Reaction Wheel Firmware Update/Re-Flash Process

Requirements

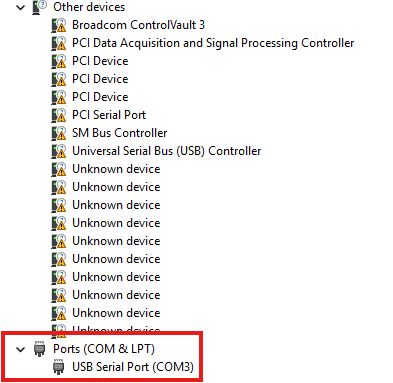
* Windows 7 or newer
* .Net v4 or newer
* Driver from <https://ftdichip.com/drivers/>
  + As recommended by DS\_TTL-232R\_CABLES (Under surface Flexstat\_Documentation folder)
* CubeSupport Tool (**Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Software\sw-bundle-master-v4.5.0.4\tools\cube-support**.)

Reset Firmware Location

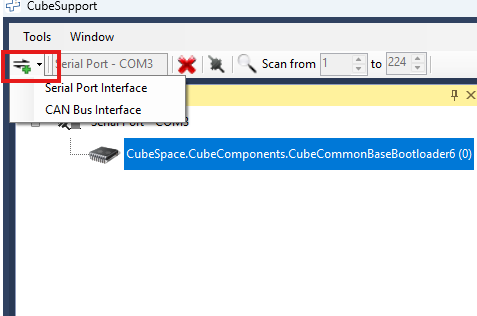
* app-control-program-cube-wheel-2-3.4.2-freertos-release-bin.cs (**Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Software\sw-bundle-master-v4.5.0.4\firmware\cube-wheel-2\control-program**)

Instructions

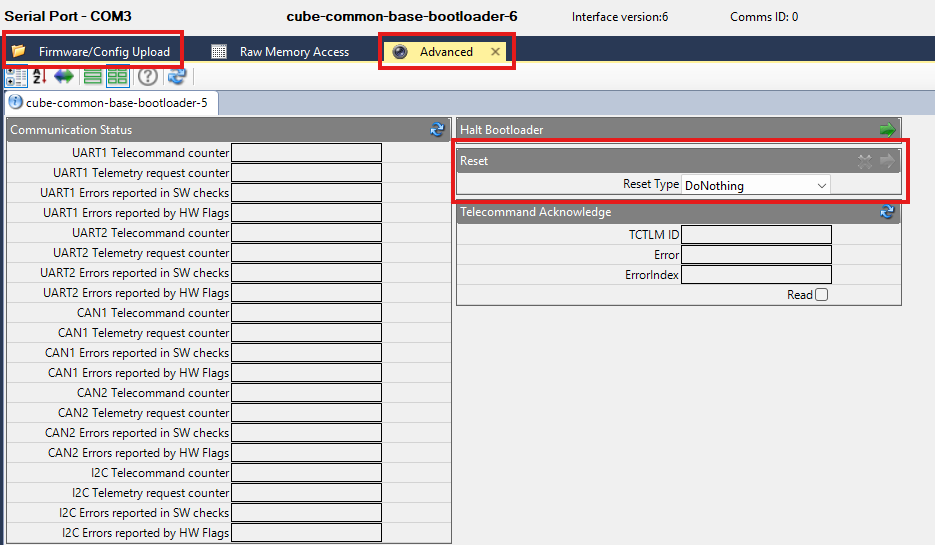
1. Ensure Reaction Wheel(RW) has power and the device is displayed under your COM ports (most likely COM3). If you do not have the TTL-232R driver installed then it will display under Other Devices with the name or a similar name to TTL-232R.



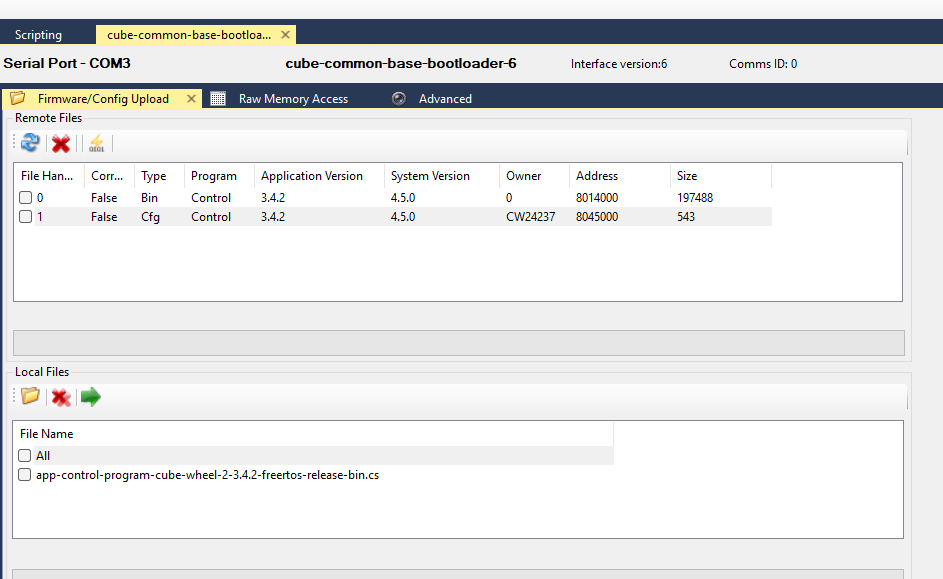
1. Launch CubeSupport software (location can be found under Requirements) and select Serial Port as your connection option (note: the USB Bus option only threw errors for me). It should auto select the correct Baud but if not it is 921600 and then select the connect button(to the right of the red X).



1. Select the Advanced tab from the resulting window and locate the Reset option, select Reset to bootloader and send the command. Once your device resets it will be in the bootloader option and you will have access to the firmware tab.



1. In the local files section you will click on the folder and navigate to the provided firmware .cs file. Click the green arrow to begin uploading the firmware. Wait for the upload to be complete then you can select the file from the Remote Files section and click the flash symbol to jump to the firmware in memory. (Note: If the firmware is a copy of the currently uploaded firmware it will just replace the identical remote file).



Refer to the following document for source of instructions.

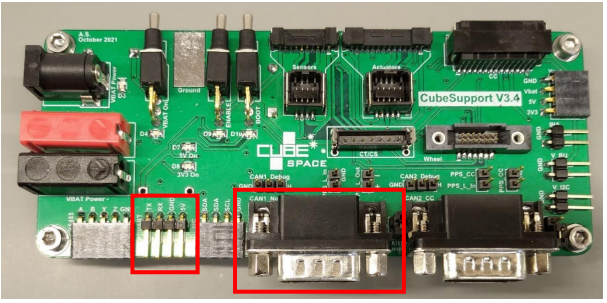
CS-DEV.UM.CU-01 CubeSupport HMI Ver.1.03 (**Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Software\sw-bundle-master-v4.5.0.4\docs\tools\cubesupport**)

Alternate Useful Sources/Information

**CS-DEV.UM.CW-01 CubeWheel Gen 2 UM Ver.1.00 (Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Documentation)**

3.4.2.1 CAN and UART

CubeSupport PCB breaks out the communication interfaces to user-friendly headers. The CubeWheel has two main communication interfaces available, CAN and UART. These channels are made available on CubeSupport PCB as the “CAN1 Nodes” and “UART” headers respectively, as indicated in Figure 4. A PC can interface with the CubeWheel through either of these channels.



**Figure 4: CubeSupport PCB “UART” and “CAN1 Nodes” headers indicated.**

**Note:** The CAN interface is the default communications channel, the UART interface is normally not used and usually remains unconnected. It can be used for reprogramming the CubeWheel software, or as a secondary communications interface.

3.4.3 Communicating with the CubeWheel using CubeSupport PCB and -Application Following the steps described in sections 3.4.1 and 3.4.2, the CubeWheel should now be setup in such a way to allow the user to be able to power it. The user is now referred to [8] for a description of how to communicate with the CubeWheel from a PC via the CubeSupport PCB using the CubeSupport application.

4.8.2 Bootloader updates The CubeSpace bootloader itself is also upgradeable, should a new version become available with features specific to the bootloader. All CubeProducts have a manufacturer provided ROM bootloader. This provides an alternative method for updating firmware; however, this interface is only accessible on the ground. Details on reprogramming the CubeSpace bootloader through the ROM bootloader is supplied in [3].

4.8.3 Software release package CubeSpace supplies a software release bundle together with the initial hardware delivery of the CubeWheel that contains the latest firmware-, bootloader- and configuration file versions of the CubeWheel that should correspond to the firmware versions on the CubeWheel. Subsequent software release bundles are also made available to offer new functions and features, address earlier software issues and so on. Each software release bundle contains a number of files: - The firmware-, configuration- and bootloader files can be found under . - The CubeProducts are available as subdirectories below it, for example - A subdirectory containing the control program (FW) and configuration file(s) for the CubeProduct is available, for example - Another subdirectory containing the bootloader application FW for the CubeProduct is available, for example . Each software release package also contains documentation to explain what is contained in the release.

4.8.4 Steps to update control application and configuration through CubeSpace Bootloader To update the configuration or the application software of the CubeADCS and the CubeWheel in particular, it must first be reset, or power cycled. Upon power-on or reset, the bootloader application will run first. The bootloader uses a backoff timer and will listen for incoming commands for 5 seconds before jumping to (starts executing) the CubeWheel application control program. The user must send a Halt (Halt bootloader) command within 5 seconds of power-on or reset (this happens when opening the connection, as discussed in section **3.4.3**). The boot process is then halted and the CubeWheel will remain in the bootloader application until a JumpToApp (Jump To Application) or Reset command is received. The CubeWheel is now ready to be programmed. The Bootloader Application Manual (see [3]) provides detailed information for interfacing with the CubeSpace bootloader to update the configuration and control application firmware.

**CS-DEV.PD.CW-01 CubeWheel Product Description Ver.1.01 (Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Documentation)**

3.6 Pre-loaded firmware applications

The CubeWheel is supplied with two pre-loaded applications on the unit. The first is a Bootloader and the other is the Control Program. 3.6.1 Bootloader The Bootloader is the first application to run when the CubeWheel is powered on. It has the following features:

• Allows for quick identification through communications messages and protocol that is common across all CubeProducts,

• Allows CubeWheel Control Program and configuration to be (remotely) updated,

• Supports FDIR,

• Exposes Bootloader API to Host Device over communication channels.

3.6.2 Control Program

The control program is the main program of the CubeWheel. Some of the main functions are in support of the CubeComputer or client master node:

• Supports FDIR,

• Supports CubeWheel management (e.g. power, status, setup, and configuration),

• Supports/Implements CubeWheel actuator commands,

• Reports CubeWheel measurement telemetry (wheel speed, current, temperature)

• Exposes Control Program API to host device.

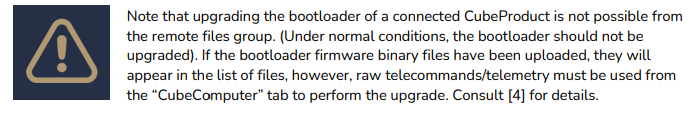
Find CubeSupport tool for firmware and bootloader update under **Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Software\sw-bundle-master-v4.5.0.4\tools\cube-support**.

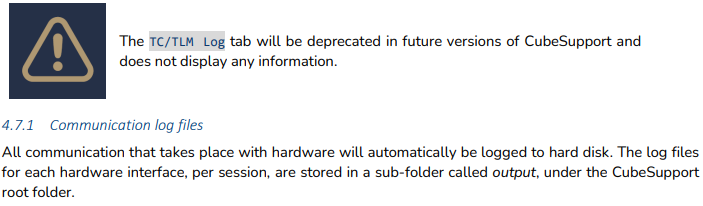
Bootloader

Bootloader File Location

* app-base-bootloader-52-cube-common-1-node-1.7-none-release-src-bin.cs (**Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Software\sw-bundle-master-v4.5.0.4\firmware\cube-common-1-node\base-bootloader-52**)

CS-DEV.UM.CU-01 CubeSupport HMI Ver.1.03.pdf (**Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Software\sw-bundle-master-v4.5.0.4\docs\tools\cubesupport**)





4.8.7

Next, configure the CubeProduct to enter the ST ROM bootloader using these steps:

1. Power down the CubeProduct, by setting its enable line low
2. Set boot pin to high
3. Power up by setting the enable line high.

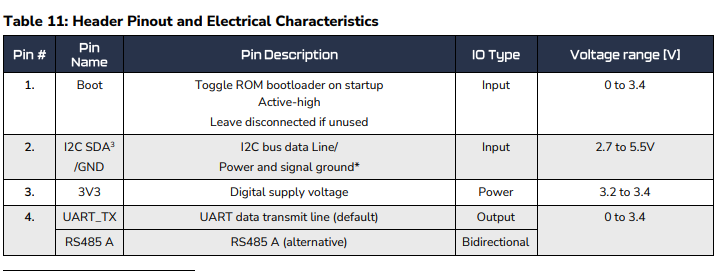
The ST ROM bootloader can be accessed through either the UART or CAN communications interface. The selection is made similar to performing a connection to the CubeSpace base bootloader or control program. The communication interface selection is made from the dropd

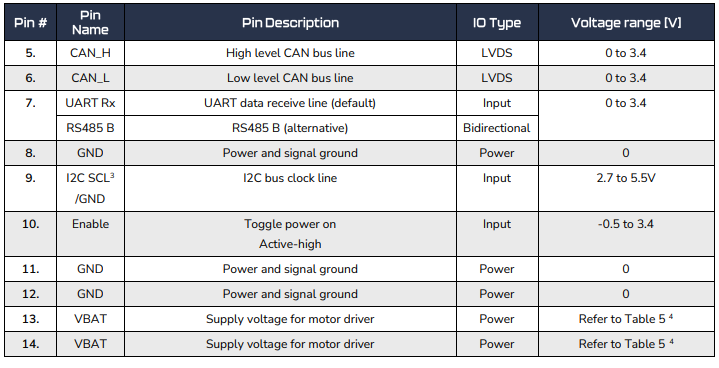
Once connected, the UI updates to show information for the connected microcontroller, as in Figure 20 (no picture available)

To program the base bootloader onto the CubeProduct’s microcontroller, click on “file upload” and navigate to the correct base bootloader firmware file for the applicable microcontroller. Next, select the file in the “File Upload” list and click on “upload” . The progress bar shows the state of the current transfer. Once it reaches 100%, the file is completely programmed, and the user can proceed to connect and communicate with the base bootloader using the steps in section 5.

29197e76-cd6d-409e-8896-e29b4813ed2c.pdf (**Flexstat\_Documentation**)

2.4 Header Pinout and Electrical Characteristics





Coincidence that the TTL-232R connector has same number of lines?? Is it referencing the TTL-232R??

CS-DEV.AMNL.BL-01 Bootloader Application Manual Ver 5.00.pdf (**Flexstat\_Documentation\Hardware Documentation\Cubespace RW\Software\sw-bundle-master-v4.5.0.4\docs\bootloader**)

Upgrading the CubeSpace bootloader is only possible via the manufacturer ROM bootloader and is not expected to be used in orbit.

2.1.1

Default Configuration A default configuration is embedded in the bootloader binary and is never modified by the application. Upon first boot, the bootloader assumes the default configuration. Modifications to the configuration are stored in a separate location in flash. Once the configuration is modified from the default via TCTLM, the modified configuration becomes the first source of configuration. The configuration is CRC protected. Upon boot, if the configuration fails CRC validation, the bootloader reverts to using the default configuration. This event will be indicated by the Config Init flag in the Errors telemetry. Note that the CubeComputer bootloader stores a redundant configuration (see section 2.11.1) and will therefore only assume the default configuration if both the primary and redundant configuration are invalid/corrupt. In the event of the user’s configuration becoming corrupt, the user must be prepared to communicate with the CubeProduct using the default values to, at a minimum, restore the configuration to the desired values. Information on how to artificially induce configuration corruption so that the recovery implementation can be tested, refer to Appendix: Testing Configuration Corruption Recovery. If the bootloader reverts to using the default configuration, the configuration of the communications interfaces for the control program application may be affected as well, if the control program is configured to use the bootloader’s configuration (see section 2.1.4).

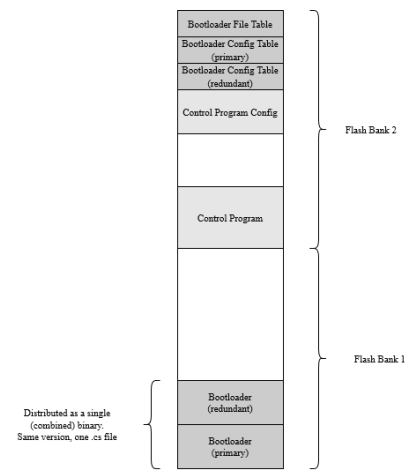
2.11.2

The bootloader binary is write-protected and will never be erased/written, unless a bootloader upgrade is performed. Therefore, the purpose of the redundant binary is to protect against ECCD errors induction by radiation.

The bootloader configuration can be written to at any time, and the redundant configuration protects against ECCD errors.

The bootloader file table is written to when a file is uploaded, or when the default jump target is set. There is not a redundant file table because the file table is not critical.

All ECC errors are reported via the Warnings telemetry.



Use Erase Memory Command to delete the size of the bootloader??? How to write to this then? Memory is write protected.

8

Appendix: Testing Configuration Corruption Recovery This section is not applicable to CubeComputer. To test configuration corruption for CubeComputer, follow section

3.1. This appendix describes how to artificially induce configuration corruption, such that the bootloader will revert to using the default configuration. Thus, the user may test their implementation of the configuration recovery to the desired values. The testing process involves using the erase memory feature (see 2.9.3) to erase the current configuration.

Step 1: Ensure that the current configuration is not the same as the default configuration. This can be done using the CubeSupport app. The most easily noticeable configuration item would be the serial number string.

Step 2: Erase the configuration. Use the Erase Memory Setup command to erase the configuration. If using the CubeSupport app, this command is only in the “Advanced” tab. The setup parameters should be as follows:

• CubeAuriga: Address = 134316032 (0x8018000) Size = 1

• CubeStar: (R5 bootloader) Address = 136298496 (0x81FC000) Size = 1

• All other CubeProducts (52 bootloader) Address = 134737920 (0x807F000) Size = 1

Step 3: Confirm that the erase was successful. Request the State telemetry. The parameters should be as follows:

• App State = StateIdle

• Previous App State = StateBusyErase

• Error Code = 0. If this is non-zero, report to CubeSpace, and retry step 2.

• All other parameters are don’t care.

Step 4: Command a soft reset. This will cause the bootloader to reload the configuration. At this point the CRC of the configuration will fail, and the bootloader will assume the default configuration.

Step 5: Confirm the configuration is now set to the default values. Request the Config telemetry.

Step 6: Recover the desired configuration.

This is implementation specific.

The most noticeable difficulty with recovering the configuration is that the CubeProduct may no longer be using the expected protocol. Implementing multiple protocols to handle this one scenario may not be desired, therefore it is recommended that the configuration recovery be implemented using hard-coded raw bytes that have been encoded using the default protocol on a given interface. It is recommended that the encoded bytes of the Identification telemetry request be stored, as this can be used as a sign-of-life, and confirmation that the loss of communication is indeed caused by configuration corruption. The desired configuration should be stored as the encoded bytes of the Config command, using the default protocol. These raw bytes can then be written directly to the interface peripheral. See [2] for information on the communication protocols.