

4 SOFTWARE

The software associated with the Interface Emulator is split into two 'sketches' written for the Arduino programming language (based on 'wiring') and the Arduino development environment (based on 'processing'). Sketches are the name for programs and the language is a set of libraries and macros built on top of the C programming language. Within a sketch, C language convention can be used, however additional functions are also available.

More information on the Arduino is available at: <http://www.arduino.cc/>.

- **UKube_Plt_Emulator_V1_4.pde**
Developed for the Platform Emulator Arduino, this sketch emulates the interfaces of the MIC on the Platform Arduino.
- **UKube_Pay_Emulator_V1_4.pde**
Developed for the Payload Emulator Arduino, this sketch provides an example of a simple Payload Controller.

Both programs interface to the user via a virtual COM port connection via USB to a terminal emulator program running on a PC. This allows direct control over the Platform Emulator for sending commands to the payload and controlling the functionality built into the emulator board. The terminal program can also be used to set response flags on the payload Arduino which will be read by the platform.

The Platform and Payload Emulator Arduinos communicate through the Payload Protocol & Packet Definition (AD-17) using both I2C and SPI data buses. The as per AD-17 I2C data bus is the primary communications method.

4.1 Function

- Implement the data exchange / payload operations concept described conforming to the I2C, SPI protocols and Clyde Space commands and packet standards.
- Demonstrate the relevant parts of the Payload Interface Requirements (described in Sec.6).
- Format commands, telemetry and payload data to be relayed through the USB UARTs, which can be input and displayed in a terminal viewer.
- Power switch command and telemetry, on/off, current sense, and status.
- *Future compatibility with scripts received through the UART.*

4.2 Arduino IDE Installation

- Install the Arduino Integrated Development Environment (IDE) according to the instructions provided on their website (<http://arduino.cc/en/Guide/HomePage>). When the procedure tells you to plug in the Arduino connect a USB cable to the **Platform Emulator**. Do not proceed past step 8, if you follow the instructions to upload the test program it will overwrite the emulator firmware programmed on the emulator.



Do NOT complete step 9 of the Arduino website instructions. This will overwrite the emulator firmware.

- Once the IDE and drivers have been installed close the program and copy the folder “TWI_long” from the software directory on the provided CD (“Data Pack\Libraries”) to the “libraries” directory in your installed Arduino IDE directory.
- Copy the folders “UKube_Plt_Emulator_V1_4” and “UKube_Pay_Emulator_V1_4” to your PC's hard drive. These can then be opened and viewed using the Arduino IDE. Do not upload the sketches to the Arduino at this point as both the Platform and Payload Emulators are supplied pre-programmed.



The Arduinos were programmed using version 0022 of the IDE.

4.3 Terminal Interface

Both Platform and Payload Emulator Arduinos require an RS232 connection to a terminal program in order to provide feedback to and accept commands from the user. The terminal program should be configured as follows:

- Baud rate 115200bps
- 8N1 packet format
- Should allow two terminal instances to be run at once (or two separate PCs should be used, each running one instance).

It is recommended that the built in terminal emulation function of the Arduino IDE is used to gain familiarity with the Platform and Payload emulation software as its compatibility with the programs has been tested.



To allow two instances of the terminal to be run, two instances of the `arduino.exe` must be initialised. This can be recognised by the presence of two separate tabs on the Windows taskbar.

As a convention any commands received by the Payload Emulator from the Platform Emulator will be preceded by “>>” in the terminal interface. Any commands issued at the payload terminal interface will be confirmed by a message preceded by “<<”. Any data not preceded by “<<” or “>>” will either be displaying data or displaying a menu option.

4.4 Commands

Commands are entered at the terminal window. Menu selections are entered by pressing the required number followed by the 'enter' key. Where values are to be entered the software will specify the format and number of digits to be entered and this guidance must be followed.

If you wish to exit a menu, enter a number which is not defined for the menu selection options. E.g. if a menu has options 1-6 listed choosing 7 will return to the main menu without taking any action.

4.4.1 Platform Emulator User Menu

When first powered on the Platform Emulator will display the top level menu within the terminal node. The menu tree for the Platform Emulator software is illustrated below.

1. Platform Commands
 1. Switch 3V3 Bus
 2. Switch 5V Bus
 3. Switch 12V Bus
 4. Switch Battery Bus
 5. Display Current Measurement
2. Payload Commands
 1. Payload Initialise
 2. Payload Status
 3. Payload Update
 4. Payload Parameter Write
 5. Payload Parameter Read
 6. Payload Priority Data Transfer
 7. Payload Data Transfer
 8. Payload SPI Data Transfer
 9. Payload Shutdown
3. Automatic Emulation Mode
4. Emulation Settings
 1. Set Slave I2C Address
 2. Set Status Update Period
 3. Set Active Power Bus

4.4.2 Platform Commands Submenu

The Platform Commands submenu allows the operator to control the functionality of the power switches built into the emulator board. When the Platform command is first selected from the top level menu a sub menu will be displayed. Along the top of the sub menu a bar will display the status of each of the power buses on the emulator board. From left to right these show 3V3, 5V, 12V and the Battery bus. A one indicates the bus is powered.

- Switch 3V3 bus: This will toggle the current status of the 3V3 bus. Note that this will toggle the status of the bus as measured by the voltage status line on the Arduino and not based on the last status set by the command menu (if an overcurrent trip has shutdown the bus, selecting this menu option will power it up).
- Switch 5V bus: This will toggle the current status of the 5V bus. Note that this will toggle the status of the bus as measured by the voltage status line on the Arduino and not based on the last status set by the command menu (if an overcurrent trip has shutdown the bus, selecting this menu option will power it up).
- Switch 12V bus: This will toggle the current status of the 12V bus. Note that this will toggle the status of the bus as measured by the voltage status line on the Arduino and not based on the last status set by the command menu (if an overcurrent trip has shutdown the bus, selecting this menu option will power it up).
- Switch Battery bus: This will toggle the current status of the Battery bus. Note that this will toggle the status of the bus as measured by the voltage status line on the Arduino and not based on the last status set by the command menu (if an overcurrent trip has shutdown the bus, selecting this menu option will power it up).
- Display current measurement: This will display a snapshot of the current being drawn on each of the buses.

4.4.3 Payload Commands Submenu

The Payload Commands submenu allows the operator to send the individual commands as specified in the Payload Protocol and Packet Definition Document (Ad-17) to the Payload Arduino. All commands are sent over the I2C bus and include a 16 bit CC-ITT checksum.

- Payload Initialise: This is required to setup the payload and must be issued to the Payload Emulator before any other commands will be accepted. Once the initialisation command is sent the Platform Emulator will begin to poll the payload for a status update at the rate specified in the Emulation settings menu (or at a default period of 30 seconds if not specified).
- Payload Operation Status: This command is issued automatically at a period defined by the Emulator Settings menu (or a default of 30 seconds) once the platform is initialised but can also be issued manually at any time. The Payload Emulator will respond with the data outlined in the protocol definition and this will be displayed on the Platform terminal program.
- Payload Operation Update: When issued manually from the Payload Command menu the user will be prompted to enter the desired payload mode and payload operation flags. These values should be represented by 2 and 4 hexadecimal digits respectively. Once all the required data has been entered the Platform Emulator will issue an update command to the Payload Emulator. The Priority Data limit and Mass Memory Data limit are hardwired in the Platform Emulator to 16776960 Bytes and 268435456 Bytes. The available memory and priority memory counters will be updated based on the number of data transfers which have taken place. It should be noted that these values are just counters and no mass memory storage is implemented on the emulator board.

- **Payload Parameter Write:** The user will be requested to choose a parameter ID to update from the menu. The Parameter IDs for the Payload Emulator are hardwired to 1-8 in the Platform Emulator software. Once a parameter ID is chosen the user will be prompted to enter 4 hexadecimal digits which will be used to update the parameter value in the Payload Emulator.
- **Payload Parameter Read:** The user will be requested to choose a parameter ID to update from the menu. The Parameter IDs for the Payload Emulator are hardwired to 1-8 in the Platform Emulator software. Once a parameter ID is chosen the Platform Emulator will transmit a request to the Payload Emulator and the Payload Emulator will return the value associated with that parameter
- **Payload Priority Data Transfer:** The Platform Emulator will request 260 bytes of priority data from the Payload Emulator. The data transmitted will consist of 2 command bytes, 256 data bytes and 2 CRC bytes. The data bytes will be fixed as the values 0-255. For testing purposes, if the payload priority data available parameter on the Payload Emulator is less than 256 bytes the transfer will still take place and the priority data available parameter will be reduced to 0. If the data received from the Payload Emulator data fails its CRC check the Platform Emulator will request data up to a total of three times (with a 3ms delay between attempts) until the CRC is passed. If the Payload Emulator data passes its CRC check the platform will deduct 256 from the priority data available parameter and will then return the data CRC to the Payload Emulator by way of acknowledgement.

In the Payload Emulator implementation if this acknowledgement does not match the CRC generated when the data was sent it will assume the original data never reached the platform and will therefore not update the priority data waiting flags. The implementation of error handling for this acknowledgement CRC left to the Payload designer and this is just an example scenario.

- **Payload Data Transfer:** The Platform Emulator will request 260 bytes of data from the Payload Emulator. The data transmitted will consist of 2 command bytes, 256 data bytes and 2 CRC bytes. The data bytes will be fixed as the values 255-0 for this test. For testing purposes, if the payload data available parameter on the Payload Emulator is less than 256 bytes, the transfer will still take place and the data available parameter will be reduced to 0. If the data received from the Payload Emulator data fails its CRC check the Platform Emulator will request data up to a total of three times (with a 3ms delay between) until the CRC is passed. If the Payload Emulator data passes its CRC check the platform will deduct 256 from the priority data available parameter and will then resend the data CRC to the Payload Emulator by way of acknowledgement.

In the Payload Emulator implementation if this acknowledgement does not match the CRC generated when the data was sent it will assume the original data never reached the platform and will therefore not update the priority data waiting flags. The implementation of error handling for this acknowledgement is CRC left to the Payload designer and this is just an example scenario.

- **Payload SPI Data Transfer:** The Platform Emulator will request 260 bytes of data from the Payload Emulator. The data transmitted will consist of 2 command bytes, 256 data bytes and 2 CRC bytes. The data bytes will be fixed as the values 255-0 for this test. The data will be transmitted from the Payload Emulator to the Platform Emulator over SPI. For testing purposes, if the payload data available parameter on the Payload Emulator is less than 256 bytes, the transfer will still take place and the data available parameter will be reduced to 0. If the data received from the Payload

Emulator data fails its CRC check the Platform Emulator will request data up to a total of three times (with a 3ms delay between attempts) until the CRC is passed. If the Payload Emulator data passes its CRC check the platform will deduct 256 from the data available parameter and will then return the data CRC to the Payload Emulator over I2C by way of acknowledgement.

In the Payload Emulator implementation if this acknowledgement does not match the CRC generated when the data was sent it will assume the original data never reached the platform and will therefore not update the priority data waiting flags. The implementation of error handling for this acknowledgement CRC is left to the Payload designer and this is just an example scenario.

- Payload Shutdown: The shutdown command prepares the Payload for power off. As the Payload Emulator makes no use of non-volatile memory and has no peripheral circuitry which must be powered down gracefully its implementation is limited to displaying a shutdown command. When the payload shutdown command is issued the Platform Emulator will cease to send periodic Payload Operation Status commands.

4.4.4 Automatic Emulation Mode

Automatic emulation mode sets the Platform Emulator to run through the MIC command loop as defined in Figure 4-13. This loop will operate autonomously with no further input possible from the Platform Emulator terminal.

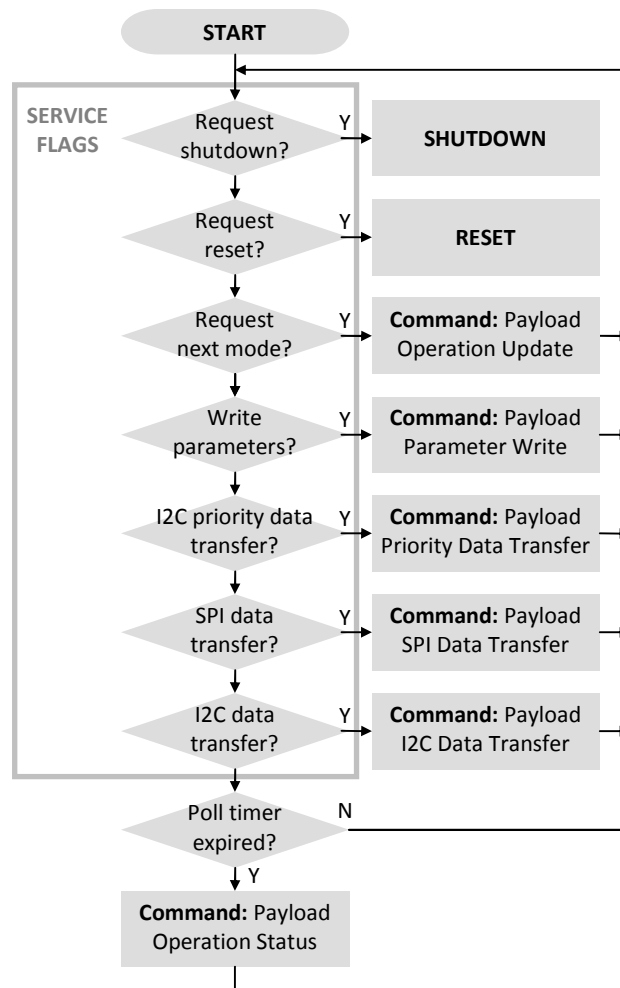


Figure 4-13; Platform Emulator command loop

The Platform Emulator will first power on the active bus as set in the Emulator settings menu (default 5V). If this bus is already powered on, the power will be cycled. Following a power cycle it is necessary to reconnect the payload terminal program

After a 30 second delay the Payload Initialise command will be issued. If the payload does not respond to the initialise command with the appropriate Acknowledgement as defined in AD-17 the platform emulator will wait 10ms then retry. A maximum of three attempts to initialise will be made before the platform emulator reverts to manual mode.

The Platform Emulator will then begin polling the Payload Emulator for status updates with the period defined in the emulator settings menu.

The user can indirectly influence the operation of the MIC command loop by setting request flags through the Payload Emulator terminal interface. The commands which can be set are detailed more fully in Sec . 4.4.6. Due to the performance of the arduino used in the platform emulator only one flag should be set at a time. Setting several flags may result in the second flag failing to acknowledge before the platform emulator timeout. This is due to the time needed to display text to the serial port at the payload emulator. When designing payloads the user should be aware of the fact that while it is possible to request several flags in one status update, these will be serviced by the platform consecutively and the

payload must be capable of responding to the service requested within the 2ms timeout specified in AD-17.

Once a flag has been set, a confirmation message will be displayed on the Payload Emulator terminal interface and the appropriate register will be set. When the Platform Emulator next polls for a status update the bit will be read by the Platform Emulator and the appropriate command will then be issued.

If multiple flags are set they will be processed in the following order:

- Payload Shutdown
- Payload Reset
- Next Payload Mode
- Payload Update
- Priority Data Transfer
- SPI Data Transfer
- Data Transfer

With the exception of Payload Shutdown and Payload Reset, processing a flag will simply delay the processing of any flags lower in the hierarchy. The Payload Shutdown and Payload Reset flags will result in power being cycled and all other pending flags being discarded.

4.4.5 Emulator Settings Submenu

The Emulator Settings Submenu allows the user to configure how the Platform Emulator will interface to any payloads connected to it.

- I2C Slave address: Specifies the I2C slave address to be used for communicating with the payload. The default is 0x71. This can be changed to any other permitted value by entering 2 valid hexadecimal digits. A table of the assigned I2C slave address is included in Figure 4-14.

I2C Address	Allocation
0x61	EADS Astrium
0x62	University of Bath
0x63	UKSEDS
0x64	University of Southampton
0x65	University of Surrey
0x66	Open University
0x67	UK ATC
0x71	Payload Emulator (default)
0x72	Platform Emulator (not used as master)

Figure 4-14; UKube-1 Address Allocation

- **Status Update Period:** This allows the user to specify the status update period which will be used in both manual command mode and automatic emulation mode to trigger Payload Status Update commands from the Platform Emulator to the Payload Emulator. A range of sample values are available covering the permitted range of 1 to 60 seconds. The value of 1 second should only be used if the user is planning to enter automatic emulation mode as it is impossible to enter menu commands for manual operation within the 1 second period. The default value is 30 seconds.
- **Active Power Bus:** This command allows the user to specify the primary power bus for the payload. This bus will be used in automatic emulation mode to power on the payload prior to initialisation and to power down or power cycle the payload in response to the appropriate flags.

4.4.6 Payload Emulator

The Payload Emulator has a limited command interface which is only used for setting response flags from the payload. These flags are polled when the platform next issues a Payload Update Status command and in automatic emulation mode will be handled on reception of this update at the Platform Emulator. Multiple flags can be set at once or multiple instances of the +256 Bytes to data counter can be called. The order of precedence of the flags is specified in Sec. 4.4.4.

- **Set Payload Update Flags:** This flag will trigger a Payload Update command when polled. In automatic emulation mode this will refresh the payload with updated counts of Data Available and Priority Data Available but will not change the operation mode or operation flags.
- **Set Next Mode Flag:** This flag will trigger a Payload Update command when polled. In automatic emulation mode the Platform Emulator will increment the Operation Mode prior to issuing the update command. It will also refresh the payload with updated counts of Data Available and Priority Data Available but will not change the operation flags.
- **Set Disable Compression Flag:** This flag will be polled by the Platform Emulator but no action will be taken as no data compression algorithms have yet been specified for Payload use.
- **+256 Bytes available for SPI transfer:** This command will add 256 Bytes to the data waiting counter and will set the data 'stream ready' flag in the payload request register. When polled an SPI data transfer command will be issued and 256 bytes will be transmitted.
- **Toggle Error Flag:** This flag will toggle a simulated error state at the Payload Emulator. While the error flag bit is set any command which triggers a read from the Payload Emulator will be responded to with an error packet as defined in the Payload Protocol Specification. The error code byte will be set to 0xFF. This is a sample implementation and it is up to the Payload designer to specify error codes and the appropriate error handling scheme.
- **Set Payload Reset Flag:** This will trigger a reset condition. When polled the Payload Emulator will issue a payload shutdown command to the platform and then 30 seconds later will cycle the active power bus. The bus will be held low for 500ms. 30 seconds after power is re-applied a Payload Initialise command will be issued by the Platform Emulator. It should be noted that many terminal emulation programs will not maintain connection through the power cycle and should either be closed or disconnected in software prior to shut down and re-opened or re-connected shortly after power on.

- Set Payload Shutdown Flag: This will trigger a shutdown condition. When polled the Payload Emulator will issue a payload shutdown command to the platform and then 30 seconds later will power down the active power bus.
- +256 Bytes Priority Data Waiting: This command will add 256 Bytes to the priority data waiting counter. When polled a priority data transfer command will be issued and 256 bytes will be transmitted.
- +256 Bytes Data Waiting: This command will add 256 Bytes to the data waiting counter. When polled a data transfer command will be issued and 256 bytes will be transmitted.

4.5 Error Handling

16 bit CRCs are included on all data commands issued from the Platform Emulator and on all responses from the Payload Emulator. If a CRC is failed on any Platform command it will be discarded by the payload. If a CRC is failed on a payload response the response will be ignored. For Priority Data Transfer, SPI Data Transfer and Data Transfer responses the Platform Emulator will attempt to resend the command up to three times with a 3ms delay between attempts. If after three attempts the response has still not passed its CRC check the Platform Emulator will give up. If the payload response passes its CRC the Platform Emulator will acknowledge this by sending the CRC from the data packet back to the Payload Emulator.

The handling of this acknowledge packet is left to the Payload designer but in the Payload Emulator an example is given where the acknowledge packet is compared with the original CRC generated on the outgoing data packet. If these CRCs do not match the platform knows that the original data packet was corrupted so it can store the data for re-transmission later instead of discarding it. No CRC is generated for the return of the acknowledge CRC to the Payload Emulator. The CRC used in the Platform and Payload Emulators is an implementation of the CCITT SDLC error polynomial ($X^{16} + X^{12} + X^5 + 1$).

4.6 Parameter IDs

Both the platform and payload emulators only have the ability to support up to 16 parameter IDs and associated values. The user can select the IDs of the numbers they wish to assign for use by following the procedure below.

1. Open the platform emulator (UKube_Plt_Emulator_V1_04.pde) in the Arduino IDE
2. Scroll down until the section of the code is reached. It is located just after the code description and header information.

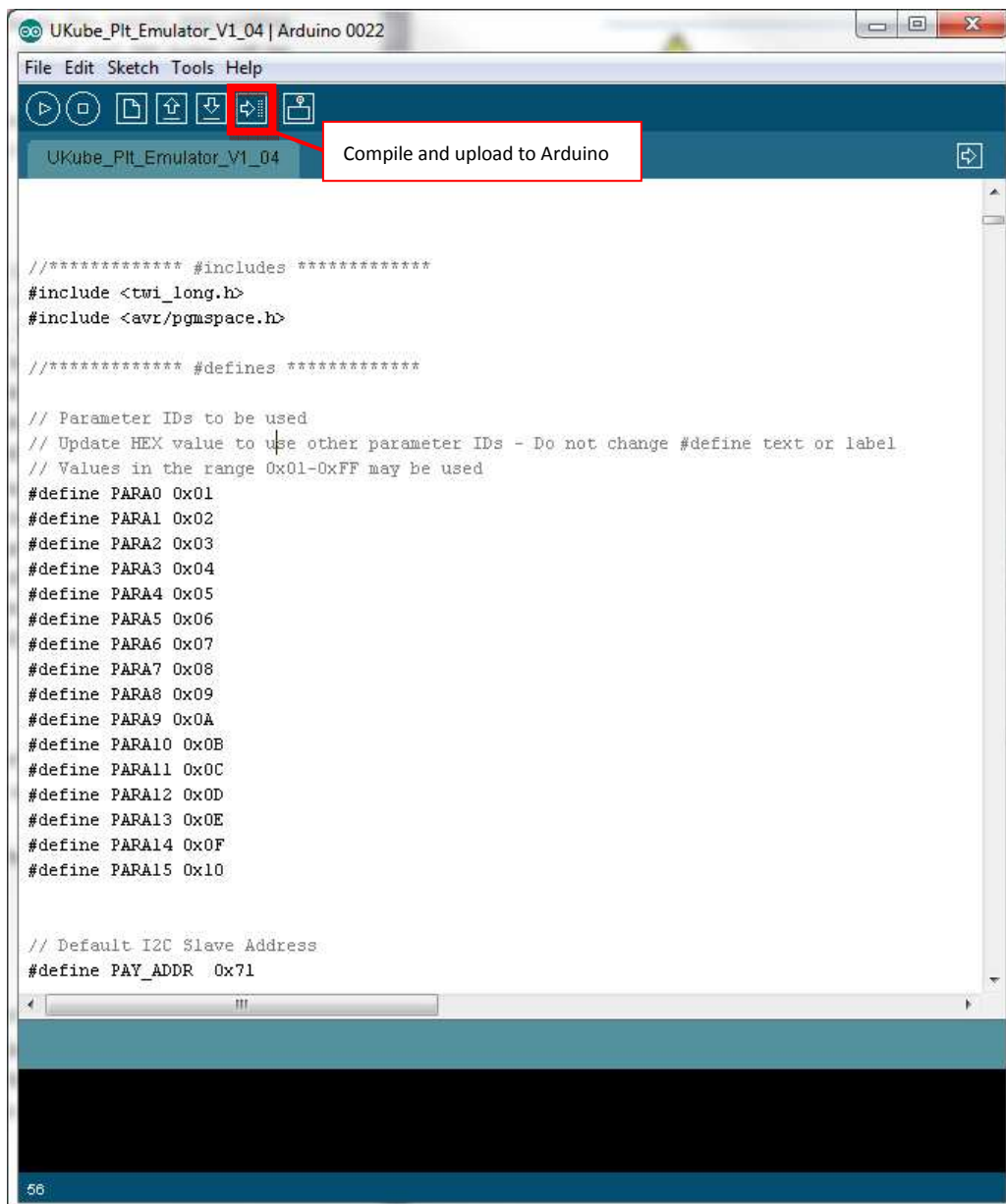


Figure 4-15 Modifying Parameter IDs

3. Modify the parameter IDs to those required by changing the HEX number on the end of each line. The values 0x01 to 0xFF can be used. Do not modify any other part of the line except the HEX value (HEX values are indicated by the digits 0x..).
4. Connect the emulator board and ensure the Platform emulator is powered on.
5. Select the platform emulator serial port
6. Press the program button highlighted in Figure 4-15.
7. Repeat steps 2 and 3 for UKube_Pay_Emulator_V1_04.pde. Ensure the same parameter IDs are entered in the same order as for the platform emulator.
8. Connect a USB cable from the payload emulator to the PC.

9. Power on the payload emulator using the serial menu from the platform emulator.
10. Select the payload emulator serial port in the payload Arduino IDE window.
11. Press the program button in the payload Arduino IDE window (highlighted in Figure 4-15).

4.7 Source Code

The source code for both the Platform and Payload Emulators is included on the USB data pack.

The source code makes use of a modified version of the TWI.h library which is provided as part of the Arduino IDE Wire.h I2C library. This library is used and provided under the GNU Lesser General Public License version 2.1 without any warranty whatsoever. The modified version of the library has been renamed TWI_long.h and allows the use of I2C packets up to 270 bytes long.



The software is provided 'as is' by Clyde Space.



Only source code for the Payload Emulator may be considered open.



With the exception of sanctioned updates, it is strongly recommended that the code within the Platform Emulator is not modified in order to maintain consistency across developments. Should a potential modification be required by a specific payload team, they may wish to prepare changes in the source code and propose a change by submitting to Clyde Space.

5 QUICK START

5.1 Advisory Precautions



Visually inspect all hardware before any electrical connections are made.



The Interface Emulator incorporates static sensitive devices and care should be taken during handling. Do not touch without proper electrostatic protection in place. All work carried out on the system should be done in a static dissipative environment.

5.2 Test Set-up

The typical test bench utilising the Interface Emulator is given in Figure 5-16.