

CubeSat/Femtosatellite Command and Communications System User Guide

Aymen Benylles

January 2021

Contents

1	Introduction	3
2	Using the Development Boards	3
2.1	IDE and Project Setup	3
2.2	Running the Code	5
2.3	Connecting the MicroSD and CubeSat Device	6
2.4	WSN Operation	6
2.5	Making Software Changes	8
3	Custom PC/104 Board	9
4	Summary	9

List of Figures

1	Project environment for ground station software.	4
2	User prompt during ground station startup.	5
3	Ground station UART terminal display when a successful link is recorded.	7
4	Ground station UART terminal display when the CubeSat fails to establish a link with the femtosatellite.	7

1 Introduction

This document is intended to be used by any member of the Emerging Space Technologies (EST) team at the University of Glasgow to be able to use the hardware and software created by Aymen Benylles for their MEng project entitled “CubeSat/Femtosatellite Command and Communications System”.

The student wrote code that enables the communication between three LAUNCHXL-CC1310 development boards, as well as designing a PCB that conforms to the PC/104 standard.

2 Using the Development Boards

2.1 IDE and Project Setup

The usage of the code the student wrote should be very simple once the IDE is setup correctly. The first step is to download Code Composer Studio (CCS), the IDE by Texas Instruments (TI) that the student used for this project [1]. Once this has been done, the user can then run any project they wish. It is recommended that an existing project by TI is imported into the IDE then flashed onto the development boards in order to make sure that both are setup correctly. TI has plenty of information available on how to do this, so the student has instead linked the web page where all of this information can be found [2]. Once this is done and both the development boards and IDE have been confirmed to be working correctly, the user can then begin to build their own projects. In this case, if the user has a new CC1310 launchpad and wishes to load the student’s code onto it, they can obtain the .c and .h files from GitHub and add them to the project environment in CCS. Since the main.c file in all of the “no_rtos” examples are the same, the user only needs to add the .c and .h files for each device. For example, if the user wishes to program a development board (or custom-made PCB) for the ground station, they could import one of the EasyLink examples made by TI, and add the “GroundStation_nortos.c” and “GroundStation_nortos.h” files from the GitHub repository (<https://github.com/AymenB98/CubeSat-PCBSat-Command-and-Communication>). These should

be added in place of the example's source code. For example, if the user imports the "rfEasyLinkEchoTx_nortos" example, they can simple replace the "rfEasyLinkEchoTx_nortos.c" file with the "GroundStation_nortos.c" file and add the "GroundStation_nortos.h" also [3]. Once this is completed and the project is setup correctly, the user can debug the project and thus program the device. The user's project environment in CCS should look similar to figure 1.

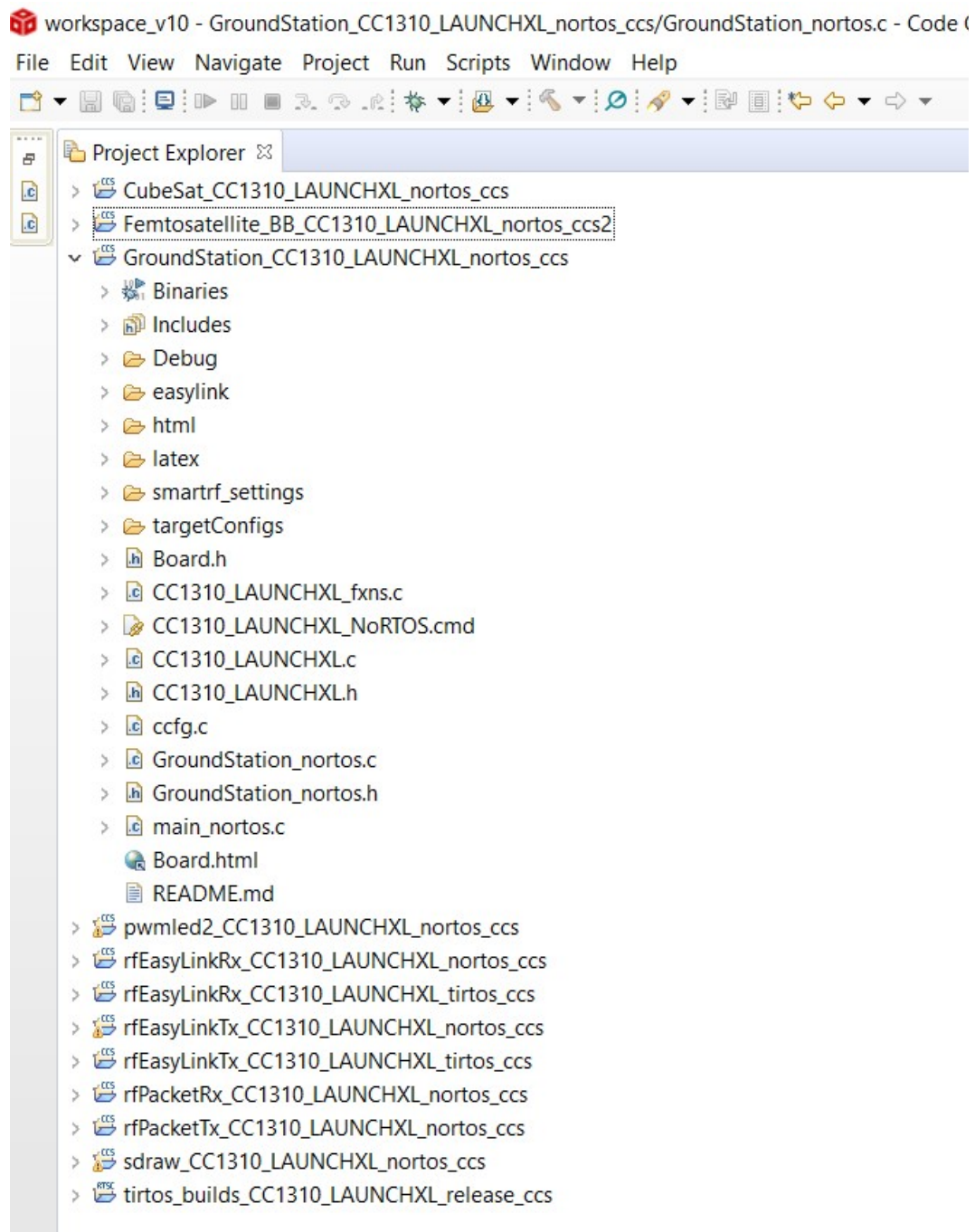


Figure 1: Project environment for ground station software.

2.2 Running the Code

When the code has been successfully built and loaded onto the appropriate target devices, the user will be able to see the wireless sensor network designed by the student work. There is a terminal built into code composer studio, but the user can also use an app such as PuTTY to view the UART display from each device [4]. This UART display has information about each of the data transmissions, and is therefore very useful for debugging and understanding the operation of the WSN. The settings used for this serial session can be seen in [4].

Upon starting the ground station serial session, the user will get prompts on what commands they wish to send, and to which femtosatellite. These can be seen in figure 2.

```
Enter number of commands you wish to send:
1
Enter the sleep time between commands(s):
2
Enter the desired femtosat address:

a: 0xDE
b: 0xBB
c: 0xFE
d: 0xBC
e: 0xCD
b
```

Figure 2: User prompt during ground station startup.

The code has been written such that the user can see what they have entered for each prompt. After each key-press, the next prompt immediately appears. If the user enters invalid data, they are asked to restart the device. In this case, all they need to do is press the reset button next to the USB input on the LaunchPad.

Invalid inputs include more than three commands, a sleep time between commands of more than 9s, and a letter not present on the femtosatellite address list.

2.3 Connecting the MicroSD and CubeSat Device

Although the code has been written in such a way that the WSN can function with or without the microSD connected to the CubeSat, if the user wishes to test this functionality they can. The student recommends that the user connects a microSD card to the CubeSat device via the Pmod microSD board [5]. The user simply needs make the following connections in table 1 below:

Pmod Pin No.	CC1310 LaunchPad pin
1	DIO21
2	DIO9
3	DIO8
4	DIO10
5	GND
6	VCC (3V3)

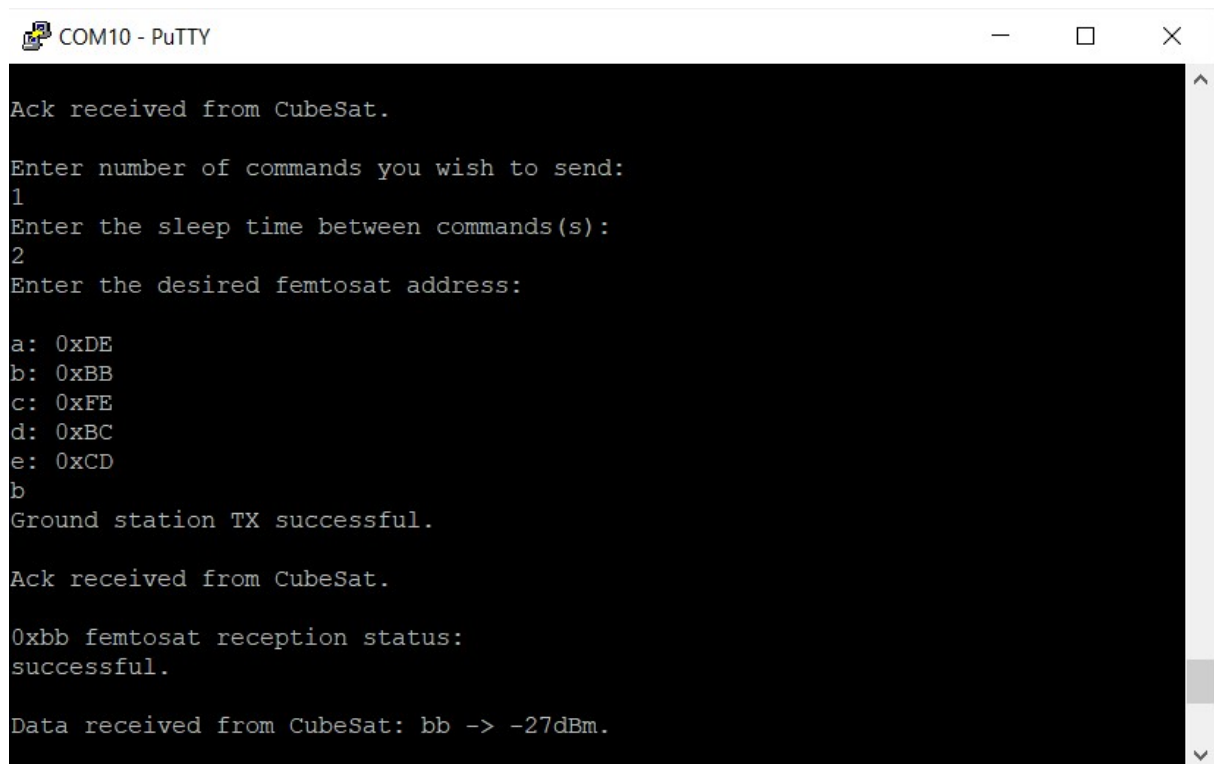
Table 1: Pin connections for CubeSat CC1310 LaunchPad and Pmod MicroSD Slot.

These connections for the CC1310 LaunchPad follow the Board_SD0 row in the Board.html file [6]. This is stated in the SD (raw) TI driver example [7]. The user is encouraged to observe the “README” for the SD (raw) example if they are unsure about these connections [7]. They can also look at [5] to see what each of the connections on the Pmod board are.

2.4 WSN Operation

Now that the UART terminal is up and running and all of the devices are connected to power, the user can expect to see the WSN working.

If the user wishes to reduce the amount of USB ports they are using, they can simply use a power bank with a USB output to power the femtosatellite. This, of course, means that they will not be able to view messages that the femtosatellite is displaying, but if the WSN is working, the ground station should be showing the received signal strength of a message sent from the femtosatellite to the CubeSat. Figure 3 below shows what the user will see if the ground station successfully sends commands to a femtosatellite.



```
COM10 - PuTTY

Ack received from CubeSat.

Enter number of commands you wish to send:
1
Enter the sleep time between commands(s):
2
Enter the desired femtosat address:

a: 0xDE
b: 0xBB
c: 0xFE
d: 0xBC
e: 0xCD
b
Ground station TX successful.

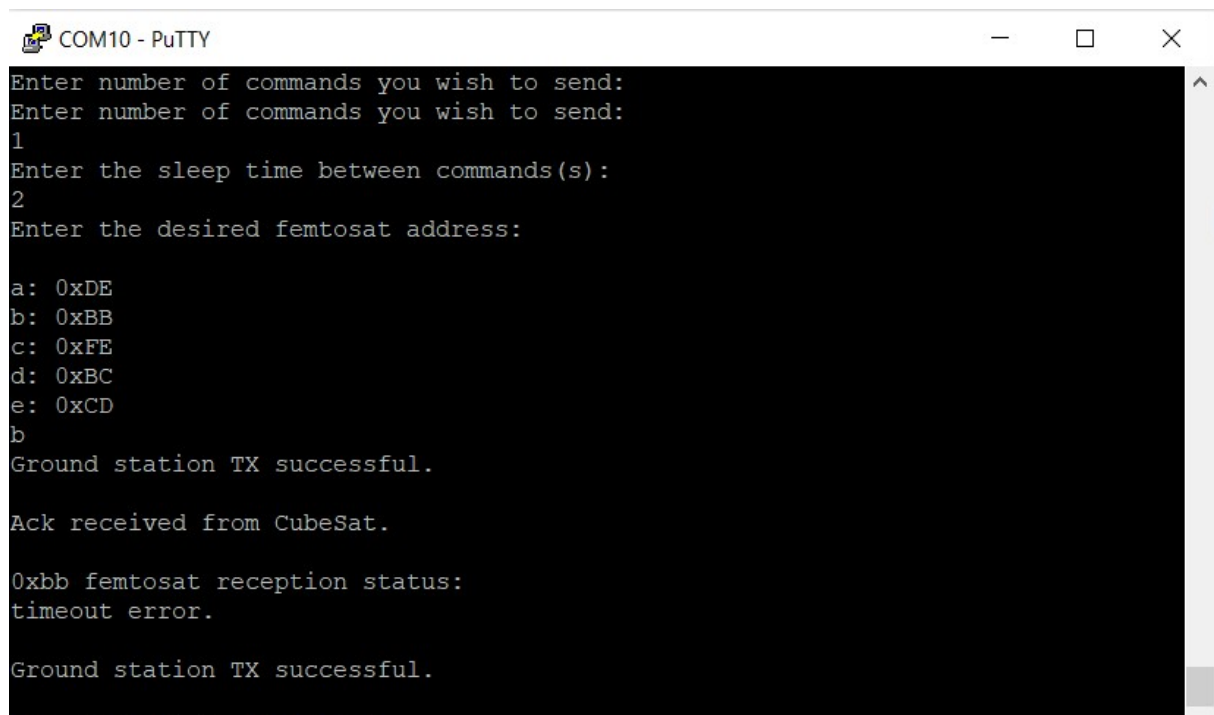
Ack received from CubeSat.

0xbb femtosat reception status:
successful.

Data received from CubeSat: bb -> -27dBm.
```

Figure 3: Ground station UART terminal display when a successful link is recorded.

Figure 4 is an example of the ground station display when the CubeSat does not successfully establish a link with the femtosatellite.



```
COM10 - PuTTY

Enter number of commands you wish to send:
Enter number of commands you wish to send:
1
Enter the sleep time between commands(s):
2
Enter the desired femtosat address:

a: 0xDE
b: 0xBB
c: 0xFE
d: 0xBC
e: 0xCD
b
Ground station TX successful.

Ack received from CubeSat.

0xbb femtosat reception status:
timeout error.

Ground station TX successful.
```

Figure 4: Ground station UART terminal display when the CubeSat fails to establish a link with the femtosatellite.

2.5 Making Software Changes

When the EST team make changes to this code as their work evolves over the next few months, there are several things that they should note. With regards to changing the output power of the RF transmission, the user should go into the “smartrf_settings.c” file and change the code as follows (obtained from SmartRF Studio) [8]:

```
// CMD_PROP_RADIO_DIV_SETUP
// Proprietary Mode Radio Setup Command for All Frequency Bands
rfc_CMD_PROP_RADIO_DIV_SETUP_t RF_cmdPropRadioDivSetup =
{
    ...
    .txPower = 0xA73F,
    ...
}
```

Smart RF Studio states that the “define CCFG_FORCE_VDDR_HH = 1 in ccfg.c” in order to set the TX power to its maximum of 14dBm [8]. SmartRF Studio can be used to see what value should be used for a certain TX power (in dBm) [9]. Instructions on how to use SmartRF Studio to define RF settings for a device can be found in [10]. In this file, the address(es) that the user wishes to set for address filtering should be defined in the following section of the “smartrf_settings.c” file [11]:

```
// CMD_PROP_RX
rfc_CMD_PROP_RX_t RF_cmdPropRx =
{
    ...
    .address0 = 0xAA,
    .address1 = 0xBB,
    ...
}
```

This would mean that the device is able to receive packets from devices with addresses 0xAA and 0xBB (in this case, the CubeSat, since it is the only one that has to communicate with more than one device).

The frequency of the transmission can be changed in the source code for each device (e.g. “CubeSat_nortos.c”) by doing the following [12]:

```
EasyLink_setFrequency(433000);
```

This sets the frequency of RF operations to 433MHz since the argument is in kHz [12].

3 Custom PC/104 Board

The PCB the student completed for this project follows the dimensions shown in figure 3 of [13]. This means that all other boards used in the stack must conform to the same dimensions as this one, otherwise a change is needed.

4 Summary

In conclusion to this user guide, any engineer working with this project will be able to use this, along with the many helpful resources made available by TI, to make run, test and change certain elements of the student’s work as they see fit. Links are provided that give full and exhaustive detail about each tool that is necessary for the use of this student’s work. It is worth noting that the PCB designed for this project has not been tested practically, so an ideal lab power supply should be used first to make sure that it works correctly upon manufacture.

References

- [1] Texas Instruments Incorporated, *Code composer studio*, (10.1.0). [Online]. Available: <https://www.ti.com/tool/CCSTUDIO>, [Accessed: 14.1.2021].
- [2] —, *Cc1310 launchpad*, www.dev.ti.com. [Online]. Available: https://dev.ti.com/tirex/explore/node?node=AMoBnKEgI1TSJ3K1G3NR6w__eCfARaV__LATEST, [Accessed: 3.1.2021].
- [3] —, *Rfeasylinkechotx*, www.dev.ti.com. [Online]. Available: https://dev.ti.com/tirex/explore/node?node=AC75puB6X9pozczQohetPcw__eCfARaV__LATEST, [Accessed: 2.1.2021].
- [4] —, *Uartecho*, www.dev.ti.com. [Online]. Available: https://dev.ti.com/tirex/explore/node?node=A04fsiTT3v-z16B2uN2vsQ__eCfARaV__LATEST, [Accessed: 2.1.2021].
- [5] Digilent, *Pmod microsd reference manual*. [Online]. Available: <https://docs.rs-online.com/9265/A700000006690481.pdf>, [Accessed: 2.1.2021].
- [6] Texas Instruments Incorporated, *Cc1310 simplelink launchpad settings & resources*, www.dev.ti.com. [Online]. Available: https://dev.ti.com/tirex/explore/node?node=ABI00YaB9BwpbLLhkKLOQA__eCfARaV__LATEST, [Accessed: 2.1.2021].
- [7] —, *Sdraw*, www.dev.ti.com. [Online]. Available: https://dev.ti.com/tirex/explore/node?node=AAGlfE5jrCpSBeq5KmP8Lg__eCfARaV__LATEST, [Accessed: 2.1.2021].
- [8] —, *SmartRF studio 7*, (2.19.0). [Online]. Available: <https://www.ti.com/tool/SMARTRFSTM-STUDIO>, [Accessed: 14.1.2021].
- [9] —, *SmartRF studio*, www.ti.com. [Online]. Available: <https://www.ti.com/tool/SMARTRFSTM-STUDIO>, [Accessed: 3.1.2021].
- [10] —, *Command code export*, software-dl.ti.com. [Online]. Available: https://software-dl.ti.com/lprf/smartrfstm_studio/docs/help/html/cmd_code_export.html, [Accessed: 3.1.2021].
- [11] —, *SmartRF_settings.c*, www.dev.ti.com. [Online]. Available: https://dev.ti.com/tirex/explore/node?node=AJFUN5hkRMowyTDsQcEhUQ__eCfARaV__LATEST, [Accessed: 14.1.2021].

- [12] —, *Easylink api reference*, software-dl.ti.com. [Online]. Available: http://software-dl.ti.com/simplelink/esd/simplelink_cc13x0_sdk/3.20.00.23/exports/docs/proprietary-rf/proprietary-rf-users-guide/easylink/easylink-api-reference.html, [Accessed: 3.1.2021].
- [13] PC/104 Embedded Consortium, *Pc/104 specification*, pc104.org, Mar. 1992. [Online]. Available: https://pc104.org/wp-content/uploads/2015/02/PC104_Spec_v2_6.pdf, [Revised Oct. 2008], [Accessed: 8.1.2021].