EzyBlue - Easy BlueTooth Development kits

USB-BLE112 Data Sheet

Figure 1 - USB-BLE112 device

Overview

The USB-BLE112 is a bluetooth evaluation kit to provide education on using the Bluetooth interface. When installed, the USB-BLE112 adaptor will appear as a virtual Com port and run at a default baud rate of 115200. We recommend using BlueTerminal as the serial emulator program to view the Bleutooth data.

Parameters

Parameter	Value	Nominal
Baud rate	1200 to 3.0M	115200,8,n,1
Supply	5V (USB Powered)	-
Current (Idle)	13 to 35mA	25mA RX to 36mA TX

Table 1 - Electrical Characteristics

Operation

Plug the USB-BLE112 adaptor into a USB port. Allow windows to automatically install the driver, or download the driver from : http://www.ftdichip.com/Drivers/VCP.htm. When the driver has installed, open the Device Manager and determine which COM port number is used by the FT232R USB UART device. Run a terminal program such as the Blue Terminal program, select the COM port, and set the baud rate as 115200 with no hardware control. Use a paperclip to reset the device and check one of the message below is shown.

EV_SLAVE
** Boot: Connection Slave, GATT Server ** Version: et-0.8 **

EV_MASTER
** Boot: Connection Master, GATT Client ** Version: et-0.8 **

EzyBlue - Easy BlueTooth Development kits

More Info

For more info, please refer to http://esdn.com.au/news.htm

EV_SLAVE program

The EV_SLAVE hex file can be compiled from the bgs or Bluegiga script file. The accessory xml files will also need to be included. Below is the project.xml file and not the flow=”false” is necessary when not usign the handshaking CTS/RTS lines.

To program, use a CC-Debugger programmer and Bluetooth Smart Software v1.1.1 (or later) and the BLE Update tool.

```
<?xml version="1.0" encoding="UTF-8" ?>

<hardware>
    <sleeposc enable="true" ppm="30" />
    <usb enable="false" />
    <sleep enable="false" />
    <txpower power="15" bias="5" />
    <script enable="true" />
    <usart channel="1" baud="115200" alternate="1" endpoint="none" mode="uart" flow="false"/>

</hardware>
```

BGS file:

```
dim result
dim endpoint
dim in(20) # endpoint data in
dim in_len
dim out(20) # endpoint data out
dim out_len

dim custom_adv_data(9)

event system_boot(major, minor, patch, build, ll_version, protocol_version, hw)

    endpoint = system_endpoint_uart1 // this is 5

    call system_endpoint_tx(endpoint, 63, "\n\r** Boot: Connection Slave, GATT Server ** Version: et-0.8 **\n") // 63 is no of charcters in message

    call system_endpoint_set_watermarks(endpoint, 0, 0) # disable watermarks

    custom_adv_data(0:1) = $04 #Length of adv data
    custom_adv_data(1:1) = $ff #AD type is Manufacturer Specific Data AD type
    custom_adv_data(2:1) = $47 #Bluegiga Company Identifier Code - octet 2
    custom_adv_data(3:1) = $00 #Bluegiga Company Identifier Code - octet 1
    custom_adv_data(4:1) = $e5 #Custom data indicating other side that we support spp_over_ble - or use line below
    #custom_adv_data(4:1) = $e6 #Custom data indicating a remote bled112 to reboot in dfu mode
    custom_adv_data(5:1) = $03 #Length of next adv data
    custom_adv_data(6:1) = $09 # AD type is Complete local name
    custom_adv_data(7:1) = $45 #E
    custom_adv_data(8:1) = $54 #T
```

```
call gap_set_adv_data(1, 9, custom_adv_data(0:9)) #overwrites the friendly name in gatt.xml with ET but also allows closed system's companion module-dongle to automatically connect
  call gap_set_mode(gap_user_data, gap_undirected_connectable)

end

event connection_status(connection, flags, address, address_type, conn_interval, timeout, latency, bonding)

  call system_endpoint_tx(endpoint, 17, "\n\r++ Connected ++")
  call system_endpoint_tx(endpoint, 26, "\n\r++ Connection interval: ")
  call system_endpoint_tx(endpoint, 4, conn_interval)
  call system_endpoint_tx(endpoint, 4, " ++\n")

end

event attributes_status(handle, flags)

  if (handle = xgatt_data) && (flags = 2) then

    call system_endpoint_tx(endpoint, 57, "\n\r++ Local CCC set by remote side to start indications ++")
    call system_endpoint_tx(endpoint, 46, "\n\r++ Transparent data exchange can start ++\n\n\r")

    call system_endpoint_set_watermarks(endpoint, 1, 0) # set RX watermark

  end if

end

event system_endpoint_watermark_rx(curr_endpoint, size)

  in_len = size
  if in_len > 20 then
    in_len = 20
  end if
  call system_endpoint_set_watermarks(endpoint, 0, $ff) # disable RX watermark
  call system_endpoint_rx(endpoint, in_len)(result, in_len, in(0:in_len))
  call attributes_write(xgatt_data, 0, in_len, in(0:in_len))

end

event attclient_indicated(connection, handle)

  if handle = xgatt_data then
    call system_endpoint_set_watermarks(endpoint, 1, $ff) # set RX watermark
  end if

end

event attributes_value(connection, reason, handle, offset, value_len, value_data)

  if handle = xgatt_data then
    out(0:value_len) = value_data(0:value_len)
    out_len = value_len
    call system_endpoint_set_watermarks(endpoint, $ff, out_len) # set TX watermark
  end if

end

event system_endpoint_watermark_tx(curr_endpoint, size)

  if curr_endpoint = endpoint then
    call system_endpoint_set_watermarks(endpoint, $ff, 0) # disable TX watermark
    call system_endpoint_tx(endpoint, out_len, out(0:out_len))
    call attributes_user_write_response(0, 0)
  end if

end

event connection_disconnected(conn, reas)

  call system_endpoint_tx(endpoint, 22, "\n\n\r-- Disconnected --\n")
```

```
                    call system_endpoint_tx(endpoint, 17, "\r-- Reason code: ")
    call system_endpoint_tx(endpoint, 2, reas)
    call system_endpoint_tx(endpoint, 4, " --\n")

    call system_endpoint_set_watermarks(endpoint, 0, 0) # disable watermarks

    call gap_set_adv_data(1, 9, custom_adv_data(0:9))
    call gap_set_mode(gap_user_data, gap_undirected_connectable)

end
```

Connectors

A connector is provided for external connection. If USB power is removed, the interface is held in reset, and an external device may talk to the BLE112.

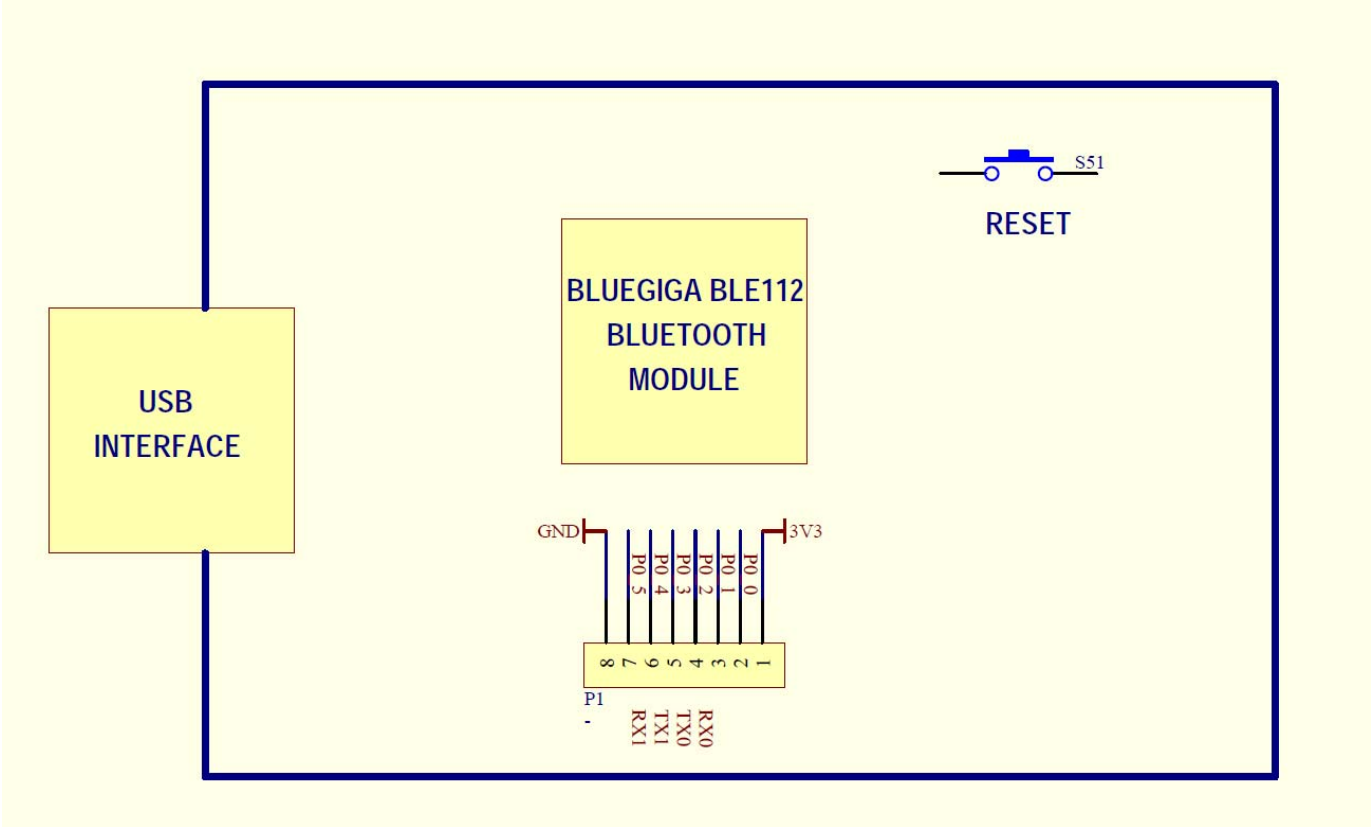

Figure 2 – Expansion Connectors