Bluegiga Forums / Community Forums / Bluetooth Smart

How to check if received byte over UART equals to...?

Answered

Mairo Leier
asked this on May 12, 2014, 15:24

Share Tweet 0 Like 0

If external MCU send data over UART to BLE112 and before I decide which service this data belongs to, I would like to check if first byte equals to, for example, 0x06. Currently it looks like this:

```
# Receiving data to UART and forwarding it to Bluetooth connection
event system_endpoint_watermark_rx(endpoint, size)
    if endpoint = system_endpoint_uart1 then
        in_len = size
        if in_len > 20 then          # Read maximum of 20 bytes
            in_len = 20
        end if

        # disable RX watermark
        call system_endpoint_set_watermarks(system_endpoint_uart1, 0, $ff)

        # Read data from UART
        call system_endpoint_rx(system_endpoint_uart1, in_len)(result, in_len, in(0:in_len))

        if in(0:1) = $06 then
            # Write data to GATT
            call attributes_write(xgatt_data, 0, in_len-1, in(1:in_len))
        else
            # Write data to GATT
            call attributes_write(xgatt_data, 0, 3, "ERR")
        end if

        # set RX watermarkto 10 bytes. In case of indications there is no need to enable watermarks
        call system_endpoint_set_watermarks(system_endpoint_uart1, 8, $ff)

        # write data back out the UART1
        call system_endpoint_tx(system_endpoint_uart1, 2, "\x06\x0d")
    end if
end
```

It just sends me ERR even I check that from external MCU 0x06 is sent out. Looks like I am doing something wrong but so foar haven't found any good example how to compare bytes.

0 people would like this to be answered. Be the first!

Comments

Jeff Rowberg
Bluegiga Technologies

Hello Mairo,

If you want to read only one byte from the UART buffer, then this line of code will not work:

```
# Read data from UART
call system_endpoint_rx(system_endpoint_uart1, in_len)(result, in_len, in(0:in_len))
```

The RX watermark event will trigger when you have ***at least*** the requested number of bytes available in the buffer, but there could be many more than that. You should set the RX watermark to 1 byte first, and then read 1 byte with this code:

```
# Read data from UART
call system_endpoint_rx(system_endpoint_uart1, 1)(result, in_len, in(0:in_len))
```

From that point, you can then re-set the RX watermark to the desired number of bytes. There is an example project which implements something like this here:

- https://bluegiga.zendesk.com/entries/28461493--BGScript-uart-echo-packet-UART1-Alt1-loopback-local-echo-with-watermarking-and-packetization

Support

May 12, 2014, 22:35

Mairo Leier

If I have for example two or more different type data (with equal packet size of 20 bytes) that needs to be received from UART and depending on the fist byte I need to decide which BLE service is sending that particular received data. How to handle this situation?

I see some ways:

1) Use if-then statement as in my example above BUT it would not work as I understand? It would be in my viewpoint the best way to define which service handles which type of data.

2) Send everything inside one BLE service and sort it out later in the receiving node. In this case I can't get any advantage of BLE different type of services and decide which type of services should be active at any particular time. As as some type of data needs max throughput and using only one service limits possible link speed.

3) Any other idea?

May 13, 2014, 12:15

Mairo Leier

There is actually one more reason why I need to check first byte.

If I receive data over UART, they are sent as a bunch once per second in particular order. I am usin UART speed 38400bps and no sleep. Even data is sent in correct order to BLE (monitoring with logic analyzer) sometimes they will be sent in a little different order over BLE to the received. I guess some bytes may be missing time to time. If I would be able to check the value of first byte I could decide whether to send this data out or skip that byte, check next one and so on until there is correct byte that means the start of my packet.

May 13, 2014, 12:43

Mairo Leier

I tried to lower UART speed from 115200 to 38400 and 9600 but still missing bytes.

I also enabled flow control only in one side and logic analyzer show that in case RTS is high, next byte won't be sent before line is low again.

I also enabled both sides two stop bits.

It is getting better but still sometimes buffer gets shifted. If I send fox example "12345" to BLE then sometimes I will get back to external MCU "45123". There will be no missing bytes but buffer is shifted for some reason.

Big question is: **How can I reset receive buffer or delete all data in that buffer after they are received from UART and sent over BLE and I am ready to receive new data?**

```
# disable RX watermark
call system_endpoint_set_watermarks(system_endpoint_uart1, 0, $ff)

# Read data from UART
call system_endpoint_rx(system_endpoint_uart1, in_len)(result, in_len, in(0:in_len))

# write data back out the UART1
call system_endpoint_tx(system_endpoint_uart1, in_len, in(0:in_len))

# Write data to GATT
call attributes_write(xgatt_data, 0, in_len, in(0:in_len))
```

# Here I would like to delete all data in "IN" buffer and set length to 0

May 15, 2014, 10:59

Jeff Rowberg
Bluegiga Technologies

Hi Mairo,

The best way that I have been told to flush the buffer though is by disabling the UART and then enabling it again. This can be done with the following code which uses the undocumented `system_reg_write()` call.

Disable UART receiver before using pin for something else, register U1CSR:

```
call system_reg_write($70f8, $80)
```

...and then enable again when desired:

```
call system_reg_write($70fb, $c0)
```

Note that these addresses are specific to UART1. You can do the same thing with UART0, but the register addresses are different. The UxCSR registers are documented in detail in the CC2540 User Guide.

May 15, 2014, 18:28

Mairo Leier

I tried to put those lines at the very end of *event system_endpoint_watermark_rx(endpoint, size)* but it started to cause problems and even RTS line didn't go low and nothing worked. As the UxCSR register contains other configurations also then maybe it would work if I read the status of the bit and restore it after disabling UART.

Also my first question is unanswered. If I have different type of data received over UART and first two bytes define which BLE service is sending this data. I can not rely on packet length because sending one byte packet before actual one causes unneeded traffic and if there is packet loss in the next packet then it would not work either. So, I need to check the value of first bytes of received packet but how?

May 16, 2014, 09:39

Jeff Rowberg
Bluegiga
Technologies

Hi Mairo,

You are correct about the UART config register containing other configuration bits. Reading it with "**system_reg_read(addr)(addr, value)**" first and rewriting it may help. I forgot you were using two stop bits and possibly no flow control.

As for your other question:

> So, I need to check the value of first bytes of received packet but how?

This can be done with a simple "if" statement:

```
if in(0:1) = $31 && in(1:1) = $32 then
```

For example, the above line will match if the first byte in the buffer is '1' (0x31) and the second byte in the buffer is '2' (0x32).

May 16, 2014, 16:30

Mairo Leier

I don't get it. "in_len" should mean the length of array not data itself?

If I put your code here, it won't work and all I get is: "Error, expecting buffer or function in line 143 "if in_len(0:1) = $06 && in_len(1:1) = $01 then" in_len is not declared".

If I try to change "in_len" to "in" then no error but still it won't work. First 2 bytes are always in my case 0x06 and 0x01.

```
[code]

# Receiving data to UART and forwarding it to Bluetooth connection
event system_endpoint_watermark_rx(endpoint, size)
  if endpoint = system_endpoint_uart1 then
    in_len = size
    if in_len > 20 then        # Read maximum of 20 bytes
      in_len = data_in_len
    end if

    # disable RX watermark
    call system_endpoint_set_watermarks(system_endpoint_uart1, 0, $ff)

    # Read data from UART
    call system_endpoint_rx(system_endpoint_uart1, in_len)(result, in_len, in(0:in_len))

    if in_len(0:1) = $06 && in_len(1:1) = $01 then
      # write data back out the UART1
      #call system_endpoint_tx(system_endpoint_uart1, 2, in(0:2))
      call system_endpoint_tx(system_endpoint_uart1, 3, "\x06\x01\x0d")

      # Write data to GATT
      call attributes_write(xgatt_data, 0, data_in_len, in(0:data_in_len))
    end if

    # set RX watermarkto n bytes. In case of indications there is no need to enable watermarks
    call system_endpoint_set_watermarks(system_endpoint_uart1, data_in_len, $ff)

  end if
end

[/code]
```

May 19, 2014, 13:00

Please answer me! It's not easy to postpone many related tasks several days...

May 20, 2014, 16:24

Mairo Leier

Jeff Rowberg
Bluegiga Technologies

Hello Mairo,

I apologize for the delay; we have a high volume of support questions to process and are not always able to respond as quickly as we like to.

You are correct about the usage of **in** vs. **in_len**, that was an oversight on my part. The two bytes should have been compared against **in**, the actual buffer, rather than **in_len**. I have modified my previous comment to reflect the correct syntax to avoid confusion for others. However, that *is* the correct syntax to verify the first two bytes of the incoming buffer. If the condition doesn't match, then the first two bytes of the buffer are not actually what is being tested against. If this is the case, then either (1) the data coming into the module over UART is not what you expect, or (2) the data is being read, buffered, and processed in your BGScript code in a way that results in unexpected behavior.

The code that you have copied above will check to make sure the first 2 bytes are **0x06** and **0x01**, and then read as much data is available up to 20 bytes, and then re-set the RX watermark to **data_in_len** bytes. The value contained in this variable is unknown based on the code you have attached, and it is possible that this kind of data flow is not what you want. If you are reading data in and the buffer is getting shifted sometimes, then you are most likely reading too many bytes at once (e.g. reading **in_len** bytes when you should be reading only **2** bytes).

If you will always be sending 20-byte packets, then it is better to set the RX watermark to 20 bytes, and then when the RX watermark event occurs, read exactly 20 bytes (not **size** bytes or **in_len** bytes) and process them all as a single block.

May 20, 2014, 16:40

Mairo Leier

Thanks, byte check is working now.

There are actually two things that frustrate.

1. I am using BLE112 sleep mode 2 and wakeup pin. If module is in sleep and I enable wakeup pin, then send data, around 20-40% of time I gan't get any answer back from BLE module after sending packets (for each packet I wait for 3 bytes from BLE). Only way that solved my problem is after enabling wakeup pin, ther has to be wait time at least 700us before sending actual data or some bytes will be lost time to time. As much I remember this is not documented.

2. I am using also RTS pin on BLE112 to wait until I could send next byte to BLE module. In case I send it immediately after RTS is low again, there will be lost bytes time to time (not so frequently as in point 1). I have made a ticket also about that. Making delay ca 0,4us after RTS is low will solve the problem. Anyway this is also a workaround that wasted 2 days to figure it out.

May 20, 2014, 17:03

Mairo Leier

In general, we can make this thread answered now after you have added some comments about my last post :)

Answer

May 20, 2014, 17:04

Mairo Leier

Add a comment

Save comment