Unified bootloader

Tomlin Tang/ 唐林

Automotive Microcontrollers & Processors (AMP)

Dec 2019

SECURE CONNECTIONS
FOR A SMARTER WORLD

Agenda

- Objective
- Bootloader practice at CAR OEMs and Tier1s
- Scope of GC AMP Bootloader
- How achieve
- Key features
- Bootloader architecture
- How to porting bootloader stack to new platform
- Bootloader used flash size and performance
- FOTA/OTA support by bootloader
- How to use UI/host

Objective

- Efficiency to implement bootloader over a new part by reusing existing stack
- Deliver a high quality new bootloader instance by reusing proven platform
- Professional / production level bootloader solution deliver to customer

Bootloader practice at CAR OEMs and Tier1s

- Trends: All ECU will apply Bootloader for SW update
- Less than 3% China CAR OEM purchases Bootloader stacks from Auto SW supplier (e.g., Vector)
- Many China CAR OEMs / Tier1s outsource Bootloader together with diagnosis
- Normally implement bootloader at very early stage of project developing
- UDS* & TP** protocol is widely used

* UDS(ISO14229): Unified diagnostic services
** TP(ISO15765-2/ISO17987-2): Transport protocol and network layer services
Note: UDS and TP is used by HIS (Herstel Ierinitiative Software, defined the bootloader data and control flow)

Scope of GC AMP Bootloader

PC
- UI
- UDS
- TP
- DLL(provide by vendor)

ECU
- UDS
- TP
- Common requirements
- Common driver (CAN/LIN)

Common driver
- CAN
- CAN-FD
- LIN
- Ethernet
- FlexRay
- …

Scope of standard
- Host UI
- UDS/TP (ECU)
- Common requirements
  I. Flash driver download from host
  II. App validity check in bootloader
  III. OBJ file (S19/BIN/HEX) parsing in host
  IV. APP and bootloader control signal reside in RAM
  V. Using reset (watchdog) switch to APP or bootloader

How achieve

- Code review
- Code rule & standard
- Stress test

Key features - bootloader

- UDS (ISO14229)
- TP (ISO15765-2/CAN, ISO17987-2/LIN)
- Common requirements
  - Flash driver download from host
  - App validity check in bootloader
  - APP and bootloader control signal reside in RAM
  - Using reset (watchdog) switch to APP or bootloader
  - CRC
  - AES
  - Random
  - Hardware driver HAL

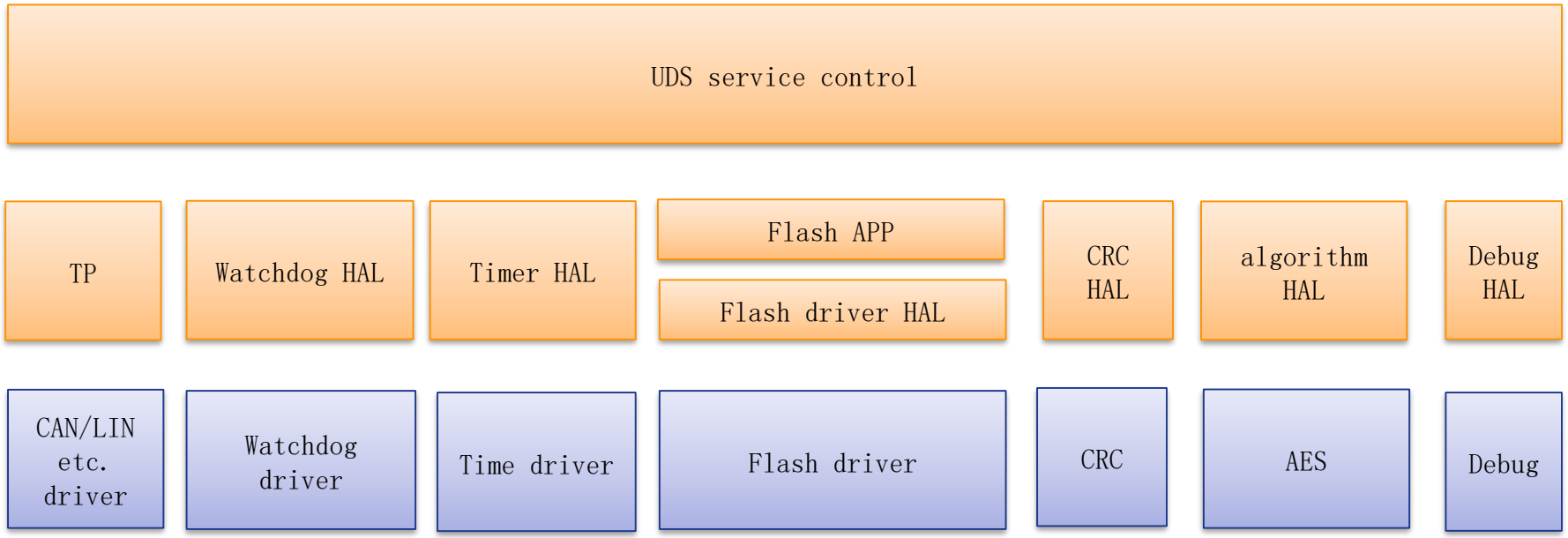
Key features – UI/host

- Common requirements
  - UDS* (ISO14229)
  - TP* (ISO1765-2/CAN, ISO17987-2/LIN)
  - Support JSON control firmware download process
  - Support import/export JSON
  - Support config JSON
  - Support print logging
  - Support parsing OBJ (S19/BIN/HEX)
  - Support select Flash driver and APP OBJ
  - Support config function and physical, respond ID for CAN. LIN can be config NAD ID
  - Support CRC and AES. Both can be changed by customer
  - Support display progress bar for download firmware
  - Status about download firmware
  - Support CAN & LIN for update firmware to ECU

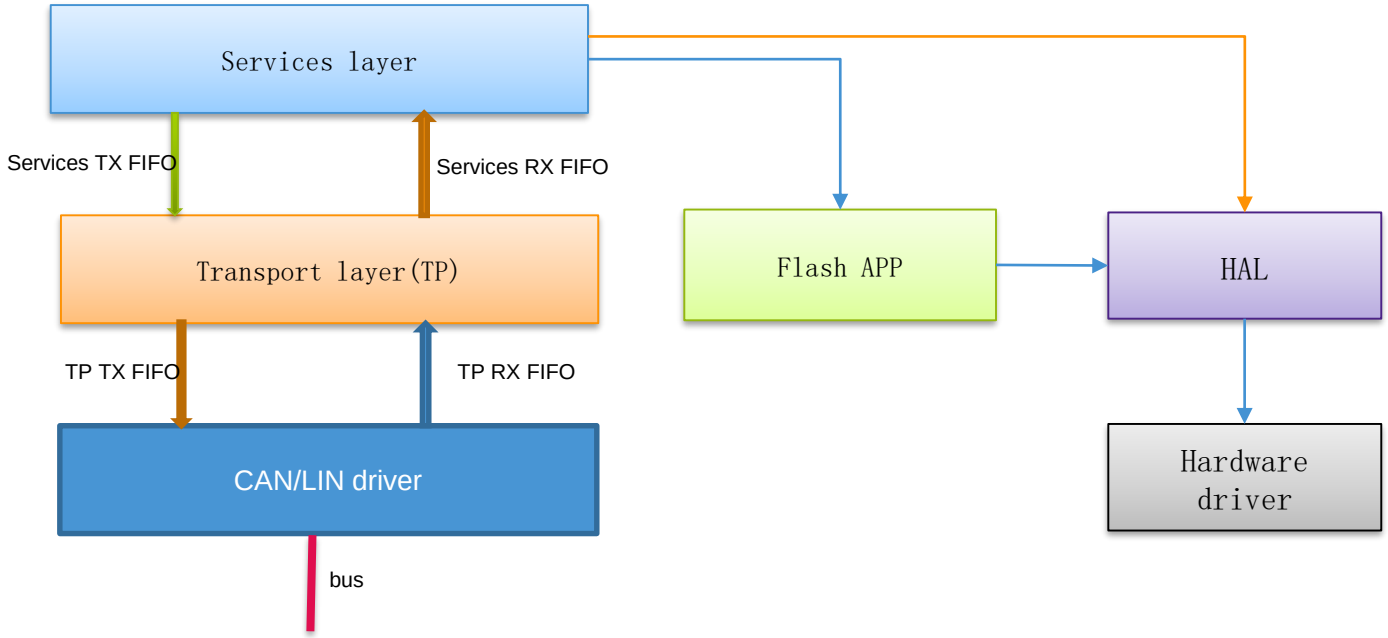
UDS and TP: UDS and TP provided by PEAK based on CAN.

UDS is developed by NXP and TP provide by PEAK based on LIN.

Bootloader system diagram



Bootloader architecture



Bootloader architecture

Bootloader have 6 modules, include services layer, transport layer, FIFO, hardware communicate driver (CAN/LIN etc.), flash APP and HAL.

- Services layer
  - Received message from TP layer (services RX FIFO)
  - Control bootloader process. Example operate flash init, erase, program etc.
  - Response message to host(transmit message to services TX FIFO)

- TP
  - Received message from TP RX FIFO (CAN/LIN etc.)
  - Analysis the data from TP RX FIFO. If the message valid and send a frame to service RX FIFO
  - Transmit message to TP TX FIFO, if have message from services layer.
  - TP only transmit a frame message to service RX FIFO and delete header and checksum.
  - TP will add header, checksum and transmit a frame message to TP TX FIFO.

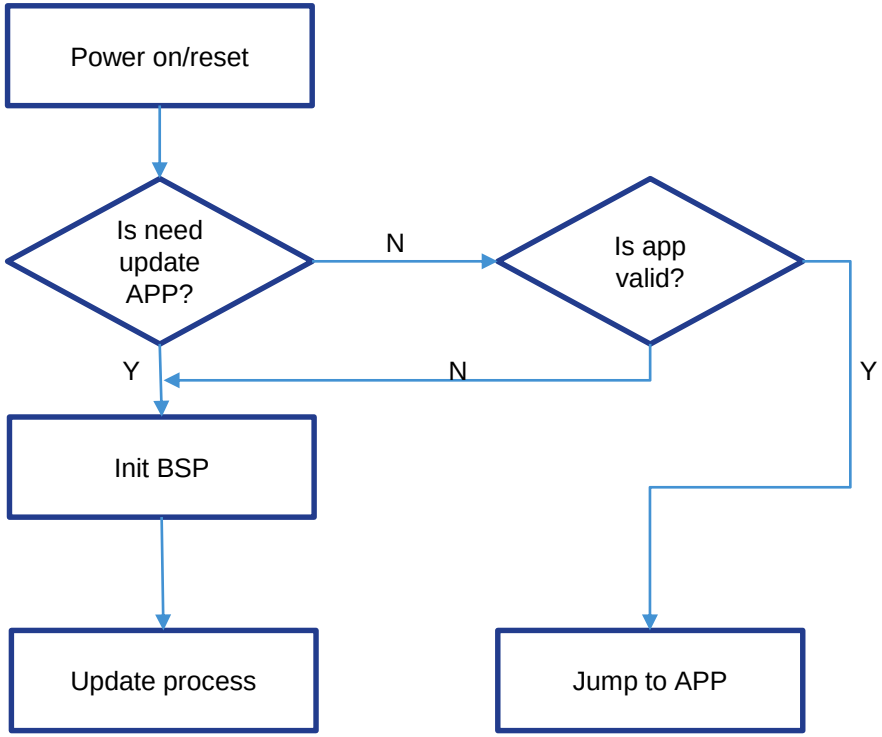
- Hardware communicate driver(CAN/LIN etc.)
  - Read data from bus and transmit data in TP RX FIFO
  - Read data from TP TX FIFO and write to bus

How to porting bootloader to new platform

For new platform, it’s easy to porting the stack.

- Have the same process
  - Divide FLASH and RAM space for bootloader and APP in link file
  - Porting driver to HAL (E.g., watchdog, timer, flash)
  - Config boot information (APP address and exchange information)
  - Config in user config file
- Different process
  - Additional to configure UDS layer. Modify diagnostic services in UDS configure layer.

Bootloader power on / reset process



How dose the bootloader and APP communicate

- Have 2 scenarios APP and bootloader should exchange information.
  - Scenarios #1 for APP received update firmware command

    APP should do:
    1. Set request enter bootloader mode
    2. Update exchange information CRC
    3. Send request more time over UDS layer
    4. Trigger MCU reset over watchdog

    For enter bootloader, check request enter bootloader was set and CRC is valid
        1. Clear request enter bootloader mode
        2. Update exchange information CRC
        3. Send enter bootloader mode command to UI/host over UDS protocol
  - Scenarios #2 for Bootloader update firmware successful

    Bootloader should do:
        1. Set APP update successful
        2. Update exchange information CRC
        3. Send request more time over UDS protocol to UI/host
        4. Watchdog reset

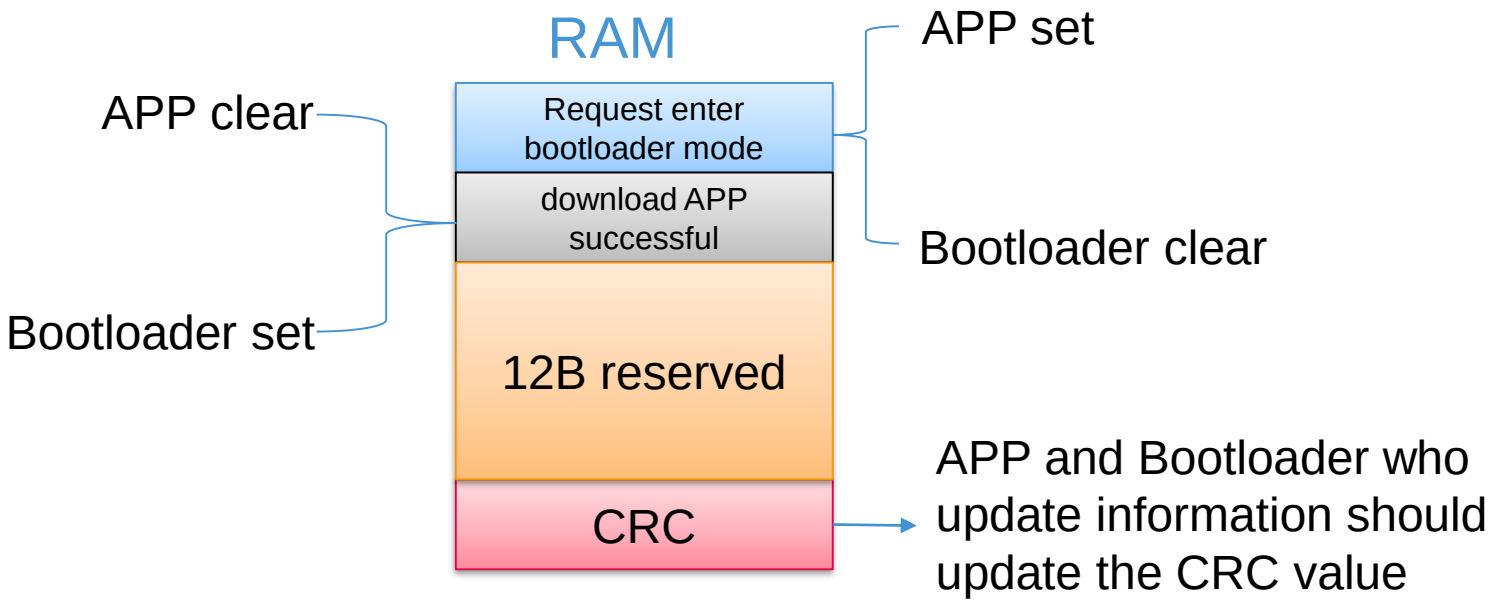
    For enter APP should do:
        1. Check update APP successful was set and exchange information CRC is valid
        2. Clear update APP successful flag and update exchange information CRC
        3. Send positive message over UDS layer to UI/host

How dose the bootloader and APP communicate

So, have some information need to exchange in APP and bootloader. For the information can storage in FLASH/EEPROM/RAM.

Normally bootloader have not flash driver and don’t need to EEPROM. So, can put the information in RAM. Retention RAM or RAM(If MCU reset (not power off), storage information not lost.)

How dose the bootloader and APP communicate



In Bootloader, how to confirm of the APP validity

- In bootloader, should confirm of the APP validity when power on or reset. These data should storage in P-FLASH.

These data includes as below information:

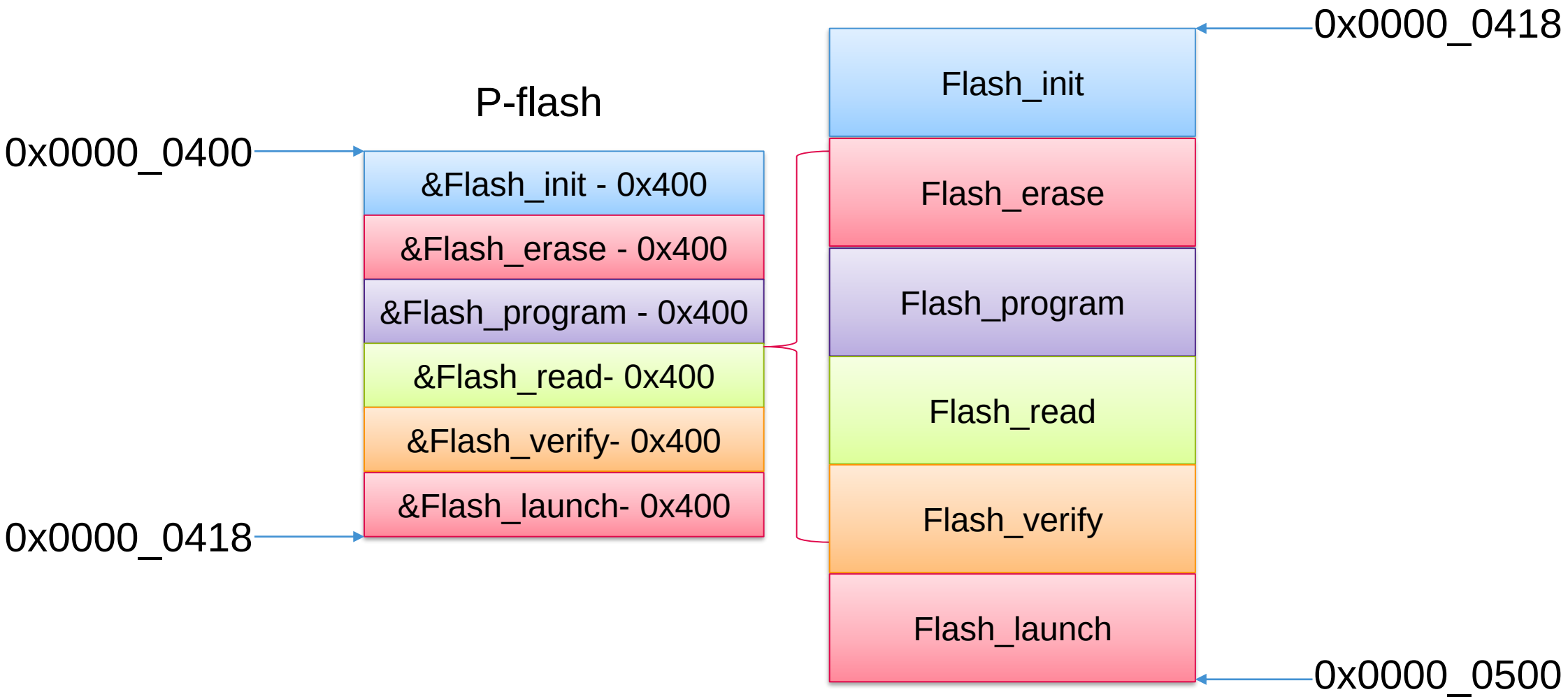
- Is APP flash space erased?
- Is APP flash space programed?
- Is APP data structure valid?
- Fingerer information
- APP image CRC (option)
- The structure CRC

If the structure CRC valid and step 1, 2, 3 is valid, then bootloader judged the APP image is valid.

Where is the flash driver storage

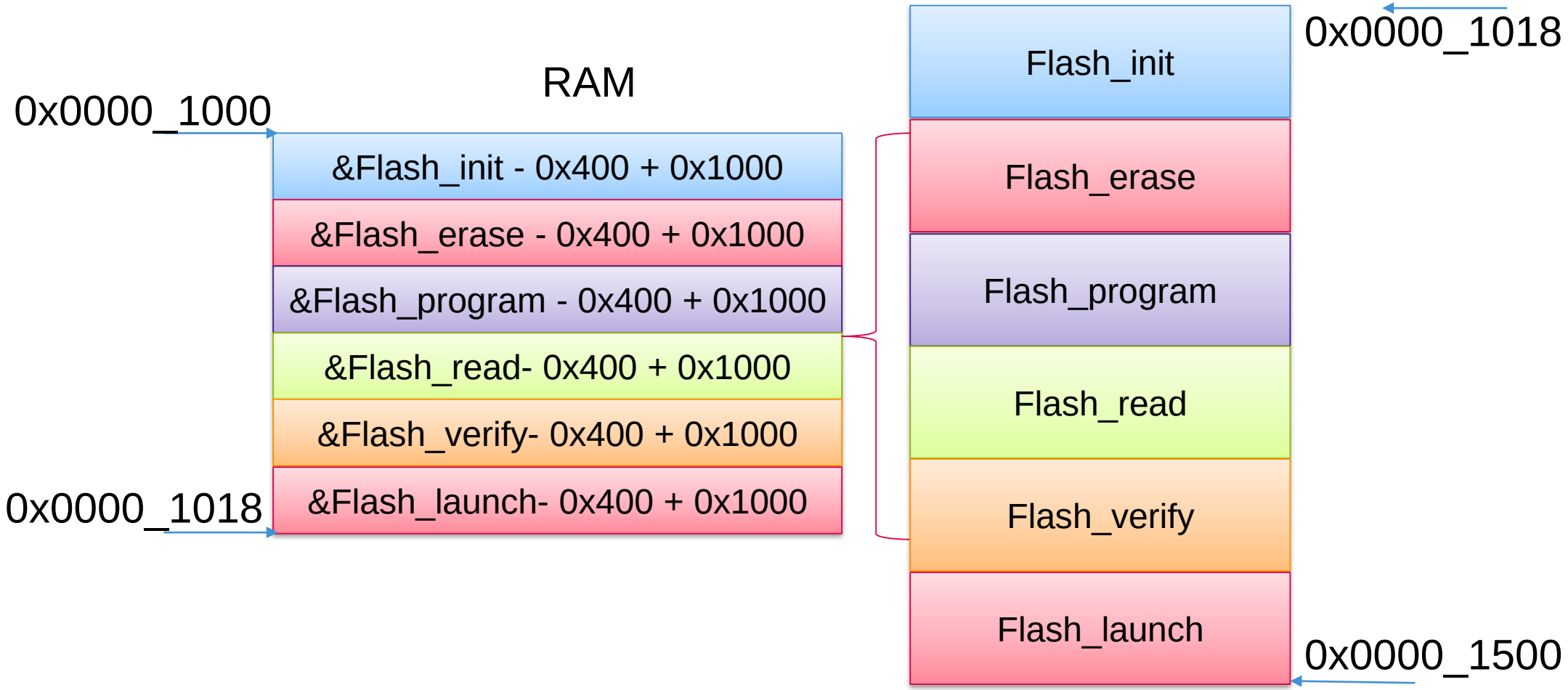
- Flash driver maybe storage in MCU or host. For HIS standard, flash driver should be storage in UI/host. For my opinion, flash driver storage in UI/host is for safety. Bootloader cannot erase/program flash content without flash driver. For download APP firmware, have some 3 important steps:
  - step 1 download flash driver in RAM
  - step 2 download APP firmware
  - step 3 erase flash driver from RAM

How to achieve flash driver is independent position code

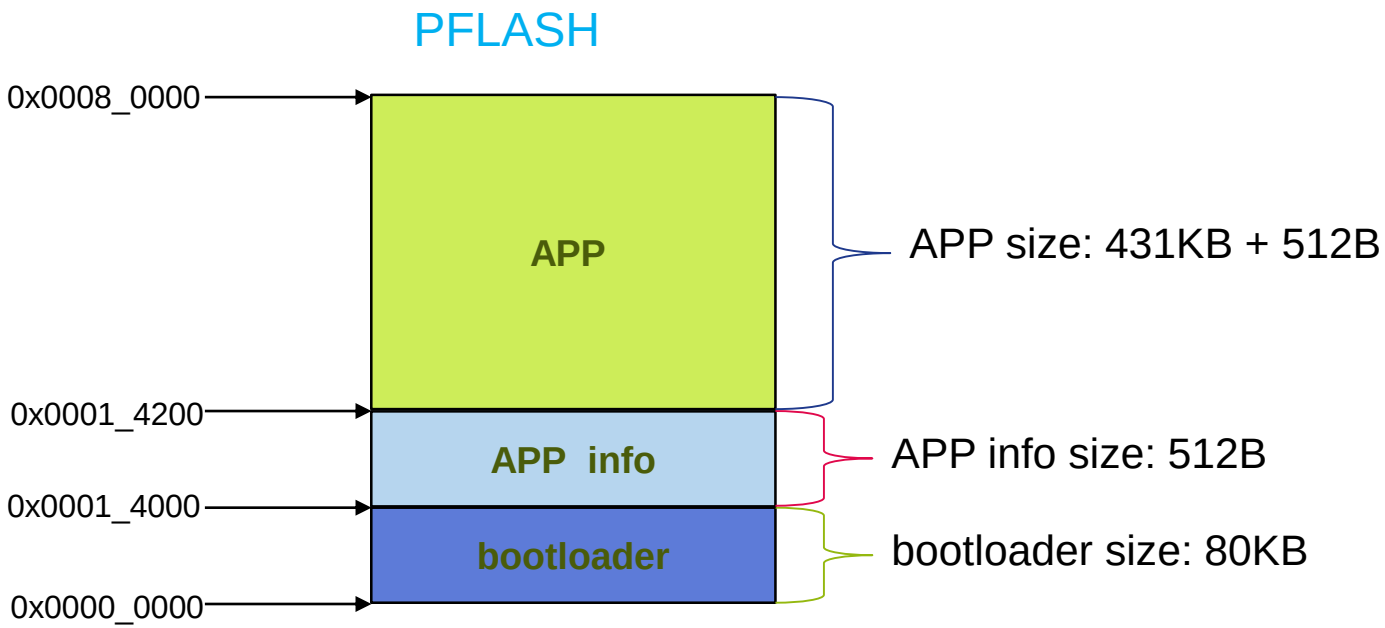


How to achieve flash driver is independent position code

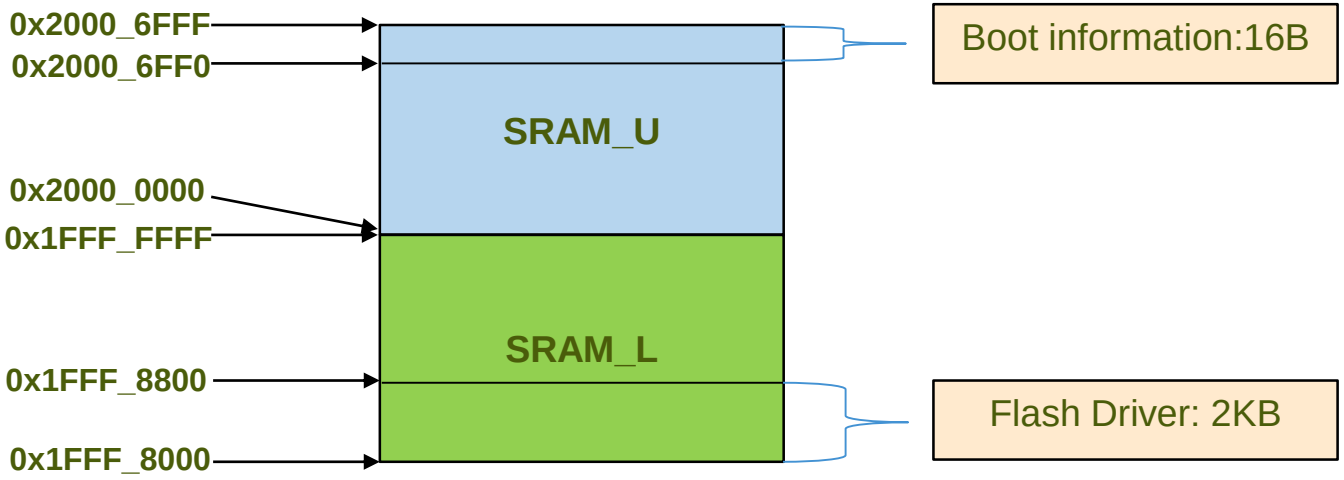
Copy flash drive in RAM. If copy flash driver in RAM with start 0x1000. So, calculate as below:



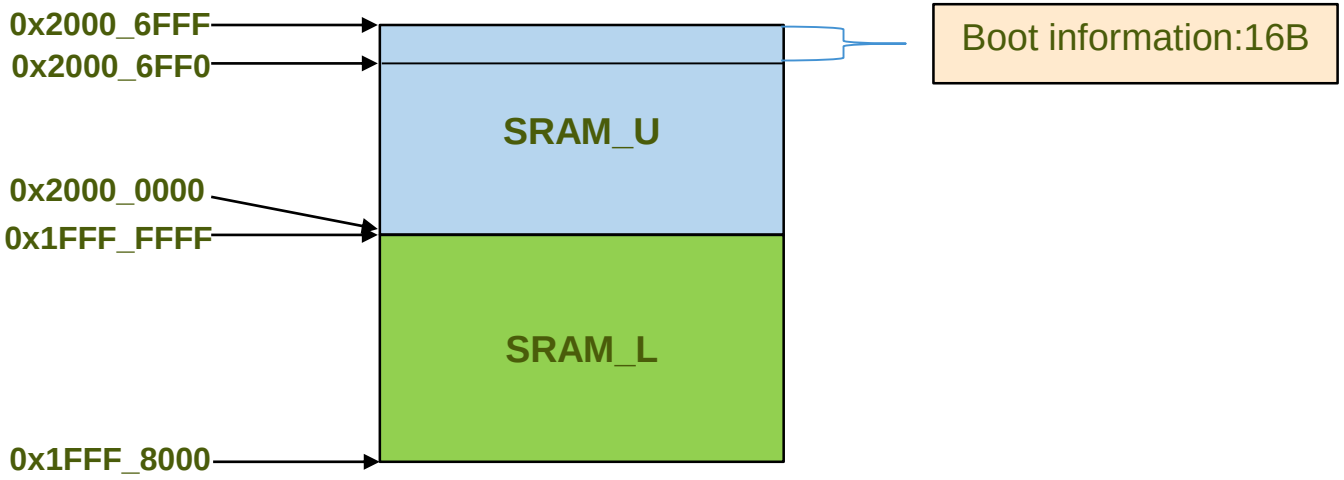
CAN Bootloader, APP memory map based on S32K144



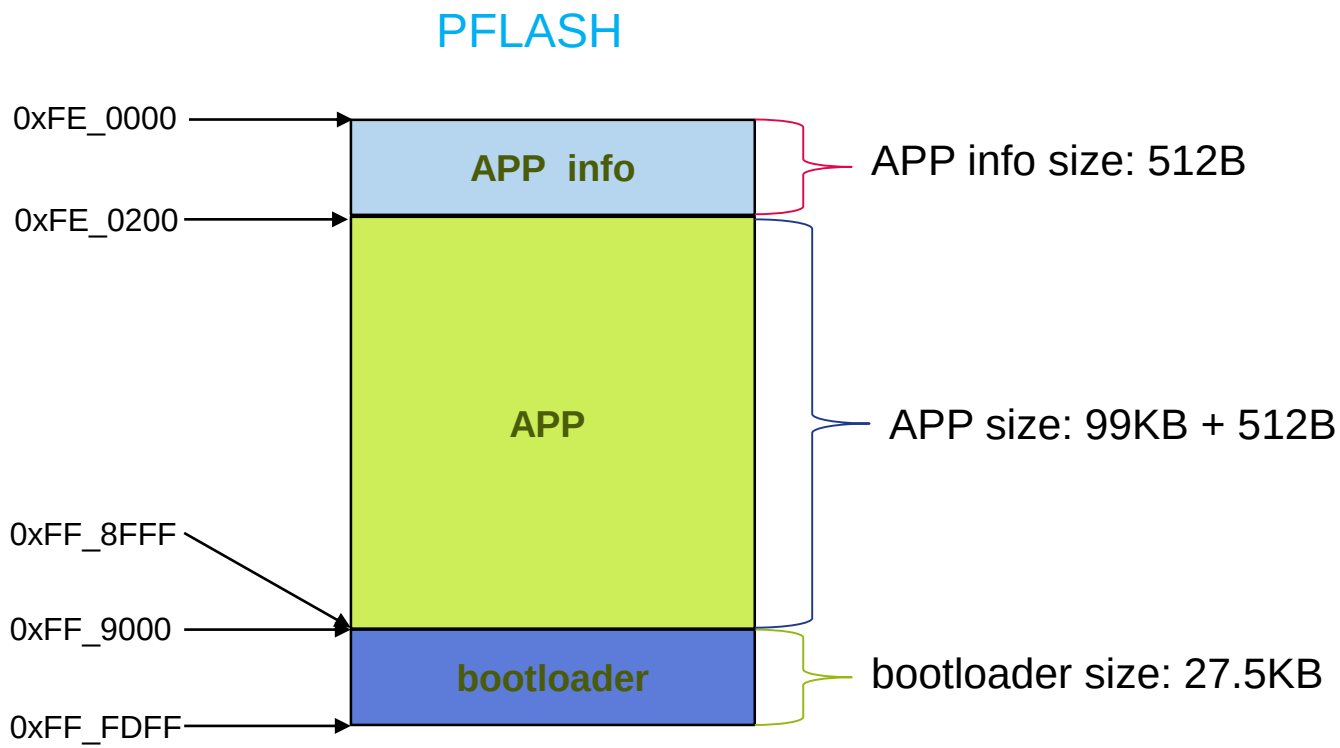
CAN Bootloader SRAM memory map based on S32K144



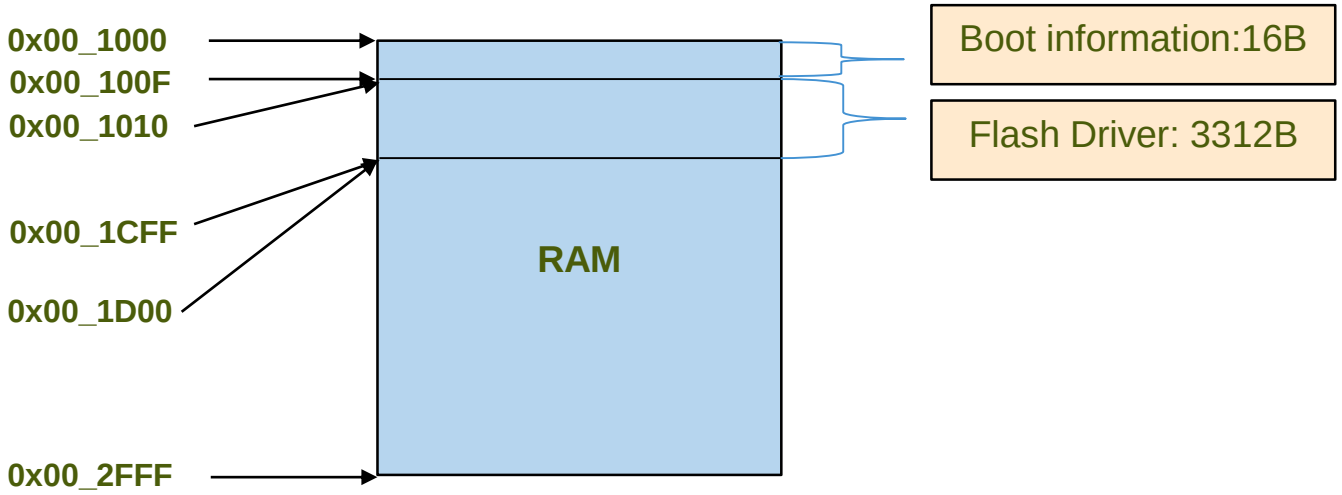
CAN Bootloader, APP SRAM memory map based on S32K144



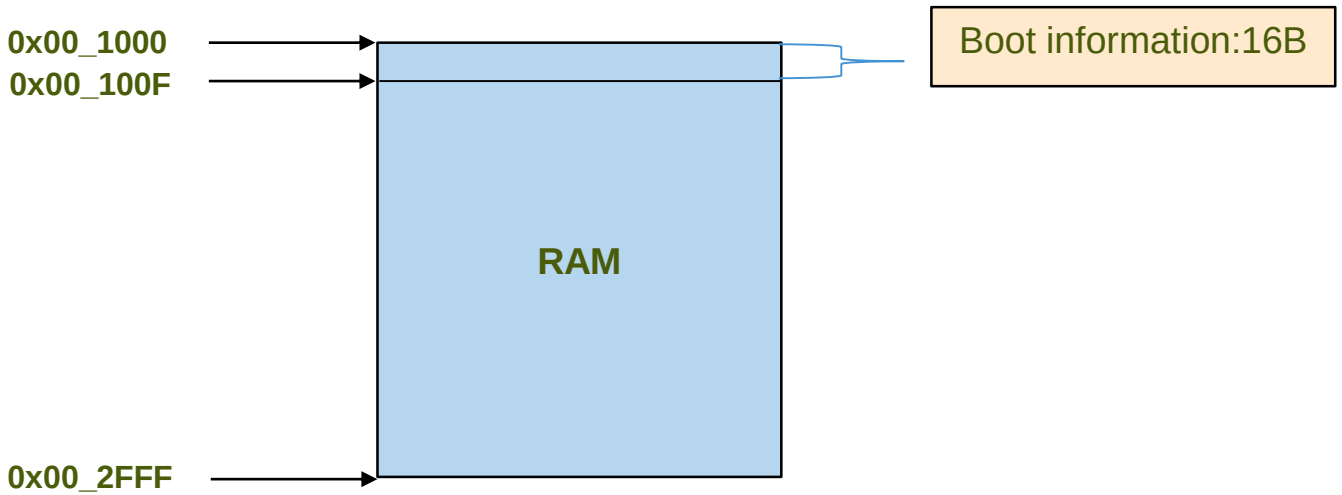
LIN Bootloader, APP memory map based on S12ZVML



LIN Bootloader RAM memory map based on S12ZVML



LIN Bootloader, APP RAM memory map based on S12ZVML



CAN/LIN Bootloader code size

stack size				
No.	Compiler	Optimization level	description	code flash size
1	GCC	-O0	without print information over UART	54KB
		-O0 & -O1	include print information over UART SDK -O1 stack - O0	41KB
		-O1	include print information over UART SDK -O1 stack - O1	33KB
		-O0	only stack	24KB
		-O1	only stack	15KB
2	IAR	-O0	only stack	18KB
		-O1	only stack	17KB
3	Codewarrior	off	stack + driver + print debug info	27KB
		off	stack + driver	25KB
		off	only stack	21KB
		1	only stack	19.4KB

CAN/LIN Bootloader performance

Platform	System clock	Flash driver size	APP size	Baud rate	STMin	Total test times	Every times used time
S32K144	48MHz	2KB	48KB	500K(CAN)	1	5000	14s
					0	100	7s
					2	100	20s
S12ZVML	50MHz	2KB	20KB	19200(LIN)	0	5000	58s

Stress test for unified bootloader

In order to achieve the production level and high quality, NXP has made some test cases for the unified bootloader. The test cases include: normal test, reset and power off/on when downloading firmware.

No.	Test case	Description	Test time	Result
1	Normally test	About 5000times (S32K144/CAN) About 4500times (S12ZVML/LIN)	5000 * 14s=19.4h 4500 * 1min=75h	pass
2	Reset test	Period trigger reset and reupdate firmware	About 5h(1000 times)	pass
3	Power on/off test	Period trigger power off/on and reupdate firmware	About 7h(1400 times)	pass

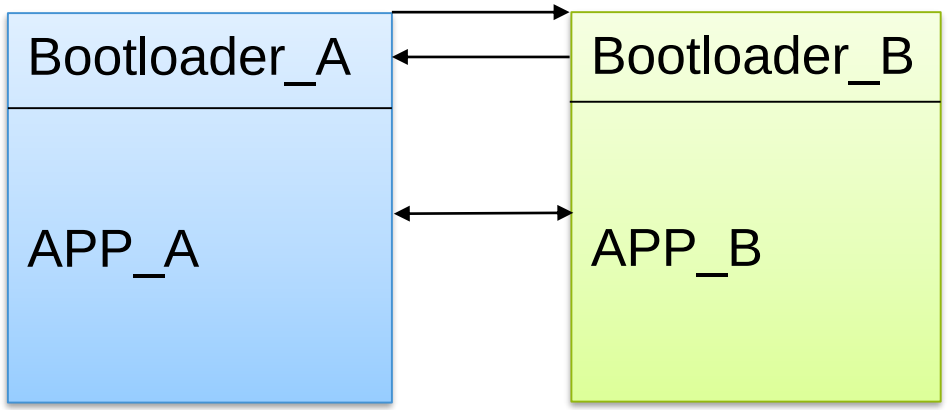
FTOA/OTA support by unified bootloader

- OTA: Over The Air (OTA) (or Over-The-Air) is a standard for the transmission and reception of application-related information in a wireless communications system.

- For achieve OTA, MCU should have two important features with hardware
  - Swap A/B when updated firmware
  - APP A/B running can download another APP B/A.

  Customer maybe need FOTA or OTA feature on MCU. But some MCUs cannot support FOTA/OTA over hardware. E.g., S32K146/8. So, should add some features in bootloader for support FOTA/OTA.

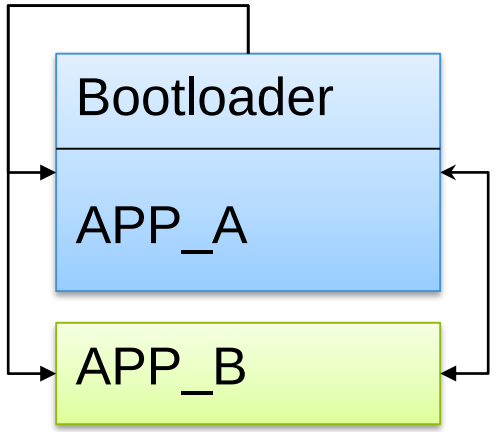
FTOA/OTA



- OTA diagram as below. Customer have some requirements for OTA:
  - Bootloader_A and APP_A can download bootloader_B and APP_B
  - Bootloader_B and APP_B can download bootloader_A and APP_A

When download firmware successful, set swap registers and reset MCU can jump to the newest firmware running.

Unified bootloader OTA solution



- For MCUs not support OTA over hardware. NXP provide some solutions for support OTA over software. Some requirements as below:
  - 2 flash partition for storage bootloader, APP A and APPB
  - Vector table can remap
- Feature description
  - Bootloader can update APP A and APP B
  - When APP A running can update APP B
  - When APP B running can update APP A
  - When APP A/B updated APP B/A and trigger watchdog reset, bootloader can jump to newest APP

Unified bootloader OTA solution

- Unified bootloader current version not support OTA. If customer need the feature, please contact us.

Unified bootloader Support ECU list

No.	Platform
1	S12ZVC128 (CAN)
2	S12ZVML128 (LIN)
3	S32K116 (CAN & LIN)
4	S32K118 (CAN)
5	S32K142 (CAN)
6	S32K144 (CAN & LIN)
7	S32K146 (CAN OTA)
8	S32K148 (CAN)
9	MPC5775E (CAN)

UI/host environment in windows

- Windows 7 or 10
- PEAK CAN tool

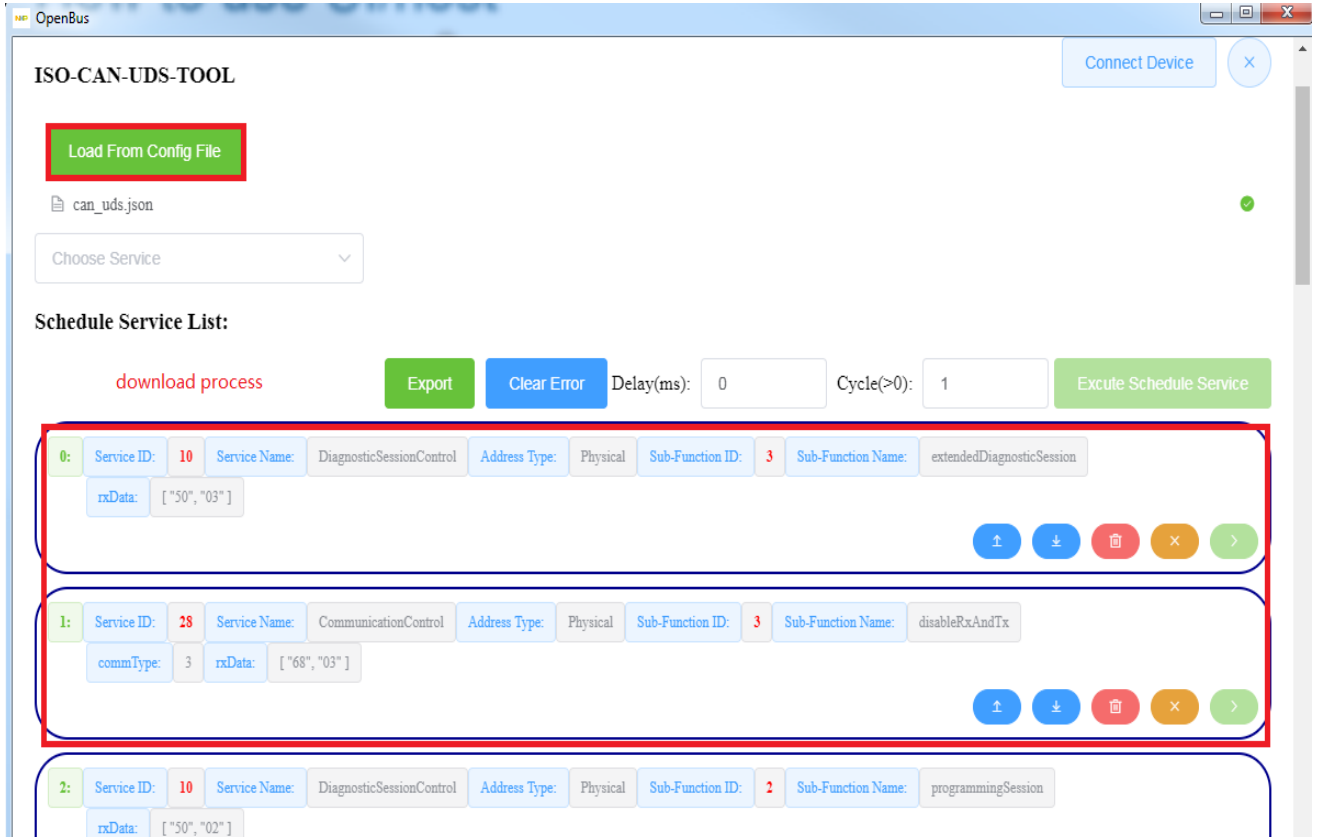
How to use UI/host

- Open NxpOpenBus.exe
- Choose CAN/LIN UDS over CAN/LIN BUS

How to use UI/host

- Load download file -> Load From Config file
- Choose CAN/LIN UDS over CAN/LIN BUS -> Connect Device

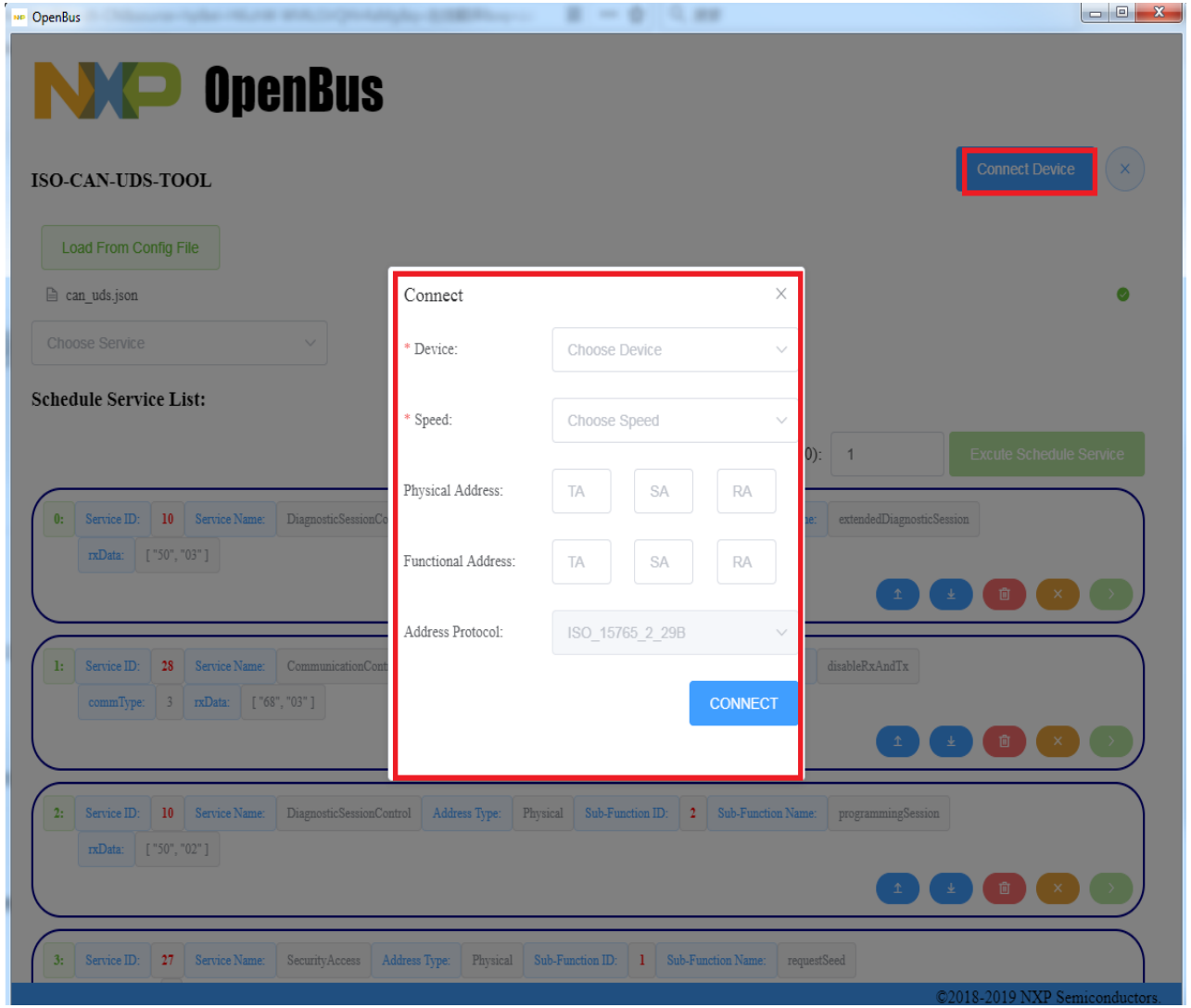
How to use UI/host



- Load download file ->

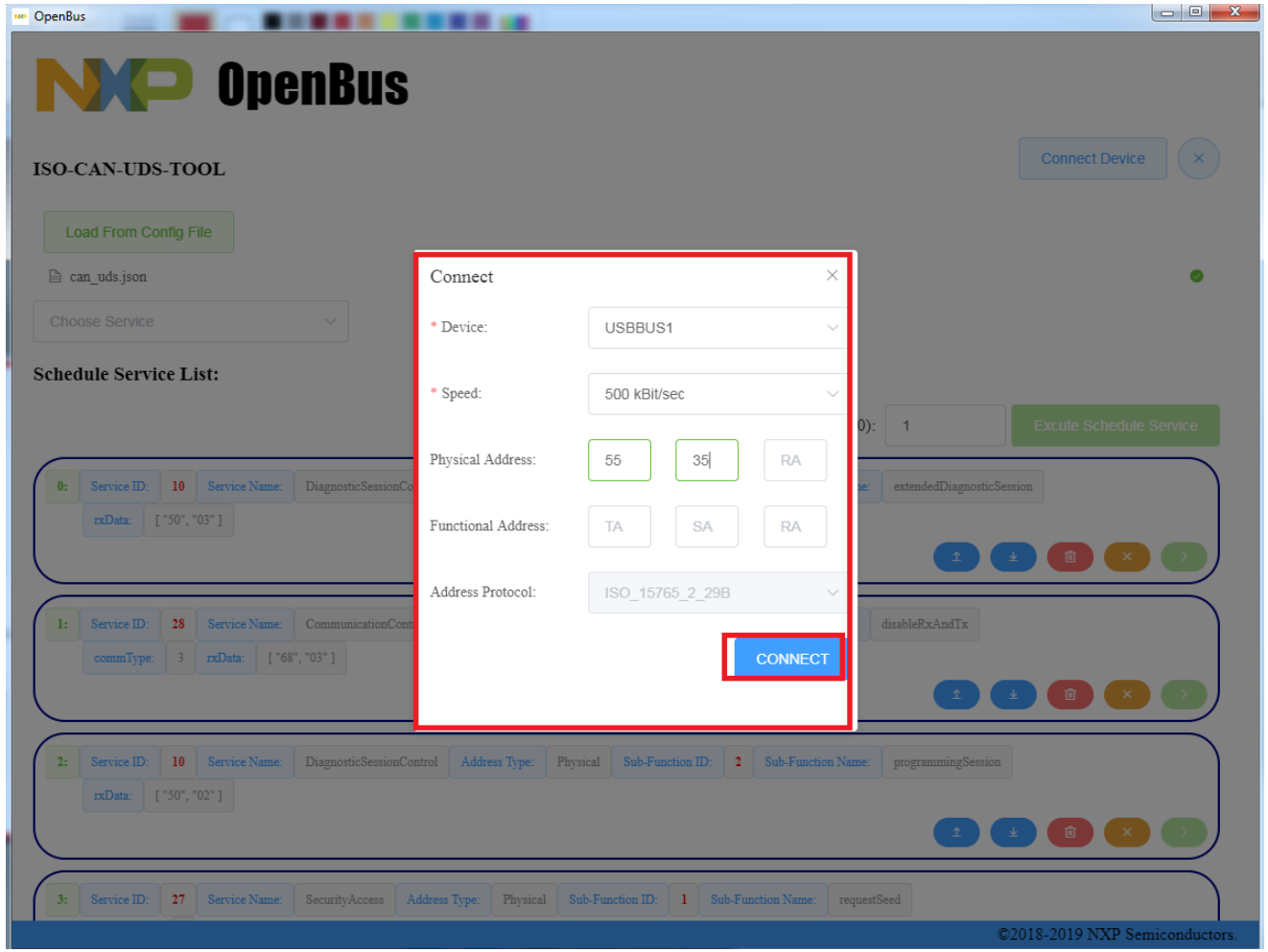
  Load From Config file

How to use UI/host



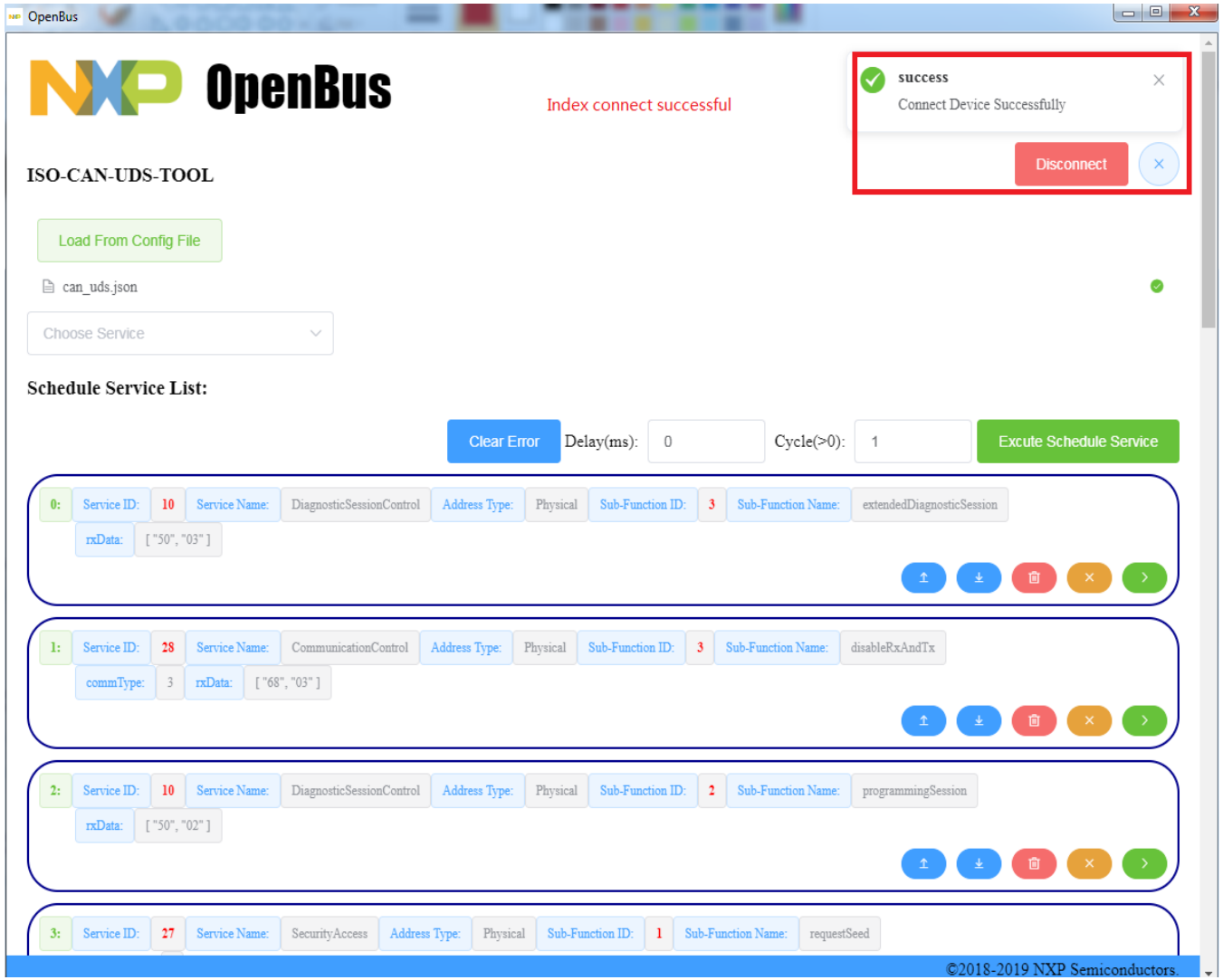
- Choose CAN/LIN UDS over CAN/LIN BUS -> Connect Device
- Choose USB device, CAN speed, config physical address (TA: target address, SA (source address)

How to use UI/host



- Choose CAN/LIN UDS over CAN/LIN BUS -> Connect Device
- Choose USB device, CAN speed, config physical address (TA: target address, SA (source address)

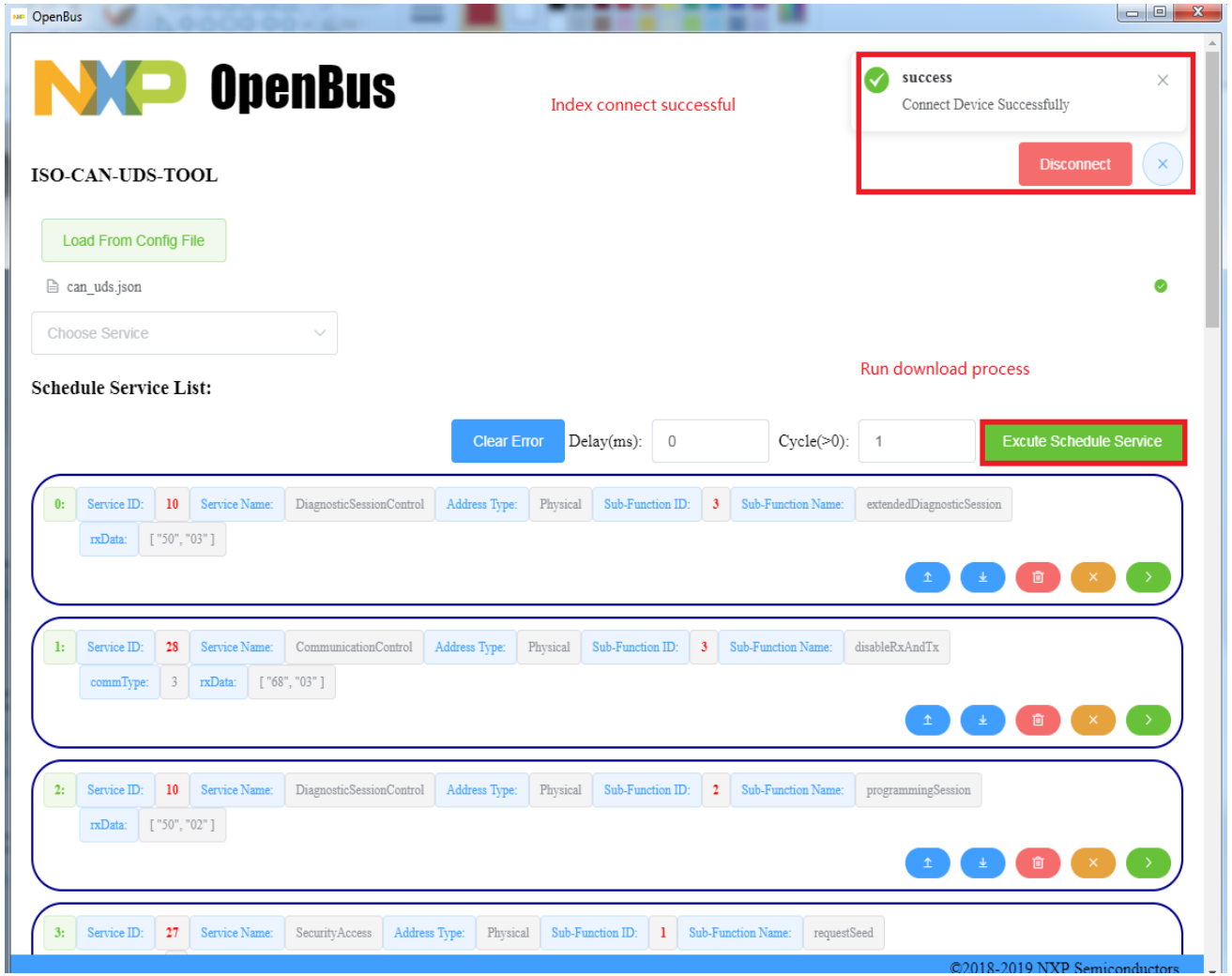
How to use UI/host



Choose CAN/LIN UDS over CAN/LIN BUS -> Connect Device

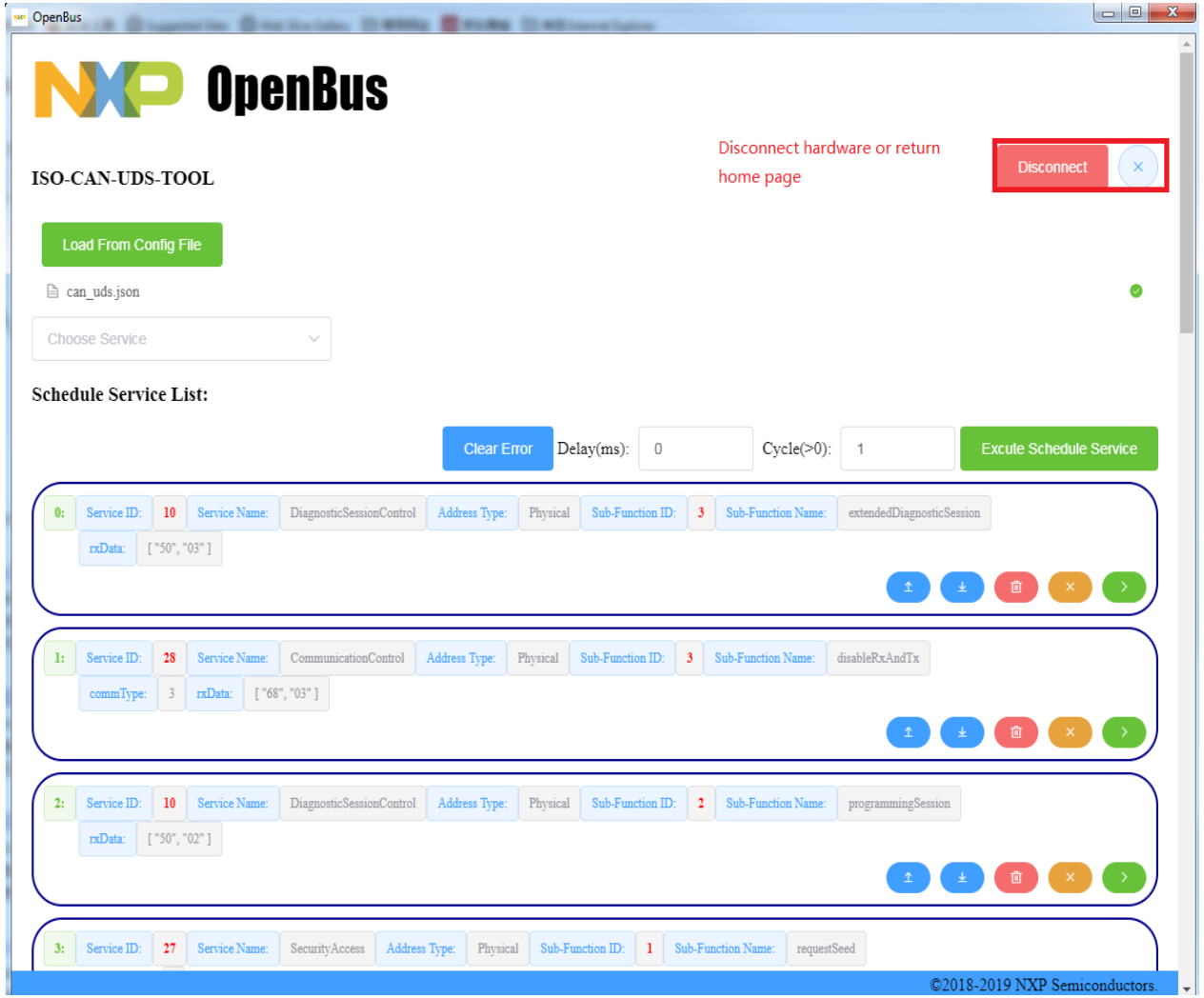
Choose USB device, CAN speed, config physical address (TA: target address, SA (source address)

How to use UI/host



- Choose CAN/LIN UDS over CAN/LIN BUS -> Connect Device
- Run download process

How to use UI/host



- Disconnect and return home page

How to select flash driver and APP

- Open JASON file with Notepad++
- Search bin, user can changed the file path for flash driver and APP

```
160         ],
161         "fileName": "C:\\Users\\nxf10035\\Desktop\\LIN\\uds_config\\S32K144_CAN\\flash_api.bin",
235         "fileName": "C:\\Users\\nxf10035\\Desktop\\LIN\\uds_config\\S32K144_CAN\\S32k144_UDS_Bootloader_App_Test.bin",
```

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com