SBL QSPI ENET

Introduction

Note

The steps in this user guide show screen shots and descriptions based on Windows. However the steps in Linux would remain the same, unless mentioned otherwise.

This bootloader does SOC initializations in addition to providing an option to receive an application image via UDP over ethernet and flashing the received application to QSPI Flash and attempts to boot a multicore appimage present at 0xA0000 location in the QSPI Flash. To flash a multicore appimage at this location, follow the steps mentioned in **Basic steps to flash files**.

This example is a UDP IP application using the ethernet driver (ENET)

On AWR294X, we can do ethernet based communication using CPSW as HW mechanism

- CPSW is a standard ethernet switch + port HW
- It uses ethernet driver underneath

The transfer of the user application image works in conjunction with the enet_uniflash.py python script mentioned in **Flashing Tools**.

If a multicore appimage is found at the location, the SBL parses it, splits it into RPRCs for each core applicable. Each core is then initialized, RPRC image is loaded, entry points are set and the core is released from reset. For more on bootflow/bootloaders, please refer **Understanding the bootflow and bootloaders**

Note that this SBL itself is transferred over UART only. It is only the files that are to be processed by this SBL are sent over ethernet.

There are 2 modes in which the user can run the sbl_qspi_enet bootloader to receive an application image over UDP via ethernet (by setting the MACRO ENETSBL_TRANSFER_START_MODE in the file sbl_enet.h),

- Timer Mode (ENETSBL_TIMER_MODE) (Default)
- Button Mode (ENETSBL_BUTTON_MODE)

The SBL runs the fetch of the application image over UDP via ethernet on its bootup and hence it is required to reset the board each time it is necessary to flash a new image over ethernet when in ENETSBL_TIMER_MODE. If the enet_uniflash.py script is not running on host PC during the reset, then the SBL will skip the transfer with the message "[ENETSBL TIMEOUT] Skipping enet transfer"

Ethernet driver initialization may be skipped upon reset by setting the MACRO ENETSBL_TRANSFER_START_MODE in the file sbl_enet.h to ENETSBL_BUTTON_MODE instead of ENETSBL_TIMER_MODE, thereby jumping directly to the user application upon reset.

When using the ENETSBL_BUTTON_MODE, after running the enet_uniflash.py python script on PC, user should press the reset switch SW1 while pressing and holding down the switch SW2 for the SBL to move into ethernet transfer mode to receive an application image over ethernet. If only a reset is performed by pressing switch SW1, the SBL will skip the fetch of the application image over ethernet with the message "[ENETSBL SKIP] Skipping enet transfer." and directly boot the existing application if found.

Note

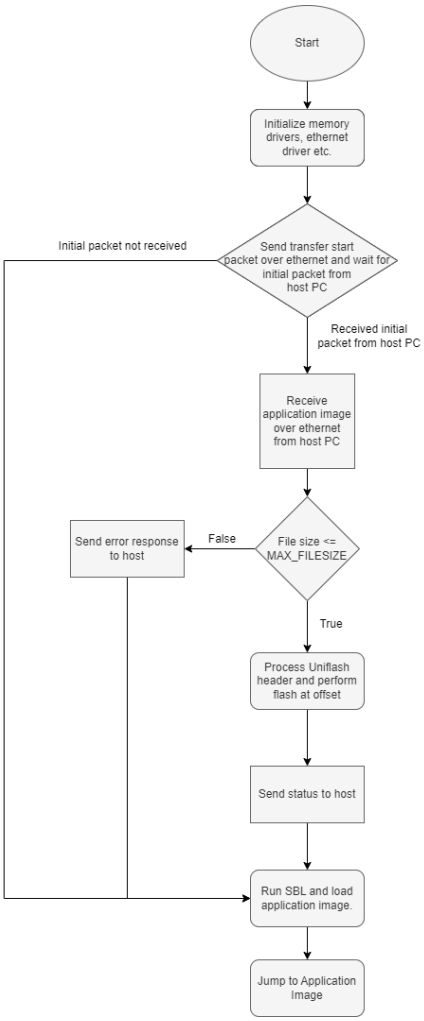
Make sure to provide a reserved area of 0x60000 at the beginning of MSS_L2 memory in the linker script of the application image that is to be flashed over ethernet to make sure that the application image does not overwrite the sbl_qspi_enet SBL.

Supported Combinations

Parameter	Value
CPU + OS	r5fss0-0 nortos
Toolchain	ti-arm-clang
Board	awr294x-evm
Example folder	examples/drivers/boot/sbl_qspi_enet

Application Flow

The below flow diagram shows the application flow for the reception of an application image via UDP over ethernet on boot of the sbl_qspi_enet SBL.



Flow Path

This application consumes more MSS_L2 memory (>0x20000) as compared to the sbl_uart_uniflash SBL due to the increased size of the ethernet driver library. Please refer to the mmwave_mcuplus_sdk_user_guide.pdf, section 4.8, on the procedure to restrict the memory used by the sbl_qspi_enet example to less than 0x20000 in MSS_L2.

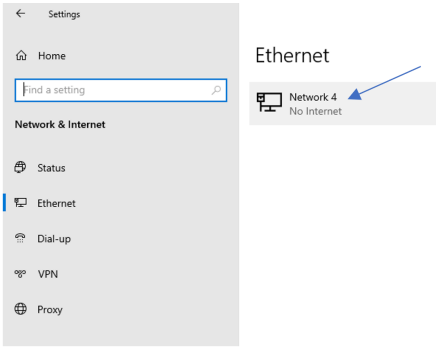
Create a network between EVM and host PC

The below steps are for the network interface corresponding to the ethernet connection between the PC and EVM, which in this case is “Network 4”.

The IP addresses used here are also used by the python script and the EVM application, so ensure to change the IP addresses in all the places if there’s any conflict with existing devices.

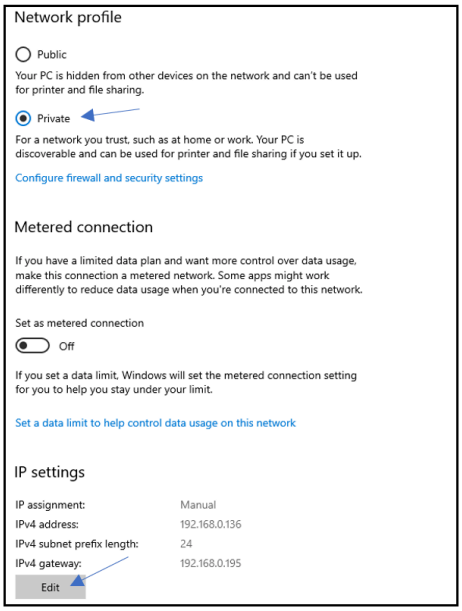
- Connect an ethernet cable between the PC and the EVM
- Open ethernet settings on your PC >> select corresponding ethernet adapter

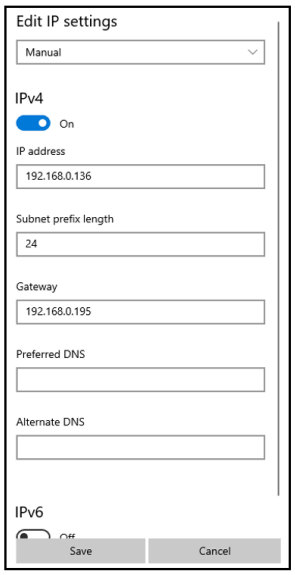
If setting the static IP for the first time, the ethernet adapter may appear as "Unidentified network", which is the network to be selected.



Network Select

- Edit the IP settings with the IP address as 192.168.0.136, Subnet Prefix Length as 24 and Gateway as 192.168.0.195.





Static IP Set

Make sure to set the connection as private and metered connection is set to off.

The above mentioned IP addresses are hardcoded in the file sbl_enet.h and hence any change in the same will also have to be edited in sbl_enet.h.

- Due to the limitations presented by Enet LLD in the Uniflash application, we need to manually add an ARP entry for the EVM in order for the PC to not drop the packets (PC will drop packets from sources for which it doesn't have an ARP entry.)

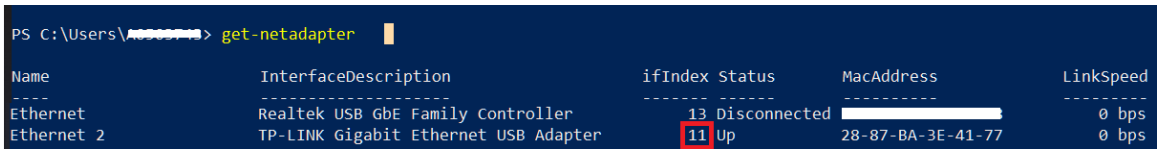
  Creating a static ARP entry requires admin privileges. Run the following commands in PowerShell as admin

```
New-NetNeighbor -InterfaceIndex <ifIndex> -IPAddress '192.168.0.195' -LinkLayerAddress '<EVM MAC
       Addr>' -State Permanent
```

Replace <ifIndex> with the interface index of the connection between PC and EVM.

To find out the interface index corresponding to the ethernet interface between the PC and the EVM, use the following PowerShell command. This does not require admin privileges.

```
Get-NetAdapter
```



IF Index

In the above figure, “Ethernet 2” is the required interface and its interface index is 11.

> **Attention**
>
> Make sure to set the value of the MACRO **ENET_HOST_PC_MAC_ADDRESS** in the file **sbl_enet.h** to the MAC address of the corresponding ethernet adapter, in the above case this is Ethernet 2's MAC address.

The default EVM MAC Address is 70:ff:76:1d:ec:f2 (as set from SysConfig (Enet(CPSW) >> System Integration Config >> MAC Address)).

Replace <EVM MAC Addr> with the MAC Address of the EVM, as a continuous string like **70ff761decf2**.

An example command to set the ARP looks as below based on the above shown images.

```
New-NetNeighbor -InterfaceIndex 11 -IPAddress '192.168.0.195' -LinkLayerAddress '70ff761decf2' -
        State Permanent
```

Steps to Run the Example

Since this is a bootloader, the example will be run every time you boot an application using this example. It is run from a QSPI boot media unlike other examples which are usually loaded with CCS. Nevertheless, you can build this example like you do for the others using makefile or build it via CCS by importing as a project.

- **When using CCS projects to build**, import the CCS project for the required combination and build it using the CCS project menu (see **Using SDK with CCS Projects**).
- **When using makefiles to build**, note the required combination and build using make command (see **Using SDK with Makefiles**)
- Refer to the page **Basic steps to flash files** to flash the sbl_qspi_enet bootloader to the EVM in UART Boot Mode. Note that you should flash this bootloader sbl_qspi_enet and not the default mentioned sbl_qspi bootloader. You can optionally also flash the application image at flash offset 0xA0000 as mentioned in **Basic steps to flash files** or you can skip it and use the below step to send the application image over ethernet.
- Once the sbl_qspi_enet bootloader image is flashed on the EVM, switch to QSPI Boot Mode and refer to the page **Basic steps to flash files over ethernet** to send an application image over ethernet to the EVM.

See Also

BOOTLOADER

Sample Output

A normal boot in ENETSBL_TIMER_MODE without image transfer looks as shown below where it skips the transfer and boots up the user application image (Log from serial terminal):

```
[ ENETSBL ] Starting Ethernet Transfer ...
EnetPhy_bindDriver: PHY 0: OUI:080028 Model:23 Ver:01 <-> 'dp83867' : OK
PHY 0 is alive
[ ENETSBL ] initQs() txFreePktInfoQ initialized with 16 pkts
[ ENETSBL ] EVM MAC address: 70:ff:76:1d:ec:f2
[ ENETSBL ] PHY 0 is alive
[ ENETSBL ] Please wait for Linkup ...
Cpsw_handleLinkUp: Port 1: Link up: 100-Mbps Full-Duplex
[ ENETSBL ] Linkup Done!
[ ENETSBL TIMEOUT ] Skipping enet transfer.
Cpsw_handleLinkDown: Port 1: Link down

Starting QSPI Bootloader ...
[BOOTLOADER_PROFILE] Boot Media       : NOR SPI FLASH
[BOOTLOADER_PROFILE] Boot Media Clock : 80.000 MHz
[BOOTLOADER_PROFILE] Boot Image Size  : 30 KB
[BOOTLOADER_PROFILE] Cores present    :
r5f0-0
[BOOTLOADER PROFILE] System_init                     :        154us
[BOOTLOADER PROFILE] Drivers_open                    :         30us
[BOOTLOADER PROFILE] Board_driversOpen               :         57us
[BOOTLOADER PROFILE] CPU load                        :    8483203us
[BOOTLOADER_PROFILE] SBL Total Time Taken            :    8483448us

Image loading done, switching to application ...
INFO: Bootloader_runSelfCpu:217: All done, reseting self ...

Hello World!
```

Boot with successful transfer of an application image over ethernet looks as below where it boots up the application image after the transfer of the same over ethernet. You should observe the message "[ENETSBL SUCCESS] Ethernet Transfer Done." if the transfer is successful (Log from serial terminal)::

```
[ ENETSBL ] Starting Ethernet Transfer ...
EnetPhy_bindDriver: PHY 0: OUI:080028 Model:23 Ver:01 <-> 'dp83867' : OK
PHY 0 is alive
[ ENETSBL ] initQs() txFreePktInfoQ initialized with 16 pkts
[ ENETSBL ] EVM MAC address: 70:ff:76:1d:ec:f2
[ ENETSBL ] PHY 0 is alive
[ ENETSBL ] Please wait for Linkup ...
Cpsw_handleLinkUp: Port 1: Link up: 100-Mbps Full-Duplex
[ ENETSBL ] Linkup Done!
[ ENETSBL ] Receiving file, please wait ...
[ ENETSBL SUCCESS ] Ethernet Transfer Done.
[ ENETSBL ] Packets Received   :  25
[ ENETSBL ] Total File Size    :  35316 Bytes
[ ENETSBL ] Flash Offset       :  0xA0000

Cpsw_handleLinkDown: Port 1: Link down

Starting QSPI Bootloader ...
[BOOTLOADER_PROFILE] Boot Media       : NOR SPI FLASH
[BOOTLOADER_PROFILE] Boot Media Clock : 80.000 MHz
[BOOTLOADER_PROFILE] Boot Image Size  : 34 KB
[BOOTLOADER_PROFILE] Cores present    :
r5f0-0
[BOOTLOADER PROFILE] System_init                     :        154us
[BOOTLOADER PROFILE] Drivers_open                    :         30us
[BOOTLOADER PROFILE] Board_driversOpen               :         57us
[BOOTLOADER PROFILE] CPU load                        :    7950991us
[BOOTLOADER_PROFILE] SBL Total Time Taken            :    7951236us

Image loading done, switching to application ...
```

```
INFO: Bootloader_runSelfCpu:217: All done, reseting self ...

[I2C] Temperature sensor found at device address 0x49
[I2C] Sample 0: 40.500000 (celcius)
[I2C] Sample 1: 40.500000 (celcius)
[I2C] Sample 2: 40.500000 (celcius)
[I2C] Sample 3: 40.500000 (celcius)
[I2C] Sample 4: 40.500000 (celcius)
[I2C] Sample 5: 40.500000 (celcius)
[I2C] Sample 6: 40.500000 (celcius)
[I2C] Sample 7: 40.500000 (celcius)
[I2C] Sample 8: 40.500000 (celcius)
[I2C] Sample 9: 40.500000 (celcius)
[I2C] Sample 10: 40.500000 (celcius)
[I2C] Sample 11: 40.500000 (celcius)
[I2C] Sample 12: 40.500000 (celcius)
[I2C] Sample 13: 40.500000 (celcius)
[I2C] Sample 14: 40.500000 (celcius)
[I2C] Sample 15: 40.500000 (celcius)
[I2C] Sample 16: 40.500000 (celcius)
[I2C] Sample 17: 40.500000 (celcius)
[I2C] Sample 18: 40.500000 (celcius)
[I2C] Sample 19: 40.500000 (celcius)
All tests have passed!!
```

The corresponding enet_uniflash.py script execution results on PC looks as below.(Log From Command Prompt)

```
C:\mcu_plus_sdk\tools\boot>python enet_uniflash.py --cfg=sbl_prebuilt\awr294x-
       evm\default_enet_app.cfg
[LOG] Parsing config file ...
[LOG] Ensure that sbl_qspi_enet has already been sent over before running this script.
[LOG] Found 1 command(s) !!!
[LOG] Creating socket
Starting Linkup ...
Starting Linkup ...
Received.
('192.168.0.195', 5001)

[LINKUP] (SUCCESS) EVM Linked up. Starting transfer ...

[STATUS] Sent file ../../examples/drivers/i2c/i2c_temperature/awr294x-evm/r5fss0-0_nortos/ti-arm-
       clang/i2c_temperature.release.appimage of size 35120 bytes in 0.03s.
[FLASH] (SUCCESS) Flashing successful !!!

[LOG] All commands from config file are executed !!!
```