|  |
| --- |
| ADI Study Watch  USB BLE Communication |
| REV 1.0.0,MAR 2021 |

Table of Contents

[1 Prerequisite 4](#_Toc66350290)

[2 USB Task 5](#_Toc66350291)

[3 BLE Task 8](#_Toc66350292)

List of Figures

[Figure 1: USB task - Receive M2M2 Request 5](#_Toc66350286)

[Figure 2: USB task - Send M2M2 Response 6](#_Toc66350287)

[Figure 3: BLE task - Receive M2M2 request 8](#_Toc66350288)

[Figure 4: BLE task - Send M2M2 response 9](#_Toc66350289)

List of Tables

No table of figures entries found.Copyright, Disclaimer & Trademark Statements

Copyright Information

Copyright (c) 2021 Analog Devices, Inc. All Rights Reserved. This documentation is proprietary and confidential to Analog Devices, Inc. and its licensors. This document may not be reproduced in any form without prior, express consent from Analog Devices, Inc.

Disclaimer

Analog Devices, Inc. (“Analog Devices”) reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent or other rights of Analog Devices

Trademark and Service Mark Notice

Analog Devices, the Analog Devices logo, Blackfin, SHARC, TigerSHARC, CrossCore, VisualDSP, VisualDSP++, EZ-KIT Lite, EZ-Extender, SigmaStudio and Collaborative are the exclusive trademarks and/or registered trademarks of Analog Devices, Inc (“Analog Devices”).

All other brand and product names are trademarks or service marks of their respective owners.

Analog Devices’ Trademarks and Service Marks may not be used without the express written consent of Analog Devices, such consent only to be provided in a separate written agreement signed by Analog Devices. Subject to the foregoing, such Trademarks and Service Marks must be used according to Analog Devices’ Trademark Usage guidelines. Any licensee wishing to use Analog Devices’ Trademarks and Service Marks must obtain and follow these guidelines for the specific marks at issue.

# Prerequisite

The steps given below expects that the reader is familiar with the Study Watch source code and source files.

# USB Task

The firmware uses CDC ACM USB class, commonly known as virtual COM port. It is with this functionality that, when Watch is connected using a USB cable, it gets enumerated as a COM port in a Windows machine.

Port can be opened and closed like a traditional serial port.

Filelocation: <InstallFolder>\study\_watch\nrf5\_sdk\_15.2.0\adi\_study\_watch\modules\system\usbd\_task.c

Two FreeRTOS tasks are defined in this file:

1. usbd\_application\_task() to handle the USBD application events, which calls the usbd library apis for init, enable, disable and stop of USBD.
2. usbd\_tx\_task() to handle actual USB write of the post office messages coming from other tasks in firmware.

Below figures shows the interaction of USB Tx task with external tools, post office task, sensor task for getting an m2m2 REQ command from the Tools and giving back an m2m2 RESP packet:

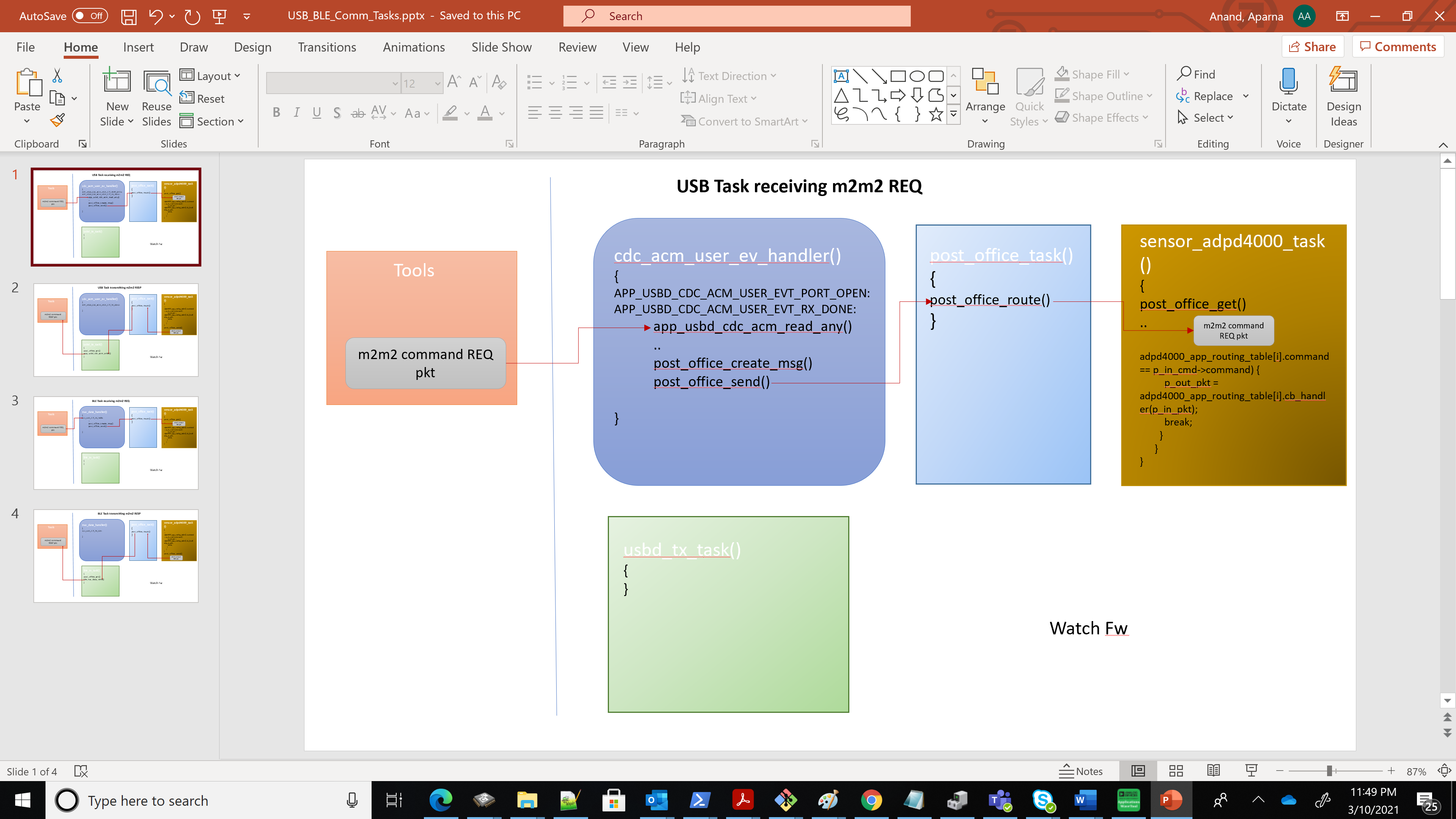


Figure 1: USB task - Receive M2M2 Request

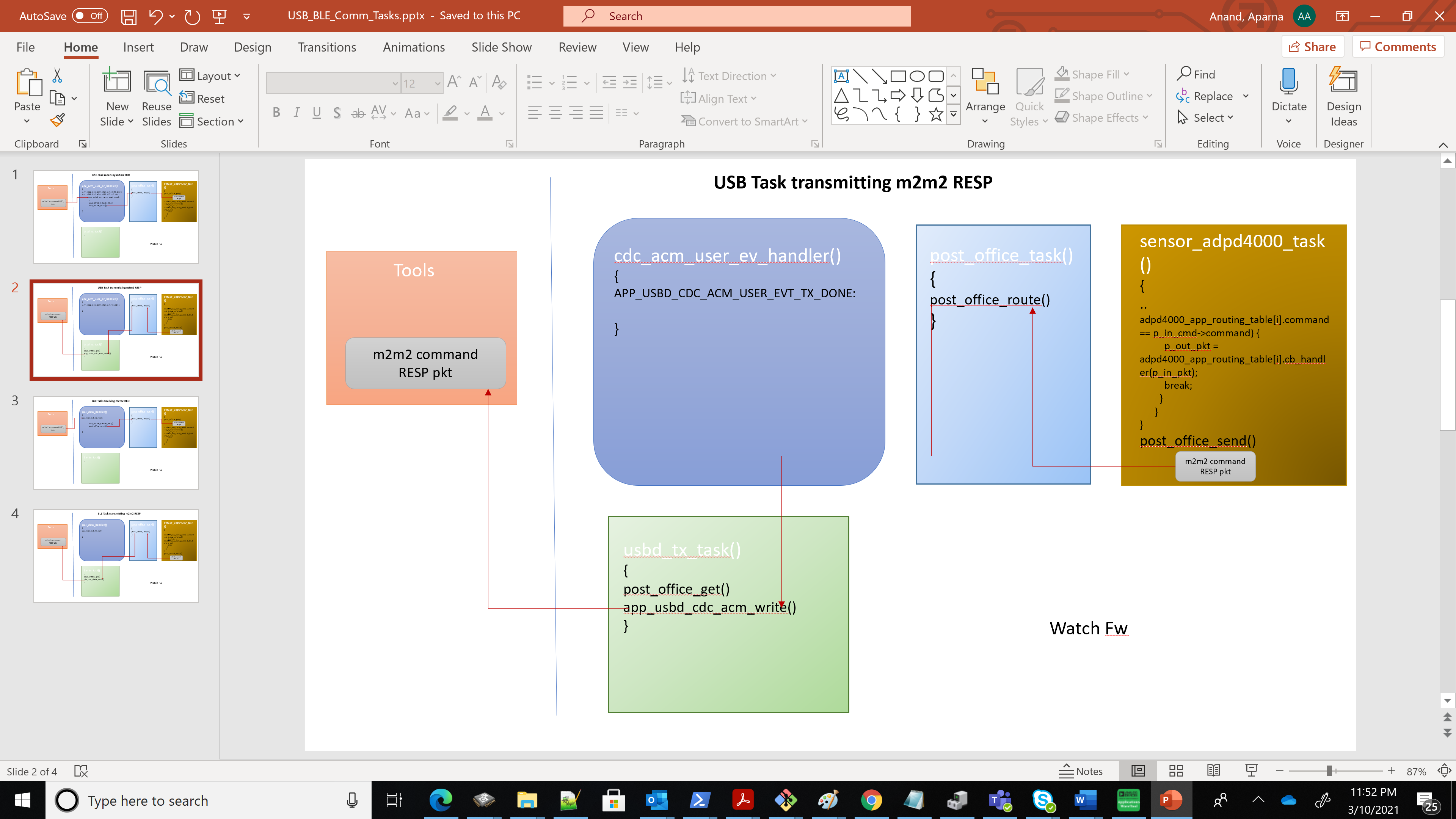


Figure 2: USB task - Send M2M2 Response

In watch firmware, USB read and write happens using the APIs from USB CDC ACM

* app\_usbd\_cdc\_acm\_read\_any() - returns data as quick as any data is available, even if the given buffer was not totally full. Return values and their meaning are the same, *but this function cannot use double buffering.*
* app\_usbd\_cdc\_acm\_write() - all command responses are given using this API from usbd\_tx\_task()

When all data is sent, event APP\_USBD\_CDC\_ACM\_USER\_EVT\_TX\_DONE is generated and a new chunk of data can be sent.

*Currently in the firmware, with USE\_USB\_READ\_ANY macro, app\_usbd\_cdc\_acm\_read\_any() is only tested and used.*

To check the number of bytes really read use app\_usbd\_cdc\_acm\_rx\_size() function.

The cdc\_acm\_user\_ev\_handler() handler registered with cdc acm class and added on the USBD driver init performs the following functions:

1. controls the state of usbd\_tx\_task() task.
2. It also calls the app\_usbd\_cdc\_acm\_read\_any() to read any more packets from the COM port withinAPP\_USBD\_CDC\_ACM\_USER\_EVT\_PORT\_OPEN and APP\_USBD\_CDC\_ACM\_USER\_EVT\_RX\_DONE events.

In short, with respect to m2m2 commands, all command requests are received using app\_usbd\_cdc\_acm\_read\_any() from cdc\_acm\_user\_ev\_handler(). All command responses are given using app\_usbd\_cdc\_acm\_write() from usbd\_tx\_task().

Since app\_usbd\_cdc\_acm\_read\_any() is based on returning the data as quickly as possible, for m2m2 commands REQ, packet is completely formed in two steps:

* reading the header to get the total size of the packet,
* reading the remaining packet content, if its greater than 64 bytes. USB CDC ACM uses the bulk transfer with endpoint size as 64.

Once the expected length as pointed in the header is received, from the cdc\_acm\_user\_ev\_handler() a m2m2 packet in formed using post\_office\_create\_msg() and sent to the PostOffice using post\_office\_send()

**References from nRF 15.2.0 SDK infocentre**:

USBD HAL and Driver:

<https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.2.0/group__nrf__drv__usbd.html>

USB device Library:

<https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v15.2.0%2Flib_usbd.html&cp=7_5_2_3_59>

USB CDC ACM module:

<https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.2.0/lib_usbd_class_cdc.html>

<https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.2.0/group__app__usbd__cdc__acm.html>

# BLE Task

The study watch uses the SoftDevice – s140 from Nordic SDK, which is a precompiled and linked binary image implementing Bluetooth 5 Low Energy protocol stack for nRF52 series of SoCs. The Watch application uses the serial port emulation over BLE and includes the Nordic UART Service (NUS).

File location: <InstallFolder>\study\_watch\nrf5\_sdk\_15.2.0\adi\_study\_watch\modules\system\ble\_task.c

FreeRTOS tasks are defined in this file:

1. ble\_tx\_task() – To handle Tx from Watch over BLE NUS

Below figures shows the interaction of BLE Tx task with external tools, post office task, sensor task for getting an m2m2 REQ command from the Tools and giving back an m2m2 RESP packet:

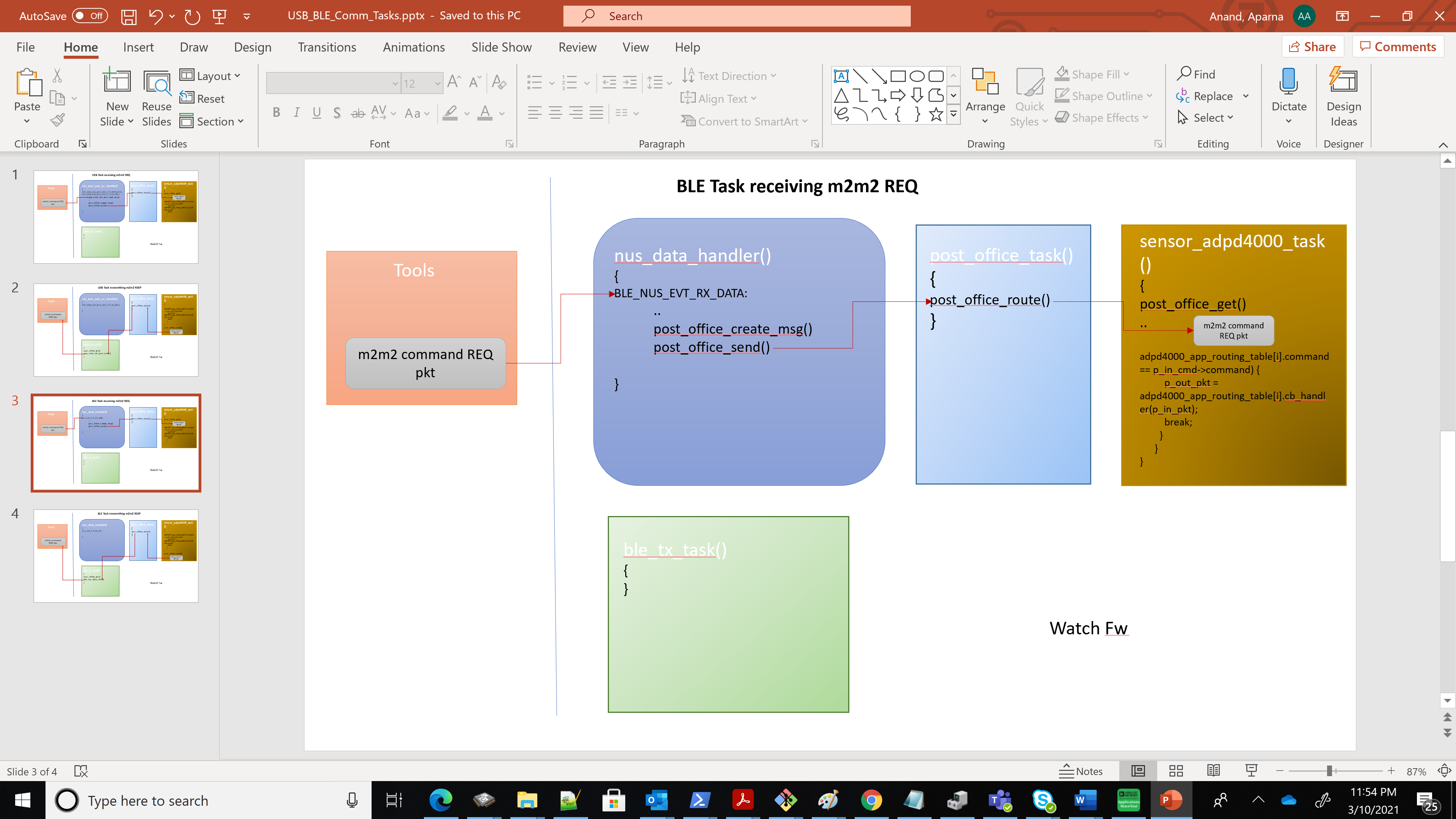


Figure 3: BLE task - Receive M2M2 request

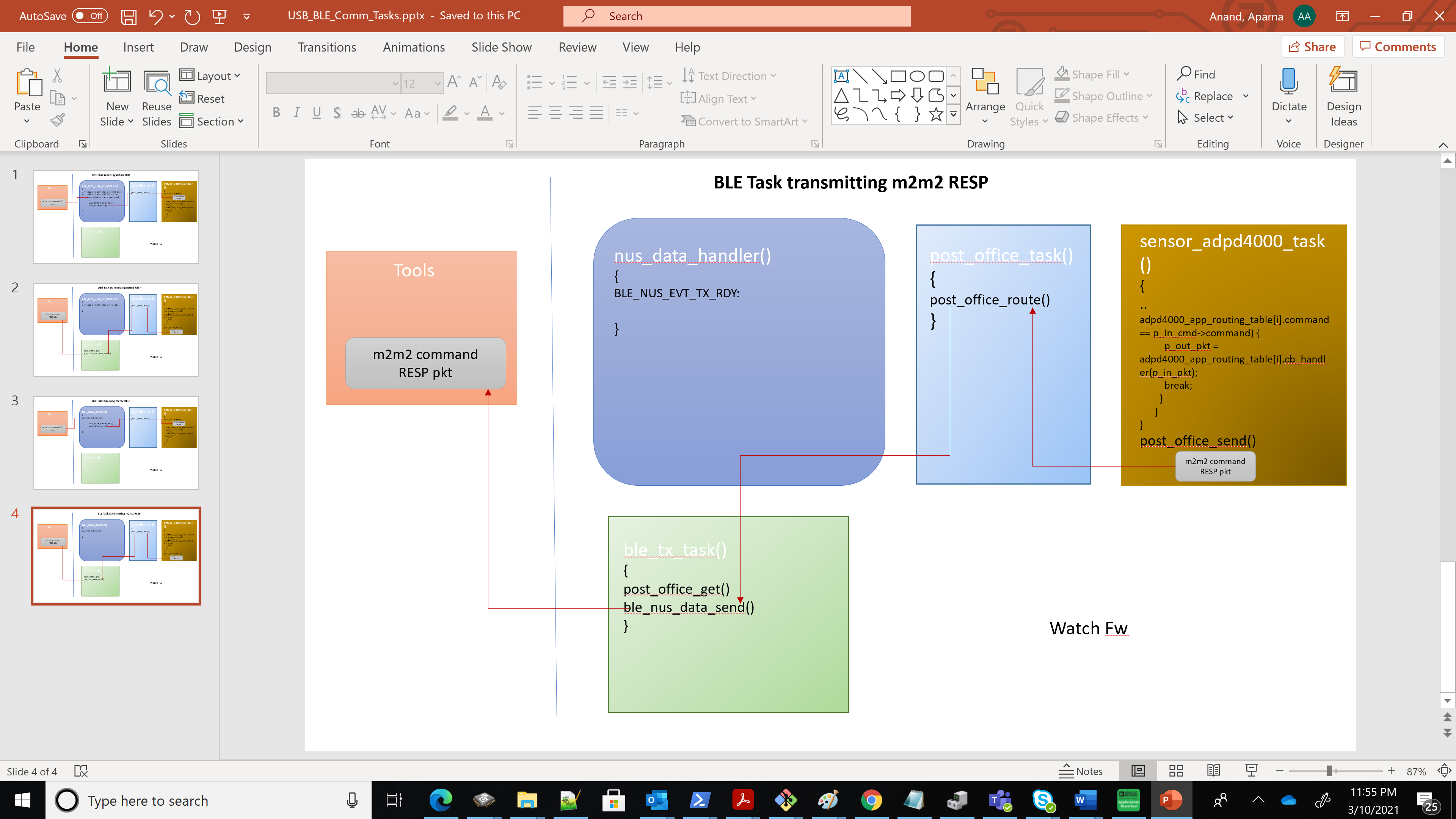


Figure 4: BLE task - Send M2M2 response

In accordance with BLE NUS, BLE\_NUS\_MAX\_DATA\_LEN = 244. This is being used in USB task also, just to maintain similarity for maximum data packet length over both USB & BLE interface.

From the watch firmware, BLE read happens at the nus\_data\_handler() and BLE\_NUS\_EVT\_RX\_DATA event type. BLE write happens using ble\_nus\_data\_send() from ble\_tx\_task()

When all data is sent, event BLE\_NUS\_EVT\_TX\_RDY event is generated and a new chunk of data can be sent.

nus\_data\_handler() handler registered with BLE NUS service added on the services\_init(), controls the state of ble\_tx\_task() task, to awaken it with BLE\_NUS\_EVT\_COMM\_STARTED event. At BLE\_NUS\_EVT\_RX\_DATA event, once the expected length as pointed in the header is received, an m2m2 packet in formed using post\_office\_create\_msg() and sent to the PostOffice using post\_office\_send()

In short, with respect to m2m2 commands, all command requests are received at nus\_data\_handler() and BLE\_NUS\_EVT\_RX\_DATA event type. All command responses are given using ble\_nus\_data\_send() from ble\_tx\_task().

In ble\_tx\_task(), m2m2 packet combining is enabled upon entering the high data rate mode. High data rate mode is decided when the first STREAM\_SUBSCRIBE response is detected within the m2m2 packet to be sent out. High data rate mode exit happens when the last STREAM\_UNSUBSCRIBE response from the m2m2 packet to be sent out is received.

Currently, MAX\_TX\_PKT\_COMB\_CNT is 4 in high data rate mode and MIN\_TX\_PKT\_COMB\_CNT is 1 otherwise. The idea of packet combining is to make sure that the buffer submitted to ble\_nus\_data\_send() api is used maximally, otherwise NRF\_ERROR\_RESOURCES was seen.

It is ensured that BLE data buffer submitted to ble\_nus\_data\_send() is as close as possible to BLE NUS characteristic length supported, (BLE\_NUS\_MAX\_DATA\_LEN = 244) during sensor data streaming mode over BLE.

!Exception to use ble\_nus\_data\_send() immediately is when it is an m2m2 REQ-RESP packet that need to go out immediately(command != M2M2\_SENSOR\_COMMON\_CMD\_STREAM\_DATA) or when length of data in the ble data buffer is about to exceed, BLE\_NUS\_MAX\_DATA\_LEN = 244, even before MAX\_TX\_PKT\_COMB\_CNT

To improve the performance throughput of the BLE connection and characterize the stable data rate, following parameters were changed in Study Watch firmware:

* BLE GAP data length: 251 (earlier it was 27) The SoftDevice handler will configure the stack with these parameters
* BLE Tx and Rx PHY: BLE\_GAP\_PHY\_2MBPS (earlier it was BLE\_GAP\_PHY\_AUTO)
* BLE Connection Interval: 7.5ms (earlier min and max connection interval was 10 and 40 ms). Now both min and max are set to 7.5ms and 35 respectively.
* BLE\_COMMON\_OPT\_CONN\_EVT\_EXT option set to one, which is the parameter for enabling extended connection events

**References from nRF 15.2.0 SDK infocentre**:

**Softdevice Handler Library:**

<https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.2.0/lib_softdevice_handler.html>

**Serial Port Emulation Over BLE Example from nRF SDK:**

<https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.2.0/ble_sdk_app_nus_eval.html>

**NUS with USB CDC with FreeRTOS Nordic example may be referred from:**

<InstallFolder>\study\_watch\nrf5\_sdk\_15.2.0\examples\peripheral\usbd\_ble\_uart\_freertos