MU User Manual

A guide to using MU middleware

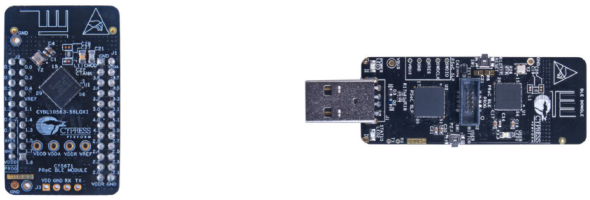
v 1.0

UNIZG

May 29, 2017

Chapter 1

Connecting to BLE device



(a) (b)

Figure 1.1: Hardware needed to establish BLE bridge: (a) BLE module on MU (CY5671), (b) BLE USB dongle (CY5670)

Hardware components:

- BLE module on MU – `http://www.cypress.com/documentation/development-kitsboards/cy5671-proc-ble-module`
- BLE dongle – `http://www.cypress.com/documentation/development-kitsboards/cy5670-cysmart-usb-dongle`
  The purpose of BLE dongle is to provide UART bridge between MU and PC. It can be used for programming MU via BLE Bootloader or for basic serial communication.

In order to connect to BLE device on aMussel, the following prerequisites have to be met:

1. You need to have BLE dongle configured.
2. BLE board on MU has to be programmed.

1.1 Configuring BLE dongle

BLE dongle needs to be programmed using MiniProg. The code is provided in `BLE_DONGLE` project of PSoC creator `MU31S-000` workspace.

Figure 1.2: Connecting BLE dongle to MiniProg3

Steps:

1. Build the project (right click on project name > Build `BLE_DONGLE`).
2. Set `BLE_DONGLE` project to active (right click on project name > Set As Active Project).
3. Connect MiniProg to dongle and the PC (see Figure 1.2).
4. Program the dongle (*program* button in PSoC creator or Ctrl+F5).
5. If prompted, select a device to program.

1.2 Programming BLE board

BLE module can be programmed using MiniProg. The code is provided in `BLE_UART_BRIDGE` project of PSoC creator `MU31S-000` workspace.

Steps:

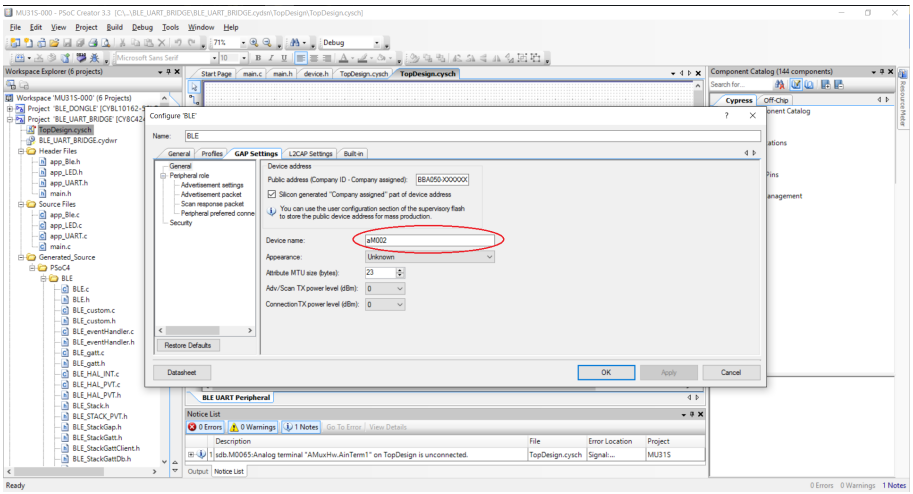


Figure 1.3: Setting the BLE device name

1. Modify the project to define aMussel ID. To do so, open TopDesign.cysch and double click on BLE module. Go to GAP Settings > General and enter aMussel ID in Device Name field (see Figure 1.3). Naming convention is as follows: `'aMXXX'`, where `'XXX'` represents aMussel ID in range $[000 - 999]$, e.g. `'aM001', 'aM023'` ...
2. Build the project (right click on project name > Build `BLE_UART_BRIDGE`).
3. Set `BLE_UART_BRIDGE` project to active (right click on project name > Set As Active Project).
4. Connect MiniProg to BLE board and the PC.
5. Program the device (*program* button in PSoC creator or Ctrl+F5).
6. If prompted, select a device to program.

1.3 Communicating with MU using BLE

To successfully send and receive data from BLE module on MU, we recommend the following setup:

1. Connect programmed BLE dongle to your PC.
2. Open Docklight [1] (or any other tool for serial communication).

[1] `docklight.de/`

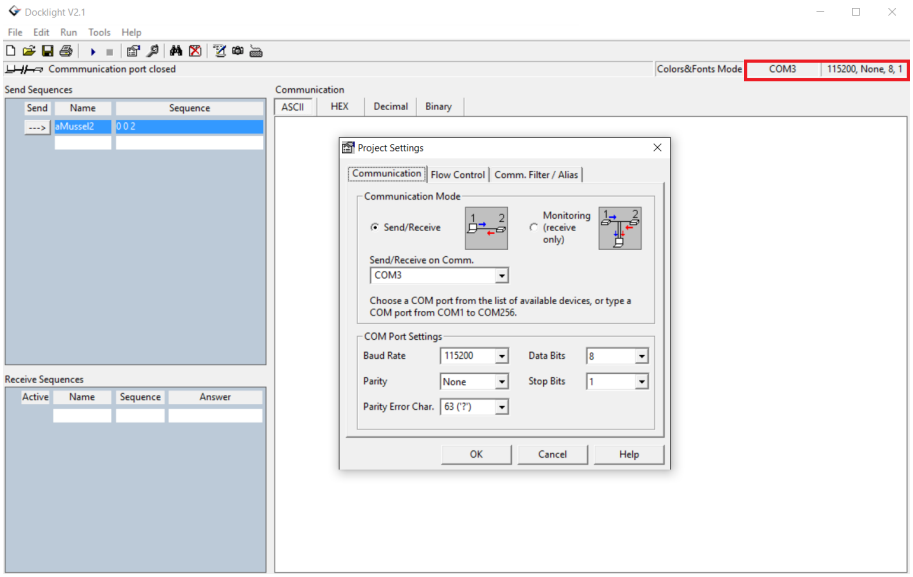


Figure 1.4: Selecting the COM port

3. Connect to the correct COM port. Port can be selected by double clicking on top right sections of status bar (indicated with red rectangle in Figure 1.4). After selecting the port, make sure the settings are matching the ones in the figure. Now you can simply press the play button to establish connection.

4. When connected to BLE dongle, you should see the output in your serial communication program as presented in Figure 1.5. If the output is not there or the program is stuck for any reason, you can always reset it by pressing the user button on dongle (see Figure 1.6).

5. The next step is to select device ID you wish to connect to. You do so by sending a sequence representing device's unique identifier. For example, if BLE board on aMussel is programmed to have ID `001`, the sequence needs to match it. For reference, look at Figure 1.7.

6. Upon connection establishment the program prints out:

```
Server with matching custom service discovered...
Connection established
Notifications enabled
Start entering data:
```

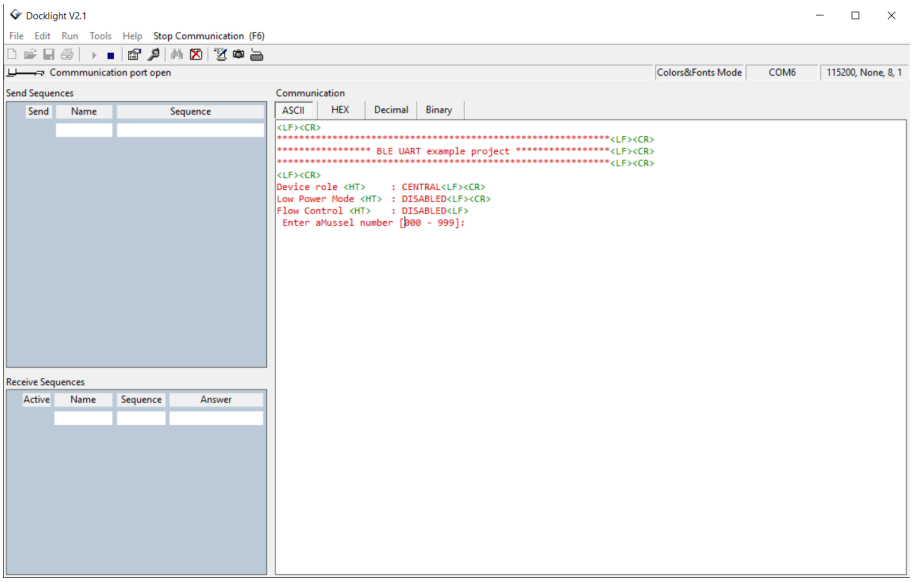


Figure 1.5: BLE bridge status

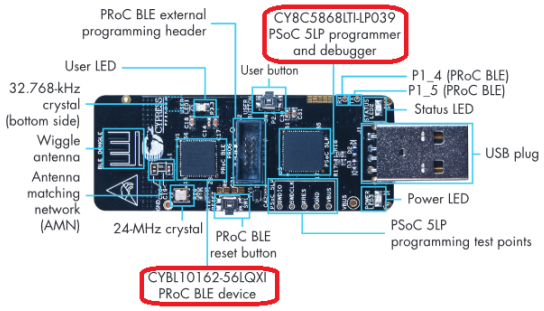


Figure 1.6: CY5670 USB Dongle

7. Now the connection is established and you can start sending and receiving data.

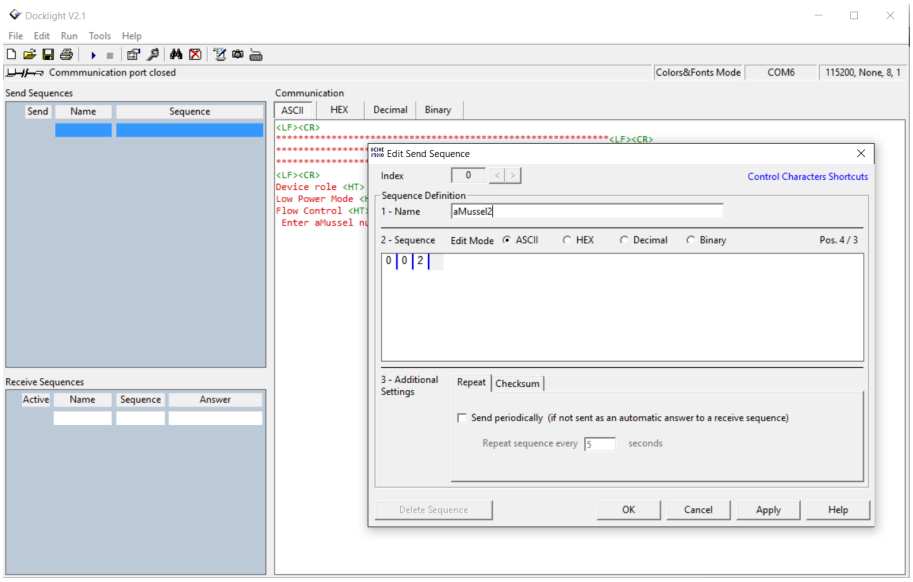


Figure 1.7: Selecting a device to connect to

Chapter 2

Programming using Bootloader

Bootloader is a short program used to burn the firmware to the microcontroller without any programmer device. It is the first thing to run on device power on and is accessible through the serial interface.

2.1 USB connection (obsolete)

Setup described in this section enables you to program Main Boards using wired connection (i.e. mini USB cable) Bootloader.

2.1.1 Programming the Bootloader on Main Board

A prerequisite to using Bootloader to program the Main Board is having the Bootloader itself configured and programmed onto the board. USB (wired) Bootloader is provided in `UART_BOOTLOADER` project of PSoC creator `MU31S-000` workspace.

Steps:

1. Build the project (right click on project name > Build `UART_BOOTLOADER`).
2. Set `UART_BOOTLOADER` project to active (right click on project name > Set As Active Project).
3. Connect MiniProg to MB and the PC.

4. Program the MB (*program* button in PSoC creator or Ctrl+F5).

5. If asked, select a device to program. Sometimes the connection between MB and MiniProg is bad so make sure you adjust the cable until the device `PSoC 5LP CY8C5888LTI*-LP097` is detected.

6. Once the device is detected, select `Port Acquire` > `Connect` > `OK` and wait for the completion of programming process.

2.1.2 Programming MB using Bootloader

In order to program Main Board using Bootloader, `Bootloadable` component on MB has to be running and awaiting for programming request. It can be achieved by setting a waiting time after each system power on during which MB enters this mode. It is, however, not efficient because we sometimes need to program MB without power cycling it. Therefore, programs devised for Main Board always need to have one *interpreter* task running which activates Bootloadable on request. This mode is entered by sending `'b'` character to MB's general purpose (debugging) UART.

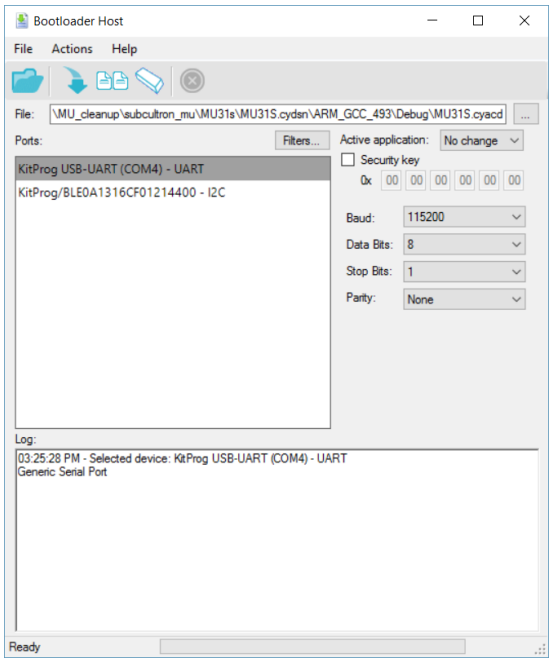


Figure 2.1: Setting up Bootloader Host

The next step is to connect to the Main Board and start the programming process.
Steps:

1. Connect the MB to your PC via mini USB cable. Make sure all connections to this COM port are closed (stop connections in Docklight).
2. Open the project you wish to program in PSoC Creator.
3. Go to `Tools` > `Bootloader Host`.
4. Select the correct COM port and adjust settings to match the ones in Figure 2.1.
5. Click on `Program` button and wait for the process to execute.

2.2 BLE Bootloader (obsolete)

The process of programming MB using BLE Bootloader is the same as the one using wired connection Bootloader. The only difference is in the way the connection to the device is established (step 1 in the previous section). For guide on setting up BLE bridge between MB and PC, refer to Chapter 1. Bootloader project that needs to be programmed onto the Main Board is `UART_Bootloader_BLE`.

2.3 I2C Bootloader

The process of programming MB using I2C Bootloader differs from previous described ones. The process of programming I2C Bootloader stays the same, only the name of the project changes to `I2C_Bootloader.cydsn` in Main Board case, or in eSense board case `I2C_Bootloader_eSense.cydsn`. Desired program is transferred to aMussel's Raspberry Pi over WiFi. Programming is done over I2C, where Raspberry Pi reprograms the Main Board. Detailed description is described in 2.3.1.

Next step is the programming process.
Steps:

1. Build the project you wish to program in PSoC Creator.
2. Put Main Board in Bootloader mode.

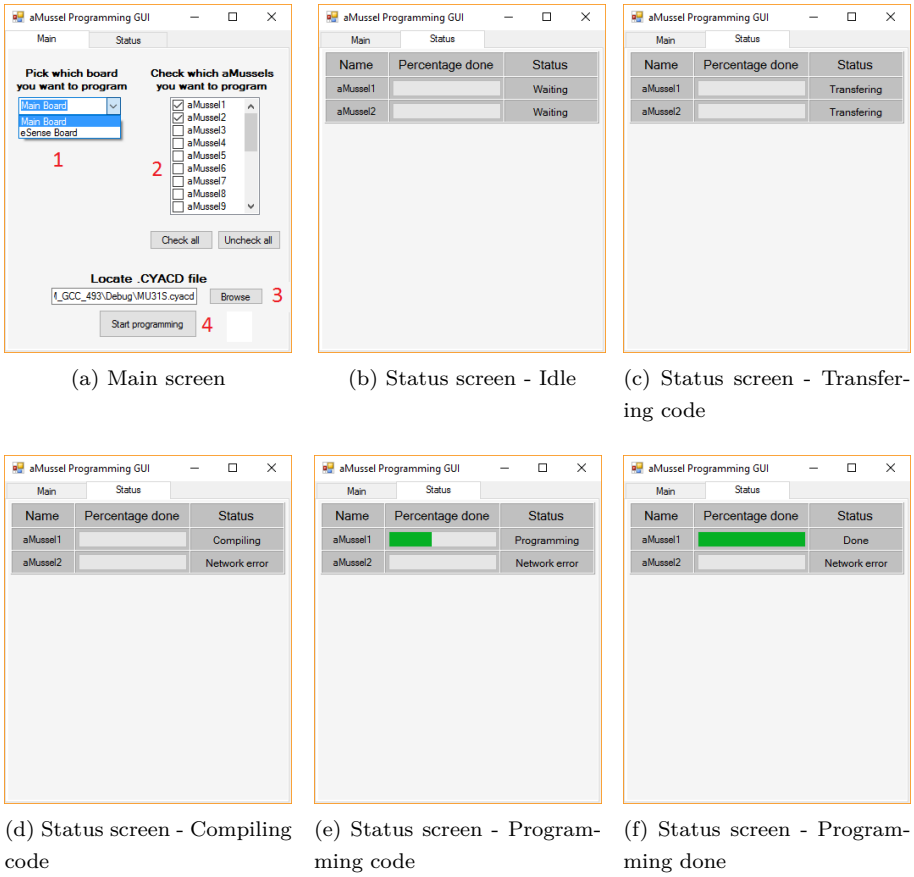
3. Make sure Access Point router is turned on (and setup properly).

4. Wait for aMussel to boot up the Raspberry Pi

5. Open aMussel programming GUI and follow programming steps described in 2.3.1.

6. Wait for programming to finish.

2.3.1 Windows programming application



(a) Main screen

(b) Status screen - Idle

(c) Status screen - Transfering code

(d) Status screen - Compiling code

(e) Status screen - Programming code

(f) Status screen - Programming done

Figure 2.2: Windows programming application

To make the process of reprogramming robotic swarm easy, an automated programming application was made. The application was made in Visual Studio, using the Visual C++ programming language.

The programming procedure is divided in five steps:

- Choosing a programming configuration
- Starting the programming procedure
- Transferring the binary file containing the code
- Compiling the program on RaspberryPI for programming the PSoC board over I2C
- Executing the compiled program

As shown in 2.2a), choosing a programming configuration is split in three steps:

- Choosing the appropriate board
- Choosing which aMussels the operator wants to program
- Choosing the binary file containing the code

Before the pressing *Start programming* button in step 2., in the status tab aMussels 1 and 2 are shown to be in waiting mode (2.2b)). After the programming procedure has started, the status indicator of each aMussel is indicating that the transferring procedure has started (2.2c)). In case of an unsuccessful file transfer, the status indicator shows "Network error", as seen in aMussel2's status(2.2d)). The compiling step is shown in 2.2d in aMussel1's status. In the final stage of the programming procedure, the progress bar indicates the percentage of flash memory currently transferred to the PSoC board (2.2e)). After successful programming, the status indicator of aMussel1 is in the "Done" state and the aMussel should start executing the newly programmed code.

Chapter 3

Testing device functionality

For individual device functionality test a special task which interprets commands sent through serial port and calls corresponding API functions has been devised. It is called `test_function` and is located in `main.c` of `MU31S` project.

To send commands, a serial connection to MB's general purpose UART needs to be established. All of the commands and debugging messages are then sent through that link.

To test the functionality, a special sequence of messages has been defined. The controls are found in the table 3.1.

Table 3.1: MB testing commands

Bootloader		
b		enter Bootloader
Power board		
P	1	switch to main battery
P	2	switch to backup battery
P	3	disable charging
P	4	get main battery voltage
P	5	get backup battery voltage
P	6	get main battery current
P	7	get backup battery current

P	8		get supply current
P	9		get main battery's charging status
P	A		get backup battery's charging status
P	B		enable charging
Electric sense			
e	b		esense board program
e	0		esense board init
e	s		esense board 's' command
e	m		esense board 'm' command
e	+		esense board '+' command
e	-		esense board '-' command
e	a		esense board get angle
e	d		esense board get distance
Turbidity sensor			
t			Get turbidity sensor reading.
GSM/GPS module			
g	c		call a number defined in test function
g	h		hang up
g	s		send SMS to a number defined in test function
g	m		receive SMS
g	u		list all unread SMS
g	a		list all SMS
g	l		clean read SMS
g	N		power on GPS module
g	F		power off GPS module
g	f		get GPS fix status
g	k		get GPS data
SPI IMU			
p	a		get accelerometer data
p	g		get gyroscope data
p	m		get magnetometer data
p	c		calibrate imu
LEDs			
l	r	num	turn red with intensity *num*
l	g	num	turn green with intensity *num*
l	b	num	turn blue with intensity *num*

l	o		turn off
l	l	num	turn into rgb defined by *num*
Buoyancy motors test			
m	u		go up
m	d		go down
m	s		stop
Pressure and temperature sensor			
r	r		reset
r	p		get pressure
r	t		get temperature
Scenarios			
s	x		power Pi on (with faulty scenario on purpose)
s	C		turn off Pi (following the procedure)
s	Z		turn off Pi (on pin)
s	1		experiment 1
s	2		experiment 2
s	3		experiment 3
s	4		experiment 4
...			
x			Cancel currently running scenario.