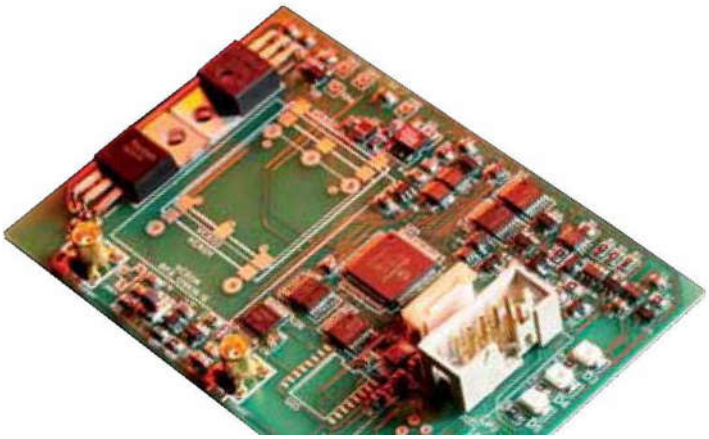
Quartzlock A6-1PPS

OEM 1PPS Timing Module

- Compact form factor
- License available
- Very fast lock to GPS



STOP PRESS Now available as a complete instrument

This is a PCB level product to control an OCXO or Rubidium oscillator from an external 1PPS.
The A6-1PPS uses a 3 state Kalman filter algorithm to measure & correct the frequency offset of the oscillator with respect to the 1PPS input. Time-tagged 1PPS to 200ps resolution & <1ns jitter.

Features

- 1PPS output
- 10MHz output
- Self-calibrating internal clock analogue interpolator
- 1PPS time tag resolution of 200ps
- < 1ns rms jitter

Benefits

- Holdover mode is initiated by failure of the 1PPS input
- Reduced 1PPS jitter
- Fast lock to high accuracy from raw GPS 1PPS

Applications

- Defence timing
- WiMax Base stations
- 3G Base stations (WCDMA, CDMA2000)
- LTE 4G
- Digital video Broadcast
- General Timing and synchronization

Tel **+44 (0)1444 232 967**

Email **sales@quartzlock.com**
www.quartzlock.com

A locking module for timing

This module is designed to lock a 10MHz stable oscillator, either OCXO or rubidium, to the 1PPS time mark signal generated from a GPS receiver. The module can be programmed for a wide range of controlled oscillator parameters, and GPS receivers. The controlled oscillator can be either on or off the board. A stable 1PPS time mark is generated from the controlled oscillator. This can be adjusted to any offset from the GPS 1PPS in 1ns steps.

The control algorithm used is designed to give optimum control results and the fastest possible acquisition from switch on.

Design strategy

This module is designed to lock a 10MHz stable oscillator, either an OCXO or a rubidium, to the 1PPS time mark signal generated from a GPS receiver.

There are a number of challenging problems involved in this, as the data rate by definition is only one measurement per second. In order to get sufficient frequency resolution to correct the oscillator, a very long averaging time would be required.

Because the 1PPS time mark is a fast rise time logic signal, the only measurement that is feasible is to time tag the incoming 1PPS edge relative to a local clock driven by the controlled oscillator. By calculating the rate of change of the arrival time over a suitable averaging period, the frequency offset of the controlled oscillator can be calculated. **An alternative strategy** would be to set the time of the first 1PPS arrival as the zero phase of a phase detector with a range of +/- 0.5s. This is equivalent to +/- Pi radians. A phase lock loop would then provide a very slow control of the oscillator.

In both systems the timing accuracy and resolution of the incoming 1PPS is important. Modern GPS receivers provide a 1PPS output jitter of between 1us RMS for a navigation receiver, to less than 7ns RMS for a special timing receiver operating in position hold mode. It is desirable that the timing resolution of the module should be better than this, as otherwise quantization noise would enter the averaging process and degrade the performance of the system. It would only be possible to compensate for this by increasing the averaging time. A suitable specification for time resolution is +/- 1ns.

To achieve this directly would need a 1GHz clock. A much more suitable method is an analog time interval expander. This device has been used in many designs of frequency counter starting with the Racal 1992. The principle is that an error pulse is generated which has a width equal to the time between the incoming edge to be timed, and the next clock pulse. For example, with a 100ns clock, the error pulse will have a width of between 0 and 100ns. This error pulse is then used to charge a capacitor or integrator. The capacitor or integrator is then discharges at a much slower rate, say 1/1000 of the rate. The resulting stretched pulse is then measured using the available clock pulses. The improvement in resolution equals the ratio of the discharge to charge rate. For the example above the resolution will be 100ps.

The next thing to consider is the choice of the control algorithm. This must provide an appropriate control bandwidth so the short term stability of the controlled oscillator (Allen variance) is optimised over a wide range. The ideal bandwidth will vary considerably between a low cost OCXO, and a rubidium.

One option is to use a simple phase lock loop. This would be a type 2 second order loop (ie with an integrator in the loop filter) with a zero to give suitable phase margins for optimum dynamic performance. **However** one problem with a phase lock loop is that it must reduce the initial phase error to zero by changing the frequency of the VCO. With the very long loop time constant necessary to remove the effect of the GPS time jitter, the eventual settling of the loop could take several days. It is also difficult to extract measures of performance from the loop, for example it is difficult to estimate the current frequency error of the VCO. **It was felt that a frequency control loop would settle quicker.** For a frequency standard we do not mind operating with a fixed phase offset, and there is no need to reduce this to zero.

Tel **+44 (0)1444 232 967**

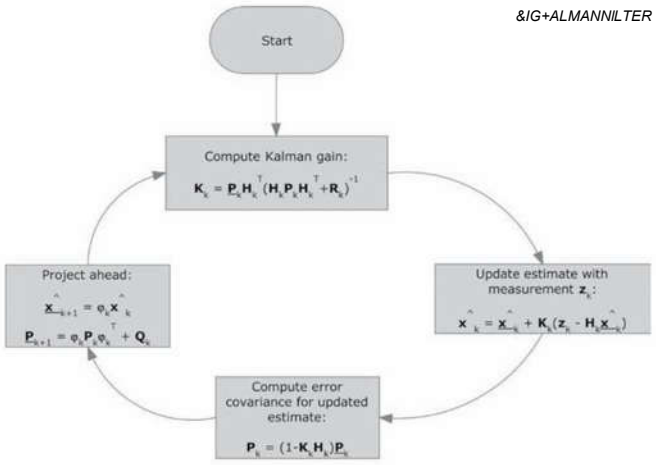
Email **sales@quartzlock.com**
www.quartzlock.com

One possible method of extracting frequency offset from phase data is a quadratic least squares fit on a block of data.
This is a standard method for extracting phase offset, frequency offset, and frequency drift from phase difference information. Having extracted the offset frequency, we can then make a correction to the controlled oscillator to remove the offset. If the control constant was known exactly, there would be no under or overshoot. The problem with this method is that we do not know how large to make the block of data that we analyse. The reliability of the fit is given by the correlation coefficient, and ideally this should be monitored on a continuous basis. What is required is a continuous least squares process. This is of course, a Kalman filter, and this was the eventual method selected for implementing the control algorithm.

The Kalman filter will be briefly described in general in a (hopefully) simple way, and then the specific implementation for our problem will be described in more detail.

A block diagram of a Kalman filter is given in figure 1. It is basically a recursive estimation, based on noisy measurements, of the future "state" of a system . The system is defined as a "state vector" and a "state transition matrix". The system in our case would be the controlled oscillator that we wish to predict, and the state vector would contain the phase offset, frequency offset, and frequency drift variables. The "state transition matrix" defines the differential

&IG+ALMANNILTER



The system is modelled as follows:

$x_{k+1} = \varphi_k x_k + w_k$

The measurement of the system occurs at discrete times:

$z_k = H_k x_k + v_k$

x_k = (n x 1) process state vector at time t_k
φ_k = (n x n) state transition matrix
w_k = (n x 1) vector sampled white gaussian noise
zk = (m x 1) vector measurement at time tk
H_k = (m x n) matrix giving the connection between the measurement and the state vector at time t_k
v_k = (m x 1) measurement error (sampled white gaussian noise)
$e_k = x_k - \hat{x}_k$ Estimation error
P_k = (n x n) Estimation error covariance matrix
Q_k = (n x n) system covariance matrix
R_k = (m x m) measurement noise covariance matrix

relationship that exists between the state variables over one time increment. The concept of a system driven by noise processes is important here. If our Rb had absolutely constant drift, its output phase would be known for all time once the initial drift , frequency offset and phase offset had been determined. Data gathered a year ago would have as much validity as data an hour old. If the Kalman filter is given this model of the Rb, the results are identical to the least squares fit of all the data. Of course the quadratic least squares fit assumes that the Rb can be modelled by three constants.

A more realistic physical model would allow the drift to vary. If this varied in a deterministic way, we should add a further term to the state vector to reflect this deterministic process. However if the variation was random, we can tell the Kalman filter that this is so. Note that the filter is only optimum for white gaussian noise processes. However in our case we can model the noise of the Rb oscillator more accurately by adding white gaussian noise to each term in the state vector. If we add some uncorrelated noise to each term in the state vector, we end up with white phase noise, white FM noise, and random walk FM noise due to the single and double integration in the model. This is shown in figure 2.

The measurements are also assumed to be contaminated with gaussian white noise. In our case we only have one measurement, that is phase offset. We do not know that the main contributer to measurement noise, the GPS receiver, is either white or gaussian. However this is a limitation of the simple Kalman filter that we intend to use. If we are sure of the characteristics of the measurement noise, we can include this knowledge by adding more terms to the state vector. We are then essentially including the known aspects of the measurement in the system model.

As well as the state vector, the Kalman filter maintains a matrix that gives the current variances (mean square error) of the quantities in the state vector. These give us current estimates of the likely errors in the state vector, in our case variances of phase offset, frequency offset, and frequency drift. These will be very useful for display to the user. They also have another use, which will be demonstrated later. In effect they control the "bandwidth" of the filter. As more data comes in, the variances decrease, and the filter gives more weight to the current estimate(which represents the complete history of the data), and less to the current measurement. The measurement variance, which we have to tell the filter, also affects the "bandwidth". If we tell the filter that the measurement is noisy, it reduces the bandwidth.

So far we have considered the Kalman filter as a device for analysing the incoming data in an optimum way. **However we need to control the Rb oscillator, and reduce the frequency offset to zero.** An elementary method would be to write periodic corrections to the Rb control DAC, and wait for the Kalman filter to track out the resulting discontinuity in the measurements.However there is a much better way. If we adjust the frequency offset term in the state vector at the same time that we correct the Rb , the filter will ignore the correction, and no extra settling time will be required. In effect we are defining the model of the Rb to have a frequency discontinuity at

a particular time, and provided the real Rb has that discontinuity, the Kalman filter will see no difference between the model, and what the measurements are telling it about the real system.

Using this technique, we can correct the Rb as often as we like. However if we are uncertain as to the exact value of the control constant, then the correction will undershoot or overshoot the model. Another trick that can be useful is if we know that there is a measurement discontinuity, but we do not know how large it is. An example would be if the GPS signal disappears for any reason. When satellites were reacquired, there could be a phase discontinuity between the GPS 1PPS and the locking module internal clock. Although we cannot tell the filter the amount or direction of the discontinuity, we can tell it that its current estimate of phase is completely unreliable. We do this by adding a large number to the appropriate term of the error covariance matrix. The filter then gives maximum weight to the measurements to reacquire the phase as quickly as possible, however as it thinks its frequency is still accurate, it does not give excessive weight to the rate of change of phase measurement, and the frequency covariance hardly rises.

The Kalman filter can predict ahead if measurement data fails. In this case both the state vector and the error covariance matrix will be updated. The previously estimated value of drift will update the frequency offset automatically. Frequency corrections can be made in the usual way.The error covariances will rise to reflect the lower confidence in the predictions as time passes. When measurements resume, the filter will automatically recover and the error covariances will start to fall. Thus the user is always aware of the reliability of the frequency output. If an unknown phase step is expected on resumption of measurements, then the phase variance should be augmented as previously described.

Technical details of design

The design is based around a PIC18F6723 microcontroller. This is a high end controller with 5 capture/compare modules and 4 timer/ counters. The time interval expander is tightly integrated with the processor internal peripherals to produce an economical design. The basic timing resolution is 400ns (one processor cycle at a 10MHz clock frequency). The time interval expander extends the resolution by 2000 times. In order to avoid the problems of expanding a pulse of zero width, one cycle of the 10MHz clock (100ns) is added to the time error pulse. This gives an unexpanded pulse width of 100ns to 500ns. After expansion, the pulse is 200us to 1ms. This is timed by the 400ns clock to give a basic +/-200ps resolution.

A time interval expander must be calibrated as otherwise a glitch will be produced when the time error pulse rolls over from 500ns to 100ns, and vice versa. This is caused by the expansion ratio not being exactly the expected 2000 times. The expansion ratio may drift with time and temperature.

As the incoming 1PPS only needs measuring once per second, the dead time is used to calibrate the time expander. The hardware generates exact pulses of 100ns and 500ns by gating from the 10MHz clock. These are expanded and measured. The calculated end points of the expanded pulse are used to correct the real measurement of the incoming 1PPS. This auto calibration operates continuously.

The control of the OCXO or other controlled oscillator uses a precision tuning voltage derived from DtoA convertors . Two 16 bit DACs are used, with the output of the fine tune DAC divided by 256 and added to the output of the coarse tune DAC. This gives effectively 24 bit resolution with an overlap between the coarse and fine tune DACs. A software normalisation process ensures that the fine tune DAC is used for tuning most of the time. Only when the controlled oscillator has drifted out of range of the fine tune DAC would the coarse tune DAC need adjusting, with the chance of a very small glitch in the tuning voltage. A precision, low noise, voltage reference is used to supply the DACs.

The microcontroller is provided with an RS232 interface. A simple set of control codes enable monitoring and set up of the controlled oscillator parameters to accomodate a wide range of controlled oscillators. A Windows front end program will use the control codes to enable the operation of the PLL to be monitored with real time graphs of performance measures.

Software design

In normal operation the auto calibration performs calibration cycles every 20ms. The approximate time of arrival of the next 1PPS input pulse is known, so the calibration cycles are paused while the 1PPS is measured. The raw measurement of the arrival time is corrected for the actual expansion ratio and is scaled to lie in the range -500.000000 to +499999999 ns relative to the internal clock.

The first valid 1PPS edge to arrive after reset is used to zero the internal clock. This makes the arrival time initially close to zero, and avoids problems with lack of precision in the floating point calculations which follow.

The corrected time tag is sent to the Kalman filter routine which runs once every second. The estimate of the controlled oscillator phase, fractional frequency offset, and drift (the state variables) is updated by the new measurement. Also updated is the error covariance matrix which provides an indication of the accuracy of the estimate of the state variables.

After update of the filter, the frequency correction for the controlled oscillator is calculated. This is done by scaling the Kalman frequency offset estimate by the known (programmed) tuning slope of the oscillator. The correction is then added to the frequency control register of the oscillator.

The tuning voltage is divided between the coarse and fine tune DACs as follows: When normalisation is performed, the fine tune DAC most significant 8 bits are set to mid point (80h). The least significant 8 bits of the fine tune DAC are set to the least significant

8 bits of the tuning word. The coarse tune DAC is then set to provide the final tuning voltage. During all subsequent tuning, only the fine tune DAC is used over its 16 bit range. If the range is exceeded, the normalisation procedure is repeated.

A state machine provides control of locking. After reset the last value of the frequency control register, which has been stored in EEPROM on a regular basis, is restored. This will retune the controlled oscillator to very nearly the correct frequency. The Kalman update is disabled and the software waits for the following all to occur (state 0):

a) Rubidium reference warm up input to go low or OCXO supply current to drop below a threshold showing the Rubidium/OCXO has warmed up

b) A 1PPS input capture has occured

The sofware then requests a reset of the internal clock (state 1). This will normally occur on the next 1PPS to be received.

Once a clock reset has occured, the Kalman filter tracking is started, however frequency corrections are not made to the controlled oscillator. (state 2) Each capture must be within 50us of the first capture, otherwise the reset state is reentered. After 100 successful captures, state3 is entered provided the performance monitor, MEANFREQERROR is below a threshold.

The performance monitor, MEANFREQERROR is calculated as follows:

The mean of the Kalman frequency offset estimate is calculated by means of a 5th order exponential filter. (In the pre lock state the mean may not be near zero, ie there may be a constant offset between the controlled oscillator and GPS time)

After each iteration of the Kalman filter, the current deviation is calculated by subtracting the current frequency offset estimate from the running mean. This value is squared, and divided by the predicted variance from the error covariance matrix that is maintained by the filter. This normalises the actual deviation that is seen by the predicted deviation from the filter. (The predicted deviation only depends upon the system and measurement noise parameters NOT on the actual behaviour of the system.)

The normalised deviation in then filtered in a 4th order exponential filter. During warmup the performance measure will be high, indicating that the controlled oscillator is still drifting fast, relative to its predicted steady state performance. When the controlled oscillator is stable, and the Kalman filter has settled, the performance measure will drop below a threshold. At this point frequency corrections will be started. (state 3)

In state 3 corrections are made to the controlled oscillator. The filter and oscillator will continue to settle, until the performance monitor falls below a second threshold. At this point the lock indicator is switched off. (state 4)

The following parameters set up the Kalman filter to match the controlled oscillator:

a) Oscillator noise parameters:

S1 variance of random walk FM noise

S2 variance of white FM noise

S3 variance of white phase noise

OC1 oscillator tuning constant in fractional frequency/volt

OC2 maximum oscillator tuning voltage in volts, assuming 0V minimum

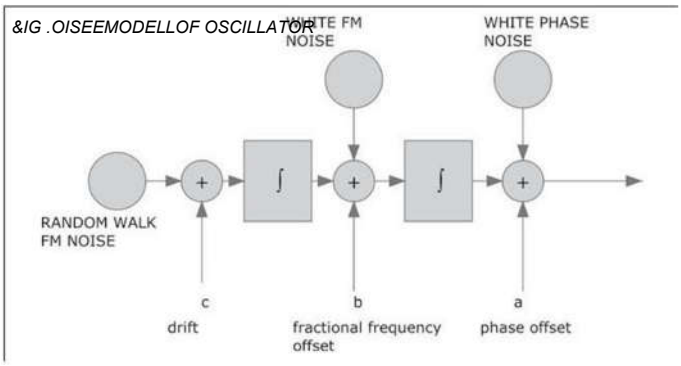
b) 1PPS noise root variance (a function of the GPS receiver used)

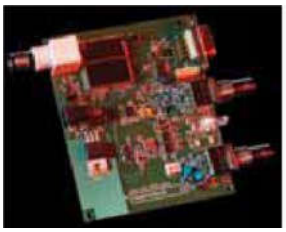
R measurement noise root variance in seconds

These parameters are programmed over the RS232 interface, and are stored in non volatile memory.

The oscillator noise parameters may be obtained from a measured Allen variance curve using a MathCad modelling program.

&IG .OISEEMODELLOF OSCILLATOR



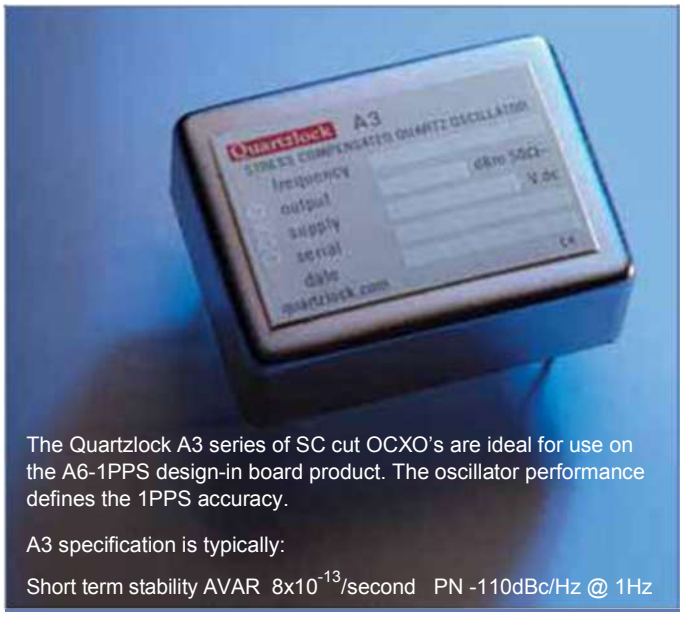


Specification

Frequency	10MHz
Input Level	100mv Pp to 5Vpp (Oscillator off board)
1PPS Input Impedance:	500 Ohms
Output Level	13+/-2 dBm (Oscillator on board)
1PPS Input Level	5V TTL/Cmos positive edge
Width	10us Minimum
Input Impedance	1000 Ohms
1PPS Output Level	5V TTL/Cmos positive edge
Width	10ms
Preset Offset Of 1PPS Output	-500000000 To +499999999 Ns in 1ns Steps
Timing Baseline	Selectable between fixed (minimum jitter) or kalman phase estimate (maximum accuracy)
External Tune Voltage	0 to span, where span is software adjustable between 5.8V and 10V
Lock Indicator	On Not Locked Off Locked, Low Phase Error Short Flash Every Second Locked, High Phase Error
Interface	See separate document
Interface Codes	See separate document
Performance	The control performance depends very much on the quality of the controlled oscillator and the source of the 1PPS synchronizing signal. For these reasons it is difficult to quote absolute performance figures.
Power Supply	14 to 30V (On board OCXO is used) An external OCXO or Rubidium may be used. 12 To 30V (No on-board OCXO)

Specification

The Following Cases Are Typical Controlled Oscillator: Rubidium	
1PPS Source	Passive Hydrogen Maser (Essentially no 1PPS Jitter) Result: Allen Variance 100s $1x10^{-12}$ 1000s $3x10^{-13}$ 10,000s $1x10^{-13}$
Controlled Oscillator: Rubidium	
1PPS Source	Quartzlock E8-Y/E8000 GPS Receiver in Position Hold Mode Result: Allen Variance 100s $1x10^{-12}$ 1000s $1x10^{-12}$ 10,000s $8x10^{-13}$
Current Consumption	150mA Typical (On-board OCXO)
Size	25 x 25 x 5mm (Without OCXO)



The Quartzlock A3 series of SC cut OCXO's are ideal for use on the A6-1PPS design-in board product. The oscillator performance defines the 1PPS accuracy.

A3 specification is typically:

Short term stability AVAR $8x10^{-13}$/second PN -110dBc/Hz @ 1Hz