OPENBLT MIGRATION ON NXP MCU - V1_1

DAWEI

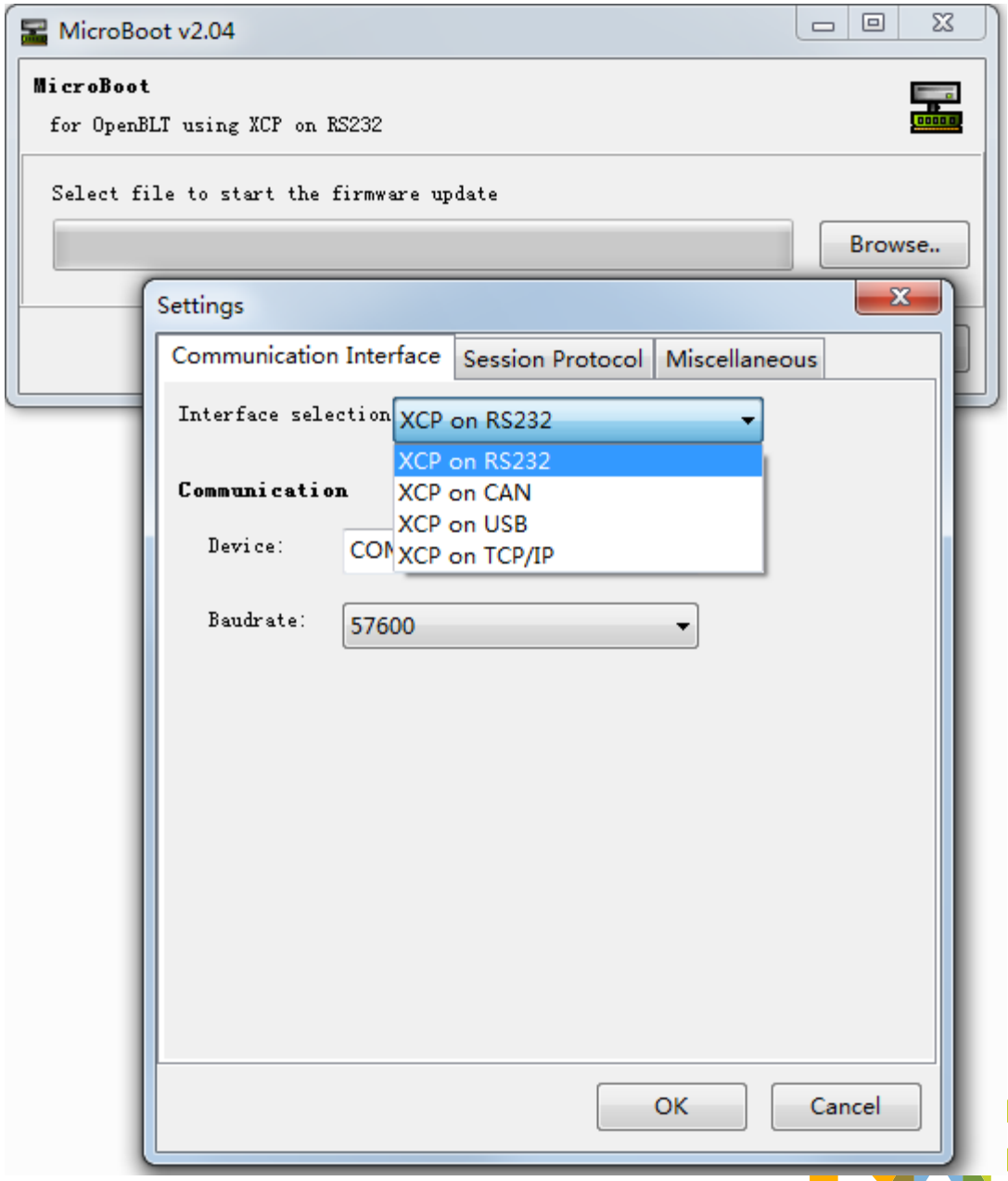
CAS GC

OpenBLT migration on NXP MCU

- 官网地址：
- https://www.feaser.com/en/openblt.php

- Wiki使用说明：
- https://www.feaser.com/openblt/doku.php

- Github镜像：
- https://github.com/feaser/openblt



OpenBLT migration on NXP MCU

家 » OpenBLT引导程序

FEASER

OpenBLT GNU GPL引导程序

OpenBLT是**用于基于微控制器的产品的开源引导加载程序。**它通过常见的通讯和存储介质（例如**RS232，USB，CAN，TCP/IP和SD卡）**以用户友好的方式使**固件更新成为**可能。OpenBLT用C编程语言编程。

在经过**1次积分**的引导程序到您的基于微控制器的产品，你的最终用户或技术人员可以工作与引导加载程序，以**方便地进行固件更新。**由于可以**在最终用户的位置进行**固件更新，因此无需将产品运回给您进行维修。这样可以减少停机时间并**提高客户对**您产品**的满意度。**

开源的

我们认为，开源是发布引导加载程序产品**的唯一明智的方法。**它为用户提供了通常需要的**灵活性，可以根据自己的特定需求和意愿调整引导加载程序的功能。**此外，我们希望您看到源代码的质量，因为引导加载程序已成为产品不可或缺的一部分。

支持目标

由于STM32微控制器的普及，OpenBLT主要用作**STM32引导程序。**但是，引导加载程序的体系结构**使其可以与任何微控制器一起使用。**目前，它支持**ST STM32，Infineon XMC，NXP S12以及TI TM4C和LM3S微控制器。**源代码组织在依赖于微控制器的功能和独立于功能之间建立了隔离，**从而可以轻松地将引导加载程序移植**到尚不支持的微控制器目标上。

入门

第一步是**下载OpenBLT。**单击下面的下载按钮可获得最新版本。下载包包含用于流行的微控制器**评估板的许多预配置演示程序。**您可以从这些低成本的板上获得一个，以首先熟悉OpenBLT引导加载程序的操作，或者直接学习并将演示程序移植到自己的硬件上。

有关其他信息，包括对引导加载程序内部工作原理的深入说明，我们**将一个单独的Wiki网站专门用于OpenBLT项目。**点击下面的链接访问该网站。

Download OpenBLT　Visit OpenBLT Wiki　FAQ

OpenBLT migration on NXP MCU

支持多种特性，程序加密及基本的校验和。

校验和是在程序写入的时候，由bootloader创建者指定生成校验值并写入特定位置（比如中断向量表的空闲位置），并在每次启动时进行检查。

固件加密

通过使用OpenBLT，您使用户可以更新产品上的固件。根据您的应用程序，这可能会**带来潜在的负面影响**，例如：

1. 您可以将固件文件提供给用户。这使他们**可以访问您的专有程序代码。**
2. 它使第三方可以**对产品上不需要的固件**进行编程。

固件加密模块可以消除这些有害的副作用。固件文件中的程序代码**使用唯一的256位AES加密密钥加密。**自举程序在固件更新过程中接收到程序代码后，将对程序代码进行解密，然后再在微控制器的非易失性存储器中对其进行编程。

实际的加密是通过**命令行程序执行的。**在编译和链接软件程序的源代码之后，可以手动调用或将其作为自动的后构建步骤来调用。

该附加模块的交付包含：

- ☐ 适用于您指定的微控制器和编译器的演示OpenBLT程序，支持程序代码解密。
- ☐ 用于指定的微控制器和编译器的演示用户程序，该用户程序配置为在构建后的步骤中自动调用加密命令行程序。
- ☐ 加密命令行程序，包括完整的源代码。
- ☐ 详细的用户手册。

Generate Quote

改进的校验和

我们提供了一种改进的校验和模块，该模块**在整个程序代码上**使用**CRC-16校验和。**使用此模块，可以确定**引导程序仅在闪存中程序代码的所有位均正确时才启动用户程序。**从而防止您的用户程序在以下情况下潜在地损害您的系统：有人**篡改**了您的微控制器，发生了闪存**故障**或**意外更改**了闪存内容。

包含一个**命令行程序**，用于计算固件整个程序代码中的校验和信息。然后，它将**校验和信息修补到固件文件中。**在编译和链接软件程序的源代码之后，可以手动调用或将其作为自动的后构建步骤来调用。

该附加模块的交付包含：

- ☐ 适用于您指定的微控制器和编译器的OpenBLT演示程序，该程序支持在启动用户程序之前验证改进的校验和。
- ☐ 用于指定的微控制器和编译器的演示用户程序，该程序具有为校验和表保留的空间。它被配置为在构建后的步骤中自动调用校验和生成命令行程序。
- ☐ 校验和生成命令行程序，包括完整的源代码。
- ☐ 详细的用户手册。

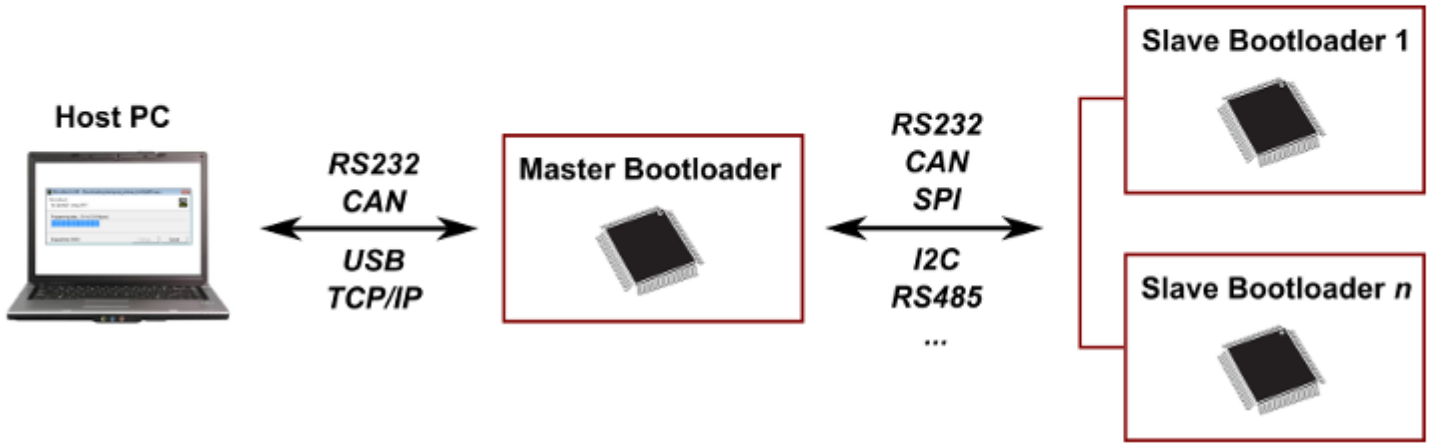
OpenBLT migration on NXP MCU

上位机既支持从PC发起也支持从某个嵌入式设备发起。

OpenBLT提供了相应的demo及程序参考。

主/从网关

在具有主机微控制器和一个或多个从属微控制器的系统中，通常只有主机具有与外界的开放通信线路。在主机上运行OpenBLT引导加载程序时，可以在主机上进行固件更新，但是如何在从机上执行固件更新？主/从网关附加模块是此问题的答案。该附加模块 **将网关添加到在主服务器上运行的OpenBLT引导加载程序，从而允许通过该网关在从服务器上进行固件更新。**



使用MicroBoot或 BootCommander启动固件更新时 ，可以配置**connection-mode参数。**这是连接到引导加载程序时发送的8位值。当主服务器上的OpenBLT引导加载程序收到连接请求时，将评估connection-mode参数的值。如果为零，则固件更新将照常在主机上继续进行。如果该值大于零，则网关将被激活，并且连接请求将传递到主/从网络，在该网络上，从服务器上运行的OpenBLT引导加载程序会检测到该请求。使用此解决方案，**最多可以选择255个单独的从站**进行固件更新。

主/从网关模块包含高级网关功能和低级通信之间的隔离层。这使网关可以与任何主/从网络一起工作。已**通过RS232，CAN，SPI，I2C和RS485成功 测试。**

该附加模块的交付包含：

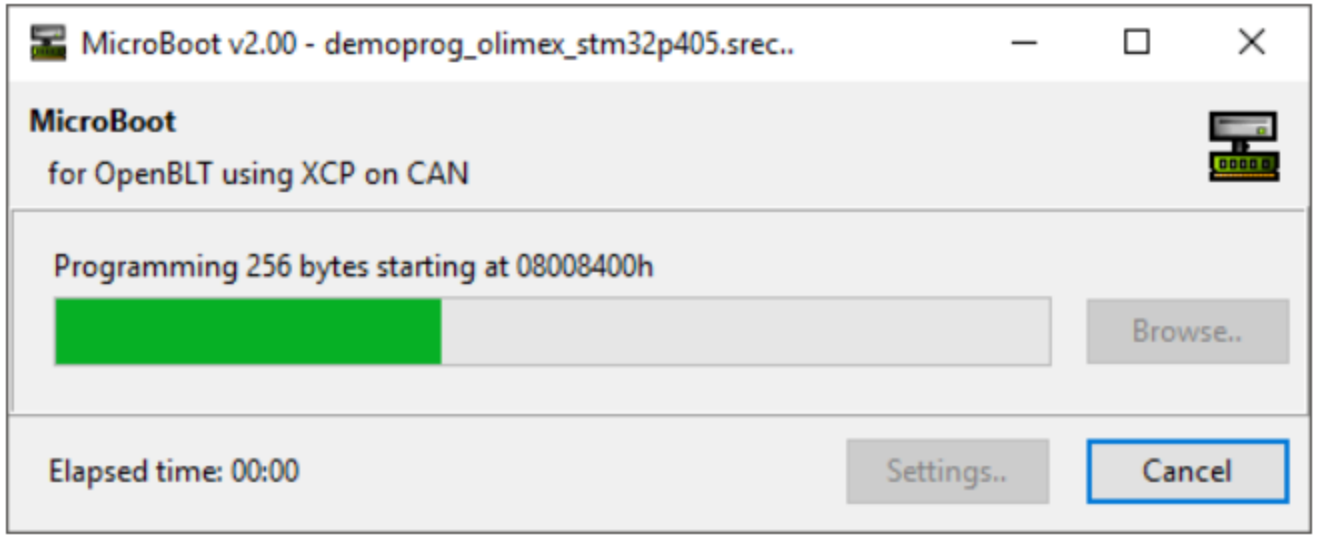
- 适用于您的主微控制器的演示OpenBLT引导加载程序，具有针对从属网络的通信类型集成和配置的主/从网关。
- 适用于从属微控制器的演示OpenBLT引导加载程序，已配置为通过从属网络的通信类型进行固件更新。
- 主机和从机上的演示用户程序，用于测试主机和从机上的固件更新过程。
- 详细的用户手册。

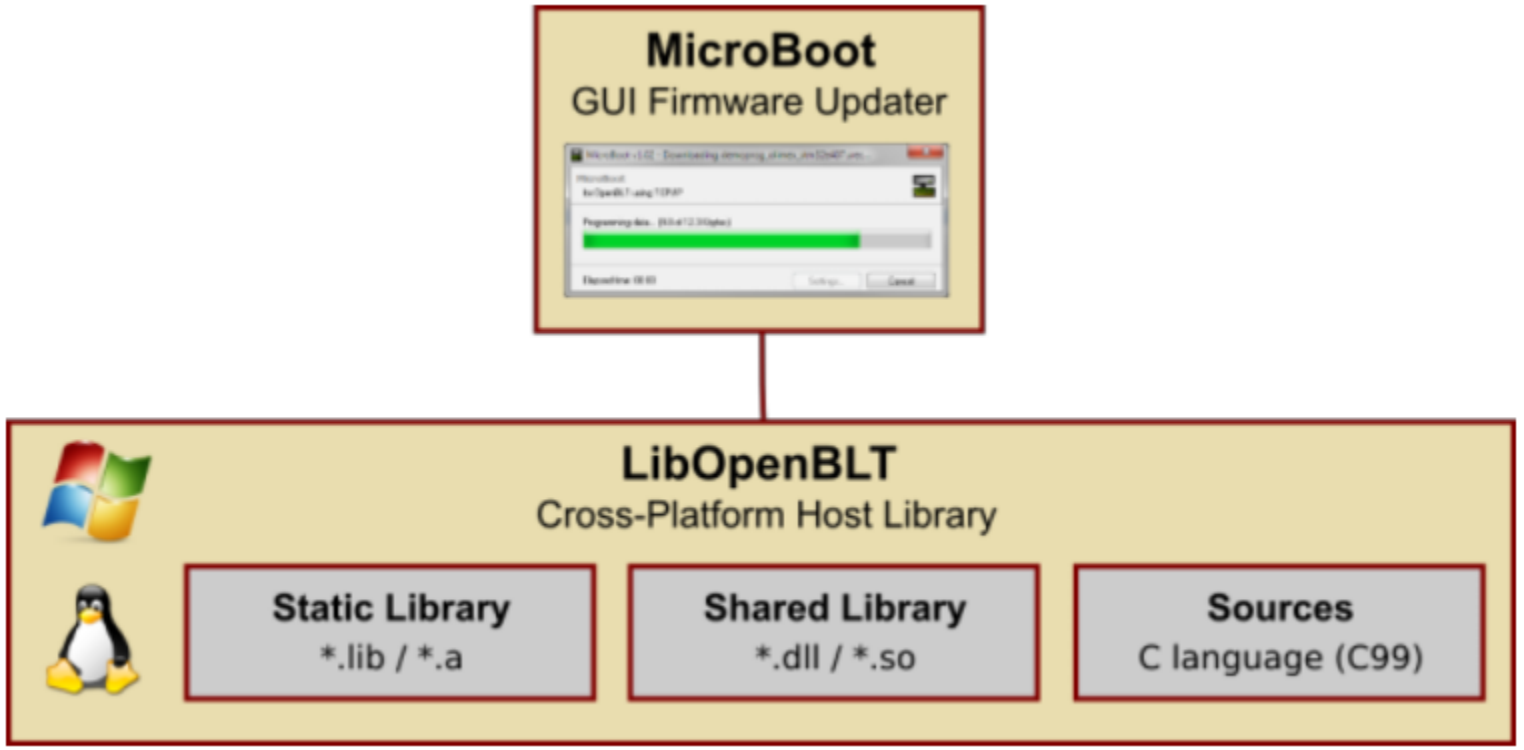
MicroBoot Utility/BootCommander Utility

MicroBoot was written in Object Pascal using the Lazarus IDE and developed with cross-platform support in mind. It has been successfully tested on a Windows PC, Linux PC and even runs on a Raspberry PI.

Under the hood, MicroBoot relies on the OpenBLT host library (LibOpenBLT) for handling the actual communication with the bootloader.

依赖LibOpenBLT库，既可以构建GUI上位机(MicroBoot)也有命令行工具（Boot Commander）。





MicroBoot Utility/BootCommander Utility

BootCommander was written in the C programming language (C99) and developed with cross-platform support in mind.

BootCommander [options] [firmware file]

-s=[name] specifies the name of the communication protocol. Supported names are:
xcp (default) = XCP version 1.0

-t=[name] specifies the name of the transport layer. Supported names are:
xcp_rs232 (default) = XCP on RS232.
xcp_can = XCP on CAN.
xcp_usb = XCP on USB.
xcp_net = XCP on TCP/IP.

BootCommander -d=COM4 demoprog.srec
BootCommander -s=xcp -t=xcp_rs232 -d=COM4 -b=57600 -t1=1000 -t3=2000 -t4=10000 -t5=1000 -t7=2000 demoprog.srec

Additional XCP version 1.0 communication protocol settings (**xcp**):

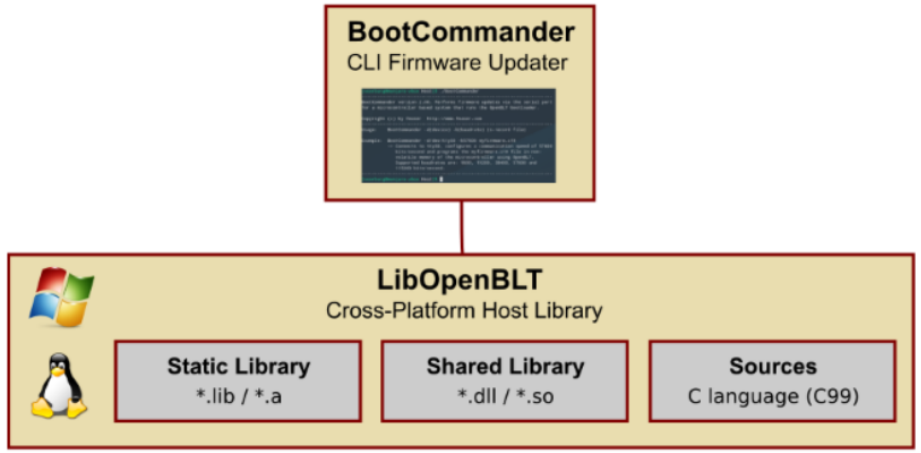
- **-t1=[timeout]** sets the command response timeout in milliseconds as a 16-bit value (default = 1000 ms).
- **-t3=[timeout]** sets the start programming timeout in milliseconds as a 16-bit value (default = 2000 ms).
- **-t4=[timeout]** sets the erase memory timeout in milliseconds as a 16-bit value (default = 10000 ms).
- **-t5=[timeout]** sets the program memory and target reset timeout in milliseconds as a 16-bit value (default = 1000 ms).
- **-t7=[timeout]** sets the busy wait timer timeout in milliseconds as a 16-bit value (default = 2000 ms).
- **-sk=[file]** sets the seek/key algorithm library filename (optional).
- **-cm=[value]** sets the connection mode value sent in the XCP connect command, as a 8-bit value (default = 0).

Additional XCP on RS232 settings (**xcp_rs232**):

- **-d=[name]** sets the name of the communication device. For example COM1 or /dev/ttyUSB0 (mandato
- **-b=[value]** sets the communication speed in bits per second, as a 32-bit value (default = 57600). Supp are: 9600, 19200, 38400, 57600, 115200.

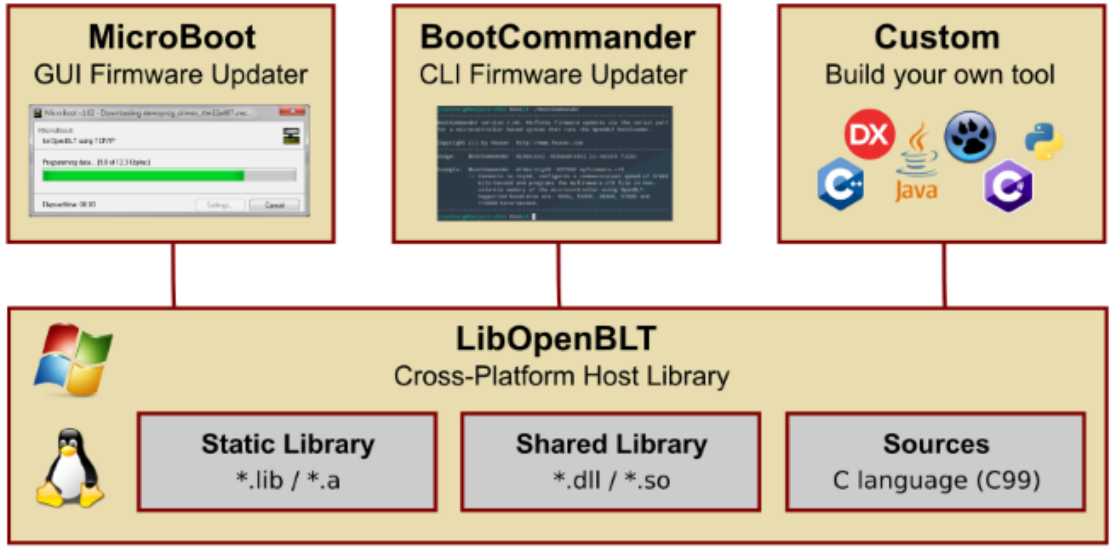
Additional XCP on TCP/IP settings (**xcp_net**):

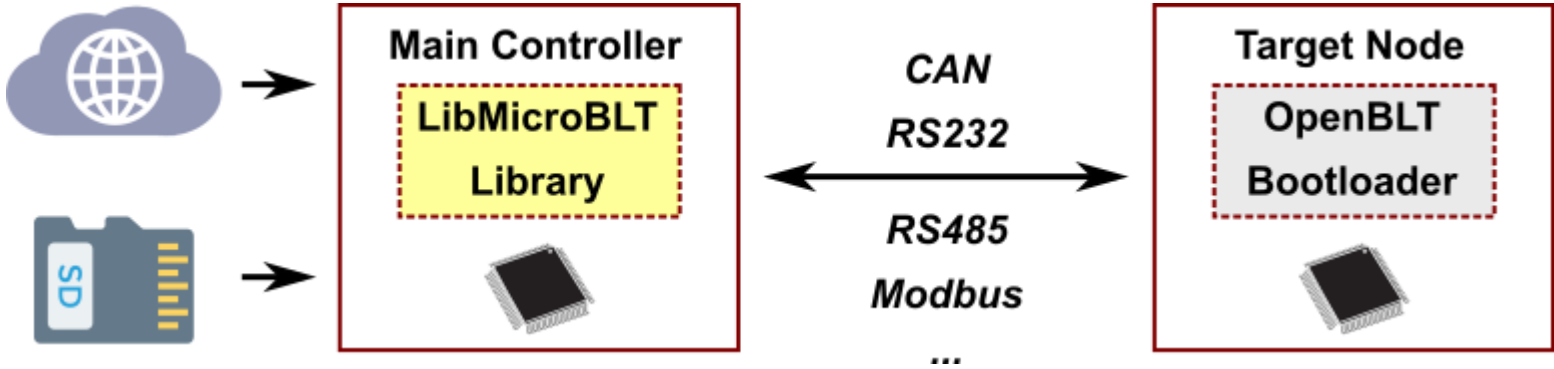
- **-a=[value]** sets the IP-address or hostname of the target to connect to. For exan
- **-p=[value]** sets the TCP port number to use, as a 16-bit value (default = 1000).



OpenBLT Host Library/OpenBLT Embedded Library

使用Library可以更容易地构建上位机，除了PC环境之外，
Embedded Library 可以用于构建嵌入式地上位机系统





OpenBLT migration on NXP MCU

OpenBLT的开源版本参照GPL V3，
对于公司应用最好使用商用版本。

本文使用开源版本，并对外释放作为
学习参考使用。

License Options

OpenBLT is open source and licensed under version 3 of the GNU GPL. **OpenBLT is free** under this license and **can be freely used and distributed** under its terms.

The freedom that this GNU GPL license offers, comes with **responsibilities and side effects** that are not always desirable. This is especially the case if you want to integrate OpenBLT in your **closed source product** and/or **don't want your customers to know** that your product contains OpenBLT.

As an alternative, OpenBLT can be made available under a **commercial license**. Under the commercial license, OpenBLT does not contain any references to the GNU GPL. Refer to the following **license comparison matrix** to determine the OpenBLT license your product requires.

Question	GNU GPL version 3	Commercial license
Is OpenBLT free?	yes	no
Do I have the right to change the OpenBLT source code?	yes	yes
Can I use OpenBLT in my closed source product?	no	yes
Do I have to open my source code?	yes	no
Do I have to open source my changes to OpenBLT?	yes	no
Do I have to offer the OpenBLT source code to users of my product?	yes	no
Do I have to document that my product uses OpenBLT?	yes	no
Can I redistribute OpenBLT in source code format?	yes	no
Can I receive professional technical support on a commercial basis?	no	yes

OpenBLT migration on NXP MCU

提供了多种已经移植好的平台，但是对于NXP目前仅仅只有以下几个平台。

NXP

Development board	Port specifics	Quickstart information		
NXP S32K118 EVB	ARM-CM0 for S32K11	GCC	IAR	
NXP S32K144 EVB	ARM-CM4 for S32K14	GCC	IAR	
EVBplus Dragon12-Plus	Freescale HCS12			Codewarrior
NXP DevKit S12G128	Freescale HCS12			Codewarrior

OpenBLT migration on NXP MCU – 目的

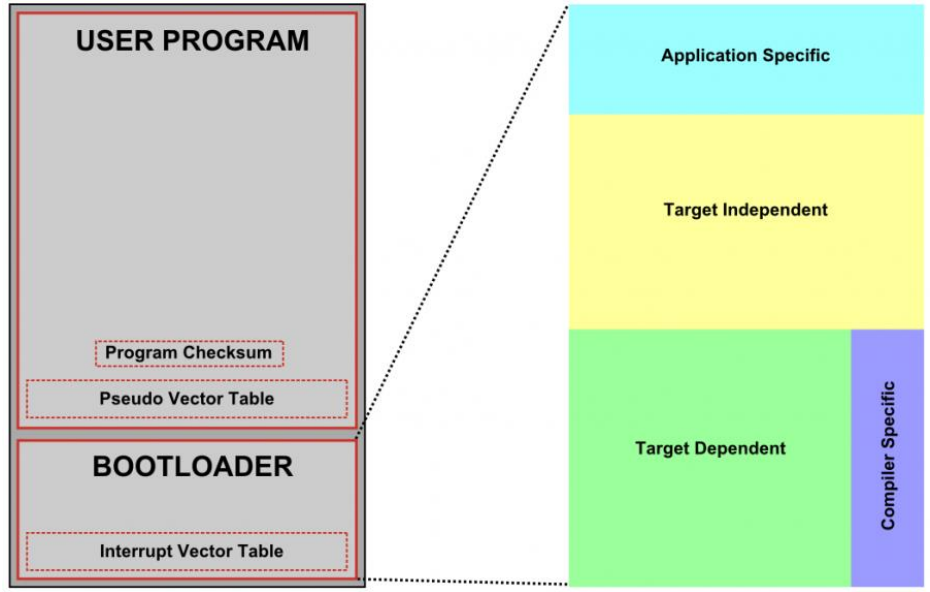
尝试在NXP MCU主流平台上移植OpenBLT（current version: openblt_v011300)

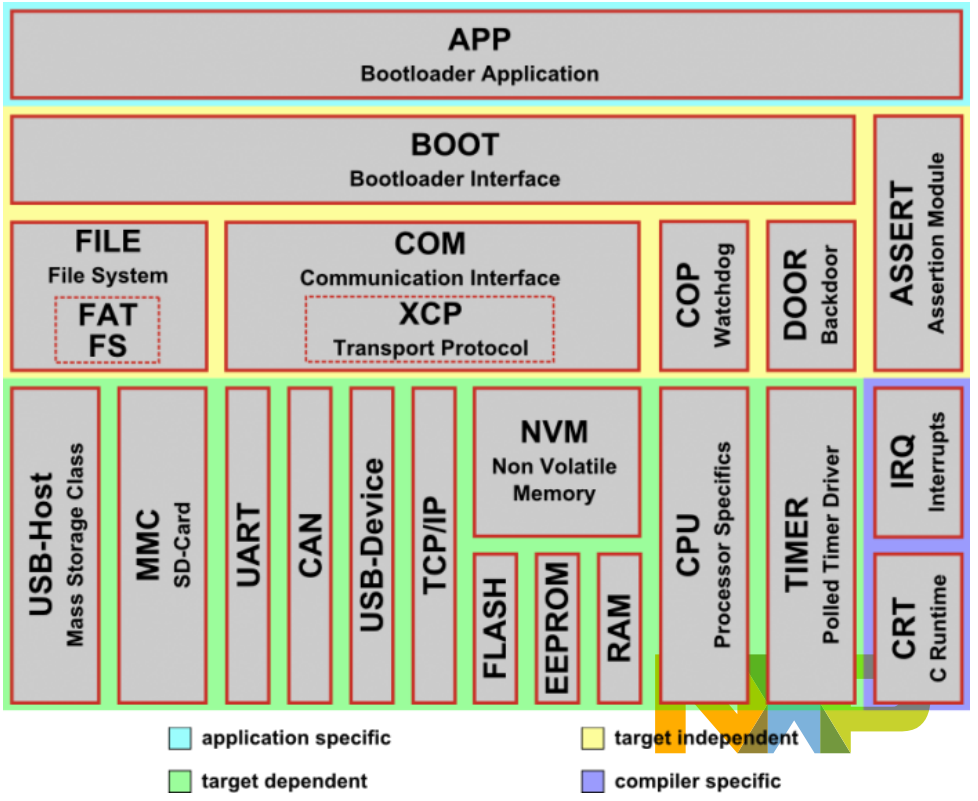
- 为NXP MCU提供使用和移植参考；尤其是如何使用NXP SDK去创建移植工程；
- 使用和分析常见的几个升级方式，并分析优略；
- 测试上位机及其软件
- 移植到LPC5536及i.MXRT1170，测试多种升级方式：串口、以太网、CAN总线
- 为NXP MCU两类比较特殊的Flash的移植提供建议和参考：Flexspi和基于ECC的Flash

OpenBLT Design Architecture

OpenBLT bootloader的构建至少需要用户构建最基本的硬件驱动层：

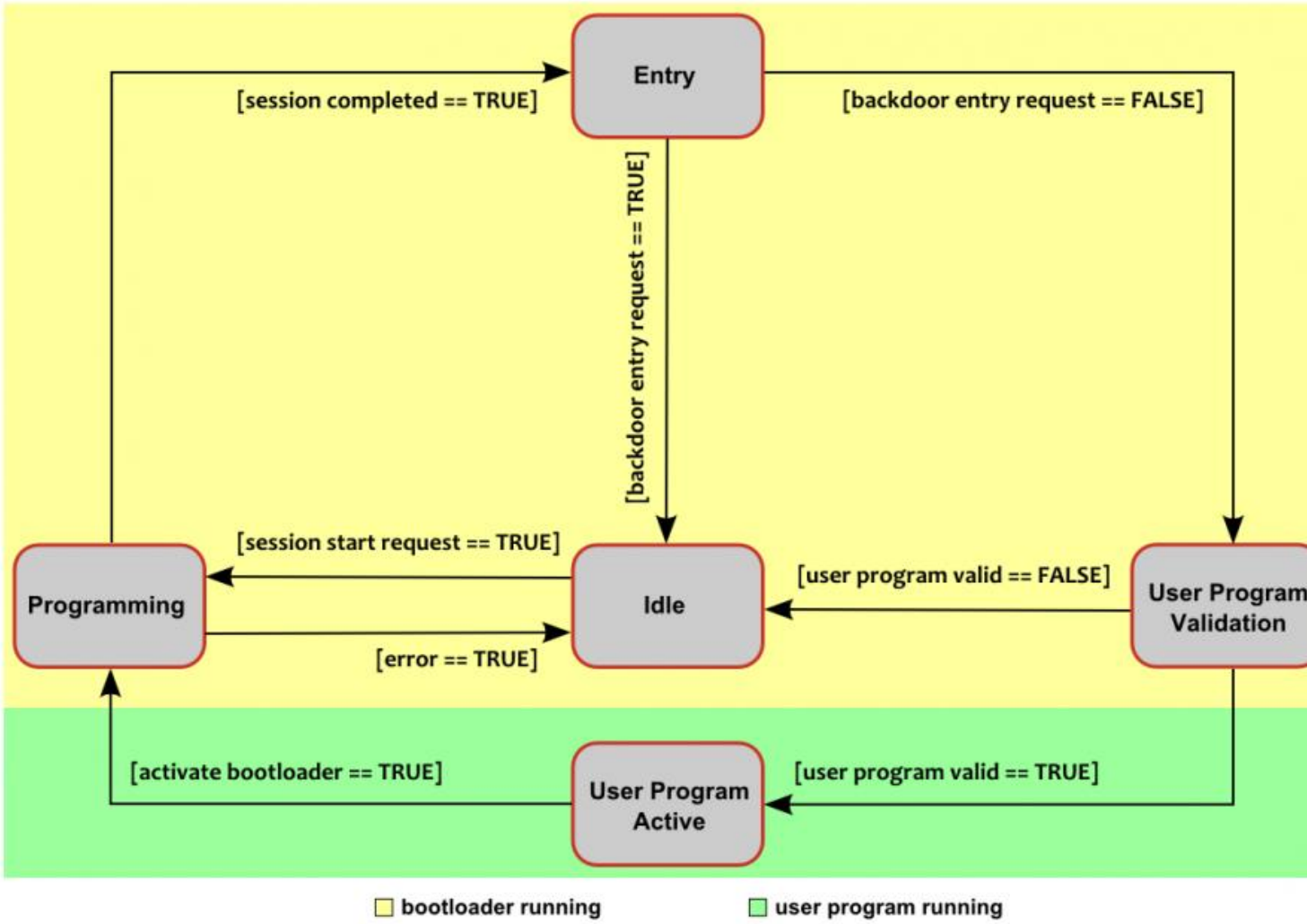
1. 提供一个能够在开发板上启动的最小系统；
2. 定时器（Systick或者其他）以测量时间；
3. Flash驱动以烧录更新APP
4. CPU specific的平台函数（使能禁止中断、中断向量表的跳转及重定向）
5. 依据所使用外设构建符合OpenBLT接口的外设驱动（至少包含UART、CAN、USB、以太网等中的一项或者多项）





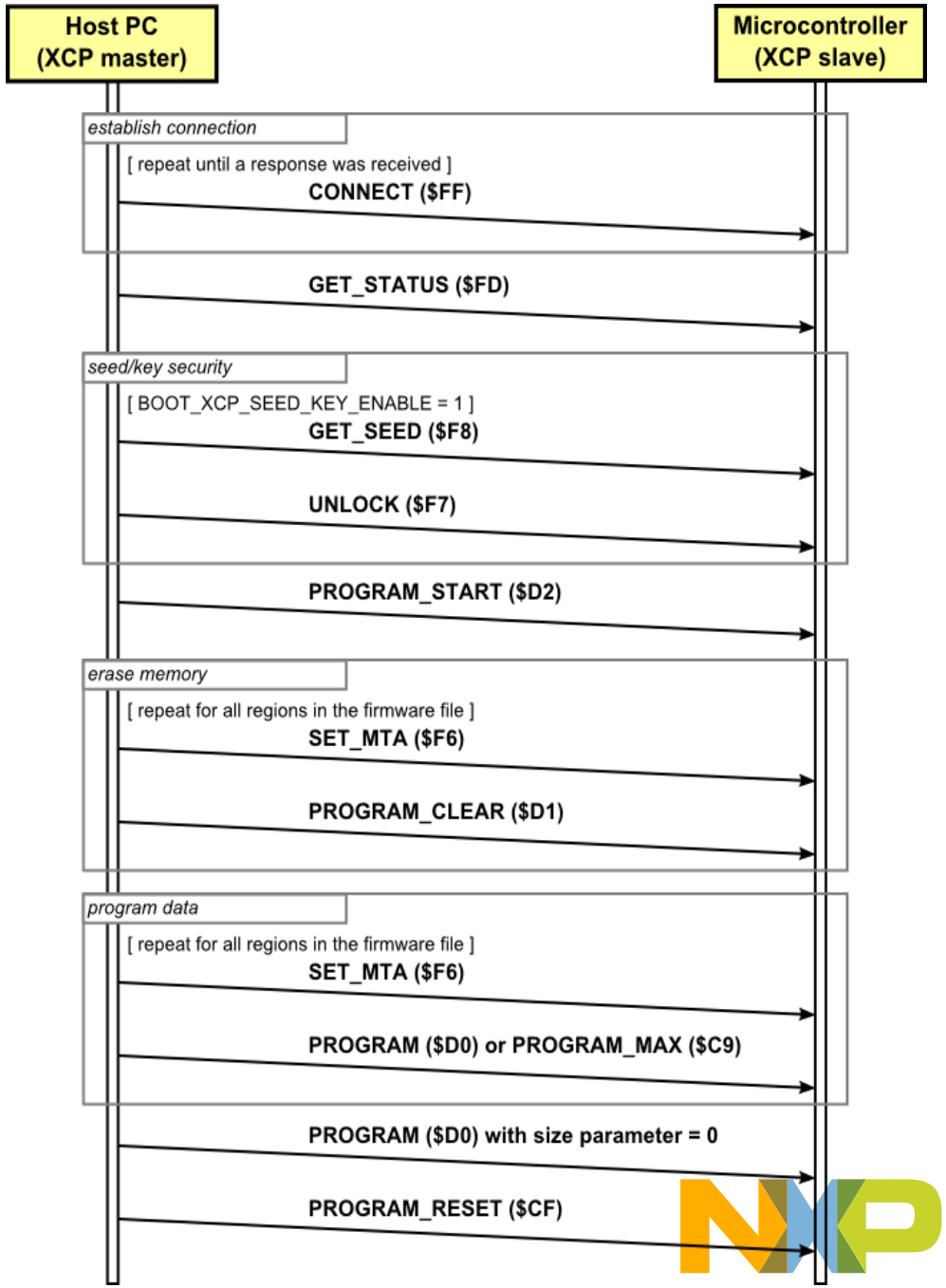
OpenBLT Design Architecture - Cont

1. OpenBLT本身分为4个状态，Entry，Idle，User Program Validation, Programming
2. Entry: 起始状态，将很快切换到其他状态
3. 如果BOOT_BACKDOOR_ENTRY_TIMEOUT_MS时间内有Back door request(比如某个按键按下或者某个RAM变量等—需要用户自行实现)，则进入Idle状态，此状态永久停留等待上位机连接；
4. 如果没有Back door request，则进入应用程序验证启动阶段，如果没有应用程序存在或者程序被损坏，也会进入Idle状态，不允许程序启动等待更新；
5. Idle状态下与上位机连接后，持续擦除编程，完成之后再次进入Entry状态；



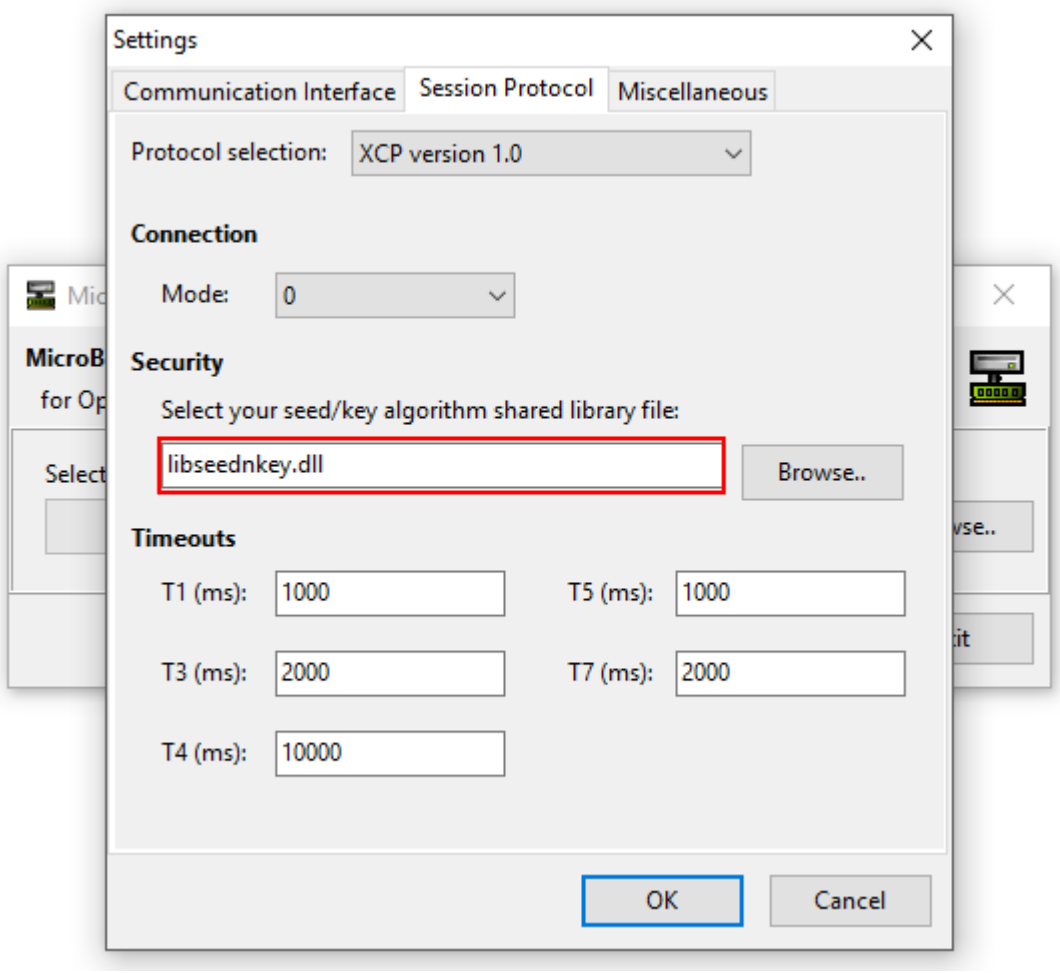
OpenBLT Design Architecture – 协议

XCP 1.0 protocol specification



OpenBLT Design Architecture – 安全

BootCommander -t=xcp_usb -sk=libseednkey.dll
demoprog_olimex_stm32h103.srec



移植步骤 RT1170

OpenBLT - Port Specifics – RT1170 - IAR

1. 在这个示例工程中除了要使用UART升级外，还测试使用ENET升级，所以导入ENET_RXTX的IAR工程作为范本；减少代码移植工作量；

2. 原始文件结构：其中和OpenBLT target工程相关的部分在最下面的Source目录下。分析该目录结构下实现移植OpenBLT的实现不如将相关代码移植到ENET_RXTX的IAR工程。

File Structure

```
+-- Doc/
|   |
|   +-- license.html                          --> GNU GPL v3 license
|   |
|   +-- UserManuals/
|   |   |
|   |   +-- UM_<product name>                 --> User manual for a specific product
|   |
|   +-- ReferenceManuals/
|       |
|       +-- RM_<product name>                 --> Reference manual for a specific product
|
+-- Host/
|   |
|   +-- MicroBoot                             --> Microboot download utility
|   |
|   +-- BootCommander                         --> BootCommander command line download utility
|   |
|   +-- Source/
|       |
|       +-- MicroBoot/                        --> Microboot source code
|       |
|       +-- BootCommander/                    --> BootCommander source code
|       |
|       +-- LibOpenBLT/                       --> OpenBLT host library source code
|       |
|       +-- SeedNKey/                         --> Seed/key unlock algorithm library source cod
|
+-- Target/
    |
    +-- Demo/
    |   |
    |   +-- <cpu_derivative_demoboard_compiler>/ --> e.g. "ARMCM0_STM32F0_Nucleo_F091RC_GCC"
    |       |
    |       +-- Boot/                       --> demo bootloader program, needs to be program
    |       |   |
    |       |   +-- *.h|*.c                 --> source code of the demo bootloader program
    |       |
    |       +-- Prog/                       --> demo use program, can be programmed with Mic
    |           |
    |           +-- *.h|*.c                 --> source code of the demo program
    |
    +-- Source/
        |
        +-- *.h|*.c                         --> target independent bootloader source code
        |
        +-- third_party/                    --> third party libraries such as uIP and FatFS
        |
        +-- <cpu_derivative>/               --> e.g. "ARM7_LPC2000"
            |
            +-- *.h|*.c                     --> target dependent bootloader source code
            |
            +-- <compiler>/                 --> e.g. "GCC"
                |
                +-- *.h|*.c|*.s             --> compiler dependent bootloader source code
```

上位机程序及源码库

演示的Boot例程及工程文件

演示的App例程及工程文件

平台无关的框架程序：OpenBLT组织架构

第三方库：uIP & FatFS

平台相关实现：驱动实现

平台相关实现：compiler相关，禁止、使能中断相关

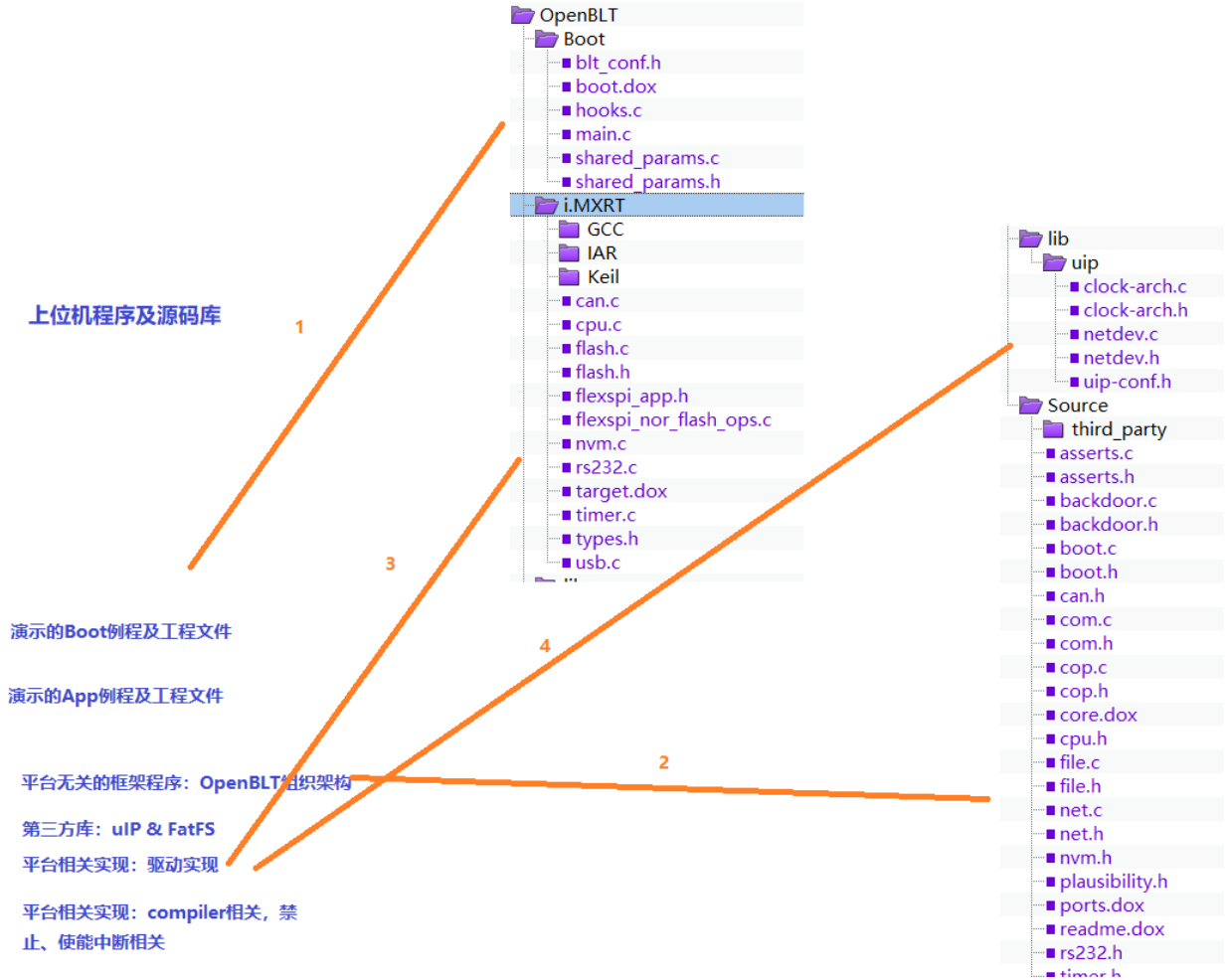
OpenBLT - Port Specifics – RT1170 - IAR

这个文件对应关系在RT1170的工程中都在OpenBLT目录下，并有如下对应：

1. 仅仅增加RT1170对应的启动部分即可，并调整blt_conf.h定义和shared_params.c中关于共享RAM的定义，可以用于和APP交换参数，用于APP引发的bootloader行为；
2. 平台无关的OpenBLT代码，无需改动；

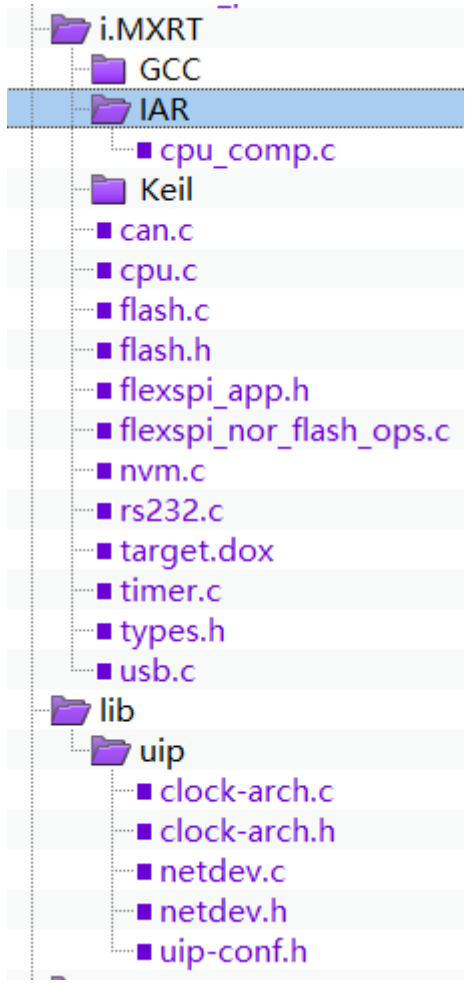
File Structure

```
+-- Doc/
|   |
|   +-- license.html                        --> GNU GPL v3 license
|   |
|   +-- UserManuals/
|   |   |
|   |   +-- UM_<product name>               --> User manual for a specific product
|   |
|   +-- ReferenceManuals/
|       |
|       +-- RM_<product name>               --> Reference manual for a specific product
|
+-- Host/
|   |
|   +-- MicroBoot                           --> Microboot download utility
|   |
|   +-- BootCommander                       --> BootCommander command line download utility
|   |
|   +-- Source/
|       |
|       +-- MicroBoot/                      --> Microboot source code
|       |
|       +-- BootCommander/                  --> BootCommander source code
|       |
|       +-- LibOpenBLT/                     --> OpenBLT host library source code
|       |
|       +-- SeedNKey/                       --> Seed/key unlock algorithm library source cod
|
+-- Target/
    |
    +-- Demo/
    |   |
    |   +-- <cpu_derivative_demoboard_compiler>/ --> e.g. "ARMCM0_STM32F0_Nucleo_F091RC_GCC"
    |       |
    |       +-- Boot/                       --> demo bootloader program, needs to be program
    |       |   |
    |       |   +-- *.h|*.c                --> source code of the demo bootloader program
    |       |
    |       +-- Prog/                       --> demo use program, can be programmed with Mic
    |           |
    |           +-- *.h|*.c                --> source code of the demo program
    |
    +-- Source/
        |
        +-- *.h|*.c                        --> target independent bootloader source code
        |
        +-- third_party/                    --> third party libraries such as uIP and FatFS
        |
        +-- <cpu_derivative>/               --> e.g. "ARM7_LPC2000"
            |
            +-- *.h|*.c                    --> target dependent bootloader source code
            |
            +-- <compiler>/                 --> e.g. "GCC"
                |
                +-- *.h|*.c|*.s             --> compiler dependent bootloader source code
```



OpenBLT - Port Specifics – RT1170 - IAR

3. 平台相关部分驱动，包括Flash驱动，UART驱动和Timer驱动，CAN驱动，USB驱动等等
4. 包括了和uIP对应的以太网驱动实现及其需要的timer实现



OpenBLT - Port Specifics – RT1170 – IAR – 细节注意事项 - Memory

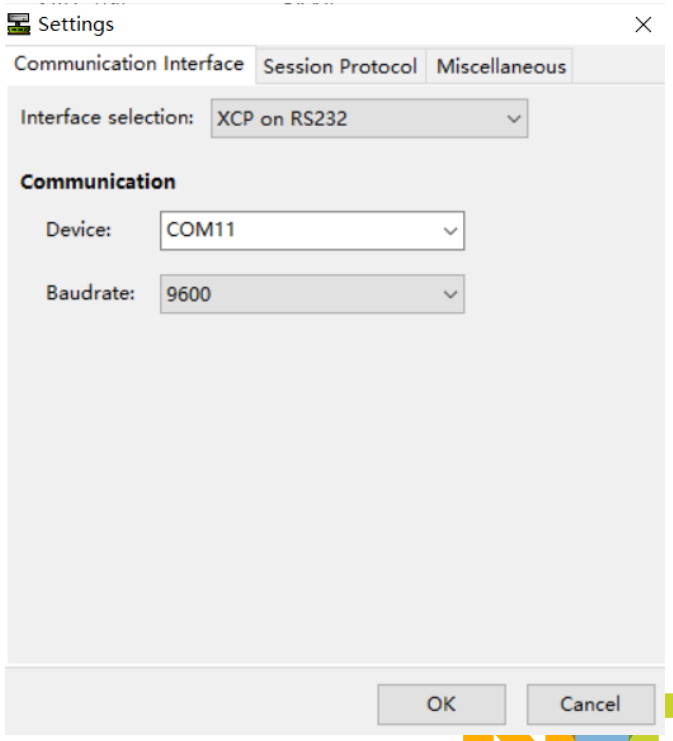
为bootloader保留64KB的空间，后续空间留给APP

在Link文件中保留ITCM中的0x40字节作为share memory。如果不用可以删除，此处仅仅是为了保留OpenBLT的设计风格。

```
define symbol __openblt_shared_size = 0x40;

define symbol __ICFEDIT_region_SHARED_start__  = isdefinedsymbol(__ram_vector_table__) ? (m_interrupts_ram_end + 1) :
(m_interrupts_ram_end);

define symbol __ICFEDIT_region_SHARED_end__    = __ICFEDIT_region_SHARED_start__ + __openblt_shared_size - 1;

place at address mem:__ICFEDIT_region_SHARED_start__ { readwrite section .shared };
```

OpenBLT - Port Specifics – RT1170 – IAR – 细节注意事项

- CPU.c 修改和程序跳转启动相关的代码CpuStartUserProgram，主要是定义APP起始位置，由于RT的APP Image默认都带有IVT、FCB等信息，在APP image build时注意（1）去除非APP必要部分，XIP_BOOT_HEADER_ENABLE=0 （2）可以保留shared memory部分
- CPU_comp.c IDE编译器的禁止使能中断代码，无需改动
- CAN.c，示例未用



OpenBLT - Port Specifics – RT1170 – IAR – 细节注意事项

- Flash.c flash size的配置，擦除编程驱动函数等，其中有几个细节设置需要注意
  FLASH_WRITE_BLOCK_SIZE：一次性写入的Block size，OpenBLT flash框架使用存在于RAM中的Block缓存，等存完一个最小Block再一次性写入。对于QSPI Flash为256字节
  BOOT_FLASH_VECTOR_TABLE_CS_OFFSET (0x1C)//checksum offset of application, put in vectable 0x1c, the checksum is from 0x00 - 0x18 一般选择APP中断向量表的某个偏移位置。当然用其他特殊的存储方式也可以实现，比如EEPROM或者Flash的独立位置；
  BOOT_FLASH_CUSTOM_LAYOUT_ENABLE=1及flashLayout：Flash layout用于规划Flash的Sector的分布，sector的大小及数量。由于Qspi Flash的size一般比较大，所以不采用静态分配的方式，而是用代码自动分配。Sector代表一次性擦除的最小单元区域。
  APP_FLASH_SECTOR_START_SECTOR： APP的起始sector地址，64KB（Sector No 16）

- RS232.c: 串口驱动读写都是采用polling方式，而不是中断或者DMA，fsl_lpuart没有合适的non blocking函数接口能够直接使用，使用LPUART_ReadBlocking加入UART_READ_RETRY_TIMES=1的超时机制能够在blocking函数接口中返回收到数据与否。这个定义直接定义在IDE的preprossor中。为了不影响默认的UART_RETRY_TIMES影响uart write，故特意在RT1170的SDK中创建这个宏。

- Timer.c:使用systick创建1ms定时器使用

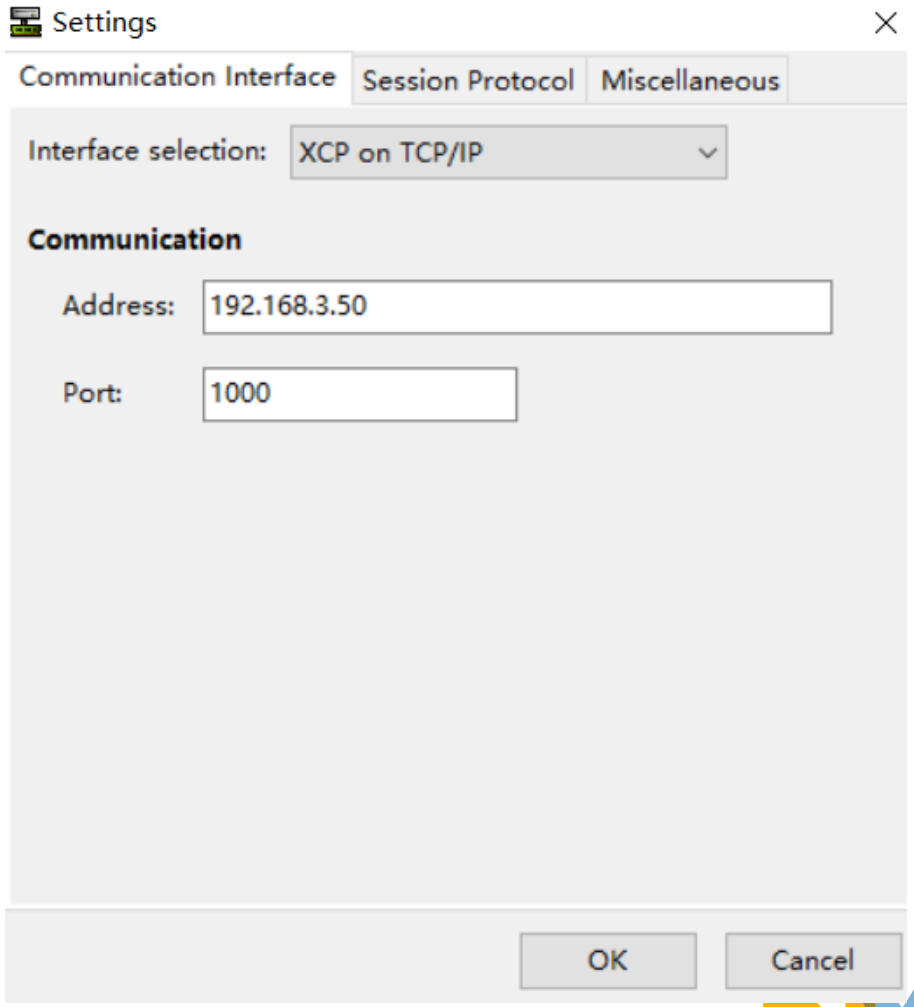
- 最后在main.c中实现Init和Deinit函数，这个可以参照SDK原始例程

以上为使用最基本的串口实现升级的最少修改。

OpenBLT - Port Specifics – RT1170 – IAR – 细节注意事项

- NetDev.c 是实现以太网升级的关键移植。OpenBLT使用uIP协议栈来实现以太网升级协议，同样也是使用polling而非中断DMA，此处根据范例使用ENET_txrx例程函数接口实现即可。

- 另外uIP也支持DHCP自动获取IP，这个用户可以自行测试



OpenBLT - Port Specifics – RT1170 – IAR – 细节注意事项

其他特殊功能

- Backdoor功能： 提供一个功能，能让BOOT_BACKDOOR_ENTRY_TIMEOUT_MS内停留在bootloader中，超时之后才开始启动程序；

- Hooks.c： 提供了很多关键逻辑的回调接口，可以log用户调试信息或者增加特定的功能
- CpuUserProgramStartHook：在bootloader启动APP之前可以检测某个IO是否被下拉，以确定是否停留在bootloader中

- CopInitHook、 CopServiceHook：自行添加看门狗服务

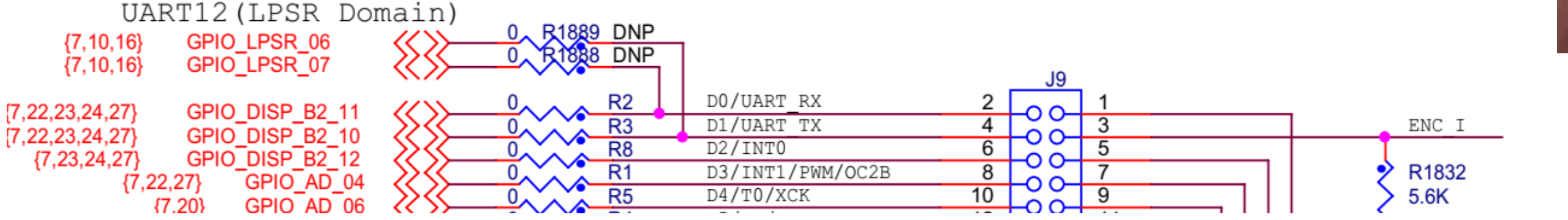
- BOOT_BACKDOOR_HOOKS_ENABLE：默认是0，如果使能， BackDoorInitHook、 BackDoorEntryHook可以用于检测是否应该进入APP还是停留在bootloader

- Blt_conf.h
  BOOT_COM_NET_ENABLE 设置使能以太网，并设置IP地址等
  BOOT_COM_RS232_ENABLE 设置波特率和LPUART channel

OpenBLT - Port Specifics – RT1170 – Demo 串口升级

RT1170 demo使能了串口及以太网升级两种方式。
连接以太网，串口，CMSIS-DAP仿真器：可以调试及通过串口打印log

1. 串口升级，使用LPUART2, 在J9上面的pin 2&4作为串口升级的串口。
2. 下载RT1170_OpenBLT到RT1170 EVK
3. 打开EVK的DAP串口，波特率115200，可以查看log
4. EVK最好采用外部供电，以防PC USB Port供电能力不足，出现一些错误；
5. 上位机打开MicroBoot.exe，串口选择USB转串口的port，选择波特率115200；
6. 准备i.MXRT1170的app文件，主要修改link文件，从0x10000开始运行，并转换输出为.srec文件。
7. igpio_led_output_cm7.srec、freertos_hello_cm7.srec是预先准备好的app文件
8. 在MicroBoot.exe选择该文件，重启板子即可升级；并且在CMSIS-DAP的虚拟串口中查看log。
9. 也可以按住SW7，复位板子，可以使MCU留在Bootloader中而不运行app。



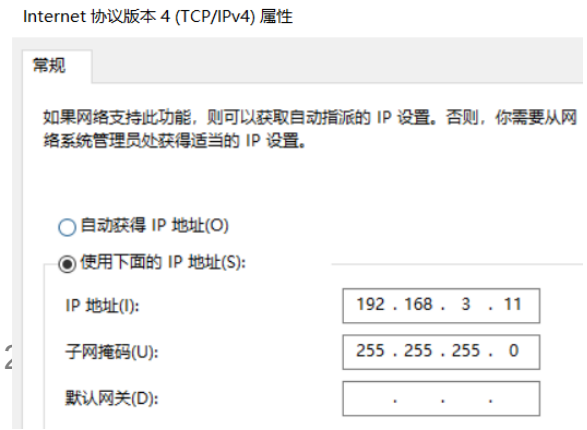
OpenBLT - Port Specifics – RT1170 – Demo 以太网升级

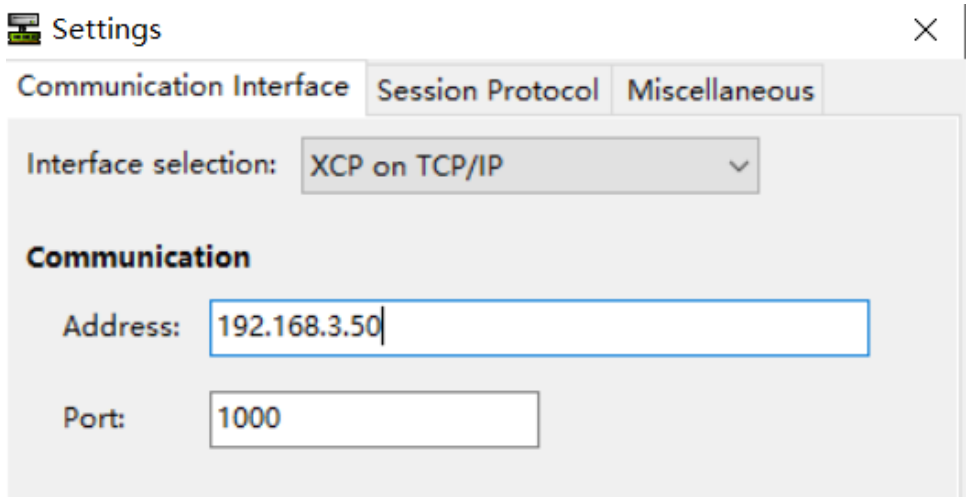
RT1170 demo使能了串口及以太网升级两种方式。
连接以太网，串口，CMSIS-DAP仿真器：可以调试及通过串口打印log

1. 使用EVK J3 RJ45口作为以太网升级口，与PC以太网口直连；PC IP地址配置为192.168.3网段，子网掩码255.255.255.0，默认EVK会被配置为192.168.3.50， Port为1000，
2. 下载RT1170_OpenBLT到RT1170 EVK
3. 打开EVK的DAP串口，波特率115200，可以查看log
4. EVK最好采用外部供电，以防PC USB port供电能力不足，出现一些错误；
5. 上位机打开MicroBoot.exe，通信接口选择以太网，配置板子IP地址为192.168.3.50， Port 1000；
6. 准备i.MXRT1170的app文件，主要修改link文件，从0x10000开始运行，并转换输出为.srec文件。
7. igpio_led_output_cm7.srec、freertos_hello_cm7.srec是预先准备好的app文件
8. 要想使能OpenBLT的以太网升级需要满足如下条件：初始化以太网及uIP协议栈属于较慢的操作，OpenBLT启动时并不会启动以太网升级检测，只有：

- 没有APP Image固件存在或者校验失败会触发以太网的后初始化；
- 通过特殊按键触发，比如复位时按住SW7，可以在启动时触发完全下载模式，而不运行APP，这时候也可以连接以太网进行下载；

9. 在MicroBoot.exe选择该文件，即会自动触发下载；





移植步骤 LPC5535

OpenBLT - Port Specifics – LPC5536 – Keil – 细节注意事项

- CPU.c 修改和程序跳转启动相关的代码CpuStartUserProgram，主要是定义APP起始位置，例程定义为0x8000(32KB）起始位置
- CPU_comp.c IDE编译器的禁止使能中断代码，无需改动

- CAN.c：使用MCAN的loop back例程，需要修改其默认的CANFD 模式，及external loopback mode。默认为中断模式，同样需要修改为polling模式

- RS232.c: 串口驱动读写都是采用polling方式，而不是中断或者DMA，fsl_usart没有合适的non blocking函数接口能够直接使用，使用ReadBlocking加入UART_READ_RETRY_TIMES=1的超时机制能够在blocking函数接口中返回收到数据与否。这个定义直接定义在IDE的preprossor中。为了不影响默认的UART_RETRY_TIMES影响uart write，故特意新创建这个宏。

- Timer.c:使用systick创建1ms定时器使用

OpenBLT - Port Specifics – LPC5536 – Keil – 细节注意事项

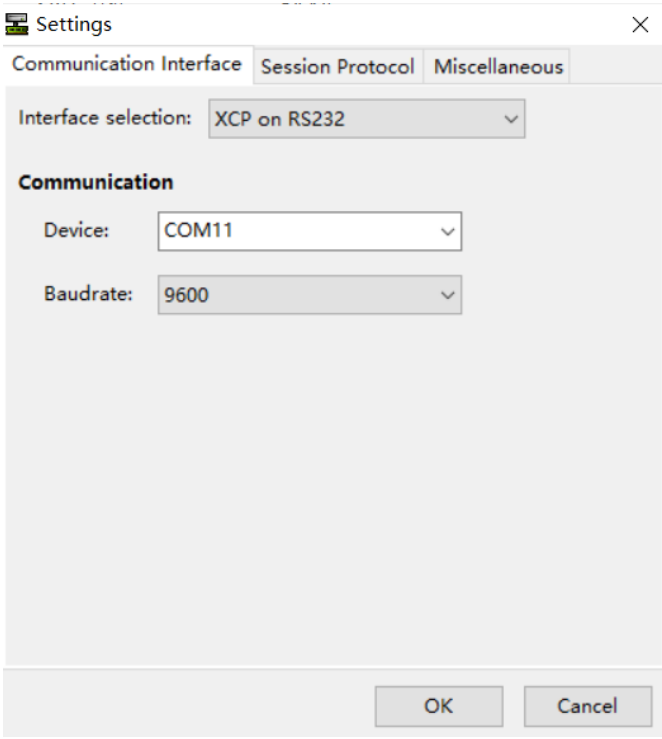
- Flash.c flash size的配置，擦除编程驱动函数等，其中有几个细节设置需要注意

FLASH_WRITE_BLOCK_SIZE：一次性写入的Block size，OpenBLT flash框架使用存在于RAM中的Block缓存，等存完一个最小Block再一次性写入。LPC553X为512字节。因为LPC553X支持的最小擦除和写入单元都是512字节的page，所以将page size和block size都设定为512Bytes

BOOT_FLASH_VECTOR_TABLE_CS_OFFSET (0x1C)//checksum offset of application, put in vectable 0x1c, the checksum is from 0x00 - 0x18 一般选择APP中断向量表的某个偏移位置。当然用其他特殊的存储方式也可以实现，比如EEPROM或者Flash的独立位置；

BOOT_FLASH_CUSTOM_LAYOUT_ENABLE=1及flashLayout：Flash layout用于规划Flash的Sector的分布，sector的大小及数量Sector代表一次性擦除的最小单元区域。OpenBLT支持客制化这个FlashLayout表，这边使用代码自动分配。

APP_FLASH_PAGE_START_PAGE： APP的起始page地址，32KB（page No 64）



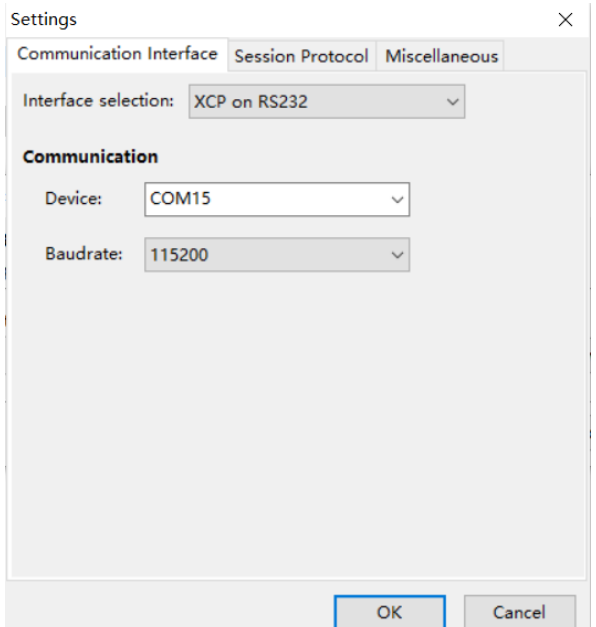
LPC55系列的Flash驱动还有几个重要的注意事项

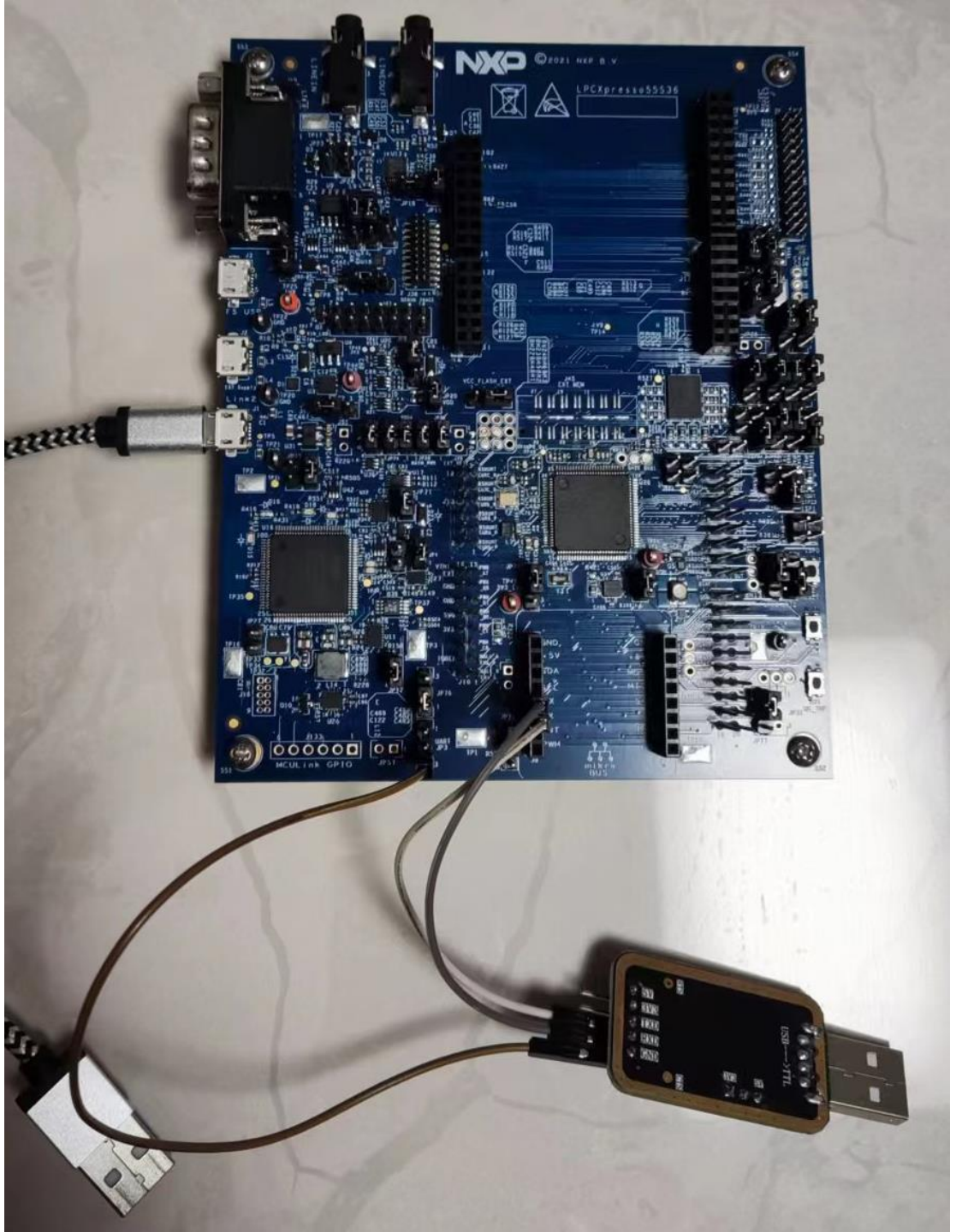
- 这里对于Sector index或者number都可能超过255，所以和这个相关的变量定义从原本的blt_int8u到blt_int32u
- Flash的最小擦除和写入最小单元是512Bytes，需要动态分配Flash layout
- Flash末尾的10KB包含配置页，需要避开使用
- Flash未写不能读，所以做APP Image checksum的时候，不能直接读APP地址区域，而是要先验证该区域是否是已擦除的，如果是就不读取该区域；
- FlashInitBlock的时候会读取Flash还有内容，经修改后再写回，与上面原因相同，不能直接读取，而应该先检查该区域是否是擦除状态；如果是采用全FF填充回读buffer。

最后在main.c中实现Init和Deinit函数，这个可以参照SDK原始例程

OpenBLT - Port Specifics – LPC5536 – Demo串口升级

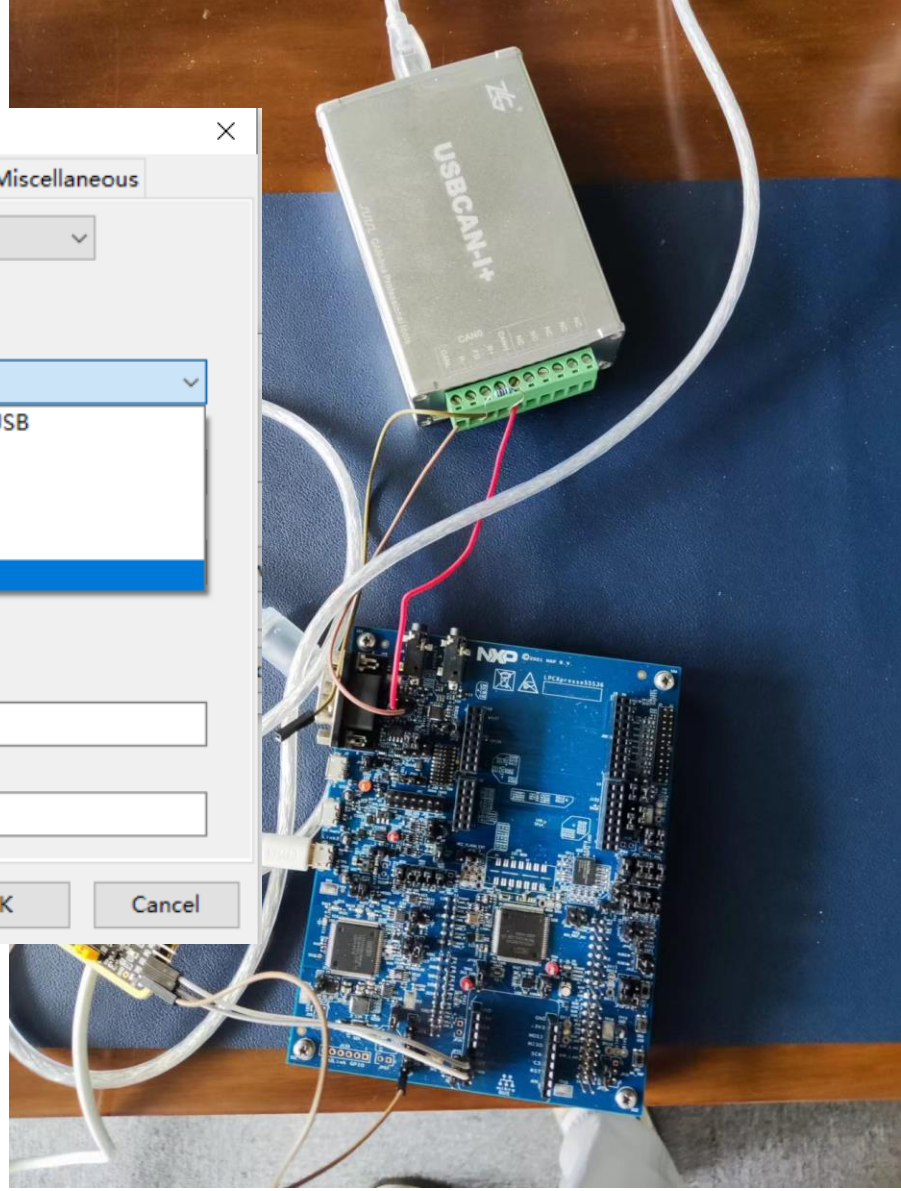
1. 串口升级，使用FC2, 在J8上面的pin 3&4作为串口升级的串口。JP51改连接 1 & 2，将FC2_TX引出
2. 上位机打开MicroBoot.exe，串口选择USB转串口的号，选择波特率115200；
3. 准备LPC5536的app文件，主要修改link文件，从0x8000开始运行，并转换输出为.srec文件。
4. fromelf.exe --m32combined --output "$L@L.srec" "#L"
5. LPC5536_app_gpio_led_output.srec、emwin_gui_demo_LPC5536.srec是预先准备好的app文件
6. 在MicroBoot.exe选择该文件，重启板子即可升级；并且在CMSIS-DAP的虚拟串口中查看log。
7. 也可以按住SW1，复位板子，可以使MCU留在Bootloader中而不运行app。

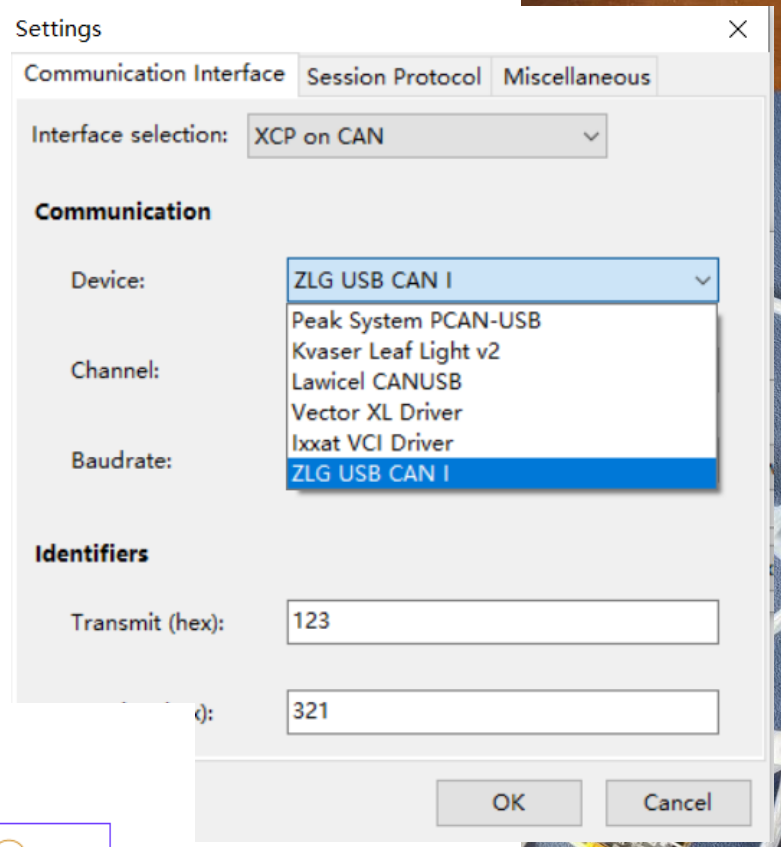


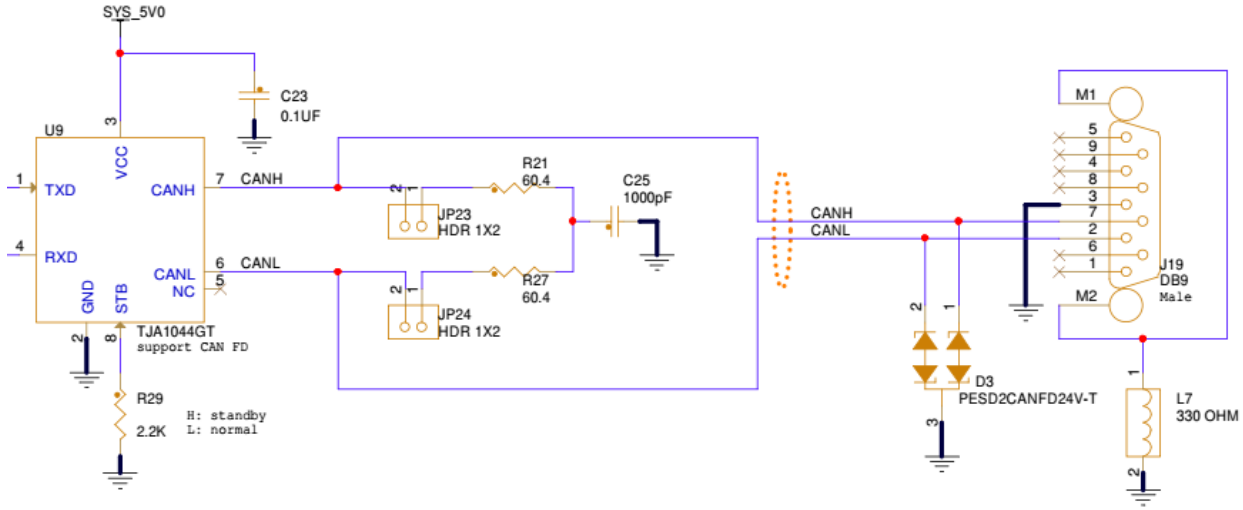


OpenBLT - Port Specifics – LPC5536 – Demo CAN 升级

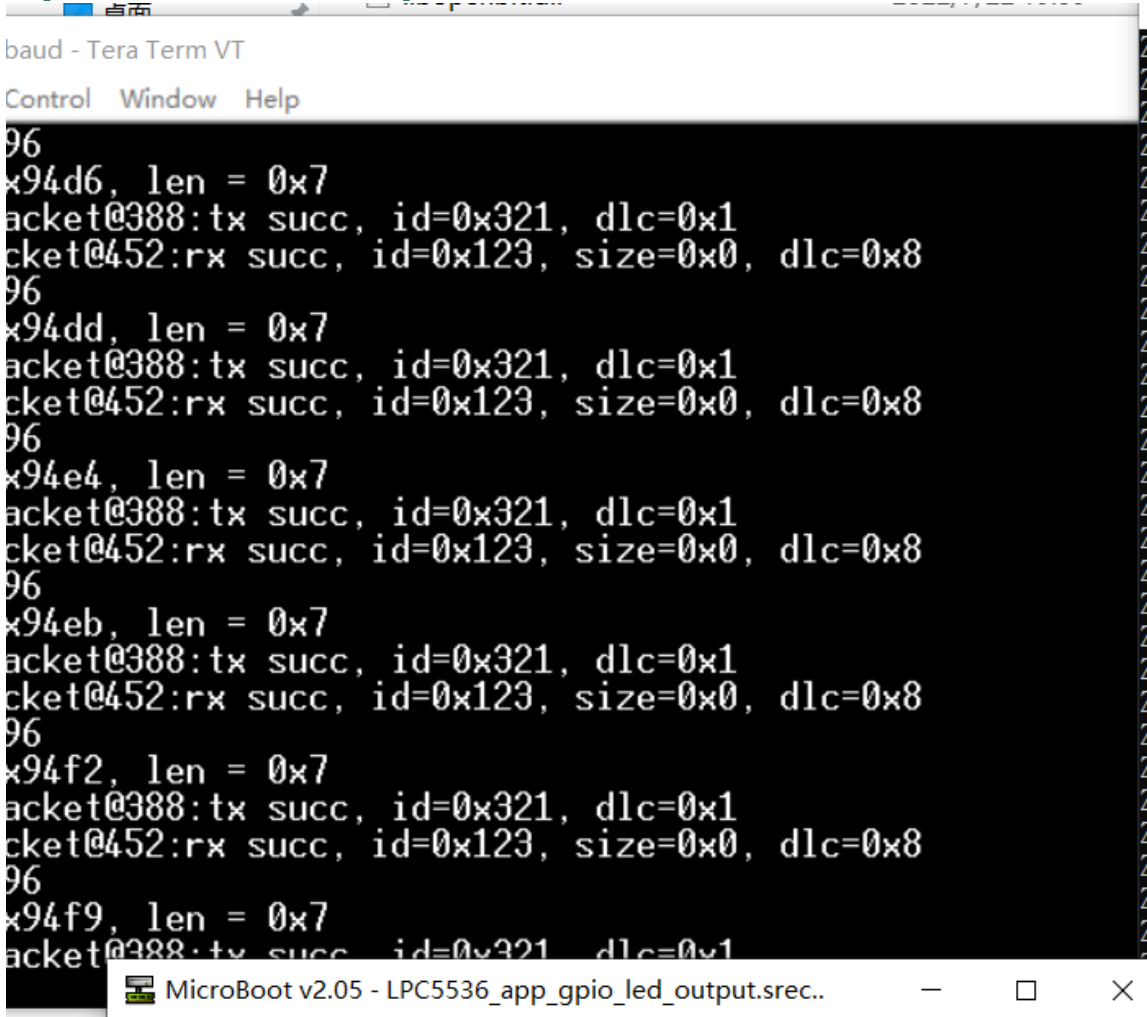
1. CAN升级，将CAN盒接到JP23/24的CANH和CANL
2. 上位机打开MicroBoot.exe，选择CAN盒列表中的对应型号（新增CAN盒的说明在下一章节）；发送CAN ID设置为0x123，接收CAN ID 0x321；波特率500kBits/s。这些固定参数都是OpenBLT里面固定的。
3. 准备LPC5536的app文件，主要修改link文件，从0x8000开始运行，并转换输出为.srec文件。
4. fromelf.exe --m32combined --output "$L@L.srec" "#L"
5. LPC5536_app_gpio_led_output.srec、emwin_gui_demo_LPC5536.srec是预先准备好的app文件
6. 在MicroBoot.exe选择该文件，重启板子即可升级；并且在CMSIS-DAP的虚拟串口中查看log。
7. 也可以按住SW1，复位板子，可以使MCU留在Bootloader中而不运行app。

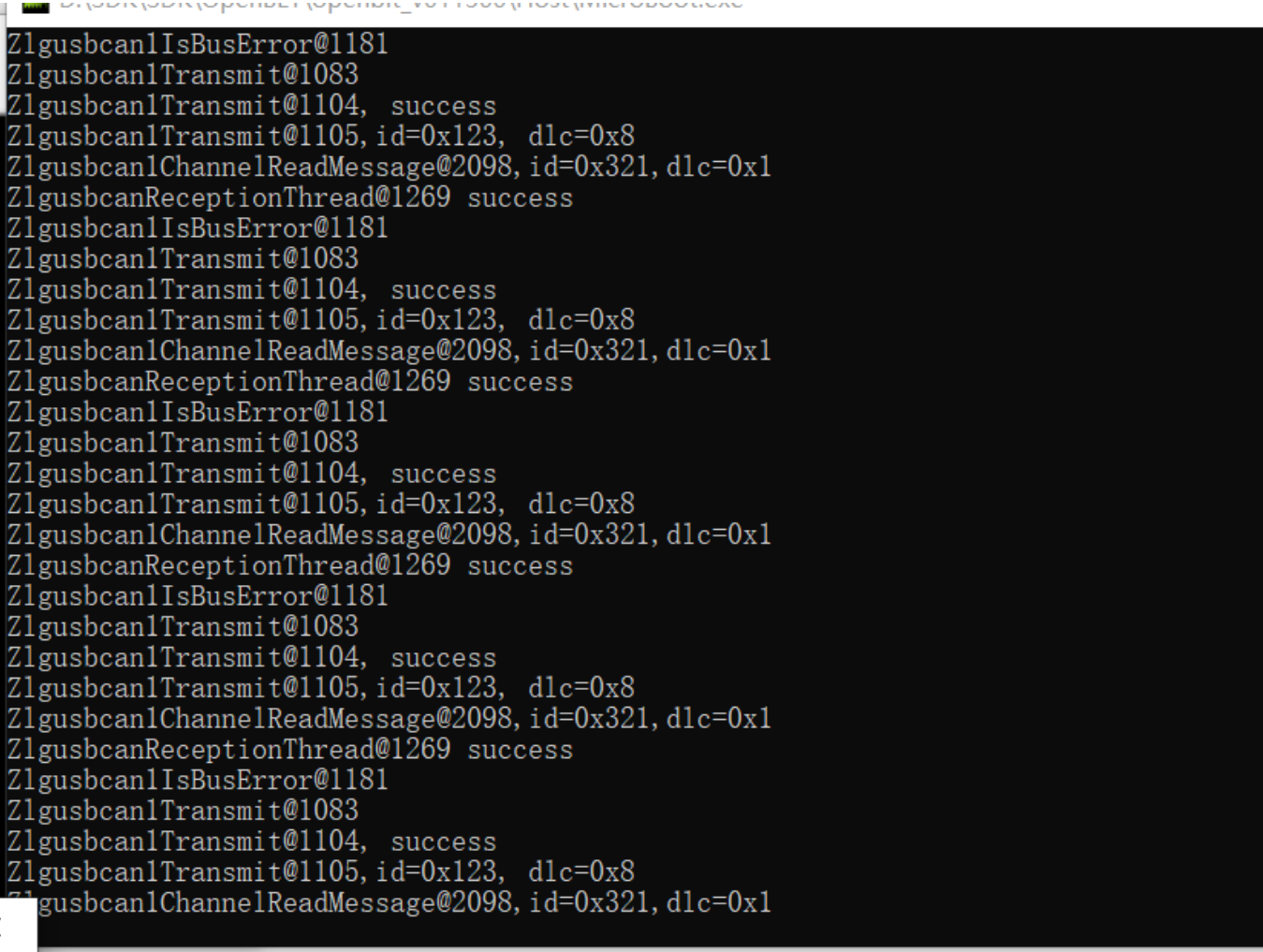


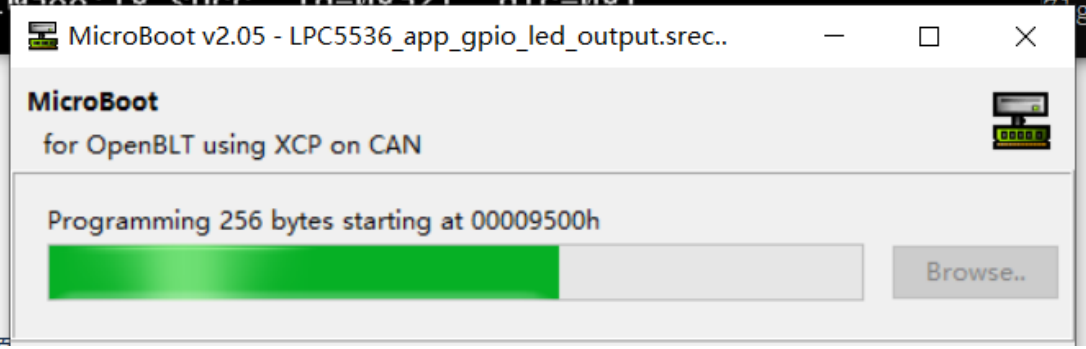




OpenBLT - Port Specifics – LPC5536 – Demo CAN 升级







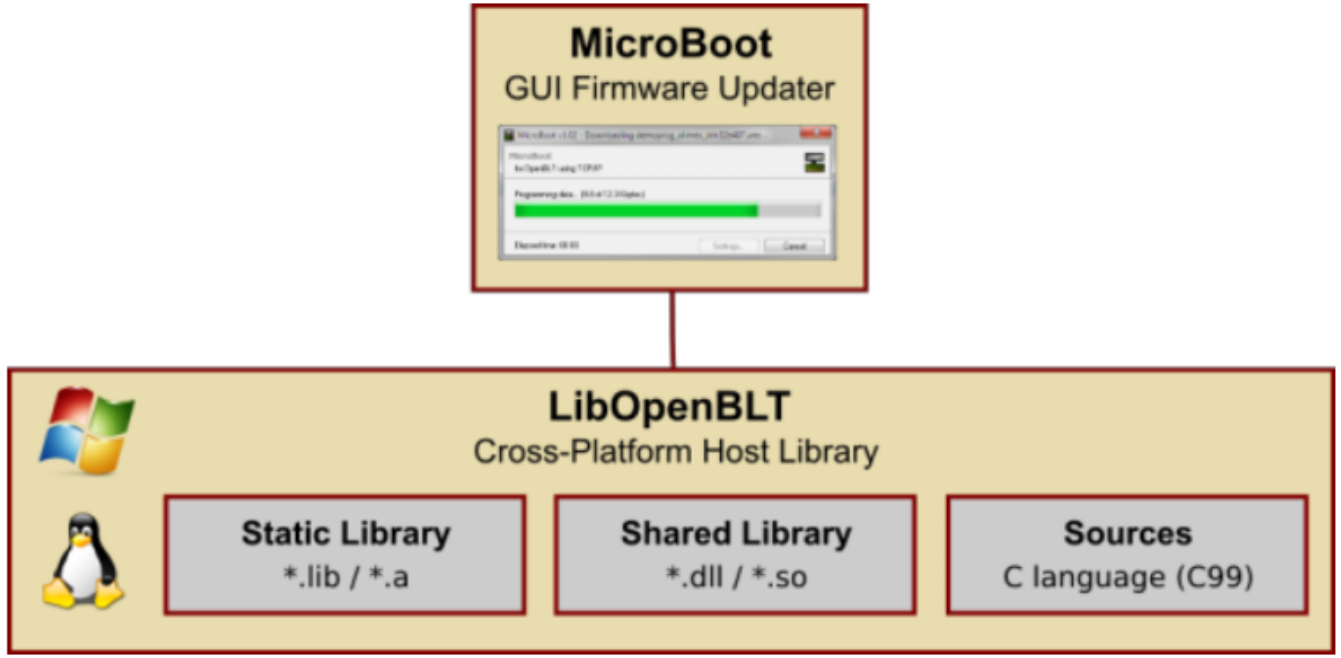
如何修改
LIBOPENBLT.DLL
MICROBOOT.EXE

为ZLG USB CAN增加接口

- 默认的CAN接口设备并不支持ZLG USB CAN I+
- 测试这个CAN盒需要在MicroBoot中新增对这类CAN盒的支持

为ZLG USB CAN增加接口

- MicroBoot只是负责界面操作；
- 实际的协议及底层设备操作封装在LibOpenBLT的dll库中，这个库是可以在不同平台之间迁移的。
- 在Windows上使用VS工具重新编译即可
- 对于MicroBoot上位机使用Lazarus IDE进行编译，这里注
  64bit版本，我这边的AMD64位PC只能选择32bit版本编译



为ZLG USB CAN增加接口

LibOpenBLT先要在build目录下创建VS编译工程环境

- Cmake ..
- cmake -A Win32 ..

Support for Visual Studio 2019
The majority of the included PC tools (the BootCommander program, the LibOpenBLT DLL and the seed/key DLL) can be built with Microsoft Visual Studio.

Visual Studio 2019 by default uses a 64-bit compiler, which theoretically works for the PC tools. However, LibOpenBLT depends on a few third party 32-bit DLLs for which no 64-bit counterparts are (yet) available. Therefore, it is important to invoke cmake with the -A Win32 flag to make sure the 32-bit compiler is used:

实测在ADM64位电脑上编译Cmake生成的64bit dll，无法配合microboot一起使用。必须使用W32的配置。

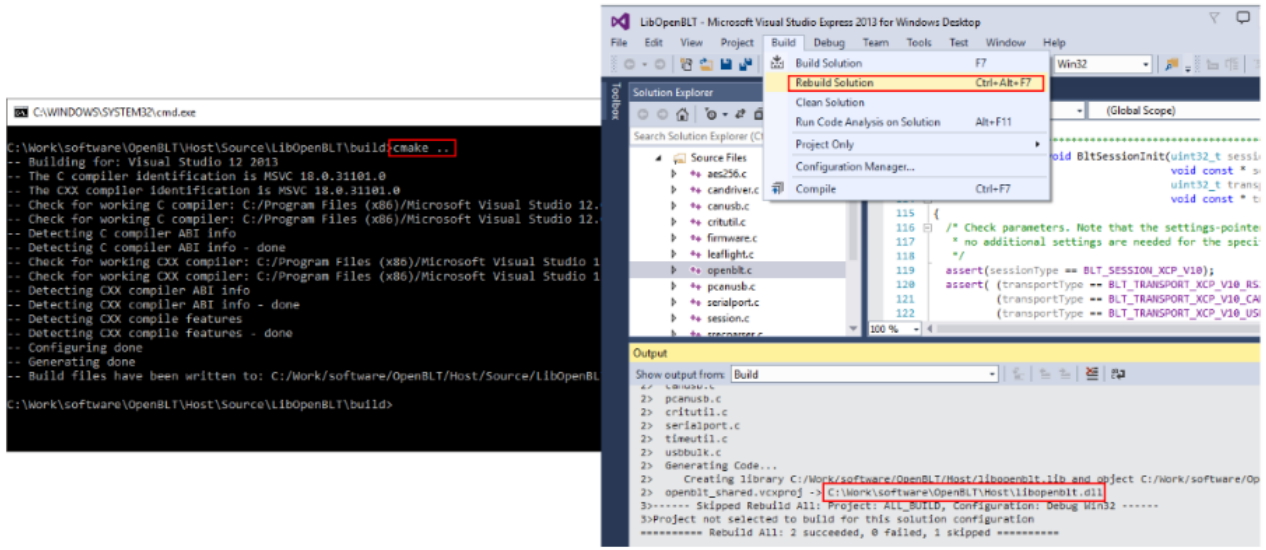
Building on Windows with Microsoft Visual C++

This method assumes that Microsoft Visual C++ is installed. During the LibOpenBLT development, Microsoft Visual Studio 2019 was used. The method outlined here should work just fine with both older and newer versions of Microsoft Visual Studio.

Using the Command Prompt in Windows, set the working directory to **.\Host\Source\LibOpenBLT\build** and type the command:

```
cmake ..
```

A solution file for Microsoft Visual C++ is then automatically generated, called LibOpenBLT.sln. Open the solution in Microsoft Visual C++ and build the program from there, by selecting Build→Rebuild Solution from the menu.

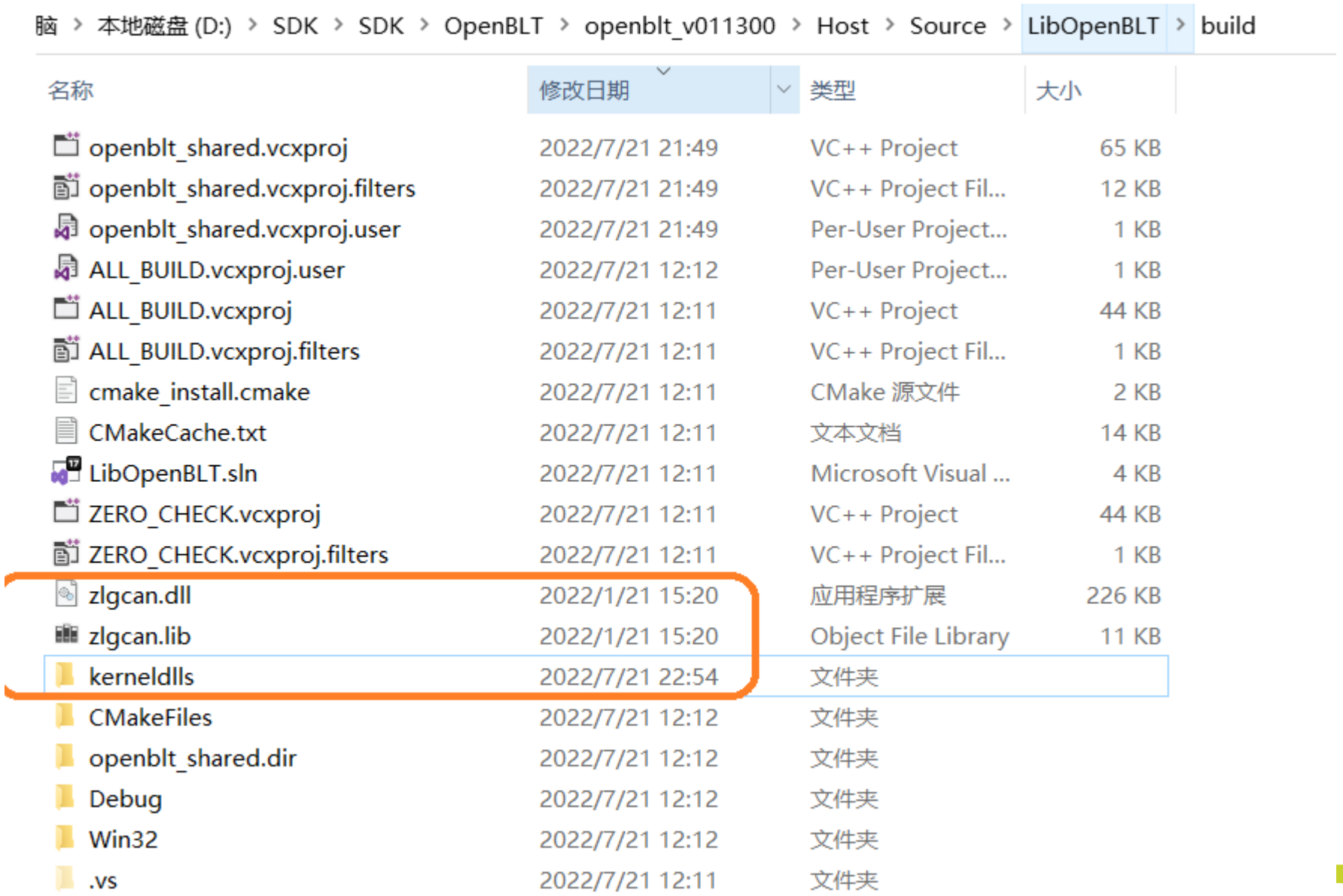


After a successful build, the shared library **libopenblt.dll** is located in the **./Host** directory.

Note that if you want to build a 32-bit version of LibOpenBLT, generate the build environment with command "**cmake -A Win32 ..**" instead. This is recommended if you use OpenBLT from before version 1.14.

为ZLG USB CAN增加接口

- 将ZLG C++的库添加到build目录下。
- MicroBoot.exe同目录下也需要放上这些动态链接库



为ZLG USB CAN增加接口

- 在canif目录下增加zlgusbcan的驱动接口
- 仿照已经存在的CANIF接口实现ZLG USB CAN的接口函数
- 并在在Lazarus工程中为CAN盒列表增加ZLG CAN盒的定义；

```
static const tCanInterface Zlgusbcan1Interface =
{
  Zlgusbcan1Init,
  Zlgusbcan1Terminate,
  Zlgusbcan1Connect,
  Zlgusbcan1Disconnect,
  Zlgusbcan1Transmit,
  Zlgusbcan1IsBusError,
  Zlgusbcan1RegisterEvents
};
```

SECURE CONNECTIONS
FOR A SMARTER WORLD