Is there some way to use low energy UART?

Answered

p b
asked this on February 19, 2014, 21:59

Hi,

I've been testing UART communication without flow control (to reduce needed pins count) for a few days and I got to a stage where it was all working fine so I started to test energy usage to see if it's still in acceptable ranges. What I've found is that my current is around 11mA. Thats a lot so i started to dig into the reason. After quite a few hours i discovered that it was a typo in <usb enable="false" endpoint="none" /> which actually enabled the USB (it was <usb enable="false" endpoint="none" /> with an extraneous 'd' after enable).

This is the part which surprised me quite a lot - when USB is disabled no data is received on the BLE112 side. It still sends bytes over UART as it did with USB enabled but nothing is parsed on RX. Is there any reason for this? I don't understand how USB and UART (on uart1 endpoint) are related and why is the hardware flow control really necessary with BGScript (i found a mention that SW flow is available with BGApi which is not the case). Does this really need to use additional 10mA to have a possibility of sending some data over only two serial wires? Hardware flow control is kind of not an option for me for various reasons. I'd also like not to re-implement serial communication on arbitrary pins of BLE112 since it would be kind of stupid thing to do.

This was my second attempt after trying to use UART over USB which didn't work (consumes to much current + conflicts with OTA updates: https://bluegiga.zendesk.com/entries/40507707-USB-UART-vs-OTA-updates). Now I'm planning to switch to I2C to get some low energy wired communication. Could someone tell me if I should expect some unpleasant surprises on that end? I'd really like not to add an additional chip just for wired communication with the BLE112 module since it would also add to the overall power consumption.

Best,
PB

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

Comments

Jeff Rowberg
Bluegiga Technologies

Answer

Hi PB,

This is expected behavior; when USB is enabled, the module is prevented from entering sleep mode. Now that it is disabled, the module can enter sleep mode, but UART RX functionality doesn't work while the module is asleep. This is the reason for the **<wakeup_pin>** tag that goes in **hardware.xml**. You need to make use of this as described in the Bluetooth Smart Configuration Guide in order to force the module awake while you are sending data to it over UART; there is no other way to use UART and sleep mode at the same time.

February 19, 2014, 22:12

p b

From what i've found in one of the documents is that the module will not enter power mode 3 (which disables UART RX as stated in the Configuration Guide on page 15) if a timer is running which is my case. So theoretically the module should never enter PM3?

February 19, 2014, 22:19

Jeff Rowberg
Bluegiga Technologies

The UART will not function in any of the PM1-PM3 modes as far as I am aware; only when the CPU is actually active (i.e. the module is not asleep). A running timer will indeed prevent the module from dropping any lower than PM2, but UART RX will not function in PM2 either.

February 19, 2014, 22:43

p b

Ok, thanks. Should I expect similar limitations with I2C communication?

February 19, 2014, 22:46

To my knowledge, yes. Any clocked peripheral interface will not function while the module is asleep.

Support

Jeff Rowberg
Bluegiga
Technologies

February 19, 2014, 22:51

p b

I understand that only wakeup pin can wake up the module... But if the BLE112 is the I2C master and it writes/request data from slave devices it should work for example in a timer (or other time when the device is active). or am i missing something?

February 19, 2014, 22:54

Jeff Rowberg
Bluegiga
Technologies

That is correct; if the module is already awake and executing some other event handler (e.g. **hardware_soft_timer**), then you can operate under the assumption that the CPU will be fully awake until the event handler code finishes.

February 19, 2014, 22:55

p b

Thanks

February 19, 2014, 22:56

Add a comment

Save comment