The

ZL1BPU SUPERCLOCK

USER MANUAL

This project is a practical application of precision timing. Time and frequency measurement are closely related, indeed one is the reciprocal of the other, and so an accurate frequency reference can make a very accurate clock.

Build this little clock and you will be more than impressed by its performance - it always tells the correct time, down to fractions of a second, and is unaffected by temperature or power glitches. Adapt the software yourself to add a wide range of functions.

This manual refers to firmware Version 0.2 (SUPRCLK2.xxx)

Murray Greenman ZL1BPU
December 2002

Introduction

Accuracy of frequency standards is usually quoted as parts in 10^6 (ppm). The time accuracy of clocks is rarely quoted, but it works out that one second per 11.5 days, or three seconds per month, is the same as one part in 10^6. Three seconds per year is about the same as one part in 10^7.[1] This project should provide this level of performance. Any better is unnecessary, as clocks displaying civil time (with or without "Daylight Saving") require attention twice a year anyway.

Electronic clocks and watches use a quartz reference, either 32.768 kHz, or 4194.304 kHz. While the performance of these isn't too bad, typically within a few seconds per week, it is easy and relatively inexpensive to do much better. In this project a TCXO[2] is used to provide good temperature stability and performance resulting in an error of a few seconds per year. TCXOs are widely used in UHF radio equipment, such as cellular phones, which need to stay on frequency over extended periods and a wide temperature range. They are also used in GPS equipment, frequency counters, high performance signal generators and other precision equipment. This project uses a TCXO from a surplus AMPS cellular phone (junked ones can often be obtained for nothing from cellular service centres if you ask nicely).

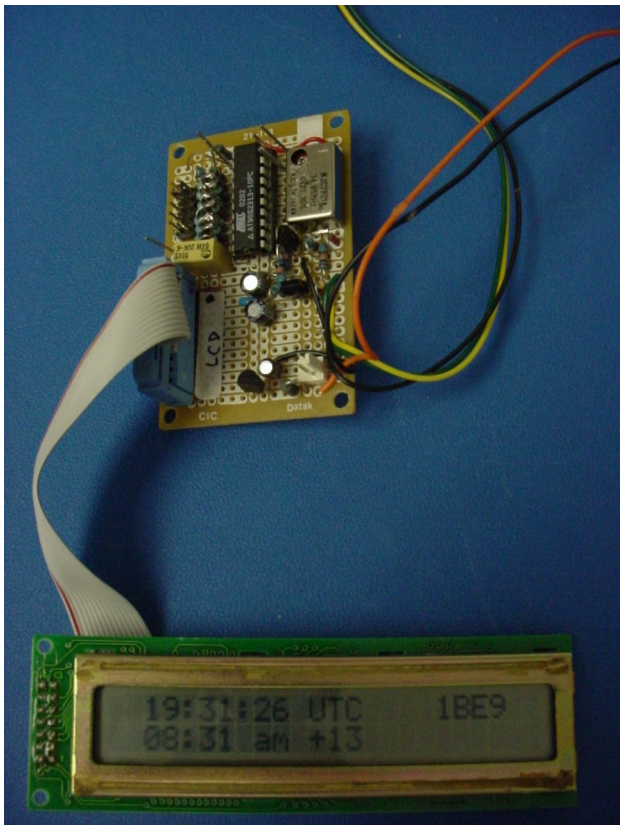

Fig. 1. The ***Superclock*** prototype

See Fig. 1. The project uses an inexpensive micro controller, a TCXO rescued from an old cellular phone, and an LCD display, also recycled from an old office telephone, for minimum cost. The unit is built on a simple Dick Smith Electronics H5608 project board. As well as UTC time and local time, the ***Superclock*** provides two reference signals, a 1 kHz square wave and a 1 Hz (1pps) pulse 1 ms long. These can be used as references for other devices, or as part of the calibration procedure. The ***Superclock*** also provides time as a message via an RS232 serial port.

[1] See Appendix "About Frequency References"
[2] Temperature Compensated (or controlled) Crystal Oscillator

Description

The micro controller has a hardware counter/timer, which is used to divide from the TCXO reference frequency and generate an interrupt at 2 kHz. Thus any TCXO frequency that is a multiple of 2kHz can be used. The micro does not need to be involved in the counting of this high frequency, or the capture of reference phase, which happen in hardware inside the chip, even while the micro is asleep.

The 1 kHz output pin is toggled in the 2 kHz timer interrupt, assuring that the duty cycle of the 1 kHz output signal is accurate at 50%. Every alternate interrupt, a further software counter is decremented. This counter divides by 1000, and is used to generate the 1 Hz pulse and trigger the timekeeping process. This pulse is precisely one millisecond wide, and occurs exactly at the second event. With a little added code, there could be additional outputs, such as 50 Hz or pulses to drive a clock mechanism. The software processes just described happen every half millisecond, and between these interrupts, the micro controller is asleep.

Once every second, a flag is set by the interrupt divider, and this allows the main program to operate briefly before going to sleep again. The main program adds one second to the UTC time, carries seconds to minutes, and minutes to hours, calculates the local time offset, checks that the power supply is OK, then sends and displays all the information, before going to sleep again. Because these processes happen so quickly, the micro controller spends most of its time asleep, and the average current drawn from the power supply is low, just a few milliamps.

Power for the ***Superclock*** is provided by a small AC adaptor, and is backed up by a small 9V alkaline battery, to ride out power failures. The battery should last its shelf life. For a more portable clock application, a six-cell NiCd pack would also work well, and could be charged by fitting a resistor (R1) across the battery's isolating diode D3 (see Fig. 2).

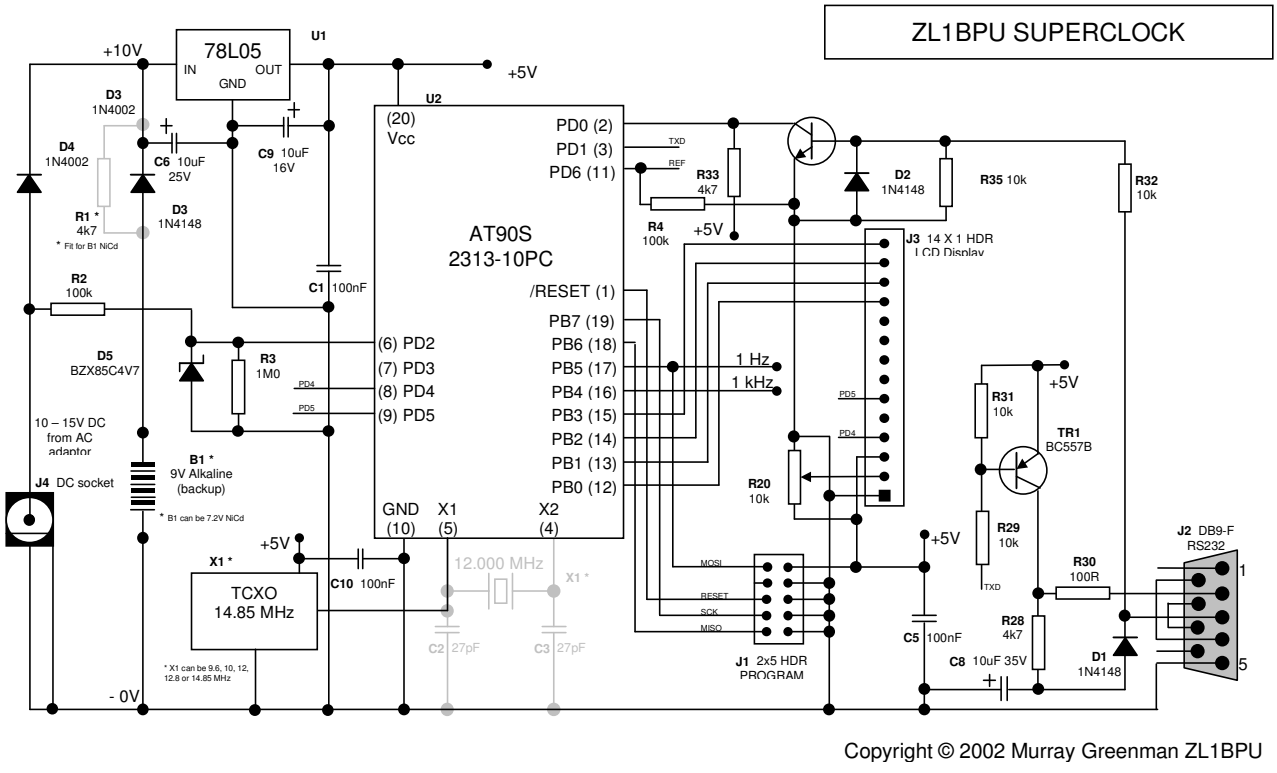

Fig. 2. The ***Superclock*** schematic

In the schematic, note the power supply arrangements on the left, with the diode isolated backup supply and AC supply, using a plug-pack. When AC power fails, the input micro input PD2 (pin 6) goes to zero, and the micro displays “BATT” on the display. If a DC battery supply is used, it might be useful to know when the

supply drops below the back-up battery rather than when it drops to zero, so to achieve this, replace R2, D3 and R3 with a comparator.

Next in the schematic are the 5V regulator U1 above and the TCXO X1 below. The micro is an inexpensive ATMEL AT90S2313, available in New Zealand and Australia from DSE, Polykomm or Jaycar. To the right are the RS232 interface, TR1 and TR2, the LCD connector J3 and the programming header J1. There really is very little to it! A discrete transistor RS232 interface is used to reduce supply current drain. Current is drawn from the supply only when transmitting data.

If the REF input (micro pin 11) isn't used, it should be grounded. This is achieved by R4. The use of this phase detector input is described later. Other spare pins are outputs and can be left open.

There is of course considerable similarity between the circuit of the ***Superclock*** and previous projects. See Fig. 3 for a detailed view of the circuit board. The LCD connects at the top left, and uses display routines specially written for operation at up to 15 MHz. The trim pot top centre sets the LCD contrast. To the right of the pot are the programming header and some unused resistors, left over from a previous project. The micro is centre right, and the power and serial connections lower left.

The two-pin connector bottom left is for the backup battery. The wiring underneath is a combination of solder-bridged pad connections and short lengths of fine insulated wire. The two centre tracks under the micro are 0V (lower) and +5V power (upper).

The same low cost RS232 interface used in other projects is in the lower centre of the board. At the far left centre is the little 5V regulator. The silver box lower right is the 14.85 MHz TCXO, and its adjustment trimmer can be clearly seen. This reference has through-hole pins, but a surface mount TCXO will work fine if you solder fine (fuse wire) leads to it before soldering to the board. An alternative crystal reference circuit is offered on the schematic, but performance without a TCXO will be severely compromised.

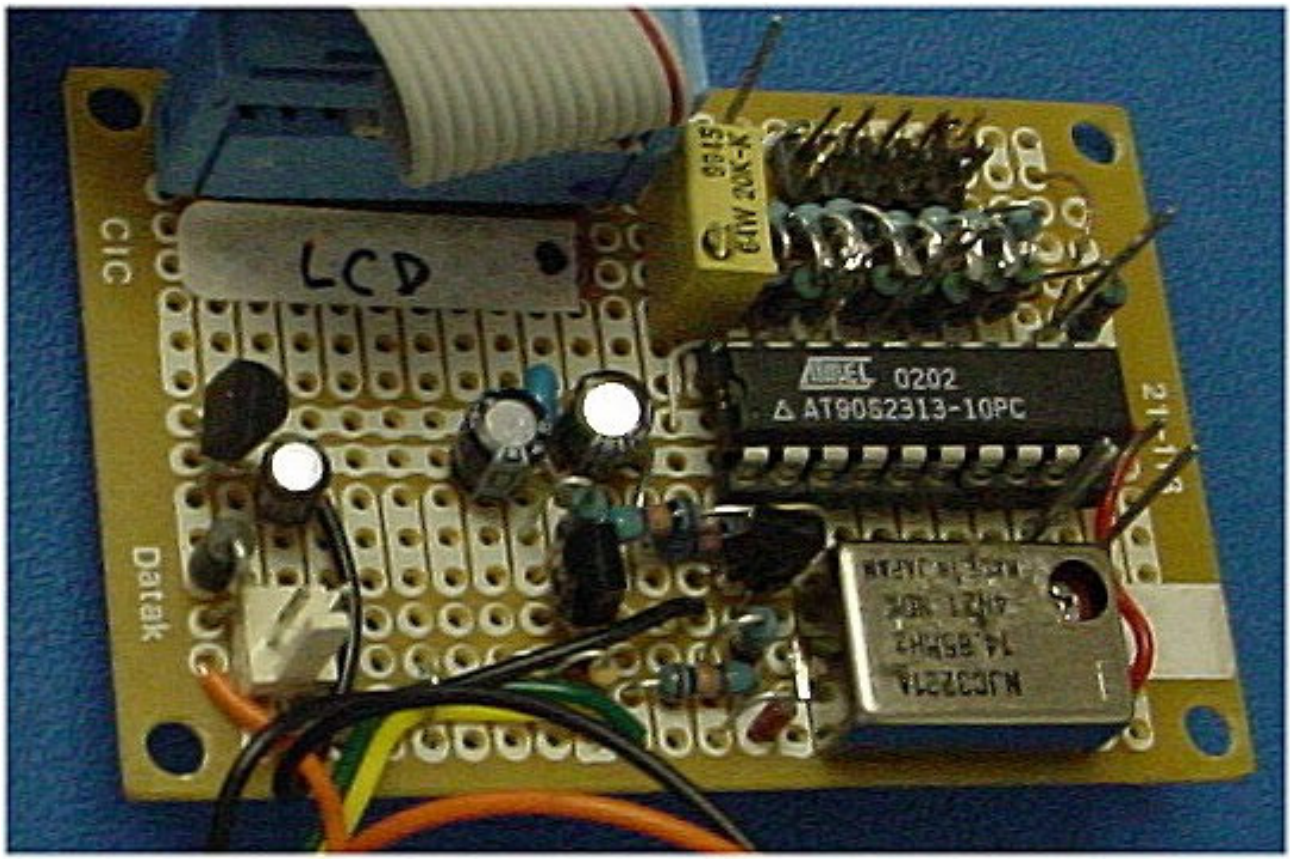

Fig. 3 Circuit board detail

One interesting feature of the ***Superclock*** is the way that time is counted. Unlike most micro controller projects, which operate in binary (e.g. a frequency counter or digital voltmeter), this clock operates in BCD,[3] in other words, it counts 0-9 and from 0 tens to 5 tens. In fact, the data is stored as "packed BCD" where the information about seconds, minutes and hours are stored in one register each.[4]

Every time a mathematical calculation is made, the result must be checked or adjusted to ensure that it is true packed BCD. Seconds and minutes are also checked to see that they do not exceed 59, and a carry and correction made if they do. This may seem a clumsy way to count, but it has the advantage that displaying the result is very straightforward - displaying both nibbles of the register as two ASCII characters simply displays 00 to 59. Time is displayed as "HH:MM:SS UTC" (see Fig. 4).

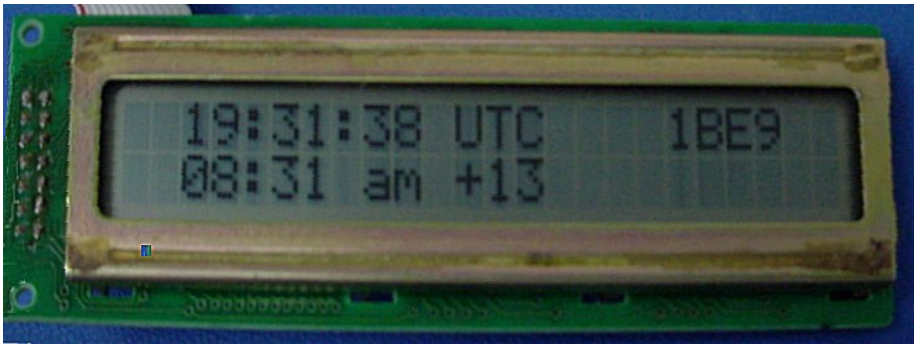

Fig. 4. The LCD display

An extra check is required after adding the hours, to return the count to zero when 24 is reached. Further checks are involved in the addition of the local time offset, and allow the offset to be either negative (west of GMT) or positive (east) with accurate results. For example, for New Zealand Daylight Time the offset is +13 hours. It might seem no big issue, but performing calculations like this in BCD using machine code is not always straightforward.

The local time is displayed as "HH:MM am +AA" (or "pm") where "AA" is the local time offset. For negative offsets, the "2's complement" of the number is entered; "FF" for -1, "FE" for -2 etc.

[3] Binary Coded Decimal
[4] Packed BCD keeps two decades in one byte. In this case, units (right 4 bits) and tens (left 4 bits).

Controls and Time Setting

There are no front panel controls on the ***Superclock***. After all, it is accurate and battery backed, so what would need adjusting? Having no controls also prevents the clock from being upset by children with fiddly fingers! The local time offset will need to be changed every six months thanks to daylight saving, and at this time the clock accuracy can be checked. Fortunately the local time offset can be stepped back and forth without affecting the time keeping. All adjustment is achieved using the serial link to a PC. There are eight commands:

Hnn	**Set UTC hours to nn (BCD) hours**
Mnn	**Set UTC minutes to nn**
Snn	**Set UTC seconds to nn**
+	**Add one to UTC seconds**
-	**Subtract one from UTC seconds**
>	**Increment local time offset by one hour**
<	**Decrement local time offset by one hour**
R	**Retard clock by 10 ms**

The first three commands **cause the clock to stop while data is entered**, so should only be used in the first stages of setting the time. They should be used in the order shown, while listening to VNG or WWV. Set the local offset, the hours, minutes, and finally set the seconds to about the right value, give or take a few seconds. At this stage, don't worry if the seconds are out by a few. However, if the clock seconds don't tick over exactly in time with the radio seconds pips, next use the "R" command repeatedly for very fine adjustment.

If an oscilloscope is available, trigger it from the 1 Hz pulse output of the ***Superclock***, and observe the seconds ticks from the radio signal on the screen. Without an oscilloscope, the 1 Hz ticks can also be line up with the time signal quite easily, and with remarkable precision, by listening to them both with a pair of headphones. If the "R" command is used too much, causing overshoot, another 100 presses are required to get close to the correct seconds phase again, so instead use the "S" command and try again.

After checking at the minute event, use the "+" or "-" command to correct the seconds display. It should change to "00" exactly as the minute tone sounds. The “>” and “<” commands to adjust local time can also be used any time, which is convenient, so stepping forward or back for “daylight saving” is a breeze. These four commands and the "R" command are the only ones permissible while the ***Superclock*** is keeping time.

The program source code for this project is provided free of charge,[5] so the constructor can make whatever modifications are preferred. There is plenty of code space, so small adaptations could for example provide:

- Time signal "pips", or even time codes similar to those used by VNG
- Additional or alternative reference outputs, such as 440 Hz, 50 Hz or 1 hour
- Chimes (in Morse!) on the hour, or even voice announcements
- An alarm or time controlled output function
- A timer or stopwatch
- A clock that keeps astronomical time or a tide clock
- Push-button or timer controlled display back-lighting

Front-panel controls could be added, provided they were scanned in the main program and did not stop the clock during adjustment. Using front panel controls for time setting is not recommended, for obvious reasons. The executable code is designed for any reference frequency from 8 to 15 MHz that is a multiple of 2 kHz, achieved by setting the division ratio, for example 7424 (0x1D00) for 14.85 MHz. Common cell-phone TCXO frequencies are 9.6, 10, 12.8 and 14.85 MHz. The only changes necessary for a different TCXO frequency are made in the EEPROM during setup. Four parameters are stored - the division ratio in two bytes (divide the TCXO frequency in Hz by 2000, then subtract one and convert to HEX), the serial port baud rate (also depends on reference frequency) and the default local time offset.

[5] See www.qsl.net/zl1bpu/micro/CLOCK

Calibration

Direct
An essential part of the setup process is to set the calibration of the TCXO. This can be done at any time. If an oscilloscope and TV receiver are available, the process is easy. With the oscilloscope time base set to 10us per division, trigger the oscilloscope from the 1 kHz output of the ***Superclock*** (PB4 pin 16), and observe the horizontal sweep pulse of the TV receiver by hanging an oscilloscope probe near the TV set. Alternatively, observe the video output of a video recorder or TV receiver. Tune to a TV station known to use a Rubidium standard (in New Zealand, TV1 or TV2). Although the line rate pulses will appear in several places on the oscilloscope screen, they should be stable, not drifting horizontally, or drifting very slowly. Watch one particular pulse and follow it for some minutes looking for drift. If there is drift, adjust the TCXO slightly until the drift is eliminated. The relationship between the 1 kHz pulses and the line frequency is complex (125:8), but despite this, the effect is easy to observe and adjust.

Unfortunately there is no simpler relationship between the reference and the TV line frequency, so the main divider cannot be set temporarily to provide a more convenient frequency, and this precludes using the simple method of observing pulses on the TV screen.[6] There are a number of other ways to achieve calibration, including counting the reference (at pin 4 of the micro controller, a method not recommended with a conventional crystal oscillator), or counting the 1 kHz output with a VERY good counter. The best way of all is to use the built-in phase detector. Adjustment by observing changes in clock time in the traditional manner would take upwards of six months to achieve similar results![7]

Phase Comparison
In Fig. 4 it can be seen that there is a cryptic number “1BE9” in the top right corner of the display. The software is intended to operate with a 16 x 2 character display, nicely centred, without this extra information. The picture shows a 24 x 2 display, and so time is offset to the left, revealing some "engineering" information to the right. The four characters are the hexadecimal phase count of the main high speed counter, with 0.1us resolution and 500us range, and can be used for calibration. The same information is added to the serial time message, so is still available when you use a 16 x 2 display. Normally the number is meaningless, and it won' t change if an external reference is not used.

If a precise reference pulse is connected to the PD6 input (pin 11), the negative edge will be used to sample the high-speed hardware counter phase. The reference frequency can be any accurate division of 2 kHz down to 1 Hz, and must have better precision than the TCXO, so a GPS 1pps signal or Rubidium-locked TV 50 Hz frame frequency can be used. The sampled value from the high-speed counter should change only slowly, up or down, and it may be possible to tweak the TCXO calibration so that there is no change over several seconds. Anything better than this is outside the capability of the simple TCXO. Even if the number changes by one every second, the error is very small - about 0.1 microseconds per second, or one part in 10^7. This technique can be used to detect offsets as small as 1 part in 10^9.

The Rubidium-locked TV network signal is a very convenient and much under-rated source of precision frequency. The colour subcarrier (4.43361875 MHz), line frequency (15.625 kHz), and frame frequency (50.00 Hz) are all locked to a Rubidium reference (1 part in 10^{10} or better).

The well-respected LM1881 sync separator chip[8] is an ideal source of the TV frame frequency 50 Hz reference. Use the video output of a TV receiver or video recorder to drive the chip. Operated from the ***Superclock*** ' s 5V supply, and fed with TV video from a Rubidium referenced TV station, this simple little 8-pin chip will generate very stable and precise pulses with the correct timing and levels to drive the REF input of the micro directly. See Fig. 5.

[6] Described in the Appendix
[7] Harrison spent many years regulating his first chronometers – and their performance (about one second / week) is eclipsed by modern quartz clocks.
See http://www.theorderoftime.com/science/sciences/articles/awalktroughtime.html
[8] See http://www.national.com/ds/LM/LM1881.pdf

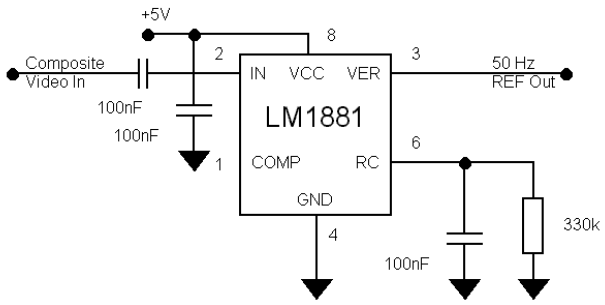

Fig. 5. 50 Hz Reference from TV Sync

The line sync output of the LM1881 is not useful for phase detection purposes in this design, for two reasons. First, it is a composite sync output, which contains twice-frequency equalizing pulses and frame pulses; and second, there is usually no easy relationship between the line frequency and the reference oscillator (4752:5, if using a 14.85 MHz reference). There is however always an easy relationship between the frame pulse and the reference (297000:1 with a 14.85 MHz reference). The only disadvantage is that phase drift detection is slower. This is really no problem, since the phase resolution is high.

Of course the TV receiver and sync chip only need to be connected for initial calibration. The ***Superclock*** will keep time adequately for years without further reference oscillator correction. It would be interesting to note the rate at which the phase changes after calibration, and to check it months later to see if the reference oscillator has aged noticeably. The ageing rate of TCXOs is typically about 1 part in 10^6 per year.

A very simple computer program is offered, which plots the reference phase over time and allows the TCXO to be trimmed. This program (SCLOCK) displays the reference phase on a line of text, every second. The UTC time is displayed, followed by a field of 64 spaces containing two special characters, and then the hexadecimal phase is displayed. The "*" special character indicates the coarse relative phase resolution of 32 samples), while the "." character indicates the fine phase. Using this software to monitor the RS232 output of the ***Superclock***, the user will see the offset of the reference as a slope in the column of "*" and "." characters. Hopefully, the TCXO trimmer can be adjusted so the "*" and even the "." Characters march down the screen in a straight line.

```
11.23.01                *    .                                  173F
11.23.02                *     .                                 1740
11.23.03                *    .                                  173F
11.23.04                *     .                                 1740
11.23.05                *     .                                 1740
11.23.06                *    .                                  173F
11.23.07                *     .                                 1740
11.23.08                *      .                                1741
11.23.09                *      .                                1741
11.23.10                *       .                               1742
```

In the above example of the screen display from SCLOCK.EXE, the phase is moving very slightly, as can be seen by the drift to the right of the "." characters.

Two versions of SCLOCK are offered - SCLOCK1.EXE for use on serial port COM1 at 9600 baud, and SCLOCK2.EXE for similar use on COM2. The source code SCLOCK.BAS, written for the QBASIC 4.2 compiler, is also available from the website.

The user can easily write much more sophisticated graphical software for monitoring reference phase (see the following section on Serial Data), but anything fancier than SCLOCK.EXE is quite unnecessary for occasional calibration use, and would probably be an over-kill for TCXO references of this performance.

Serial Data

Commands are sent to the ***Superclock*** in ASCII format, RS232, 9600 bps, N-8-1.[9] Lower case is interpreted as upper case. The commands are of two types – those that stop time keeping, and those that do not. Those that stop time keeping cause the ***Superclock*** to stop until sufficient data is received for the command to be processed, after which time keeping is resumed. In all cases two extra characters are expected. These are the **H**, **M**, and **S** commands.

Commands that do not stop time keeping do not need the ***Superclock*** to wait for data – these are the single character commands **+**, **-**, **<**, **>**, and **R**.

The clock transmits three separate data products. During normal operation, the UTC time and the reference divider phase are transmitted. Whenever a command (which stops time keeping) is sent to the ***Superclock***, it replies with a message. The data format is also 9600-N-8-1.

UTC Time

The UTC time message is transmitted every second, starting just after the second event. The format is **HH:MM:SS**, in eight bytes and is followed by a space. The time sent is the same as shown on the LCD screen. It would be possible to build and use the ***Superclock*** without an LCD display.

Reference Phase

Following the UTC time, the reference phase is transmitted as four ASCII characters **PPPP**, followed by CR/LF.[10] The value can be from 0x0000 (zero) to 0x1CFF (1999) or thereabouts, depending on the reference frequency used. The maximum value is the same value programmed into the EEPROM for the reference frequency divider during setup. This phase value can be manually tracked, or using a simple BASIC computer program, graphed over time.

Command Responses

When a command that suspends time keeping is received, the UTC time and reference phase message sent every second is suspended along with time keeping. These commands (**H**, **M**, and **S**) are echoed back to the PC, along with the command data, in the same format as the commands.

If a command is sent to the ***Superclock*** which is not understood, or if the data accompanying the command is not understood, the clock sends the error message "?", followed by CR/LF. Time keeping then continues, although a time-keeping delay will have been incurred.

Commands that do not suspend time keeping (**+**, **-**, **<**, **>**, and **R**) do not invoke command responses.

[9] No parity, 8 data bits, 1 stop bit
[10] "Carriage Return" (ASCII 0x0D) and "Line Feed" (ASCII 0x0A)

Programming and Setup

This procedure assumes the constructor uses the same programming tool as the author[11]. Since this is free, it's a reasonable assumption! If other tools are used, familiarity with them will be required, as the following procedure is specific to ISP V2.65.

The following are required:

- The ATMEL ISP AVR programmer, ISP.EXE V2.65 or similar.
- A programming cable.[12]
- The ***Superclock*** firmware SUPRCLKn.HEX, from the author's website ("n" is the firmware version).
- A PC running Windows 3.1, 95 or 98 to run the programming software.

Connect the DB25 end of the cable to the PC printer (parallel) port, and the 10-way header to the ***Superclock*** programming header. Make sure the cable is the right way round on the header – no damage will be done if it is wrong, but nothing will work.

If two computers are available, or the computer can run two applications at once, connect up the ***Superclock*** serial port to a PC serial port. You will need a "modem" serial cable (pin-to-pin with no crossover), NOT a "null modem" cable. Run a Terminal Emulation program (Windows 3.1 Terminal is ideal), setting it to the correct port, and to 9600 bps. This will allow progress to be monitored during programming and later allows the Clock to be set.

Run the ISP software ISP.EXE. There is also a DOS version; the most recent version supports 32 bit operating systems. The author recommends V2.65, which will work on a 486 PC running Windows 3.1, as well as later computers.[13] Make a project (from the menu, **Project**/**New Project**), selecting AT90S2313 as the device.

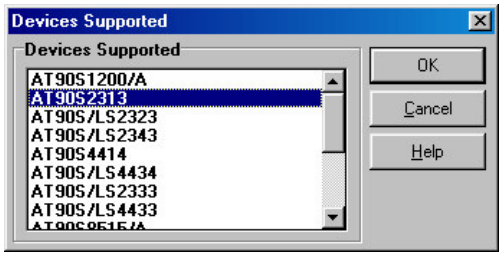

Fig. 6. Select the device

Press OK and then on the "Manager" tab of the next screen, give the project a title, and then from the main program menu select **Project**/**Save Project** to save the project file.

The next step is to check for communications with the micro. Select **Options**/**Change Printer Port…** and a "Port available" message will be observed. If all is well, select **CANCEL**. If not, try changing the port in case it isn' t LPT1.

[11] ISP.EXE V2.65, available from:
user.cs.tu-berlin.de/~sirhenri/sides/up_avr/avrisp.zip.
The latest version is available from www.atmel.com/atmel/products/prod203.htm
[12] See www.qsl.net/zl1bpu/micro/ for a suitable cable. The Jaycar KC-5340 kit will also do the trick.
[13] There have been problems with port timeout errors with later versions.

Next, select **Program/Read EEPROM**, which should be something safe to do (it won't matter since the micro is already blank). If all is well, the data from the micro will fill the "Data EEPROM Memory" window:

Fig. 7. The "Data EEPROM Memory" window

This window can be seen by clicking on it, or from the menu **Window/EEPROM Data Memory**.

If the cable is faulty or not connected, an error message like this will be displayed:

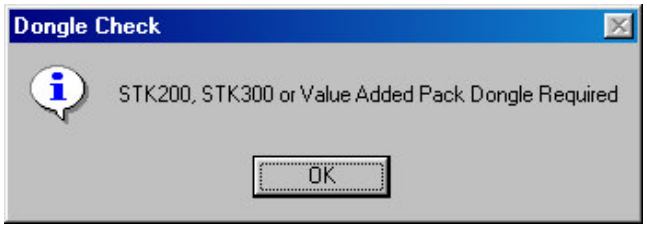

Fig. 8. Oh dear, no / faulty programming cable!

If the cable is connected, but there is a fault at the **Superclock** (perhaps the programming header is wired incorrectly, is plugged in backwards, or there is no power applied to the clock), there will be a different message - "AT90S2313 Part not Detected":

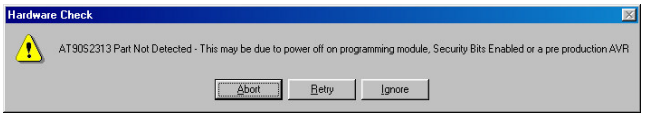

Fig. 9. No communications with device to be programmed

This message also appears if the wrong device has been selected in the Project Manager.

If everything has worked fine so far, the system is ready for programming. The software implicitly loads files into the current window, so in order to load the executable firmware, select the "Program Memory" window, by clicking on it, or from the menu **Window/Program Memory**.

Next select **File/Load…** and locate and select the executable file (SUPRCLKn.HEX). The current version is SUPRCLK2.HEX. The file is in "Intel HEX" format. The computer is now ready to program the device.

From the menu, select **Program/Erase** to erase the device, then **Program/Program Device** to program it. It should take 10 – 20 seconds, depending on the speed of the computer. If the device is faulty or not blank, it may take longer and will report errors. It may also take ages and report errors if the ***Superclock*** power supply is low. At this point the micro starts talking to the serial port, and garbled time messages should be seen on the Terminal screen.

The final step is to set the EEPROM parameters, which are contained in the first few bytes of EEPROM. The example shown below is for a 14.85 MHz reference, 9600 bps and a +13 hour default local time reference.

First row of EEPROM memory:

`FF FF FF FF`	First four bytes not used
`60`	Baud rate (9600 baud with 14.85 MHz)
`1D 00`	T1 reload for 2 kHz interrupt with 14.85 MHz TCXO
`13`	NZDT offset
`FF FF FF FF FF FF FF`	Rest not used

The ”EEPROM Data Memory” window will probably show all values as “FF”, so leave them as they are (it won’t matter what they contain). Simply plug the values shown into the first row of numbers. If the EEPROM in the device is blank, reading it will restore all values in the window to ‘FF”. Next, count along to the 5th byte in the first row, and change it to your required data rate. The next two bytes set the reference division, and the 8th byte the local time offset. So, for example, the first line of the EEPROM memory when ready to program should read:

```
FF FF FF FF 60 1D 00 13 FF FF FF FF FF FF FF
```

Then program the EEPROM. From the menu, select **Program**/**Program EEPROM** to load the data. It should take just a couple of seconds. If all is well, you should see time messages on the Terminal screen. If the message from the ***Superclock*** is still garbled, check your maths again, and check the Terminal data rate. Table 1 gives the value to load for a range of common TCXOs, and communication speeds of 9600 bps.

Crystal MHz	Baud Rate	T1 Reload
9.6	0x3D	0x12BF
10.000	0x40	0x1387
12.000	0x4D	0x1770
12.800	0x52	0x18FF
14.850	0x60	0x1D00

Table 1. Typical EEPROM values

Specifications

Clock –
UTC time **00:00:00 UTC** to **23:59:59 UTC**
Local time **00:00:00 am** to **11:59:59 pm**
Local offsets ±13 hours
Independent setting of offset, hours, minutes, seconds via serial link
Incremental time trimming in 10ms steps via serial link
No front panel controls, one internal adjustment (reference frequency)

Reference –
TCXO at any frequency 8.00 to 15.00 MHz that is an exact multiple of 2 kHz
Typical offset 1 part in 10^6 or better
Typical error over temperature range1 part in 10^6
Typical ageing rate 1 part in 10^6 per year
Calibration using phase detector to 1 part in 10^7 (3 sec per year)

Outputs –
UTC time, reference phase via RS232, 9600-N-8-1
1000.0 Hz 50% square wave, CMOS 5V level
1.000 Hz 1ms pulse, CMOS 5V level, positive true

LCD Display –
16 x 2 standard HD44780 display, in 4 bit mode (larger displays optional)
UTC time, **00:00:00 UTC** to **23:59:59 UTC**
Local time, **00:00:00 am** to **11:59:59 pm**
Local offset
Reference phase, four character (only on 24x2 or bigger display)
Power failure message "BATT", four character (only on 24x2 or bigger display)

Power Supply –
AC supply via 10 – 15V DC adaptor, 200mW max
Backup from 9V alkaline battery 15mA (about 8 hrs) or 500mAH NiCd cells charged in-circuit (several days)
External 12V DC supply operation 10 - 15V DC
True glitch-free no-break operation, power failure detection

Firmware –
Written in machine code for ATMEL WAVRASM compiler
Open source code, free for private use
Code is copyright (no publishing or distributing patched code without permission)
Code size about 1k bytes
Plenty of room for adaptations and improvements

Appendix - About Frequency References

Not everyone is interested in precision frequency references, but those of us that are, seem to have a passion for the subject. Here are some simple techniques for making and calibrating references that offer good performance without breaking the bank.

Using a Frequency Reference

Most Amateurs are content with the calibration (or lack of it) of their HF transceiver, perhaps use a simple crystal calibrator with older or homebrew gear, and never even think about the accuracy of their VHF equipment. Amateurs who enjoy constructing, maintaining or adapting equipment often invest in or build a frequency counter, but how well is it calibrated? What do they check it against? A frequency reference is what is required.

The first thing most folk think about when considering frequency references is to use a Standard Frequency HF transmission, such as from WWV or VNG. Those keen on the Frequency Measuring Contest will also know about HD2IOA (Ecuador) which transmits on 3810 kHz. However, while these standards may be very precise, the signals received from them in New Zealand can be inconvenient (no propagation when you need it) and quite inaccurate, due to propagation effects. Fig. 15, which is a spectrogram, a graph of signal frequency versus time, 20 Hz wide and about an hour long, shows how bad reception can be. The wobbly line that splits in two in places records the received carrier frequency of VNG on 16 MHz. The reasonably straight line is my own local frequency reference, GPS locked. VNG moves in frequency up to ±5 Hz as a result of Doppler, caused by movement of the ionosphere' s reflecting layers.

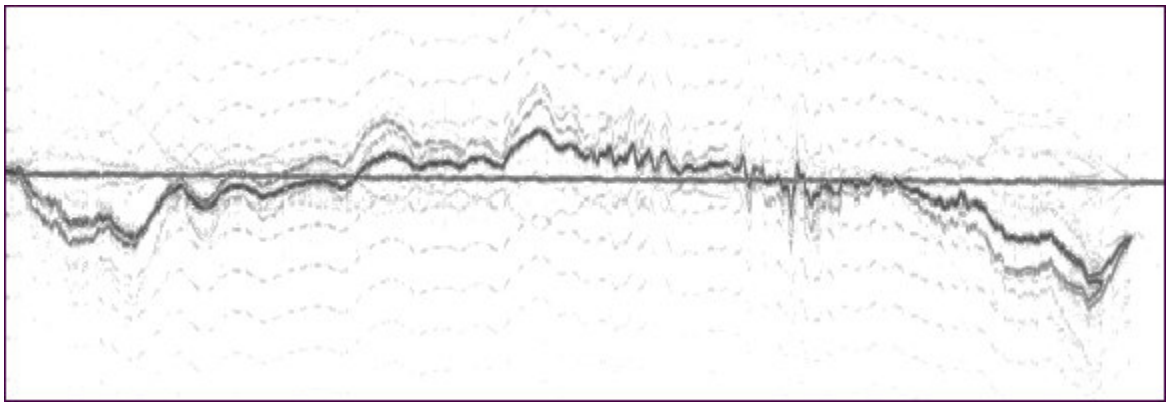

Fig. 1. VNG reception on 15 MHz

For low accuracy calibrators, beating the oscillator against WWV or VNG may suffice, but for anything more accurate, this technique is quite inadequate, as can be surmised from the picture.

There are other frequency references freely available in New Zealand with much better performance. The frame, line and colour burst frequencies of the TV1 and TV2 networks are all controlled by a Rubidium Standard.[14] The carrier frequencies of JJY on 40 and 60 kHz are very accurate, and readily received in New Zealand at night. The seconds ticks available from some GPS units are also very accurate.[15] There are several ways to use these references, ranging from simple to complex, which I will describe. The technique chosen largely depends on the quality of results required.

Defining Frequency Accuracy

Frequency reference performance is not just a matter of quoting how far off frequency it is (the offset). The offset also changes over time (ageing rate), and has short-term variations (perturbations). These variations are caused by temperature, varying load and supply voltage, and sideband noise effects. Offset is usually adjustable, if you have something better to calibrate against, and the calibration reference needs to be at least one decade better. The better the reference gets, the harder it is to measure and calibrate. Here are some reference examples:

Reference Type	Ageing Rate	Perturbations
Inexpensive TCXO[16]	Not quoted	0.5 in 10^6
Good OCXO[17]	1 in 10^9 per week	1 in 10^{10}
Rubidium Standard	1 in 10^{11} per month	1 in 10^{11}

[14] See NZART Call Book, page 8-1. See also www.irl.cri.nz/teams/msl/tiservi.htm
[15] The NMEA time message may be correct on average, but will wander in time. The second tick however is frequently held within 1us of UTC.
[16] Temperature Compensated Crystal Oscillator
[17] Oven Controlled Crystal Oscillator

Suitable References

Good quality TCXOs can be purchased inexpensively,[18] or recovered from defunct cellular phones. TCXOs are little silver boxes, the size of a postage stamp or smaller, typically on useful frequencies such as 10.000 MHz or 12.800 MHz. Some cellular phone units are on 9.60 or 14.85 MHz, which may not seem very useful, but using a micro controller, you can easily generate an accurate and useful reference. This month' s micro controller project is a clock based on one of these devices.

For higher performance, an oven controlled reference is necessary. These can be built, but suitable crystals are very difficult to find and expensive to buy; the oscillator has to be very good, and the oven control very accurate, all requiring considerable experimentation. The best solution is to hunt for surplus equipment that may contain a suitable reference. Look for old Omega receivers, pre-GPS satellite navigation receivers, telephone transmission equipment and communications test equipment. The older the better, to some extent, as the oscillator ageing rate will be lower. The two very high performance units I have came from Magnavox MX4102 Satellite Navigation receivers made in the mid 1980s.

If you need high accuracy, but cannot locate a suitable high performance reference, consider phase locking a lower performance voltage controlled oscillator (VCXO) to an external reference. The TV line frequency is a popular choice[19] as the circuitry is simple, but from experience lock tends to be unreliable. An excellent micro controller design[20] from the Philips Research Labs, which uses an LF reference, would adapt well to TV line frequency. GPS is quite complex to use due the the low reference frequency (1 Hz) but has proved effective using a micro controller approach.[21]

It is my experience that if you have a really good reference (such as a top quality OCXO), there is no point in attempting to phase lock it to another reference, such as the TV. The phase locked reference is likely to have much degraded short-term stability, and of course is quite dependent on having the external reference connected for it to be any use. Voltage controlled OCXOs or TCXOs depend on very low noise control voltages, something not too easy to achieve. I prefer to operate a manually adjusted local reference continuously (I never turn it off), and simply monitor its drift against a TV reference every few weeks. When the offset becomes bothersome, you can tweak the calibration so it is off in the other direction and drifts through over the next few months. It is not usually possible to set an oscillator to an offset much under 1 in 10^9, so there will always be an offset. Since the offset can be known precisely using phase comparison, frequency measurements and calibrations can be made with high reliability.

TV Phase Comparator

Calibrating a local reference which is a multiple of 15.625 kHz (250 kHz, 500 kHz, 1 MHz etc) is really simple, and costs virtually nothing. Simply include a divider from the reference which generates a 15.625 kHz signal (1 MHz divided by 64). A CMOS HC4020 device is ideal for the purpose. Calibration can be achieved by injecting this signal into the antenna of a TV set tuned to TV1 or TV2 and receiving a good signal. Look closely at the screen, and you should see one or two very narrow vertical lines superimposed on the picture. One should be white, the other black. Decide (according to picture content) which is easiest to see, put a sticky label on the screen to mark where the line is, start your stopwatch, and wait. From the time it takes the line to move off the screen to the left or right, round the back (so to speak!) and return to where it started from the other side, you can work out the offset of your reference. One minute is about 1 in 10^6, 10 minutes is about 1 in 10^7, and so on. If the line moves to the right, your oscillator is low in frequency. In under an hour you can easily adjust the oscillator as close as you need.

This technique is called phase comparison, and in doing this you are comparing the phase of your local reference with the TV line frequency at 15.625 kHz. This manual technique becomes tedious if your local reference is very good, or you wish to calibrate frequently. In that case, you need a more automated approach.

A simple phase comparator can be made using a flip-flop and a meter, as in Fig. 2. Using video from a TV receiver (a video recorder tuner is ideal), the sync separator, for example an LM1881, is used to recover the Line Sync. The leading edge of the 15.625 kHz Line Sync is used to set an RS flip-flop. The local reference is also divided down to 15.625 kHz, and the leading edge of the local reference used to reset the flip-flop. Thus the flip-flop is set for the period between one leading edge and the other, so is a measure of the phase difference. Since the inertia of a meter is too high to follow the 15.625 kHz signal, a simple 5V DC meter indicates the phase. As in the TV set approach, you can watch the meter and time the phase rotation, but a much better way is to use a chart recorder or logging DVM to record the phase drift over several hours. This technique easily allows measurement to one part in 10^8.

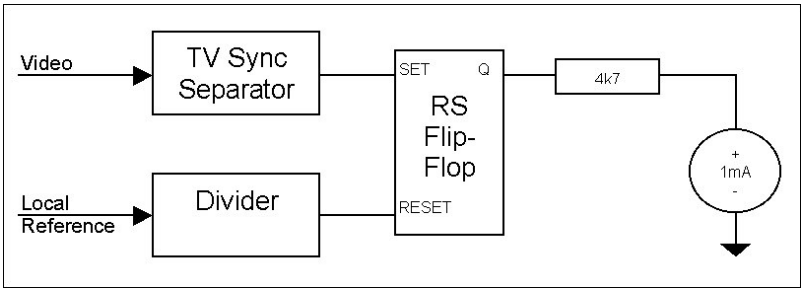

Fig. 2. Simple Phase Comparator

[18] See www.rakon.co.nz
[19] Electronics Australia, Nov 1986 page 34.
[20] Martin Ossmann, EDN, August 3, 1998 page 115. Also see http://www.e-insite.net/ednmag/archives/1998/080398/16di.htm
[21] Brooks Shera W5OJM, QST, July 1998. Also see http://www.rt66.com/~shera/index_fs.htm

Micro Phase Comparator

Although it is possible to compare the phase of a local reference to the TV Line or Frame frequency digitally, for example using digital counters and gates, using a micro controller is really much simpler. See Fig. 3. The "Divider" is a timer inside the micro, where high resolution is available. Use an AT90S2313 Atmel device, and drive Timer 1 with 1 MHz derived from the local 5 MHz precision oscillator. The timer is set to divide by 20,000, and thus resets automatically every 20 milliseconds. Use the TV Frame Sync to sample the value in the timer every 20 milliseconds. Using the Input Capture (ICP) feature of the micro, this is achieved without using any interrupts, so the precision stuff is not at risk of timing errors. See Fig. 3.

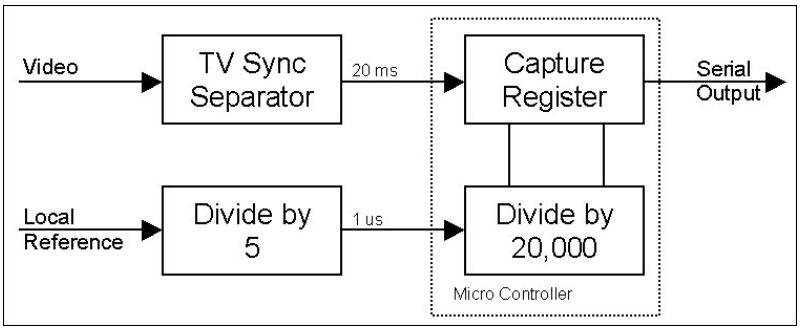

Fig. 3. Micro controller phase comparator

The Capture Register captures the number in the timer at the TV Frame sample point, and so represents the phase of the local reference, a count of 0 to 19,999, with a resolution of 1 microsecond. This register can be read and the data sent to a PC, which plots the phase and calculates the frequency offset. Using this technique, offsets as small as 1 part in 10^{10} can be detected, and running the PC software for an hour or so every few weeks will keep track of the ageing rate with ease. Fig. 4 shows part of a PC software display, which plots about 9 hours of reference phase (the upward sloping line). The plot is magnified because only the least significant eight bits of the 16 bit count are displayed. In 9 hours the phase has slipped 64 microseconds, so the frequency offset is -0.00083 parts in 10^6 (indicated on top line), or 8.3 parts in 10^{10}. An error of this order is equivalent to losing one second in 38 years!

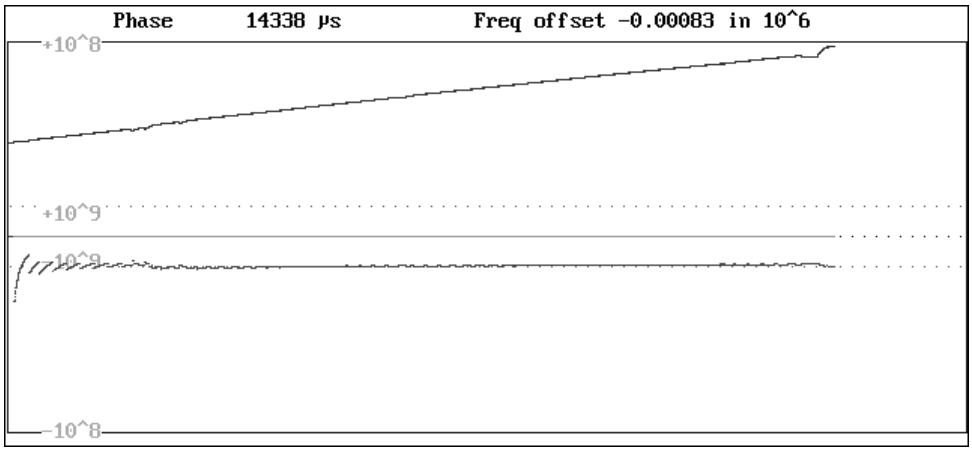

Fig. 4. PC phase comparison display

The phase plot has a couple of kinks in it, which I attribute to occasions where the TV network is not using its precision reference. The best time to run these tests is at night, and the worst during the weekend, when outside broadcasts are frequently used and the TV reference can be degraded.

The other line on the display in Fig. 4 is the frequency offset. It is essentially a horizontal line at about -1 part in 10^9, and it is to this line that the calibration marks refer. Apart from the interesting effect at the start, when the data has insufficient resolution to resolve the offset, the offset shows as a dead straight line, indicating that the short term perturbations are well within 1 part in 10^{10}. Over a period of weeks this line moves up slowly as the offset changes with crystal ageing. My best reference (illustrated here) moves about 1 part in 10^{10} per week.

Other Improvements

A number of additional features can be easily added to a micro controlled device such as that just described. An obvious thing to do is to add an LCD display, to indicate phase and frequency offset digitally, and since there is plenty of room for software and plenty of available processing time, to add a super-stable clock as well (remember - one second in 38 years). The clock could also provide time to the PC, via the serial output, along with the phase and frequency offset. See this month' s micro project!

A system such as has been described need not have TV video supplied all the time, and of course the PC software need only be operated when a reading is needed. Thus the system is reasonably economical of power use.

In order to compare phase over long periods, or include a clock, you must have a backup power source. Good crystal ovens use about 1W of heater power, and typically operate from 12V, so a 1 AH sealed lead-acid battery makes an appropriate backup supply. For high

precision, the oven supply needs to be closely regulated, and when AC power is lost, the reference phase will move slightly for the duration of the failure. Use a 13.8 V regulated charger for the battery, capable of supplying the complete load. The battery drives a low dropout 12V regulator for the oven supply, and a conventional 5V regulator for the logic supply. Provided the power outages are short, the oven supply stays in regulation as the battery voltages drops down from 13.8 to about 12.5V. Outages more than an two or three hours long run the risk of phase error, although the clock would not noticeably lose time.

Using a Frequency Reference

Probably the best use of a local precision reference is as a direct reference for a frequency counter. The frequencies generally used are 1 MHz or 10 MHz. The well known Dick Smith Frequency / Period Counter, and others based on the ICL7216 chip, can use either frequency. My frequency counter design[22] uses 4 MHz, but adapting it to 1 MHz, 5 Mhz or 10 MHz would be relatively straightforward.

A frequency reference can also be used to calibrate the oscillator in a frequency counter. Simply read the reference with the highest precision possible, for example reading the 4.43361875 MHz TV colour burst frequency with a 10 second gate time (resolution 1 part in 10^7), and adjust the counter to read the known reference frequency (including any known offset). To do the best possible job, run the counter continuously, and check it every week or so. Write down the calibration date and offset on a label attached to the counter, so you' ll know when to calibrate next time, and learn what the drift performance is like.

A good frequency reference makes an ideal source for a frequency synthesizer, such as a phase locked loop (PLL) type or a direct digital synthesis VFO. Unfortunately DDS type synthesizers generally require very high reference frequencies, although these can of course be phase locked to a lower frequency. I use a 12.8 MHz TCXO as the reference in my LF Exciter,[23] which can generate any frequency from zero to 400 kHz with 85 mHz (milliHertz) resolution.

Bob ZL2CA and I have experimented with precision frequency transmissions on 80m. I used an old 20W AM transmitter, crystal controlled on 3750 kHz. An output from the reference was devised at 1250 kHz (5 MHz divided by four), and since this is also 3750 kHz divided by three, by injecting a small amount of this signal into the transmitter crystal oscillator, it could be locked to the reference. It would pull in from several Hz away, and provided a transmission with extremely high precision.

[22] See Break-In September/October 2002, page 4.
[23] See www.qsl.net/zl1bpu/micro/EXCITER

