AN13940

Connect i.MX RT1060 to USB 4G Module (RNDIS Mode)

Rev. 0 — 18 April 2023 **Application note**

Document Information

Information	Content
Keywords	i.MX RT, RT, USB, 4G, LTE
Abstract	This application note discusses how to connect i.MX RT1060 EVK to Internet by USB 4G module like EC200A-CN from Quectel.

1 Introduction

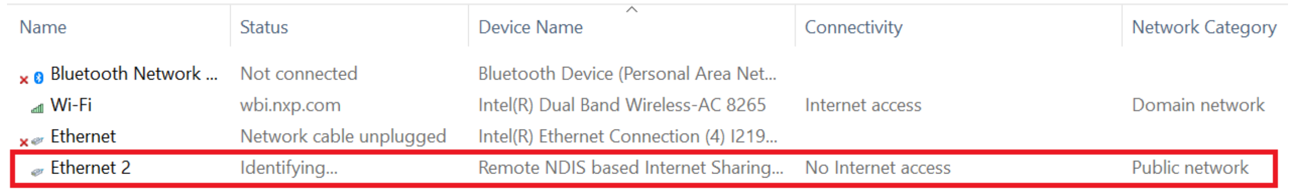
There is one SDK example ***evkbmimxrt1060_lwip_dhcp_usb_bm*** in the i.MX RT1060 EVK SDK. It connects the USB to the Internet with the three cell phones mentioned in the guide. But if customer wants to use a USB 4G module, instead of a cell phone listed in the guide, to connect to the Internet, and it is possible that this example cannot work in such condition. Then we must adapt this example to new USB 4G module.

This application note discusses how to implement the adaption when the terminal becomes a USB 4G module like EC200A-CN from Quectel, and then connect to the Internet with this USB 4G module.

2 Basics

2.1 Introduction of RNDIS

Remote Network Driver Interface Specification (RNDIS) is a USB protocol defined by Microsoft. The windows OS can recognize the USB device which follows this specification automatically and treat it as a network device, as shown in Figure 1.



Name	Status	Device Name	Connectivity	Network Category
Bluetooth Network ...	Not connected	Bluetooth Device (Personal Area Net...		
Wi-Fi	wbi.nxp.com	Intel(R) Dual Band Wireless-AC 8265	Internet access	Domain network
Ethernet	Network cable unplugged	Intel(R) Ethernet Connection (4) I219...		
Ethernet 2	Identifying...	Remote NDIS based Internet Sharing...	No Internet access	Public network

Figure 1. USB 4G module connected to windows

2.2 Network layer model for USB RNDIS

Figure 2 shows the network layer model for USB RNDIS.

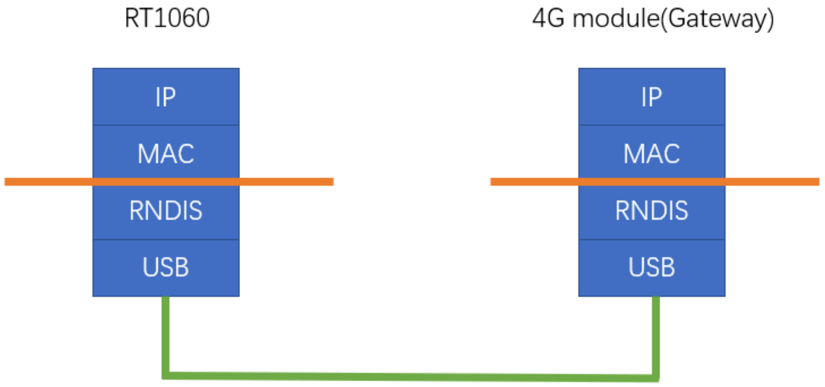


Figure 2. Network layer model for USB RNDIS

2.3 RNDIS interface/endpoint analysis

In the original SDK example, it uses two interfaces and three endpoints.

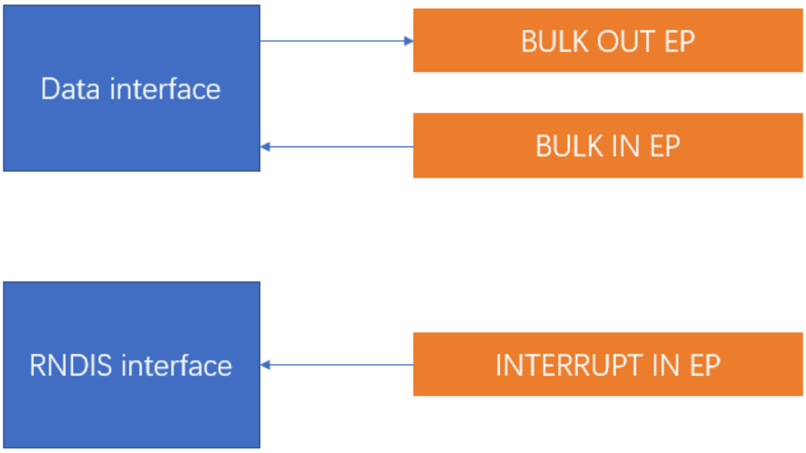


Figure 3. Endpoint used in default SDK example

Here, the data interface is used for Ethernet package communication, and there is one endpoint for each direction. RNDIS interface is used for device status polling.

2.4 EC200A-CN interface/endpoint analysis

EC200A-CN implements five interfaces, which include the interfaces introduced above.

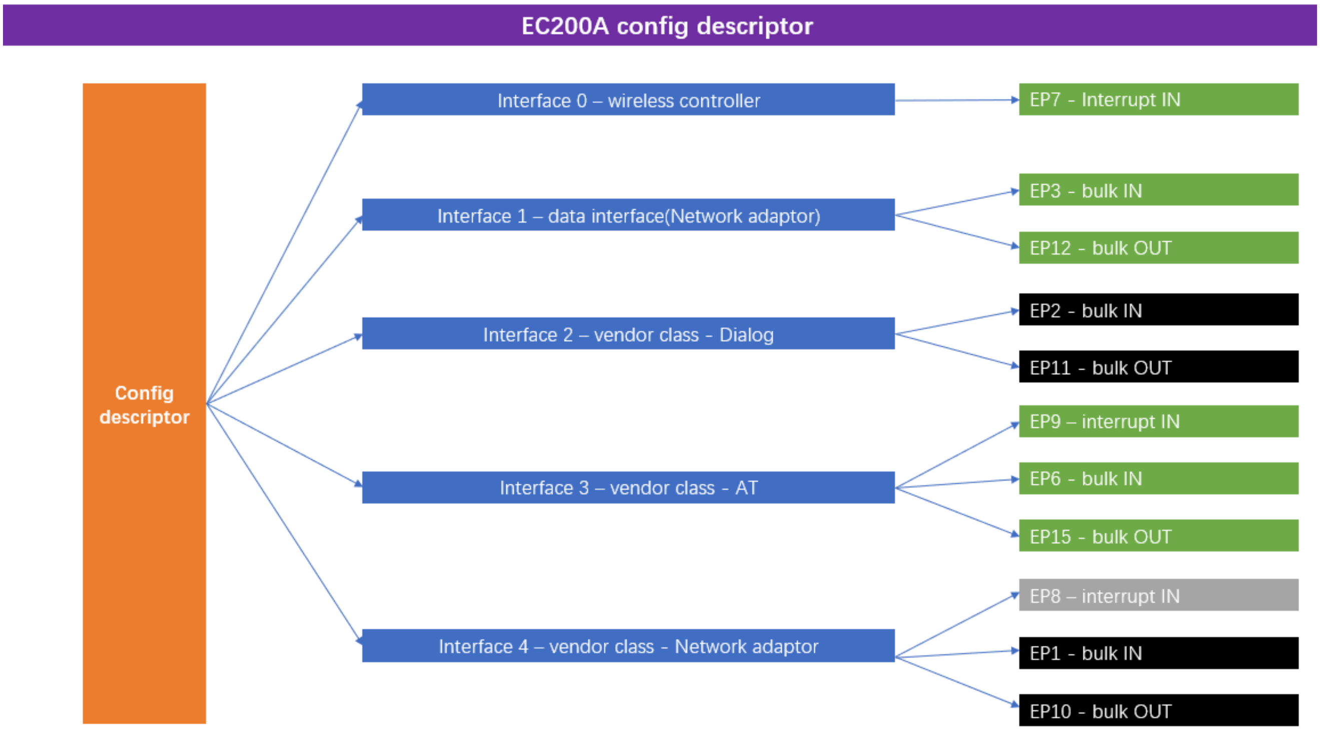


Figure 4. EC200A-CN interface/endpoint analysis

The additional interface used here is the AT interface.

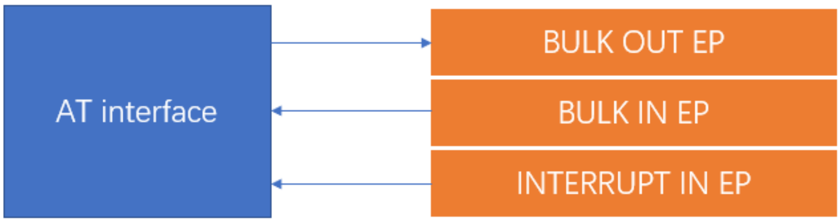


Figure 5. AT interface

The AT interface is used to dial (send AT command and get response), poll, and control the USB 4G module. We need poll the USB 4G module status and send the command to it to set it to the online mode. Then, the IP package from/to RT1060 EVK can be sent to/received from the Internet.

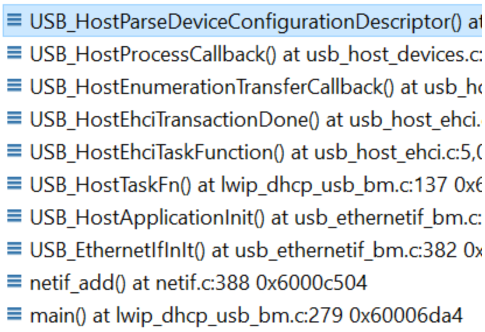
To enable/adapt to the new 4G module, we must:

- **Enable AT interface**
- **Dial on AT interface**

The other interfaces defined by the EC200A-CN are not important for us here, so they are not introduced in this document.

2.5 Some basics for USB host application by NXP SDK

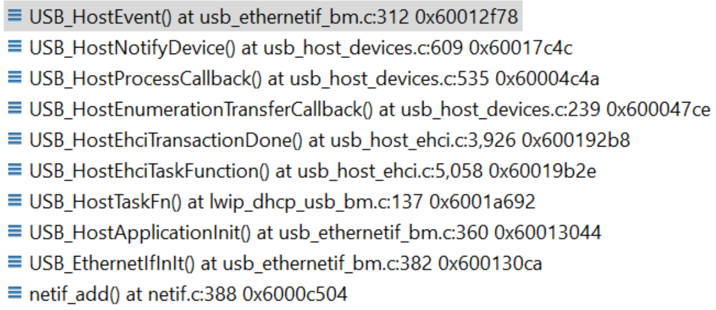
- NXP SDK USB stack decodes the configuration descriptor automatically. It can detect all the interfaces and endpoints in the configuration descriptor.
  See `USB_HostParseDeviceConfigurationDescriptor()` in *usb/host/usb_host_devices.c*.
  Figure 6 shows the related call stack. Call stack analysis is useful for us to understand how the SDK USB stack is organized.



```
≡ USB_HostParseDeviceConfigurationDescriptor() at
≡ USB_HostProcessCallback() at usb_host_devices.c:
≡ USB_HostEnumerationTransferCallback() at usb_h
≡ USB_HostEhciTransactionDone() at usb_host_ehci.
≡ USB_HostEhciTaskFunction() at usb_host_ehci.c:5,
≡ USB_HostTaskFn() at lwip_dhcp_usb_bm.c:137 0x6
≡ USB_HostApplicationInit() at usb_ethernetif_bm.c:
≡ USB_EthernetIfInIt() at usb_ethernetif_bm.c:382 0x
≡ netif_add() at netif.c:388 0x6000c504
≡ main() at lwip_dhcp_usb_bm.c:279 0x60006da4
```

Figure 6. Call stack for `USB_HostParseDeviceConfigurationDescriptor()`

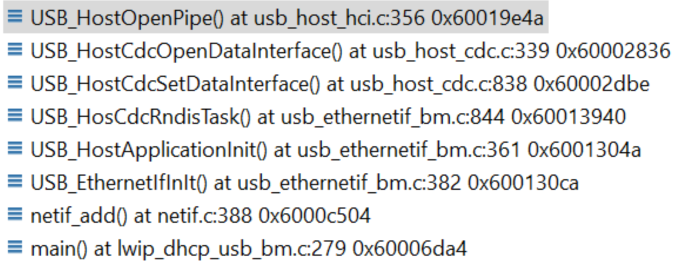
- For the event callback, see `USB_HostEvent()` in *lwip/port/usb_ethernetif_bm.c*.
  Figure 7 shows the related call stack.



```
USB_HostEvent() at usb_ethernetif_bm.c:312 0x60012f78
USB_HostNotifyDevice() at usb_host_devices.c:609 0x60017c4c
USB_HostProcessCallback() at usb_host_devices.c:535 0x60004c4a
USB_HostEnumerationTransferCallback() at usb_host_devices.c:239 0x600047ce
USB_HostEhciTransactionDone() at usb_host_ehci.c:3,926 0x600192b8
USB_HostEhciTaskFunction() at usb_host_ehci.c:5,058 0x60019b2e
USB_HostTaskFn() at lwip_dhcp_usb_bm.c:137 0x6001a692
USB_HostApplicationInit() at usb_ethernetif_bm.c:360 0x60013044
USB_EthernetIfInIt() at usb_ethernetif_bm.c:382 0x600130ca
netif_add() at netif.c:388 0x6000c504
```

Figure 7. Call stack for `USB_ HostEvent ()`

- Open pipe for each interface.
  See `USB_HostCdcOpenDataInterfac()` and `USB_HostCdcOpenControlInterface()` in *usb_host_cdc.c*.
  Figure 8 shows the related call stack.



```
USB_HostOpenPipe() at usb_host_hci.c:356 0x60019e4a
USB_HostCdcOpenDataInterface() at usb_host_cdc.c:339 0x60002836
USB_HostCdcSetDataInterface() at usb_host_cdc.c:838 0x60002dbe
USB_HosCdcRndisTask() at usb_ethernetif_bm.c:844 0x60013940
USB_HostApplicationInit() at usb_ethernetif_bm.c:361 0x6001304a
USB_EthernetIfInIt() at usb_ethernetif_bm.c:382 0x600130ca
netif_add() at netif.c:388 0x6000c504
main() at lwip_dhcp_usb_bm.c:279 0x60006da4
```

Figure 8. Call stack for `USB_ HostCdcOpenDataInterfac ()`

- State machine maintenance
  For state machine maintenance, refer to `USB_HosCdcRndisTask()` and `USB_HostCdcRndisControlCallback()` in *lwip/port/usb_ethernetif_bm.c*.
  Figure 9 shows the default state machine. Adjust it to enable the AT interface and dial on AT interface.

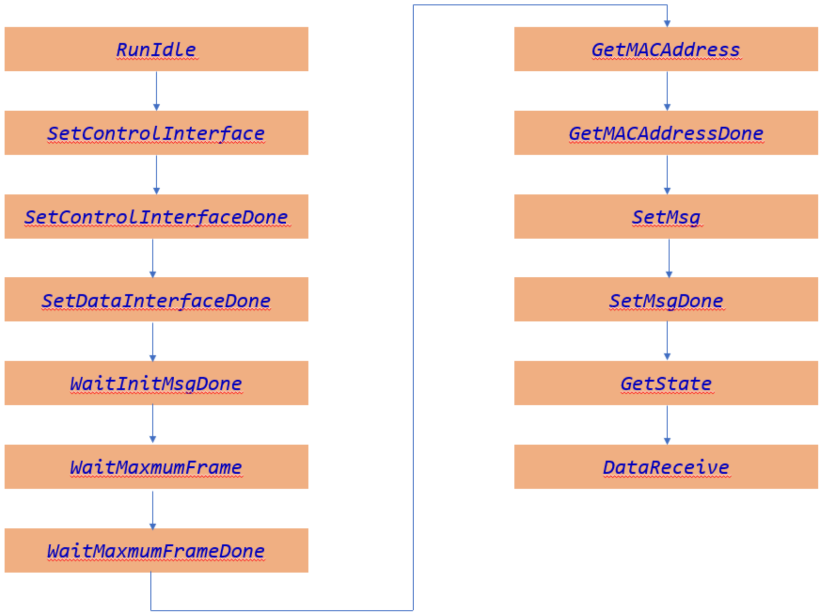


Figure 9. State machine maintenance

3 Details for implementation

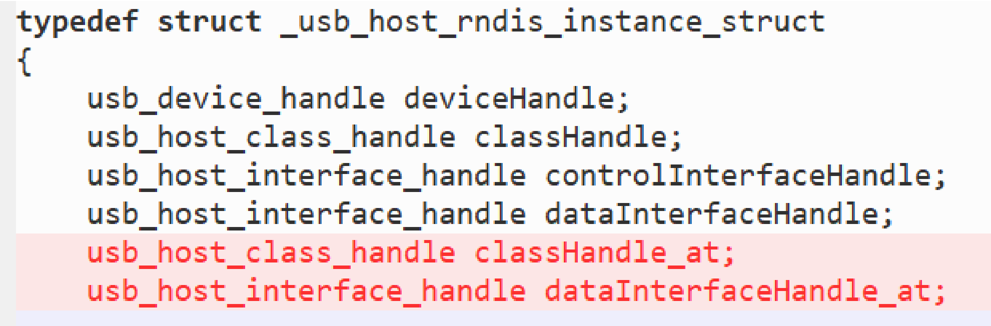
Based on the introduction above, this section discusses the implementation details. Some code is presented here. For more details, see AN13940SW.

The main code changes are in:

- *usb_ethernet_bm.c*
- *usb_ethernet.h*

Some key code changes include:

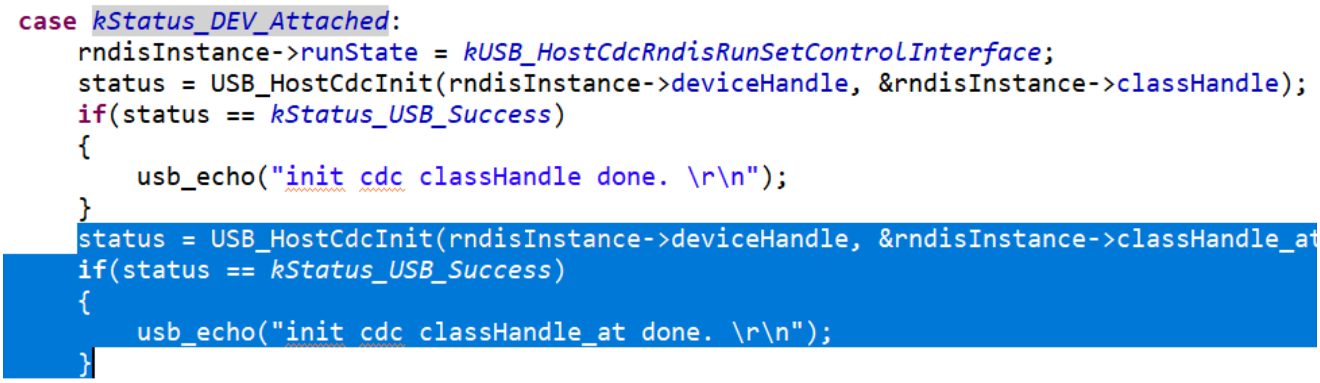
- Add the data structure for AT interface.
  It is in *sdk_root\lwip\port\usb_ethernetif.h*.



```
typedef struct _usb_host_rndis_instance_struct
{
    usb_device_handle deviceHandle;
    usb_host_class_handle classHandle;
    usb_host_interface_handle controlInterfaceHandle;
    usb_host_interface_handle dataInterfaceHandle;
    usb_host_class_handle classHandle_at;
    usb_host_interface_handle dataInterfaceHandle_at;
```

Figure 10. Add the data structure for AT interface

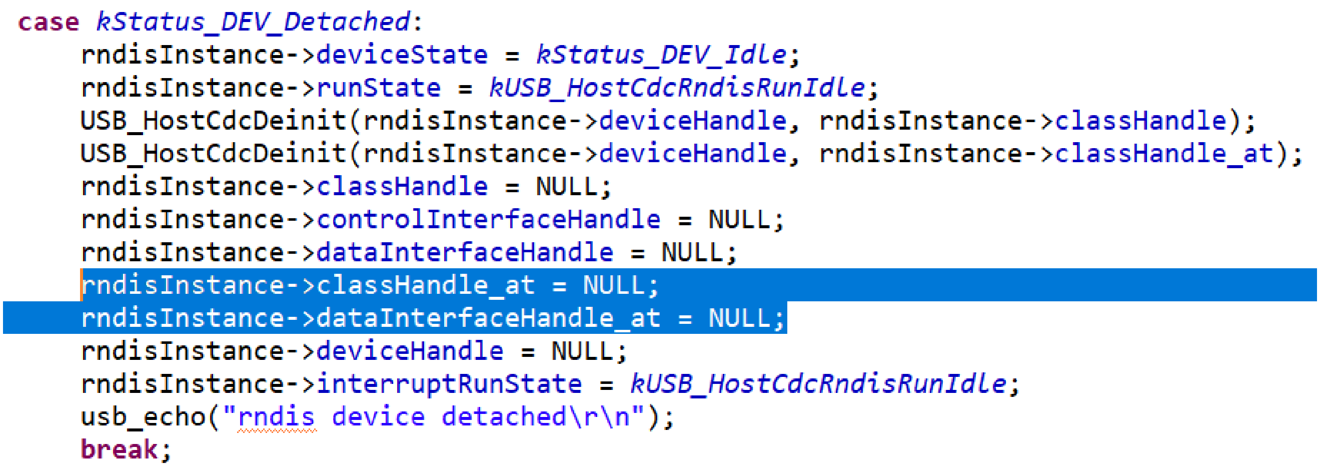
- Add class initial code in attach event.
  It is in *lwip\port\usb_ethernetif_bm.c*.



```
case kStatus_DEV_Attached:
    rndisInstance->runState = kUSB_HostCdcRndisRunSetControlInterface;
    status = USB_HostCdcInit(rndisInstance->deviceHandle, &rndisInstance->classHandle);
    if(status == kStatus_USB_Success)
    {
        usb_echo("init cdc classHandle done. \r\n");
    }
    status = USB_HostCdcInit(rndisInstance->deviceHandle, &rndisInstance->classHandle_at
    if(status == kStatus_USB_Success)
    {
        usb_echo("init cdc classHandle_at done. \r\n");
    }
```

Figure 11. Add class initial code

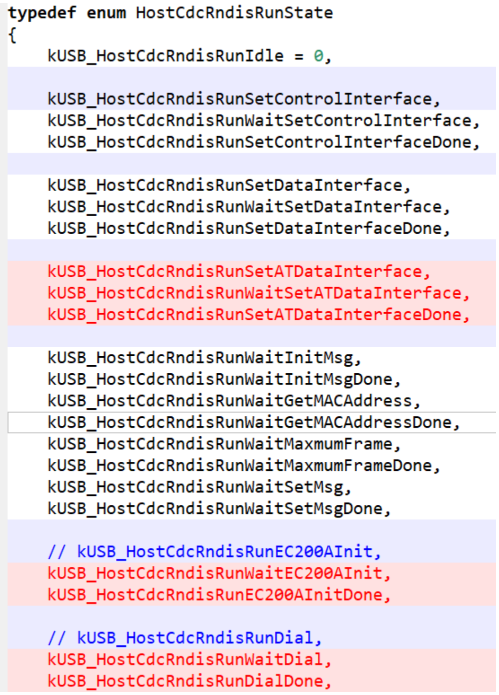
- Add de-init code in de-attach event.
  It is in *lwip\port\usb_ethernetif_bm.c*.



```
case kStatus_DEV_Detached:
    rndisInstance->deviceState = kStatus_DEV_Idle;
    rndisInstance->runState = kUSB_HostCdcRndisRunIdle;
    USB_HostCdcDeinit(rndisInstance->deviceHandle, rndisInstance->classHandle);
    USB_HostCdcDeinit(rndisInstance->deviceHandle, rndisInstance->classHandle_at);
    rndisInstance->classHandle = NULL;
    rndisInstance->controlInterfaceHandle = NULL;
    rndisInstance->dataInterfaceHandle = NULL;
    rndisInstance->classHandle_at = NULL;
    rndisInstance->dataInterfaceHandle_at = NULL;
    rndisInstance->deviceHandle = NULL;
    rndisInstance->interruptRunState = kUSB_HostCdcRndisRunIdle;
    usb_echo("rndis device detached\r\n");
    break;
```

Figure 12. Add de-init code in de-attach event

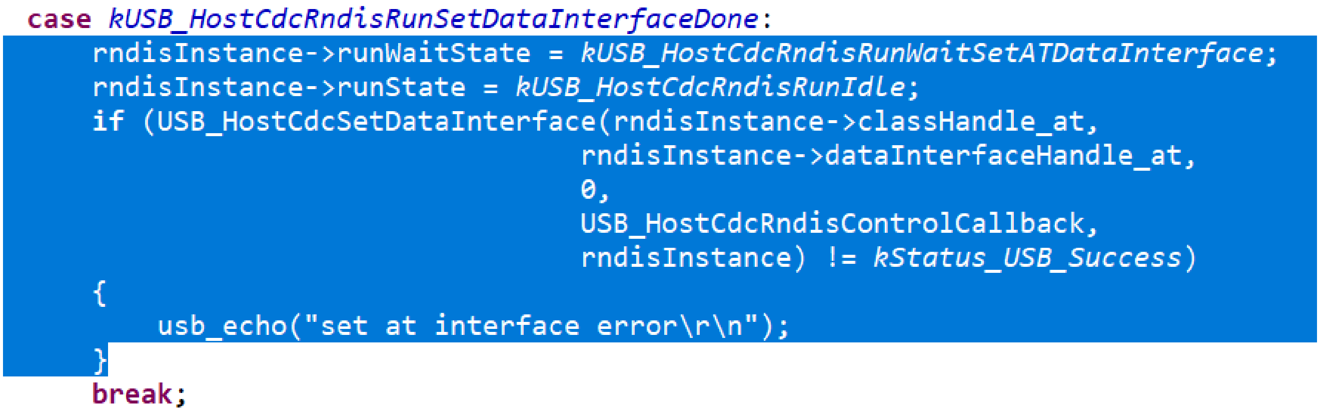
- Update the state machine `enum` variable.
  It is in *lwip\port\usb_ethernetif.h*.



```
typedef enum HostCdcRndisRunState
{
    kUSB_HostCdcRndisRunIdle = 0,

    kUSB_HostCdcRndisRunSetControlInterface,
    kUSB_HostCdcRndisRunWaitSetControlInterface,
    kUSB_HostCdcRndisRunSetControlInterfaceDone,

    kUSB_HostCdcRndisRunSetDataInterface,
    kUSB_HostCdcRndisRunWaitSetDataInterface,
    kUSB_HostCdcRndisRunSetDataInterfaceDone,

    kUSB_HostCdcRndisRunSetATDataInterface,
    kUSB_HostCdcRndisRunWaitSetATDataInterface,
    kUSB_HostCdcRndisRunSetATDataInterfaceDone,

    kUSB_HostCdcRndisRunWaitInitMsg,
    kUSB_HostCdcRndisRunWaitInitMsgDone,
    kUSB_HostCdcRndisRunWaitGetMACAddress,
    kUSB_HostCdcRndisRunWaitGetMACAddressDone,
    kUSB_HostCdcRndisRunWaitMaxmumFrame,
    kUSB_HostCdcRndisRunWaitMaxmumFrameDone,
    kUSB_HostCdcRndisRunWaitSetMsg,
    kUSB_HostCdcRndisRunWaitSetMsgDone,

    // kUSB_HostCdcRndisRunEC200AInit,
    kUSB_HostCdcRndisRunWaitEC200AInit,
    kUSB_HostCdcRndisRunEC200AInitDone,

    // kUSB_HostCdcRndisRunDial,
    kUSB_HostCdcRndisRunWaitDial,
    kUSB_HostCdcRndisRunDialDone,
```

Figure 13. Update state machine

- Update state machine implementation function.
  It is in *lwip\port\usb_ethernetif_bm.c*.



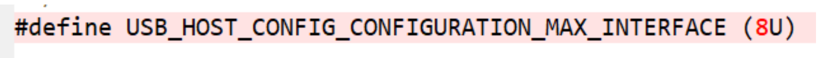
```
case kUSB_HostCdcRndisRunSetDataInterfaceDone:
    rndisInstance->runWaitState = kUSB_HostCdcRndisRunWaitSetATDataInterface;
    rndisInstance->runState = kUSB_HostCdcRndisRunIdle;
    if (USB_HostCdcSetDataInterface(rndisInstance->classHandle_at,
                                    rndisInstance->dataInterfaceHandle_at,
                                    0,
                                    USB_HostCdcRndisControlCallback,
                                    rndisInstance) != kStatus_USB_Success)
    {
        usb_echo("set at interface error\r\n");
    }
    break;
```

Figure 14. Update state machine implementation function

- Add related API and callback. The function parameter is not displayed here. For more details, see AN13940SW.

```
USB_HostCdcRndisATInCallback();
USB_HostCdcRndisATOutCallback();
dial_tx();  // Send command on AT interface
dial_rx();  // Receive message from AT interface
lte_dial(); // Dial, then USB 4G module can connect to internet
USB_HostCdcRndisEC200ACallback();
ep0_communicate(); // API used for AT interface enablement, to send/receive
 command/message to/from EP0
init_ec200a(); // Initiate EC200A, then AT interface communication is available
```

- The maximum interfaces number is supported.
  The maximum interfaces number is different when connected to different USB 4G module. The update is required.
  For example, if using the LE910C1-EU, update the value to 8.



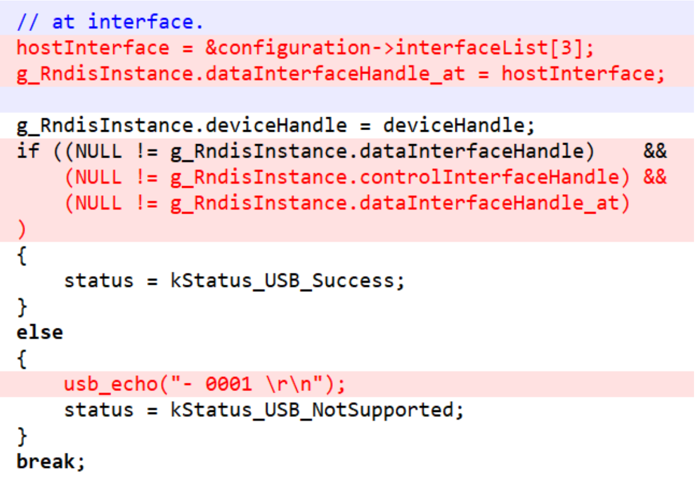
```
#define USB_HOST_CONFIG_CONFIGURATION_MAX_INTERFACE (8U)
```

Figure 15. Maximum interfaces number supported

  And for EC200A-CN used in this document, the default value, 5, can work directly.

- Implement the new state machine for AT interface enablement and dial on AT interface.
  The main code change is in function `USB_HostCdcRndisControlCallback()` and `USB_HosCdcRndisTask()` in *lwip\port\usb_ethernetif_bm.c*. For more details, see AN13940SW.

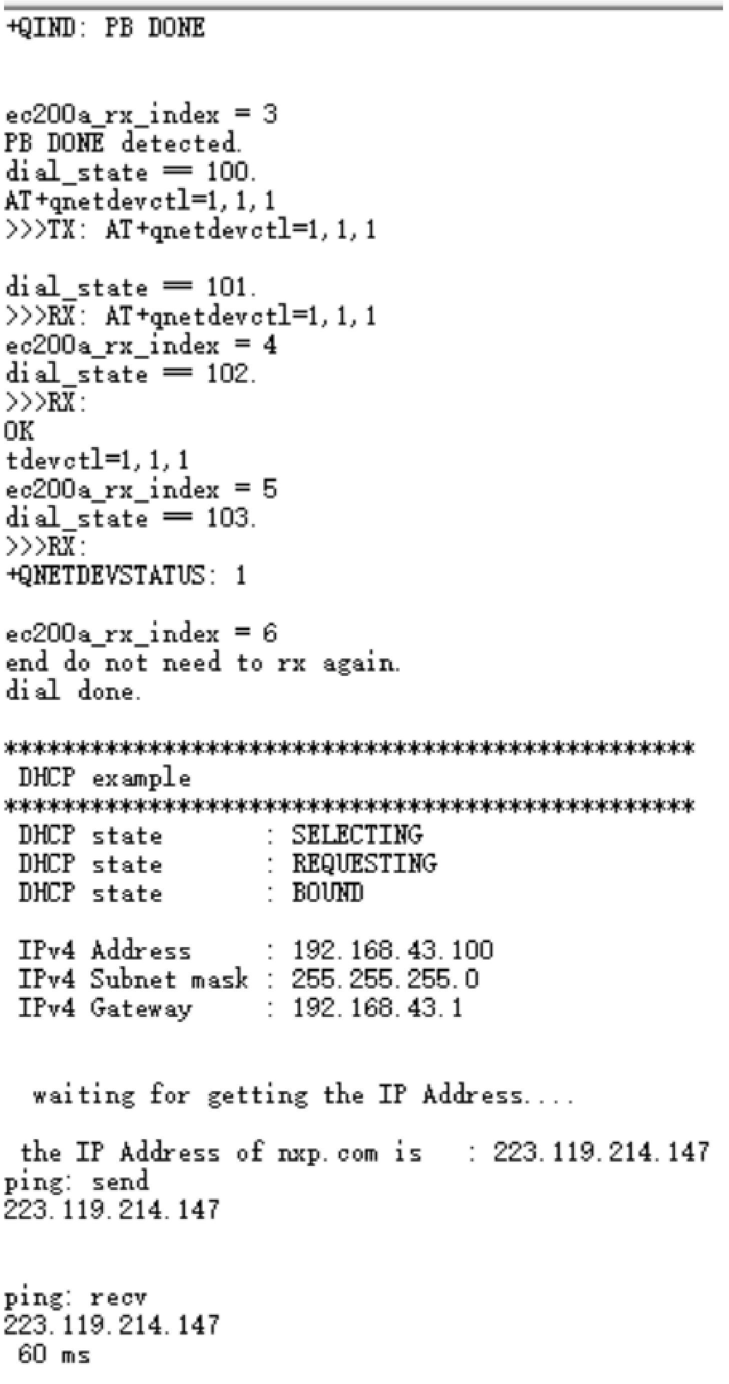
- Adapt AT interface.
  The AT data interface index is decoded from USB analyzer when connecting the USB 4G module to a PC. The code is from `USB_HostCdcRndisEvent()` in *lwip\port\usb_ethernetif_bm.c*,
  it assigns `interfaceList[3]` to the `g_RndisInstance`, `dataInterfaceHandle_at`, and implements the binding.



```
// at interface.
hostInterface = &configuration->interfaceList[3];
g_RndisInstance.dataInterfaceHandle_at = hostInterface;

g_RndisInstance.deviceHandle = deviceHandle;
if ((NULL != g_RndisInstance.dataInterfaceHandle)    &&
    (NULL != g_RndisInstance.controlInterfaceHandle) &&
    (NULL != g_RndisInstance.dataInterfaceHandle_at)
)
{
    status = kStatus_USB_Success;
}
else
{
    usb_echo("- 0001 \r\n");
    status = kStatus_USB_NotSupported;
}
break;
```

Figure 16. Bind AT interface

Run the code after all code changes, and we can see that the i.MX RT1060 EVK is connected to the Internet successfully with USB 4G module EC200A-CN.



```
+QIND: PB DONE


ec200a_rx_index = 3
PB DONE detected.
dial_state == 100.
AT+qnetdevctl=1,1,1
>>>TX: AT+qnetdevctl=1,1,1

dial_state == 101.
>>>RX: AT+qnetdevctl=1,1,1
ec200a_rx_index = 4
dial_state == 102.
>>>RX:
OK
tdevctl=1,1,1
ec200a_rx_index = 5
dial_state == 103.
>>>RX:
+QNETDEVSTATUS: 1

ec200a_rx_index = 6
end do not need to rx again.
dial done.

************************************************
 DHCP example
************************************************
 DHCP state       : SELECTING
 DHCP state       : REQUESTING
 DHCP state       : BOUND

 IPv4 Address     : 192.168.43.100
 IPv4 Subnet mask : 255.255.255.0
 IPv4 Gateway     : 192.168.43.1


  waiting for getting the IP Address....

 the IP Address of nxp.com is   : 223.119.214.147
ping: send
223.119.214.147


ping: recv
223.119.214.147
 60 ms
```

Figure 17. Connect to the Internet with USB 4G module EC200A-CN

4 Attention

Make sure that the power supply is enough for USB 4G module. In this case, the i.MX RT1060 EVK is powered with an external power adapter, instead of with a USB cable. Otherwise, the USB 4G module resets when it tries to enable the RF to connect to the Internet.

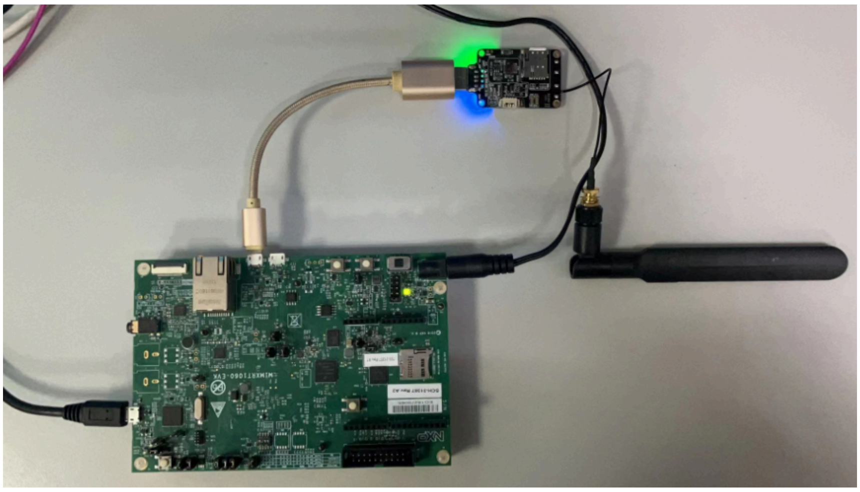


Figure 18. Power i.MX RT1060 EVK from external power adapter

5 Conclusion

This application note discusses the basic knowledge and details about how to connect i.MX RT1060 EVK to the Internet by USB 4G module EC200A-CN. It can be helpful for user reference when they want to connect to the Internet with the same or other USB 4G modules.

6 Legal information

6.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

6.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

For more information, please visit: http://www.nxp.com

Date of release: 18 April 2023
Document identifier: AN13940