

A Complete IF Software GPS Receiver: A Tutorial about the Details

Kent Krumvieda, DFC; Premal Madhani, CCAR; Chad Cloman, DFC; Eric Olson, DFC; Dr. John Thomas, DFC; Dr. Penina Axelrad, CCAR; Dr. Wolfgang Kober, DFC.

BIOGRAPHY

Mr. Krumvieda has an M.S. in Civil and Environmental Engineering, a B.S. in Chemical Engineering from the University of Colorado in Boulder, and over 15 years industrial experience. He has extensive experience involving programming for all of *Data Fusion Corporation's* sensor management and tracking efforts. Recently he was the chief software architect for the development of *Data Fusion Corporation's* GPS Near Far Resistant Receiver and Orbital Tracking Toolbox.

ABSTRACT

Research and development continues to extend the capabilities and to increase the robustness of Global Navigation Satellite Systems (GNSSs) receivers. Software receivers are quite valuable in evaluating potential improvements because of their flexibility. In addition, software receivers afford batch data processing options that are not available in hardware implementations. In developing such a receiver, we have found that many existing texts and papers on software receivers omit important implementation details. Here we discuss our complete Intermediate Frequency (IF) software GPS receiver and reveal details of the lessons learned during its construction. Specific topics addressed in this paper include:

- linear and non-linear adaptive analog-to-digital conversion;
- acquisition using coherent and non-coherent integration;
- search techniques (e.g., Vernier, M of N, and Tong) performed in both the time domain and the frequency domain;
- limitations of the DFT approach including leakage;
- zero padding and additional DFT enhancements;
- tracking utilizing FLLs, PLLs and DLLs;
- bit synchronization and navigation data recovery;
- frame synchronization and parity decoding;

- calculation of the navigation solution; and,
- Summary of our MATLAB and C toolkits

This paper will present examples and results we have generated using sampled GPS signals collected with a high speed data recorder

INTRODUCTION

The IF Software GPS Receiver has been developed as part of a USAF Contract entitled "Novel Wide-Band Receiver Solution for 'Near-Far' Communication Problem". The IF Software GPS Receiver, hereafter referred to as Baseline Receiver, served both as a Baseline to compare the results of the Novel Near-Far Resistant (NFR) receiver, and to provide an architecture in which to host the NFR components.

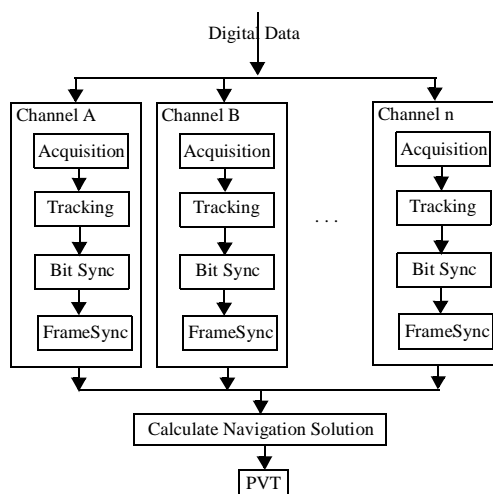


FIGURE 1. Baseline Receiver Architecture

The Baseline Receiver Architecture is designed to handle 'n' channels in parallel. Once a minimum of four channels have finished Frame Synchronization and have read some data, the navigation solution may be computed. The Baseline receiver requires that the input data be at a user defined Intermediate Frequency (IF) and converted to digital representation at some user defined sampling rate.

In the absence of any data supplied by the USAF initially we implemented a World Simulator and Signal Generator to supply our receiver with data. The World Simulator was of sufficient fidelity to create realistic signal, i.e., spherical earth, circular satellite orbits, multiple coordinate systems, Doppler, thermal noise, etc. After the signal had been realistically simulated the receiver had to convert the signal from RF to IF and discretely sample the signal. To realistically model this we implemented a "Real-World" Linear Analog to Digital Converter.

ANALOG TO DIGITAL CONVERSION (ADC)

Data Fusion Corporation implemented and we shall discuss the Idealized Linear ADC, Linear Real-World ADC, as well as the Non-Linear ADC which has nice properties in high CW interference.

IDEALIZED LINEAR ADC

The theoretical idealized transfer function for a ADC is a straight line with an infinite number of steps. A practical idealized transfer function is a uniform staircase with a finite number of steps.

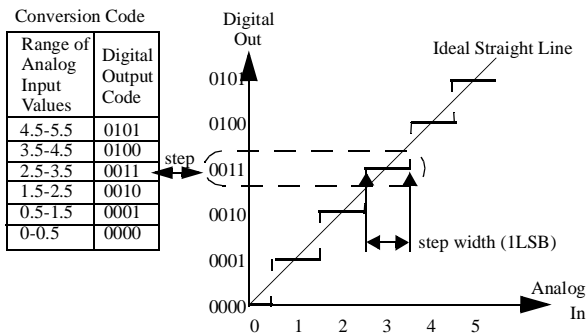


FIGURE 2. Ideal Transfer Function (ADC)

Where 1 LSB is (one least significant bit), the inherent quantization error is $\pm 1/2$ LSB. To ensure that our receiver was robust to non-ideal situations a “Real-World” Linear ADC was implemented.

Linear Real-World ADC

The simulation model for the real world ADC implemented expands on the ideal quantizer by incorporating: Number of Bits, Max Value, Minimum Value, Offset Error, Gain Error, Differential Nonlinearity (DNL) Error, Integral Nonlinearity (INL) Error, Aperture Errors, and Dynamic Errors.

ADCs provide an intrinsic sampling function of the dynamic analog input signal. Dynamic errors which are determined from the sample and hold (S/H) specification are usually defined in terms of two parameters: aperture error and clock jitter.

Aperture error refers to the variation in the length (aperture) of the sampling period. Direct measurement of this error source is very difficult and a specification is usually arrived at through a process of estimation. As a guideline to how much aperture error is tolerable,

$$dt_{MAX} = \frac{1}{\pi \cdot f \cdot 2^{N+1}}$$

This equation is derived from a full-scale sine wave which represents a worst-case condition.

Clock jitter refers to the time between an input clock edge that defines the sampling period and the actual switching instant internal to the ADC.

The locked histogram test can be used to estimate aperture error in flash ADC. To perform this test the same frequency is used for both the sampling clock and input sine wave. A full-scale or greater analog input is synchronized to the sampling clock so that the same point on the input is repetitively sampled. The effects of aperture error will become more evident as the signal slew rate increases. In order to estimate the ADC aperture error, the standard deviation of the histogram is typically used. This is equivalent to the RMS aperture error.

$$Aperture_{error} = Ta = \frac{\sigma \cdot V_{LSB}}{Slewrate}$$

$$\text{where } Slewrate = V_{AMP} \cdot 2\pi \cdot f_{in}$$

Static errors are those errors that affect the accuracy of the converter when it is converting static (D.C.) signals, and can be completely described by just four terms. These terms are offset error, gain error, integral nonlinearity (INL) and differential nonlinearity (DNL). Each of these terms are expressed in LSB units.

Offset Error is the difference between the nominal and actual offset points. The offset point is the mid-step value when the digital output is zero. This error affects all codes by the same amount and can usually be compensated for by a trimming process.

Gain Error is the difference between the nominal and actual gain points on the transfer function after the offset error has been corrected to zero. The gain point is the mid-step value when the digital output is full scale. This error represents a difference in the slope of the actual and ideal transfer functions and as such corresponds to the same percentage error in each step. This error can also usually be adjusted to zero by trimming.

Differential Nonlinearity (DNL) Error is the difference between an actual step width and the ideal value of 1 LSB. If the DNL exceeds 1 LSB, there is a possibility that the converter can become nonmonotonic. This means that the magnitude of the output gets small for an increase in the magnitude of the input. In an ADC there is also a possibility that there can be missing codes, i.e., one or more of the possible 2^N binary codes are never output. To measure differential linearity, a delayed sweep and timebase multiplier of the oscilloscope can be used to scan individual segments of the sawtooth error waveform. By initially calibrating the x-axis to the ideal width of 1 LSB, the variation in width of the sawtooth segments can then be measured to find the DNL. Other tests that can be used are the Servo-Loop Code Transition Measurement, Digital Analysis, [8].

Integral Nonlinearity (INL) Error is the deviation of the values on the actual transfer function from a straight line. This straight line can be either a best straight line which is drawn so as to minimize these deviations or it can be a line drawn between the endpoints of the transfer function once the gain and offset errors have been nullified. The Servo-Loop Code Transition Measurement and Digital Analysis, can be used to determine DNL and INL, [8].

The Effective Number of Bits (ENOB), expressed in decibels, takes into account several dynamic specifications and is determined by the resolution of the ADC.

$$ENOB = \frac{SINAD - 1.76}{6.02}$$

where SINAD is the ratio of signal to noise plus distortion. This includes SNR along with the total harmonic distortion (THD) of the ADC. THD is the ratio of the sum of the harmonic distortion amplitudes to the original input amplitude. THD is caused by the ADC nonlinearities.

A curve fitting algorithm is often used to measure the ENOB or resolution of an ADC when digitizing a sine-wave input. The best-fit sine-wave calculation must determine all four parameter that describe a sine wave: amplitude, phase, frequency, and offset. The fitted sine wave is used as a reference from which to calculate the error in the actual data.

The error introduced by representing a continuous signal as a discrete function of a finite number of states is called quantization noise, and can be represented by \bar{N}^2 . The total mean square error[11], \bar{N}^2 , over the whole conversion area of the ADC is the sum of each quantization levels mean square multiplied by its associated probability. The error at the j th step is $E_j = (V_j - V_j)$. Where V_j is the Voltage at the midpoint of the j th step and V_j is the actual Voltage. The mean square error over the entire step is

$$\bar{E}_j^2 = \frac{1}{q_1} \cdot \int_{-q/2}^{q/2} E_j^2 dE = \frac{q^2}{12} \quad (1)$$

Assuming equal steps, the total error, or Mean square quantization noise is $\bar{N}^2 = \frac{q^2}{12}$.

Signal to Noise Ratio (SNR) refers to the noise level produced by the ADC down from full scale. SNR can be expressed with a signal present (one tone usually) or with no signal present. If the SNR changes as a function of input frequency then there is usually a jitter problem reflected in the aperture uncertainty specification, see above. If the SNR remains flat as the input is exercised over the entire Nyquist range then there usually isn't a jitter problem.

Considering a sine wave input $F(f)$ of amplitude A so that $F(t) = A \sin \omega t$ which has a mean square value of $F^2(t)$,

$$\text{where } F^2(t) = \frac{1}{2\pi} \int_0^{2\pi} A^2 \sin^2(\omega t) dt$$

which is the signal power. Therefore the SNR is given by

$$SNR(dB) = 10 \text{Log} \left[\frac{A^2}{2} / \frac{q^2}{12} \right]$$

Spur Free Dynamic Range (SFDR) is the dynamic range (from full scale of the ADC) down to the level of the highest tones produced by the ADC whether they are harmonics, intermods or unrelated spurious tones. Essentially, this is an indication of how far it is possible to go below the full-scale input signal without hitting noise or distortion. It can be defined as the ratio of the full-scale input signal to the highest harmonic or spurious noise component amplitude. This is usually a different number than SNR, see above. SFDR is always specified with signals present. In some cases the SFDR is much better than the SNR. To see the tones you must either zoom on them with a decimating filter and look over the whole input frequency range of the ADC or use a bigger and bigger FFT to push the noise down so that the harmonics, intermods and spurious tones can be seen. Just put in a filtered (to clean up the generator harmonics) single tone from a generator. Take a big FFT and observe the highest harmonic.

Non-Linear ADC

The design of an ADC can have significant impact on the performance of a GPS receiver in the presence of CW interference. Amoroso [10] proposed a scheme of threshold adaptation and post quantization weighting that performs remarkably well making decisions on a relatively small percentage of the demodulated chips. For most systems it is desirable to quantize as coarsely as performance will permit. For a one bit per chip ADC in the presence of Gaussian noise the SNR is degraded 1.96 dB [10] and the degradation due to CW interference is much greater.

Amoroso's 2 bit ADC technique reduces the degradation in Gaussian interference to less than 0.6 dB, and the performance in CW interference is excellent. This is done by exploiting the non-Gaussian nature of the CW interference in setting the thresholds for the non-linear transfer function. Statistical control with feedback is used to set the thresholds to ensure thresholds are exceeded with the required statistical frequency. For example, the sign threshold requires 50% above and 50% below the threshold, and about 15% of the chips fall above the upper magnitude threshold and 15% fall below the lower magnitude threshold. Actually, the magnitude thresholds are dependant on the SNR, i.e., the higher the CW interference the higher the magnitude threshold must be to achieve perfect

chip decisions. Care must be taken to ensure that a substantial number of chips survive the thresholding operation to allow for correlation. The transfer function appears in below:

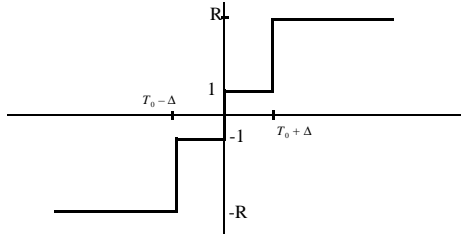


FIGURE 3. Transfer Function on Non-Linear ADC

Where R is the weighted digital output, and T_0 is the sign threshold. Samples falling below the magnitude thresholds are given unity weight in the correlator while samples exceeding the magnitude thresholds are given weight R . It is the setting of R much greater than unity that enables the rejection of CW interference. Assigning the samples which fall below the magnitude threshold ± 1 ensure good performance in Gaussian noise, where discarding them (assigning them to zero) leads to poor performance in Gaussian noise.

No one magnitude threshold setting is optimal for all system error rates. The adaptive 2 bit ADC must be evaluated for each system under consideration, taking into account processing gain, threshold bit error rate, and relative importance of the performance in Gaussian noise versus CW interference. The Software IF GPS Baseline receiver allows the user to specify R and Δ . The default settings of 3 and 1σ respectively, were found to give good results in the most cases.

ACQUISITION

This section describes the GPS signal acquisition process for both conventional and software receiver architecture. Acquisition is a coarse synchronization process giving estimates of the PRN code offset and the carrier Doppler. This information is then used to initialize the tracking loops.

GPS signal acquisition is a two dimensional search process in which a replica code and carrier are aligned with the received signal. The correct alignment is identified by measurement of the output power of the correlators. In other words, when both the code and carrier Doppler match the incident signal, the signal is despread and a carrier signal is recovered. The result of the two dimensional search is an estimate of the code offset to within one half-chip and the Doppler to within half the Doppler search bin size (several hundred Hz). In a conventional receiver, acquisition is performed using a carrier signal and C/A code replica. The short length of the C/A code (1 ms)

allows for a full search of the code sequence. Once the C/A signal is tracked, interpretation of the HOW within the navigation message permits a straightforward transition to P(Y) Code tracking for PPS receivers. The key aspects of acquisition are the definition of the search space, the criteria set for signal detection, and the performance of the algorithms under various signal levels.

The Search Space

The search space must cover the full range of uncertainty in the code and Doppler offset. Because the C/A code is fairly short, typically the range space will include all possible code offset values. The resolution of the code search is usually $\frac{1}{2}$ chip increments but often the sampling freq is used to specify the resolution. The range of the Doppler dimension is governed by the vehicle and GPS satellite dynamics and the stability of the receiver oscillator. For a terrestrial user system this is typically in the range of 5 to 10 kHz. The frequency resolution is determined by the coherent integration time (or dwell time). The relation is

$$D = \frac{2}{3T} \quad [6]$$

where D is a frequency bin width in Hz and T

is the predetection integration time in seconds. The dwell time is based on the C/N0 ratio of the GPS transmitters to be acquired.

The selection of a path through the search space is a function of the vehicle dynamics and requirements for acquisition speed and reliability. Typically the Doppler is set to the expected value and the code is searched over all possible delays. If this fails, the search is continued in the next Doppler bin, with the sequence of bins alternating above and below the starting Doppler. Additional considerations, such as selecting a forward search through the code to avoid false acquisition of a multipath signal are also included in designing the search process. FIGURE 4., depicts the acquisition uncertainty region.

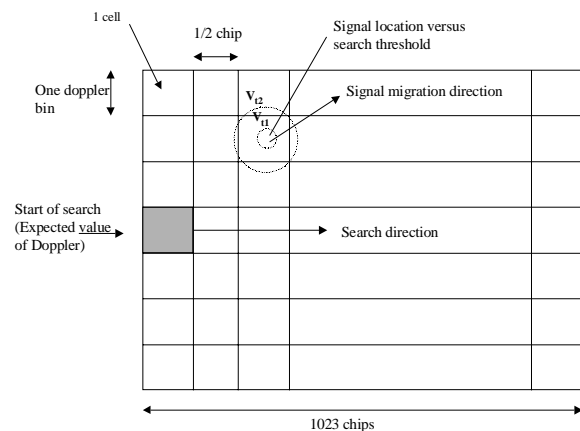


FIGURE 4. Acquisition uncertainty region [6]

Detection Criteria

The acquisition is based on a measurement of the correlator output. The correlators provide a measure of the total I and Q signal voltages over the coherent integration time. The total amplitude is then given by the envelope $\sqrt{I^2 + Q^2}$. When the replica and reference signals are aligned, the amplitude of the recovered signal is at a maximum. During the dwell time in each cell, the In-phase and Quadrature components I and Q respectively are formed by stripping off the reference code and the carrier from the received signal. The envelope which is the measure of the amplitude of the incoming signal is computed and compared with a threshold. In the presence of noise, one must set a threshold based upon an acceptable probability that a noisy measurement that does not contain the signal will appear to match the replica. Specifically the amplitude threshold is set by [6]: $V_t = \sigma_n \sqrt{-2 \ln P_{fa}}$ where P_{fa} is the single trial probability of false alarm, σ_n is the 1-sigma noise amplitude. σ_n is frequently obtained by calculating using a reference PRN which is known to be absent.

For the chosen threshold V_t , any cell envelope that is at or above the threshold is detected as the presence of the signal. Any cell envelope that is below the threshold is detected as noise. The detection of the signal is a statistical process because each cell either contains noise with the signal present or noise with the signal absent.

Each case has its own probability density function as FIGURE 5.

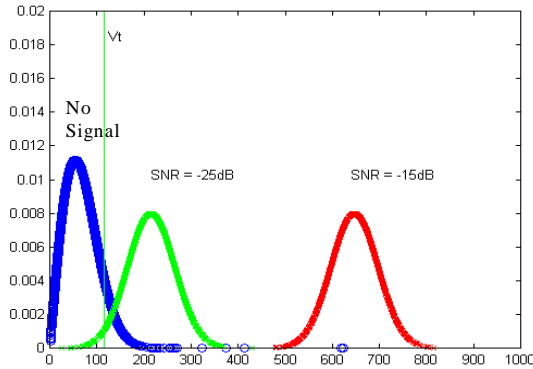


FIGURE 5. Probability density function for various cases.

In the absence of the signal the envelope has a Rayleigh distribution and in the presence of the signal the envelope has a Ricean distribution; I and Q signals are normally distributed if the noise is gaussian.

In FIGURE 5 one can see that if the SNR is high, it is easy to set a threshold that provides both low probability of false alarm, and also has low risk of a missed detection. As the SNR is reduced it is no longer possible because of the significant overlap of the signal distributions. If the

threshold is set very high to avoid false detections, then there is a high probability that weak signals will not be detected. In GPS this is the typical situation so that a single trial detection is not effective.

FIGURE 6. shows the acquisition system block diagram consisting of noncoherent correlator, PRN code generator and synchronization control scheme.

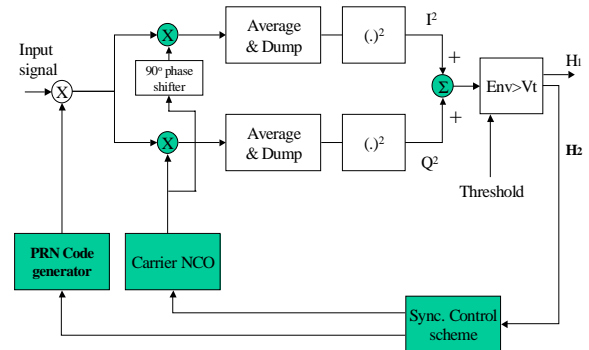


FIGURE 6. Acquisition scheme

Data Fusion Corporation's GPS software receiver performs a two-dimensional C/A-code search to achieve the initial acquisition of the GPS signal. The search involves replicating all 1,023 C/A-code phase states in the range dimension and numerous Doppler states. If the range and Doppler uncertainty are known, then the search pattern should cover the 3-sigma values of the uncertainty. *Data Fusion Corporation's* GPS software receiver assumes Doppler and range uncertainty is unknown. Currently the code phase is searched in fractional chip increments corresponding to the sampling frequency and the Doppler states correspond to the receiver's dynamics. For example a stationary receiver with, $f_{IF}=1.25$ MHz, $f_s=5$ MHz, and $T=1$ ms there would be 5000 range bins corresponding to $\sim 1/5$ -chip increments 21 Doppler bins corresponding to 666.67 Hz increments, where f_{IF} is the intermediate frequency, f_s is the sampling frequency and T is the coherent dwell or integration time. The Doppler bin size is defined as $2/(3T)$.

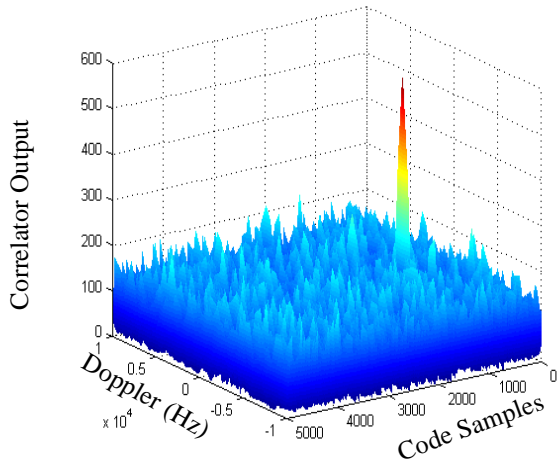


FIGURE 7. Correlation Surface

Thus when the uncertainty is large or unknown the search pattern is correspondingly large and the search time increases. Various detectors have been developed to perform this search.

Search methods used for GPS signal acquisition

The false alarm rate and the probability of detection from single dwell time detectors (single trial decisions) are usually unsatisfactory for GPS applications. So single dwell time schemes are seldom used.

Various search detectors exist. Kaplan [6] identifies the Variable Dwell Time and the Fixed Dwell Time detectors as two types which are commonly used in GPS receiver designs. A variable dwell detector makes a Boolean decision that a signal is present based upon a pre-defined criteria in an unspecified or variable amount of time. A fixed dwell time detector makes a Boolean decision that a signal is present in a based upon a pre-defined criteria in a fixed amount of time. An example of a sequential variable dwell time search detector is called the Tong detector.

Tong Search Detector

Data Fusion Corporation implemented a modified version of the Tong detector described in Kaplan. "The Tong detector has a reasonable computational burden and is excellent for detecting signals with an expected C/N_0 of 25 dB-Hz or higher." [6] To increase effectiveness under heavy jamming and/or interference conditions a hybrid version had to be developed. The hybrid version performs an exhaustive search over all code offsets and Doppler bins with the maximum normalized result indicating the detected signal. Figure 8 presents the Tong search detector algorithm.

To utilize the Tong algorithm a few variables must be initialized. These variables are K and A . The counter vari-

able K and the confirmation threshold A must be initialized based on the operational environment. The operational environment will determine the relative importance of acquisition speed versus probability of detection and false alarm. Maximum acquisition speed will be achieved by setting $K=1$. To achieve a higher probability of detection and a lower probability of false alarm, at the expense of speed, K may be set to 2 or higher. When K reaches A the signal is declared present. A typical range for A is [8-12] which corresponds to high to low C/N_0 respectively.

Assuming the search space has been defined as described in the previous section execution of the Tong Algorithm is relatively simple. A correlation envelope $\sqrt{I^2 + Q^2}$ is formed for a given search-grid-cell every T seconds. If the correlation envelope exceeds the threshold V_t , then the counter K is incremented by one, else K is decremented by one. If K equals A then the signal is declared present and the search is over. If K equals zero then the signal is determined not to be present and the entire process starts over in the next search-grid-cell. Finally, to avoid the possibility that the Tong will get caught in a situation where it satisfies neither condition, *Data Fusion Corporation* modified the algorithm such that only a finite number of correlation envelopes will be calculated for a particular search-grid-cell. When this maximum dwell threshold is exceeded the algorithm give up on that cell, determines the signal is not present and the entire process starts over in the next search-grid-cell.

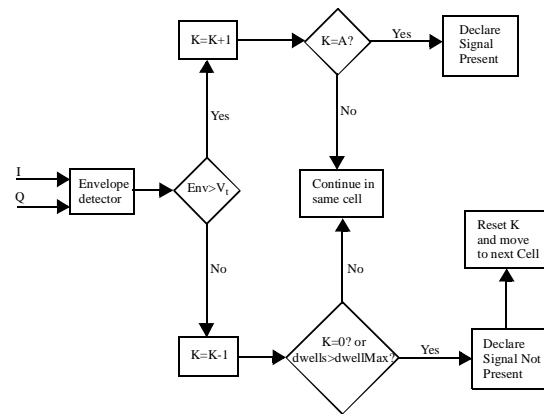


FIGURE 8. Tong Search detector algorithm

One important performance criteria of GPS receivers is the mean time of acquisition. A variable dwell procedure is adopted for its most significant improvement in reducing the expected acquisition time over a single dwell procedure.

For the Tong-search detector the mean number of correlation envelopes that must be calculated ($K_{init}=1$) to dismiss a cell containing noise is [6]:

$$N = \frac{1}{1 - 2P_{fa}} \quad (2)$$

Utilizing the fact that most of the time is spent searching cells that contain noise only, the overall speed of acquisition can be estimated [6]:

$$R = \frac{d}{NT} = \frac{d(1 - 2P_{fa})}{T} \text{ chips/sec.} \quad (3)$$

where $d = \text{chips/cell}$ and $T = \text{predetection integration time in seconds}$.

For example for a P_{fa} of 0.1%, $T=1$ msec, and 1/5th chip/cell, the code search rate is 160chips/sec. Note that the search speed increases as probability of false alarm decreases. For this example the minimum time to acquire the signal would be when the search starts in the cell where the signal is present. In this the time to acquire would be $1/160 = 6.3$ ms. The maximum time to acquire the signal would be when the signal is present in the last cell in the search pattern. If the frequency uncertainty is ± 5 KHz and the frequency bin-width is 500Hz, then the maximum time to acquire the signal is $\frac{21 \times 1023}{160} = 134.27$ secs.

The mean time to acquire in seconds for various SNR and integration times is shown in the table below. $P_{fa}=0.1$,

TABLE 1. Tong Mean Time to Acquire

SNR dB	T=1 ms	T=3 ms	T=5 ms
-20	6.32	18.28	28
-25	6.59	19.12	31
-30	*	19.72	32.6
-35	*	19.93	34

$K=2$, $A=10$, $f_s=5$ MHz, $IF=1.25$ MHz, the correct offset was at the 2500th sample, * indicates No detection. The threshold was kept constant over each column and the search was done in the bin in which the signal was present.

As mentioned earlier to increase effectiveness in low C/N_0 environments a hybrid version was implemented. This hybrid version included the concept of maxDwell mentioned earlier as well as an exhaustive search over the entire search grid. To perform this search in a reasonable amount of time a Discrete Fourier Transform version was developed.

DFT Acquisition

The advantage of the DFT version is that it calculates the correlation for a entire range dimension (selected Doppler) in a single step. The disadvantage is that when Doppler is non-zero the reference signal when convolved produces some errors.

In this technique a DFT (discrete fourier transform) is applied to the incoming GPS signal and multiplied by the conjugate DFT of the reference signal. Taking the inverse DFT of the product gives the correlation result in the time domain for all the 1023 code phase offsets. This method is computationally more efficient and faster than the conventional time domain technique. Hence this method is used in the software GPS receiver.

The DFT of a sampled signal $x(n)$ is given as:

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp(-j2\pi n \frac{k}{N}) \quad (4)$$

Multiplying the DFTs of two signals $x(n)$ and $y(n)$ and taking the inverse DFT of the product corresponds to convolution in the time domain. However since we correlate the incoming GPS signal with the reference signal in the time domain, this corresponds to multiplying the conjugate DFT of $x(n)$ with DFT of $y(n)$, and then taking inverse DFT of the product [13].

For the N point DFT of an N-point sequence, N^2 additions and multiplications are required which is the same as the time domain. However if the sequence length is limited to a power of 2, then using FFT (Fast Fourier transform) $N \log N$ additions and $\frac{N}{2} \log N$ multiplications would be required. This results in a reduction in computational time, at the cost of accuracy.

DFT Acquisition versus Time Domain Acquisition

FIGURE 9. is the difference between the time (evolving) acquisition envelope and the DFT envelope over the entire correlation length. FIGURE 10. is the difference between the time (circularly convolved) acquisition envelope and the *Data Fusion Corporation* envelope over the entire correlation length.

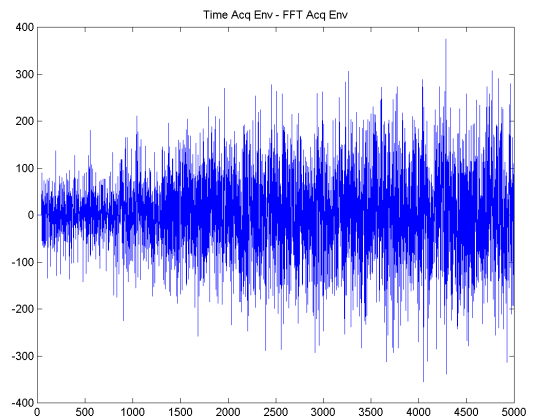


FIGURE 9. Difference of Time & Frequency Acq Envelope

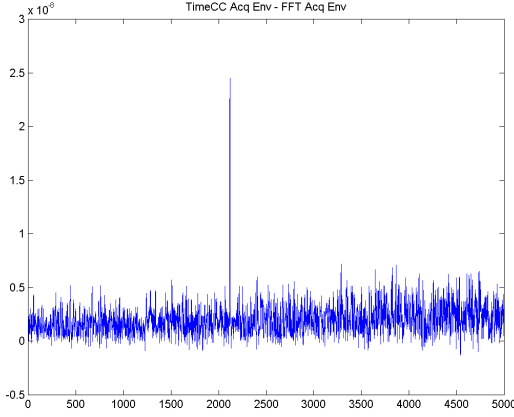


FIGURE 10. Difference of Time & Frequency Acq Envelope

FIGURE 10. indicates that the majority of the error is due to non-zero Doppler on the reference signal. This is due to the way the reference signal is created. The reference signal has constant Doppler

$$x(i) = \sin(\omega t + \pi C) \quad (5)$$

where $\omega = 2\pi(f_{IF} + f_d)$, C is the C/A chip as a function of time incorporating chip dilation and contraction, and i is from 1 to the length of y . For the DFT acquisition circularly convolved where in the Time domain acquisition x evolves in time and therefore has no discontinuities in phase. A relatively simple fix which mitigates most of the errors involves stripping off the $(f_{IF} + f_d)$ before performing the DFT.

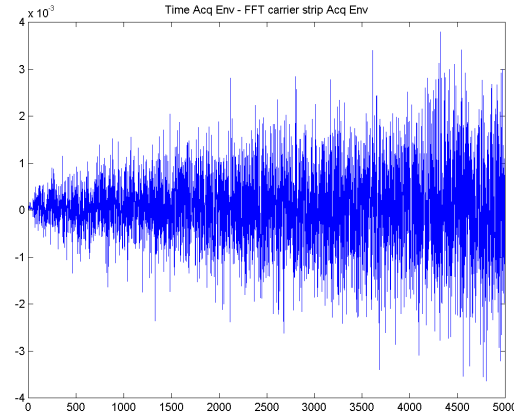


FIGURE 11. Difference of Time & Frequency Acq Envelope

Examining FIGURE 11. the errors, as defined as the difference from the Time domain acquisition, have been reduced by 100,000 times. Results from different coherent

integration lengths are presented in Table 2 on page 8 and Table 3 on page 8.

TABLE 2. Normalized Means and Standard Deviations

CI	mean	std	mean	std
	T-DFT	T-DFT	T-DFTcs	T-DFTcs
1	-1.1896e-4	2.8492e-3	-1.1896e-4	2.8492e-3
2	9.1226e-6	2.6206e-3	9.1226e-6	2.6206e-3
3	-1.2191e-2	2.5486e-2	-4.3753e-5	1.6592e-3
4	-5.2558e-3	1.7034e-2	-1.8417e-5	1.3777e-3
5	1.1784e-3	1.5624e-2	-2.5415e-5	1.5112e-3
6	6.7638e-3	1.9594e-2	-1.4562e-4	1.9978e-3
7	-1.6811e-3	1.0512e-2	2.5115e-5	1.0752e-3
8	-1.8648e-3	9.8580e-3	-1.1505e-5	1.0840e-3
9	-2.3152e-3	1.1959e-2	-6.6621e-5	1.3151e-3
10	5.4746e-5	8.4567e-4	5.4746e-5	8.4567e-4

TABLE 3. Amplitude Increase due to Coherent Integration

CI	Theory	Time	DFT	TimeC	DFTcs
1	1	1	1	1	1
2	1.4142	1.1767	1.1845	1.1845	1.1845
3	1.7321	2.1147	1.4798	1.4798	2.1155
4	2	2.4013	2.0638	2.0638	2.4013
5	2.2361	2.1279	2.2785	2.2785	2.1238
6	2.4495	1.5051	1.9285	1.9285	1.4993
7	2.6458	2.8843	2.7043	2.7043	2.892
8	2.8284	2.7554	2.6433	2.6433	2.7649
9	3	2.3827	2.2155	2.2155	2.3796
10	3.1623	3.0214	3.0329	3.0329	3.0329

where the subscript cs refers to performing carrier stripping before performing the DFT.

M of N

M of N is a fixed dwell time search detector. The M of N search detector takes N envelopes and compares them to the threshold, if M or more of them exceed the threshold, then the signal is declared present. The $P_{fa}=0.1$, $M=7$,

TABLE 4. M of N Mean Time to Acquire

SNR dB	T=1 ms	T=3 ms	T=5 ms
-20	24.98	75	125
-25	*	74.87	124.95
-30	*	75.06	125.1
-35	*	*	125.05

$N=10$, $f_s=5$ MHz, $IF=1.25$ MHz, the correct offset was at the 2500th sample, * indicates No detection. The threshold was kept constant over each column and the search was done in the bin in which the signal was present.

Comparing Table 1 on page 7 with Table 4 on page 8 you can see that Tong generally acquires faster than M of N .

Vernier

Vernier search detector as it has been implemented by **Data Fusion Corporation** is simply a time domain Tong detector with a limited 2D search space. The search space is typically +/- 2 chips and +/- 2 DFT Doppler bins at a finer resolution. Vernier is used for polishing the result of the DFT Tong to achieve better estimates of Doppler and code offset. Vernier can resolve Doppler down to $83.3/T$ Hz and code offset down to about a tenth of a chip. These polished results are then passed to the Tracking loops.

TRACKING

The acquisition loop gives a coarse estimate of the carrier Doppler and PRN code offset of the incoming signal. Control is then handed over to the tracking loops, the function of which are to track the variations in the carrier Doppler and code offset due to line of sight dynamics between the satellite and the receiver.

The unknown parameters of interest in the incoming GPS signal are the carrier Doppler and carrier phase and the PRN code Doppler and code phase. These parameters are functions of time because of the relative motion between the satellite and the user receiver. Hence it is important to track these parameters and get a very good estimate of the same. Another important function of the tracking loops is to demodulate the Navigation data from the incoming GPS signal. The PRN code phase can be used to determine the pseudorange whereas the carrier phase is used to accurately determine the small changes in the pseudorange.

I/Q Demodulation

We use I/Q demodulation in all of the standard GPS tracking loops. This simple process has two functions:

- Demodulates the navigation data from the carrier
- Provides an indicator of the frequency and phase errors between the reference and input signals

This section discusses I/Q demodulation outside the scope of GPS to simplify the later tracking loop sections. Figure 12 shows a typical time-domain I/Q demodulation circuit. Our discussion in this section uses the continuous time domain, but it applies equally well to digital processing.

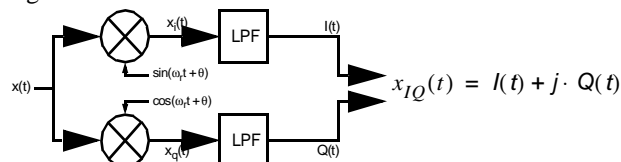


FIGURE 12. Standard I/Q Demodulation Diagram

Let us assume an input signal $x(t)$ with no noise:

$$x(t) = D(t) \cdot \sin(\omega_c t) \quad (6)$$

where $D(t)$ is the modulated data signal, t is time, ω is frequency and the subscripts c, r, e refer to carrier, reference and error respectively.

We also assume that $\omega_e = \omega_r - \omega_c$ is fairly small. $x(t)$ is divided into 2 channels, in-phase (I) and quadrature (Q). The I-channel is multiplied by the reference signal, $\sin(\omega_r t + \theta)$, and low-pass filtered, which gives

$$I(t) = \frac{1}{2} \cos(\omega_e t + \theta) \cdot D(t) \quad (7)$$

where θ is the phase difference between the carrier and the reference. The Q-channel is multiplied by the reference signal plus a 90° phase offset, $\cos(\omega_r t + \theta)$, and low-pass filtered, which gives

$$Q(t) = -\frac{1}{2} \sin(\omega_e t + \theta) \cdot D(t). \quad (8)$$

These two output values are combined to form the complex number $x_{IQ}(t) = I(t) + jQ(t)$. The phase angle of $x_{IQ}(t)$ is an indicator of the phase error between the input and reference carriers. Thus we can use $x_{IQ}(t)$ in the phase detector portion of a tracking loop to provide the phase error $\theta_e(t)$. A typical discriminator looks like

$$\theta_e(t) = k_p \cdot \text{atan}\left(\frac{Q(t)}{I(t)}\right) \quad (9)$$

where k_p is the discriminator gain. (This is explored further in the tracking loop sections below.)

Once we've achieved lock, (ω_e and θ have been minimized), the I-channel contains the navigation data. With good lock, we get

$$I(t) \cong \frac{1}{2} D(t) + \text{noise} \quad (10)$$

and

$$Q(t) \cong \text{noise} \quad (11)$$

When the input and reference signals are in phase the I-channel contains the data signal at half amplitude and the Q-channel contain nothing but noise.

Integration & Dump

I/Q results assume a continuous time domain with $x_{IQ}(t)$ updating instantly in response to changes in the reference or input signals. The digital GPS receiver, however, integrates the input data over a time interval and then dumps a single x_{IQ} value -- which changes the characteristics of

the x_{IQ} output. In this section we examine the effects of integration & dump on the phase angle of x_{IQ} . For simplicity's sake, we assume there is no noise and the navigation data equals a constant value of 1 for the duration of the integration.

Let us assume an initial phase offset θ and a constant frequency error ω_e , as defined above. Without integration & dump, we have

$$\begin{aligned} x_{IQNoIntegration}(t) &= \frac{1}{2} \cdot [\cos(\omega_e t + \theta) - j \cdot \sin(\omega_e t + \theta)] \\ &= \frac{1}{2} \cdot e^{-j(\omega_e t + \theta)} \end{aligned} \quad (12)$$

with the phase angle

$$\text{phase}(x_{IQNoIntegration}) = (-1)(\omega_e t + \theta) \quad (13)$$

Given an integration interval, $T_{CI} > 0$, a start time t_1 , a stop time $t_2 = t_1 + T_{CI}$, and a scale factor $\frac{1}{T_{CI}}$, the integration is

$$x_{IQ}(t) = \frac{1}{T_{CI}} \int_{t_1}^{t_2} \left(\frac{1}{2} e^{-j(\omega_e t + \theta)} \right) dt \quad (14)$$

which resolves to

$$x_{IQ}(t) = \frac{1}{2\omega_e T_{CI}} \left[e^{j\left(\frac{\pi}{2} - \omega_e t_2 - \theta\right)} - e^{j\left(\frac{\pi}{2} - \omega_e t_1 - \theta\right)} \right] \quad (15)$$

This final answer is difficult to simplify. **Data Fusion Corporation** is willing to state without formal proof, that the phase angle magnitude of Equation (15) is

$$|\text{phase}(x_{IQ})| = \left| \omega_e \cdot \left(t_1 + \frac{T_{CI}}{2} \right) + \theta \right| \quad (16)$$

which is the average phase angle over the integration range. Note the difference between this and Equation (13).

To show the significance of this difference, let us take a simple example. Assume $\theta = 0$ and $t_1 = 0$, but $\omega_e \neq 0$. After a time interval T_{CI} , we perform the integration and dump and get the following

$$|\text{phase}(x_{IQNoIntegration})| = |\omega_e T_{CI} + \theta| \quad (17)$$

$$|\text{phase}(x_{IQ})| = \left| \frac{\omega_e T_{CI}}{2} + \theta \right| \quad (18)$$

The phase angles are quite different, and we should keep this in mind when analyzing carrier tracking loops.

Generic Tracking Loop

The goal of a tracking loop is to produce a replica (reference) signal that matches some input $x(n)$. Figure 13 shows the block diagram for a generic time-domain digital tracking loop.

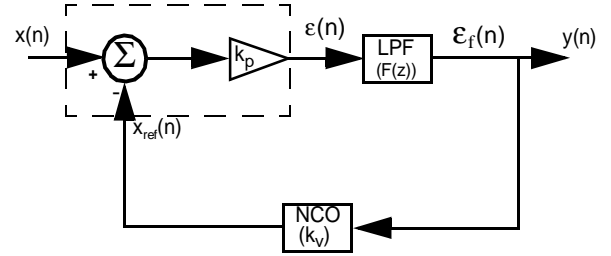


FIGURE 13. Standard Tracking Loop Diagram

The digital input signal, $x(n)$, is run through a discriminator (dotted box) to produce the error value $\epsilon(n)$. The discriminator is represented as a simple subtraction with an associated gain of k_p but actual GPS discriminators are more complex. The output is low-pass filtered to reduce noise. The resulting value is used by the numerically-controlled oscillator (NCO), with a gain of k_v , to produce a new replica signal for the next iteration of the loop. We want the loop output, $y(n)$, to be as close to zero as possible – at which point the replica signal will equal the input carrier.

Note: The following discussion is for the PLL tracking loop but is applicable to other loops as well.[3]

The performance analysis of a tracking loop is quite complex, particularly during the pull-in process or in the presence of noise. In order to simplify the analysis, a linear model of the feedback tracking loop shown below in Figure 14 is often used [5].

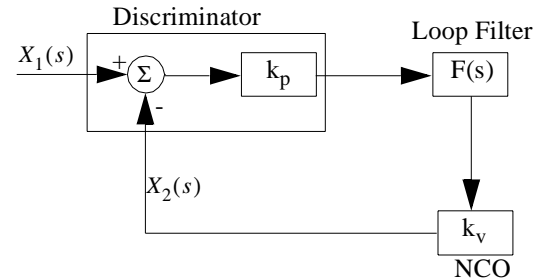


FIGURE 14. Linear model of a PLL in Laplace ('s') domain.

The loop transfer function is given by:

$$H(s) = \frac{k_v k_p F(s)}{s + k_v k_p F(s)} \quad (19)$$

where k_p is the discriminator gain, k_v is the NCO gain, and $F(s)$ is the loop filter transfer function.

Because of the ease of implementation and because of the presence of an integrator in its transfer function the active proportional-integrator filter is used as the loop filter. Akos[3] provides the following derivations for the transfer functions:

$$F(s) = \frac{sT_2 + 1}{sT_1} \quad (20)$$

So the transfer function becomes

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (21)$$

where loop natural frequency $\omega_n = \sqrt{\frac{k_v k_p}{T_1}}$

and the damping ratio $\zeta = \frac{\omega_n T_2}{2}$

From this the filter parameters, T_1 and T_2 can be picked taking into account the gain $k_v k_p$ to achieve the desired response. These parameters define a number of key properties associated with tracking loops such as: lock-in time, pull-in range, and pull-out range from which the performance of the implementation can be accurately predicted. The steady state error of this loop to a step change in phase or the frequency (i.e., first or second derivative of the phase angle) at the input is zero [4].

It is a second order tracking loop since the denominator of $H(s)$ is a second order of s . A first order loop is not used for tracking GPS signals since the PLL steady state error due to a frequency step or frequency ramp (i.e., second or third derivative of the phase angle) does not go to zero due to the presence of only one integrator.

In order to implement this loop in software, the continuous system must be changed into a discrete system.

The transfer from continuous 's' domain into discrete 'z' domain is accomplished through bilinear transform,

$$s = \frac{2(1-z^{-1})}{t_s(1+z^{-1})} \quad \text{where } t_s \text{ is the sampling interval (i.e., the}$$

coherent integration time, T_{CI}).

The loop filter transfer function in the 'z' domain is

$$F(z) = \frac{C_1 + C_2 - C_1 z^{-1}}{1 - z^{-1}} \quad (22)$$

where $C_1 = \frac{8\zeta\omega_n t_s}{K(4 + 4\zeta\omega_n t_s + (\omega_n t_s)^2)}$

and $C_2 = \frac{4(\omega_n t_s)^2}{K(4 + 4\zeta\omega_n t_s + (\omega_n t_s)^2)}$

and $K = k_v k_p$ [3].

This provides the most accurate discrete implementation of the linear phase lock loop model and is thus the approach used in the implementation in the software receiver for Costas PLL. **Data Fusion Corporation** has found it to be applicable for all of our tracking loops.

Natural Frequency ω_n and Damping Ratio ζ

The damping ratio plays an important role in the dynamic performance of the tracking loop. When working with an underdamped system ($0 < \zeta < 1$) the step response is rapid and will overshoot the desired state and will oscillate before settling down to the desired state. If the system is overdamped ($\zeta > 1$), the step response will be slow in achieving the desired state but will do so without any oscillation. The optimally flat-response is achieved using $\zeta = 0.707$, which corresponds to a second order Butterworth low pass filter. Thus this will be the choice used for both carrier and code tracking loops in the software receiver.

The selection of the natural frequency of the loop is a compromise. A relatively small natural frequency will provide excellent noise performance but will be unable to track dynamics induced on the signal. A relatively large natural frequency will be able to track signal dynamics but will have poor noise performance. These conclusions are based strongly on the noise bandwidth B_l approximation for the tracking loop given as

$$B_l = \frac{\omega_n(\zeta + 1/4\zeta)}{2}. \quad (23)$$

Based on the dynamics expected, typical noise bandwidths for the code and carrier tracking loops are 1Hz and 25Hz respectively. For a damping ratio of 0.707 this corresponds to a natural frequency of 2Hz and 50Hz respectively for the code and carrier tracking loops.

The performance of a tracking loop is determined by the measure of the tracking error variance at steady state. Given below is a table of tracking errors for different SNR(dB).

TABLE 5. Tracking Errors

dB	DLL			PLL		
	-15	-20	-25	-15	-20	-25
Chip error in # of chips	0.005	0.01	0.03	-	-	-
Frequency error in Hz	-	-	-	1	3	5
Phase error in degrees	-	-	-	2	5	8

GPS Specifics

In the GPS system our input signal, $x(n)$, is a combination of carrier, C/A-code, P(Y)-code, navigation data, and noise. A typical L1 signal, for example, looks like

$$x(n) = [\sin(\omega_c t + \theta_c) \cdot D(t - \tau) \cdot C(t - \tau)] + \quad (24)$$

$$\frac{1}{\sqrt{2}} [\cos(\omega_c t + \theta_c) \cdot D(t - \tau) \cdot P(t - \tau)] + N(t)$$

where

- ω_c is the carrier frequency adjusted for Doppler
- θ_c is the phase offset of the carrier at $t=0$,
- τ is the code and data offset at $t=0$
- $D(t-\tau)$ is the navigation data
- $C(t-\tau)$ is the C/A-code
- $P(t-\tau)$ is the P(Y)-code
- $N(t)$ represents the noise.

Note the $\frac{1}{\sqrt{2}}$ in front of the cosine term. This is the -3db gain applied by the GPS transmitter to the P(Y)-code signal. Also remember that we are dealing with an a down-converted carrier frequency (IF range) instead of the actual GPS frequency of 1.5GHz.

The GPS reference signal, $x_{ref}(n)$, is a combination of carrier and code. For example, we use the following signal in coarse (C/A) tracking:

$$x_{ref}(n) = \sin(\omega_c t + \theta_c) \cdot C(t - \tau) \quad (25)$$

GPS systems typically use 2 loops, one each for code and carrier tracking. The carrier loop gives us a good Doppler offset, while the code loop provides a precise code offset.

Carrier Loops

Figure 15 shows a typical GPS carrier tracking loop, which is a combination of the generic tracking loop in Figure 13 and the I/Q demodulation in Figure 12. Given the input signal $x(n)$ we strip the code and carrier, leaving navigation data as the output $y(n)$. The complex signal, $I(n) + j \cdot Q(n)$ is passed through a discriminator and filtered to determine the carrier tracking error $\epsilon_f(n)$, which is then used by the NCO to generate a new reference signal.

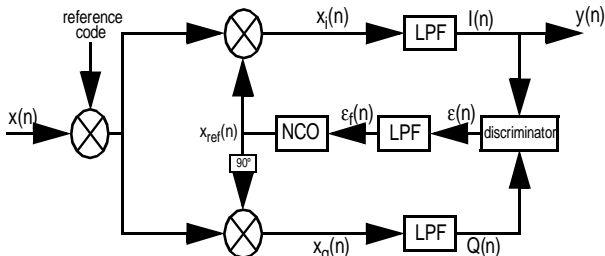


FIGURE 15. GPS Carrier Tracking Loop

Costas Phase Lock Loop (PLL)

A block diagram for the Costas loop implementation is given in Figure 16 [4],

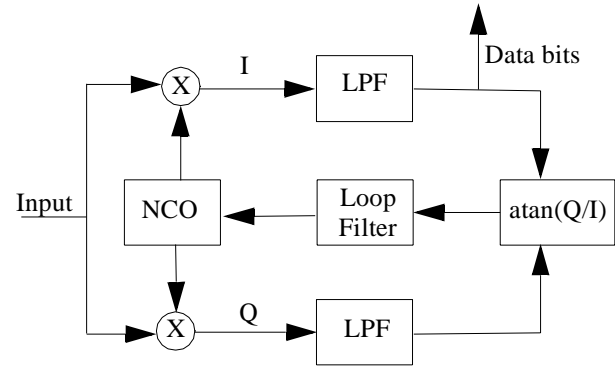


FIGURE 16. Costas Loop used for carrier tracking.[3]

It is very similar to the structure shown in Figure 15. The main difference is in the phase detector function. It uses the arctangent discriminator function. This is the optimal (maximum likelihood estimator) phase detector, but also the most computationally expensive [6]. Assuming the loop is operating in the lock mode, the modulated data bits are present on the in-phase arm of the Costas loop directly after the low pass filter.

The advantage of a Costas PLL over a regular PLL is that it can easily handle 180° phase shifts and continue tracking. The difference is the discriminator's use of the two-quadrant atan() function instead of the four-quadrant atan2() function.

Frequency Lock Loop (FLL)

A Frequency Lock Loop (FLL) uses frequency error to track the input carrier, unlike a PLL which uses phase error. Referring back to Figure 15, the discriminator output $\epsilon(n)$ is the frequency error $\omega_e(n)$, which is then filtered and provided as input to the NCO. In general, a Costas PLL tracks more tightly, but the FLL can track a larger noise bandwidth and is more robust in the presence of dynamic stress [6].

Much of the available GPS literature tends to focus on the PLL, with a cursory discussion of the FLL (if any). This is understandable, given that a PLL may be sufficient under normal conditions. In those situations, however, where noise is an issue or where a large noise bandwidth is required, the receiver may not be able to track without an FLL. Kaplan recommends using a PLL/FLL combination that closes the bandwidth with an FLL and then switches to the PLL [6].

Our experience with FLLs at *Data Fusion Corporation* (discussed in more detail below) has shown they do not perform well with coherent integration values greater than

2 msec, due to the increased frequency of navigation bit flips.

FLLs are also known as Automatic Frequency Control (AFC) loops.

Calculating the Frequency Error

In continuous-time systems, a frequency $\omega(t)$ is defined as the change in phase angle over time

$$\omega(t) = \frac{d}{dt}\theta(t) \quad (26)$$

In digital systems, we approximate the derivative by using the change in phase angle over two consecutive inputs.

$$\omega(n) = \frac{\Delta\theta}{\Delta t} = \frac{\theta(n) - \theta(n-1)}{t(n) - t(n-1)} \quad (27)$$

The phase angles are available via the `atan2()` function, and for GPS the denominator in Equation (27) is the coherent integration time T_{CI} . Thus, the FLL discriminator will collect and demodulate (See ‘‘I/Q Demodulation’’ on page 9.) two consecutive input values, then compute the frequency error

$$\omega_e(n) = \frac{\Delta\theta_e}{T_{CI}} = \frac{\text{atan2}(X_{IQ}(n)) - \text{atan2}(X_{IQ}(n-1))}{T_{CI}} \quad (28)$$

The astute reader may ask why we switched from frequency $[\omega(n)]$ to error frequency $[\omega_e(n)]$. The answer lies in the fact that the complex phase angles of the x_{IQ} terms correspond to the phase error $\theta_e(n)$.

More on Discriminators

The general discriminator given in Equation (28) has a high computational burden. Kaplan [6] provides a list of discriminators requiring less calculations. This section addresses one of those discriminators, which is equivalent to Equation (28):

$$\frac{\text{atan2}(\text{cross}, \text{dot})}{t(n) - t(n-1)} \quad (29)$$

where

$$\text{dot} = I(n-1) \cdot I(n) + Q(n-1) \cdot Q(n) \quad (30)$$

$$\text{cross} = I(n-1) \cdot Q(n) - I(n) \cdot Q(n-1) \quad (31)$$

At first glance, it may not be obvious how Equations (28) and (29) are similar, so we shall spend the rest of this section explaining it.

In order for Equation (29) to equal the frequency error $\omega_e(n)$, the following must be true:

$$\text{atan2}(\text{cross}, \text{dot}) = \theta(n) - \theta(n-1) \quad (32)$$

The nomenclature for ‘cross’ and ‘dot’ gives a bit of insight into what is happening. They are formed by taking the complex dot product of the current and previous x_{IQ} values. (Recall that the complex dot product multiplies the first value times the complex conjugate of the second.) Given two complex numbers, $x_2 = a_2 + j \cdot b_2$ and $x_1 = a_1 + j \cdot b_1$, the complex dot product is

$$\begin{aligned} x_2 \cdot \overline{x_1} &= (a_2 + j \cdot b_2) \cdot (a_1 - j \cdot b_1) \\ &= a_2 a_1 - j(a_2 b_1) + j(b_2 a_1) - (j^2)(b_2 b_1) \\ &= [a_2 a_1 + b_2 b_1] + j[b_2 a_1 - a_2 b_1] \end{aligned} \quad (33)$$

Replacing x_2 with $x_{IQ}(n)$ and x_1 with $x_{IQ}(n-1)$

$$\begin{aligned} x_{IQ}(n) \cdot \overline{x_{IQ}(n-1)} &= [I(n)I(n-1) + Q(n)Q(n-1)] \end{aligned} \quad (34)$$

which is exactly equal to $\text{dot} + j \cdot \text{cross}$. Thus, the `atan2(...)` term in Equation (29) returns the complex phase angle of $x_{IQ}(n) \cdot \overline{x_{IQ}(n-1)}$. Next we will show that this angle is equal to $\theta(n) - \theta(n-1)$.

Switching to exponential form, we have

$$x_{IQ}(n) = e^{j\theta(n)} \quad (35)$$

$$\overline{x_{IQ}(n-1)} = e^{-j\theta(n-1)} \quad (36)$$

multiplying these gives

$$x_{IQ}(n) \cdot \overline{x_{IQ}(n-1)} = e^{j\theta(n)} e^{-j\theta(n-1)} = e^{j[\theta(n) - \theta(n-1)]}$$

which has the desired complex phase angle.

We can now see that Equations (28) and (29) are equivalent, except that (29) is less computationally intensive.

Increasing the Number of Outputs

In the discrete-time domain, FLLs produce 1 output for every 2 inputs. A typical algorithm is shown here in pseudocode:

```

WHILE true
  <get next GPS input data>
  x1 = <calculate xIQ(n-1)>
  <get next GPS input data>
  x2 = <calculate xIQ(n)>
  <calculate freq error from x2 & x1>
  <update reference signal parms>
END WHILE

```

While this may be acceptable, we typically want to acquire and track as quickly as possible. So it is desirable to update our reference signal *every* time we receive a new input. One must be careful, however, when doing this. A common mistake is:

```

<get GPS input data>
x1 = <calculate  $x_{IQ}(n-1)$ >
WHILE true
  <get next GPS input data>
  x2 = <calculate  $x_{IQ}(n)$ >
  <calculate freq error from x2 & x1>
  <update reference signal parms>
  x1 = x2
END WHILE

```

As you can see, this pseudocode fragment processes the first input without providing an output. Then, for each successive input, it uses the current and previous inputs to generate the adjustment for the reference signal. Except for the very first input, it produces an output for every input.

This method is flawed because the reference signal used to calculate x1 has a different frequency than the one for x2. A correct (and more computationally intensive) method is to recalculate the older value once we've updated the reference signal frequency:

```

<get GPS input data>
x1 = <calculate  $x_{IQ}(n-1)$ >
WHILE true
  <get next GPS input data>
  x2 = <calculate  $x_{IQ}(n)$ >
  <calculate freq error from x2 & x1>
  <update reference signal parms>
  x1 = <calculate  $x_{IQ}(n)$  again>
END WHILE

```

FLLs and Navigation Data Bit Flips

As discussed in [6] the FLL ignores navigation data, *provided the input samples do not straddle a bit flip*. When this occurs, the $\Delta\theta$ value is offset by π which (for $T_{CI} = 1\text{msec}$) is an unfiltered frequency error of 500Hz--a significant amount.

Using a standard coherent integration time of $T_{CI} = 1\text{msec}$, we have at most 1 bit flip per 20 inputs, meaning 1 out of every 10 outputs may be in incorrect. The loop filter helps alleviate this. If we modify the FLL to provide an output for every input (see previous section), then it becomes 1 in 20. Also, once we get past initial tracking and complete the bit synchronization process, we know when the bit flips will occur and can adjust for them.

The situation becomes exacerbated, however, when we utilize a larger coherent integration time. For $T_{CI} = 3\text{msec}$, there is the possibility of a bit flip every 6.7 input samples, meaning every 3rd or 4th output may be invalid.

As the coherent integration time increases, FLL tracking performance is degraded. The effects of the navigation bits become more significant, and we may lose lock and/or tracking.

The FLL is especially vulnerable to navigation data bit flips during the initial stages of tracking. To help alleviate this problem, *Data Fusion Corporation* has implemented two algorithms. The first is a retry when transitioning from acquisition to tracking. If tracking fails, we restart it with the acquisition Doppler and code offset. If the cause of the divergence was an untimely bit flip, the second try has a reasonable chance of succeeding. We return to acquisition, however, if tracking fails after a user-defined time and/or number of retries.

The second method is a "safety net". Depending upon the quality of lock, *Data Fusion Corporation's* software receiver switches between FLLs and PLLs with varying bandwidths. When these transitions occur the tracking loop is reset and restarted with the new parameters. And similar to the switch from acquisition to tracking, the loops are especially vulnerable for a short period immediately after restarting. If a loop fails due to a ill-timed nav bit flip, tracking is lost and the receiver returns to the lengthy acquisition process. The safety net algorithm provides an alternative fallback position by continuing to run the previous tracking loop (the "safety net") in parallel. If the new loop fails within a user-defined time interval, the receiver reverts to the safety net.

FLLs and Phase Information

A close look at Equation (28) brings up another interesting point about FLL tracking loops--*they do not provide phase information*. Thus, while the NCO-generated reference signal may be the correct frequency, it will most likely have some phase offset from the input carrier.

Maximum Error and Coherent Integration

As mentioned previously, the atan2() function returns a value between $\pm\pi$. Equation (29), therefore, limits the maximum frequency error to $\frac{500\text{Hz}}{n}$ (where 'n' is the number of msec of coherent integration). Thus, $T_{CI} = 1\text{msec}$ is 500Hz, 3msec is 167Hz, and 10msec is 50Hz. This value is the maximum error the discriminator can handle before it starts to return invalid results. (Note the difference between this and the Costas PLL which tracks very well when it is exactly 180° out of phase due to its use of the atan() function instead of atan2().)

As long as the frequency error is within this boundary, the loop will function nicely. If we exceed the maximum error, however, atan2() returns the wrong value. For example, if $\Delta\theta$ is 1.2π , the atan2() function returns $(-0.8)\pi$ and the reference frequency is adjusted in the wrong direction.

Once again we see that FLLs and coherent integration don't mix well. For $T_{CI} = 3\text{msec}$, our maximum error is 167Hz -- a reasonable value in the early stages of tracking.

Code Tracking

Figure 17 shows a typical GPS code early/late delay lock loop (DLL). It is a combination of the I/Q demodulation in Figure 12 and a tracking loop with 2 error terms (early and late). This implementation of a DLL is very similar to the generic feedback structure shown in Figure 13. The majority of components in the block diagram of DLL represent the phase detector (shaded parts of Figure 17).

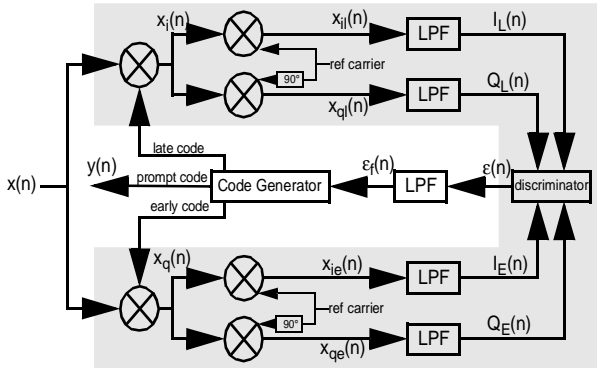


FIGURE 17. GPS Code Tracking Loop

The linear PLL model has been shown to be an adequate theoretical approximation of the early/late non-coherent delay lock loop [3].

The code tracking loop in the software receiver is used to despread the incoming signal as well as provide time-of-transmission measurements critical of range measurements and subsequently a position solution.

The code tracking loop used in the software receiver is the popular non-coherent Early-Late gate delay lock loop. The input to the loop is the composite signal consisting of the carrier modulated with navigation message and the PRN code. The input signal is split into two paths and correlated with two versions, an early and late of the locally generated PRN code. The two versions are equally spaced, typically ± 0.5 chip about the prompt synchronized code. Each of these two paths are mixed to baseband, generating in-phase and quadrature components. The energy in the early and late paths is differenced and the result is filtered and input to the NCO which clocks the PRN code generator. In this case bias in the error signal indicates which path, early or late, contains more energy and thus whether NCO needs to speed up or slow down the locally generated PRN code. Ideally these two paths are balanced and the resulting difference is zero.

Lock Detectors

When the GPS signal is being tracked, we need to know when it is tracking well and when it has lost track. Lock detectors are required to perform this function.

C/No

One way to implement a lock detector is by averaging the power $I^2 + Q^2$ over a certain period (say 100ms) and comparing that to a preset threshold. An averaged power below the threshold indicates loss of lock [1].

CdLi

Another way is the implementation in the GEC Plessey receiver. After the acquisition of the signal the code lock indicator CdLi is set to 4dB above the noise floor and subsequent CdLi values are calculated based on the difference equation $CdLi(n+1) = 0.99CdLi(n) + 0.01(I^2 + Q^2)$ [7]

If the value of CdLi goes below a preset threshold, which is about 2dB below the acquisition threshold, loss of lock is indicated.

Carrier Divergence

If, during the first few seconds of tracking, the reference frequency moves a significant distance away from the acquisition frequency, we are not tracking. For a cutoff value, **Data Fusion Corporation** uses the maximum size of one DFT acquisition bin -- 667Hz.

Carrier Deviation

If the standard deviation of the reference frequency is high (say 300Hz), we are most likely not tracking. **Data Fusion Corporation** calculates the standard deviation using the last 60 frequency values.

BIT SYNCHRONIZATION & DATA COMPRESSION

After initial C/A-code acquisition, databit timing is subject to navigation (NAV) bit offset ambiguity. This ambiguity is due to a lack of knowledge in the databit timing, namely the offset of the databits transmitted. The beginning of each C/A-code period is known, but it is not known where the NAV-bit data, which is composed of C/A-code periods, begins. If the assumed databit offset does not correspond to the actual databit offset, C/A-code periods from two databits will be assumed to be from the same databit period. As a result, the transmitted data determined from the data bits, would be in error.

Bit Synchronization Algorithm

Multiple techniques can be used to determine databit timing and eliminate the NAV-bit offset ambiguity. The histogram method partitions the assumed databit period into C/A -code periods. The 20 ms databit length is separated into 20 1-ms C/A -code periods. The algorithm senses sign changes between successive code periods and records these sign changes by incrementing the count in the bin corresponding to that particular code period. The code offset can be determined from a peak in the histogram that exceeds a pre-specified upper threshold. In the absence of noise, a peak will occur only at the true offset value. Due to random, noise-related perturbations to the signal, it is possible that peaks may appear at bins other than the true code offset. In the presence of noise, two possibilities exist: either the noise is weak enough such that the noise-influenced bins appear as background noise about the dominant peak or it is strong enough such that there is more than one dominant peak, with each of these peaks exceeding a pre-specified lower threshold.

The procedure in *Data Fusion Corporation's* histogram implementation is as follows: [1]

1. A cell counter K_{cell} is arbitrarily set and runs from 0 to 19.
2. Each sensed sign change is recorded by adding 1 to the histogram cell corresponding to K_{cell} .
3. The procedure continues until one of the following occurs:
 - a) Two cell counts exceed threshold NBS_2 .
 - b) Loss of lock
 - c) One cell count exceeds threshold NBS_1 .
4. If (a) occurs, bit synchronization fails because of low C/N_0 or lack of bit sign transitions, and bit synchronization is reinitialized. If (b) occurs, lock is reestablished. If (c) occurs, bit synchronization is successful, and the C/A -code period count offset is set to the correct value.

An example of a successful bit synchronization histogram is seen in Figure 18. The upper and lower dashed lines represent the thresholds NBS_1 and NBS_2 respectively. The

code ambiguity has been resolved and the NAV bit offset has been determined to be 3.

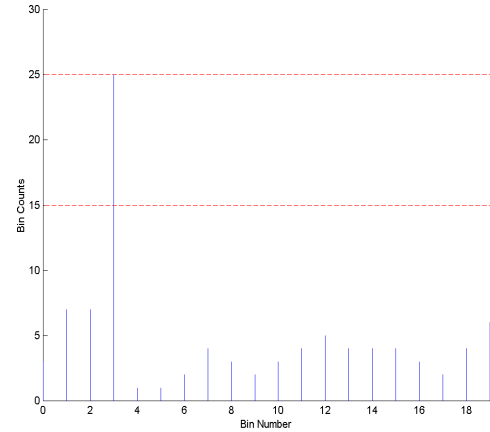


FIGURE 18. Bit synchronization histogram

The NAV-bit offset is defined as the number of C/A code periods preceding the first code period of a complete bit. For example, a C/A code sequence with an offset of 0 would begin with the first code period of a complete bit. An offset of 4 would have four preceding code periods of the previous bit and then begin with the first period of the current bit.

Transmitted Bits:

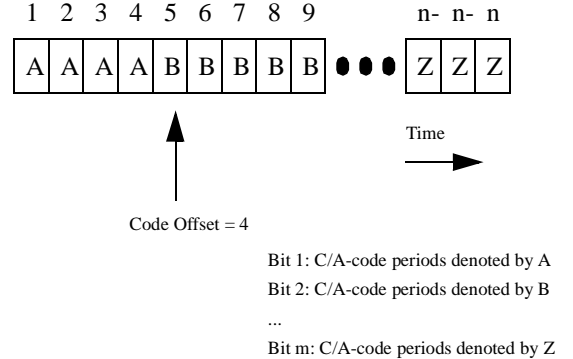


FIGURE 19. Pictorial representation of the NAV-bit offset

The upper and lower histogram thresholds, NBS_1 and NBS_2 , are derived in the following analysis. The probability of making an error in determining a sign change at a particular S/N_0 is

$$P_{esc} = 2P_e(1 - P_e) \quad (37)$$

where

$$P_e = \text{erfc}' \left[\sqrt{2 \left(\frac{S}{N_0} \right) T} \right] \quad (38)$$

if a phase-locked loop (PLL) is being used and T is C/A -code period (1 ms), where

$$erfc'(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{y^2}{2}} dy \quad (39)$$

The definition of the complementary error function in Equation (39) differs from the definition used by MathWorks. In MATLAB, the complementary error function is defined as

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \quad (40)$$

by a simple change of variables, one can relate the two definitions.

Let $t = \frac{y}{\sqrt{2}}$ and $dt = \frac{dy}{\sqrt{2}}$. Thus, Equation (40) becomes

$$erfc(x) = \frac{\sqrt{2}}{\sqrt{\pi}} \int_x^\infty e^{-\frac{y^2}{2}} dy \quad (41)$$

It is obvious that one erfc definition is only a multiplicative scaling of the other. The definition of the complementary error function used by **Data Fusion Corporation**, in terms of MATLAB's definition, is given as

$$erfc'(x) = \frac{1}{2} erfc_{MATLAB}(x) \quad (42)$$

The number of sign changes in a bin, N_{sc} , has a binomial distribution. In the correct offset bin, the average number of sign changes in T_{bs} seconds is $25T_{bs}$. Thus, $N_{sc} = 25T_{bs}$ in the correct bin and $N_{sc} = 50T_{bs}P_{esc}$ in the other bins.

Based on the bit synchronization period T_{bs} , the upper threshold can be specified as

$$NBS_1 = 25T_{bs} \quad (43)$$

On average, bit synchronization will be accomplished in T_{bs} seconds. The parameter T_{bs} is user-defined and **Data Fusion Corporation** has chosen to use $T_{bs} = 1$ second.

The standard deviation of N_{sc} in any bin is

$$\sigma_{N_{sc}} = \sqrt{50T_{bs}P_{esc}(1-P_{esc})} \quad (44)$$

The lower threshold can be chosen between two bounds. According to [1], NBS_2 should be chosen such that

$$25T_{bs} - 3(\sqrt{50T_{bs}P_{esc}(1-P_{esc})}) \leq NBS_2 \leq 50T_{bs}P_{esc} \quad (45)$$

where the upper bound is a three sigma spread from the mean of the bin corresponding to the correct offset and the lower bound is the mean of the bins corresponding to the other bins.

In order to reduce the number of computations for bit synchronization, it was assumed that the SNR remains relatively constant over the bit synchronization period. The upper and lower thresholds, NBS_1 and NBS_2 , are con-

stants set by the user. Theoretically the SNR could be recalculated for every code period. However, analysis over the range of expected SNR, values of $NBS_1 = 25$ and $NBS_2 = 15$ seems to satisfy most processing requirements.

Data Compression Algorithm

After bit synchronization has been achieved, it is possible to perform data compression to recover the original transmitted databits. All C/A-code periods, corresponding to a particular databit, are averaged over the NAV-bit length and depending on the sign, the databit is determined to be either +1 or -1. This algorithm is robust to noise and can tolerate multiple noise spikes that flip the sign of the true databit in a single databit period.

The performance of data compression is measured in terms of bit error rate (BER). For a PLL, the bit error rate is

$$BER = erfc' \left[\sqrt{40 \left(\frac{S}{N_0} \right) T} \right] \quad (46)$$

The bit error rate is derived from Equation (38) and is based upon the necessity of at least ten sign changes in a single NAV-bit for a bit error to occur. In the case of the PLL, there exists a sign ambiguity that is resolved during frame synchronization and as part of the parity algorithm.

FRAME SYNCHRONIZATION

Frame synchronization is required in order to process the GPS data. When timing uncertainties are large, parity decoding is not possible since the boundaries of the words are unknown. The following section details the procedure necessary to synchronize frames.

The GPS navigation message frame structure is shown in Each frame is partitioned into 5 subframes. Each subframe is subdivided into ten 30-bit words with the two leading

words being the telemetry (TLM) word and the handover (HOW) word.

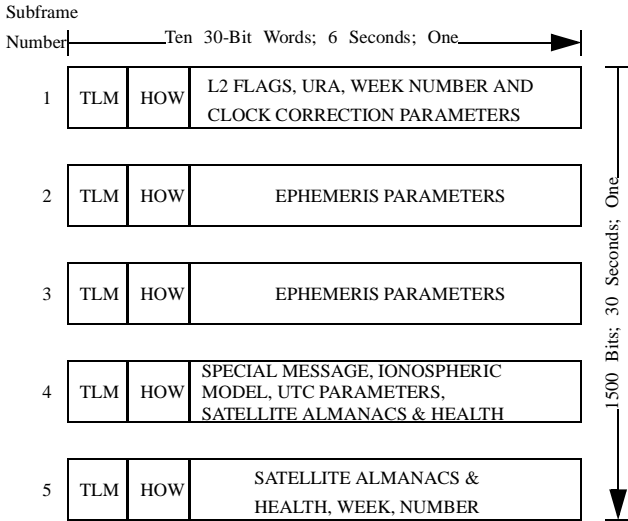


FIGURE 20. GPS navigation message dataframe structure

The structure for the TLM and HOW words is given in Figure 21. The TLM word is 30 bits long, occurs every six seconds in the data frame and is the first word in each subframe. Each TLM word begins with a preamble, followed by the TLM message, two reserved bits and six parity bits. Apparently in [2] the reserved bits are set to '1'.

The HOW word is 30 bits long, occurs every six seconds in the data frame and is the second word in the subframe, immediately preceded by the TLM word. The convention is that the most significant bits (MSB) are transmitted first. The HOW word begins with the seventeen MSBs of the time-of-week (TOW) count. The full TOW count consists of the 19 least significant bits (LSB) of the 29-bit Z-count. Bit 18 is the momentum flag (for SV configuration 000) or "alert" flag (for SV configuration 001) and bit 19 is the synchronization flag (for SV configuration 000) or anti-spoof flag for (SV configuration 001). [1]

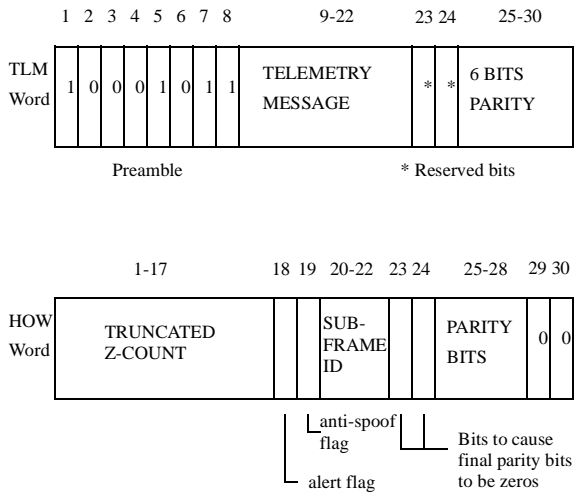


FIGURE 21. Structure of telemetry and handover words

The algorithm for positively finding the correct preamble and performing frame synchronization is as follows: [1]

1. Search for either an upright or inverted preamble.
2. When one is found, which could be a legitimate pattern somewhere else in the data stream, a check is required to see if it is the beginning of a 30-bit word. This is accomplished by collecting the following 22 bits and checking parity. If parity does not pass, the candidate preamble is discarded.
3. If parity passes, it verifies that the preamble existed at the beginning of a word. The parity algorithm will also resolve the sign ambiguity. However, there are also legitimate such patterns at beginning of other words, so additional checks are required. If it is the correct TLM word, the following word must be a HOW word that contains a truncated Z-count. The first eight bits of this truncated Z-count can also resemble a preamble.
4. Parity should pass on the HOW word. If not, the frame synchronization procedure should be restarted. Two checks can be made to verify a legitimate HOW word--the Z-count is reasonable, and it agrees with the subframe count. Of course, there is a small probability that these conditions could also occur elsewhere in the message. Thus, further checking is required.
5. If the HOW seems legitimate, provisional demodulation of the other words can commence, and they can be stored in memory. A final check on the next preamble and next Z-count solidifies the frame synchronization. That is, the preamble is where it is supposed to be, and the Z-count increments by one.

GPS time is referenced to a UTC zero time-point defined as midnight on the night of January 5, 1980. The largest unit used in stating GPS time is one week defined as 604,800 seconds. The Z-count is convenient for precisely counting and communicating GPS time. The Z-count is given as a 29-bit binary number consisting of two parts.

The nineteen least significant binary bits of the Z-count are referred to as the time of week (TOW) count and is defined as the number of X1 epochs that have occurred since the transition from the previous week. The TOW count range is from 0 to 403,199 X1 epochs (equaling one week) and is reset to zero at the end of each week. To aid rapid ground lock-on to the P-code signal, a truncated version of the TOW-count, consisting of the 17 most significant bits, is contained in the HOW word.

The ten most significant binary bits of the Z-count are assigned to a sequential number representing the present GPS week (modulo 1024). The range is from 0 to 1023 with its zero state defined as the week which began with the X1 epoch occurring at approximately midnight on the night of January 5, 1980.

Parity Decoding

It is sufficient to assume that the last 6 bits of each 30-bit word are the parity bits for decoding purposes. A (32,26) Hamming code is used as the parity encoding algorithm. The 24 bits of data are XOR'ed with the last raw bit of the previous word. The GPS Signal Specifications then provide equations for the calculation of the remaining 6 parity bits. The last two raw bits (D_{29}^* and D_{30}^*) from the previous word, prior to processing, are XOR'ed in combination with the 24 bits of data yielding the 6 parity bits.

$$D_1 = d_1 \oplus D_{30}^* \quad (47)$$

$$D_2 = d_2 \oplus D_{30}^* \quad (48)$$

$$D_3 = d_3 \oplus D_{30}^* \dots D_{24} = d_{24} \oplus D_{30}^* \quad (49)$$

$$D_{25} = D_{29}^* \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_{10} \oplus \dots \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{17} \oplus d_{18} \oplus d_{20} \oplus d_{23} \quad (50)$$

$$D_{26} = D_{30}^* \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{11} \oplus \dots \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{18} \oplus d_{19} \oplus d_{21} \oplus d_{24} \quad (51)$$

$$D_{27} = D_{29}^* \oplus d_1 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_8 \oplus \dots \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{19} \oplus d_{20} \oplus d_{22} \quad (52)$$

$$D_{28} = D_{30}^* \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus \dots \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{20} \oplus d_{21} \oplus d_{23} \quad (53)$$

$$D_{29} = D_{30}^* \oplus d_1 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_9 \oplus \dots \oplus d_{10} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{18} \oplus d_{21} \oplus d_{22} \oplus d_{24} \quad (54)$$

$$D_{30} = D_{29}^* \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus \dots \oplus d_{11} \oplus d_{13} \oplus d_{15} \oplus d_{19} \oplus d_{22} \oplus d_{23} \oplus d_{23} \quad (55)$$

where

- d_1, d_2, \dots, d_{24} are the source data bits;
- the symbol '*' is used to identify the last 2 bits of the previous word of the subframe;
- $D_{25}, D_{26}, \dots, D_{30}$ are the computed parity bits;
- $D_1, D_2, \dots, D_{29}, D_{30}$ are the bits transmitted by the SV;
- \oplus is the modulo-2 or exclusive-or (XOR) operation.

In order to check the parity of the received word and decode the data it is necessary to reverse the encoding process. The last two raw bits (D_{29}^* and D_{30}^*) from the previous word are used with the first 24 bits of the 30-bit word to form a 26-bit vector d , after XOR'ing the 24 bits with D_{30}^* to resolve the sign ambiguity. Parity verification is performed by forming

$$D = H \oplus d \quad (56)$$

where D is the received 6-bit parity vector and H is a 6 x 26 mask vector. The mask vector has ones in elements corresponding to the XOR's in the parity equations and zeros elsewhere. If the result of the matrix XOR in Equation (56) is equal to D then parity has been verified. The last 24 bits of d are then the decoded bits of data with sign ambiguity resolved.

An example flow chart summarizes the previous analysis and defines one way of recovering data and checking parity, see Figure 22.

This parity algorithm can detect up to three simultaneous bit errors and can correct 1-bit errors. It is particularly susceptible to burst errors. For random bit errors, the BER is binomially distributed and given by

$$BER \approx \left(\frac{32!}{4!28!} \right) P_e^4 (1 - P_e)^{28} \quad (57)$$

For simplicity, Equation (57) provides only the dominant term in a series of probabilities. In general, the BER for the (32,26) Hamming code is the probability of four or more errors occurring. Note that the probabilities are based on thirty two bits per word rather than thirty, this is due to the algorithm using the last two raw bits of the previous word for the parity equations. When more than four errors occur, it may be possible to detect them using the parity algorithm. The probability of the errors not being detected is the undetected BER (UBER) given by

$$UBER \approx \frac{4}{32} \left(\frac{32!}{4!28!} \right) P_e^4 (1 - P_e)^{28} \quad (58)$$

The scaling term for the UBER is based on the Hamming code's ability to detect errors beyond the number it is guaranteed to detect. It will always detect three errors or less, but it may or may not detect errors that number beyond three.

The probability of receiving a word correctly is based on the probability of receiving every bit correctly (correct word rate, CWR). Assuming independence, the probability of receiving a word correctly is equal to the product of the probabilities of receiving each bit incorrectly including the last two bits from the previous word.

$$CWR \approx (1 - P_e)^{32} \quad (59)$$

The probability of losing a word (word error rate, WER) is the complement of CWR and is given as follows:

$$WER = 1 - (1 - P_e)^{32} \quad (60)$$

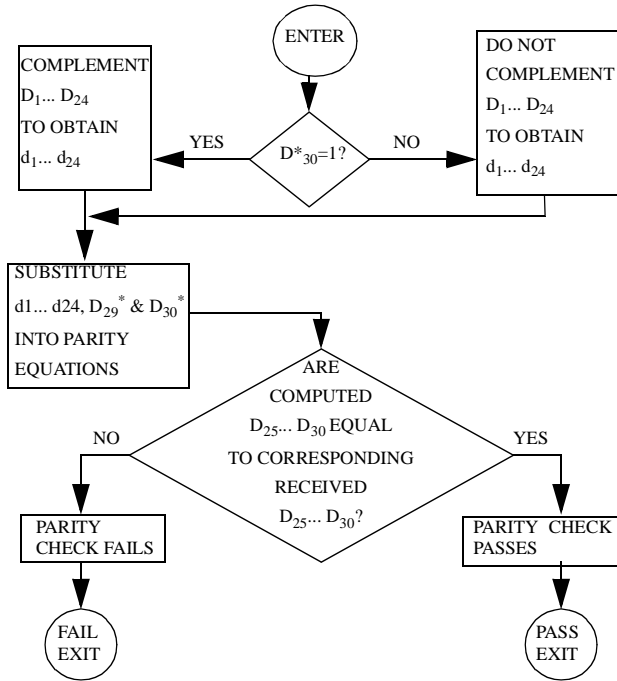


FIGURE 22. Example flow chart for parity algorithm

NAVIGATION SOLUTION

Fundamentally, a Navigation solution is an estimate of the user position and any other required parameters. The term 'state' is used to describe all the parameters to be determined. The typical states in a GPS navigation estimator are three components of position, clock offset and clock drift. In a moving application three components of velocity are added. There are many applications where the GPS is integrated with one or more other sensors, such as an altimeter or an inertial navigation system (INS). In such configurations the state may be expanded to include specific sensor error states; however, in this section, we restrict ourselves to stand-alone GPS navigation estimation.

In order to compute the position estimate, the Navigation algorithm should have a knowledge of pseudorange (explained later in this section) to at least four GPS satellites and the position of those satellites. The pseudorange is obtained from the DLL (Delay Lock Loop) tracking loop which tracks the incoming PRN code and the position of the satellites can be determined from the Navigation data demodulated using the PLL (Phase Lock Loop). Also the error models to correct the satellite clock offset and atmospheric delays are required. These parameters are contained in the navigation message. The navigation message contains in addition to other parameters the Keplerian orbital parameters of the GPS satellites in view, which are used to compute the satellite position.

Single Point Solution

The GPS point positioning problem is concerned with the task of solving for four unknowns; X_r , Y_r , Z_r , the receiver coordinates in a suitable frame of reference (usually ECEF) and δ_r , the receiver clock bias. The DLL provides the PRN code offset which can be used along with the Navigation data demodulated by the PLL to determine the range to each of the satellites being tracked. There are various error sources which corrupt the range measurement, they are:

- Errors due to Ionospheric and tropospheric delay - the ionospheric correction parameters are included in the Navigation data and corrects for about 70% of the delay. Various tropospheric models are available which can be used to correct for tropospheric delay.
- Satellite and receiver clock errors.- satellite clock bias and drift term is included in the Navigation message. Receiver clock bias term is part of the navigation solution as explained later.
- Errors due to multipath and receiver noise.

The range measured by the receiver without any of the above corrections is called the Pseudorange.

The model equation for the code pseudorange at L1 frequency is given below:

$$P_i = R + \delta^s - \delta_r + \rho_{iono} + \rho_{tropo} + \rho_{multi} + \rho_{unknown} \quad (61)$$

where P_i is the pseudorange between *Satellite_i* and the receiver in meters, δ_r is the receiver clock error in meters, δ^s is the satellite clock error in meters, R is the true geometric range between the satellite and the receiver, ρ_{iono} is the ionospheric delay in meters, ρ_{tropo} is the tropospheric delay in meters, ρ_{multi} is the error due to multipath in meters, and $\rho_{unknown}$ is the unmodeled sources of error in meters.

The single frequency users can get an approximate correction for ionospheric delays by using the model of the ionosphere with model parameters transmitted in the downlink datastream [6]. However the ionosphere varies in a manner difficult to predict, hence the model provides only an approximate correction. The two-frequency user with access to the P(Y) code can correct for most of this delay error by measurement.

Many tropospheric delay models are available [6]. The one planned to be used in future version of the Baseline receiver is the zenith delay model. The satellite clock error model can be read from the navigation message. The error due to multipath is not modelled at present.

Let $\rho_i = P_i - \rho_{tropo} - \rho_{iono} - \delta^s$, then ignoring the error due to multipath and unmodeled errors, equation (61) can be written as

$$\rho_i = R + \delta_r = F(x_p, y_p, z_p, \delta_r) \quad (62)$$

The geometric range is given by

$$R = \sqrt{(x_s - x_r)^2 + (y_s - y_r)^2 + (z_s - z_r)^2} \quad (63)$$

where x_s, y_s and z_s are Satellite coordinates and x_r, y_r and z_r are the receiver coordinates in the same frame of reference. The receiver requires at least four non-linear equations of type (62) above to solve for the receiver coordinates and the receiver clock error term. These non-linear equations can be solved for the unknowns by employing iterative techniques based on linearization.

Linearization

Using an approximate receiver position location (x', y', z') and time bias estimate δ'_r , an approximate pseudorange can be calculated as

$$\rho'_i = \sqrt{(x_s - x')^2 + (y_s - y')^2 + (z_s - z')^2} + \delta'_r \quad (64)$$

The unknown user position and receiver clock offset is considered to consist of an approximate component and an incremental component:

$$\begin{aligned} x_r &= x' + \Delta x \\ y_r &= y' + \Delta y \\ z_r &= z' + \Delta z \\ \delta_r &= \delta' + \Delta \delta_r \end{aligned}$$

So we have

$$F(x_p, y_p, z_p, \delta_p) = F(x' + \Delta x, y' + \Delta y, z' + \Delta z, \delta'_r + \Delta \delta_r)$$

This function can be expanded about the approximate point and associated predicted receiver clock offset using a Taylor series expansion. The higher order (>1) partial derivatives are neglected to eliminate non-linear terms in the Taylor series expansion. After going through a little mathematics, the result boils down to:

$$\rho_j - \rho'_j = \frac{x' - x_s}{r'} \Delta x + \frac{y' - y_s}{r'} \Delta y + \frac{z' - z_s}{r'} \Delta z + \Delta \delta_s \quad (65)$$

where

$$r' = \sqrt{(x_s - x')^2 + (y_s - y')^2 + (z_s - z')^2} \quad (66)$$

In general for more than four satellites the vector-matrix equation would look like

$$\Delta X = (H^T H)^{-1} H^T \Delta \rho \quad (67)$$

where $\Delta \rho = [\Delta \rho_j]$ for $j=1,2,\dots,i$ is a $i \times 1$ column vector and

$$\Delta X = [\Delta x \ \Delta y \ \Delta z \ \Delta \delta_r]^T \quad (68)$$

$$H = \begin{bmatrix} \frac{x' - x_1}{r_1} & \dots & \frac{z' - z_1}{r_1} & 1 \\ \dots & \dots & \dots & \dots \\ \frac{x' - x_i}{r_i} & \dots & \frac{z' - z_i}{r_i} & 1 \end{bmatrix} \quad (69)$$

This is the unweighted least squares solution for the receiver position offset ΔX

$\Delta \rho$ is the vector offset of the error-free pseudorange values corresponding to the receiver's actual position and the pseudorange values that correspond to the linearization point. H is the $i \times 4$ matrix, where $i = \text{number of visible satellites}$. ΔX consists of the position offset of the receiver from the linearization point and the receiver time bias from the bias assumed at the linearization point.

For the weighted least squares solution, the vector-matrix equation looks like:

$$\Delta X = (H^T W H)^{-1} H^T W \Delta \rho \quad (70)$$

where $W = \begin{pmatrix} 1 \\ \sigma_d^2 \end{pmatrix} I_{ixi}$

the error terms are assumed uncorrelated, otherwise the off-diagonal elements would be non-zero.

σ_d^2 is the variance of observation noise.

The solution starts with an apriori estimate of the user state (position and clock offset). The least squares equation is solved to calculate ΔX . Subtracting this from the apriori estimate gives an improved estimate. This new estimate is closer to the true value of the state. If the apriori estimate used to construct H , is off by a lot, the least squares solution may be iterated until the change in the estimate is sufficiently small.

The accuracy of the solution is decided by two factors, the measurement quality and the user to satellite geometry. The measurement quality is described by the variance of the measurement error, which for a typical pseudorange is in the range of 0.3 to 30 m, depending on the error conditions. The geometry is described by the H matrix which is composed of line of sight vectors and ones for the receiver clock states.

In weighted least squares equation the matrix $(H^T W H)^{-1} W H^T$ which is sometimes called the least-squares solution matrix is a $4 \times i$ matrix and depends only on the relative geometry of the receiver and satellites participating in the least square solution computation. The pseudorange errors $\Delta \rho$ are considered to be random vari-

ables and assumed to have components that are jointly Gaussian and zero mean.

The goal is to come up with a solution ΔX which will minimize the error between the data model and the measurements in a least square sense.

The covariance matrix $(H^TWH)^{-1}$ is calculated, whose diagonal elements correspond to the variance of the error in the computed value of the receiver position as shown below:

$$\begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \cdot & \cdot \\ \cdot & \sigma_x^2 & \cdot & \cdot \\ \cdot & \cdot & \sigma_x^2 & \cdot \\ \cdot & \cdot & \cdot & \sigma_x^2 \end{bmatrix} \quad (71)$$

All the terms are not shown. The off-diagonal elements are the cross-correlation terms.

The following DOPs (dilution of precision) summarize the contribution of the geometry.

$$A = (H^TWH)^{-1}$$

$$GDOP = \sqrt{\text{trace}(A)} \quad \text{geometrical DOP}$$

$$PDOP = \sqrt{A_{11} + A_{22} + A_{33}} \quad \text{position DOP}$$

$$TDOP = \sqrt{A_{44}} \quad \text{time DOP}$$

Thus, the total position error can be estimated by $\sigma_{dx}PDOP$.

The equations for a single-point velocity solution are identical with the pseudorange measurements replaced by pseudorange rates

MATLAB AND C TOOLKITS

Data Fusion Corporation through the course of the research has developed multiple toolkits. The various toolkits are a MATLAB toolkit, a C toolkit (either object code or source), and a DSP/C toolkit package. Each toolkit consists of a library of files and programs used to implement our IF GPS Baseline receiver. These toolkits will enable users to simulate specific GPS scenarios without the need to code and test the generic GPS algorithms. Furthermore, these toolkits are an excellent starting point to perform additional research and development in the field of GPS.

The toolkits are in beta test now and are available at a substantial discount. We invite everyone to contact *Data Fusion Corporation* for demos and additional details.

ACKNOWLEDGMENTS

This work was supported in great extent by the U.S. Air Force, Air Force Research Laboratory, Wright-Patterson AFB, under contract number F33615-98-C-1316. The authors gratefully acknowledge the following for their contribution to this research effort: Major Clyde Heddings, USAF; Peter Howe, WPAFB; James Leonard, WPAFB; Captain Andrew Proud, USAF; and Dr. James Tsui, WPAFB.

REFERENCES

1. Parkinson, Bradford W. (Ed) and James J. Spilker Jr. (Ed), Global Positioning System: Theory and Applications Volume I, Washington, DC: American Institute of Aeronautics and Astronautics, Inc., 1996.
2. IRN-200C-001, ICD-GPS-200C, 13 Oct 1995.
3. Akos, Dennis, "A Software Radio approach to Global Navigation Satellite System receiver design", Ph.D. dissertation, Ohio University, August 1997.
4. Best, Roland E., Phase Locked Loops: Design, Simulation, and Applications, 4th edition, Mc-Graw Hill.
5. James Bao-Yen Tsui, Fundamentals of Global Positioning System Receivers. A software Approach, Wiley Inter-Science.
6. Kaplan, Elliot D., Editor, Understanding GPS: Principles and Applications, Artech House.
7. GPS Builder Designer's Guide, GEC Plessey Semiconductors, GPS Group, Wiltshire, United Kingdom, Nov. 1994.
8. Demler, M.J., High-Speed Analog-to-Digital Conversion, Academic Press, Inc, 1991.
9. Daugherty, K.M., Analog-to-Digital Conversion -- A Practical Approach, McGraw-Hill, Inc., 1995.
10. Amoroso, F., "Adaptive A/D Converter to Suppress CW Interference in DSPN Spread-Spectrum Communications", IEEE Transactions on Communications, Vol. Com 31, No.10, October 1983.
11. Texas Instruments, "Understanding Data Converters" SLAA013 July 1995.
12. Meyr, H. & Ascheid, G., Synchronization in Digital Communications, Vol.1, Wiley Interscience, 1990.
13. Van Nee, J.R. & Coenen, J.R.M., "New Fast GPS Code-Acquisition Technique Using FFT", Electronics Letters, 17th January 1991, Vol. 27, No. 2.