GPS-synchronised Analogue Clock Driver

Design: John Clarke Software: Geoff Graham Words: Nicholas Vinen

Traditional clocks (with hands) are fairly accurate – but every now and then you have to get them down off the wall and adjust them so they show the real time. And daylight saving means you have to adjust it twice a year anyway! Wouldn't it be nice if the clock adjusted itself so it was ALWAYS 100% spot on AND adjusted itself for daylight saving? Build this *GPS Analogue Clock Driver* and your wishes will come true!

Battery-powered quartz crystal clocks are inexpensive, look good hanging on the wall and for many people, they are the preferred way to check the time.

But (despite what many people think) they usually aren't that accurate, drifting by as much as two seconds per day, which means they can be out by up to one minute after a month.

And you have to remember to change them twice a year if you have daylight saving time (DST) in your area. That's especially troublesome if the clock is mounted up high since you need to get up on a ladder or chair to adjust the time. Wouldn't it be nice if you never had to do that again? Well, at least until it's time to change the battery…

This design replaces the electronics in a standard quartz wall clock with a controller that always knows the correct time, thanks to the Global Positioning Satellite (GPS) system.

It uses an inexpensive ($25) GPS module to get the precise time from orbiting atomic clocks and a microcontroller to drive clock hands.

It will run for up to two years on two alkaline AA cells (or one year with a sweep second hand movement) and over that period will keep the time accurate to within one second.

If you don't know the difference between 'sweep' and 'stepped', a sweep second hand appears to rotate in a continuous movement, while a stepped second hand will appear to 'jump', usually in one-second steps.

If your clock has a stepped second hand, you can even program your local DST rules into the clock using a USB cable from your computer and then, when the time comes, the clock will automatically go forward or back by an appropriate amount of time. It does this by either advancing the second hand twice per second, or not at all, until the time shown is correct again.

Fig.1: inside a typical quartz clock mechanism with stepped second hand, showing the modifications we made to terminate the connecting leads to the stepper motor coil.

For clocks with step hands, all you have to do is set all three hands to the 12 o'clock position before inserting the battery. The controller will use its onboard GPS module to get the current time and then step the clock hands at high speed around the dial until it has reached the correct time. It will then drop back into normal timekeeping mode with the time derived from a 32,768Hz crystal oscillator.

For clocks with sweep hands, the procedure is similar, but rather than setting it to 12 o'clock, you set it to the next full half hour and the firmware will then wait an appropriate amount of time before driving the clock mechanism, so that the time shown is correct.

To conserve the battery, the GPS module is only used to synchronise the clock every 44 hours and following synchronisation, the clock will either skip seconds or double-step to reach the correct time.

After synchronisation, the microcontroller is also able to calculate the inherent inaccuracy of its crystal oscillator and will then compensate by occasionally skipping or double-stepping

a second, without the GPS module needing to be powered up.

This process can also compensate for aging of the crystal and will keep the clock accurate between synchronisations. This also means that you will probably never even notice the clock making a correction; the time will simply be right!

Battery status monitoring

The controller monitors the battery voltage and when it has dropped below 2V (ie, 1V per cell), the microcontroller will stop the clock at a convenient position.

For clocks with stepped hands, it will stop at exactly 12 o'clock before the battery is so flat that it can no longer drive the mechanism.

You then replace the battery and it will then automatically advance to the correct time again.

For clocks with sweep hands, the firmware will halt the clock at exactly the hour or half-hour position. Before you replace the battery you need to set the hands to the next hour or half hour, but hopefully, you will not have to mess with the second hand because it should have stopped at the exact 12 o'clock position.

Either way, if during operation the GPS signal level drops to a point that is too low for the module to get a lock on enough satellites, the clock will stop at exactly five minutes before the hour/half hour. Similarly, if the GPS module stops running altogether, the clock will stop at 10 minutes before. These indications make it easy to differentiate between a low battery and something more serious. In either event, the firmware will try to acquire a GPS lock again ten times with a 4-hour delay between each attempt before it gives up. This gives the GPS module plenty of opportunities to come good.

Internally, the firmware measures time in eighths of a second. This allows for much finer tracking of errors and control of where the clock's hands are pointing.

Features and specifications

- **Drives virtually any battery-powered quartz clock movement**
- **Works with a sweep or stepped second hand**
- **Long battery life from two AA cells: about one year for clocks with sweep second hand and two years for stepped second hand**
- **Small enough to mount on the back of most clocks**
- **Time synchronised to GPS satellites every 44 hours (configurable)**
- **Can use a variety of GPS modules, including low-cost types**
- **Automatically skips or adds extra seconds to keep clock accurate**
- **Automatically trims internal crystal oscillator based on GPS updates**
- **Automatically sets time when fresh cells are inserted (with stepped second hand only)**
- **Automatically adjusts for Daylight Saving Time (with stepped second hand only)**

Theoretically, it will mean a higher degree of accuracy, although this is offset to some extent by the fact that most clocks with sweep hands will lose a fraction of a second when they start up. This is something that the firmware is not aware of and cannot correct for.

Revised design

Astute readers (or those with long memories!) may recall our original *GPS-synchronised Analog Clock* articles from March 2011 issues, which was for clocks with step hands.

This new design works with either type of movement (step or sweep) and features a number of benefits over the earlier design.

The EM-408 module used in that project is now obsolete and difficult to get; the VK2828U7G5LF module we are using this time is substantially cheaper and has a number of benefits including support for GALILEO (European) and GLONASS (Russian) positioning satellites in addition to the GPS (United States) system.

In fact, it can use satellites from all three systems simultaneously to increase the chance of getting a signal indoors, as a GPS fix relies on receiving signals from multiple satellites (normally at least three).

This module is based on the u-blox Neo-7 chip and has slightly better sensitivity than the previously used EM-408, with a specified tracking sensitivity of –162dBm compared to –159dBm.

It also has a slightly lower current drain, at around 30mA compared to 44mA. Plus it has a faster 'cold start' average time of 26 seconds compared to 42 seconds, meaning it doesn't need to be powered up for as long to get the time.

We have also substantially increased the power efficiency of the GPS module supply. While the GPS module is only powered up about once every two days, it does draw significant current during that time, and so any improvement in efficiency should extend battery life both through draining less charge each time, as well as reducing the temporary voltage drop due to the load on the cells, which may push them below the 1V cut-out threshold.

We have also ditched the old-fashioned DB9/DE9 serial cable and fitted a micro-USB port so that you can

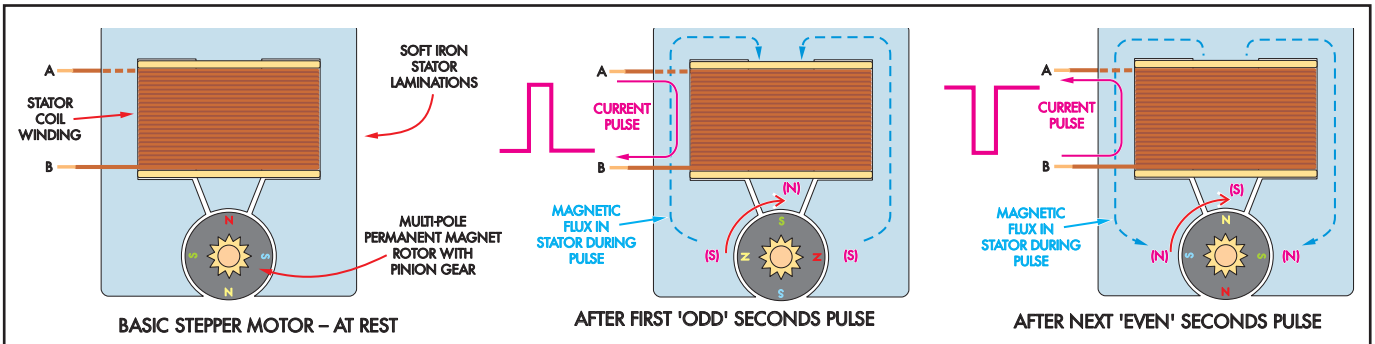


Fig.2: the clock motor consists of a multi-pole permanent magnet rotor inside a circular gap in a soft-iron stator. It's made to step in one direction by reversing the polarity of the current pulse at each step.

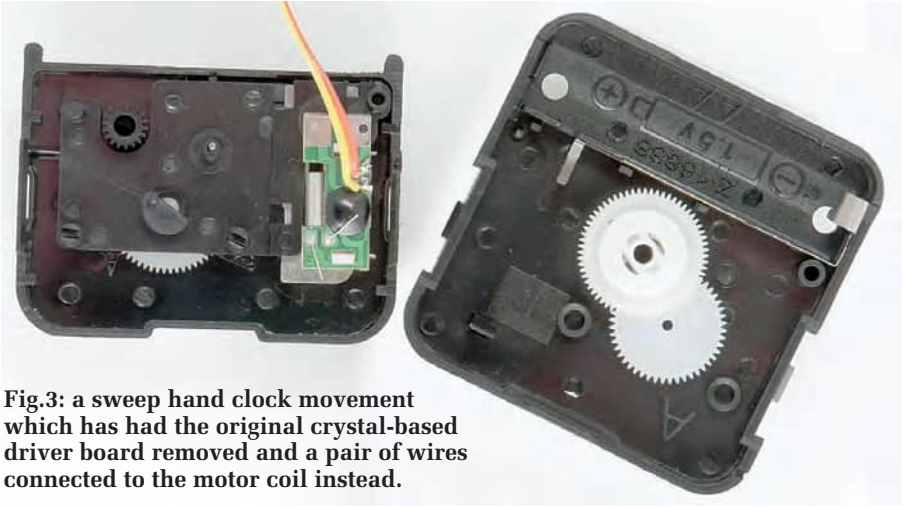


Fig.3: a sweep hand clock movement which has had the original crystal-based driver board removed and a pair of wires connected to the motor coil instead.

easily hook it up to your computer to set up the DST rules and make other setting tweaks.

How it works

A standard battery-operated wall clock uses a crystal oscillator and binary divider to generate a pulse once per second which drives a simple stepper motor and, via gears, the hands of the clock.

The motor consists of a coil with a soft iron core and a small bar magnet (the rotor) positioned in the magnetic field (see Fig.1). When an alternating current flows through the coil, this causes an alternating magnetic field and the rotor rotates to follow this field. It is this rotation that, via gears, drives the clock's hands (see Fig.2).

The crystal oscillator is normally quite accurate, especially when the clock is new – but it's affected by age, temperature and battery voltage, all of which can add up to 14 seconds a week. Our circuit replaces the clock's electronics and generates compatible pulses to drive the stepper motor.

A clock with sweep hands works essentially the same way except that its gearing has a higher reduction ratio, so many more pulses are needed to advance the hands by one second (see Fig.3).

This allows the pulses to be produced more-or-less continuously so the hand moves in a smooth manner. In exchange for a greater battery drain (due to the much higher duty cycle operating the motor), you eliminate the 'tick-tick-tick' noise, making for a much more luxurious timekeeping (and, for some people, sleeping!) experience.

By contrast with the standard clock, at the heart of our circuit is a PIC16LF88 microcontroller which uses a 32,768Hz watch crystal to drive a timer within the chip.

This timer generates an interrupt which is used by the software running on the microcontroller to keep time and also generate pulses to drive the clock motor.

Fig.4 shows how the clock motor is driven by the microcontroller. One end of the clock coil is connected to the junction of the two (nominally) 1.5V cells, while the other end is driven by three paralleled output pins which can momentarily be connected to Vdd, Vss or left open-circuit.

The resulting bipolar waveform for continuous sweep hand clocks has 16 pulses per second, while the waveform for stepping hands is similar but has just one pulse per second (positive or negative).

Fig.5 shows a scope grab of this same waveform, without the mechanism connected, while Fig.6 shows the same waveform with the coil in-circuit.

For clocks with sweep hands, the rotor in the clock's movement has a certain amount of momentum which keeps it spinning while driven by this pulse train, so it never stops. This is different to the stepping clock movement where the voltage pulse on the coil pulls the rotor around and then stops it dead – once every second – thereby creating that ticking sound.

Besides driving the motor, the software also needs to keep track of time, calculate the DST state and time zone offset, as well as periodically power up the GPS receiver and interpret its output.

As a result, the software is really quite complex. As an illustration of this complexity, drafting the circuit took just a few hours, while the software took many weeks to develop.

A normal clock cycle starts at the beginning of each second. The timer generates an interrupt which causes the processor (CPU) in the microcontroller to wake up and execute the interrupt code. The program will perform some calculations (more on this later) and then simultaneously drive output pins 15, 17 and 18 either high or low. It then sets the timer to generate another interrupt after a few tens of milliseconds and promptly puts itself back to sleep.

When the timer expires again, it wakes the CPU up and the program sets these outputs back to being high-impedance.

If the clock has a stepping hand, its job is done and it can wait until the next 'tick' and repeat the whole process. But if it has a sweep hand, it will set the timer to wake up again after another short period to deliver the next driving pulse.

During the sleep period, everything except the crystal oscillator and the timer is shut down, resulting in a current drain of only a few microamps by the microcontroller.

In addition, the CPU in the microcontroller will run at full speed for only 60-100µs while processing an interrupt, so the total current drawn by the microcontroller is negligible.

Most of the current, in fact, is drawn by the clock stepper motor – which is the case with a 'standard' battery-

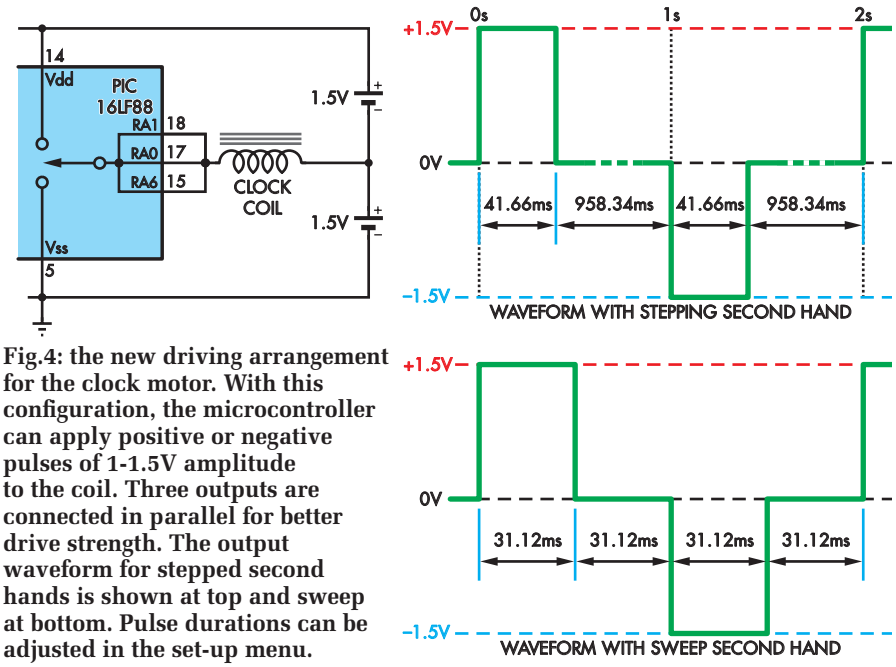


Fig.4: the new driving arrangement for the clock motor. With this configuration, the microcontroller can apply positive or negative pulses of 1-1.5V amplitude to the coil. Three outputs are connected in parallel for better drive strength. The output waveform for stepped second hands is shown at top and sweep at bottom. Pulse durations can be adjusted in the set-up menu.

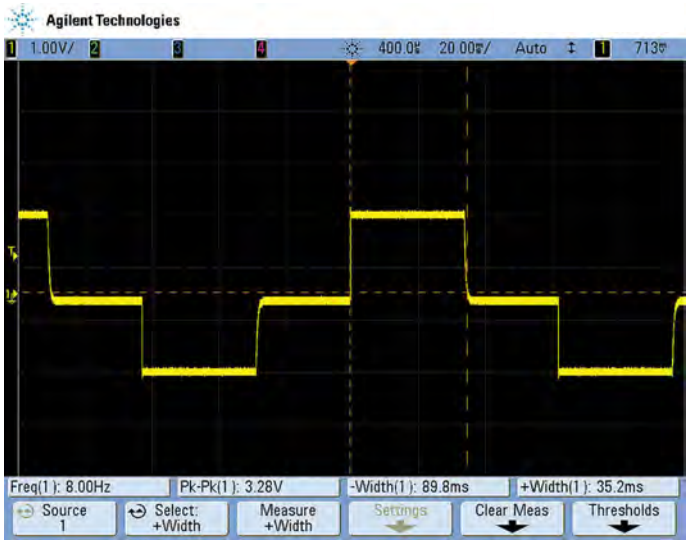


Fig.5: this scope screen grab shows the output signal from pins 15, 17 and 18 of microcontroller IC1 with no load connected and is measured with the centre point of the cells as the ground reference.

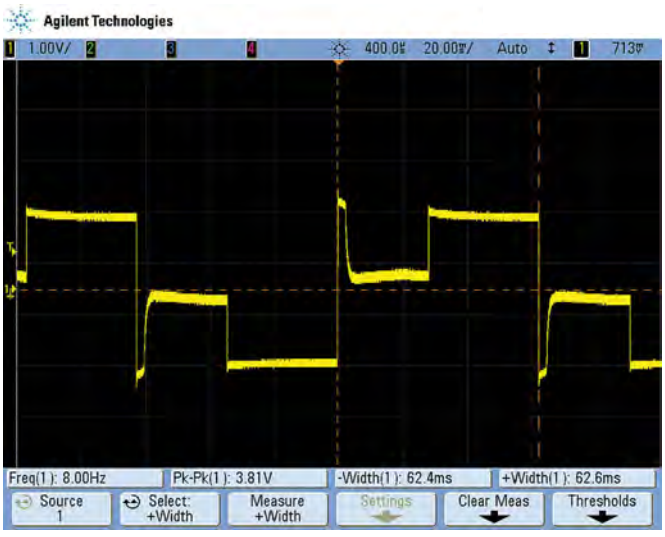


Fig.6: the same measurement as in Fig.5 but with the clock movement connected. The voltage spikes are created by the motor's inductance each time the drive current is reduced to zero. They are clipped by schottky diodes D3 and D4.

operated clock (see the box: Calculating Battery Life).

At the start of each second, the program compares where the clock hands are actually positioned and where we would like them to be. Depending on the result of this comparison, the program may bring the clock's hands closer in agreement to the correct time by skipping a pulse to the clock's stepper motor or by generating a double step.

For example, when DST starts, the software simply adds 3600 seconds (one hour) to the desired position and the clock will then automatically 'fast forward' until it is an hour ahead.

When it is time to synchronise (ie, once every 44 hours), rather than going back to sleep after handling the interrupt, the micro switches on power to the boost regulator which provides either 3.3V or 5V to the GPS module. This is derived from the 2-3V battery voltage.

Once the GPS module has acquired enough satellites to get an accurate time reading, the microcontroller extracts this from the serial data stream and converts it into an internal representation (seconds since 1 January, 2000), applies the time zone offset, calculates if DST applies, calculates the internal crystal oscillator error, and so on – all the steps necessary to keep the clock showing the right time.

When it has finished and the current time setting is confirmed as correct, the GPS module is powered down and the unit goes back to normal operation.

The GPS module

We normally use a GPS module to find our position on the globe. However, the GPS system is based on time signals derived from extremely accurate atomic clocks, so the UTC time is also supplied in the GPS receiver output.

In fact, most time standard bodies around the world use the GPS system as a 'standard beacon' to transfer accurate clock readings between each other. And let's face it, at £20, a GPS module is a tad cheaper than an atomic clock – even a used one!

Most GPS modules follow the NMEA (National Marine Electronics Association) standard for data output and generate a serial data stream at 4800 or 9600 baud, with eight bits per character. They generally use a TTL-level version of the RS-232 serial protocol.

The NMEA standard also describes the content of the data and we use the RMC (Recommended Minimum data) message, which is part of the default output for almost every GPS module made.

You don't have to use the VK2828U7G5LF module; any GPS module which can run off 3.3V or 5V and supply a TTL-level RS-232 stream at 9600 baud should work.

But keep in mind that if its sensitivity is inferior to the VK2828's, or the current drain is higher, your clock might not work as well as our prototype.

Stepping or sweep hands?

Believe it or not, some people actually like the 'tick, tick, tick' sound of stepped clocks and find them soothing and conducive to sleep.

Others may find that noise terribly annoying. So it's really up to you; just keep in mind that if you choose a clock with continuous sweep hands, you will be changing the battery more often.

Also note that if you are using a clock with sweep hands, the DST adjustment cannot take place automatically and you will also need to do a bit of extra work whenever you insert fresh cells (see below for details).

While it's quite hard to find clocks with a battery-powered continuous sweep movement, the movements are readily available on eBay and Ali Express for just a few dollars.

So if you want a sweep hand clock, we suggest you purchase a clock based on its appearance, then replace its mechanism. You can do that at the same time as fitting the GPS timekeeping module. Just make sure to purchase a movement with the correct shaft diameter and length.

Basically, once you have your clock, take the hands off the shaft and then remove the movement from the clock. Measure the shaft diameter and length and find a sweep movement with an equivalent shaft.

The replacement movements are often advertised along with shaft dimensional diagrams so you can match them to your clock. A good place to look is **aliexpress.com** – search for 'JL6262' and you'll find mechanisms for under a fiver (including delivery).

Many of the movements are also supplied with hands, so you can decide whether to keep the hands that came with your clock or replace them with the new ones.

If you want to try to purchase a clock with sweep hands, then the search terms that are worth using in conjunction with 'clock' are: 'sweep', 'continuous sweep', 'silent' and 'mute'.

By the way, if you have a clock with a failed movement but you prefer a stepping second hand, Ali Express and eBay are also an excellent source of low-cost replacement stepping movements, so you can keep your favourite clock in operation almost indefinitely.

Note that the circuit is exactly the same for driving either type of movement, the only difference is in the

firmware; you simply program the chip with the firmware appropriate to the type of movement you are using.

Sweep hand driving limitations

Because the motor on a clock with a continuous sweep second hand needs to be driven constantly, rather than just delivering the occasional pulse, and since the motor is designed to operate at a certain speed, it can only really be sped up or slowed down by around 6%. This is perfectly fine for making one or two second adjustments to keep the clock accurate, but it would take too long to make up an hour during DST transitions.

As a result, if you want automatic DST adjustments, you need to use a clock movement with a stepping second hand.

Having said that, manual DST adjustments on a clock with sweep hands is not that difficult; you let the clock continue to operate, driving the second hand, and wind the minute/hour hands backward or forward by an hour (or whatever the appropriate time period is) and ensure that the minute hand agrees with the position of the second hand as it sweeps around. This is much easier than having to find an accurate time source to completely reset the clock.

Also, when using a clock with a stepping hand and inserting a fresh pair of cells, you simply set the hands to the 12 o'clock position and the clock will then advance the hands to the correct time. This is not possible for the same reason as stated above, so with a clock with a sweep second hand, what you do is set the time to the next half hour (eg, if it's 11:18, set it to 11:30) and it will then wait until the hands are in the correct position before driving the movement.

Circuit description

The full circuit is shown in Fig.7 and the key component is IC1, the PIC16LF88 microcontroller. This drives the clock's stepper motor, controls the power to the GPS module and interprets the output of the module.

Note that the LF version of the PIC16F88 is guaranteed to operate down to 2V, while the standard version is only rated to work down to 4V. Having said that, you may well find that a standard PIC16F88 will operate without fault to below 2V; it just isn't guaranteed.

The 10kΩ resistor and 470µF capacitor connected to pin 4 of IC1 (via a 1kΩ current-limiting resistor) serve to hold the microcontroller in reset for a few seconds after the battery is connected. This provides enough time for you to properly seat the cells in the holder before the microcontroller starts executing its program.

Diode D1 prevents the capacitor from discharging into the microcontroller when the cells are removed.

The serial interface connector CON2 is linked to the microcontroller via a few protective resistors. This design relies on the fact that nearly all modern serial RS-232 interfaces use a threshold of about 1.5V between a high and low signal. This is not what the full RS-232 standard specifies, but we use this fact to provide a simple interface to a personal computer for configuring the clock.

You can use a PICAXE-style serial cable terminated with a 3.5mm stereo jack plug to connect to CON2.

We think most constructors will lack such a cable, so we've provided a mounting location on the board for a low-cost CP2102-based USB/serial converter which has an onboard micro-USB socket. This connects to the serial transmit/receive pins on IC1 (via the same resistor network) and also to GND. Since there is no power connection, you still need the battery in place to set the unit up.

Crystal X1 provides a stable timebase for the clock with the two 22pF capacitors providing the correct loading. Normally, you would need to trim at least one of these capacitors for the clock to be accurate, but since the software automatically corrects for crystal timekeeping errors by periodically comparing the internal (RTC) time to the GPS time, this is not required.

The microcontroller applies power to the GPS module by pulling its pin 3 low. This turns on PNP transistor Q2, which switches on and charges the 220µF capacitor at its emitter to around 2.8V, powering the boost regulator.

This is based around REG1, the MAX756 DC-DC converter. REG1 operates by drawing a current through inductor L1 and then suddenly cutting it off. The collapsing magnetic field causes a positive voltage spike across the inductor that is dumped via schottky diode D2 into the 220µF output capacitor, powering the GPS module.

REG1 can operate with a low supply voltage (down to at least 1.8V) and still deliver a closely regulated output of 3.3V or 5.0V. The actual output voltage is controlled by pin 2 and this can be configured using JP1, to suit the GPS module in use.

L1 must have a saturation current rating of 1A or greater. This means that it should be wound with heavy gauge wire on a powdered iron core; an RF choke will not work. The parts list provides two alternatives. Also, both the 220µF capacitors must have low ESR (equivalent series resistance).

The configuration of Q2 is one of the improvements we've made to the circuit; the original design used a Darlington pair which caused a voltage drop of around 0.7-0.8V from the battery to REG1. This reduced its efficiency markedly and caused it to draw more current from the battery, draining it faster.

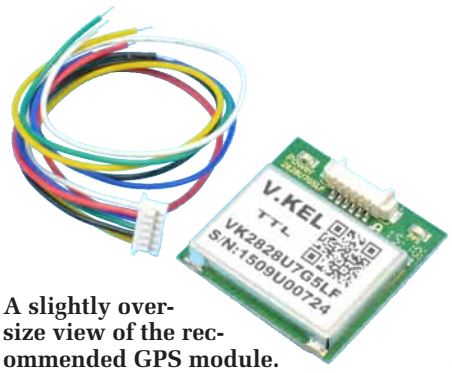
With a single transistor and a higher base drive current of 4.5-10mA (due to the 270Ω base resistor), Q2 is capable of supplying at least 400mA – more than enough for REG1 to start up and operate, with an overall efficiency improvement of between 29% (at 3V) and 65% (at 2V).

REG1 generates an internal reference voltage of 1.25V which is used in regulating its output voltage. This reference voltage is also made available at pin 3 of the chip and we pass it back to the microcontroller which uses it as a reference to measure the battery voltage. By accurately measuring the battery voltage, we can stop the clock at the 12 o'clock position before the battery gets too low to operate the microcontroller.

Incidentally, the microcontroller is programmed to measure the battery voltage at the time of greatest current draw (about 160mA) when the GPS module is starting up. If you measure the battery voltage without a load, you will probably get a higher reading.

The GPS module is one of the simpler parts of the circuit. It has two connections for power, two for communications to the microcontroller (transmit and receive data) and an enable signal. We connect the enable line to its V+ pin so that the module is always enabled when power is applied.

As we do not send anything to the GPS module (the manufacturer's default configuration suits us just fine), the receive data line is also pulled high, by a 1kΩ resistor. The 10kΩ resistor in series with pins 8 and 10 of the microcontroller limits



A slightly oversize view of the recommended GPS module. Other modules should work, but we know this one will!

the current into the microcontroller when the GPS module operates at a higher voltage.

The microcontroller drives the clock stepper motor from pins 15, 17 and 18, which are paralleled for a higher output current. When these pins are at a high impedance, no current flows through the clock motor. If they are driven high, there is about +1.5V across CON1, while if they are driven low, there is about –1.5V. The micro produces alternate high and low pulses to drive the motor, at 1Hz for stepping second hand clocks and 8Hz for sweep hand clocks.

Schottky diodes D3 and D4 clamp inductive spikes from the motor windings to the supply rails. These occur when output pins 15, 17 and 18 switch to a high impedance after delivering a pulse to the motor windings and are caused by back-EMF from the collapsing magnetic field of said windings (see Fig.6).

Finally, pushbutton S1 can be held down during start-up to signal microcontroller IC1 to go into configuration mode, where its settings can be changed over the serial/USB interface. LED1 flashes at start-up and indicates whether the clock is in set-up mode or operating normally. The USB module has on-board LEDs to indicate when it has power (red) and if it has a GPS signal (green).

Construction

All of the components for the *GPS-Synchronised Analog Clock* driver, including the GPS module and the AA cell holders, are mounted on a PCB measuring 140 × 61.5mm and coded 04202171. This board is available from the *EPE PCB Service*. The component overlay is shown in Fig.8.

Start by fitting the wire link next to Q2, then follow with diode D1 and the resistors. Check each resistor value with a multimeter before soldering it in place. Follow with D2-D4, being careful to orient all diodes in the same direction as shown in Fig.8.

Next, fit the socket for IC1 (notch at top), switch S1 and REG1. REG1 should be soldered directly to the board and be careful to orient it as shown.

Now solder the ceramic and MKT capacitors in place, followed by the electrolytic capacitors, with their positive (longer) leads through the pads marked '+' on the diagram. Fit Q2, followed by the pin header for JP1 and then LED1, which can be pushed right down or soldered with short leads. Its longer (anode) lead must go through the hole marked 'A'.

Push CON2's pins through the slots in the board and make sure it is flat on the board and its edge is parallel with the edge of the PCB before soldering all five in place. Install CON1 at the same time.

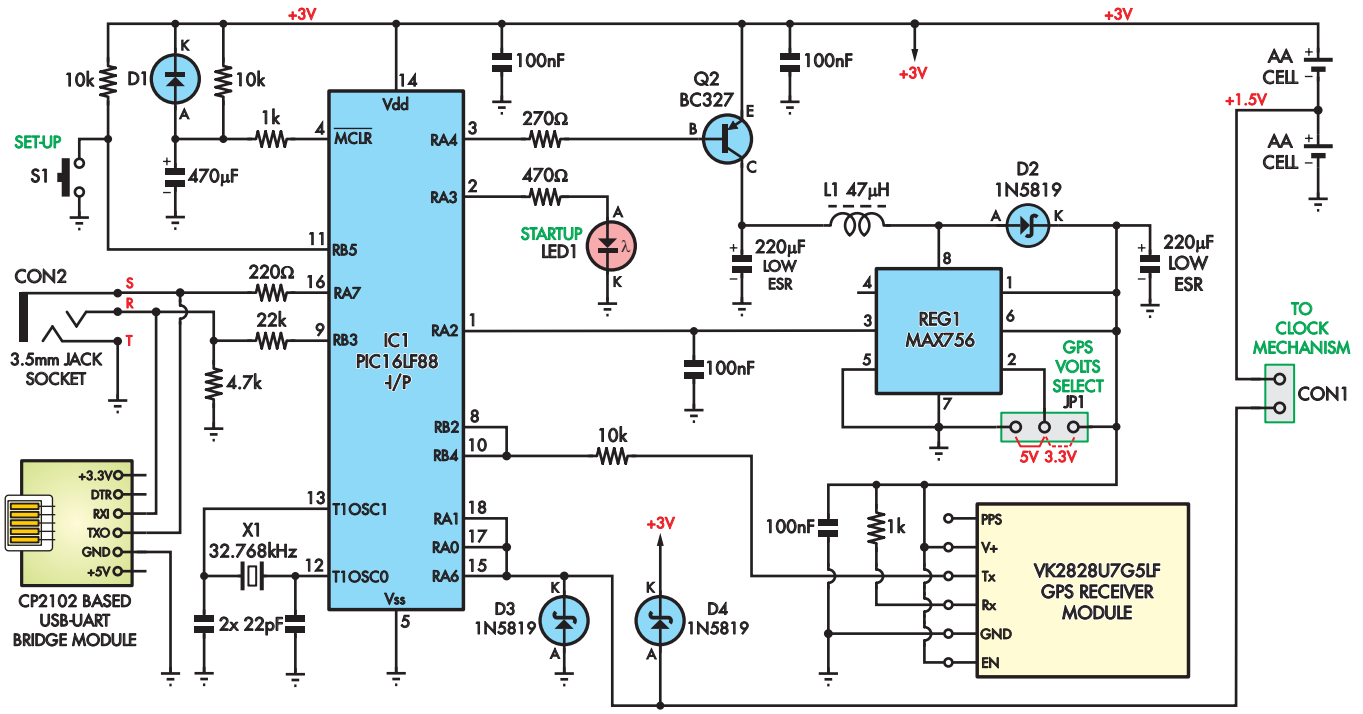
Now use double-sided tape to attach the two cell holders and the GPS module to the board. This is important because it prevents the solder joints from breaking when you insert and remove cells. Solder and trim the cell holder leads. Be careful when soldering them as the plastic can easily be melted if you apply too much heat.

You can now strip the ends of the wire supplied with the GPS module and solder them to the pads with colour coding as shown in Fig.8, then plug the connector into the GPS module socket.

Loop a cable tie through the central hole of toroidal inductor L1 and down through the hole on the board, up through the other hole and tighten it, with the square plastic part on top of the board (so it doesn't stop it from sitting with the bottom side flat against the back of the clock later). Once L1 is held firmly in place on the PCB, solder and trim its two leads.

The PIC16LF88 (IC1) must be programmed using the file coded 0420217A.hex (for a stepping second hand) or 0430217A.hex (for a sweep second hand), both of which can be downloaded from the *EPE* website.

Alternatively, you can purchase a pre-programmed microcontroller. Either way, once it has been programmed, straighten its pins and plug it into the socket with its notched end aligned



GPS SYNCHRONISED CLOCK

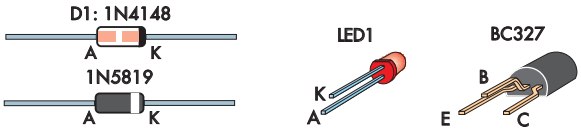


Fig.7: complete circuit for the *GPS Analogue Clock Driver*, omitting only the clock movement which connects to CON1. Microcontroller IC1 powers up the GPS module via transistor Q2 and boost regulator REG1, and receives its serial data stream at pins 8 and 10. When the GPS module is not powered, it uses its internal Real-Time Clock and watch crystal X1 to keep time and produce the pulses from output pins 15, 17 and 18 to drive the clock mechanism. Note that there is no Q1 due to a late circuit update.

Parts list – GPS- Synchronised Clock Driver

1 PCB available from the *EPE PCB Service*, coded 04202171, 140 × 61.5mm
1 VK2828U7G5LF GPS module*
1 CP2102-based USB/serial interface module with microUSB socket#
1 32768Hz crystal (X1)
1 47µH 1A+ inductor
1 small cable tie
1 3.5mm switched stereo socket (CON2)
1 vertical PCB-mount tactile momentary pushbutton switch (S1)
2 single AA PCB-mounting cell holders (Altronics S5029)
1 18-pin DIL IC socket
1 3-way pin header, 2.54mm pitch, plus shorting block (JP1)
1 2-way polarised right-angle PCB-mount header, 2.54mm pitch (CON1)
1 2-way polarised header plug, 2.54mm pitch
1 short length light duty twin lead
1 short length tinned copper wire or component lead off-cut
2 AA alkaline cells

Semiconductors

1 PIC16LF88-I/P microcontroller programmed with either 04120217A.hex (stepping movement) or 04130217A.hex (sweep movement) (IC1) #
1 MAX756CPA DC-DC Converter (REG1; element14 1290853, RS 786-1287)
1 BC327 PNP transistor (Q2)
1 1N4148 diode (D1)
3 1N5819 schottky diodes (D2-D4)
1 3mm high-brightness LED (LED1)

Capacitors

1 470µF 10V electrolytic
2 220µF 25V low-ESR electrolytic (Jaycar RE6324, Altronics R6144)
4 100nF 50V MKT, ceramic or multi-layer ceramic
2 22pF ceramic

Resistors (all 0.25W, 5%)

1 22kΩ 3 10kΩ 1 4.7kΩ
2 1kΩ 1 470Ω 1 270Ω
1 220Ω

* this module suits the PCB pattern and also has an integral antenna. Other modules can be used but they may have different pin-outs and cable arrangements and some may require an external antenna.

available from the SILICON CHIP online shop

with the socket (ie, towards the top of the board).

Finally, place a jumper on header JP1. We recommend using the 3.3V setting with the specified module; although this is the minimum specified operating voltage for the VK2828U7G5LF, it will reduce the power consumption while the GPS is active by around 35% and should not affect performance. If you have trouble getting it to work, you can switch to 5V later. If you're using a different GPS module, check its data sheet to see what supply voltage it needs before fitting the shunt. If you leave it off, it could damage the GPS module.

Powering up

At this point, temporarily unplug the GPS module so that you can make some tests. With IC1 in its socket, insert two fresh cells in the battery holder. After a second, you should see one flash from the Startup LED (LED1), followed by a further two flashes another second or so later. These indicate that the microcontroller and the DC-DC converter, respectively, are working. If you do not get these indications, refer to the section below on troubleshooting.

After the double flash, the microcontroller will wait for two minutes, expecting some data from the GPS module before shutting down the DC-DC converter. In this time, you need to measure the voltage at the connector to the GPS module. Ours measured 3.33V and you should get a similar reading. If it's below 3.3V, consider removing a cell and changing to the 5V setting. If you do, it's a good idea to re-measure the voltage to ensure it's correct.

Now that you have confirmed that you will not blow up your GPS module you can remove a cell and plug in the GPS module. Finally, replace the cell and the controller should go through the whole startup sequence as described in the section on troubleshooting.

Modifying the clock mechanism

Now it's time to connect the driver to the clock movement, which involves removing the existing quartz-crystal-based drive circuit and replacing it with a cable to go to the new driver board. Start by removing the cover from the clock mechanism. Identify the leads to the stepper motor coil, cut these, strip them and solder them to a twin-core lead terminated with a 2-way header plug. Insulate the solder joints and anchor the cable (eg, using some silicone sealant) before replacing the cover.

The stepper motor coil should be easily identified, as it will be a large coil of enamelled copper wire. Every clock is different, so you will be on a journey of discovery here.

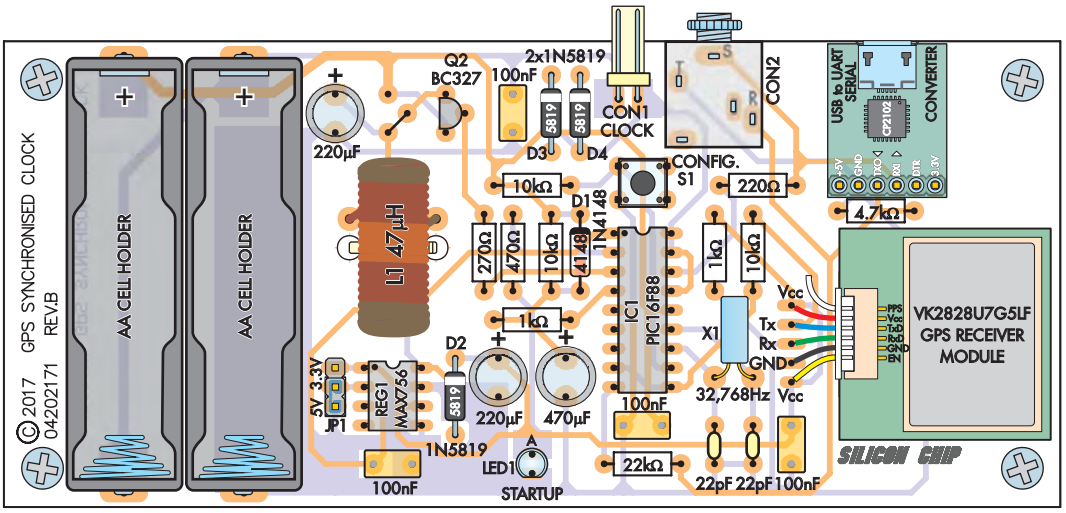
You can check your modification by using a 1.5V alkaline cell. Just connect the cell to the wires leading to the stepper motor coil, then reverse the cell and repeat. On each connection, the clock's second hand should step by one second (for a stepping clock) or 1/16th of a second (for a clock with sweep hands).

The method of attaching the driver PCB to your clock will also vary, but in the simplest case you can use double-sided adhesive tape to hold it onto the back of the clock.

Troubleshooting

Hopefully, your clock will work first time, but if it does not, you can use the Startup LED (LED1) to help isolate the problem. This LED will flash during normal initialisation (when the set-up button is not pressed) to indicate that each step of the initialisation has been completed. The point at which it does not flash will indicate where you should start hunting. When you insert the battery, you should see the following signals in sequence:

- **One flash:** the microcontroller has started up. If you do not get this then something is fundamentally wrong with the microcontroller or the cells.
- **Two flashes:** the MAX756 DC-DC converter has started up (determined by measuring a voltage on pin 3 of REG1 via pin 1 of IC1). If you fail to get this signal, check REG1 and its associated components. Check for about 2.7V (with fresh cells) on the collector of Q2 and between 1.23V and 1.27V on pin 3 of REG1.
- **Three flashes:** the GPS module is working and has transmitted its startup message. If you do not get this then check the wiring to the module and that the GPS power supply is between 3.3V and 5.5V. If you have an oscilloscope, check that there is less than 150mV peak-to-peak noise superimposed on the supply rail to the GPS module.



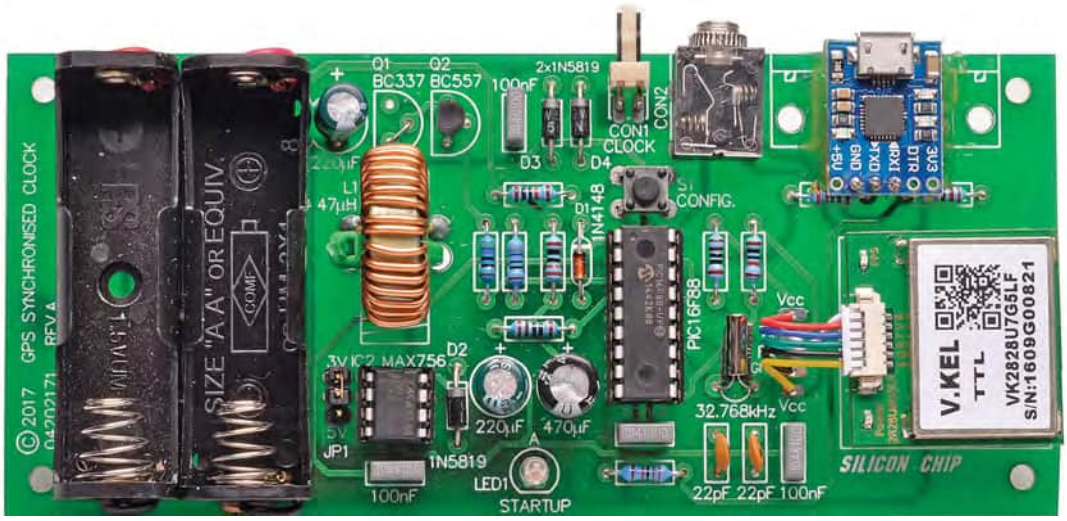


Fig.8: follow this PCB overlay diagram and the same-size photograph below* to build the *GPS Analogue Clock Driver*. Use a socket for IC1 but not REG1. If you use the specified GPS receiver, it will be supplied with a cable colour coded as shown here. Otherwise, you will need to determine the module's pinout from its data sheet and match it up to the labels on the PCB. If it has an enable input, it should normally be tied high (ie, to VCC) for normal operation but check the data sheet to make sure.

***Note that this photo is of the prototype – there is no Q1 (it has been bridged out with a link) and Q2 is now a BC327 (not a BC557), as shown in the overlay diagram above.**

- **Four flashes:** the GPS module has locked on to sufficient satellites and has responded with an accurate time signal. This can take up to 90 seconds or more, so be patient. If you don't get this, try putting the board closer to a window and open any metal blinds. If your indoor GPS signal is poor, you will need to keep this in mind when choosing a location for the clock.

Immediately following the GPS lock (four flashes), the clock should double-step around the dial to reach the correct time (assuming a stepped second hand.

If this does not happen, it means that the crystal oscillator (X1) is not working or the clock's stepper motor is not correctly wired to the controller. In particular, check that you have isolated the clock's electronic module and soldered your wires properly to the stepper motor coil. See the 'Setting it up' section below for more information on how to check the connection to the clock motor.

Testing the clock drive

For stepping clock mechanisms, the most important test is that the drive pulse is long enough to reliably step the clock with a supply as low as 2V. If you have a bench supply, you can use clip leads to connect its negative output to the spring in the right-hand cell holder and its positive output to the cathode of D4. You will also need to wire a 47Ω resistor across each cell holder, to provide the 'centre tap' voltage to drive the clock mechanism.

If you don't have a suitable supply, you will just need to scrounge up some almost-but-not-quite-completely-dead AA cells that produce close to 1V each under a moderate load.

Either way, you just need to leave the clock running for a few minutes and check that it doesn't miss any steps. If it does, use the set-up menu (explained below) to increase the pulse width by 8ms and try again. Repeat if necessary, until it works reliably.

Another point to note is that you must sit the clock upright in its normal position while testing. The clock's motor has very little power and if it is going to misbehave, it will occur while the clock is trying to push the second hand up against gravity.

Sweep movements need to be tested more thoroughly and the firmware has a function in the setup menu that makes this quite easy. It will run the clock for an exact number of minutes and then stop. A good test is for 60 minutes and the idea is that the minute and second hands should return to exactly the same spot as they started from. Any error, even by half a second, will indicate a problem.

Once again, you should run this test with a 2V supply, if at all possible, as explained above. It is at that low voltage point that problems will surface if they are going to.

As with the step movement, orient the clock vertically during testing. If the clock does lose some time, the answer again is to increase the pulse width in the set-up menu. This allows the pulse width to be varied in steps of one millisecond with increasing values delivering more energy to the clock's motor at the cost of battery life.

Note that you need to start the test at a normal voltage (about 3V) because the serial interface will not work at low voltages and the clock will not start running at low voltages. Once the test has started running, you can reduce the supply voltage. If you don't have a variable supply, this may be possible to arrange by initially paralleling fresh

Calculating battery life

With an application such as this, battery life is important. After all, what is the point of a clock that does not need adjustment if you are forever changing the batteries? To calculate the consumption, we need to divide the activity of the circuit into phases according to the current drawn from the battery.

Then, for each phase, we determine the current consumption and its duty cycle (the percentage of time that the current is drawn). Finally, we can calculate the average current drawn per hour and then the battery lifetime for a given battery capacity. The tables below are the results for our prototype.

These tables indicate what is the major power user and this is the current drawn while driving the clock's stepper motor. This is where you should concentrate your efforts if you wish to improve the battery life. One way to do this is to reduce the width of the pulse using the set-up menu, but you have to be careful doing this because you may cause the clock to become inaccurate at lower battery voltages.

If you plan to experiment with this, you should connect a variable power supply (with simulated centre tap) in place of the battery and test that your clock steps correctly at less than 2V, the minimum expected battery voltage. Don't just test it on its back either; stand the clock upright in its normal position as you might find that the stepper motor does not have enough power to lift the second hand against gravity.

Power consumption for clocks with stepping hands

Function	Current drain (mA)	On time (seconds)	Total time (seconds)	Duty cycle	Consumption (mAh)
PIC in sleep	0.004	158355	158400	99.97%	0.004
Clock step pulse	3	0.04	1	4.00%	0.120
During GPS sync	80	45	158400	0.03%	0.023
Battery self discharge*	0.009	1	1	100%	0.009
Total					0.158

Expected lifetime for alkaline AA cells (capacity of 2400mAh) in months: 21

Power consumption for clocks with sweep hands

Function	Current drain (mA)	On time (seconds)	Total time (seconds)	Duty cycle	Consumption (mAh)
PIC in sleep	0.004	79200	158400	50%	0.002
Clock step pulse	0.6	0.5	1	50%	0.300
During GPS sync	80	45	158400	0.03%	0.023
Battery self discharge*	0.009	1	1	100%	0.009
Total					0.334

Expected lifetime for alkaline AA cells (capacity of 2400mAh) in months: 10

cells with the slightly flat cells, then disconnecting them later to more thoroughly test the arrangement.

Setting it up

The set-up menu varies depending on which firmware you have installed. That's because the sweep hands firmware does not support DST changes, so the related options have been eliminated. The menu for clocks with step hands is shown in Fig.9 and for sweep hands, in Fig.10.

For clocks with stepping hands, by default the controller is configured for Australia in the NSW, Victorian and Tasmanian time zone and DST rules. If you live in these states and the government has not changed the DST rules since January 2017, then you do not need to do anything.

If you live elsewhere, you will need to change the settings by connecting the *GPS Analogue Clock Driver* to a

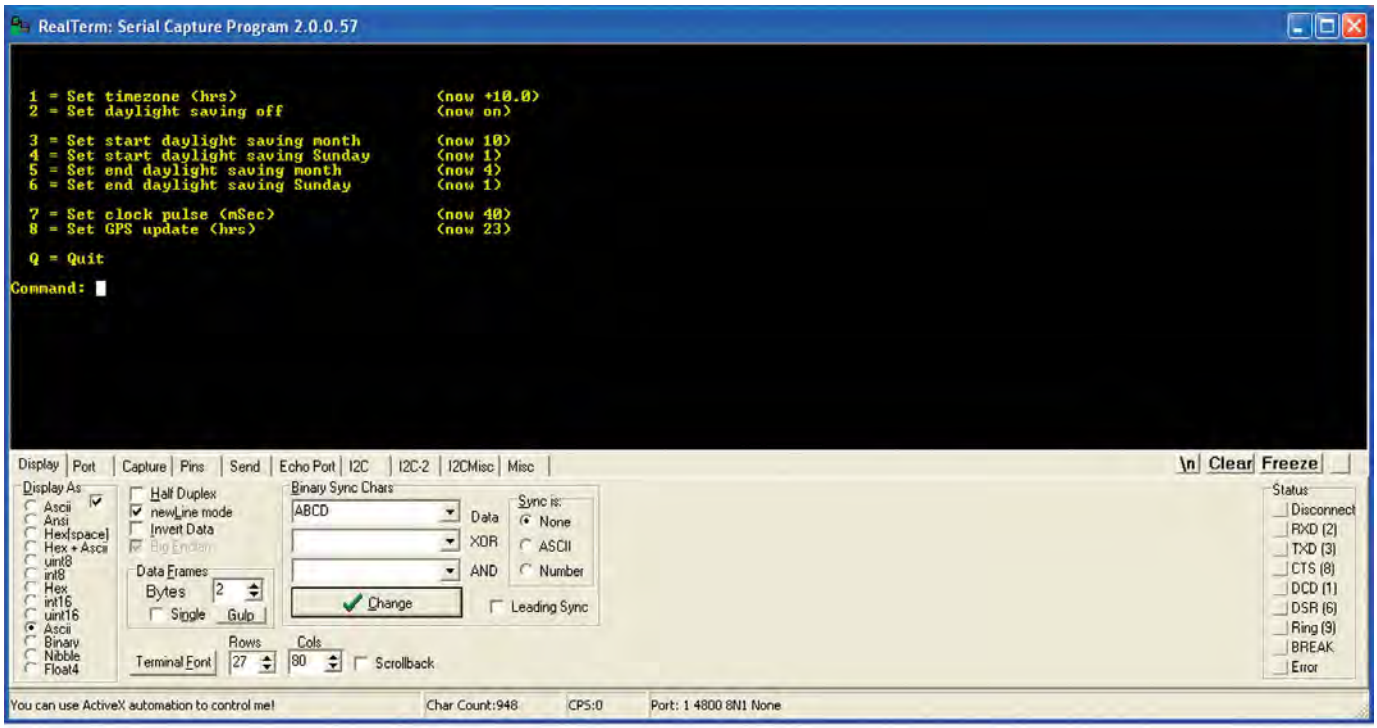


Fig.9: connect the unit to your PC using a microUSB to USB cable, configure a terminal emulator, hold down switch S1 and insert a pair of fresh AA cells to access the configuration menu. The one shown here is for clocks with a stepping second hand. Changing settings is fairly self-explanatory once you've established serial communications.

USB port on your PC via the onboard adaptor. Or if you have a PICAXE programming cable, you can connect this to CON2 instead.

You will also need a serial terminal emulation program running on your computer configured for 9600 baud, 8 data bits, no parity and one stop bit. Many free programs are available on the Internet, including TeraTerm Pro, PuTTY, RealTerm or Hercules Terminal Emulator. Use Google to search for one or more of these names.

To enter set-up mode, hold down the Set-up button (S1) while you install fresh cells and continue to hold it down until you see the menu via the terminal emulator on your PC. The Startup LED (LED1) will also flash when the microcontroller transmits a character to your computer, and this may help in diagnosing communication problems.

If your loction observes DST, you can select any month (1-12) for the end and start. You can also set the day for the event (1st, 2nd, 3rd or last Sunday in the month). The time of the day that DST starts (2am) is fixed in the program, as is the end time (3am).

For either type of clock, the clock pulse width can be changed in steps of 1ms and this setting might need to be adjusted to suit your clock.

Most clocks work with the default setting, but some may need slightly longer pulses to reliably step with a low battery voltage. Also, to gain a little extra accuracy or improve battery life, you can change the interval between GPS synchronisations.

All changes are saved in non-volatile memory and therefore will be retained, even when you remove the battery.

Setting the time

We explained this earlier, but you may not remember the details, so here's a quick run-down.

For clocks with stepping hands, simply set it so that all the hands point at the 12 o'clock position and insert the cells. Once the GPS module has a good signal and IC1 is able to determine the correct time, the hands will 'quick-step' around the dial until the time is correct and then it will run normally.

To save the clock from having to double-step for hours to reach the correct time, it makes sense to power up the clock shortly after 12 o'clock (ie, your local time).

In that way, it will only take about ten minutes or so for the clock to finish double stepping and revert to normal accurate time keeping.

For clocks with sweep hands, it's a bit more tricky. First, check the current time and then set the hour and minute hands so that they are pointing to the immediately following half-hour.

For example, if it's 3:08, set the clock to show 3:30 before inserting the cells. But there's a problem in that the second hand will be pointing at some random position on the dial and when you insert the battery, the clock will sit motionless until it is time to start.

As the time adjustment on most clocks does not affect the second hand you will not have an opportunity to set the second hand to 12 o'clock before the clock starts – and then it is too late.

To solve this, while the clock is waiting for the half-hour to roll around (during which time LED1 flashes slowly), you can press the set-up button (S1) and while you hold this button down, the clock will run, causing the second hand to move around the dial.

When the second hand reaches the exact 12 o'clock position, release S1 and use the normal time setting facility of the clock to adjust the hour and minute hands to the correct position.

A close-up of the micro-USB module (left) and the optional 3.5mm programming socket (CON2, right).

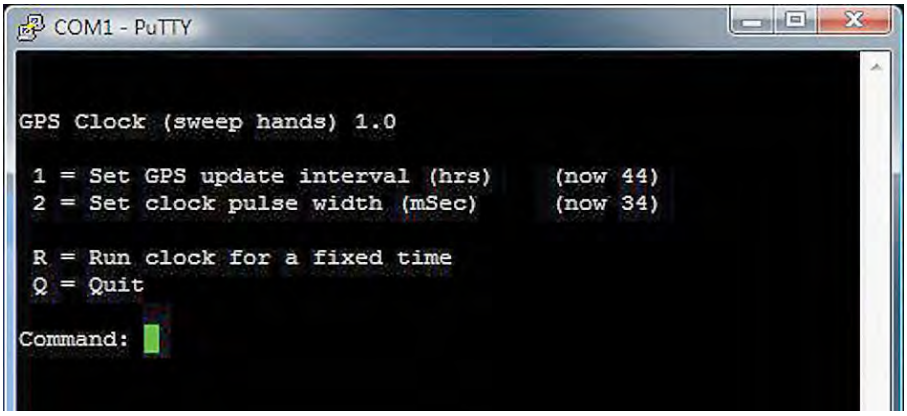


Fig.10: the set-up menu for clocks with sweep hands, shown here, is much simpler than for stepped hands because it does not include any of the DST options. However, it does include the option to run the clock for a fixed time so that you can check that it isn't losing any time. This should ideally be checked with a supply voltage of around 2V (see text).

Source code

The firmware for this project is written in the C language and can be compiled with either the CCS C compiler or the Hi-Tech C compiler Lite for PIC10/12/16 microcontrollers.

The Hi-Tech C compiler was purchased by Microchip some time ago and is now obsolete, but it can still be downloaded and used. The good thing about it is that it is totally free, so if you want to get into the C language and play around with the code, this is a good way to do it.

Download links and installation instructions are available at: **www.cs.ucr.edu/~eblock/pages/pictools/install.html**

Conclusion

Well, that's it. With your clock properly set up, you can hang it on the wall and be assured that at least one clock in the house is always accurate. Just make sure it has a decent GPS signal where it's located (eg, not deep inside under a corrugated iron roof!) so that it will stay synchronised. (Note that you can also check the clock's accuracy at any time if you have Internet time enabled on your desktop computer.)

Here's how we secured the PCB to the clock – a little bit of judicious filing removed a couple of ridges, then a few dollops of silicone sealant holds the PCB securely in place. This method allows easy battery change later on.