PTX Tunneling Library v1.4.1 for STM32CubeIDE

The PTX Tunneling library can be used to evaluate and optimize the performance (antenna matching, system/RF configuration, etc.) of any custom-made device using a PTX100x device via SPI serial interface.

Embedding this library into the device firmware enables the translation of communication from **UART to SPI**, so that the full functionality of the **PTX100x * Config Tool** can be used in a custom environment. This document also provides instructions on how to create a sample application using an ST32L562E-DK development board.

Contents

1. Requirements

The footprint of the library is ~13kB Flash and 10kB RAM. Moreover, a hardware abstraction layer must be implemented by the user for the particular uC/Board, which executes the low-level commands requested by the library. From the resource point of view, only the SysTick timer, UART, SPI, and the IRQ pin will be used.

The library can be seamlessly integrated into a CMAKE project as well, but the STM32CubeIde is used in this document (for more information, see STM32CubeIde is used in this document).

2. Sample Firmware

The sample application is used for creating and serving the tunnel between the host PC UART interface and the PTX100x chip connected by SPI. The library can be used either as a precompiled static library or as a source-library – most steps are the same for both cases.

2.1 Creating the Project

The used HAL source code will be picked from the SDK and copied to the project folder by the *STM32CubeIde* based on the configuration file (.ioc). This is the most convenient way to start a new project.

2.1.1. Configuration File

The configuration file `ptxTunneling.ioc` must be created in an arbitrary place with the following content:

```
#MicroXplorer Configuration settings - do not modify
File.Version=6
GPIO.groupedBy=Group By Peripherals
KeepUserPlacement=false
LPTIM1.ClockPrescaler=LPTIM_PRESCALER_DIV16
LPTIM1.IPParameters=ClockPrescaler
LPTIM2.ClockPrescaler=LPTIM_PRESCALER_DIV16
LPTIM2.IPParameters=ClockPrescaler
Mcu.ContextProject=TrustZoneDisabled
Mcu.Family=STM32L5
Mcu.IP0=LPTIM1
Mcu.IP1=LPTIM2
Mcu.IP2=NVIC
Mcu.IP3=PWR
Mcu.IP4=RCC
Mcu.IP5=SPI3
Mcu.IP6=SYS
Mcu.IP7=USART1
Mcu.IPNb=8
Mcu.Name=STM32L562QEIxQ
Mcu.Package=UFBGA132
Mcu.Pin0=PB4 (NJTRST)
Mcu.Pin1=PG9
Mcu.Pin10=VP_LPTIM2_VS_LPTIM_counterModeInternalClock
Mcu.Pin11=VP_PWR_VS_DBSignals
Mcu.Pin12=VP_SYS_VS_Systick
Mcu.Pin2=PB5
Mcu.Pin3=PA9
Mcu.Pin4=PA10
Mcu.Pin5=PF5
Mcu.Pin6=PF11
Mcu.Pin7=PB13
Mcu.Pin8=PF12
Mcu.Pin9=VP_LPTIM1_VS_LPTIM_counterModeInternalClock
Mcu.PinsNb=13
Mcu.ThirdPartyNb=0
Mcu.UserConstants=
Mcu.UserName=STM32L562QEIxQ
MxCube.Version=6.0.0
MxDb.Version=DB.6.0.0
NVIC.BusFault_IRQn=true\:0\:0\:false\:false\:true\:false\:false
NVIC.DebugMonitor_IRQn=true\:0\:0\:false\:false\:true\:false\:false
NVIC.ForceEnableDMAVector=true
NVIC.HardFault_IRQn=true\:0\:0\:false\:false\:true\:false\:false
NVIC.LPTIM1_IRQn=true\:0\:0\:false\:false\:true\:true\:true
NVIC.LPTIM2_IRQn=true\:0\:0\:false\:false\:true\:true\:true
NVIC.MemoryManagement_IRQn=true\:0\:0\:false\:false\:true\:false\:false
NVIC.NonMaskableInt_IRQn=true\:0\:0\:false\:false\:true\:false\:false
NVIC.PendSV_IRQn=true\:0\:0\:false\:false\:true\:false\:false
NVIC.PriorityGroup=NVIC_PRIORITYGROUP_3
NVIC.SPI3_IRQn=true\:3\:0\:true\:false\:true\:true\:true
NVIC.SVCall_IRQn=true\:0\:0\:false\:false\:true\:false\:false
NVIC.SysTick_IRQn=true\:0\:0\:false\:false\:true\:false\:true
NVIC.USART1_IRQn=true\:0\:0\:false\:false\:true\:true\:true
NVIC.UsageFault_IRQn=true\:0\:0\:false\:false\:true\:false\:false
PA10.Locked=true
PA10.Mode=Asynchronous
PA10.Signal=USART1_RX
PA9.Locked=true
PA9.Mode=Asynchronous
PA9.Signal=USART1_TX
```

```
PB13.GPIOParameters=PinState,GPIO_Label
PB13.GPIO_Label=SPI3_NSS
PB13.Locked=true
PB13.PinState=GPIO_PIN_SET
PB13.Signal=GPIO_Output
PB4\ (NJTRST).GPIOParameters=GPIO_Speed,GPIO_PuPd,GPIO_Label,GPIO_Mode
PB4\ (NJTRST).GPIO_Label=SPI3_MISO
PB4\ (NJTRST).GPIO_Mode=GPIO_MODE_AF_PP
PB4\ (NJTRST).GPIO_PuPd=GPIO_NOPULL
PB4\ (NJTRST).GPIO_Speed=GPIO_SPEED_FREQ_MEDIUM
PB4\ (NJTRST).Locked=true
PB4\ (NJTRST).Mode=Full_Duplex_Master
PB4\ (NJTRST).Signal=SPI3_MISO
PB5.GPIOParameters=GPIO_Speed,GPIO_Label
PB5.GPIO_Label=SPI3_MOSI
PB5.GPIO_Speed=GPIO_SPEED_FREQ_MEDIUM
PB5.Locked=true
PB5.Mode=Full_Duplex_Master
PB5.Signal=SPI3_MOSI
PF11.Locked=true
PF11.Signal=GPIO_Output
PF12.Locked=true
PF12.Signal=GPIO_Output
PF5.GPIOParameters=GPIO_Label
PF5.GPIO_Label=PTX_IRQ
PF5.Locked=true
PF5.Signal=GPIO_Input
PG9.GPIOParameters=GPIO_Speed,GPIO_Label
PG9.GPIO_Label=SPI3_SCK
PG9.GPIO_Speed=GPIO_SPEED_FREQ_MEDIUM
PG9.Locked=true
PG9.Mode=Full_Duplex_Master
PG9.Signal=SPI3_SCK
PinOutPanel.CurrentBGAView=Top
PinOutPanel.RotationAngle=0
ProjectManager.AskForMigrate=true
ProjectManager.BackupPrevious=false
ProjectManager.CompilerOptimize=6
ProjectManager.ComputerToolchain=false
ProjectManager.CoupleFile=false
ProjectManager.CustomerFirmwarePackage=
ProjectManager.DefaultFWLocation=true
ProjectManager.DeletePrevious=true
ProjectManager.DeviceId=STM32L562QEIxQ
ProjectManager.FirmwarePackage=STM32Cube FW_L5 V1.3.0
ProjectManager.FreePins=false
ProjectManager.HalAssertFull=false
ProjectManager.HeapSize=0x200
ProjectManager.KeepUserCode=true
ProjectManager.LastFirmware=false
ProjectManager.LibraryCopy=1
ProjectManager.MainLocation=Core/Src
ProjectManager.NoMain=false
ProjectManager.PreviousToolchain=STM32CubeIDE
ProjectManager.ProjectBuild=false
ProjectManager.ProjectFileName=demo2.ioc
ProjectManager.ProjectName=demo2
ProjectManager.RegisterCallBack=
ProjectManager.StackSize=0x400
ProjectManager.TargetToolchain=STM32CubeIDE
ProjectManager.ToolChainLocation=
```

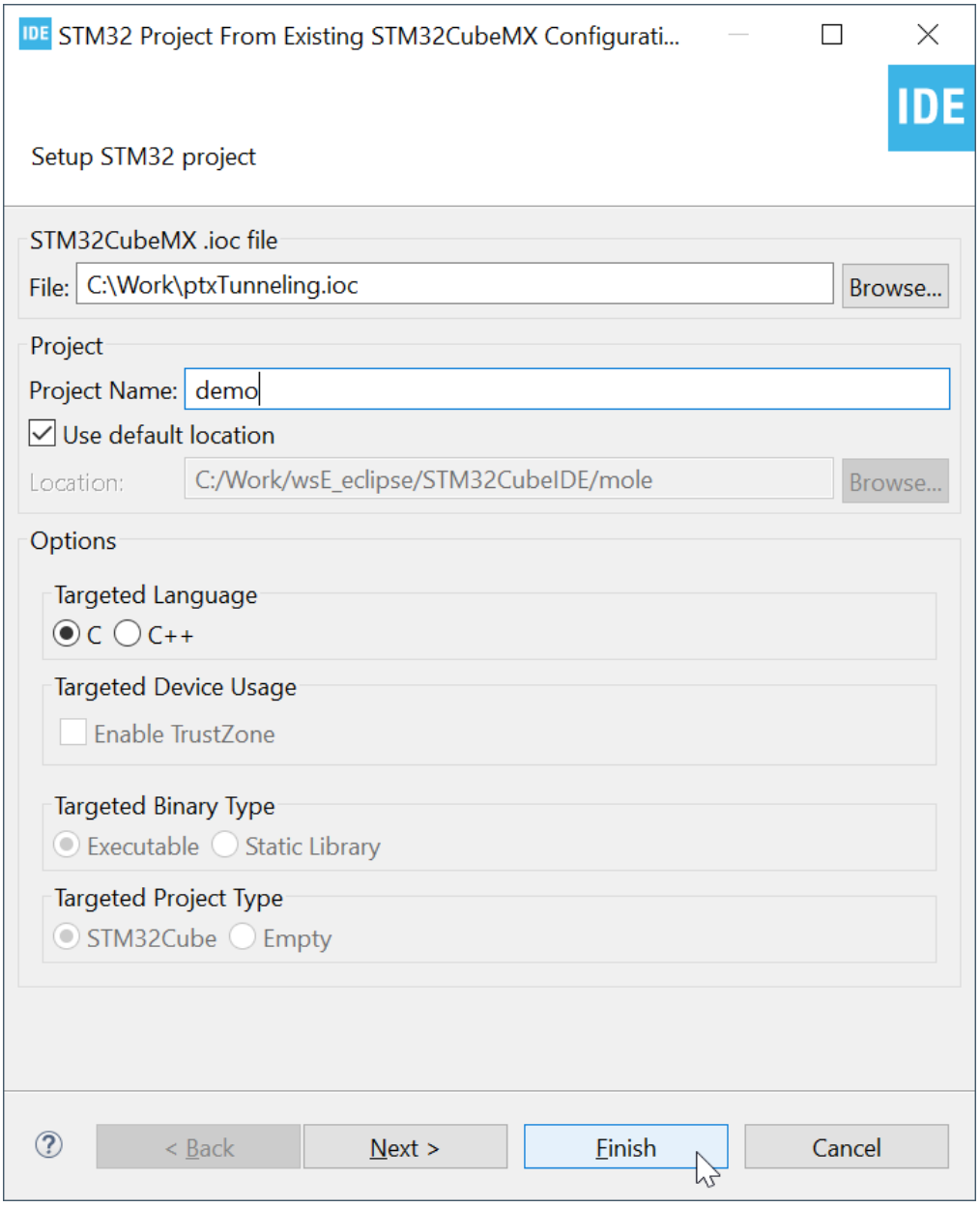
```
ProjectManager.UnderRoot=true
ProjectManager.functionlistsort=1-MX_GPIO_Init-GPIO-false-HAL-true,2-SystemClock_Config-
RCC-false-HAL-false,3-MX_LPTIM1_Init-LPTIM1-false-HAL-true,4-MX_LPTIM2_Init-LPTIM2-
false-HAL-true,5-MX_SPI3_Init-SPI3-false-HAL-true,6-MX_USART1_UART_Init-USART1-false-
HAL-true,0-MX_PWR_Init-PWR-false-HAL-true
RCC.ADCFreq_Value=96000000
RCC.AHBFreq_Value=110000000
RCC.APB1Freq_Value=110000000
RCC.APB1TimFreq_Value=110000000
RCC.APB2Freq_Value=110000000
RCC.APB2TimFreq_Value=110000000
RCC.CK48CLockSelection=RCC_USBCLKSOURCE_MSI
RCC.CRSFreq_Value=48000000
RCC.CortexFreq_Value=110000000
RCC.DFSDMAudioFreq_Value=48000000
RCC.DFSDMFreq_Value=110000000
RCC.FCLKCortexFreq_Value=110000000
RCC.FDCANFreq_Value=110000000
RCC.FamilyName=M
RCC.HCLKFreq_Value=110000000
RCC.HSE_VALUE=8000000
RCC.HSI48_VALUE=48000000
RCC.HSI_VALUE=16000000
RCC.I2C1Freq_Value=110000000
RCC.I2C2Freq_Value=110000000
RCC.I2C3Freq_Value=110000000
RCC.I2C4Freq_Value=110000000
RCC.IPParameters=ADCFreq_Value,AHBFreq_Value,APB1Freq_Value,APB1TimFreq_Value,APB2Freq_V
alue,APB2TimFreq_Value,CK48CLockSelection,CRSFreq_Value,CortexFreq_Value,DFSDMAudioFreq_
Value,DFSDMFreq_Value,FCLKCortexFreq_Value,FDCANFreq_Value,FamilyName,HCLKFreq_Value,HSE
_VALUE,HSI48_VALUE,HSI_VALUE,I2C1Freq_Value,I2C2Freq_Value,I2C3Freq_Value,I2C4Freq_Value
,LPTIM1CLockSelectionVirtual,LPTIM1Freq_Value,LPTIM2CLockSelectionVirtual,LPTIM2Freq_Val
ue,LPTIM3Freq_Value,LPUART1Freq_Value,LSCOPinFreq_Value,LSE_VALUE,LSI_VALUE,MCO1PinFreq_
Value,MSIClockRange,MSI_VALUE,OCTOSPIMFreq_Value,PLLM,PLLN,PLLPoutputFreq_Value,PLLQoutp
utFreq_Value,PLLRCLKFreq_Value,PLLSAI1M,PLLSAI1N,PLLSAI1P,PLLSAI1PoutputFreq_Value,PLLSA
I1QoutputFreq_Value,PLLSAI1RoutputFreq_Value,PLLSAI1Source,PLLSAI2PoutputFreq_Value,PWRF
req_Value,RNGFreq_Value,SAI1Freq_Value,SAI2Freq_Value,SDMMCClockSelection,SDMMCFreq_Valu
e,SYSCLKFreq_VALUE,SYSCLKSource,UART4Freq_Value,UART5Freq_Value,USART1Freq_Value,USART2F
req_Value,USART3Freq_Value,USBFreq_Value,VCOInput2Freq_Value,VCOInput3Freq_Value,VCOInpu
tFreq_Value,VCOOutputFreq_Value,VCOSAI1OutputFreq_Value,VCOSAI2OutputFreq_Value
RCC.LPTIM1CLockSelectionVirtual=RCC_LPTIM1CLKSOURCE_HSI
RCC.LPTIM1Freq_Value=16000000
RCC.LPTIM2CLockSelectionVirtual=RCC_LPTIM2CLKSOURCE_HSI
RCC.LPTIM2Freq_Value=16000000
RCC.LPTIM3Freq_Value=110000000
RCC.LPUART1Freq_Value=110000000
RCC.LSCOPinFreq_Value=32000
RCC.LSE_VALUE=32768
RCC.LSI_VALUE=32000
RCC.MCO1PinFreq_Value=110000000
RCC.MSIClockRange=RCC_MSIRANGE_11
RCC.MSI_VALUE=48000000
RCC.OCTOSPIMFreq_Value=110000000
RCC.PLLM=12
RCC.PLLN=55
RCC.PLLPoutputFreq_Value=31428571.42857143
RCC.PLLQoutputFreq_Value=110000000
RCC.PLLRCLKFreq_Value=110000000
RCC.PLLSAI1M=4
RCC.PLLSAI1N=48
RCC.PLLSAI1P=RCC_PLLP_DIV17
```

```
RCC.PLLSAI1PoutputFreq_Value=11294117.647058824
RCC.PLLSAI1QoutputFreq_Value=96000000
RCC.PLLSAI1RoutputFreq_Value=96000000
RCC.PLLSAI1Source=RCC_PLLSAI1SOURCE_HSI
RCC.PLLSAI2PoutputFreq_Value=54857142.85714286
RCC.PWRFreq_Value=110000000
RCC.RNGFreq_Value=48000000
RCC.SAI1Freq_Value=11294117.647058824
RCC.SAI2Freq_Value=11294117.647058824
RCC.SDMMCClockSelection=RCC_SDIOCLKSOURCE_CLK48
RCC.SDMMCFreq_Value=48000000
RCC.SYSCLKFreq_VALUE=110000000
RCC.SYSCLKSource=RCC_SYSCLKSOURCE_PLLCLK
RCC.UART4Freq_Value=110000000
RCC.UART5Freq_Value=110000000
RCC.USART1Freq_Value=110000000
RCC.USART2Freq_Value=110000000
RCC.USART3Freq_Value=110000000
RCC.USBFreq_Value=48000000
RCC.VCOInput2Freq_Value=4000000
RCC.VCOInput3Freq_Value=48000000
RCC.VCOInputFreq_Value=4000000
RCC.VCOOutputFreq_Value=220000000
RCC.VCOSAI1OutputFreq_Value=192000000
RCC.VCOSAI2OutputFreq_Value=384000000
SPI3.BaudRatePrescaler=SPI_BAUDRATEPRESCALER_16
SPI3.CalculateBaudRate=6.875 MBits/s
SPI3.DataSize=SPI_DATASIZE_8BIT
SPI3.Direction=SPI_DIRECTION_2LINES
SPI3.IPParameters=VirtualType,Mode,Direction,BaudRatePrescaler,CalculateBaudRate,DataSiz
e
SPI3.Mode=SPI_MODE_MASTER
SPI3.VirtualType=VM_MASTER
USART1.FIFOMode=FIFOMODE_DISABLE
USART1.IPParameters=VirtualMode-Asynchronous,FIFOMode,RXFIFOThreshold
USART1.RXFIFOThreshold=RXFIFO_THRESHOLD_HALFFULL
USART1.VirtualMode-Asynchronous=VM_ASYNC
VP_LPTIM1_VS_LPTIM_counterModeInternalClock.Mode=Counts__internal_clock_event_00
VP_LPTIM1_VS_LPTIM_counterModeInternalClock.Signal=LPTIM1_VS_LPTIM_counterModeInternalCl
ock
VP_LPTIM2_VS_LPTIM_counterModeInternalClock.Mode=Counts__internal_clock_event_00
VP_LPTIM2_VS_LPTIM_counterModeInternalClock.Signal=LPTIM2_VS_LPTIM_counterModeInternalCl
ock
VP_PWR_VS_DBSignals.Mode=DisableDeadBatterySignals
VP_PWR_VS_DBSignals.Signal=PWR_VS_DBSignals
VP_SYS_VS_Systick.Mode=SysTick
VP_SYS_VS_Systick.Signal=SYS_VS_Systick
board=STM32L562E-DK
boardIOC=true
isbadioc=false
```

This file will be used as a template when creating the project.

2.1.2. Creating the Project

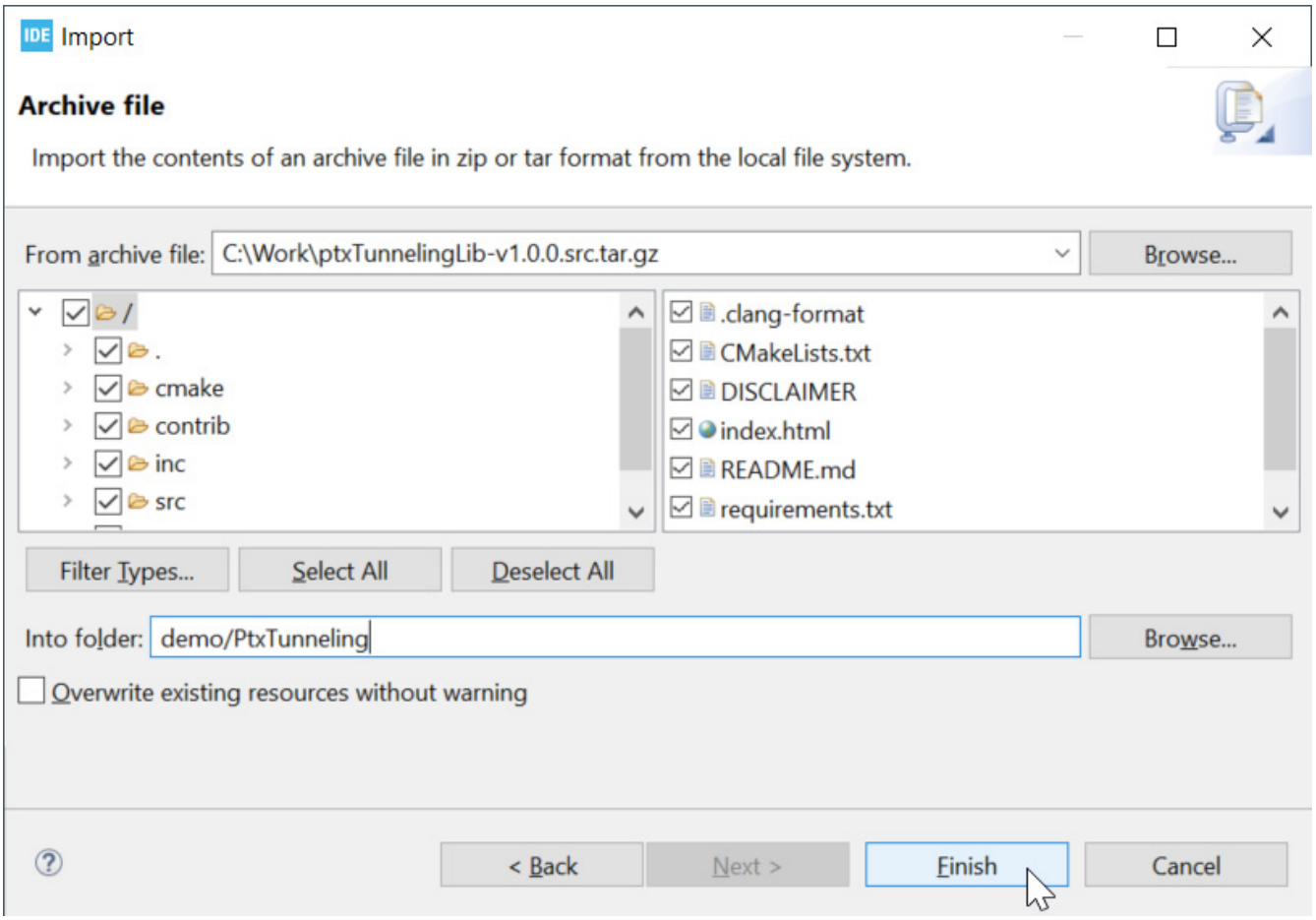
Create a new project in *stm32cubeide* from Menu **File > New > STM32Project from an Existing STM32CubeMX Configuration file (*.ioc)** with selecting the `ptxTunneling.ioc` file in the dialog window. After specifying a name to the project, it can be generated by clicking on the **Finish** button. If the requested SDK version has not been downloaded yet, this process may take a few minutes to complete.



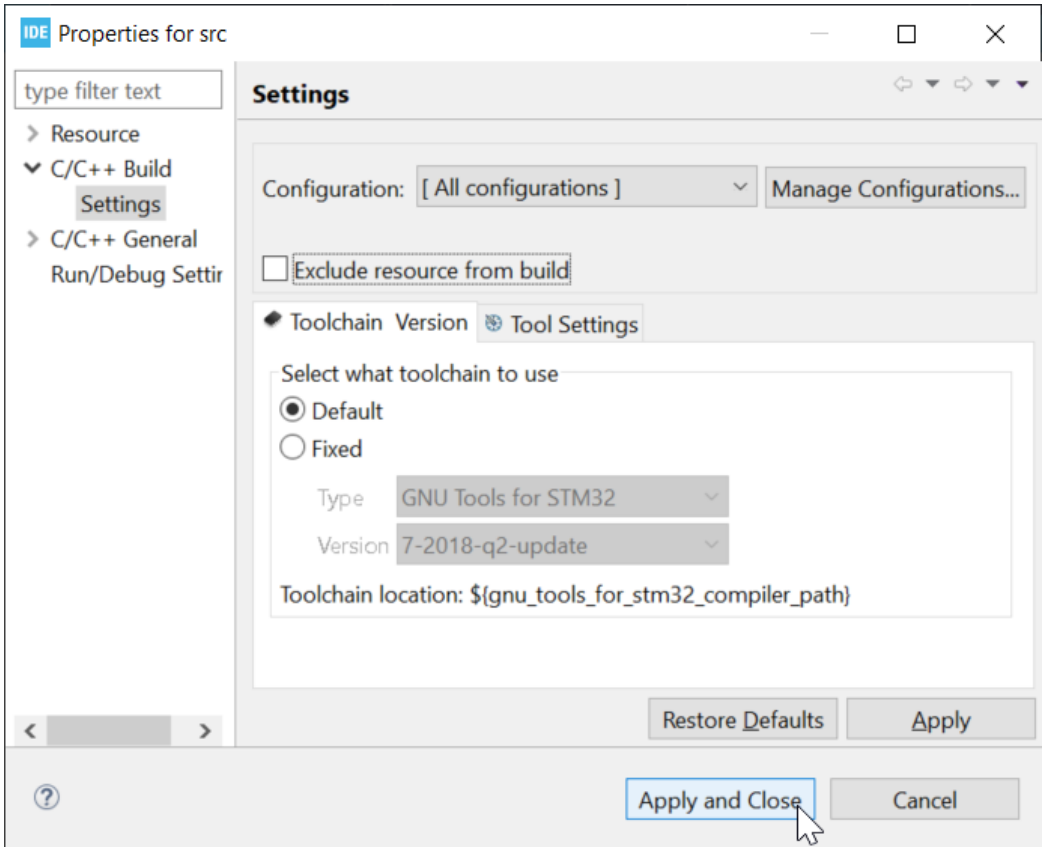
2.2 Importing the Library

There is no difference whether the source or the precompiled package is being used: the library archive must be imported to the project using **File > Import > Archive File**.

To keep the folder structure clean, the library will be imported to the `PtxTunneling` subfolder by appending it to the default location displayed in following figure.

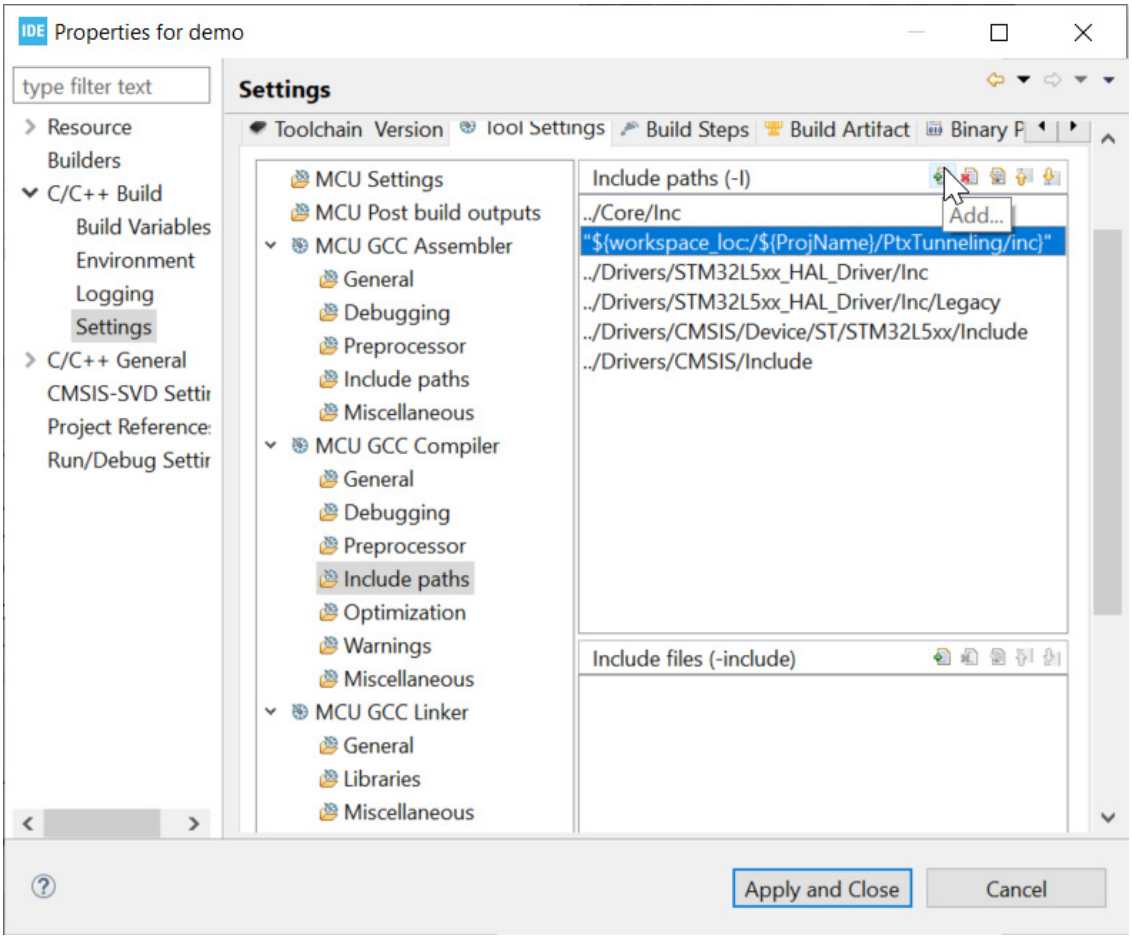


In the case where the library will be used in source form, the subfolder `PtxTunneling` and `PtxTunneling/src` folders must be included in the build. This can be done by opening the context menu with a right-mouse click on the folder name in the **Project Explorer** and selecting **Properties**. The checkbox **Exclude resource from build** found in **C/C++ Build > Settings** must be unchecked as displayed in the following figure.



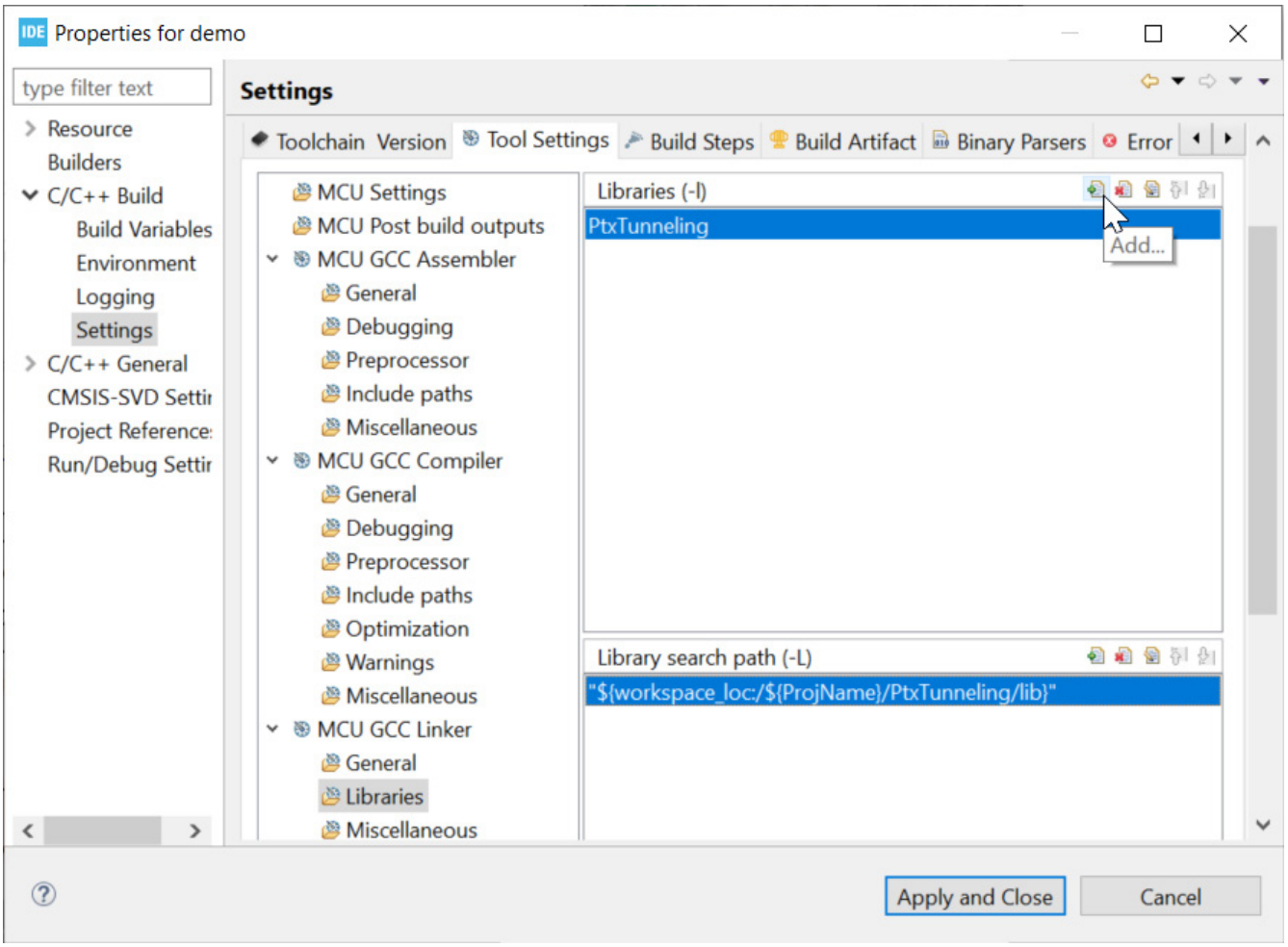
2.2.1. Adding the Include Path

In order for the compiler to find the header (.h) files containing the API functions, the library folder `inc` must be added to the list of user-defined include directories. This can be done by navigating to **Project > Properties > C/C++ Build > Settings > Tool Settings > MCU GCC Compiler > Include paths**. Next, click on the **Add** button on the right side of the small toolbar and use the **Workspace** button in the popup window to locate the folder.



2.2.2. Adding the Library File

This step is required only if you are working with the precompiled binary package. Since there is no source code to be compiled, the linker must be able to find the functions in the library. In the same dialog window, changing to **MCU GCC Linker > Libraries** section, the `PtxTunneling/lib` folder can be added to the list of folders (lower pane) where the compiler is looking for external libraries. Additionally, the exact library needs also to be specified (upper pane) by its name `PtxTunneling`. From this the compiler will automatically find the static library file `libPtxTunneling.a`.



2.3 Implementing the HAL

The library functions cannot access the underlying hardware or software resources; they require to access the Hardware Abstraction Layer (HAL) which then performs the requested action. Since this layer depends on the specific hardware configuration, it must be implemented for the exact setup.

The PtxTunneling library includes the header file ptx_tunneling_hal.h. It contains all the functions that must be provided by the host platform.

For the current case there should be the file `ptx_tunneling_hal.c` created in the source code folder `Core/Src` with the following content.

```
/*
    ------------------------------------------------------------------
    SPDX-License-Identifier: BSD-3-Clause

    Copyright (c) 2024, Renesas Electronics Corporation and/or its affiliates


    Redistribution and use in source and binary forms, with or without modification,
    are permitted provided that the following conditions are met:

    1. Redistributions of source code must retain the above copyright notice, this
       list of conditions and the following disclaimer.

    2. Redistributions in binary form must reproduce the above copyright notice, this
list of
       conditions and the following disclaimer in the documentation and/or other
       materials provided with the distribution.

    3. Neither the name of Renesas nor the names of its
       contributors may be used to endorse or promote products derived from this
       software without specific prior written permission.



    THIS SOFTWARE IS PROVIDED BY Renesas "AS IS" AND ANY EXPRESS
    OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
    OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE ARE
    DISCLAIMED. IN NO EVENT SHALL RENESAS OR CONTRIBUTORS BE
    LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
    CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
    GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
    HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
    LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
    OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
    ------------------------------------------------------------------

    Project     : PtxTunneling
    Module      : HAL
    File        : ptx_tunneling_hal.c

    Description : Implementation of HAL for tunneling
*/
/*
 *
###########################################################################################
############################
 * INCLUDES
 *
###########################################################################################
############################
 */
#include <assert.h>
#include <cmsis_compiler.h>
#include <string.h>

#include "main.h" // for GPIO/SPI/Timer names
#include "stm32l5xx.h"

#include "ptx_tunneling_hal.h"

/*
```

```
 *
###############################################################################################
############################
 * INTERNAL FUNCTIONS
 *
###############################################################################################
############################
 */

// user definable context not used
struct ptxHal
{
};

// to access the peripherals directly
UART_HandleTypeDef *hUart.
extern SPI_HandleTypeDef hspi3;

/*
 *
###############################################################################################
############################
 * private FUNCTIONS
 *
###############################################################################################
############################
 */
#define OFFSET_RSP_LENGTH_BYTE 0
#define OFFSET_CMD_LENGTH_BYTE 1
#define OFFSET_CMD_CODE_BYTE 0

#define COMMS_MAX_MESSAGE_LENGTH 280
#define COMMS_HEADER_SIZE 2

#define CMD_CODE_TUNNELING_MSG 0x55

static uint8_t uartRxBuf[2048];
static uint16_t readPos = 0;
static uint16_t writePos = 0;
static uint8_t rx[COMMS_MAX_MESSAGE_LENGTH];
static uint16_t rxi = 0;

void startUartReceiving(UART_HandleTypeDef *huart)
{
    // save handle for future use
    hUart = huart;

    // start receiving the header
    HAL_StatusTypeDef st = HAL_UART_Receive_IT(huart, rx, 1);
    assert(!st);
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    // receive the header and the length byte
    if (rxi != 0 || rx[rxi] == CMD_CODE_TUNNELING_MSG)
    { // if there are some invalid data received while waiting for sync byte, just
discard them
        rxi++;
    }
```

```
    if (rxi >= COMMS_HEADER_SIZE)
    {
        uint16_t packLen = COMMS_HEADER_SIZE +
                           (rx[OFFSET_CMD_LENGTH_BYTE] == 0 ? 256 :
rx[OFFSET_CMD_LENGTH_BYTE]);
        if (rxi >= packLen)
        {
            // whole packet has been received
            memcpy(uartRxBuf + writePos, rx, rxi);
            writePos += rxi;
            rxi = 0;
        }
    }

    // continue receiving data
    HAL_UART_Receive_IT(huart, &rx[rxi], 1);
}
/*
 *
############################################################################################
############################
 * API FUNCTIONS
 *
############################################################################################
############################
 */

bool ptxTunneling_GPIO_IsIrqPinAsserted(ptxHal_t *context)
{
    return HAL_GPIO_ReadPin(PTX_IRQ_GPIO_Port, PTX_IRQ_Pin) == GPIO_PIN_SET;
}

int ptxTunneling_UART_rxLength(ptxHal_t *context)
{
    UNUSED(context);
    __disable_irq();
    const count = writePos - readPos;
    __enable_irq();
    return count;
}

int ptxTunneling_UART_read(ptxHal_t *context, uint8_t *buf, unsigned int len)
{
    UNUSED(context);
    assert(len < sizeof(uartRxBuf));
    __disable_irq();
    int readCount = writePos - readPos;
    if (readCount > len)
        readCount = len;

    memcpy(buf, uartRxBuf + readPos, readCount);

    readPos += readCount;
    if (readPos == writePos)
    {
        readPos = 0;
        writePos = 0;
    }
    __enable_irq();
    return readCount;
}
```

```
int ptxTunneling_UART_write(ptxHal_t *context, const uint8_t *buf, unsigned int len)
{
    UNUSED(context);
    HAL_StatusTypeDef res = HAL_OK;
    if (len)
    {
        do
        {
            res = HAL_UART_Transmit(hUart, buf, len, 1000);
        } while (res == HAL_TIMEOUT);
        assert(!res);
    }

    return res;
}

void ptxTunneling_Timer_stopwatchStart(ptxHal_t *context, ptxTimeDiff_t *startVal)
{
    *startVal = HAL_GetTick() * 1000;
}

void ptxTunneling_Timer_stopwatchStop(ptxHal_t *context, ptxTimeDiff_t *startStopVal)
{
    *startStopVal = HAL_GetTick() * 1000 - *startStopVal;
}

void ptxTunneling_Timer_ThreadSleep(ptxHal_t *context, uint32_t msSleep)
{
    HAL_Delay(msSleep);
}

void ptxTunneling_NVIC_disableInterrupts()
{
    __disable_irq();
}

void ptxTunneling_NVIC_enableInterrupts()
{
    __enable_irq();
}

int ptxTunneling_SPI_trx(ptxHal_t *context, uint8_t *const txBuf[], const size_t
txLen[],
    size_t numBuffers, uint8_t *rxBuf, size_t *rxLen)
{
    const uint32_t spiTimeout = 100000;
    HAL_StatusTypeDef st = HAL_ERROR;

    /* At this point the SPI transfer operation is triggered */
    HAL_GPIO_WritePin(SPI3_NSS_GPIO_Port, SPI3_NSS_Pin, GPIO_PIN_RESET);

    // Tx operation is required always: to send and to receive anything on SPI. So, tx
buffers have
    // to be provided always.
    if ((NULL != txBuf) && (NULL != txLen))
    {
        /* Tx part of the overall transaction. */
        size_t index = 0;
        while (index < numBuffers)
        {
```

```
            assert((txBuf[index] != NULL) && (txLen[index] > 0));

            st = HAL_SPI_TransmitReceive(&hspi3, txBuf[index], rxBuf, txLen[index],
spiTimeout);
            assert(!st);
            if (st != HAL_OK)
            {
                break;
            }
            index++;
        }
    }
    else if ((NULL != rxBuf) && (NULL != rxLen) && (*rxLen > 0))
    /* Let's see if there is something to read. */
    {
        st = HAL_SPI_Receive(&hspi3, rxBuf, (uint16_t)(*rxLen), spiTimeout);
        assert(!st);
    }

    /* In any case, at this point the SPI transfer operation is finished */
    HAL_GPIO_WritePin(SPI3_NSS_GPIO_Port, SPI3_NSS_Pin, GPIO_PIN_SET);

    return st;
}
```

Note: This implementation is specific to the ST32L562E-DK board and is not guaranteed to work on any other hardware.

2.4 Calling the Library Functions

The main loop will provide the tunneling functionality by calling the library's superloop function, the `ptxTunneling_poll()`. This function performs the data processing and translation, and also the SPI communication. The **main()** function can be found in the `Core/Src/main.c` file. Update this file with the following code:

```
/* USER CODE BEGIN PV */
#include "ptx_tunneling.h"  // including the library functions
void startUartReceiving(UART_HandleTypeDef *huart); // implemented in HAL
/* USER CODE END PV */

[...] // existing code not shown

int main(void)
{
  [...] // lines of initialization code not shown

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  startUartReceiving(&huart1);
  ptxTunneling_init();
  while (1)
  {
    ptxTunneling_poll(NULL);
    /* USER CODE END WHILE */

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

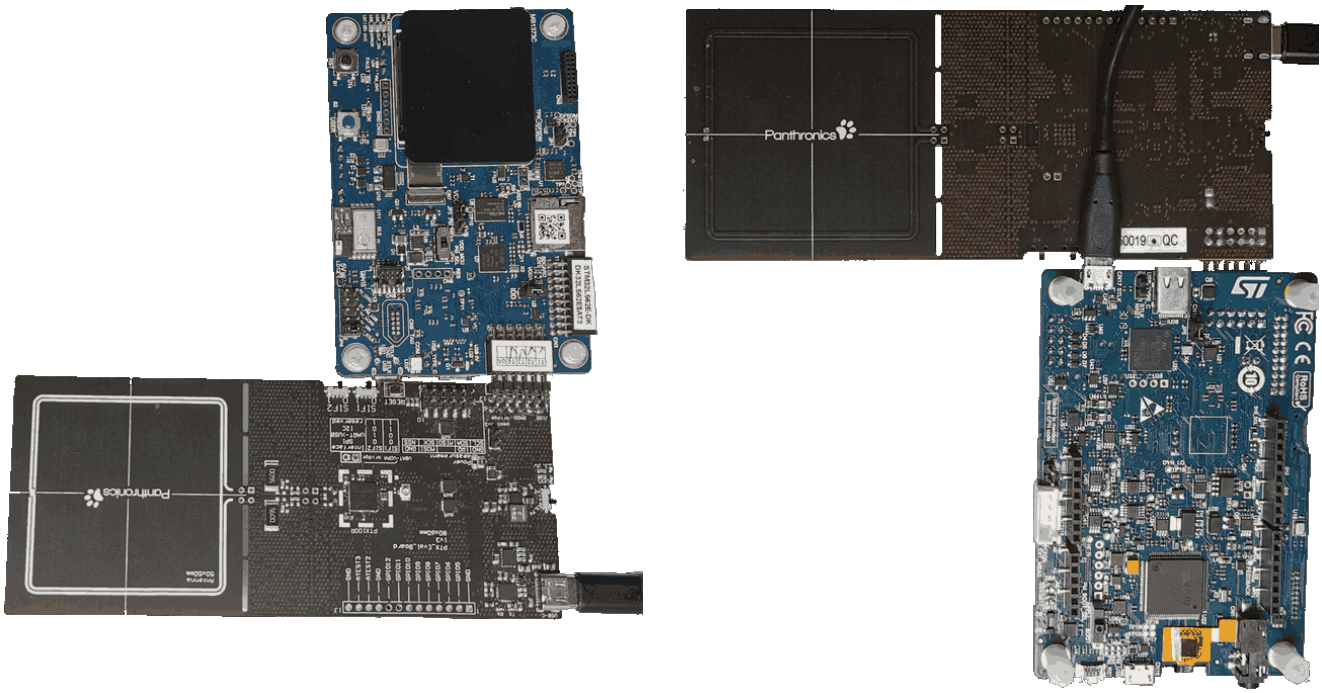
2.5 Building the Firmware

After the source files have been created, the project can be built with the **Project > Build Project**. When the build process has finished successfully, a table similar to the following will show with the footprint sizes.

```
arm-none-eabi-size --format=berkeley "demo.elf"
   text    data     bss     dec     hex filename
  23460     124    7620   31204    79e4 demo.elf
```

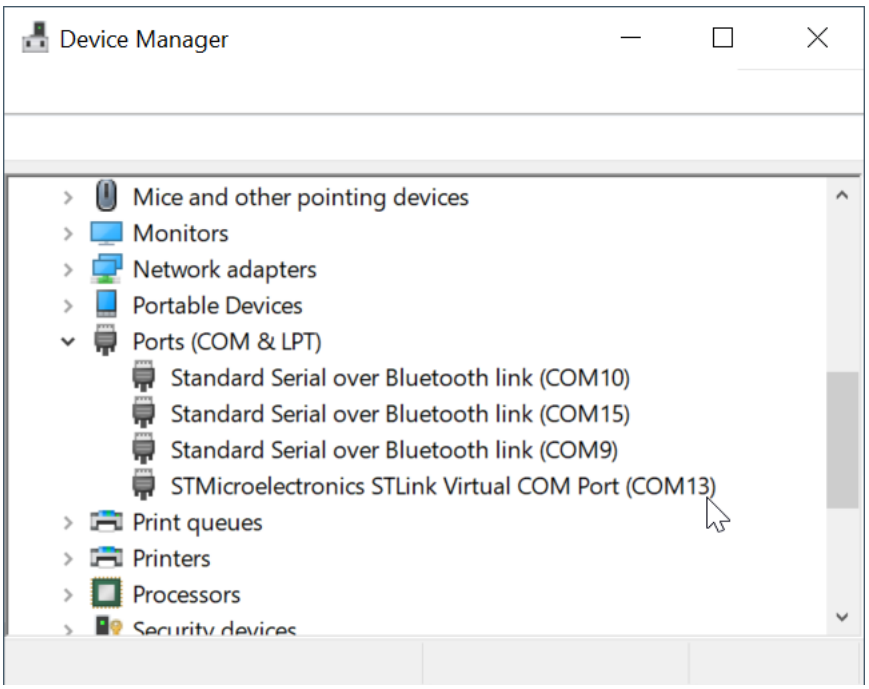
3. Preparing the Hardware

To demonstrate the usage of the tunneling functionality, a PTX evaluation board must be connected to the ST32L562E Evaluation Kit through the PMOD connector.



Before powering up the PTX evaluation board from the USB-C port, it is important to remove the jumper named **pmod 3v3** and set the **Serial Interface switches** to **"0"** state to select the SPI communication protocol.

The ST32L562E-DK board must be connected to the host PC via the **CN17 STLNK** USB connector. This connection provides both a UART interface named **STMicroelectronics STLink Virtual COM Port** (used in PTX100x * Config Tool) and access to the on-board ST-Link debugger.

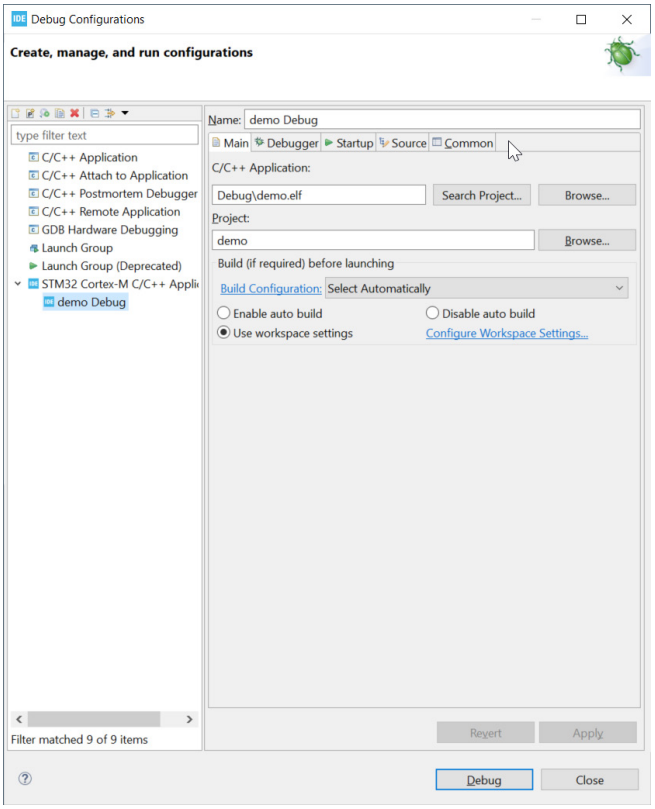


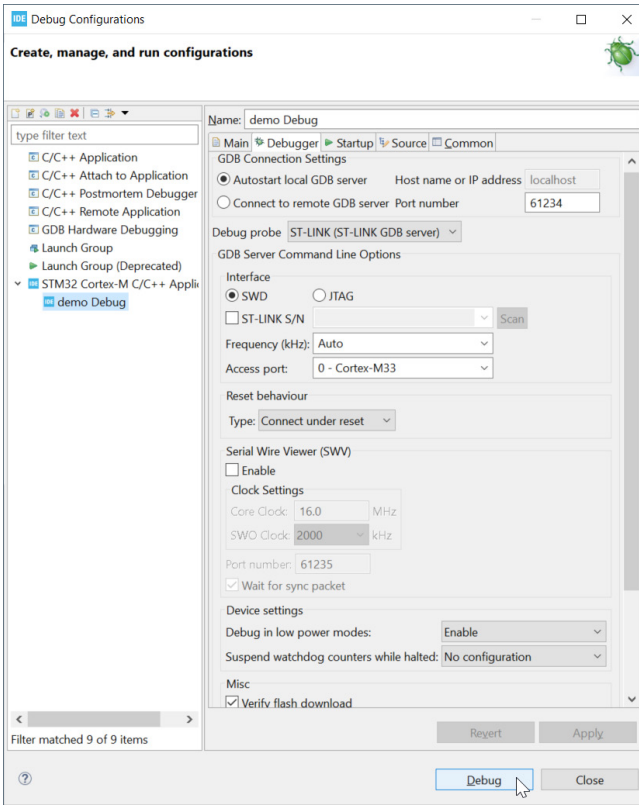
3.1 Debug Configuration

In order to upload the firmware onto the device, the Debug configuration must be created from the **Run > Debug Configurations**. Please create a new configuration (if not yet present) for the **STM32 Cortex-m C/C++ Application**. The most important settings are:

- Select the `Debug/demo.elf` application from the project
- Select ST-LINK (ST_LINK GDB server) among the debug probes

When the settings are updated, pressing the **Debug** button initiates the debug session with flashing the firmware onto the chip.

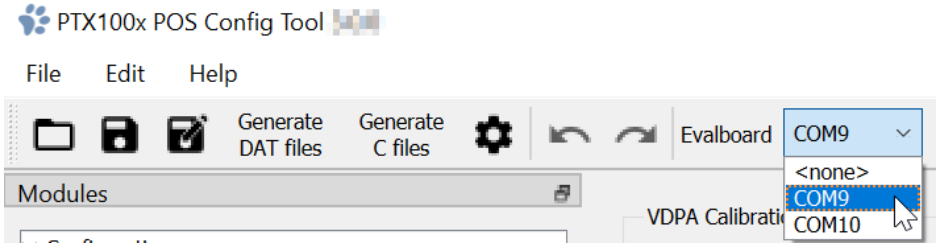




The firmware execution will halt at the main function by default; `F8` must then be pressed to continue the execution.

4. Using the Tunneling Feature

To use of the tunneling functionality, the **PTX100x * Config Tool** must be started and configured to use the USB serial communication port (identified in Device manager previously) by selecting the correct entry in the dropdown list in toolbar.



The configuration is now ready. Any test started will communicate with the PTX100x via the tunneling firmware.

5. Revision History

Revision	Date	Description
1.01	Jun 18, 2024	Updated license text in ptx_tunneling_hal.c file.
1.00	Jan 15, 2024	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.