Bluegiga Forums / Community Forums / Bluetooth Smart

BLE113 controlled by external microcontroller over UART

Answered

Simon Deleersnyder
asked this on November 2, 2014, 20:04

I'm using an external microcontroller (Atmel Atmega328p) to send data over UART (4800 baud) to a BLE113 module but it's not working (after trying for the better part of a week). I can control the module with BLEGUI and the BLED112 dongle but I haven't been able to get it working with my external microcontroller.

The problem seems to be that the module doesn't get any of the data I send it, or doesn't respond to it. I want to turn on an LED connected to port P1_0 on the BLE113 module, which means I have to send it the following commands:

ble_cmd_hardware_io_port_config_direction(1,1);

ble_cmd_hardware_io_port_write(1,1,1);

Or equivalently (using the BGAPI reference manual):

06 00 02 07 03 01 01

07 00 03 07 06 01 01 01

Note that I'm sending the packet length first since I put the BLE113 in packet mode.

I know that my read_api_packet is not working at the moment but the send_api_packet does work and is all I need for this, right? When I connect my microcontroller to another microcontroller I receive the correct data over UART which means that the BLE113 should get it too. I tried both with the send_api_packet function and with directly sending the above bytes to the BLE113 'myself'.

But the BLE113 does not respond to the data by turning on the LED attached to port P1_0. I've tried almost every possible hardware.xml configuration but it does not work. Does anyone have ideas? They would be very much appreciated!

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

Comments

Simon Deleersnyder

For people having similar problems: I finally fixed it. I added <wakeup_pin enable="true" port="0" pin="0"/> to my hardware.xml file and plugged the P0_0 port into the 5V and now everything works. Obviously, it's better to only set it 5V when you need to send something.

November 2, 2014, 23:22

Chung Mui Bluegiga Technologies

Hi Simon

Thanks for your suggestions. Actually, you can disable sleep mode (power mode) by adding <sleep enable="false" /> in hardware.xml if current consumption is not a problem for your application. If you want to wake up the module by asserting P0_0, it is better to connect it to DVDD. It is not a good idea to connect GPIO pins to 5V.

Thanks!

Support

November 3, 2014, 06:33

Simon Deleersnyder

Hi Chung,

Thanks for your quick response!

The reason that I spent almost a week trying to get it working is that I indeed assumed that when disabling sleep mode it would work without a wakeup pin. But it does not! Could you check this behaviour? Because if you can't reproduce it maybe my BLE113 is faulty?

It works with this hardware.xml file (and with the wakeup pin plugged into 5V or DVDD):

```
<hardware>
  <sleeposc enable="true" ppm="30" />
  <sleep enable="true" />
  <wakeup_pin enable="true" port="0" pin="0"/>
  <txpower power="15" bias="5" />
  <usart mode="packet" channel="1" alternate="1" baud="4800" endpoint="api" flow="false" />
  <port index="0" pull="down" tristatemask="0" />
  <port index="1" pull="down" tristatemask="0" />
  <port index="2" pull="down" tristatemask="0" />
</hardware>
```

But it does not work with this file (I only changed sleep to false, and removed the wakeup pin tag):

```
<hardware>
  <sleeposc enable="true" ppm="30" />
  <sleep enable="false" />
  <txpower power="15" bias="5" />
  <usart mode="packet" channel="1" alternate="1" baud="4800" endpoint="api" flow="false" />
  <port index="0" pull="down" tristatemask="0" />
  <port index="1" pull="down" tristatemask="0" />
  <port index="2" pull="down" tristatemask="0" />
</hardware>
```

And one other question I have: you told me not to connect the wakeup pin to 5V but instead to DVDD. I'm using 5V logic so isn't that the same in my case?

November 3, 2014, 12:41

Chung Mui Bluegiga Technologies

Hi Simon

I will test your hardware.xml later when I have spare time.

-- And one other question I have: you told me not to connect the wakeup pin to 5V but instead to DVDD. I'm using 5V logic so isn't that the same in my case?

Chung>> In my opinion, you can't use 5V logic for BLE113. The recommended operating voltage for DVDD is 3.6V. The absolute max. rating on GPIO pin is DVDD + 0.4. 5V is higher than 3.6 + 0.4 so it is not recommended. For the details, see section 2.4 and 2.5 in the datasheet.

November 3, 2014, 13:26

Hi Chung,

Thanks, I appreciate your quick replies!

Simon Deleersnyder

I was confused since I'm using Jeff Rowberg's breakoutboard which has a 5V pin. But I just read that this pin is connected to a 3.3V regulator to keep it within the required range. So the BLE113 is indeed operating at 3.3V in my case and I should connect the wakeup pin to 3.3V and not to 5V.

Does this also mean I cannot connect the RX and TX pins of my Atmega (operating at 5V) directly to the RX and TX pins of the BLE113 (operating at 3.3V)?

November 3, 2014, 13:45

Chung Mui Bluegiga Technologies

Hi Simon

-- Does this also mean I cannot connect the RX and TX pins of my Atmega (operating at 5V) directly to the RX and TX pins of the BLE113 (operating at 3.3V)?

Chung>> No, you can't.

Luckily, there is an easy solution. 3.3V is okay for TTL (5V) logic (just the noise margin is not as good as 5V but there should not be a big problem). That is to say, you can make a direct connection for Tx (3.3V) of BLE113 to 5V-logic inputs.

For Rx pin of BLE113, you can use a voltage divider to change 5V logic to 3.3V logic.

November 3, 2014, 16:17

Simon Deleersnyder

Thanks Chung, that is good to know! I added the voltage divider to my design.

I'm still struggling with my read_api_packet() method though. The problem lies within the ble_get_msg_hdr() method that gets called. Even when I manually set the correct header within the read_api_packet method, the message that gets returned is NULL.

I rerecreate a ble_evt_system_boot header by setting the header to '80 0C 00 00' regardless of what is received. So the ble_get_msg_hdr() method should be able to return the correct message. But it does not.

Here is my read_api_packet method:

```
int read_api_packet(int timeout_ms)

{

struct ble_header hdr;

unsigned char data[256];


// Read received header

hdr.type_hilen = uart_rx();

hdr.lolen = uart_rx();

hdr.cls = uart_rx();

hdr.command = uart_rx();


    // Recreate ble_evt_system_boot header (so the read received header is never used)

hdr.type_hilen = 0x80;

hdr.lolen = 0x0C;

hdr.cls = 0x00;

hdr.command = 0x00;


const struct ble_msg *msg = ble_get_msg_hdr(hdr);


if(!msg)

{

// This is where the code always ends up
```

```
exit(1);

}

msg->handler(data);


return 0;

}
```

Any thoughts?

November 3, 2014, 17:34

Chung Mui
Bluegiga Technologies

Hi Simon

I have just tired your hardware.xml but no problem was found -- I could send commands and receive responses without asserting P0_0. I attached my project for your reference. Below is the log and I used SDK v1.3.1 - build 119 for me testing.

2014.11.04 09:40:30.667 ble_cmd_system_get_info
2014.11.04 09:40:30.670 TX: 0400000008

2014.11.04 09:40:30.719 ble_rsp_system_get_info major: 1 (0x0001) minor: 3 (0x0003) patch: 1 (0x0001) build: 119 (0x0077) ll_version: 3 (0x0003) protocol_version: 1 (0x01) hw: 255 (0xff)
2014.11.04 09:40:30.720 RX: 000c000801000300010077000300001ff

For your software questions, I am afraid I am not familiar with c programming and I am unable to help.

Thanks!

uartdemo_with_sleep-mode-disabled_4800.rar

November 4, 2014, 03:48

Simon Deleersnyder

Answer

Thank you. Turns out I broke the BLE113 by applying 5V to the wakeup pin. I ordered a new breakoutboard from Jeff Rowberg and it works fine now without the wakeup pin.

As for the other error (in C), it turns out it was a memory issue. The BGLib library was taking up all my Atmega's 2KB of RAM. I moved the library to flash and it works now.

Thanks for the help!

November 9, 2014, 18:52

Kenny Kuchera

I came across this post and have a follow up question. (If I should start a new post for this let me know.)

@Chung About converting 3.3V and 5V for UART (on 2 unidirectional lines), how is this usually done in production? I understand that you can simply do this with a voltage divider to drop it down and with a transistor to bring it up. However I'm not sure that this is the conventional way to do this? I read that it should be possible to also do it with http://www.nxp.com/documents/data_sheet/BSS138BKS.pdf but not sure if this is the way to go? Thanks for your time.

November 15, 2014, 13:21

Jeff Rowberg
Bluegiga Technologies

Hi Kenny,

The BSS138 chip is a very common cheap way to handle level-shifting of UART lines; another single-chip solution is the **TXB0104 from TI**, which is a flexible and extremely simple approach.

Here's an example of an even more basic transistor + voltage divider approach:

- http://blog.sunyday.net/?p=36

Generally speaking, there is no reason why the TXB0104 approach (for example) is really *better* than another approach, given typical UART designs, other than the convenience of a single-chip solution.

November 17, 2014, 16:20