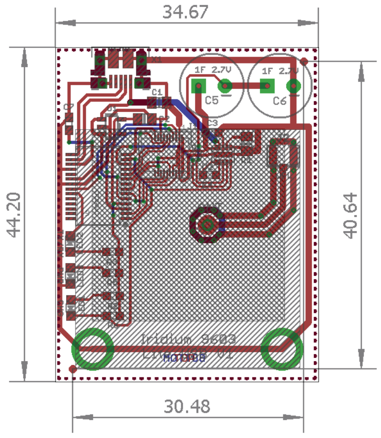
# Iridium 9603 Lite USB



*33 grams of satellite communication joy!*

**Background:**

Following on from the work I did on the Iridium_9603_Beacon,

https://github.com/PaulZC/Iridium_9603_Beacon

which got some nice coverage on Hackaday,

http://hackaday.com/2016/12/19/a-beacon-suitable-for-tracking-santas-sleigh/

I decided to set myself the challenge of designing a really small, lightweight unit that could be used to send messages via the Iridium® satellite network from high altitude balloons and other remote locations. You can already buy the fantastic RockBLOCK Mk2 from Rock7 Mobile , which is also available from Sparkfun, but what if you wanted something even smaller and lighter…?

Inspiration came in the form of the new SparkFun USB UART Serial Breakout which features the Cypress CY7C65213 USB to UART Bridge. This chip makes all eight serial port lines available (TXD, RXD, RTS, CTS, DSR, DTR, DCD and RI) plus eight general purpose input/output (GPIO) pins that can be used to control other functions. In the Iridium_9603_Lite_USB, I use it to: provide a full USB to UART interface for the Iridium 9603 (including the Ring Indicator which can let you know a “mobile terminated” message is waiting to be collected); provide power for the 9603 via an LTC3225EDDB supercapacitor charger *direct from the USB port*; and use the GPIO lines for housekeeping.

I like to use Python where I can and was delighted to find that Taisuke Yamada had written a library called python-ucdev with which I was able to control the CY7C65213 GPIO pins (in manufacturing mode).

The current status is: I have a working breadboard prototype which has been tested successfully with Python 2.7 code running on Windows 10 (64-Bit). Have a look at:

https://github.com/PaulZC/Iridium_9603_Lite_USB_Prototype

The design in this repo condenses the prototype onto a small, 2-layer PCB. The dimensions are 35mm x 44mm x 18mm and the final weight will be close to 33g (including the antenna). The design is ready for manufacture and I plan to have some boards made for testing shortly.

## What does the Iridium 9603 Lite USB actually do?

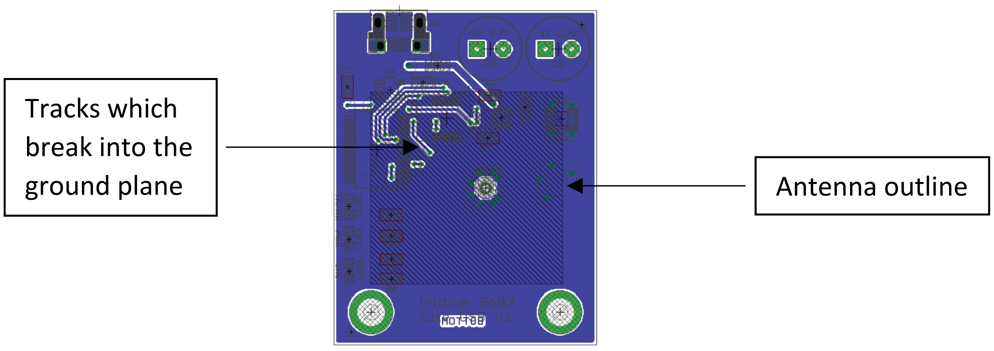
It enables you to send and receive short data messages anywhere using the Iridium® satellite network. The constellation of 66 cross-linked Low Earth Orbit (LEO) satellites provide high-quality data connections over the planet's entire surface, including across oceans, airways and polar regions.

The Iridium_9603_Lite_USB could be connected to a laptop, tablet, phone or an embedded computer like a Raspberry Pi and provide a data link in areas without a mobile phone or WiFi signal.

It is the perfect solution for many tracking or remote monitoring applications.

## So, what's the catch?

The board design in this repo is currently untested. I'm quite confident that the circuit design is sound and will work since both the Iridium_9603_Beacon and the Iridium_9603_Lite_USB_Prototype work well. However, this is a new design on a single 2-layer PCB with the Taoglas Iridium patch antenna mounted on the bottom of the PCB, directly under the 9603 module itself. The patch antenna needs a continuous ground plane for best results. I achieved this on the Iridium_9603_Beacon by having the antenna mounted on its own separate PCB above the 9603. In this design the 9603 and antenna share the same PCB. Although I've kept the number of tracks on the bottom of the PCB – right next to the antenna – to an absolute minimum they do still break up the ground plane to some degree. There is a chance that: the 9603 transmission and reception won't be perfect as the performance of the antenna will be degraded by some – hopefully small – amount; the transmission bursts from the antenna may couple into the PCB tracks and could cause interference and erroneous behaviour.



I’m quite confident that a 4-layer board would work perfectly, as that would allow the bottom layer to become a dedicated unbroken ground plane for the antenna. But as that would push the manufacturing costs up, I’m keen to try 2-layers first.

**<u>Surely this is good enough to be a product. Why aren’t you making and selling them?</u>**

I have a demanding day job and don’t have the time or the funds to have the board CE marked and FCC approved. The 9603 itself is already approved to: ETSI EN 301 441 V1.1.1 (2000-05); ETSI EN 301 489-20 V1.2.1 (2002-11); ETSI EN 301 489-1 V1.9.2 (2011); EN61000-4-2 : 2009; EN61000-4-3 : 2006 + A1: 2008 + A2: 2010; EN61000-4-4 : 2004 + A1: 2010; EN61000-4-6 : 2009; EN55022:2006 + A1: 2007; EN60950-1:2006 + A11: 2009 + A1: 2010 + A12: 2011; FCC CFR47 parts 2 (2013), 15B (2013), and 25 (2013); Industry Canada RSS170 Issue 2, March, 2011; and Industry Canada RSS-GEN Issue 3, December, 2010. Anyone wanting to market the Iridium_9603_Lite_USB as a product would need to have the board (re)certified against the same standards. I would certainly be interested in talking to a manufacturer who would be willing to take this on as a product, look after the certification, and pay me *very* modest royalties in return for continued support and further software development.

Also, the PCB design has been developed using the educational version of Eagle. Anyone wanting to market the Iridium_9603_Lite_USB would need to have the commercial version of Eagle.

**<u>OK. But why make it open source? Why not keep the design to yourself?</u>**

Making the design open source is the right thing to do. Part of the design is based on the SparkFun USB UART Serial Breakout, the design of which is released under a Creative Commons Share-alike 4.0 licence. And so, for that reason alone, the design of the Iridium_9603_Lite_USB needs to be issued under the same licence.

Also, I’m making use of Python code written by Tai Yamada which was released under the GNU Lesser General Public Licence. Sharing my version of Tai’s code under the same licence is the right thing to do, both morally and legally.

If you’re still not convinced, take time to watch the following talks from the amazing people behind Sparkfun and Adafruit. You can make money from open source designs, but you need to go about it in the right way.

https://www.youtube.com/watch?v=aglidIqSrnE

https://www.youtube.com/watch?v=Rfu_MKgu2Ik

https://www.youtube.com/watch?v=Ca1dbM582Sc

https://www.youtube.com/watch?v=2vpECBbWVAY

**Tell me about the design.**

The key components of the Iridium_9603_Lite_USB are:

- Cypress CY7C65213 USB-UART Bridge (in a 28-pin SSOP package)
  - http://www.cypress.com/part/cy7c65213-28pvxi
  - Available from e.g. Mouser (Part# 727-CY7C65213-28PVXI)
- Iridium 9603 Module
  - Available (in the UK) from e.g.:
  - http://www.ast-systems.co.uk/Product-Pages/Iridium-9603-SBD—Satellite-Tracking-Transceiver.aspx
  - http://www.rock7mobile.com/products-iridium-sbd
  - Other UK and International distributors can be found at:
  - https://iridium.com/products/details/Iridium-9603?section=wtb
- Linear Technology LTC3225EDDB SuperCapacitor Charger
  - http://www.linear.com/product/LTC3225
  - Available from e.g. Farnell / Element14 (Part# 1715231)
  - Charges two e.g. Bussmann HV0810-2R7105-R 1F 2.7V capacitors (Farnell / Element14 Part# 2148482)
- Taoglas IP.1621.25.4.A.021 Iridium Patch Antenna
  - Available from e.g. Mouser (Part# 960-IP1621254A02)

The schematic, board layout and bill of materials can be found in:

https://github.com/PaulZC/Iridium_9603_Lite_USB/blob/master/Iridium_9603_Lite_USB_V1.pdf

The Eagle PCB files can be found in:

https://github.com/PaulZC/Iridium_9603_Lite_USB/tree/master/Eagle/V1

**OK. I want to build one. What do I do first?**

Start by having the PCB made. I can recommend Multi-CB who will manufacture PCBs very quickly and at a very reasonable cost. It would be wise to order one or more solder paste stencils at the same time. Multi-CB will allow you to upload the Eagle files direct, avoiding the need to work with Gerber files.

The fiducials FD1 and FD2 will help you to line up the solder stencil (assuming you are using one). Populate the PCB with components C1 to X1. Ignore the non-surface mount components S1 to ANT for now. Populating the surface mount components and reflowing the solder paste is the subject of a whole different tutorial. If you plan on doing it yourself, please be aware that positioning the Samtec connector (J1) and the LTC3225EDDB (U2) will be challenging. The other components can be hand-soldered if required.

Once the surface mount components are in place, you can use a press or vice to press the two press-in nuts S1 and S2 into the **bottom** of the PCB. Then hand-solder the two super capacitors C5 and C6, followed by the Taoglas patch antenna ANT. The capacitors go on the top of the board, the antenna on the bottom. The antenna pin is slightly off-centre, make sure you orientate it correctly before using the adhesive pad to stick it to the board.

**OK. Now what?**

The next step is to configure the CY7C65213. Don't attempt to install the 9603 yet.

Read Sparkfun's CY7C65213 hookup guide:

- https://learn.sparkfun.com/tutorials/sparkfun-usb-uart-breakout-cy7c65213-hookup-guide

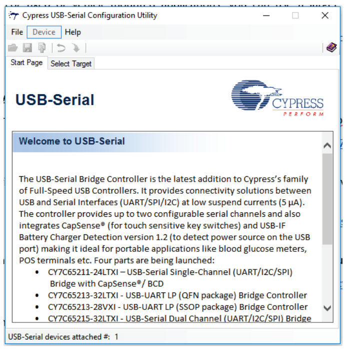
You will also want to download and install Cypress' USB-Serial Software Development Kit:

- http://www.cypress.com/documentation/software-and-drivers/usb-serial-software-development-kit

You might find Cypress' CY7C65213 Reference Design documentation useful too:

- http://www.cypress.com/documentation/development-kitsboards/cyusbs232-usb-uart-lp-reference-design-kit

Plug in your board, run the USB Serial Configuration Utility and check that it detects the CY7C65213:


Cypress USB-Serial Configuration Utility

File | Device | Help

Start Page | Select Target

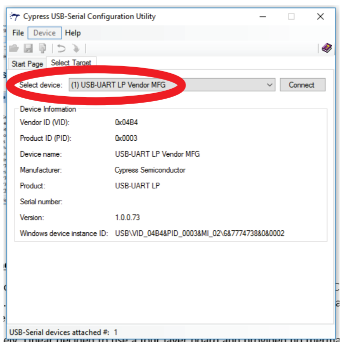
USB-Serial

Welcome to USB-Serial

The USB-Serial Bridge Controller is the latest addition to Cypress's family of Full-Speed USB Controllers. It provides connectivity solutions between USB and Serial Interfaces (UART/SPI/I2C) at low suspend currents (5 µA). The controller provides up to two configurable serial channels and also integrates CapSense® (for touch sensitive key switches) and USB-IF Battery Charger Detection version 1.2 (to detect power source on the USB port) making it ideal for portable applications like blood glucose meters, POS terminals etc. Four parts are being launched:

- CY7C65211-24LTXI – USB-Serial Single-Channel (UART/I2C/SPI) Bridge with CapSense®/ BCD
- CY7C65213-32LTXI - USB-UART LP (QFN package) Bridge Controller
- CY7C65213-28VXI - USB-UART LP (SSOP package) Bridge Controller
- CY7C65215-32LTXI - USB-Serial Dual Channel (UART/I2C/SPI) Bridge

USB-Serial devices attached #: 1


Cypress USB-Serial Configuration Utility

File | Device | Help

Start Page | Select Target

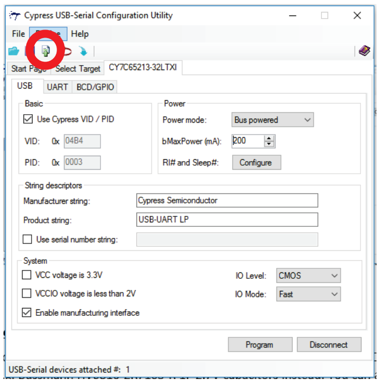
Select device: (1) USB-UART LP Vendor MFG | Connect

Device Information

| | |
|---|---|
| Vendor ID (VID): | 0x04B4 |
| Product ID (PID): | 0x0003 |
| Device name: | USB-UART LP Vendor MFG |
| Manufacturer: | Cypress Semiconductor |
| Product: | USB-UART LP |
| Serial number: | |
| Version: | 1.0.0.73 |
| Windows device instance ID: | USB\VID_04B4&PID_0003&MI_02\6&7774738&0&0002 |

USB-Serial devices attached #: 1

Select the "USB-UART LP Vendor MFG" and connect to it. Click the "Open Configuration from Device" icon to ensure the default device configuration has been read successfully. Click the Save icon and save the default configuration using a filename like "CY7C65213_Default.cyusb". That way, you can restore the default configuration if you need to.


Cypress USB-Serial Configuration Utility

File | Device | Help

Start Page | Select Target | CY7C65213-32LTXI

USB | UART | BCD/GPIO

Basic: ☑ Use Cypress VID / PID; VID: 0x 04B4; PID: 0x 0003

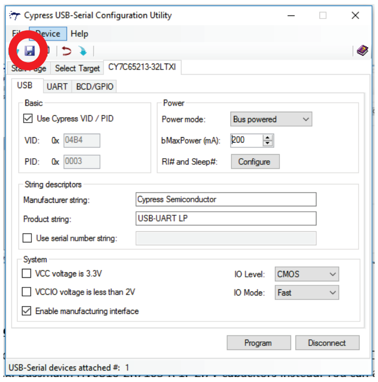
Power: Power mode: Bus powered; bMaxPower (mA): 200; RI# and Sleep#: Configure

String descriptors: Manufacturer string: Cypress Semiconductor; Product string: USB-UART LP; ☐ Use serial number string

System: ☐ VCC voltage is 3.3V; ☐ VCCIO voltage is less than 2V; ☑ Enable manufacturing interface; IO Level: CMOS; IO Mode: Fast

Program | Disconnect

USB-Serial devices attached #: 1


Cypress USB-Serial Configuration Utility

File | Device | Help

Start Page | Select Target | CY7C65213-32LTXI

USB | UART | BCD/GPIO

Basic: ☑ Use Cypress VID / PID; VID: 0x 04B4; PID: 0x 0003

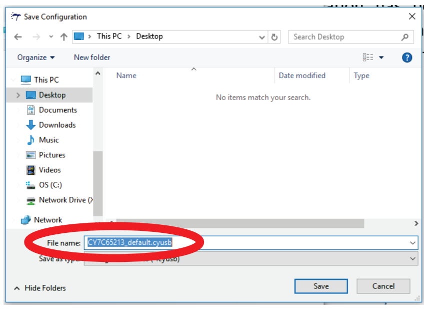
Power: Power mode: Bus powered; bMaxPower (mA): 200; RI# and Sleep#: Configure

String descriptors: Manufacturer string: Cypress Semiconductor; Product string: USB-UART LP; ☐ Use serial number string

System: ☐ VCC voltage is 3.3V; ☐ VCCIO voltage is less than 2V; ☑ Enable manufacturing interface; IO Level: CMOS; IO Mode: Fast

Program | Disconnect

USB-Serial devices attached #: 1


Save Configuration

This PC › Desktop | Search Desktop

Organize | New folder

This PC, Desktop, Documents, Downloads, Music, Pictures, Videos, OS (C:), Network Drive, Network

Name | Date modified | Type

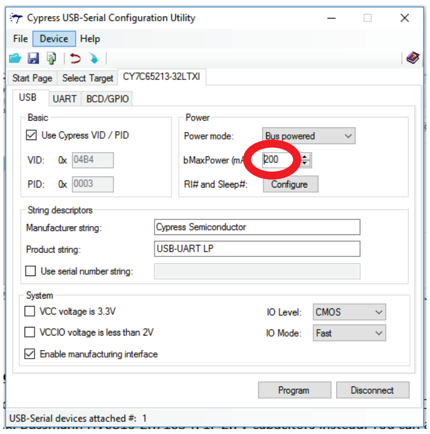
No items match your search.

File name: CY7C65213_default.cyusb

Save as type: (*.cyusb)

Hide Folders | Save | Cancel

Set "bMaxPower (mA) to 200":


Cypress USB-Serial Configuration Utility

File | Device | Help

Start Page | Select Target | CY7C65213-32LTXI

USB | UART | BCD/GPIO

Basic: ☑ Use Cypress VID / PID; VID: 0x 04B4; PID: 0x 0003

Power: Power mode: Bus powered; bMaxPower (mA): 200; RI# and Sleep#: Configure

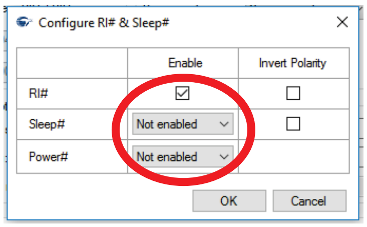
String descriptors: Manufacturer string: Cypress Semiconductor; Product string: USB-UART LP; ☐ Use serial number string

System: ☐ VCC voltage is 3.3V; ☐ VCCIO voltage is less than 2V; ☑ Enable manufacturing interface; IO Level: CMOS; IO Mode: Fast

Program | Disconnect

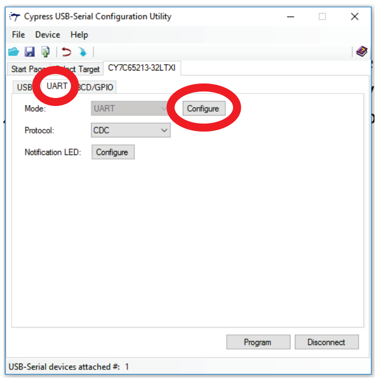
USB-Serial devices attached #: 1

Click the Configure button next to "RI# and Sleep#". Leave RI enabled. Set "Sleep#" and "Power#" to "Not enabled", then click OK. (You could leave Sleep# set to GPIO 04 if you want the USB host to be able to put the supercapacitor charger to sleep automatically, but I haven't tested this as yet.)


Configure RI# & Sleep#

| | Enable | Invert Polarity |
|---|---|---|
| RI# | ☑ | ☐ |
| Sleep# | Not enabled | ☐ |
| Power# | Not enabled | |

Select the UART tab and then click the Configure button next to “Mode”:


Cypress USB-Serial Configuration Utility

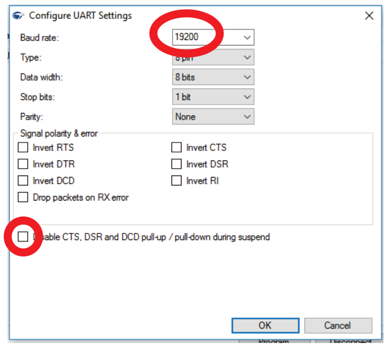
Mode: UART

Protocol: CDC

Notification LED:

USB-Serial devices attached #: 1

Set the Baud rate to 19200, untick the “Disable CTS...” box and then click OK.


Configure UART Settings

Baud rate: 19200

Type: 8 pin

Data width: 8 bits

Stop bits: 1 bit

Parity: None

Signal polarity & error

☐ Invert RTS ☐ Invert CTS

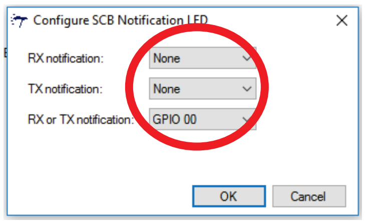
☐ Invert DTR ☐ Invert DSR

☐ Invert DCD ☐ Invert RI

☐ Drop packets on RX error

☐ Disable CTS, DSR and DCD pull-up / pull-down during suspend

Click the Configure button next to “Notification LED”. Set “RX notification” and “TX notification” to “None”. Set “RX or TX notification” to “GPIO 00”. Then click OK.

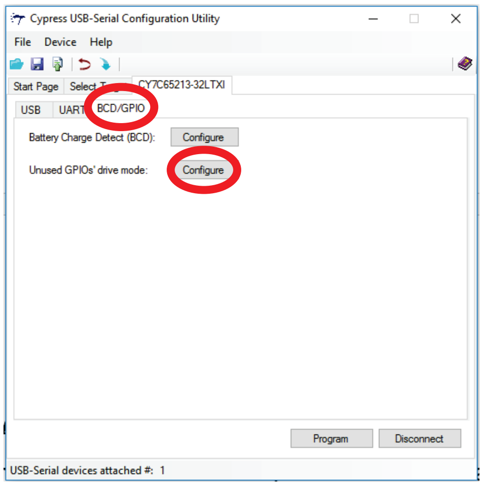

Configure SCB Notification LED

RX notification: None

TX notification: None

RX or TX notification: GPIO 00

Select the BCD/GPIO tab and then click the Configure button next to “Unused GPIOs’ drive mode”:


Cypress USB-Serial Configuration Utility

File Device Help

Start Page | Select Target | CY7C65213-32LTXI

USB | UART | BCD/GPIO

Battery Charge Detect (BCD): Configure

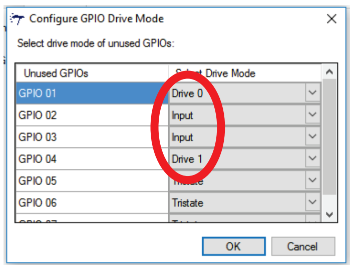
Unused GPIOs' drive mode: Configure

Program | Disconnect

USB-Serial devices attached #: 1

Select the following drive modes then click OK:

- GPIO 01: “Drive 0” (this will turn the 9603 off by default)
- GPIO 02: “Input” (for the supercapacitor charger PGOOD signal)
- GPIO 03: “Input” (for the Iridium 9603 Network Available signal)
- GPIO 04: “Drive 1” (this will power up the supercapacitor charger by default)
    - (GPIO 04 will be missing from the list if it is allocated to Sleep#)

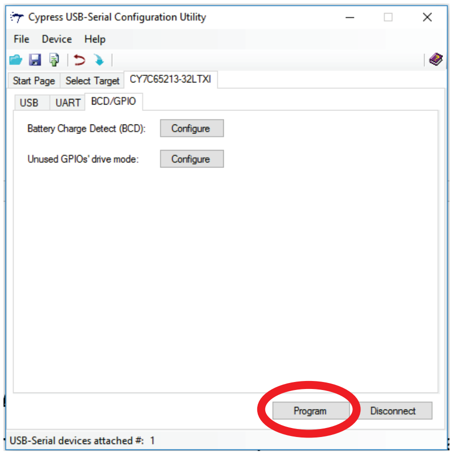

Configure GPIO Drive Mode

Select drive mode of unused GPIOs:

| Unused GPIOs | Select Drive Mode |
|---|---|
| GPIO 01 | Drive 0 |
| GPIO 02 | Input |
| GPIO 03 | Input |
| GPIO 04 | Drive 1 |
| GPIO 05 | Tristate |
| GPIO 06 | Tristate |

OK | Cancel

Click “Program” to program these settings into the CY7C65213. Check that the programming is successful.


Cypress USB-Serial Configuration Utility

File Device Help

Start Page | Select Target | CY7C65213-32LTXI

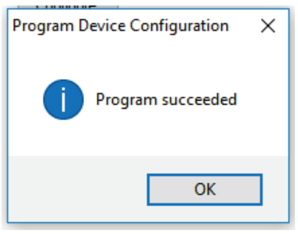
USB | UART | BCD/GPIO

Battery Charge Detect (BCD): Configure

Unused GPIOs' drive mode: Configure

Program | Disconnect

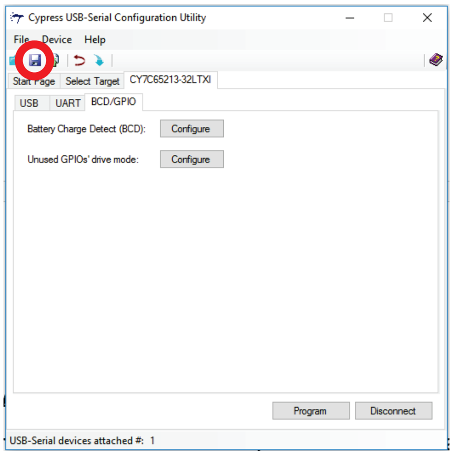
USB-Serial devices attached #: 1

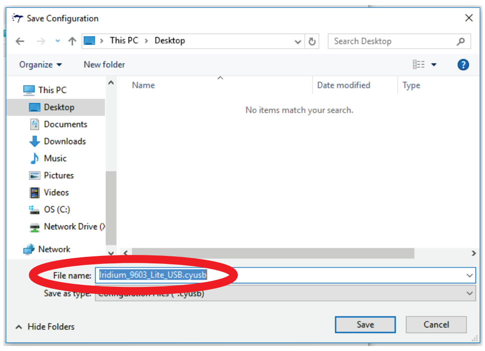

Program Device Configuration

Program succeeded

OK

Save the configuration into a new .cyusb file called something like Iridium_9603_Lite_USB.cyusb.

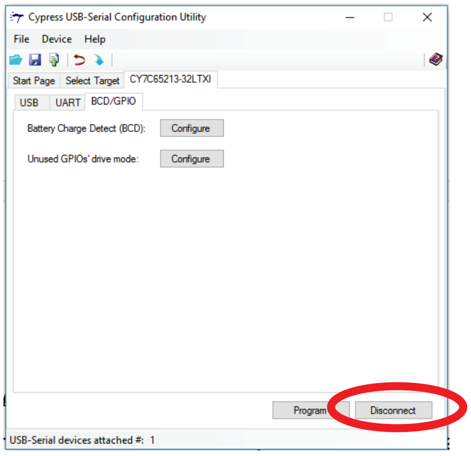

Cypress USB-Serial Configuration Utility
File Device Help
Start Page Select Target CY7C65213-32LTXI
USB UART BCD/GPIO
Battery Charge Detect (BCD): Configure
Unused GPIOs' drive mode: Configure
Program Disconnect
USB-Serial devices attached #: 1



File name: Iridium_9603_Lite_USB.cyusb

Save as type: Configuration Files (*.cyusb)

Finally, click "Disconnect" and unplug the CY7C65213. It will use the new configuration the next time it is powered up.


Cypress USB-Serial Configuration Utility
File Device Help
Start Page Select Target CY7C65213-32LTXI
USB UART BCD/GPIO
Battery Charge Detect (BCD): Configure
Unused GPIOs' drive mode: Configure
Program Disconnect
USB-Serial devices attached #: 1

## OK. The CY7C65213 is configured. Now what do I do?

At this point I would recommend testing the supercapacitor charger and to do that you will need to run the Python code Iridium_9603_Lite_USB.py:

https://github.com/PaulZC/Iridium_9603_Lite_USB/blob/master/Iridium_9603_Lite_USB.py

You will also need to clone or download:

- https://github.com/PaulZC/python-ucdev
- Run "python setup.py install" to install the library, or make sure the cy7c65213 sub-directory is copied into the same directory as Iridium_9603_Lite_USB.py

If you haven't installed it before, you will also need CFFI:

- https://pypi.python.org/pypi/cffi?
- http://cffi.readthedocs.io/en/latest/

The code won't work without Cypress' cyusbserial.dll (on Windows machines):

- The dll gets installed as part of the SDK (see above)
- You will usually find the 64-Bit Windows version in:
  - C:\Program Files (x86)\Cypress\USB-Serial SDK\library\cyusbserial\x64
- The 32-Bit Windows version is usually found in:
  - C:\Program Files (x86)\Cypress\USB-Serial SDK\library\cyusbserial\x86

There are versions of the SDK for OS-X, Linux and Android but I haven't had time to test those yet.

Run the code. The code will power up the supercapacitor charger and you can check that the capacitors charge to 4.8V (2.4V each; flip the board over and probe the pins of C5 and C6, or probe carefully on C3 on the top of the board). The code will stall at "Writing AT Expecting OK". That's fine. Press Ctrl-C to exit the code. If the code keeps printing "Waiting for PGOOD to go high..." for more than a minute then you have a fault you need to investigate.

The green "PWR" LED will light up whenever the board is powered. The yellow "TX/RX" LED will flash whenever there is activity on the USB interface. The red "CHG" LED will light up while the supercapacitors are charging and go out once they are charged.

## <u>OK. Is it time to mount the 9603?</u>

Yes. Make sure the board is unplugged and wait for C5 and C6 to discharge. This can take a while! If you're feeling brave, you can use a 100R resistor to 'short out' C3.

Take anti-static precautions. Ideally you should be wearing a wrist strap.

Rotate the PCB so the edge with the uFL connector is facing you. Flip the 9603 over and lie it to the left of the PCB with its uFL connector facing you. Carefully connect the short Molex uFL cable to the 9603 and then to the uFL connector on the PCB. Gently pick up the 9603, bend the Molex cable right of centre and gently press the 9603 onto the Samtec connector. Now position the two spacers between the 9603 and the press-in nuts and finally use the two 2-56 screws to secure the 9603 in position. Don't over-tighten the screws.

Reconnect the board, run the code again and your first message should be delivered to the email address you registered when you purchased your 9603 module. Don't forget you need to flip the board over and give the patch antenna a clear view of the sky. It is unlikely to work indoors.

## <u>How are the GPIO pins used?</u>

- GPIO0 will flash the "TX/RX" LED whenever data is transmitted or received over the USB port
- GPIO4 is used to enable or disable the LTC3225EDDB supercapacitor charger
    - Hold GPIO4 high to enable the LTC3225EDDB
    - Low puts the LTC3225EDDB into Shutdown
- GPIO2 is connected to the LTC3225EDDB PGOOD signal
    - High indicates that the power is good (CHG LED off)
    - Low indicates that the capacitors are still charging (CHG LED on)
- GPIO1 is used to turn the Iridium 9603 on (high) or off (low)
    - At power-up, hold GPIO1 low until GPIO2 (PGOOD) goes high
- GPIO3 is connected to the Iridium 9603 Network Available signal
    - This pin can be monitored to check whether the 9603 can see an available satellite network
    - High indicates that the network is available
    - Monitoring this pin is equivalent to issuing an AT+CSQ command

**Are there any tricks I need to know?**

The main one is that the Iridium 9603 will refuse to communicate unless both the RTS and DTR pins are pulled low. Using pyserial, you need to set both pins to “1” to produce a low output:

- set_RTS(1)
- set_DTR(1)

**What other resources do you recommend?**

Here are a few useful links:

- http://www.makersnake.com/rockblock/
- https://cdn.sparkfun.com/datasheets/Wireless/General/IRDM_ISU_ATCommandReferenceMAN0009_Rev2.0_ATCOMM_Oct2012.pdf
- https://www.networkinv.com/wp-content/uploads/2012/08/Iridium-9603-Developers-Guide-v2-1.pdf

**Will the Iridium 9603 Lite USB really weigh 33g?**

Yes, I think so. Here is how the mass estimate breaks down:

| | |
|---|---|
| • PCB (estimate): | 8.3g |
| • Iridium 9603 Module: | 11.6g |
| • Iridium Antenna: | 9.3g |
| • Module Fixings: | 1.7g |
| • Super Capacitors: | 1.7g |
| • uFL Cable: | 0.4g |
| • Total: | 33g |

**Why do you need the Super Capacitors?**

The Iridium 9603 module draws a peak current of 1.3A when transmitting its short data bursts. That’s too much for a standard USB port. The LTC3225 super capacitor charger draws 150mA from the USB port to charge two 1F 2.7V capacitors, connected in series, to 4.8V. The capacitors then deliver the 1.3A to the module when it sends the data burst.

**<u>Can I alter the 150mA charge current?</u>**

Yes. Replacing R1 (12K) with a 60.4K resistor will limit the charge current to 30mA:

$$\text{Charge Current (A)} = 1800 / R1$$

The capacitors will of course take much longer to charge. Consult the LTC3225EDDB datasheet for further details. You can then reduce the CY7C65213 bMaxPower setting to match.

**<u>What data can I send and receive?</u>**

"Mobile Originated" messages – messages sent from the 9603 – can be up to 340 bytes and sent as text or binary data.

"Mobile Terminated" messages – messages sent to the 9603 – can be up to 270 bytes.

You can receive MO data as an email attachment from the Iridium system. The email itself contains extra useful information:

- Message sequence numbers (so you can identify if any messages have been missed)
- The time and date the message session was processed by the Iridium system
- The status of the message session (was it successful or was the data corrupt)
- The size of the message in bytes
- The approximate latitude and longitude the message was sent from
- The approximate error radius of the transmitter's location

E.g.:

*From: sbdservice@sbd.iridium.com*
*Sent: 20 August 2016 16:25*
*To:*
*Subject: SBD Msg From Unit: 30043406174*****
*Attachments: 30043406174****_000029.sbd*

*MOMSN: 29*
*MTMSN: 0*
*Time of Session (UTC): Sat Aug 20 15:24:57 2016 Session Status: 00 - Transfer OK*
*Message Size (bytes): 61*

*Unit Location: Lat = 55.87465 Long = -2.37135 CEPradius = 4*

## Thanks!

This project wouldn't have been possible without the open source designs and code kindly provided by:

- **Sparkfun**:
  - The USB UART Serial Breakout CY7C65213 (Product BOB-13830)
  - https://www.sparkfun.com/products/13830
- **Taisuke Yamada (Tai)**
  - Python library for Cypress CY7C65211/CY7C65215 USB-Serial bridge
  - https://github.com/tai/python-ucdev

## The small print