Tuner Studio Operating System Manual

Firmware Version 0.1 01 July 2023

Henry Wright

# Summary

This program is intended to run on Arduino based systems programed via the Arduino IDE such as the Arduino Mega2560. It enables communication with Tuner Studio (TS), a calibration and data acquisition tool available from <https://www.tunerstudio.com/>

The basic functionality is to:

1. Define variables which are saved to and recalled from the internal memory (EEPROM) of the Arduino.
2. Send the variables in memory to TS and receive new data when variables in TS are changed.
3. Write (Burn) these variables to the EEPROM.
4. Send measurement variables at a defined rate to TS for display and logging.
5. Receive commands to execute certain functions on the Arduino. i.e. switch an output on or off.
6. Provide a task based timing system to execute tasks with feedback if those tasks have overrun.

In addition, a simple CAN\_BUS broadcast implementation using the MCP2515 transceiver over spi is implemented.

It is up to the user to build their own functionality into this operating system.

The Latest release can be found in the ->release sub directory.

Documentation can be found in the ->docs sub directory.

Latest version of this program can be found on github.

# Quick Start

## Getting Connected

A quick overview to get started compiling and running the program. Details are in the sections below.

1. You need an Arduino Mega2650 or Uno. Just for now run it with no shields or connections other than the USB.
2. Download Tuner Studio from <https://www.tunerstudio.com/>. For developers the paid version will be the most useful, however the program will work with the free version.
3. Download and install latest Arduino IDE <https://www.arduino.cc/>.
4. Download the entire program folder from github. Recommend placing it in the Documents -> Arduino directory.
5. Either compile and download the program to the Arduino using the IDE or use the release.hex file in ->release to flash the controller.
   1. Note, if re-using an old Arduino, best to run EEPROM\_clear sketch from examples to make sure EEPROM does not contain any leftover data.
6. Open tuner studio and start a new project. When prompted browse for the release.ini file in the -> release folder.
   1. Make sure you select the size of the EEPROM in the project configuration. 1KB for UNO and 8KB for Mega.
7. Open the base.msq to load initial variables into Tuner Studio.
8. Connect to the controller on a serial port.
9. If prompted, write all variables to the controller.
10. Make sure the burn is complete.
11. Reset the controller.
12. You should now be connected

Your screen should look similar to the image below:

A picture containing device, measuring instrument, gauge, clock

Description automatically generated

Figure Main Default Gauge Cluster

## Read Port Data and Control Outputs

The hardware testing button has a few options. Open them all and you will see

1. The status of all digital ports, whether they are input or output. On = green = Logical high (5V).
   1. Note1. The Mega only has 54 ports, even though 64 are shown.
   2. Note2. The port numbering matches that shown on the Arduino case.
2. The Analog inputs as shown on a live graph. They are also available as gauges on the main screen.
3. The Test ports section where you can override the status of a digital port.

A close-up of a computer screen

Description automatically generated with low confidence

Figure Menu Items in Hardware Testing

If you head to the Test Ports section. You should be able to “enable Test Mode.” If it is greyed out you first need to enable “Allow Hardware Test Commands” in system settings.

A screenshot of a computer

Description automatically generated with medium confidence

Figure System settings

Once that is working, you should be able to manually override each port. Note the following:

1. Ports defined as INPUT will change the value seen by the program but will not change the physical port.
2. Ports defined as OUTPUT will change the physical state of the port as well as what is shown.
3. Remember that some ports are used for other functions such as serial communication and you may not be able to change them or changing them causes errors in other functions.

A screenshot of a computer

Description automatically generated

Figure Hardware Testing and Override View

## Inserting your own code

The code inherits all the normal Arduino libraries. When starting out, it’s best to open the whole project in a reasonable editor such as notepad++. Open the whole folder as a workspace so you can quickly navigate between files.

Open userfunctions.ino

Write a simple function, such as

A screenshot of a computer program

Description automatically generated with low confidence

Make sure you add the prototype of the function to the userfunctions.h file. Then you head over to the TunerStudio\_OS\_Dev.ino file and find the correct task rate to call the function.

Notes:

1. Avoid writing a lot of code and if statements in the task schedulers in TunerStudio\_OS\_Dev.ino, it gets messy fast.
2. Note how we are using setDigitalPort and readDigitalPortinstead of digitalRead and digitalWrite. This wrapper is what allows the Test port commands to work.

A picture containing text, line, font, screenshot

Description automatically generated

When you are done. Save your work and compile the code in the Arduino. Then download and run the new program.

A screen shot of a computer

Description automatically generated with low confidence

Figure Example of Low Pass filter. Alpha of approximately200