

# Q1

Follow below command to send then found there is an error in below API.

```
dtm_cmd_send(0x0208); //Set PHY to 2M
dtm_cmd_send(0x880B); //Set TX power to 8dBm (980B = -40dBm)
dtm_cmd_send(0x8094); //Transmitter Test with channel 0, length 37, PRBS9
dtm_cmd_send(0xC000); //Test End
```

```
dtm_cmd_send(0x0204); //Set PHY to 1M
```

Due to there is no  
`m_tx_power == RADIO_TXPOWER_TXPOWER_Pos8dBm`

```
uint32_t dtm_radio_validate(int32_t m_tx_power, uint8_t m_radio_mode)
{
    // Initializing code below is quite generic - for BLE, the values are fixed
    // are constant. Non-constant values are essentially set in radio_prepare()
    if (!(m_tx_power == RADIO_TXPOWER_TXPOWER_0dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos4dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg30dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg20dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg16dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg12dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg8dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg4dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos3dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg40dBm
        ) ||
        !(
#ifdef NRF52840_XXAA
            m_radio_mode == RADIO_MODE_MODE_Ble_LR125Kbit ||
            m_radio_mode == RADIO_MODE_MODE_Ble_LR500Kbit ||
#endif //NRF52840_XXAA
            m_radio_mode == RADIO_MODE_MODE_Ble_1Mbit ||
            m_radio_mode == RADIO_MODE_MODE_Ble_2Mbit
        )
    )
    {
        return DTM_ERROR_ILLEGAL_CONFIGURATION;
    }
}
```

# Q2

Follow below command to send then found there is an error in below API.

```
dtm_cmd_send(0x0210); //Set PHY to 500K
dtm_cmd_send(0x880B); //Set TX power to 8dBm
dtm_cmd_send(0x8094); //Transmitter Test with channel 0, length 37, PRBS9
```

When set PHY 500K, the `m_radio_mode = RADIO_MODE_MODE_Ble_LR500Kbit`.

When set 0x880B (set TX power), there is no judgment to check

`m_radio_mode == RADIO_MODE_MODE_Ble_LR500Kbit`. It will not run `m_packet_type = DTM_PKT_VENDORSPECIFIC`;

```
uint32_t dtm_cmd(dtm_cmd_t cmd, dtm_freq_t freq, uint32_t length, dtm_pkt_type_t payload)
{
    // Save specified packet in static variable for tx/rx functions to use.
    // Note that BLE conformance testers always use full length packets.
    m_packet_length = (m_packet_length & 0xC0) | ((uint8_t)length & 0x3F);
    m_packet_type    = payload;
    m_phys_ch        = freq;

    // If 1 Mbit or 2 Mbit radio mode is in use check for Vendor Specific payload.
    if ((m_radio_mode == RADIO_MODE_MODE_Ble_1Mbit || m_radio_mode == RADIO_MODE_MODE_Ble_2Mbit) && payload == (
    {
        /* Note that in a HCI adaption layer, as well as in the DTM PDU format,
         the value 0x03 is a distinct bit pattern (PRBS15). Even though BLE does not
         support PRBS15, this implementation re-maps 0x03 to DTM_PKT_VENDORSPECIFIC,
         to avoid the risk of confusion, should the code be extended to greater coverage.
        */
        m_packet_type = DTM_PKT_VENDORSPECIFIC;
    }
}
```

```
// ...
```

# Q2

Code will run case DTM\_PKT\_0XFF not case DTM\_PKT\_VENDORSPECIFIC

```
case DTM_PKT_0XFF:  
    // Bit pattern 11111111 repeated. Only available in coded PHY (Long range).  
    memset(m_pdu.content + DTM_HEADER_SIZE, RFFHY_TEST_0XFF_REF_PATTERN, m_packet_length);  
    break;  
  
case DTM_PKT_VENDORSPECIFIC:  
    // The length field is for indicating the vendor specific command to execute.  
    // The frequency field is used for vendor specific options to the command.  
    return dtm_vendor_specific_pkt(length, freq);
```

When send dtm\_cmd\_send(0x8094); it will show error in dtm\_cmd () below code because m\_state is not in STATE\_IDLE

```
if (m_state != STATE_IDLE)  
{  
    // Sequencing error - only TEST_END/RESET are legal while test is running  
    // Note: State is unchanged; ongoing test not affected  
    m_event = LE_TEST_STATUS_EVENT_ERROR;  
    return DTM_ERROR_INVALID_STATE;  
}
```

# Q3

Follow below command to send then found there is an error in below API.

```
dtm_cmd_send(0x0210); //Set PHY to 500K
dtm_cmd_send(0x980B); //Set TX power to -40dBm
dtm_cmd_send(0x8094); //Transmitter Test with channel 0, length 37, PRBS9
dtm_cmd_send(0xC000); //Test End
```

When set dtm\_cmd\_send(0x980B); it will get  
new\_power8 is 0xD8 but m\_tx\_power is 0xFFFFFD8

```

    This function may be called dire
    DTM_PKT_VENDORSPECIFIC as payloa
*/
bool dtm_set_txpower(uint32_t new_tx_power
){
    // radio->TXPOWER register is 32 bits,
    int8_t new_power8 = (int8_t)(new_tx_po

    // The two most significant bits are n
    // These two bits are 1's if and only
    // All valid negative values have the
    // All valid positive values have the
    // By checking this bit, the two most
    new_power8 = (new_power8 & 0x30) != 0

    if (m_state > STATE_IDLE)
    {
        // radio must be idle to change th
        return false;
    }

    m_tx_power = new_power8;

    return true;
}

```

Name	Value	Type
m_tx_power	0xFFFFFD8	int
new_power8	0xD8 '?'	signed ...
<Enter expression>		

## Q3

After set command `dtm_cmd_send(0x0208)`; again, it will entry `dtm_radio_validate()` and show an error because `m_tx_power` is `0xFFFFFFFFD8` not `0xD8` (`RADIO_TXPOWER_TXPOWER_Neg40dBm`)

```
uint32_t dtm_radio_validate(int32_t m_tx_power, uint8_t m_radio_mode)
{
    // Initializing code below is quite generic - for BLE, the values
    // are constant. Non-constant values are essentially set in radio_
    if (!(m_tx_power == RADIO_TXPOWER_TXPOWER_0dBm ||
#ifdef NRF52840_XXAA //20171108
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos2dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos5dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos6dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos7dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos8dBm ||
#endif
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos4dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg30dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg20dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg16dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg12dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg8dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg4dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Pos3dBm ||
        m_tx_power == RADIO_TXPOWER_TXPOWER_Neg40dBm
    ) ||

        !(
#ifdef NRF52840_XXAA
            m_radio_mode == RADIO_MODE_MODE_Ble_LR125Kbit ||
```

# Q3

Need to modify it to `m_tx_power = (int32_t)(new_power8 & 0xff);`

```
bool dtm_set_txpower(uint32_t new_tx_power)
{
    // radio->TXPOWER register is 32 bits, low octet a signed value
    int8_t new_power8 = (int8_t)(new_tx_power & 0xFF);

    // The two most significant bits are not sent in the 6 bit field
    // These two bits are 1's if and only if the tx_power is a negative value
    // All valid negative values have the fourth most significant bit set
    // All valid positive values have the fourth most significant bit clear
    // By checking this bit, the two most significant bits can be corrected
    new_power8 = (new_power8 & 0x30) != 0 ? (new_power8 | 0xC0) : (new_power8 & 0x30);

    if (m_state > STATE_IDLE)
    {
        // radio must be idle to change the tx power
        return false;
    }

    //m_tx_power = new_power8;
    m_tx_power = (int32_t)(new_power8 & 0xff);

    return true;
}
```

# Q4

Follow below command to send then found there is an error in below API.

```
dtm_cmd_send(0x0204); //Set PHY to 1M
dtm_cmd_send(0x880B); //Set TX power to 8dBm
dtm_cmd_send(0x8097); //Set Coded PHY
```

When send 0x980B, DTM code will run case DTM\_PKT\_VENDORSPECIFIC.

When send 0x8097, DTM code will also run case DTM\_PKT\_VENDORSPECIFIC.

How to check to run DTM\_PKT\_VENDORSPECIFIC/ DTM\_PKT\_0XFF when PKT is '11'

Test Setup and Test End commands														
	CMD		Control				Parameter				DC			
	0	0	0	0	0	0	1	0	0	0	0	1	0	0
Transmitter Test and Receiver Test commands														
	CMD		Frequency				Length				PKT			
T	1	0	0	0	0	0	0	0	1	0	0	1	0	1
R	0	1	0	0	0	0	0	0	1	0	0	1	0	0
Vendor														
	CMD		Vendor_Option				Vendor Cmd				PKT			
	1	0	0	1	1	0	0	0	0	0	0	1	0	1
0204	0	0	0	0	0	0	1	0	0	0	0	0	1	0
980B	1	0	0	1	1	0	0	0	0	0	0	0	1	1
8094	1	0	0	0	0	0	0	0	1	0	0	1	0	0
C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0
020C	0	0	0	0	0	0	1	0	0	0	0	0	1	0
0210	0	0	0	0	0	0	1	0	0	0	0	1	0	0
8097	1	0	0	0	0	0	0	0	1	0	0	1	0	1

  

0204	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	Test Setup PHY set to 1M
980B	1	0	0	1	1	0	0	0	0	0	0	0	1	1	1	Vendor Spec : TX_Power = -40dBm
8094	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	Transmit : 2402 + PRBS9 + 0x25
C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Test End
020C	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	Test Setup PHY set to 125K
0210	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	Test Setup PHY set to 500K
8097	1	0	0	0	0	0	0	0	1	0	0	1	0	1	1	Coded PHY