Hollow Clock

# Hollow Clock V New Software

This clock is based on the plans in Thingiverse. It is sold under license from the designer. I have, however, modified the software. The software now has a serial menu system that can be used to fine tune the clock’s accuracy and several other settings. I also changed the software to use the µSecond (microsecond) counter for more accuracy. The original code used the µSecond counter multiplied by 1000 to give µSeconds, but this of course doesn’t really increase the accuracy since the counter only reads mSeconds. In addition to this the code did not handle the counter overflow that happens about every 49 days. This caused a gain of up to a minute each time the overflow happened.

## Software Version

* 1.02 The first released version.
* 1.03 Added the current seconds to the menu display so you can see where the clock is.
* 1.04 Removed the uSec debugging message.
* 1.05 Fixed the ‘F’ command that was broken in the last version.
* 1.06 Cleaned up the stepper code by removing the double subscript array and using a field of 4 bits to enable each port for stepping.

## Starting the Clock

The best way to start and set the clock to the correct time is to disconnect the USB C connector from the clock, set the hands to the correct time, and then plug in the USB C connector. Follow these steps to start the clock.

* Remove the rotor assembly by lifting it from the base.
* Move the clock hands to the correct time, perhaps the next minute.
* Replace the rotor assembly on the base while preventing the hands from moving.
* When the time (using an accurate clock, like a cell phone) matches the clock hands start the clock by connecting the USB C connector.

## Fine Tuning the Clock Accuracy

Read the information on the ‘F’ command below to see how to calibrate the clock to achieve maximum accuracy. The clock has been approximately calibrated when assembled, but you might wish to make it even more accurate.

# Serial Port Menu System

## USB Connection to Computer

The clock must be connected to the computer to use the menu system. A USB C to A or C type connector may be used. The PC must be running some serial port software, such as SSCOM PuTTY or similar. The Arduino IDE is an excellent choice, it is free and easy to install. Enable the serial port monitor and a section of the IDE will display the text coming from the clock. The serial port is set to 9600 baud, 8 bits, no parity, 1 stop bit.

### Using PuTTY on Windows

Download and install PuTTY from here: [Download PuTTY: latest release (0.81)](https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html) or from the Windows Store.

The actual serial port for the clock will be needed. It can be found by running Windows “Device Manager” and opening the Ports section. The clock will be a USB Serial Device. Remember the name inside (). COM13 in this case.

Start PuTTY and select “Serial” as shown. Enter the com port name in the “Serial Line” field.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Press the “Open” button and the PuTTY window will open. Press the enter (aka return) key to show the menu.

A screenshot of a computer

Description automatically generated

Clock commands can be entered using the keyboard. The enter key sends the command to the clock. The menu will be displayed again with any changed values. Entering a blank line just displays the menu. Every minute the clock will write a line showing the time that the clock has been running. This number is useful when calibrating the clock to keep more exact time.

## Received Data

Every minute a line will be sent from the clock to the serial port. This information is useful for calibrating the clock and setting certain other values.

Here is a sample line.

run time: 6 minutes = 0.100 Hours

The minutes and hours values are how long the clock has been running. The hours value is the same as the minutes divided by 60 and is used for the ‘F’ command to calibrate the clock.

## The Menu

The menu is displayed after entering ‘?’ or a blank line in the serial terminal software. Each command is a single character, some of which require additional information after the command.

------ Current Settings ------

Firmware version : 1.01

uSeconds calibrate per minute : -16635 or -23.95 sec/day

Wait/Run State : Running

Stepper Delay (mSec) : 2

Test Mode : Off

Reverse Motor : No

--- Commands ---

+<n> : Advance n minutes

-<n> : Reverse n minutes

A<n> : Adjust Minute Position (+/- 256 is a full minute)

W : Wait, toggle running state of clock

C<n> : Calibrate uSeconds per minute, is default, change as needed, +speeds up, -slows down

F<sec> <hours> : Figure correction using seconds and hours (floats), e.g. F -2.5 24.0 if 2.5 seconds slow per day

S<n> : Set stepper motor delay, default is 6, range 2 to 120

T : Test mode (enter anything while running to stop)

R : Reverse motor setting

Command?

The first part of the menu shows some current values. The last part lists the available commands with short descriptions. Note that this clock was running too fast. The -16635 was used to slow it down.

### Commands

#### +<n>

The integer <n> is used to advance the clock by that many minutes. E.g. +5 will move the clock forward by 5 minutes.

#### -<n>

The integer <n> will move the clock backwards by that many minutes.

#### A<n>

This value is used to adjust the minute pointer by small increments, so it lines up with the minute marks exactly. Both positive and negative values may be entered. If a negative value is entered the clock will move backwards and forwards again to take up the gear drive slack for more accuracy.

Follow these steps to calibrate the hands for maximum accuracy.

* Use the W command to stop the clock during this procedure.
* Remove the rotor assembly from the base.
* Rotate both hands to noon.
* Put the rotor back on the base.
* Use -2 and +2 commands to remove the slack in the drive gears.
* Examine the hour hand. If it is not exactly at noon use the +/- commands to align it properly with noon. +1 is the smallest amount, but sometimes 2 or larger might be required.
* If the minute hand was moved by more than 1 remove the rotor, move the minute hand to noon, and insert the rotor back into the base.
* Gently push the minute hand to the left to take up any gear slack. You can also use -1 followed by +1 to do this with more accuracy.
* If the minute hand is not exactly at noon use the A command to move it left or right with A+<n> or A-<n> where <n> is a number between 1 and 256. 256 corresponds to a full minute of motion, typically values will be around 50 to 100 for this adjustment.
* Set the hands to the correct time by using the + or – commands, or by lifting the rotor and moving the hands.
* At the correct time restart the clock with the W command.

#### W

This command makes the clock stop. It is useful if you want to start the clock at a chosen exact time. Set the hands to the desired time and when the time is reached press ‘W’ followed by enter on the serial monitor program to start the clock running again. This command also resets the time running command to make it useful for calibration.

#### C<n>

This command is used to adjust the running speed of the clock in µSeconds per minute. Positive values make the clock run faster and negative values make it run slower. This value can be calculated from monitoring the clock run for several days. Note that the F command is much easier and will calculate the correct value from the number of seconds over the number of hours that the clock is losing or gaining time.

#### F<sec> <hour>

This command will calculate the correct speed adjustment from the number of seconds per the number of hour that the clock is fast or slow. Both numbers are entered as floating point (decimal) numbers. Seconds can be positive or negative. Negative indicates that the clock is running slow. The most accurate values are produced by monitoring the clock for longer periods of time, at least 12 hours, but several days is even better. As an example, if the clock loses 10.5 seconds in 2 days the correction would be entered as “F -10.5 48”. Recording the start time from a cell phone and then many hours later noting how many seconds have been gained or lost will supply the number of seconds. The easiest way to measure the seconds is to start the clock when the cell phone shows 0 seconds. Then the seconds gained or lost can be noted on the cell phone by watching when the clock advances the minute hand. The hour value is displayed by the clock on the serial display but can also be noted from the cell phone time.

If there is already a current value for adjustment this command will add or subtract the new amount that is calculated. This is done to allow further refinements to the speed calibration without having to always start from the 0 calibration value.

Here are the steps to follow.

* Use ‘W’ to stop the clock.
* Set the clock timer offset to 0 with the command: C 0 ‘enter’
* Set the clock to the correct time, the previously mentioned steps can be followed.
* Use ‘W’ to start the clock and note the time on a cell phone or other accurate source. The ‘W’ command also resets the hour timer that is displayed.
* Wait at least 12 hours.
* Wait for the next minute display to show on the serial data. Note the time and calculate how many seconds the clock is fast or slow. Remember the number of hours shown on the text line.
* Enter the F command as: F <seconds> <hours> ‘enter’
* Use – seconds if the clock is slow and losing time.

#### S<n>

S is used to set the stepper motor stepping delay. Weaker motors might need longer delays, or you might just like the different sound of slower or faster stepping.

#### T

This command toggles test mode. When test mode is on the clock runs continuously. This can test the servo and gears to make sure it rotates freely. Pressing enter will stop the test mode and return to normal running.

#### R

The ‘R’ command is used to reverse the servo. This is needed when chosen servo runs in the opposite direction. It only needs to be set once, unless you want your clock to run backwards of course!