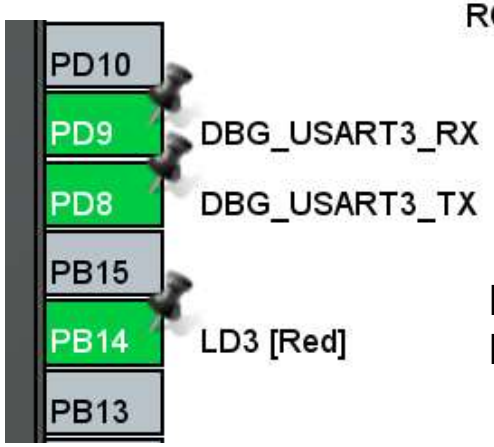
Timebase Source: SysTick











PD9: DBG_USART3_RX
PD8: DBG_USART3_TX

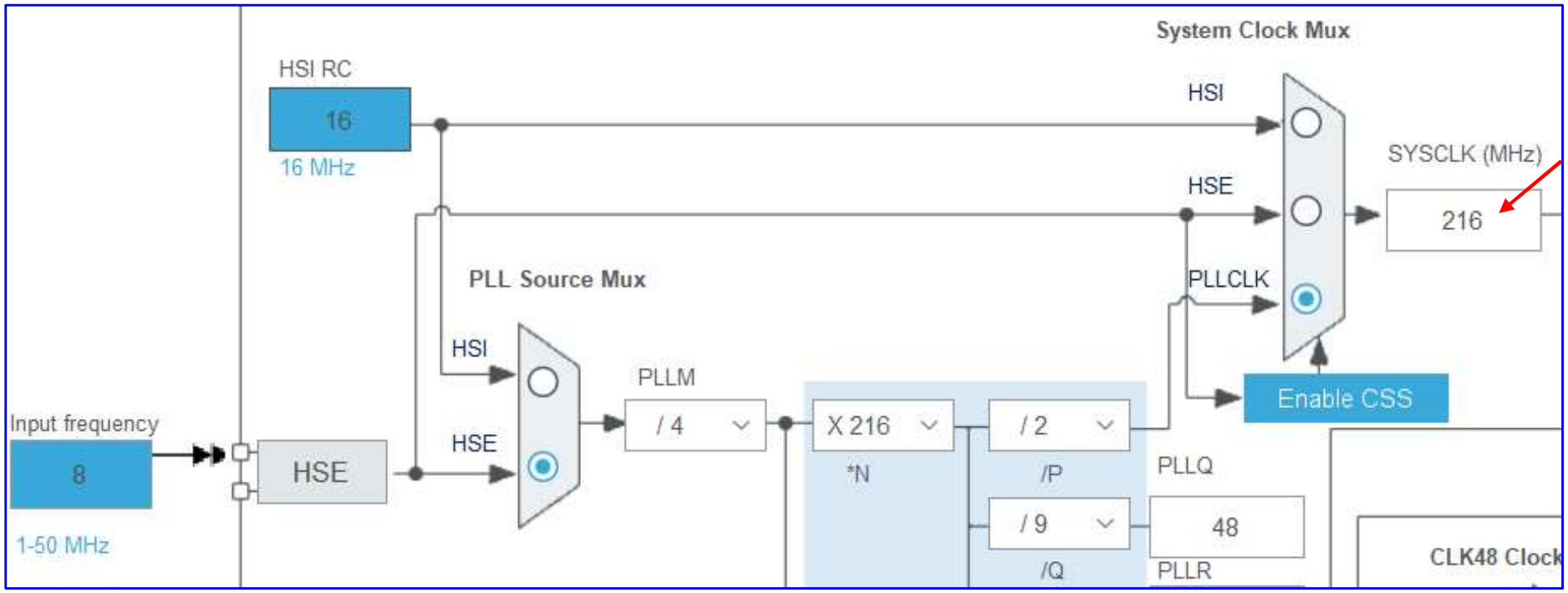
TCK (Test Clock)
TMS (Test Mode Select)
SWO (Single/Serial Wire Output)

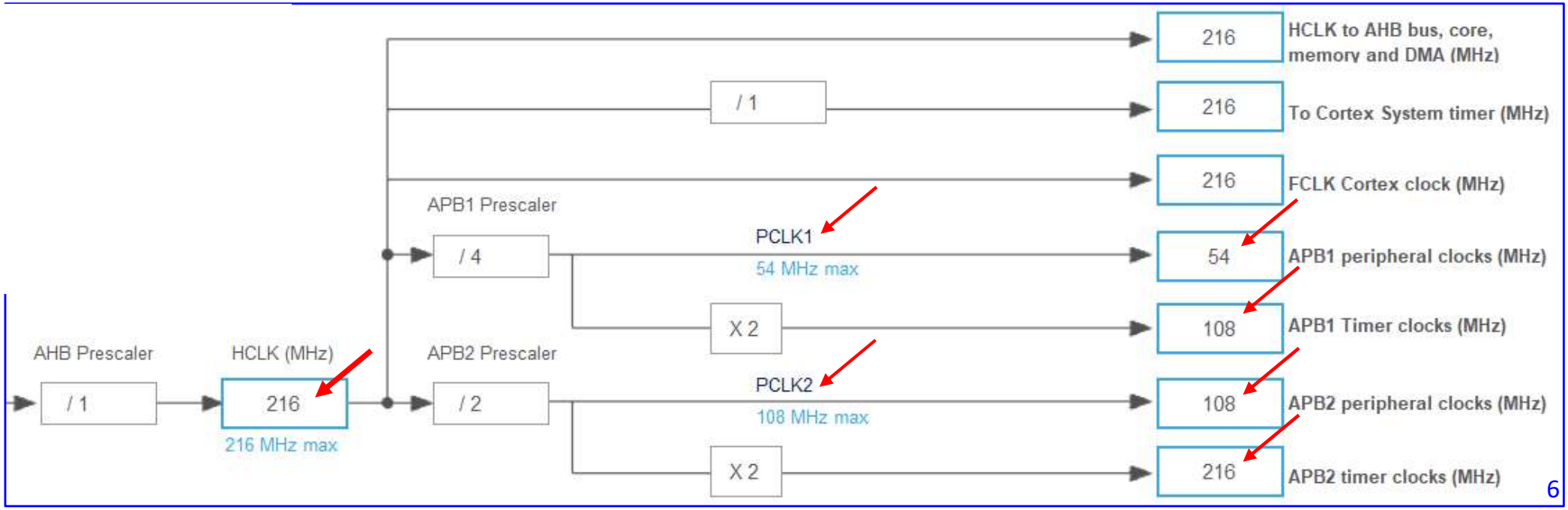
After re-set the unused pins

Hands-On CAN TX

Hands-On CAN TX

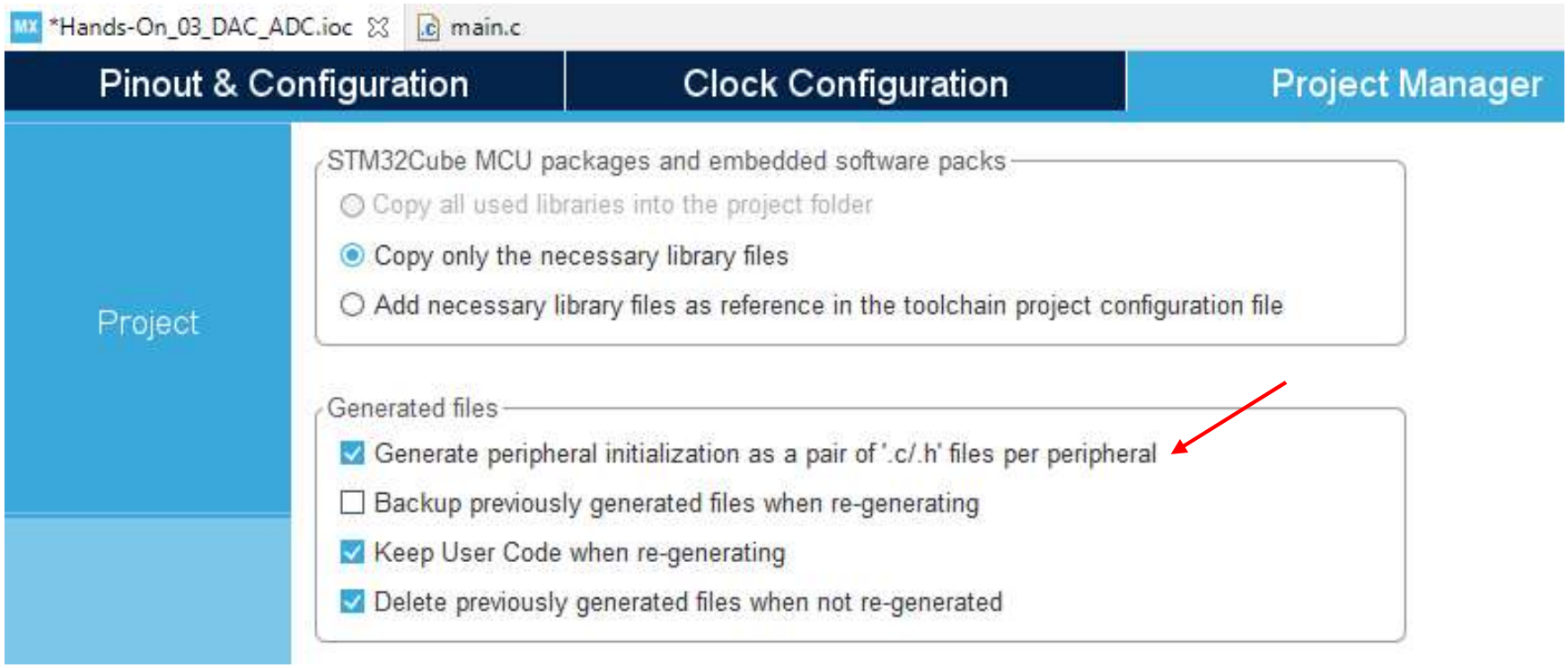
Clock Configuration: Use maximum frequency for clock settings





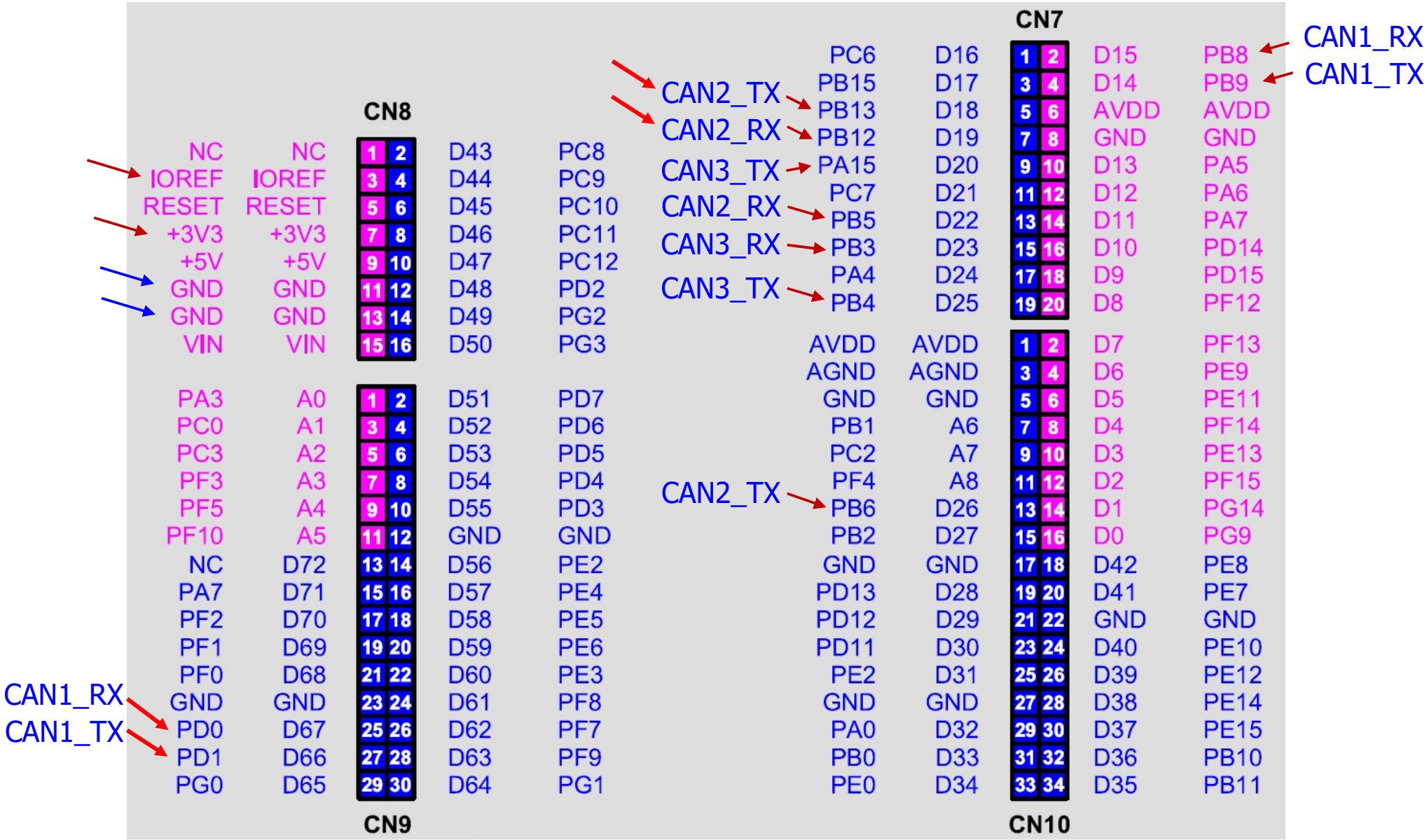
Hands-On CAN TX

- 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 TX

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 TX

CAN Configuration: select CAN1, Activated, enter values: 27, 11, 4, 4, shown below

Categories | A->Z

System Core
Analog
Timers
Connectivity

CAN1
CAN2
CAN3
ETH
FMC
I2C1
I2C2
I2C3
I2C4
MDIOS
QUADSPI
SDMMC1
SDMMC2
SPI1
SPI2
SPI3
SPI4
SPI5
SPI6
UART4
UART5

CAN1 Mode and Configuration

Mode

Activated

For Baud Rate = 125 kbps

Configuration

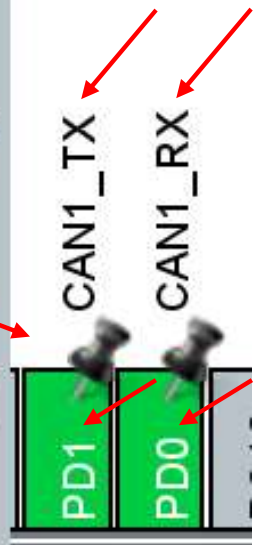
Reset Configuration

NVIC Settings | GPIO Settings
Parameter Settings | User Constants

Configure the below parameters :

Search (Crtl+F)

Bit Timings Parameters	
Prescaler (for Time Quantum)	27
Time Quantum	500.0 ns
Time Quanta in Bit Segment 1	11 Times
Time Quanta in Bit Segment 2	4 Times
Time for one Bit	8000.00 ns
Baud Rate	125000 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



Hands-On CAN TX

Save All and Build

STM32_CS397 - Hands-On_2-1_CAN_TX/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Hands-On_2-1_CAN_TX
  - 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
  - Debug
  - Hands-On_2-1_CAN_TX.ioc
  - Hands-On_2-1_CAN_TX.pdf
  - Hands-On_2-1_CAN_TX.txt
  - STM32F767ZITX_FLASH.ld
  - STM32F767ZITX_RAM.ld

Report

Hands-On_2-1_CAN_TX.ioc | main.c

```
1  /* USER CODE BEGIN Header */
2  /**
3    ******************************************************************************
4    * @file           : main.c
5    * @brief          : Main program body
6    ******************************************************************************
7    * @attention
8    *
9    * Copyright (c) 2022 STMicroelectronics.
10   * All rights reserved.
11   *
12   * This software is licensed under terms that can be found in the LICENSE file
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"
```

Problems | Tasks | Console | Properties

CDT Build Console [Hands-On_2-1_CAN_TX]

```
arm-none-eabi-size   Hands-On_2-1_CAN_TX.elf
arm-none-eabi-objdump -h -S  Hands-On_2-1_CAN_TX.elf  > "Hands-On_2-1_CAN_TX.list"
   text    data     bss     dec     hex filename
  12076      20    1740   13836    360c Hands-On_2-1_CAN_TX.elf
Finished building: default.size.stdout

Finished building: Hands-On_2-1_CAN_TX.list


12:36:10 Build Finished. 0 errors, 0 warnings. (took 3s.44ms)
```

Hands-On CAN TX

Generated **can.c**

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

CAN_HandleTypeDef hcan1;

/* CAN1 init function */
void MX_CAN1_Init(void)
{
  hcan1.Instance = CAN1;
  hcan1.Init.Prescaler = 27;
  hcan1.Init.Mode = CAN_MODE_NORMAL;
  hcan1.Init.SyncJumpWidth = CAN_SJW_4TQ;
  hcan1.Init.TimeSeg1 = CAN_BS1_11TQ;
  hcan1.Init.TimeSeg2 = CAN_BS2_4TQ;
  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
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);
  }
}
```

stm32f7xx_hal_can.h

CAN_InitTypeDef

```c
FunctionalState AutoRetransmission;
/*!< Enable or disable the non-automatic retransmission mode.
     This parameter can be set to ENABLE or DISABLE. */
```

stm32f7xx_hal_can.h

Hands-On CAN TX

```
/* @brief  CAN handle Structure definition  */

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);/*!< CAN Tx Mailbox 0 complete callback */
  void (* TxMailbox1CompleteCallback)(struct __CAN_HandleTypeDef *hcan);/*!< CAN Tx Mailbox 1 complete callback */
  void (* TxMailbox2CompleteCallback)(struct __CAN_HandleTypeDef *hcan);/*!< CAN Tx Mailbox 2 complete callback */

  void (* TxMailbox0AbortCallback)(struct __CAN_HandleTypeDef *hcan);   /*!< CAN Tx Mailbox 0 abort callback    */
  void (* TxMailbox1AbortCallback)(struct __CAN_HandleTypeDef *hcan);   /*!< CAN Tx Mailbox 1 abort callback    */
  void (* TxMailbox2AbortCallback)(struct __CAN_HandleTypeDef *hcan);   /*!< CAN Tx Mailbox 2 abort callback    */

  void (* RxFifo0MsgPendingCallback)(struct __CAN_HandleTypeDef *hcan); /*!< CAN Rx FIFO 0 msg pending callback */
  void (* RxFifo0FullCallback)(struct __CAN_HandleTypeDef *hcan);       /*!< CAN Rx FIFO 0 full callback        */

  void (* RxFifo1MsgPendingCallback)(struct __CAN_HandleTypeDef *hcan); /*!< CAN Rx FIFO 1 msg pending callback */
  void (* RxFifo1FullCallback)(struct __CAN_HandleTypeDef *hcan);       /*!< CAN Rx FIFO 1 full callback        */

  void (* SleepCallback)(struct __CAN_HandleTypeDef *hcan);             /*!< CAN Sleep callback                 */
  void (* WakeUpFromRxMsgCallback)(struct __CAN_HandleTypeDef *hcan);   /*!< CAN Wake Up from Rx msg callback   */

  void (* ErrorCallback)(struct __CAN_HandleTypeDef *hcan);             /*!< CAN Error callback                 */

  void (* MspInitCallback)(struct __CAN_HandleTypeDef *hcan);           /*!< CAN Msp Init callback             */
  void (* MspDeInitCallback)(struct __CAN_HandleTypeDef *hcan);         /*!< CAN Msp DeInit callback            */

#endif /* (USE_HAL_CAN_REGISTER_CALLBACKS) */
} CAN_HandleTypeDef;
```

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"

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 TX

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

```c
/* can.c */
/* USER CODE BEGIN 0 */
CAN_TxHeaderTypeDef   TxHeader;
/* 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 Configure Transmission process */
    TxHeader.StdId = 0x321;
    TxHeader.ExtId = 0x01;
    TxHeader.IDE = CAN_ID_STD;
    TxHeader.RTR = CAN_RTR_DATA;
    TxHeader.DLC = 2;
    TxHeader.TransmitGlobalTime = DISABLE;
}
/* USER CODE END 1 */
```

```c
/* can.h */
/* USER CODE BEGIN Private defines */
extern CAN_TxHeaderTypeDef TxHeader;
/* USER CODE END Private defines */
/* USER CODE BEGIN Prototypes */
void CAN_Config(void);
/* USER CODE END Prototypes */
```

stm32f7xx_hal_can.h (Partial)

```
/**   * @brief  CAN filter configuration structure definition   */
typedef struct
{
  uint32_t FilterIdHigh;          /*!< Specifies the filter identification number
           (MSBs for a 32-bit configuration, first one for a 16-bit configuration).
    This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF. */

  uint32_t FilterIdLow;           /*!< Specifies the filter identification number
           (LSBs for a 32-bit configuration, second one for a 16-bit configuration).
    This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF. */

  uint32_t FilterMaskIdHigh;      /*!< Specifies the filter mask number or
         identification number, according to the mode (MSBs for a 32-bit configuration,
         first one for a 16-bit configuration).
    This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF. */

  uint32_t FilterMaskIdLow;       /*!< Specifies the filter mask number or
         identification number, according to the mode (LSBs for a 32-bit configuration,
         second one for a 16-bit configuration).
    This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF. */

  uint32_t FilterFIFOAssignment;  /*!< Specifies the FIFO (0 or 1U) which will be
       assigned to the filter. This parameter can be a value of @ref CAN_filter_FIFO */
```

stm32f7xx_hal_can.h (Partial)

```
  uint32_t FilterBank;            /*!< Specifies the filter bank which will be
        initialized. For single CAN instance(14 dedicated filter banks),
        this parameter must be a number between Min_Data = 0 and Max_Data = 13.
                              For dual CAN instances(28 filter banks shared),
        this parameter must be a number between Min_Data = 0 and Max_Data = 27. */

  uint32_t FilterMode;            /*!< Specifies the filter mode to be initialized.
                              This parameter can be a value of @ref CAN_filter_mode */

  uint32_t FilterScale;           /*!< Specifies the filter scale.
                             This parameter can be a value of @ref CAN_filter_scale */

  uint32_t FilterActivation;      /*!< Enable or disable the filter.
                         This parameter can be a value of @ref CAN_filter_activation */

  uint32_t SlaveStartFilterBank;  /*!< Select the start filter bank for the slave CAN
        instance. For single CAN instances, this parameter is meaningless.
        For dual CAN instances, all filter banks with lower index are assigned to
        master CAN instance, whereas all filter banks with greater index are assigned
        to slave CAN instance.
        This parameter must be a number between Min_Data = 0 and Max_Data = 27. */

} CAN_FilterTypeDef;
```

stm32f7xx_hal_can.h (Partial)

```
/** @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 */
/**  * @}  */
```

Hands-On CAN TX

stm32f7xx_hal_can.h (Partial)

```
/** @brief  CAN Tx message header structure definition */
typedef struct
{
  uint32_t StdId;  /*!< Specifies the standard identifier.
       This parameter must be a number between Min_Data = 0 and Max_Data = 0x7FF. */

  uint32_t ExtId;  /*!< Specifies the extended identifier.
       This parameter must be a number between Min_Data = 0 and Max_Data = 0x1FFFFFFF. */

  uint32_t IDE;    /*!< Specifies the type of identifier for the message that will be
                 transmitted. This parameter can be a value of @ref CAN_identifier_type */

  uint32_t RTR;    /*!< Specifies the type of frame for the message that will be
    transmitted. This parameter can be a value of @ref CAN_remote_transmission_request */

  uint32_t DLC;      /*!< Specifies the length of the frame that will be transmitted.
                 This parameter must be a number between Min_Data = 0 and Max_Data = 8. */

  FunctionalState TransmitGlobalTime; /*!< Specifies whether the timestamp counter value
captured on start of frame transmission, is sent in DATA6 and DATA7 replacing pData[6]
and pData[7].  @note: Time Triggered Communication Mode must be enabled.
               @note: DLC must be programmed as 8 bytes, in order these 2 bytes are sent.
               This parameter can be set to ENABLE or DISABLE. */

} CAN_TxHeaderTypeDef;
```

Hands-On CAN TX

stm32f7xx_hal_can.h (Partial)

```
/** @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 CAN TX

Add Code to **main.c**, USER CODE: Includes, PV, 2, and WHILE

```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 variables */
/* USER CODE BEGIN PV */
uint8_t   TxData[8] = {0};
uint32_t  TxMailbox;
uint8_t   g_nCnt = 0;
/* USER CODE END PV */

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

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

```c
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART3_UART_Init();
  MX_CAN1_Init();
  /* USER CODE BEGIN 2 */
  CAN_Config();
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      HAL_GPIO_TogglePin(GPIOB, LD1_Pin);
      TxData[0] = g_nCnt;
      g_nCnt = g_nCnt + 1;
      if( g_nCnt > 0xff ) g_nCnt = 0;
      if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader,
          TxData, &TxMailbox) != HAL_OK)
      {
          /* Transmission request Error */
          Error_Handler();
      }
    printf("vales = %d (0x%X), TxMailbox = %ld \r\n",
           TxData[0], TxData[0], TxMailbox);
      HAL_Delay(50);
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

stm32f7xx_hal_can.c (Partial)

```
  ==============================================================================
                      ##### How to use this driver #####
  ==============================================================================
    [..]
      (#) Initialize the CAN low level resources by implementing the
          HAL_CAN_MspInit():
         (++) Enable the CAN interface clock using __HAL_RCC_CANx_CLK_ENABLE()
         (++) Configure CAN pins
             (+++) Enable the clock for the CAN GPIOs
             (+++) Configure CAN pins as alternate function open-drain
         (++) In case of using interrupts (e.g. HAL_CAN_ActivateNotification())
             (+++) Configure the CAN interrupt priority using
                   HAL_NVIC_SetPriority()
             (+++) Enable the CAN IRQ handler using HAL_NVIC_EnableIRQ()
             (+++) In CAN IRQ handler, call HAL_CAN_IRQHandler()

      (#) Initialize the CAN peripheral using HAL_CAN_Init() function. This
          function resorts to HAL_CAN_MspInit() for low-level initialization.

      (#) Configure the reception filters using the following configuration
          functions:
            (++) HAL_CAN_ConfigFilter()

      (#) Start the CAN module using HAL_CAN_Start() function. At this level
          the node is active on the bus: it receive messages, and can send
          messages.
```

stm32f7xx_hal_can.c (Partial)

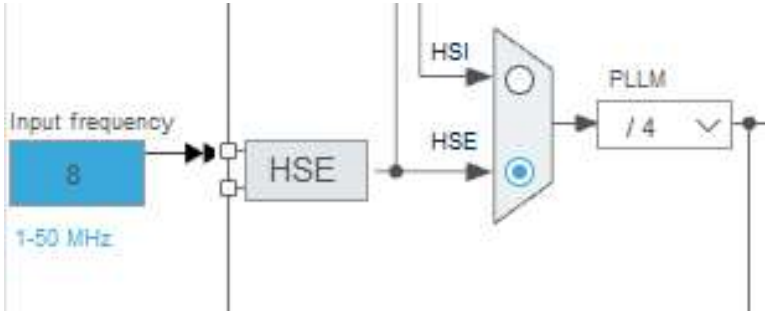
```
(#) To manage messages transmission, the following Tx control functions
    can be used:
      (++) HAL_CAN_AddTxMessage() to request transmission of a new
           message.
      (++) HAL_CAN_AbortTxRequest() to abort transmission of a pending
           message.
      (++) HAL_CAN_GetTxMailboxesFreeLevel() to get the number of free Tx
           mailboxes.
      (++) HAL_CAN_IsTxMessagePending() to check if a message is pending
           in a Tx mailbox.
      (++) HAL_CAN_GetTxTimestamp() to get the timestamp of Tx message
           sent, if time triggered communication mode is enabled.

(#) When a message is received into the CAN Rx FIFOs, it can be retrieved
    using the HAL_CAN_GetRxMessage() function. The function
    HAL_CAN_GetRxFifoFillLevel() allows to know how many Rx message are
    stored in the Rx Fifo.

(#) Calling the HAL_CAN_Stop() function stops the CAN module.

(#) The deinitialization is achieved with HAL_CAN_DeInit() function.

    .
    .
    .
```

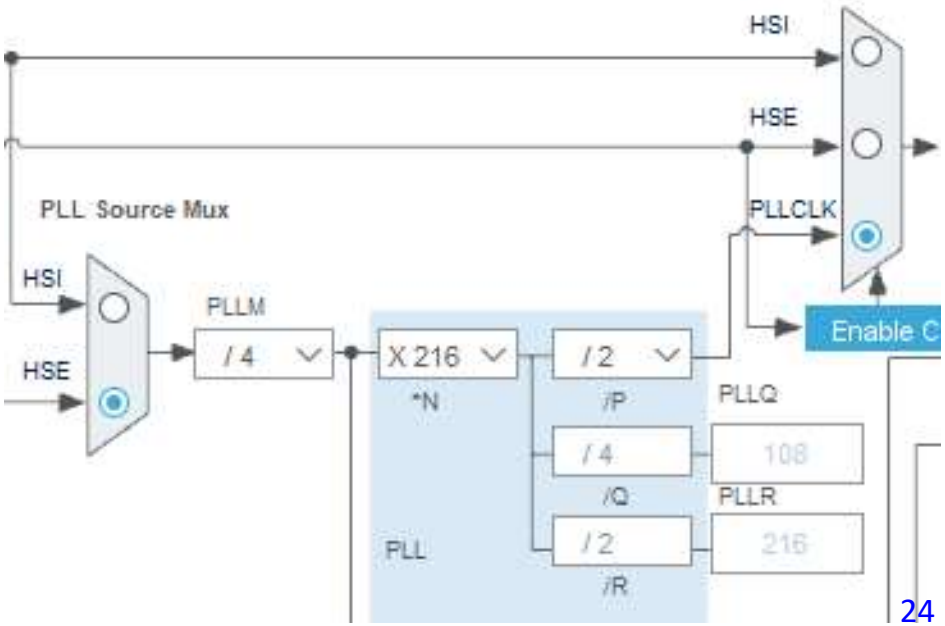
Add Code to **main.c**, USER CODE 4

```
/* 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 */
```

System Clock Configuration (generated code in **main.c)** (1/2)

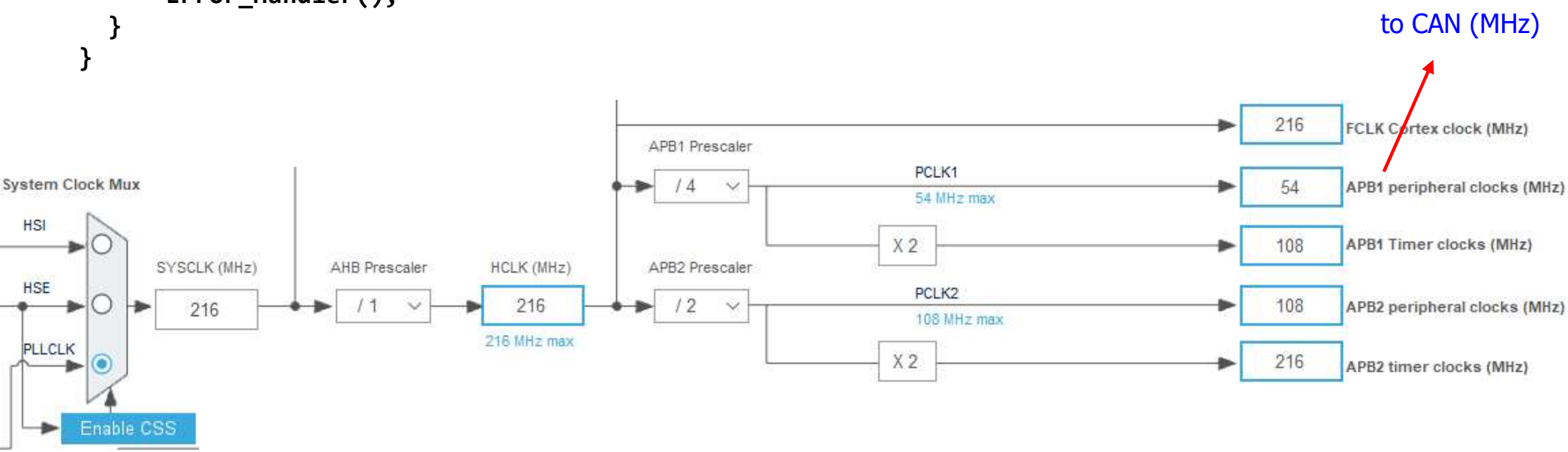
```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure LSE Drive Capability */
  HAL_PWR_EnableBkUpAccess();
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 216;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  RCC_OscInitStruct.PLL.PLLR = 2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Activate the Over-Drive mode */
```





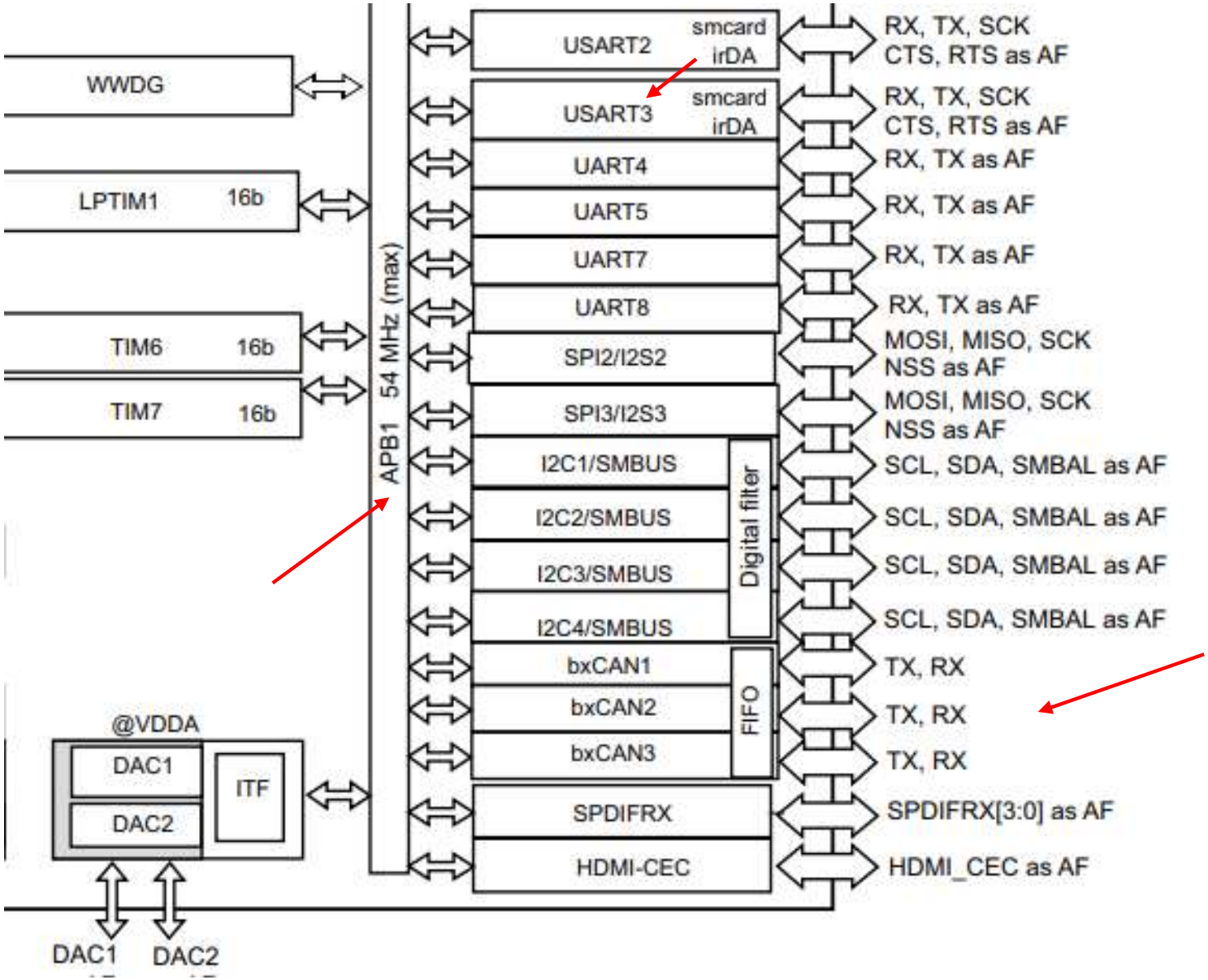
System Clock Configuration (generated code in **main.c)** (2/2)

```
/** Activate the Over-Drive mode */
  if (HAL_PWREx_EnableOverDrive() != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_7) != HAL_OK)
  {
      Error_Handler();
  }
}
```

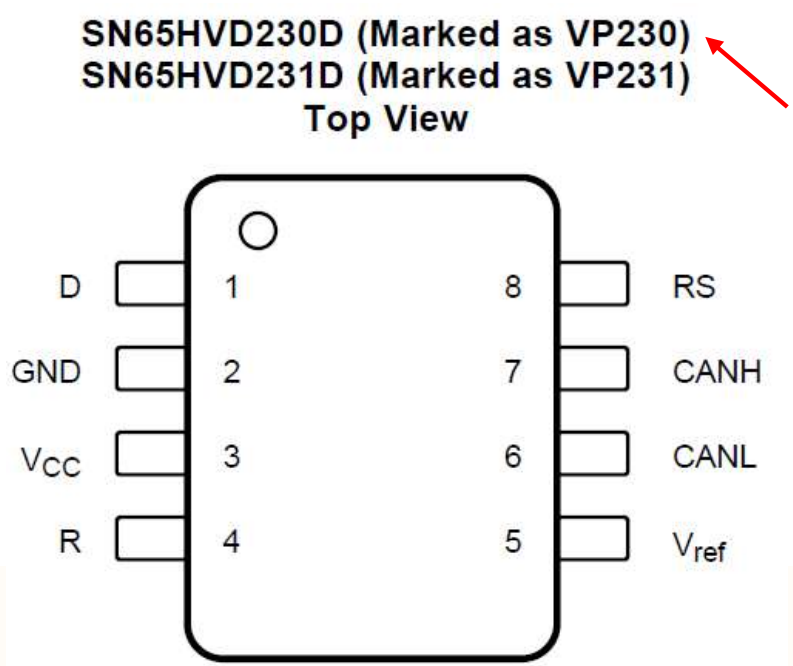


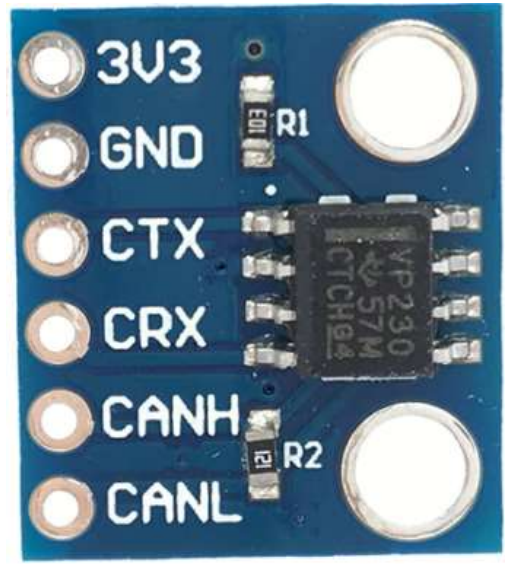
Hands-On CAN TX

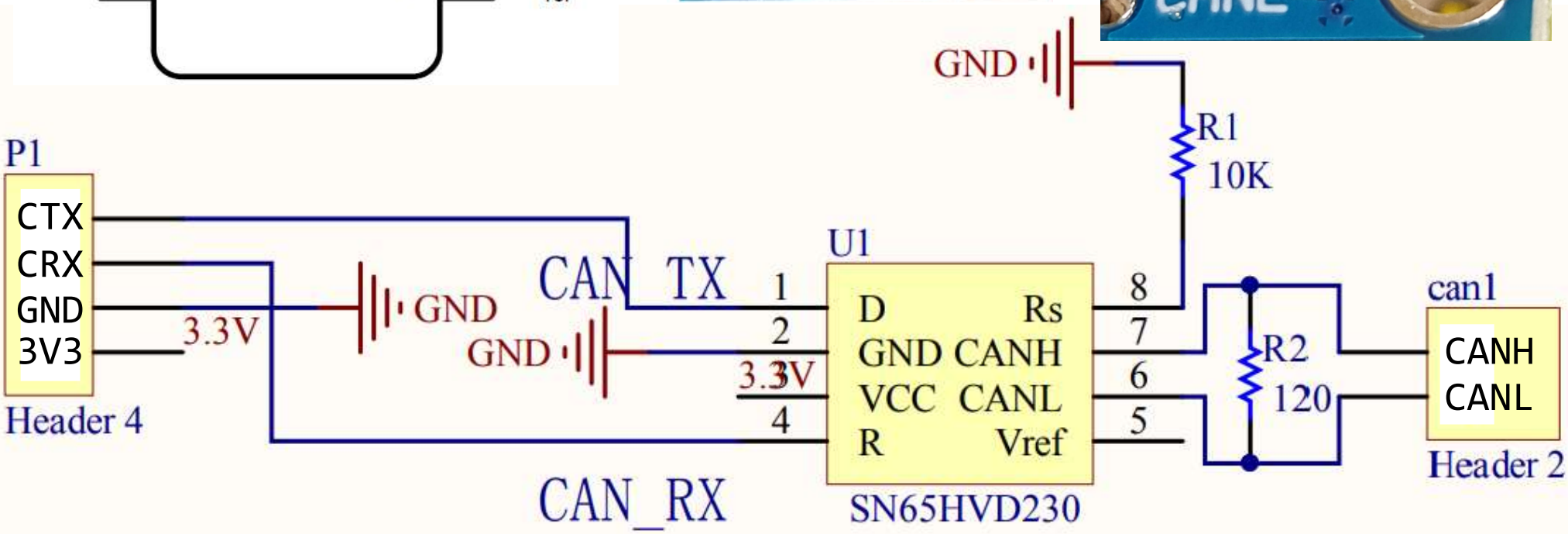
[Ref_02-1]: Figure 2, Pg. 20/252, STM32F767xx block diagram (Partial)



SN65HVD230 (VP230) [Ref_29] CAN Bus Transceiver Module

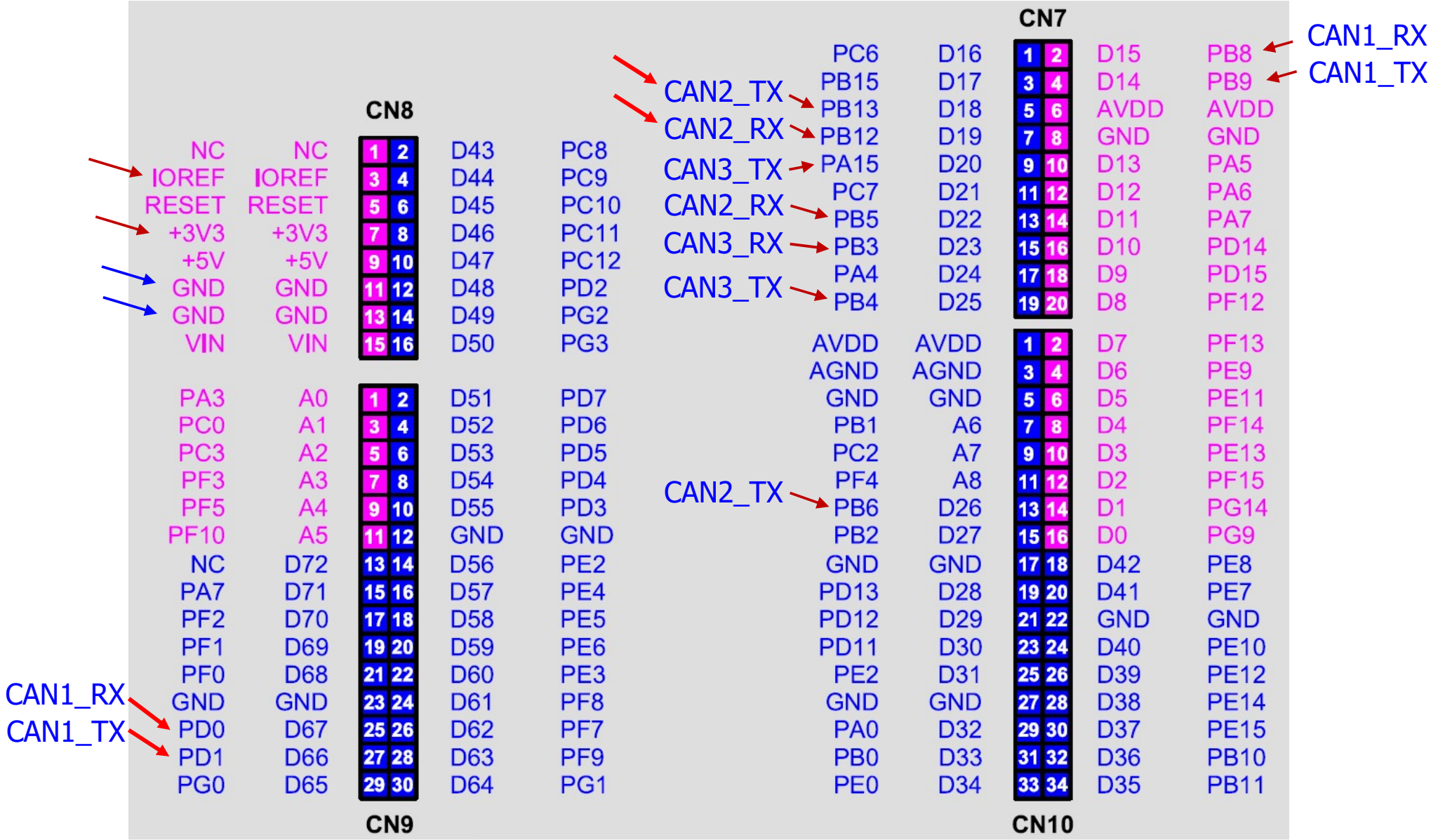






Hands-On CAN TX

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

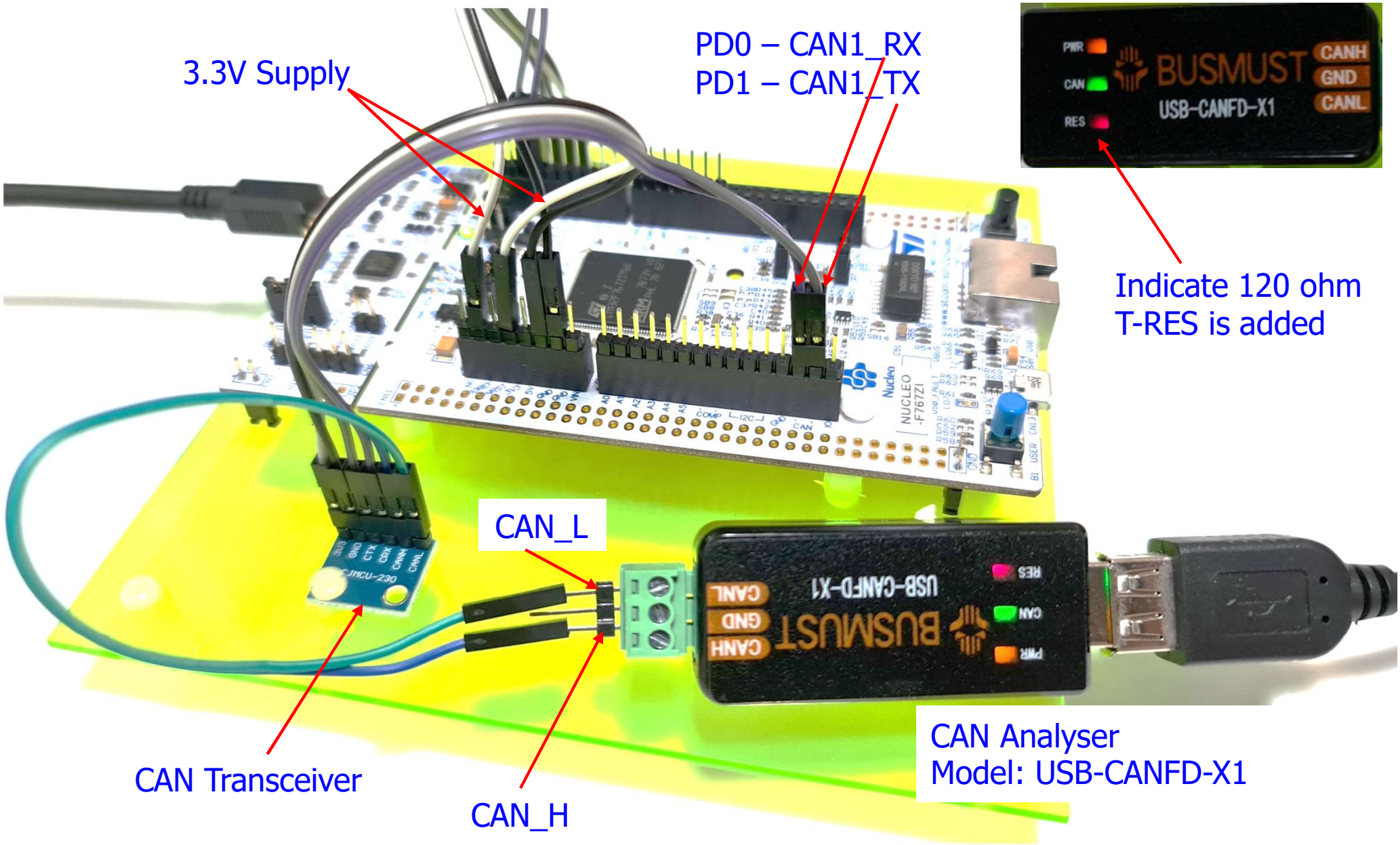


CAN1 RX : PD0
CAN1 TX : PD1

CAN2 TX : PB13
CAN2 RX : PB12

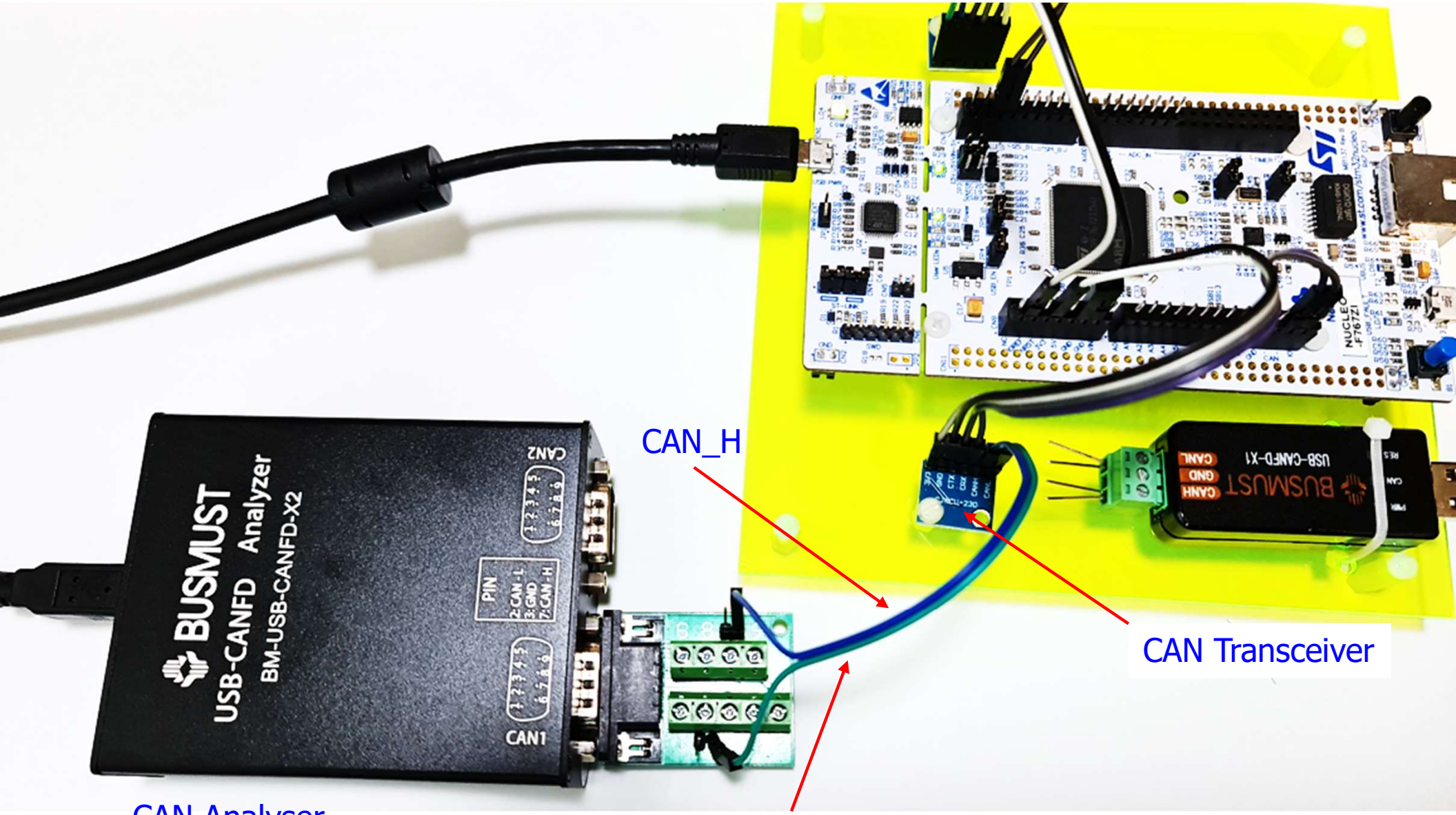
Hands-On CAN TX

Connect CAN Analyzer to Nucleo-F767ZI via CAN Transceiver



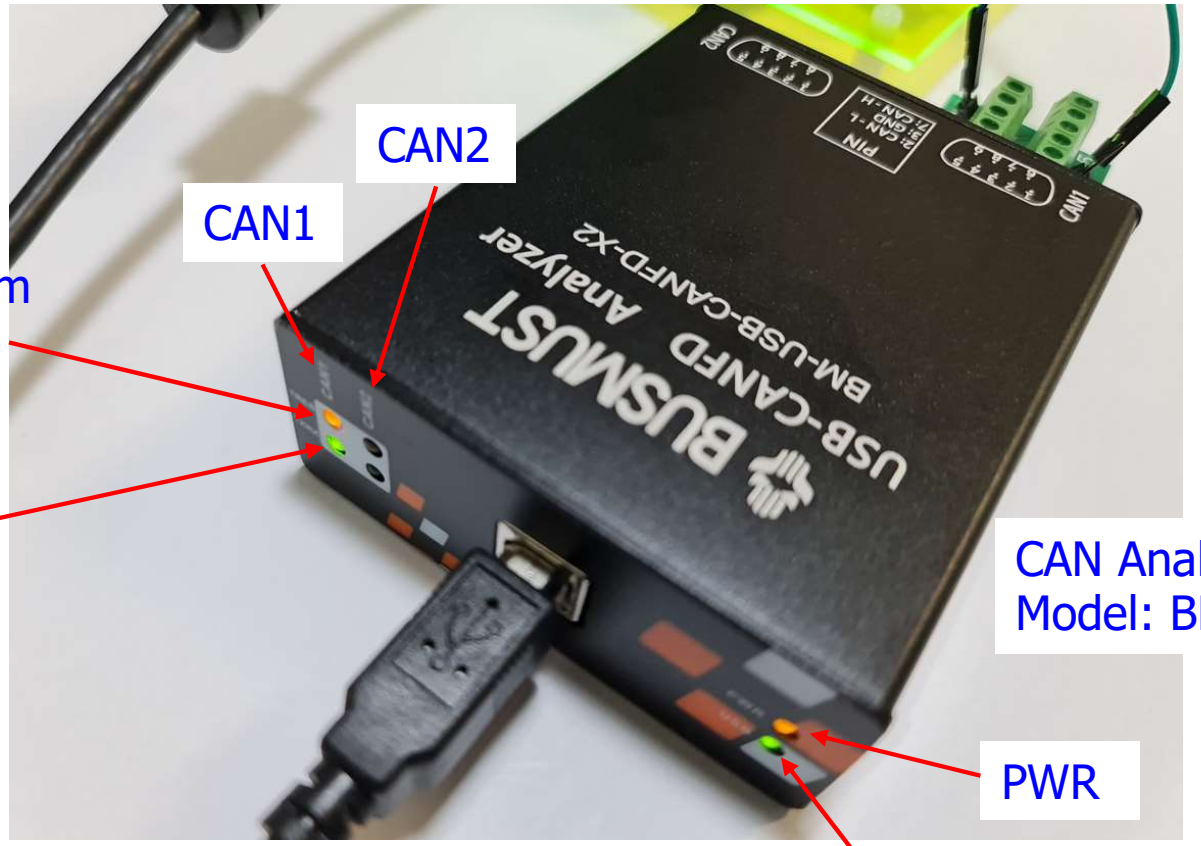
Hands-On CAN TX

Connect CAN Analyzer to Nucleo-F767ZI via CAN Transceiver



Hands-On CAN TX

CAN Transceiver



Indicate 120 ohm
TRES is added

TXRX

CAN2

CAN1

CAN Analyser
Model: BM-USB-CANFD-X2

PWR

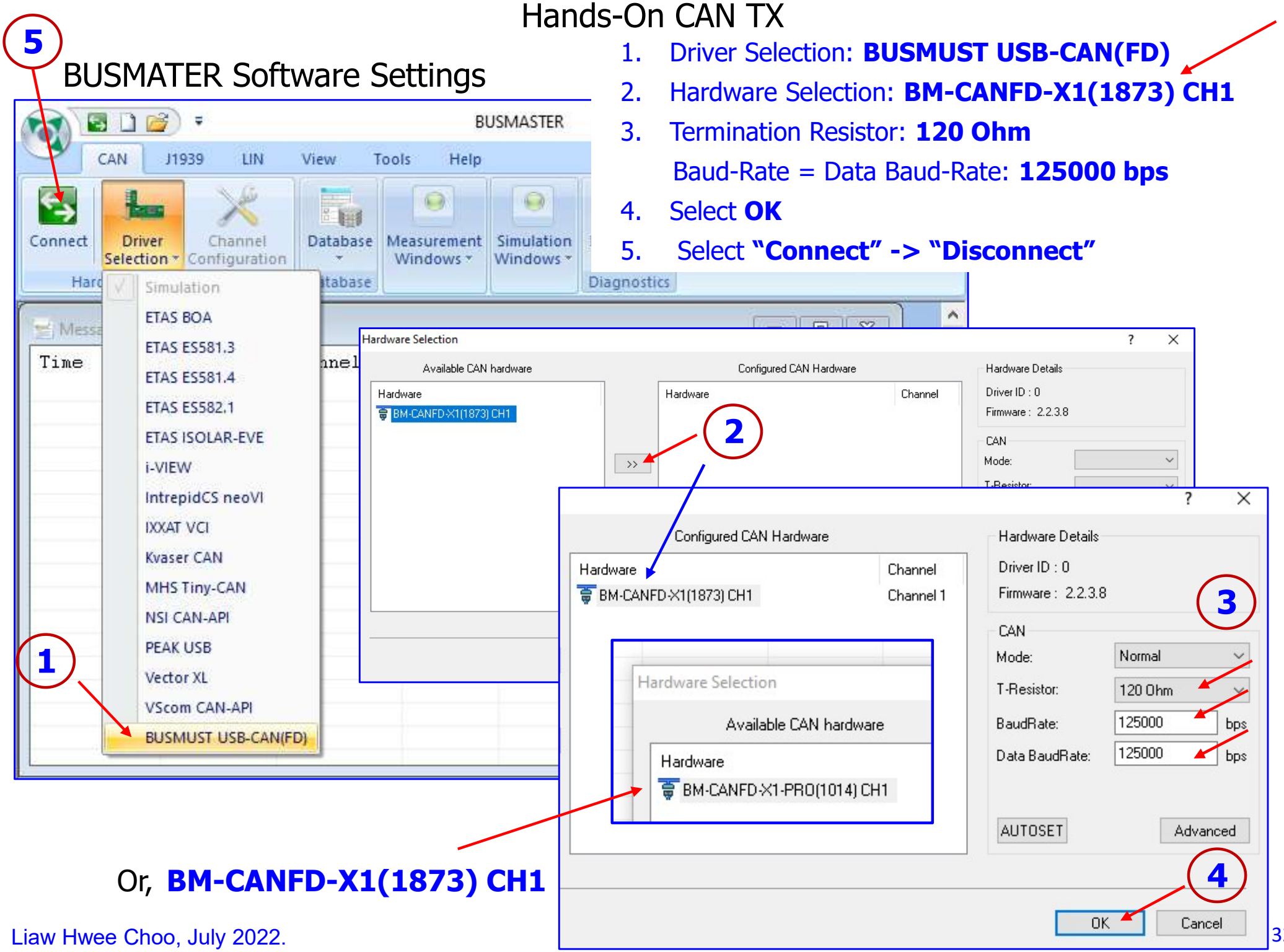
USB

Hands-On CAN TX

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: **125000 bps**
4. Select **OK**
5. Select **"Connect" -> "Disconnect"**



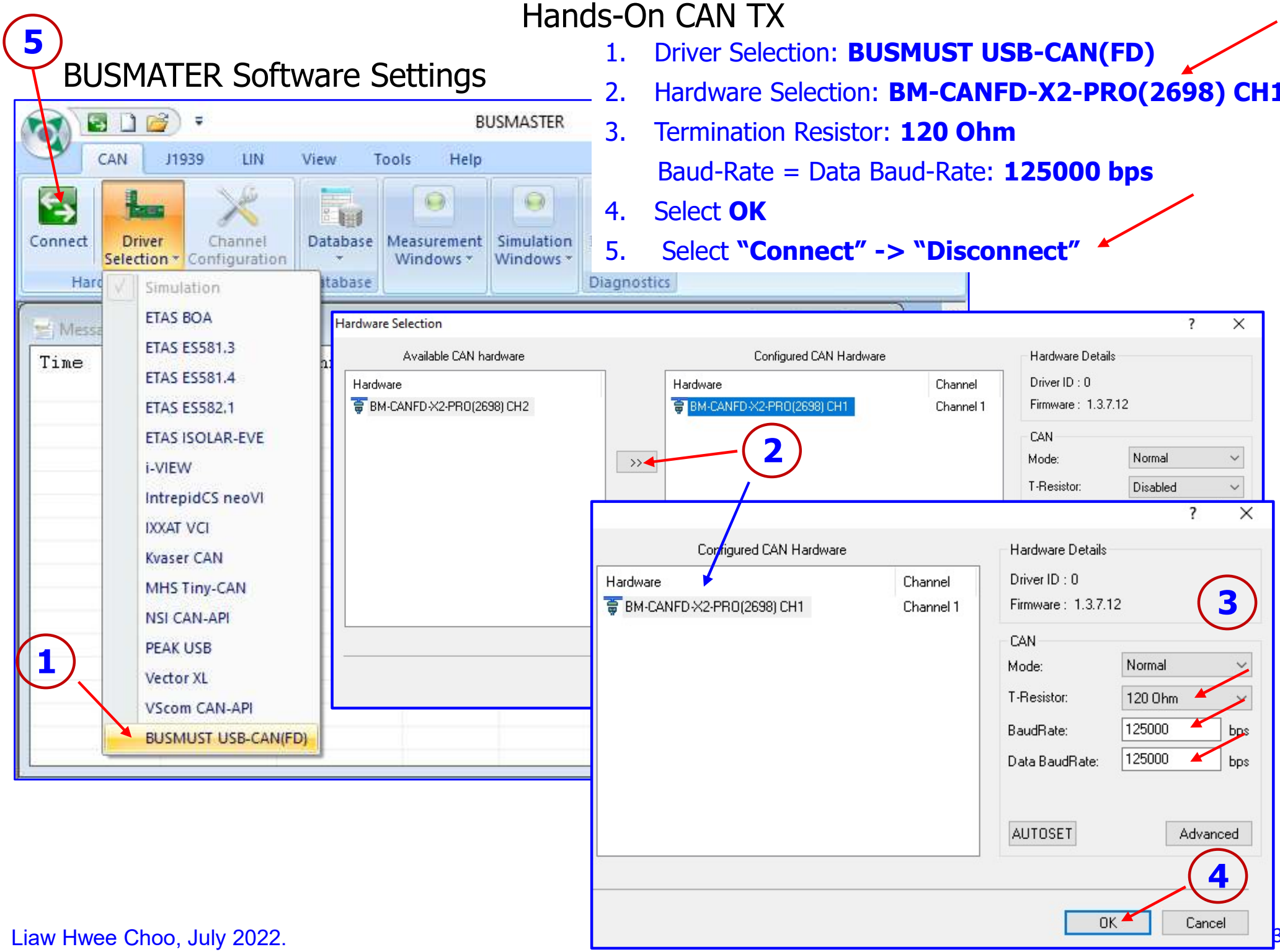
Or, **BM-CANFD-X1(1873) CH1**

Hands-On CAN TX

BUSMATER Software Settings

1. Driver Selection: **BUSMUST USB-CAN(FD)**
2. Hardware Selection: **BM-CANFD-X2-PRO(2698) CH1**
3. Termination Resistor: **120 Ohm**

   Baud-Rate = Data Baud-Rate: **125000 bps**
4. Select **OK**
5. Select **"Connect" -> "Disconnect"**



Hands-On CAN TX
Understand the program with CAN Analyzer (BUSMASTER) and TM Terminal

```
68 int main(void)
69 {
70   /* USER CODE BEGIN 1 */
71
72   /* USER CODE END 1 */
73
74   /* MCU Configuration--
75
76   /* Reset of all periphe
77   HAL_Init();
78
79   /* USER CODE BEGIN Init
80
81   /* USER CODE END Init *
82
83   /* Configure the system
84   SystemClock_Config();
85
86   /* USER CODE BEGIN SysI
87
88   /* USER CODE END SysIni
89
```

Console | Terminal | Probl

COM12

```
vales = 16 (0x10)
vales = 17 (0x11)
vales = 18 (0x12)
vales = 19 (0x13)
vales = 20 (0x14)
vales = 21 (0x15)
vales = 22 (0x16)
vales = 23 (0x17)
vales = 24 (0x18)
vales = 25 (0x19)
vales = 26 (0x1A)
vales = 27 (0x1B)
vales = 28 (0x1C)
vales = 29 (0x1D)
vales = 30 (0x1E)
vales = 31 (0x1F)
vales = 32 (0x20)
vales = 33 (0x21)
```

TM Terminal Results

BUSMASTER

CAN | J1939 | LIN | View | Tools | Help

Disconnect | Driver Selection | Channel Configuration | Database | Network Statistics | Signal Graph | Filters | Message Window | Signal Watch | Logging | Simulation Windows | Diagnostics

Hardware Configuration | Database | Measurement Windows | Diagnostics

Message Window - CAN

Time	Tx/Rx	Channel	Msg Type	ID	Message	DLC	Data Byte
15:36:34:4224	Rx	1	s	0x321	0x321	2	21 00

Message Window | Signal Watch

- Activate
- Enable Filters
- Configure
- Clear
- Time Mode
- Overwrite
- Interpret

Trace Window

Default C | CAN Recording... | J1939 Recording... | 1 Channel(s) - BUSMUST

Hands-On CAN TX
CAN Analyzer (BUSMASTER) and TM Terminal

```
/* Reset of all
HAL_Init();

/* USER CODE BEG

/* USER CODE END

/* Configure the
SystemClock_Conf

/* USER CODE BEG

/* USER CODE END
```

Console | Terminal

COM12

```
vales = 235 (0xEB)
vales = 236 (0xEC)
vales = 237 (0xED)
vales = 238 (0xEE)
vales = 239 (0xEF)
vales = 240 (0xF0)
vales = 241 (0xF1)
vales = 242 (0xF2)
vales = 243 (0xF3)
vales = 244 (0xF4)
vales = 245 (0xF5)
vales = 246 (0xF6)
vales = 247 (0xF7)
vales = 248 (0xF8)
vales = 249 (0xF9)
vales = 250 (0xFA)
vales = 251 (0xFB)
vales = 252 (0xFC)
```

BUSMASTER

CAN | J1939 | LIN | View | Tools | Help

Disconnect | Driver Selection | Channel Configuration | Database | Network Statistics | Signal Graph | Filters | Message Window | Signal Watch | Logging | Simulation Windows | Diagnostics

Hardware Configuration | Database | Measurement Windows | Diagnostics

Message Window - CAN

Time	Tx/Rx	Channel	Msg Type	ID	Messag	DLC	Data Byte(s
16:00:40:2845	Rx	1	s	0x321	0x321	2	E7 00
16:00:40:3365	Rx	1	s	0x321	0x321	2	E8 00
16:00:40:3885				0x321	0x321	2	E9 00
16:00:40:4405				0x321	0x321	2	EA 00
16:00:40:4925				0x321	0x321	2	EB 00
16:00:40:5445				0x321	0x321	2	EC 00
16:00:40:5965				0x321	0x321	2	ED 00
16:00:40:6485				0x321	0x321	2	EE 00
16:00:40:7005				0x321	0x321	2	EF 00
16:00:40:7525				0x321	0x321	2	F0 00
16:00:40:8045				0x321	0x321	2	F1 00
16:00:40:8565				0x321	0x321	2	F2 00
16:00:40:9085				0x321	0x321	2	F3 00
16:00:40:9605				0x321	0x321	2	F4 00
16:00:41:0125				0x321	0x321	2	F5 00
16:00:41:0645				0x321	0x321	2	F6 00
16:00:41:1165				0x321	0x321	2	F7 00
16:00:41:1685				0x321	0x321	2	F8 00
16:00:41:2205	Rx	1	s	0x321	0x321	2	F9 00
16:00:41:2725	Rx	1	s	0x321	0x321	2	FA 00
16:00:41:3245	Rx	1	s	0x321	0x321	2	FB 00

Message Window | Signal Watch | Logging | Simulation Windows

- ✓ Activate
- Enable Filters
- Configure
- Clear
- Time Mode ▸
  - ✓ System Time
  - Absolute Time
  - Relative Time
- Overwrite
- Interpret

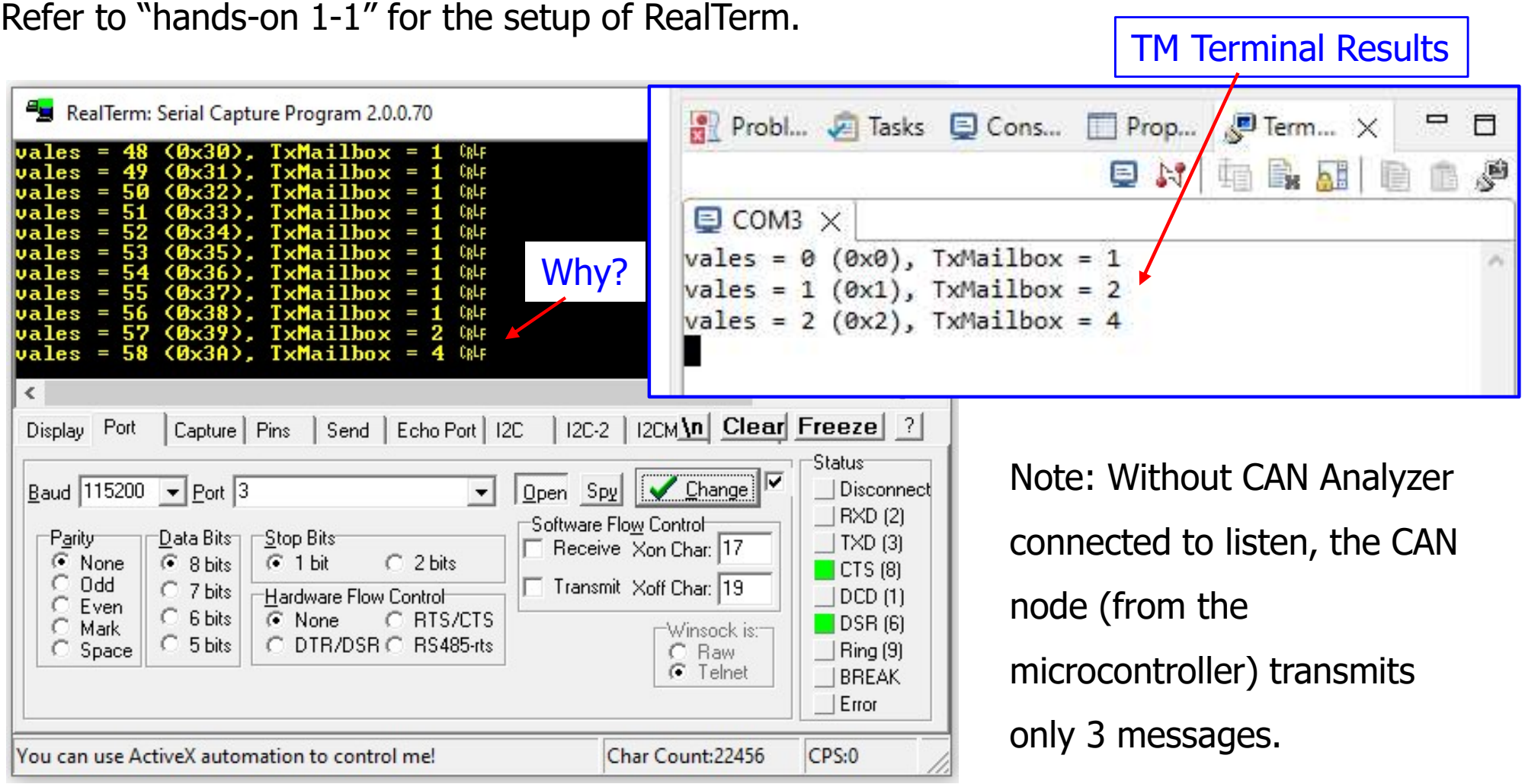
Default C | CAN Recording... | J1939 Recording... | 1 Channel(s) - BUSMUST

TM Terminal Results

Hands-On CAN TX

Study and understand the program with CAN Analyzer and TM terminal or RealTerm

Refer to "hands-on 1-1" for the setup of RealTerm.



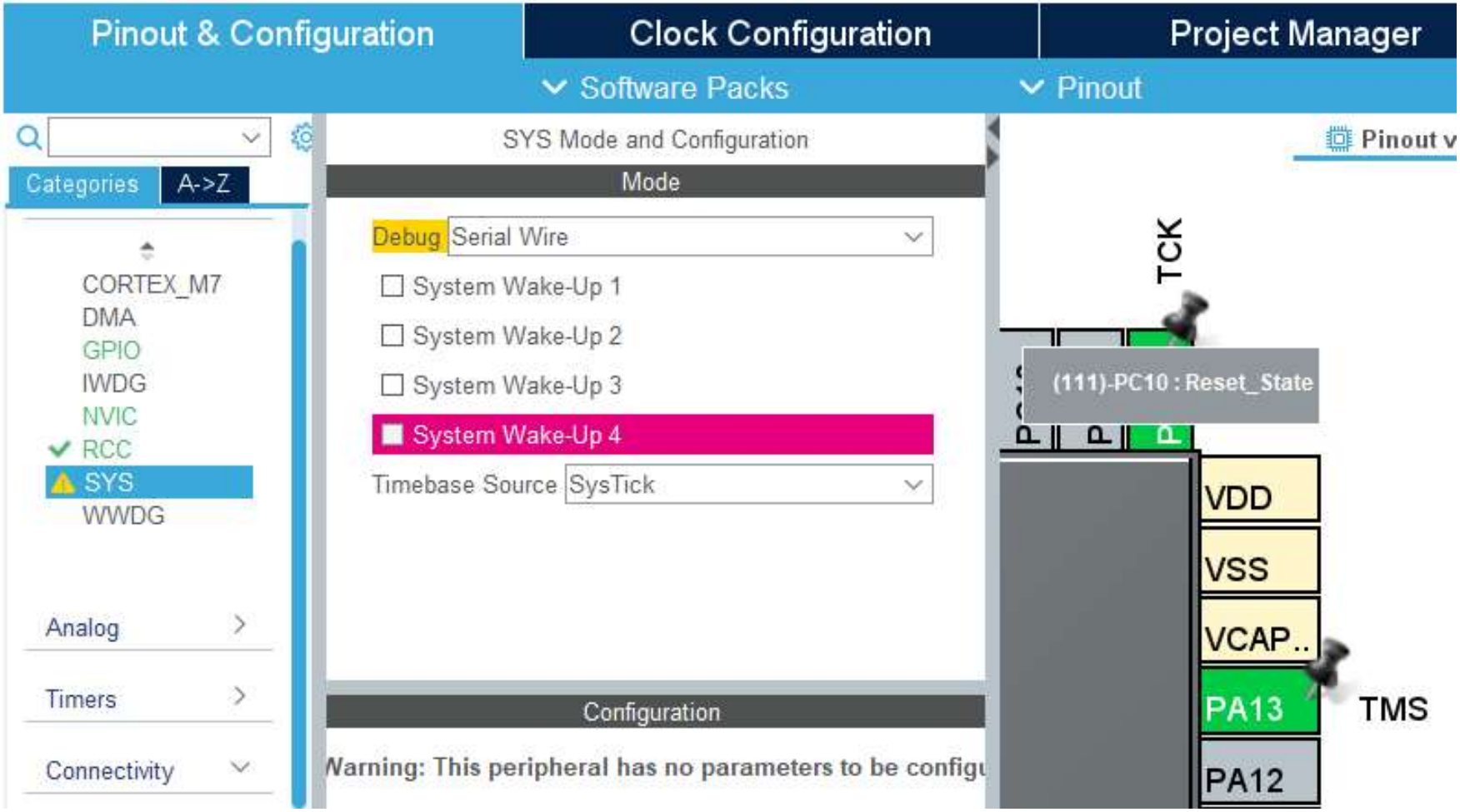
Note: Without CAN Analyzer connected to listen, the CAN node (from the microcontroller) transmits only 3 messages.

Note: Interrupt input from User Button with additional coding can be used for debugging.

Hands-On CAN TX

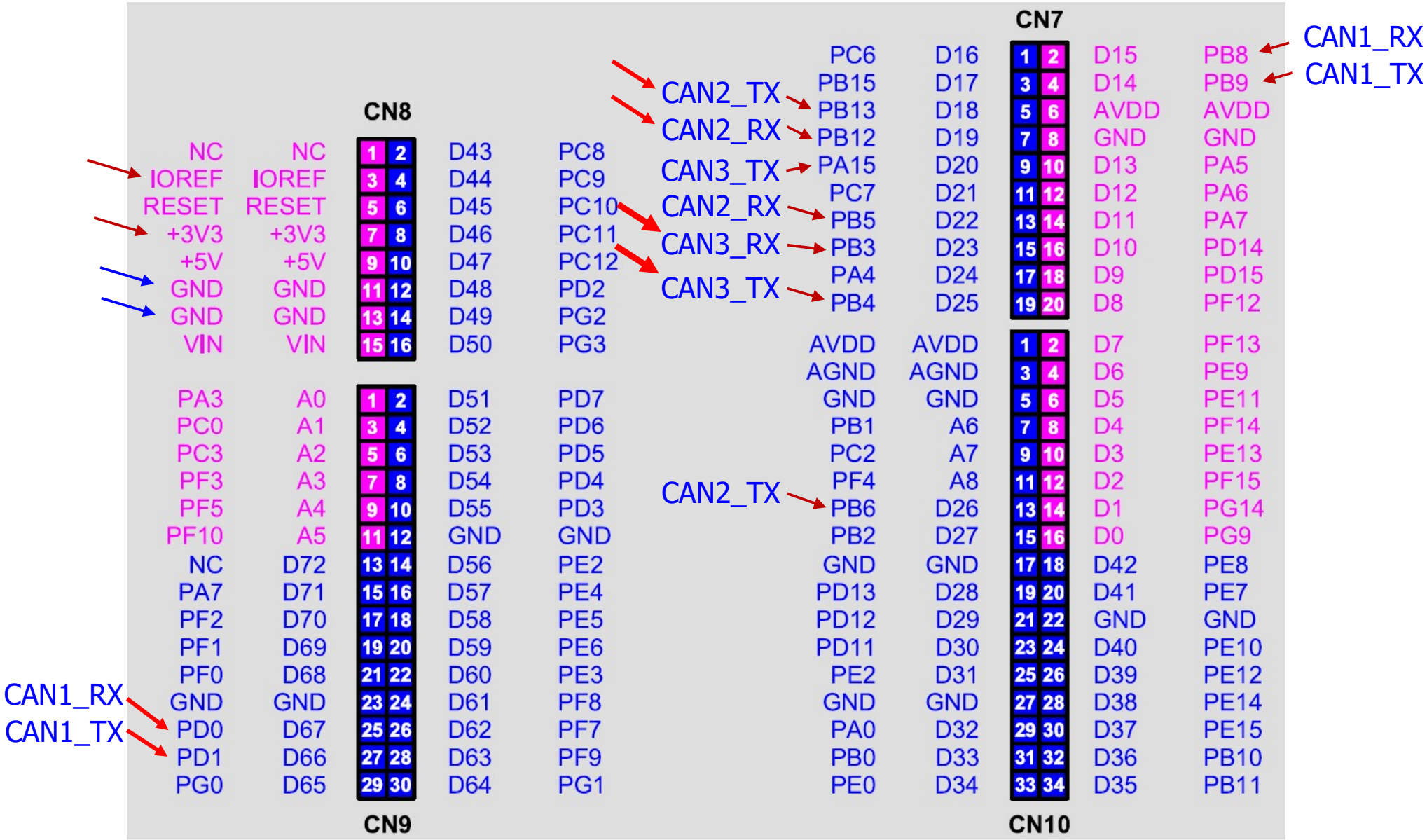
Implement **CAN3** as a CAN Analyzer in **Normal Mode** to Receive Message

As **CAN3** requires pin-out **PB3** (same pin-out for **SWO**), the **Debug Mode** is therefore changed to **Serial Wire.**



Hands-On CAN TX

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



CAN1 RX : PD0
CAN1 TX : PD1

CAN3 RX : PB3
CAN3 TX : PB4

CAN3 Settings:

Hands-On CAN TX

Pinout & Configuration | Clock Configuration | Project M

Software Packs | Pinout

Categories | A->Z

CORTEX_M7
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG

Analog
Timers
Connectivity

CAN1
CAN2
CAN3
ETH
FMC
I2C1
I2C2
I2C3
I2C4
MDIOS
QUADSPI

CAN3 Mode and Configuration

Mode

Activated

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)	27
Time Quantum	500.0 ns
Time Quanta in Bit Segment 1	11 Times
Time Quanta in Bit Segment 2	4 Times
Time for one Bit	8000 ns
Baud Rate	125000 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	Disable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable
Advanced Parameters	
Operating Mode	Normal

CAN3_TX
CAN3_RX
PB4
PB3

Bit Timings Parameters are same as CAN1

CAN3 Settings: Hands-On CAN TX

Enable **CAN3 RX0** (Receive FIFO0) Interrupt

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

System Core
Analog
Timers
Connectivity
CAN1
CAN2
CAN3
ETH
FMC
I2C1
I2C2
I2C3

CAN3 Mode and Configuration

Mode

Activated

Configuration

Reset Configuration

User Constants | NVIC Settings | GPIO Settings

Parameter Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
CAN3 TX interrupt	☐	0	0
CAN3 RX0 interrupt	☑	0	0
CAN3 RX1 interrupt	☐	0	0
CAN3 SCE interrupt	☐	0	0

CAN3_TX
CAN3_RX
PB4
PB3

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...	Fast ...	User Label	Modified
PB3	CAN3_RX	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	n/a		☐
PB4	CAN3_TX	n/a	Alternate Function Push Pull	No pull-up and no pull-down	Very High	n/a		☐

Add Additional Code to **can.h**

```
/* can.h */

#include "main.h"
extern CAN_HandleTypeDef hcan1;
extern CAN_HandleTypeDef hcan3;

/* USER CODE BEGIN Private defines */
extern  CAN_TxHeaderTypeDef   TxHeader;
extern  CAN_RxHeaderTypeDef   RxHeader3;
extern  uint8_t               RxData3[8];
/* USER CODE END Private defines */

void MX_CAN1_Init(void);
void MX_CAN3_Init(void);

/* USER CODE BEGIN Prototypes */
void CAN_Config(void);
void CAN3_Config(void);
/* USER CODE END Prototypes */
```

Hands-On CAN TX

Add Additional Code to **can.c**

```c
/* can.c */

#include "can.h"

/* USER CODE BEGIN 0 */
CAN_TxHeaderTypeDef   TxHeader;
CAN_RxHeaderTypeDef   RxHeader3;
uint8_t   RxData3[8] = {0};
/* USER CODE END 0 */
CAN_HandleTypeDef hcan1;
CAN_HandleTypeDef hcan3;

/* USER CODE BEGIN 1 */
void CAN_Config(void)
{     . . .   }

void CAN3_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 = 0;
```

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

    /* #3 Configure Transmission process */
    TxHeader.StdId = 0x321;  // 801
    TxHeader.ExtId = 0xFFF;  // 4095
    TxHeader.IDE = CAN_ID_EXT; // CAN_ID_STD
    TxHeader.RTR = CAN_RTR_DATA;
    TxHeader.DLC = 2;
    TxHeader.TransmitGlobalTime = DISABLE;
}
```

Hands-On CAN TX

Add Additional Code to **can.c**

```
//  sFilterConfig.SlaveStartFilterBank = 14;
    if (HAL_CAN_ConfigFilter(&hcan3, &sFilterConfig) != HAL_OK)
    {
        /* Filter configuration Error */
        Error_Handler();
    }

    /* #2 Start the CAN peripheral */
    if (HAL_CAN_Start(&hcan3) != HAL_OK)
    {
        /* Start Error */
        Error_Handler();
    }

    /* #3 Activate CAN RX notification */
    if (HAL_CAN_ActivateNotification(&hcan3,  CAN_IT_RX_FIFO0_MSG_PENDING) != HAL_OK)
    {
        /* Notification Error */
        Error_Handler();
    }

    /* #4 Configure Transmission process */
    /*
    TxHeader3.StdId = 0x321;
    TxHeader3.ExtId = 0x01;
    TxHeader3.IDE = CAN_ID_STD;
    TxHeader3.RTR = CAN_RTR_DATA;
    TxHeader3.DLC = 2;
    TxHeader3.TransmitGlobalTime = DISABLE;
    */
}
```

Hands-On CAN TX

Add Additional Code to **can.c**

```
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    if (hcan->Instance == CAN3)
    {
        /* Get RX message */
        if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader3, RxData3) != HAL_OK)
        {
            Error_Handler();
        }
    }
}

/* USER CODE END 1 */
```

Add Additional Code to **main.c**

```c
/* main.c */
#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 variables */
/* USER CODE BEGIN PV */
uint8_t   TxData[8] = {0};
uint32_t  TxMailbox;
uint8_t   g_nCnt = 0;
/* USER CODE END PV */

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

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART3_UART_Init();
  MX_CAN1_Init();
  MX_CAN3_Init();
  /* USER CODE BEGIN 2 */
```

```c
/* USER CODE BEGIN 2 */
CAN_Config();
CAN3_Config();
/* USER CODE END 2 */

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

Add Additional Code to **main.c**

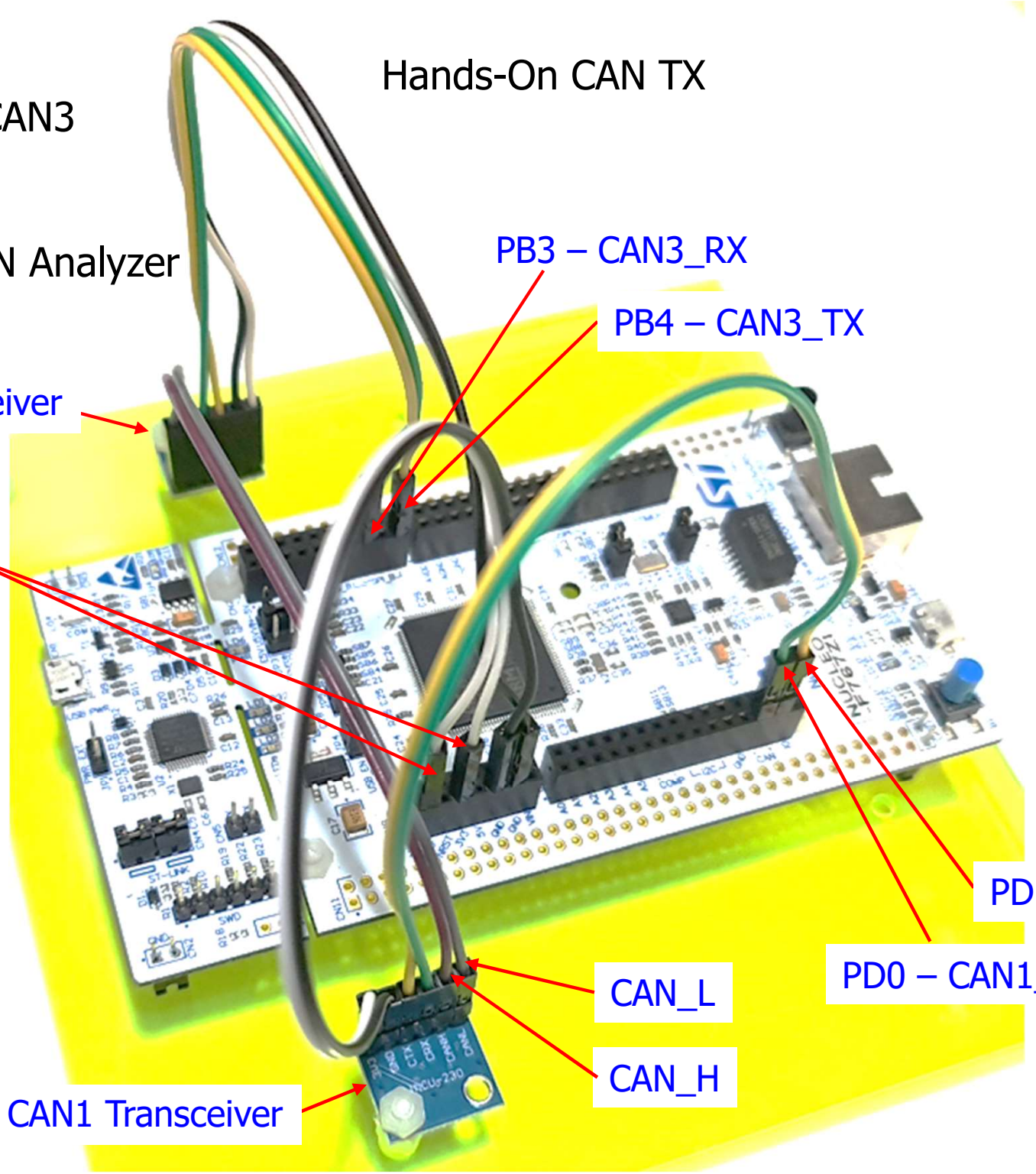
```
/* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      HAL_GPIO_TogglePin(GPIOB, LD1_Pin);
      TxData[0] = g_nCnt;
      TxData[1] = g_nCnt;
      g_nCnt = g_nCnt + 1;
      if( g_nCnt > 0xff ) g_nCnt = 0;
      if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox) != HAL_OK)
      {
          /* Transmission request Error */
          Error_Handler();
      }
      /* Wait transmission complete */
      while(HAL_CAN_GetTxMailboxesFreeLevel(&hcan1) != 3) {}

      printf("Transmitted data = %d (0x%X), %d (0x%X), TxMailbox = %ld, Received data:  %d (0x%X),
              %d (0x%X), StdId = %ld, ExtId = %ld, IDE = %ld, DLC = %ld \r\n",
              TxData[0],  TxData[0],  TxData[1],  TxData[1], TxMailbox,
             RxData3[0], RxData3[0], RxData3[1], RxData3[1],
             RxHeader3.StdId, RxHeader3.ExtId, RxHeader3.IDE, RxHeader3.DLC );

      HAL_Delay(50);

    /* USER CODE END WHILE */

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

Hands-On CAN TX

CAN1 and CAN3

Connection

Without CAN Analyzer



Hands-On CAN TX

CAN1 and CAN3 Test Results

Console | Terminal | Problems | Executables | Memory

COM12

```
Transmitted data = 64 (0x40), 64 (0x40), TxMailbox = 1, Received data:  64 (0x40), 64 (0x40), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 65 (0x41), 65 (0x41), TxMailbox = 1, Received data:  65 (0x41), 65 (0x41), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 66 (0x42), 66 (0x42), TxMailbox = 1, Received data:  66 (0x42), 66 (0x42), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 67 (0x43), 67 (0x43), TxMailbox = 1, Received data:  67 (0x43), 67 (0x43), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 68 (0x44), 68 (0x44), TxMailbox = 1, Received data:  68 (0x44), 68 (0x44), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 69 (0x45), 69 (0x45), TxMailbox = 1, Received data:  69 (0x45), 69 (0x45), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 70 (0x46), 70 (0x46), TxMailbox = 1, Received data:  70 (0x46), 70 (0x46), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 71 (0x47), 71 (0x47), TxMailbox = 1, Received data:  71 (0x47), 71 (0x47), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 72 (0x48), 72 (0x48), TxMailbox = 1, Received data:  72 (0x48), 72 (0x48), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 73 (0x49), 73 (0x49), TxMailbox = 1, Received data:  73 (0x49), 73 (0x49), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 74 (0x4A), 74 (0x4A), TxMailbox = 1, Received data:  74 (0x4A), 74 (0x4A), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 75 (0x4B), 75 (0x4B), TxMailbox = 1, Received data:  75 (0x4B), 75 (0x4B), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 76 (0x4C), 76 (0x4C), TxMailbox = 1, Received data:  76 (0x4C), 76 (0x4C), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 77 (0x4D), 77 (0x4D), TxMailbox = 1, Received data:  77 (0x4D), 77 (0x4D), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 78 (0x4E), 78 (0x4E), TxMailbox = 1, Received data:  78 (0x4E), 78 (0x4E), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 79 (0x4F), 79 (0x4F), TxMailbox = 1, Received data:  79 (0x4F), 79 (0x4F), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 80 (0x50), 80 (0x50), TxMailbox = 1, Received data:  80 (0x50), 80 (0x50), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 81 (0x51), 81 (0x51), TxMailbox = 1, Received data:  81 (0x51), 81 (0x51), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 82 (0x52), 82 (0x52), TxMailbox = 1, Received data:  82 (0x52), 82 (0x52), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 83 (0x53), 83 (0x53), TxMailbox = 1, Received data:  83 (0x53), 83 (0x53), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 84 (0x54), 84 (0x54), TxMailbox = 1, Received data:  84 (0x54), 84 (0x54), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 85 (0x55), 85 (0x55), TxMailbox = 1, Received data:  85 (0x55), 85 (0x55), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 86 (0x56), 86 (0x56), TxMailbox = 1, Received data:  86 (0x56), 86 (0x56), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 87 (0x57), 87 (0x57), TxMailbox = 1, Received data:  87 (0x57), 87 (0x57), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
Transmitted data = 88 (0x58), 88 (0x58), TxMailbox = 1, Received data:  88 (0x58), 88 (0x58), StdId = 0, ExtId = 4095, IDE = 4, DLC = 2
```

CAN1 TX **CAN2 RX**