



**The Abdus Salam
International Centre for Theoretical Physics**



2135-1

**Second Workshop on Satellite Navigation Science and Technology for
Africa**

6 - 23 April 2010

GPS Receivers, Receiver Signals and Principles of Operation

Phillip W. Ward
*NavWard Consultants
Garland, TX
USA*



Introduction to GPS Receiver Design Principles

Phillip W. Ward, P.E.
Navward GPS Consulting

Introduction to GPS Receiver Design Principles

Session Outline

This introductory course on GPS receiver design principles is intended to familiarize the student with the internal components of any global navigation satellite system (GNSS) receiver architecture. Although every GNSS receiver design is uniquely tailored to its intended market and operational application, there is much in common with every GNSS receiver design. This course is taught in the context of the first civil user code and currently the most widely used GNSS signal in space, the GPS coarse/acquisition (C/A) code. As the name implies and as an interesting historical fact, the GPS C/A code was originally intended only to be the “stepping stone” signal into the GPS precision (P) code and was not originally intended to be a steady state satellite navigation signal. Fortunately, the C/A code was so well designed that it became the de-facto civil code after the U.S. Department of Defense (DoD) decided to encrypt the P code, beginning with the GPS Block II satellites. This was done to deny access to enemy military forces and to make it more difficult for an enemy to spoof (falsify) the resulting military P(Y) code signal.

The course consists of six sessions beginning with a high level description of the GPS signals that are processed using a “Generic digital GPS receiver” design as the teaching architecture. This is followed by “GPS receiver baseband processes” that take place after the signal is digitized. Within these baseband processes, there is more design elaboration on the following important topics: “Code tracking loop design;” “Carrier tracking loop design;” “Code and carrier loops filter design;” and, “Extracting measurements from code and carrier loops.” These six sessions should provide each student a sound foundation for understanding the principles of operation of all GNSS receivers.

Session Outline



- I - Generic digital GPS receiver
- II- GPS receiver baseband processes
- III - Code tracking loop design
- IV - Carrier tracking loop design
- V- Code and carrier loops filter design
- VI - Extracting measurements from code and carrier loops

Session I



Session I

Generic digital GPS receiver



- Satellite signal modulation and acquisition overview
- Generic digital receiver block diagram
- Analog-to-digital (A/D) conversion
- Digital receiver channel block diagram
- Baseband processor code and carrier tracking loops block diagram
- Baseband signal processing
- Code phase assignments and initial code sequences for C/A codes
- Code generator design: polynomials and initial states

Satellite signal modulation

The GPS space vehicles (SVs) transmit two carrier frequencies called L1, the primary frequency, and L2, the secondary frequency. The carrier frequencies are modulated by spread spectrum codes with a unique pseudo random noise (PRN) sequence associated with each SV and by the navigation data message. All SVs transmit at the same two carrier frequencies, but their signals do not interfere significantly with each other because of the PRN code modulation. Since each SV is assigned a unique PRN code and all of the PRN code sequences are almost uncorrelated with each other, the SV signals can be separated and detected by a technique called code division multiple access (CDMA). In order to track one SV in common view with several other SVs by the CDMA technique, a GPS receiver must replicate the PRN sequence for the desired SV along with the replica carrier signal, including Doppler effects. Two carrier frequencies are provided to permit the two-frequency user to measure the ionospheric delay since this delay is related by a scale factor to the difference in signal time of arrival for the two carrier frequencies. Single frequency (L1 only) users must estimate the ionospheric delay using modeling parameters which are broadcast to the user in the navigation message. The characteristics of these signals will be explained in more detail.

Satellite signal modulation

- GPS space vehicles (SVs) transmit two carrier frequencies
 - L1 = primary frequency = 1575.42 MHz
 - L2 = secondary frequency = 1227.60 MHz
- Carrier frequencies modulated by
 - Spread spectrum codes with unique pseudo random noise (PRN) sequence associated with each SV
 - Navigation data message = 50 bps

Frequencies and modulation format

The table below summarizes the frequencies and modulation that are used on the GPS satellites. Note that the code usually selected by the Control Segment on L2 is P(Y)-code, but it is possible that the C/A-code could be turned on instead of P(Y)-code on L2. Also note that the 50 Hz navigation data message is usually modulated on L2 P(Y)-code, but can be turned off by the Control Segment to improve jamming performance. This is because a pure PLL carrier tracking loop has a 6 dB higher threshold than a Costas PLL carrier tracking loop and because the predetection integration time can be increased by more than 20 ms, which increases the tracking thresholds of both the code and carrier tracking loops. We will discuss these techniques in detail in this course. For now, just remember that for the L2 link there are three possibilities: P(Y)-code with data, P(Y)-code with no data and C/A-code with data.

Frequencies and modulation format

Signal Priority	Primary	Secondary
signal designation	L1	L2
carrier frequency (Hz) $f_0 = 10.23 \times 10^6$	1575.42×10^6 $154 f_0$	1227.60×10^6 $120 f_0$
PRN codes chipping rate (chips/s) $R_0 = 10.23 \times 10^6$	$P(Y) = R_0$ and $C/A = R_0/10$	$P(Y) = R_0$ or $C/A = R_0/10$
navigation message (bps)	50	50

GPS signal acquisition overview

In practice, a GPS receiver must replicate the PRN code that is transmitted by the SV that is being acquired by the receiver, then it must shift the phase of the replica code until it correlates with the SV PRN code. The same correlation properties occur when cross-correlating the transmitted PRN code with a replica code as occurs for the mathematical autocorrelation process for a given PRN code. When the phase of the GPS receiver replica code matches the phase of the incoming SV code, there is maximum correlation. When the phase of the replica code is offset by one chip or more on either side of the incoming SV code, there is minimum correlation. This is indeed the manner in which a GPS receiver detects the SV signal when acquiring or tracking the SV signal in the code-phase dimension. It is important to understand that the GPS receiver must also detect the SV in the carrier-phase dimension by replicating the carrier frequency plus Doppler (and usually eventually obtains carrier phase lock with the SV signal by this means). Thus, the GPS signal acquisition and tracking process is a two-dimensional (code and carrier) signal replication process. In the code or range dimension, the GPS receiver accomplishes the cross-correlation process by first searching for the phase of the desired SV and then tracking the SV code state by adjusting the nominal chipping rate of its replica code generator to compensate for the Doppler-induced effect on the SV PRN code due to line-of-sight relative dynamics between the receiver and the SV. There is also an apparent Doppler effect on the code-tracking loop caused by the frequency offset in the receiver's reference oscillator with respect to its specified frequency. This error effect, which is the time bias rate determined by the navigation solution, is quite small for the code-tracking loop and is usually neglected. The code correlation process is implemented as a real-time multiplication of the phase-shifted replica code with the incoming SV code, followed by an integration and dump process. The objective of the GPS receiver is to keep the prompt phase of its replica code generator at maximum correlation with the desired SV code phase. However, if the receiver has not simultaneously adjusted (tuned) its replica carrier signal so that it matches the frequency of the desired SV carrier, then the signal correlation process in the range dimension is severely attenuated by the resulting frequency response roll-off characteristic of the GPS receiver. Consequently, the receiver never acquires the SV. If the signal is successfully acquired because the SV code and frequency are successfully replicated during the search process, but then the receiver subsequently loses track of the SV frequency, then the receiver subsequently loses code track as well. Thus, in the carrier Doppler frequency dimension, the GPS receiver accomplishes the carrier matching (wipe-off) process by first searching for the carrier Doppler frequency of the desired SV and then tracking the SV carrier Doppler state. It does this by adjusting the nominal carrier frequency of its replica carrier generator to compensate for the Doppler induced effect on the SV carrier signal due to line-of-sight relative dynamics between the receiver and the SV. There is also an apparent Doppler error effect on the carrier loop caused by the frequency offset in the receiver's reference oscillator with respect to its specified frequency. This error, which is common with all the satellites being tracked by the receiver, is determined by the navigation filter as the time bias rate in units of seconds per second.

The two-dimensional acquisition and tracking process can best be explained and understood in progressive steps. The clearest explanation is in reverse sequence from the events that actually take place in a real-world GPS receiver. The two-dimensional search and acquisition process is easier to understand if the two-dimensional steady-state tracking process is explained first. The two-dimensional code and carrier tracking process is easier to understand if the carrier tracking process is explained first. This is the explanation sequence that will be used. The explanation will first be given in the context of a generic GPS receiver architecture with minimum use of equations. This high-level overview will then be followed by more detailed explanations of the carrier and code tracking loops, including the most useful equations.

GPS signal acquisition overview

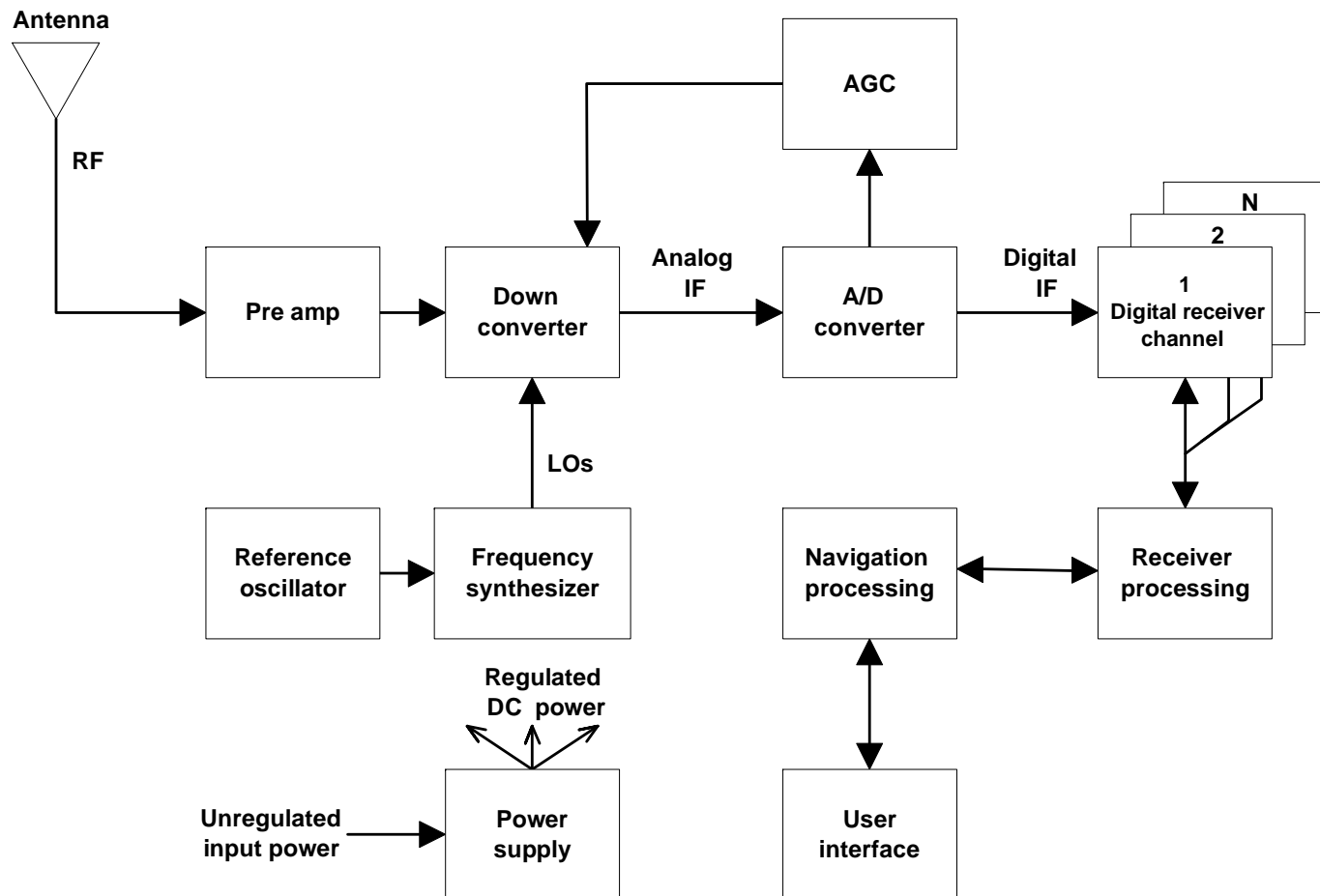
- GPS receiver signal acquisition
 - Receiver must replicate PRN code transmitted by SV
 - Receiver must shift phase of replica code until it correlates with SV PRN code as received at GPS receiver antenna
 - Receiver simultaneously adjusts (tunes) replica carrier to match desired SV carrier frequency (two-dimensional search) as received at GPS receiver antenna

Generic digital GPS receiver block diagram

In the figure below, the GPS radiofrequency (RF) signals of all SVs in view are received by a right hand circularly polarized antenna with nearly hemispherical gain coverage. These RF signals are amplified by a low noise preamplifier (preamp) which effectively sets the noise figure of the receiver. There may be a passive bandpass prefilter between the antenna and preamp to minimize out-of-band RF interference. These amplified and signal conditioned RF signals are then down-converted to an intermediate frequency (IF) using signal mixing frequencies from local oscillators (LOs). The LOs are derived from the reference oscillator by the frequency synthesizer based on the frequency plan of the receiver design. One LO per down converter stage is required. Two-stage down conversion to IF is typical, but one-stage down conversion and even direct L-band digital sampling have also been used. The LO signal mixing process generates both upper and lower sidebands of the SV signals, so the lower sidebands are selected and the upper sidebands and leak-through signals are rejected by a postmixer bandpass filter. The signal Dopplers and the PRN codes are preserved after the mixing process. Only the carrier frequency is lowered. The analog-to-digital (A/D) conversion process and automatic gain control (AGC) functions take place at IF. Not shown in the block diagram are the baseband timing signals that are provided to the digital receiver channels by the frequency synthesizer phase locked to the reference oscillator's stable frequency. The IF must be high enough to provide a single-sided bandwidth that will support the PRN code chipping frequency. An anti-aliasing IF filter must suppress the stopband noise (unwanted out-of-band signals) to levels that are acceptably low when this noise is aliased into the GPS signal passband by the A/D conversion process. All of the visible GPS signals are buried in the thermal noise at IF.

At this point the digitized IF signals are ready to be processed by each of the N digital receiver channels. No demodulation has taken place, only signal conditioning and conversion to the digital IF. These digital receiver channel functions are usually implemented in one or more application specific integrated circuits (ASICs). This is why these functions are shown as separate from the receiver processing function in the block diagram of the figure below. The name "digital receiver channel" is somewhat misleading since it is not the ASIC but the receiver processing function which usually implements numerous baseband functions such as the loop discriminators and filters, data demodulation, C/N_0 meters, phase lock indicators, etc. The receiver processing function is usually a microprocessor. The microprocessor not only performs the baseband functions, but also the decision-making functions associated with controlling the signal preprocessing functions of each digital receiver channel. It is common that a single high-speed microprocessor supports the receiver, navigation and user interface functions.

Generic digital receiver block diagram



Generic digital receiver block diagram – A/D converter

The generic digital receiver block diagram shown below has the analog-to-digital (A/D) converter framed to illustrate where this function takes place. Note that the A/D converter changes the analog intermediate frequency (IF) into a digital IF. Also note that only one A/D converter is required per receiver L-band front-end down-converter. This functional block diagram depicts only one L-band frequency down conversion. If this were a two-frequency receiver, there would be two L-band L-band down-converters and two A/D converters, one for each frequency. There might be only one wideband antenna with a frequency splitter or there might be two separate antennas under the same radome with separate outputs. In either case an antenna is required that is a right-hand circularly polarized antenna capable of receiving the full bandwidth of the global navigation satellite system (GNSS) signals that are replicated and tracked by the digital receiver channels. Note that each L-band carrier is capable of supporting a multiplicity of GNSS pseudo random noise (PRN) codes. For example, the modernized GPS III satellites will provide C/A code, P(Y) code, L1C code and M code all on the L1 carrier.

The L1C code is of particular future interest so some additional information is provided here about this recognized “international” signal. When the U.S. Air Force released the initial draft of Interface Specification IS-GPS-800, describing L1C and the novel characteristics of the optimized L1C signal design, this signal became the international standard to provide advanced capabilities while offering to receiver designers considerable flexibility in how to use these capabilities.

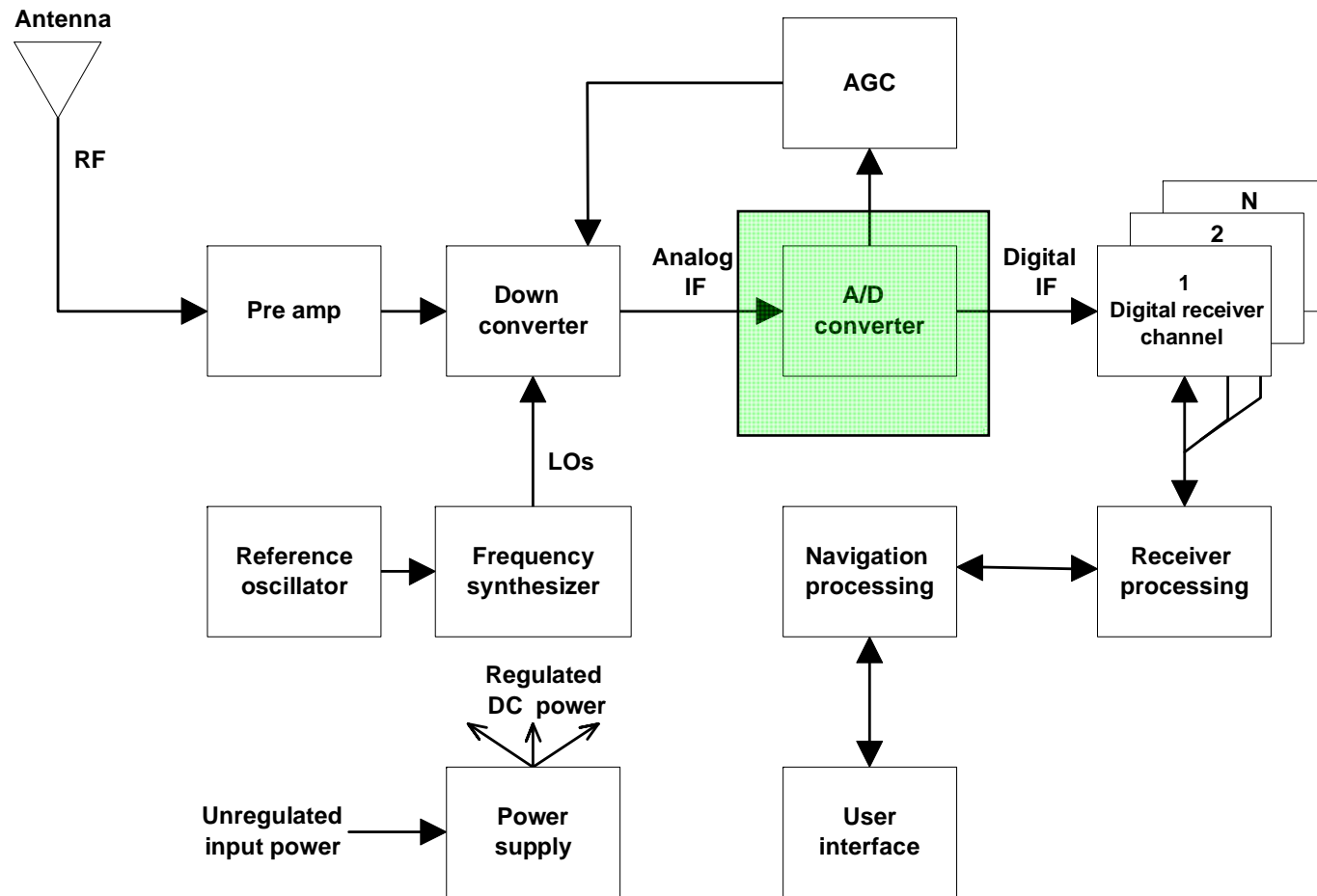
The development of L1C represents a new stage in international GNSS: not only is the signal being designed for transmission from GPS satellites, its design also seeks to maximize interoperability with Galileo’s Open Service signal. Further, Japan’s Quasi-Zenith Satellite System (QZSS) will transmit a signal with virtually the same design as L1C.

L1C has been designed to take advantage of many unique opportunities. Its center frequency of 1575.42 MHz is the pre-eminent GNSS frequency for a variety of reasons, including the extensive existing use of GPS L1 C/A code, the lower ionospheric error at L1 band relative to lower frequencies, spectrum protection of the L1 band, and the use of this same center frequency by GPS, Galileo, QZSS, and satellite-based augmentation system (SBAS) signals for open access service and safety-of-life applications.

Other unique opportunities that the L1C design leverages include advances in signal design knowledge, improvements in receiver processing techniques, developments in circuit technologies, and enhancements in supporting services such as communications. The L1C design has been optimized to provide superior performance, while providing compatibility and interoperability with other signals in the L1 band.

L1C provides a number of advanced features, including: 75 percent of power in a pilot component for enhanced signal tracking, advanced Weil-based spreading codes, an overlay code on the pilot that provides data message synchronization, support for improved reading of clock and ephemeris by combining message symbols across messages, advanced forward error control coding, and data symbol interleaving to combat fading. It also provides much more resistance to continuous wave (CW) interference than does the L1 C/A code.

Generic digital receiver block diagram – A/D converter



Analog-to-digital (A/D) converter

The analog-to-digital (A/D) converter performs two processes on the incoming analog signal: (1) sampling and (2) quantization.

The sampling process must be chosen by the design engineer to be consistent with the Nyquist criteria. Nyquist proved that if there is at least two samples per cycle at the frequency above which there is no data (including noise), then no information is lost in the sampling process. The adverse consequence of disobeying this criteria is that any analog information that is present above the Nyquist frequency will be aliased (folded) back into the sampled data that can never be removed by any subsequent digital signal processing. Long ago, Norbert Wiener proved that the Nyquist criteria is theoretically impossible to achieve, but practically speaking, it is possible to design anti-aliasing filters that suppress the unwanted signals, including noise, to a sufficiently low level that the aliasing that results contributes negligible distortions.

The quantization level is usually the focus of design attention (often to the neglect of the Nyquist sampling theorem). Earlier versions of GPS receivers used analog correlators when digital technology was not fast enough to A/D convert and process the IF signals at appropriate sampling rates. These A/D converters were implemented at baseband just after the code wipeoff process with analog anti-alias filtering that also provided a portion of the predetection integration process. Typically, these were 8-bit A/D converters (256 quantization levels) and a multiplicity of them were required for each receiver channel. These are called post-correlation A/D converters and the receivers designed in this manner were called analog receivers. Currently, all GNSS receiver designs employ the A/D converter in the location shown in the generic GPS receiver design. These are called pre-correlation A/D converters and the receivers are called digital receivers. The vast majority of these A/D converters are 1, 2 and 3-bit converters with 2, 4 and 8 levels of precision, respectively. There is little to be gained with quantization precision above 3-bits provided that the intended operational environment is benign (no RF interference (RF)) or jamming).

For RFI or jamming environments, very high resolution A/D conversion is required, typically 10 to 12 bits or more, in order to implement RFI or jamming mitigation either with spectral excision for narrowband emitters or with a controlled reception pattern antenna (CRPA) for wideband emitters. The CRPA uses a multiple antenna element array that requires a separate RF front end per element. Each RF front end contains a high-quantization precision A/D converter.

A low-cost non-linear three-level flash A/D converter (ADC) can be very effective in some RFI environments [1]. When used in combination with a digital gain controlled automatic gain control (AGC) design, this ADC can effectively suppress continuous wave (CW) narrow-band jammers. It can also detect the presence of any RFI, then measure and characterize it as either narrowband or wideband. This part of the GNSS receiver continues to operate effectively even when the interference level is so high that the receiver cannot acquire and track the GNSS signals.

[1] Ward, P., "Simple Techniques for RFI Situational Awareness and Characterization in GNSS Receivers," Proceedings of The Institute of Navigation National Technical Meeting 2008, San Diego, CA, January 28-30.

Analog-to-digital (A/D) conversion

- Most receivers today digitize after down-conversion and prior to correlation
 - Post-correlation A/D was norm long ago
- Vast majority of commercial receivers use between 1-3 bits (2 - 8 output levels)
 - In benign environments, no gain from additional levels
 - For interference excision (anti-jam) applications, 10 to 12 bit A/D converters are required
 - For continuous wave (CW) interference mitigation, non-linear 1.5 bit A/D converters are very effective

GPS receiver code and carrier tracking

Most modern GPS receiver designs are digital receivers; i.e., they are pre-correlation A/D receiver designs where each L-band signal is digitized at IF. The intent of digital receiver design is to move the digital component design as close toward the antenna analog signal input as possible because digital technology has evolved much faster than analog technology in recent years. As a result, digital receivers have evolved rapidly toward higher and higher levels of digital component integration. The remaining analog components are the antenna, the RF front end, A/D converter, reference oscillators, power supplies, and user visual interfaces such as displays and lighted indicators. These analog technologies have also evolved rapidly. The most rapid technological evolution and subsequent cost reduction has been in the area of embedded GPS technology in mobile phones. There are more than 500 million L1 C/A code digital receivers currently in use in mobile phones and the number is increasing daily. Most of these have been implemented on a single chip including the RF front end and A/D converter.

The generic digital receiver block diagram shown earlier is representative of the designs of both commercial and military digital receivers. The GPS receiver code and carrier tracking processes will be explained in the context of this generic block diagram and subsequent expansions. There will be similarities and differences between this generic design and other digital receivers. The differences are often due to the origin of the design; i.e., the functional partitioning of the original design was driven in large part by a different situation in available parts at the time as well as the expected market size for the end product. Other differences are due to the evolution of the design. In both cases the design is driven in large part by component cost. Even though high level integration can reduce parts count without reducing the design complexity, there are usually constraints at every technology generation level that force changes to the original architecture. Especially in commercial GNSS receiver design, there is a significant emphasis on reducing design complexity in order to reduce parts count. Many commercial GPS receiver manufacturers have found clever ways to greatly simplify their GPS receiver designs, but there is usually a performance penalty when the functions shown in this generic architecture are altered excessively.

GPS receiver code and carrier tracking

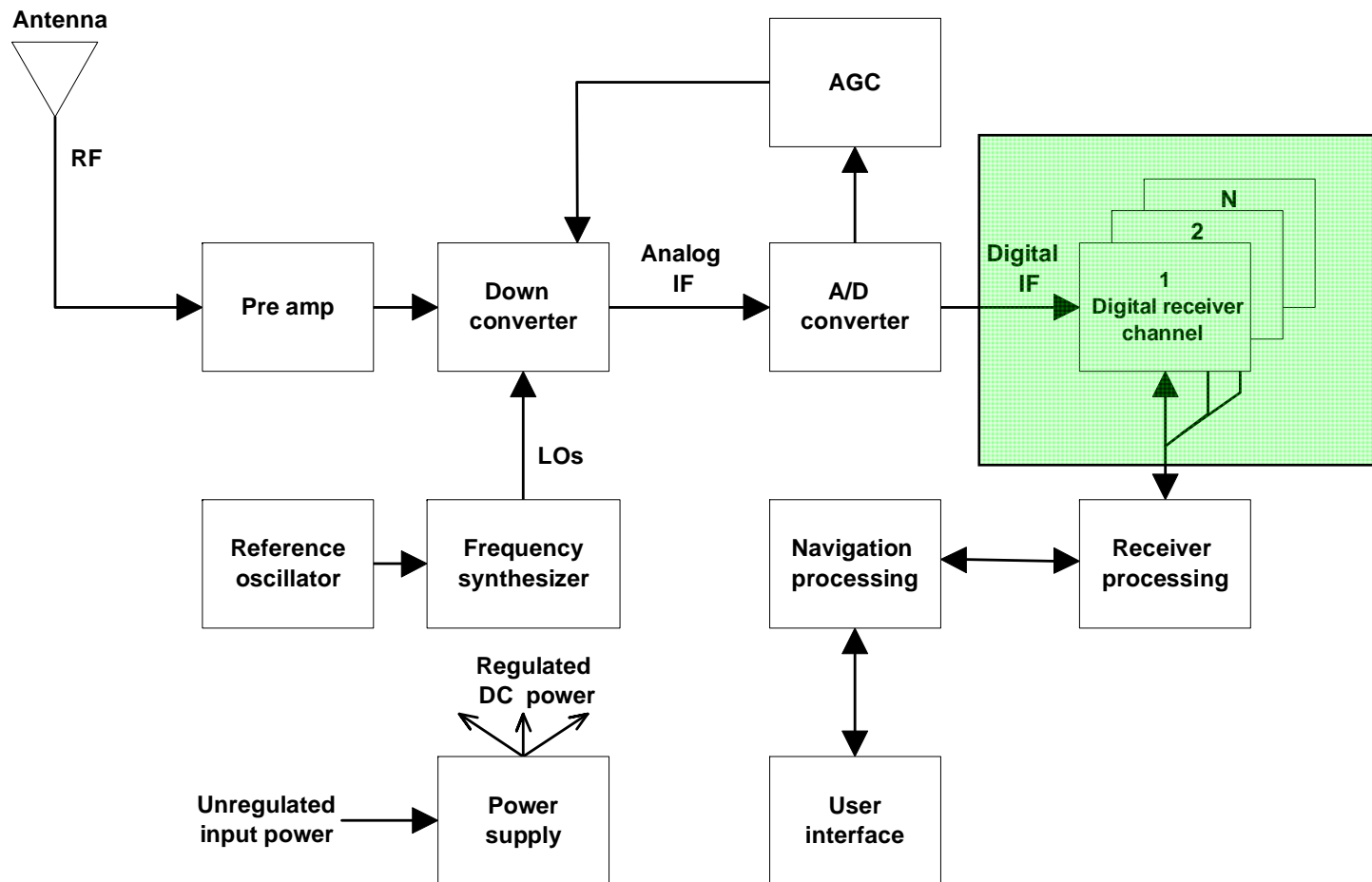
- Most modern GPS receiver designs are digital (signal is digitized prior to detection)
- Receiver designs have evolved rapidly toward higher levels of digital component integration
- Modern GPS receiver is represented by a generic digital GPS receiver architecture
- GPS receiver code and carrier tracking explained in context of this generic digital architecture
- Modern digital receivers similar to generic architecture but designs driven by origin/evolution/cost
 - Performance penalties result if cost reduction focus is primarily on component reduction instead of technology advances

Generic digital receiver block diagram – Digital receiver channels

The generic digital receiver block diagram shown below frames 1 through N digital receiver channels, where N is the number of channels for this frequency. This illustrates how multiple digital receiver channels are all connected to the the same digital IF signal but each one extracts its own PRN code satellite signal that is in view of the antenna sky coverage.

Since every channel is capable of tracking any satellite PRN signal, all channels are identical to each other. The only difference is the PRN code that they happen to be tracking and this is chosen by the receiver processor which assigns the PRN number to each channel. For this reason, only one digital receiver channel is described.

Generic digital receiver block diagram – Digital receiver channels



Digital receiver channel block diagram

The digital receiver channel block diagram represents one channel of the receiver where the digitized intermediate frequency (IF) signals are applied to the input. For simplification, only the functions associated with the code and carrier tracking loops are illustrated and the receiver channel is assumed to be tracking the SV signal in steady state. First, the digital IF signal of the SV being tracked is stripped of the carrier (plus carrier Doppler) by the replica carrier (plus carrier Doppler) signal to produce in-phase (I) and quadrature (Q) signals. Note that the replica carrier signal is being mixed with all of the GPS SV signals (buried in noise) at the digital IF. Also note that this is the first digital process associated with the sampled and quantized analog IF data; i.e., A/D converted IF data. The I and Q signals at the outputs of the mixers have the desired phase relationships with respect to the detected carrier of the desired SV. However, the code stripping processes which collapse these signals to baseband have not yet been applied. Therefore, the I signal at the output of the in-phase mixer would be mostly thermal noise multiplied by the replica digital sine wave (to match the digitized SV carrier at IF) and the Q signal at the output of the quadrature mixer would be the product of mostly thermal noise and the replica digital cosine wave (to match the digitized SV carrier at IF). The desired SV signal remains buried in noise until the I and Q signals are collapsed to baseband by the code stripping process which follows. The replica carrier (including carrier Doppler) signals are synthesized by the carrier numerical controlled oscillator (NCO) in combination with the discrete sine and cosine mapping functions.

The NCO generates an internal digital staircase function whose period is the desired replica carrier plus Doppler period. The sine and cosine map functions convert the NCO internal digital staircase amplitudes into the corresponding digital amplitudes of the respective sine and cosine functions. By producing I and Q component phases 90 degrees apart, the resultant signal amplitude can be computed from the vector sum of the I and Q components and the phase angle with respect to the I-axis can be determined from the arctangent of Q/I. In closed loop operation, the carrier NCO is controlled by the carrier tracking loop in the receiver processor. In phase lock loop (PLL) operation, the objective of the carrier tracking loop is to keep the phase error between the replica carrier and the incoming SV carrier signals at zero. Any misalignment in the replica carrier phase with respect to the incoming SV signal carrier phase is detected in the receiver processor as a non-zero error in the phase angle of the prompt I and Q vector magnitude, then corrected in the carrier NCO output so the I signals are maximum and the Q signals are nearly zero.

The I and Q signals are then correlated with early, prompt and late replica codes (plus code Doppler) synthesized by the code generator, a 2-bit shift register, and the code NCO. These six correlated outputs are then filtered by integrate and dump circuits, then fed to the receiver processor. In closed loop operation, the code NCO is controlled by the code tracking loop in the receiver processor. In this design, the code NCO produces twice the code generator clocking rate, $2f_{co}$, and this is fed into the 2-bit shift register. The code generator clocking rate, f_{co} , which contains the code chipping rate (plus code Doppler) is fed into the code generator. With this combination, the shift register produces three phases of the code generator output, each phase shifted by $\frac{1}{2}$ chip apart. Not shown are the controls to the code generator which permit the receiver processor to preset the initial code tracking phase states which are required during the code search and acquisition process.

The prompt replica code phase is aligned with the incoming SV code phase producing maximum correlation if it is tracking the incoming SV code phase. In this design, the early phase is aligned $\frac{1}{2}$ chip early and the late phase is aligned $\frac{1}{2}$ chip late with respect to the incoming SV code phase and these correlations produce about half the maximum correlation. Any misalignment in the replica code phase with respect to the incoming SV code phase produces a difference in the vector magnitudes of the early and late correlated outputs so that the amount and direction of the phase change can be detected and corrected by the code tracking loop.

Digital receiver channel block diagram – Code and carrier tracking loops

The digital receiver channel block diagram shown below frames the receiver processor function. All of the sophisticated, but fortunately much less stringent computationally, processes of each digital receiver channel are performed by the receiver processor.

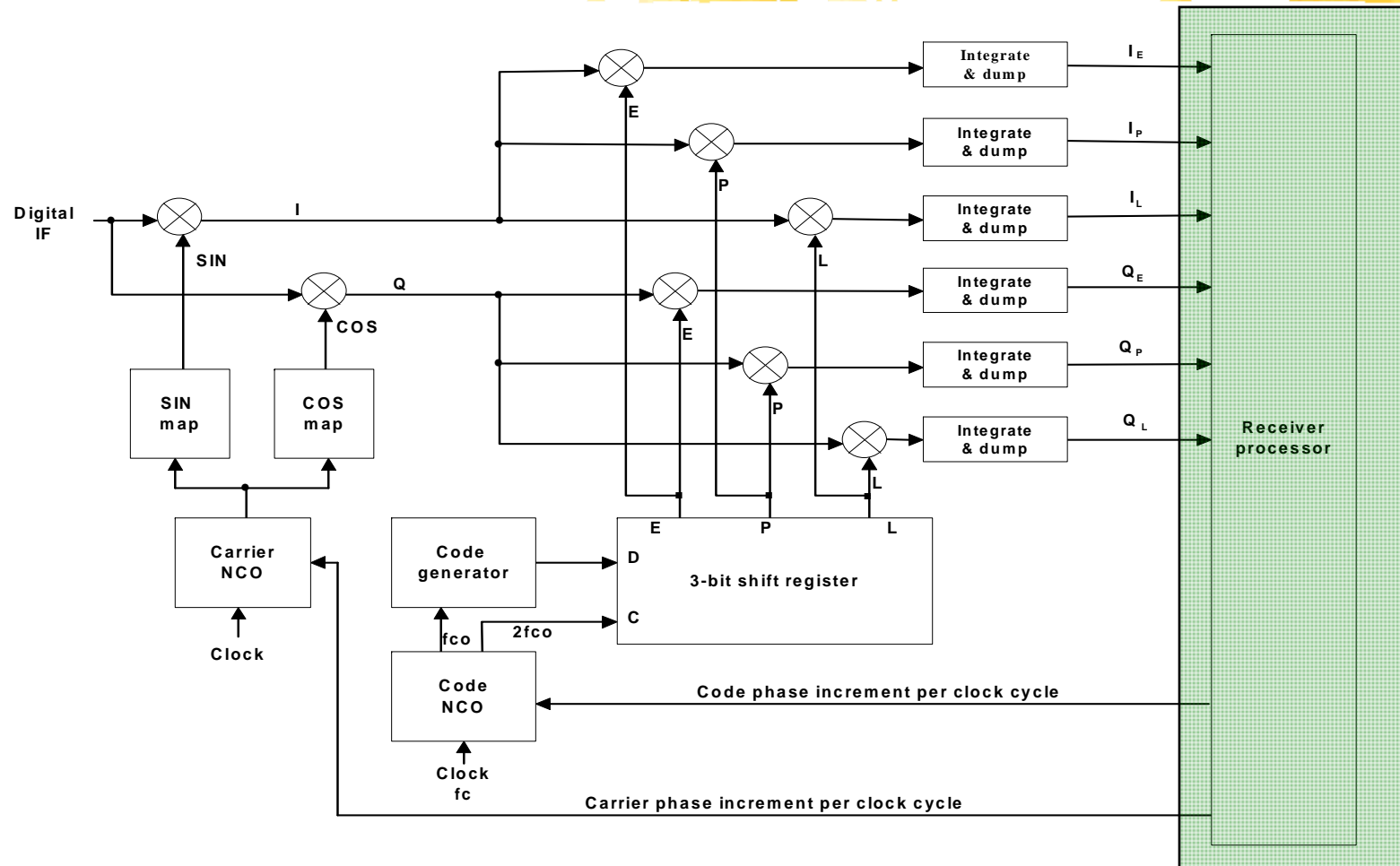
There are several noteworthy features. First, there are six I and Q signals shown as the digital receiver channel inputs to the receiver processor. Each I and Q pair is called a complex I,Q pair or simply a complex input. So there are 3 complex inputs to the receiver processor: an early, prompt and late set. In fact, every digital receiver channel provides a unique set of three complex inputs to the receiver processor. Inside the receiver processor, there is one re-entrant baseband firmware program that can process N channels of these complex pairs as N code and N carrier tracking loops, where N is the number of digital receiver channels. The code tracking loop processes the early and late complex pairs and the carrier tracking loop processes the prompt complex pair. There are two feedback outputs per channel from these two tracking loops: the code loop provides feedback to the code NCO (called "code phase increment per clock" in the diagram) and the carrier loop provides feedback to the carrier NCO (called "carrier phase increment per clock" in the diagram).

There are a multiplicity of other signals required to control these processes but are not shown in the illustration. For example, the receiver processor interrupt signal from each receiver channel when it has stored its receiver processor data into the memory of the receiver processor. There is a fundamental time frame (FTF) signal derived from the reference oscillator that synchronizes the receiver processor functions to the receiver analog front-end to provide receiver time reference epochs. This is typically a 20 ms interrupt and is sometimes called set time. There are many other interrupt signals required, but these are the most important ones. There are arming signals that synchronize the receiver processor feedback signals with the clock signals that latch them in their respective NCO circuits. There are other control signals to perform built-in test, etc.

There is a class of GNSS receiver called a software defined receiver (SDR) in which the functions of the N digital hardware receiver channels are performed in software. Modern digital hardware receiver designs use an application specific integrated circuit (ASIC). The benefit of the SDR is that when new signals are defined or to improve or correct a latent defect in the SDR, a new firmware release defines the new receiver. In the case of an ASIC, a new (and very expensive) ASIC design is required. However, the downside of the SDR is that the ASIC processes that are replaced require many orders of magnitude more processing throughput than the tasks performed in the receiver processor design shown below. The classic digital receiver hardware functions are those that are relatively simple in design but very intensive computationally. The hardware ASIC approach solves this problem because these functions are identical in each channel and the ASIC can perform these functions in parallel or recursively in series if the ASIC is fast enough.

For the L1 C/A code receiver design example used in this course, the hardware A/D converter operates at 4 to 17 MHz sample rates. The SDR carrier NCO, replica carrier mapping and wipe-off functions have to operate at the same rate as the A/D converter sample rate and the code NCO and code replica generator functions have to operate at 1.023 MHz plus or minus code Doppler rates and the code wipe-off functions must operate at the carrier wipe-off rate. The integrate and dump functions must also operate at the code wipe-off rate. Finally, all of these functions per channel operate sequentially in the SDR, not in parallel as is the case for an ASIC and there are N channels which multiply this effective processing rate by N. Consider that the fastest receiver processing rate in this hardware based block diagram is 1 KHz times N channels. This allows 1 ms per channel to complete the receiver processing before the next interrupt arrives. Compare this to a 4 MHz interrupt rate for a 12 channel L1 C/A code receiver. There have been many SDR developments that were "poor performers" because of underestimated processing resources. The only really capable ones of recent vintage are those that have used field program able gate arrays (FPGAs) instead of ASICs to provide the very desirable SDR features. These are typically more expensive than the classic ASIC designs.

Digital receiver channel block diagram – code and carrier tracking loops

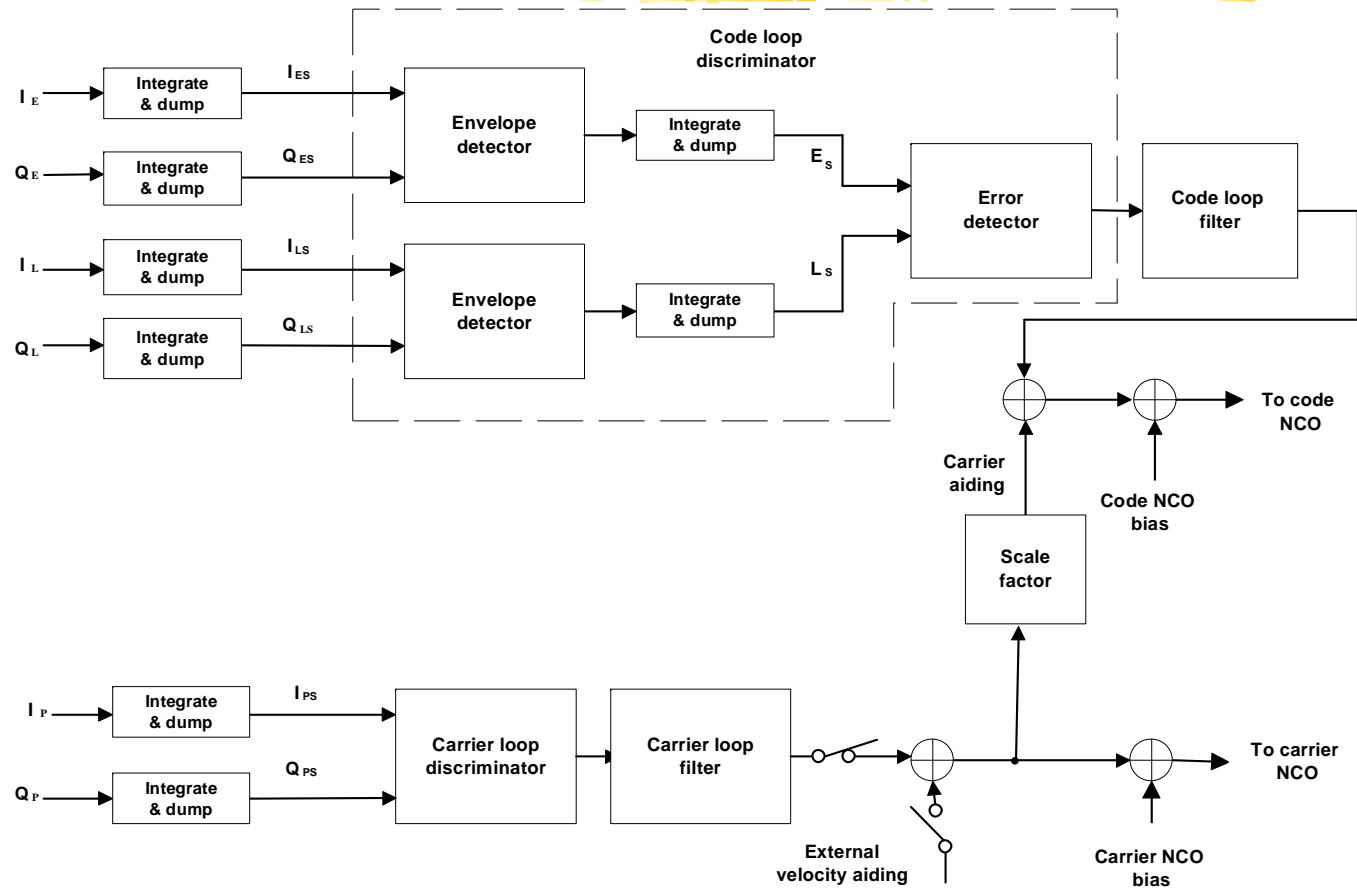


Baseband signal processing for code and carrier tracking loops

The receiver processor shown in the previous figure receives the I and Q signals from N digital receiver channels, but only one channel is illustrated. These signals are called baseband signals because the carrier has been removed by the carrier wipe-off process and the spread spectrum bandwidth has been collapsed to baseband by the code correlation process. The only remaining signal present is the data modulation of the 50 Hz navigation data message.

The receiver processor implements code and carrier tracking loops to process these baseband signals which determine the error between the replica code and carrier signals and the incoming code and carrier signals. These code and carrier error signals are fed back to the code and carrier replica generation circuits. The figure below provides more detail on the code and carrier tracking loops that are implemented in the receiver processor. .

Baseband processor code and carrier tracking loops block diagram



Baseband signal processing

The previous figure illustrates typical baseband code and carrier signal processing functions for one receiver channel in the closed loop mode of operation. These functions are typically performed by the receiver processor. The combination of these code and carrier baseband tracking functions and the digital receiver channel replica code and replica carrier generation, wipe-off and integrate and dump functions form the code and carrier tracking loops on one GPS receiver channel. The integrate and dump functions are also called predetection integration functions because they are indeed integrators and they precede the code and carrier error detection functions performed in the receiver processor. receiver processor shown in the previous figure receives the I and Q signals from N digital receiver channels, but only one channel is illustrated. These signals are called baseband signals because the carrier has been removed by the carrier wipe-off process and the spread spectrum bandwidth has been collapsed to baseband by the code correlation process. The only remaining signal present is the data modulation of the 50 Hz navigation data message.

The baseband signal processing functions in the receiver processor are usually implemented in firmware. Note that the firmware need only be written once since the microprocessor runs all programs sequentially (as opposed to the simultaneous parallel processing that takes place in the hardware digital receiver channels). Therefore, the microprocessor program can be designed to be re-entrant with a unique variable memory, also called read-alterable memory (RAM) for each receiver channel so that only one copy of each firmware algorithm is required to service all N receiver channels. This reduces the firmware memory, also called programmable read-only memory (PROM) requirements. This is the optimum use of RAM and PROM in the receiver process memory. It also ensures that every receiver baseband process is identical for each receiver channel receiver processor implements code and carrier tracking loops to process these baseband signals which determine the error between the replica code and carrier signals and the incoming code and carrier signals. These code and carrier error signals are fed back to the code and carrier replica generation circuits. The figure below provides more detail on the code and carrier tracking loops that are implemented in the receiver processor. As illustrated, the three complex pairs may be integrated and dumped for an additional period of time prior to the detection process. For example, if the receiver hardware channels send baseband signals to the receiver processor every 1 ms, then they have already been integrated and dumped for that period of time. Since the GPS L1 C/A code contains 50 Hz data and that data modulation is still on the I and Q baseband signals, then the receiver processor will typically integrate and dump these signals up to 5 more times to increase the predetection integration time to 5 ms when the receiver does not yet know where the data bit boundaries are for this 20 ms data. When a data bit changes, this reverse the phases of the I and Q signals 180 degrees.

A data bit transition half way between a 5 ms integrate and dump process causes that noisy integrated and dumped result to go to approximately zero but the following three results are assured to be transition-free and have full signal strength. (Note that odd predetection integration times are desirable at or above 10 ms when the data bit boundaries are unknown because this randomizes the locations of the "dump" boundaries with respect to the unknown incoming data bit boundaries.)

After the bit synchronization process has taken place, the predetection integration time is typically increased to 20 ms in high performance receivers, since the receiver processor now knows which 1 ms samples contain the data bit boundary. (Note that since the satellites are typically at different ranges with respect to the receiver antenna phase center, their data bit boundaries are typically NOT aligned with each other at the receiver even though they are almost perfectly aligned to GPS time when they emerge from their respective satellites.)

Baseband signal processing



- Baseband signal processing functions are typically performed by the receiver processor
- Software code/carrier tracking functions at baseband providing feedback correction errors to the code/carrier wipe-off hardware followed by predetection integration functions form code/carrier tracking loops of one receiver channel
- Digital receiver channel hardware run in parallel
- Microprocessor runs all baseband signal processing programs sequentially

Digital receiver channel – Code generator

The replica code generator is framed in the digital receiver channel functional block diagram illustrated below. There is (eventually) a satellite interface control document (ICD) for every operational or planned GNSS signal. All interface requirements are described in the relevant ICD. The ICD that describes how to design a compatible GPS L1 C/A code replica code generator is the latest version of: GPS Joint Program Office. 1997. *ICD-GPS-200: GPS Interface Control Document*. ARINC Research.

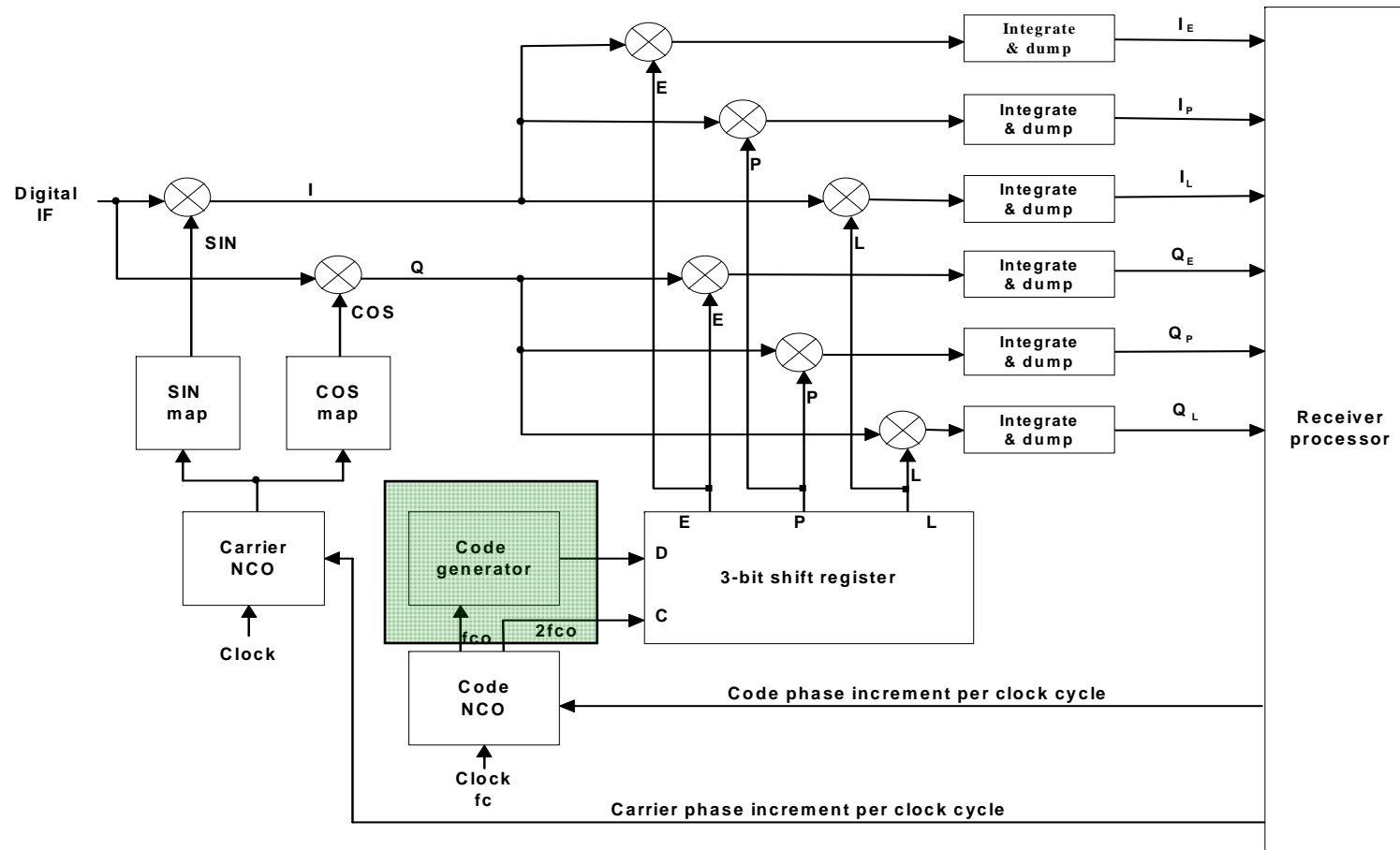
This is available as a PDF document on line from the U.S. Coast Guard website. To obtain the current version of ICD-200c visit this website at:

<http://www.navcen.uscg.gov/GPS/ICD200c.htm>

Select the line at the bottom which reads:

Download the [ICD200c \(includes IRN-200C-001 thru IRN-200C-004\)](#) from NAVCEN'S Website. (PDF 516KB)

Digital receiver channel block diagram – code generator



C/A code generator

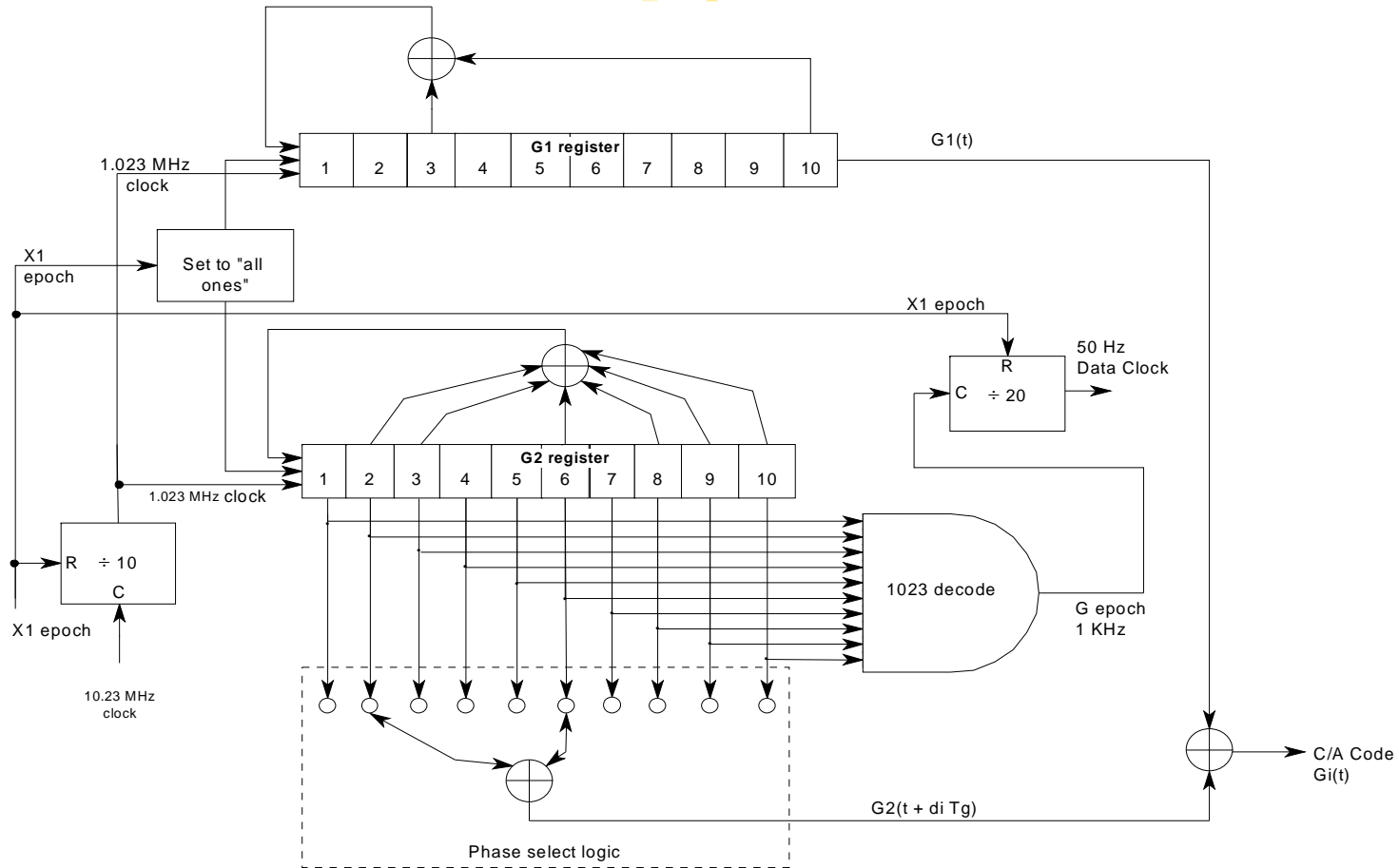
The GPS C/A code is a Gold code (named after its inventor Robert Gold) with a sequence length of 1023 chips. (One chip is one non-information bearing bit.) Since the chipping rate of the C/A code is 1.023 million chips per second (Mcps), then the repetition period of the C/A code pseudo random noise (PRN) sequence is the sequence length (1023) in chips divided by the chipping rate (1.023×10^6 chips/s) or 1 millisecond (ms).

The figure below illustrates the design architecture of the GPS C/A code generator. Not included in this diagram are the controls necessary to set or read the code phase states of the shift registers or the counters. Note that two 10-bit shift registers are required to synthesize the C/A code. These are called the G1 and G2 registers. These both generate maximum length pseudo noise (PN) codes with a length of $2^{10} - 1$ bits = $1024 - 1 = 1023$ bits. The one unused state that these shift registers must not get into is the all-zero state!

It is common to describe the design of linear code generators by means of polynomials of the form $1 + x^i$, where x^i means that the output of the i th cell of the shift register is used as an input to a modulo-2 adder and the 1 means that the output of the adder is fed to the first cell. The design specification for C/A code calls for the feedback taps of the G1 shift register to be connected to stages 3 and 10. These register states are combined with each other by an exclusive-or circuit and fed back to stage 1. The polynomial that describes this shift register architecture is: $G1 = 1 + X^3 + X^{10}$. The much longer polynomial that describes the G2 shift register architecture is: $G2 = 1 + X^2 + X^3 + X^6 + X^8 + X^9 + X^{10}$.

The unique C/A code for each SV is the result of a delayed version of the G2 output sequence and the G1 direct output sequence. The delay effect in the G2 PN code is obtained by combining the output of two G2 stages by modulo-2 addition; i.e., the exclusive-or of the outputs of two pre-defined tap positions on the G2 shift register. These taps and their effective G2 delay are described in a following table.

C/A code generator



Code phase assignments and initial code sequences for C/A-code and P-code

The delays associated with the C/A code and P-code PRN sequences are summarized in the table below taken from ICD-GPS-200. The C/A code delay can be implemented by a simple, but equivalent, technique that eliminates the need for a delay register. The C/A code tap selection column describes the two-tap combinations associated with each SV PRN number. This two-tap delay technique used for the C/A code generator is explained in more detail when the C/A code generator is described.

Note that the P code delay in chips is identical to its PRN number, but that the C/A-code delay does not bear a mathematical relationship to the PRN number (either the two-tap combination or the delay must be obtained by table look-up). Also note that the first 10 C/A-chips from the beginning of each C/A-code epoch (every 1 ms) and the first 12 P-chips from the beginning of the GPS week are shown in octal notation in this table. The octal notation is explained both here and in the footnotes. In the octal notation for the first 10 chips of the C/A code, the first digit represents a "1" for the first chip and the last three digits are the conventional octal representation of the remaining nine chips. For example, the first 10 chips of the SV PRN number 1 L1 C/A code are 110010000.

The footnotes also describe that PRN codes 33 through 37 are reserved for other uses; e.g., ground transmitters (GTs). These were the GT PRN codes used for the Yuma Proving Ground inverted range in the early test phases of the GPS concept demonstration program that started before the first Block I SVs were launched. Note that C/A codes for PRN 34 and PRN 37 are identical GT C/A codes, but are different P codes. This was because no more two-tap C/A code combinations with the desirable cross correlation properties were left. However, numerous other C/A code combinations with acceptable properties are available. These are used in the FAA Wide Area Augmentation System (WAAS) for the Geostationary C/A code only satellites. The WAAS C/A code generator requires a design extension in the two-tap scheme.

Code phase assignments and initial code sequences for C/A- and P-code

SV PRN number	C/A-code tap selection	C/A-code delay (chips)	P-code delay (chips)	First 10 C/A-chips(octal)*	First 12 P-chips (octal)*
1	2⊕6	5	1	1440	4444
2	3⊕7	6	2	1620	4000
3	4⊕8	7	3	1710	4222
4	5⊕9	8	4	1744	4333
5	1⊕9	17	5	1133	4377
6	2⊕10	18	6	1455	4355
7	1⊕8	139	7	1131	4344
8	2⊕9	140	8	1454	4340
9	3⊕10	141	9	1626	4342
10	2⊕3	251	10	1504	4343
11	3⊕4	252	11	1642	"
12	5⊕6	254	12	1750	"
13	6⊕7	255	13	1764	"
14	7⊕8	256	14	1772	"
15	8⊕9	257	15	1775	"
16	9⊕10	258	16	1776	"
17	1⊕4	469	17	1156	"
18	2⊕5	470	18	1467	"
19	3⊕6	471	19	1633	"
20	4⊕7	472	20	1715	"
21	5⊕8	473	21	1746	"
22	6⊕9	474	22	1763	"
23	1⊕3	509	23	1063	"
24	4⊕6	512	24	1706	"
25	5⊕7	513	25	1743	"
26	6⊕8	514	26	1761	"
27	7⊕9	515	27	1770	"
28	8⊕10	516	28	1774	"
29	1⊕6	859	29	1127	"
30	2⊕7	860	30	1453	"
31	3⊕8	861	31	1625	"
32	4⊕9	862	32	1712	"
33**	5⊕10	863	33	1745	"
34**	4⊕10***	950***	34	1713***	"
35**	1⊕7	947	35	1134	"
36**	2⊕8	948	36	1456	"
37**	4⊕10***	950***	37	1713***	4343

* In the octal notation for the first 10 chips of the C/A-code as shown in this column, the first digit (1) represents a "1" for the first chip and the last three digits are the conventional octal representation of the remaining 9 chips. For example, the first 10 chips of the SV PRN number 1 C/A-code are 1100100000. **PRN codes 33 through 37 are reserved for other uses; e.g., ground transmitters. ***C/A-codes 34 and 37 are identical.

C/A code generator design: Polynomials and initial states

The table below summarizes the polynomial definitions for both C/A codes and P codes. The initial states are presented as a help for the designer of these code generators to determine if the sequence actually begins correctly. Both are taken from ICD-GPS-200, the GPS interface control document for these two codes. P codes will not be elaborated on in this presentation, but they are included when the information is combined in ICD-GPS-200 tables.

C/A code generator design: polynomials and initial states

Register	Polynomial	Initial State
C/A-code G1	$1 + X^3 + X^{10}$	1111111111
C/A-code G2	$1 + X^2 + X^3 + X^6 + X^8 + X^9 + X^{10}$	1111111111
P-code X1A	$1 + X^6 + X^8 + X^{11} + X^{12}$	001001001000
P-code X1B	$1 + X^1 + X^2 + X^5 + X^8 + X^9 + X^{10} + X^{11} + X^{12}$	010101010100
P-code X2A	$1 + X^1 + X^3 + X^4 + X^5 + X^7 + X^8 + X^9 + X^{10} + X^{11} + X^{12}$	100100100101
P-code X2B	$1 + X^2 + X^3 + X^4 + X^8 + X^9 + X^{12}$	010101010100

Session II



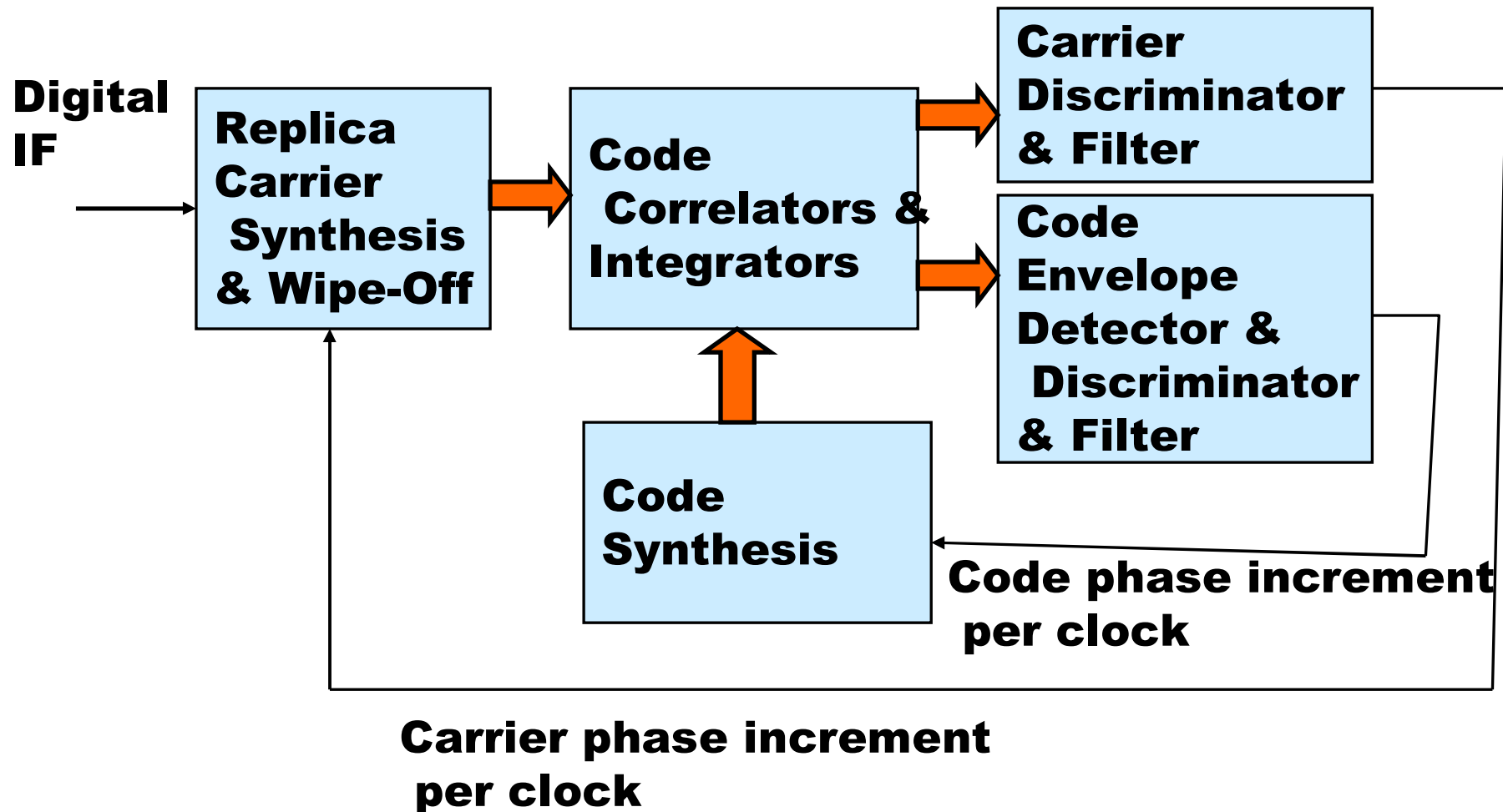
Session II

GPS receiver baseband processes

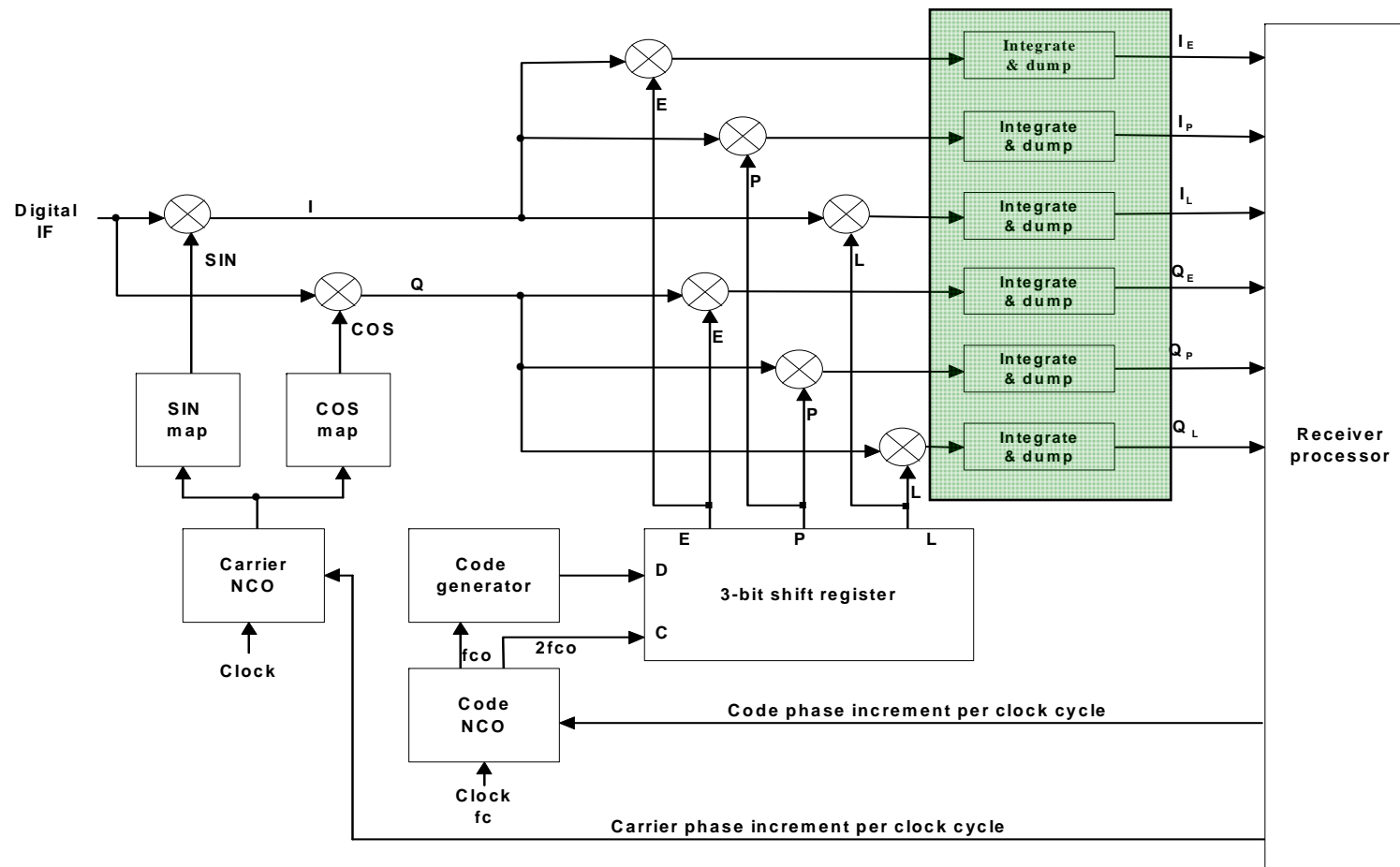


- Baseband signal processes block diagram
- Predetection integration: dealing with GPS data transition boundaries
- Carrier aiding of code loop: scale factors for carrier aided code
- External aiding: where and how it is injected in each tracking loop
- Digital frequency synthesizer block diagram and output waveforms
- Digital frequency synthesizer design

Baseband signal processes block diagram



Digital receiver channel block diagram – predetection integration



Predetection integration

Predetection is the signal processing after the IF signal has been converted to baseband by the carrier and code stripping processes, but prior to being passed through a signal discriminator; i.e., prior to the non-linear signal detection process. Extensive digital predetection integration and dump processes occur after the carrier and code stripping processes. This causes very large numbers to accumulate even though the IF A/D conversion process is typically with only one to three bits of quantization resolution.

The generic digital receiver channel block diagram shows three complex correlators required to produce three in-phase components which are integrated and dumped to produce I_E , I_P , I_L and three quadrature-phase components integrated and dumped to produce Q_E , Q_P , Q_L . The carrier wipe-off and code wipe-off processes must be performed at the digital IF sample rate which is around 5 MHz for C/A-code and 50 MHz for P(Y)-code. The integrate and dump accumulators provide filtering and resampling at the processor baseband input rate which is around 200 Hz (which can be higher or lower depending on the desired predetection bandwidth). The 200 Hz rate is well within the interrupt servicing rate of modern high-speed microprocessors, but the 5 or 50 MHz rates would not be manageable. This further explains why the high speed, but simple processes are implemented in a custom digital ASIC while the low speed, but complex processes are implemented in a microprocessor.

The hardware integrate and dump process in combination with the baseband signal processing integrate and dump process (described below) defines the predetection integration time. The predetection integration time is a compromise design. It must be as long as possible to operate under weak or RF interference signal conditions and it must be as short as possible to operate under high dynamic stress signal conditions.

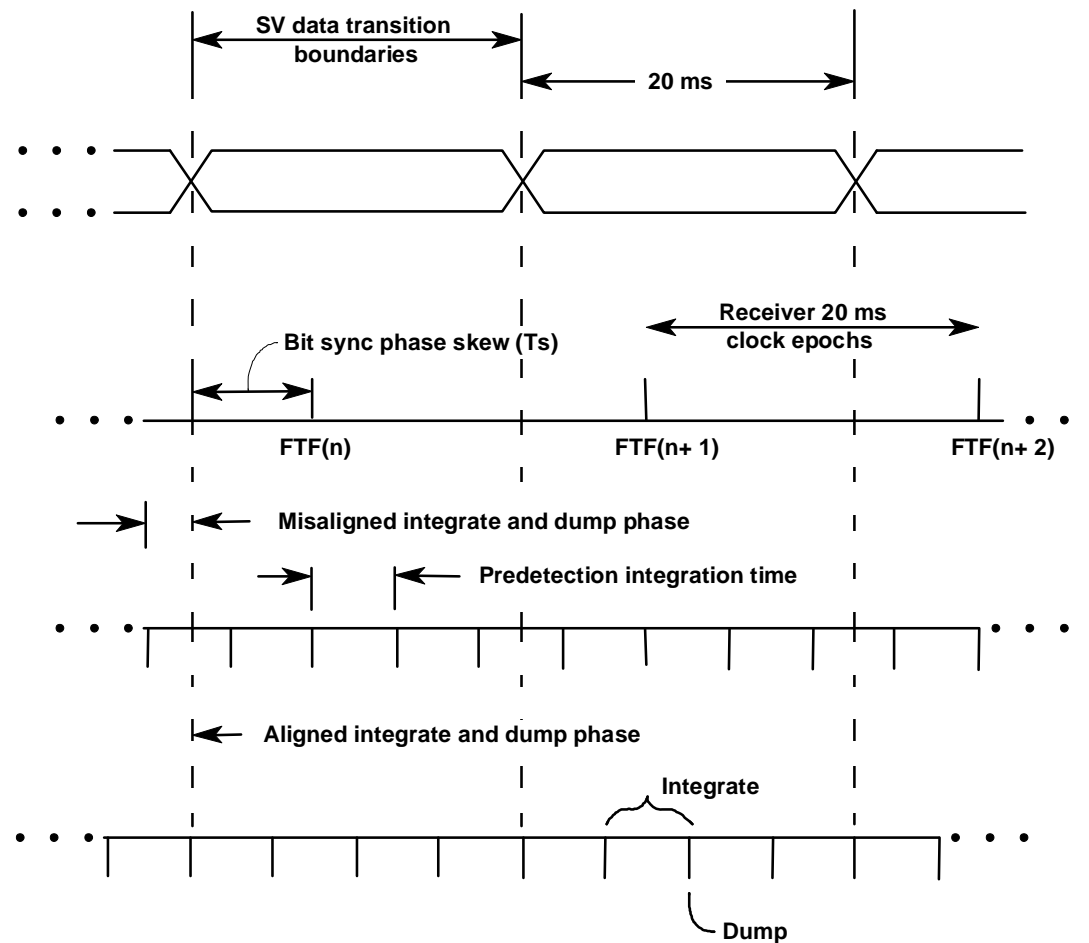
Predetection integration

- Signal integration and dump after carrier and code stripping but prior to signal discriminators
- Can accumulate large numbers with only one to three bits A/D quantization
- Code/carrier wipe-off processes at digital IF sample rates: - 5 MHz for C/A-code and - 50 MHz for P(Y)-code
- Integrate and dump output rates: 50 to 200 Hz
 - These rates well within the interrupt servicing rate of microprocessors

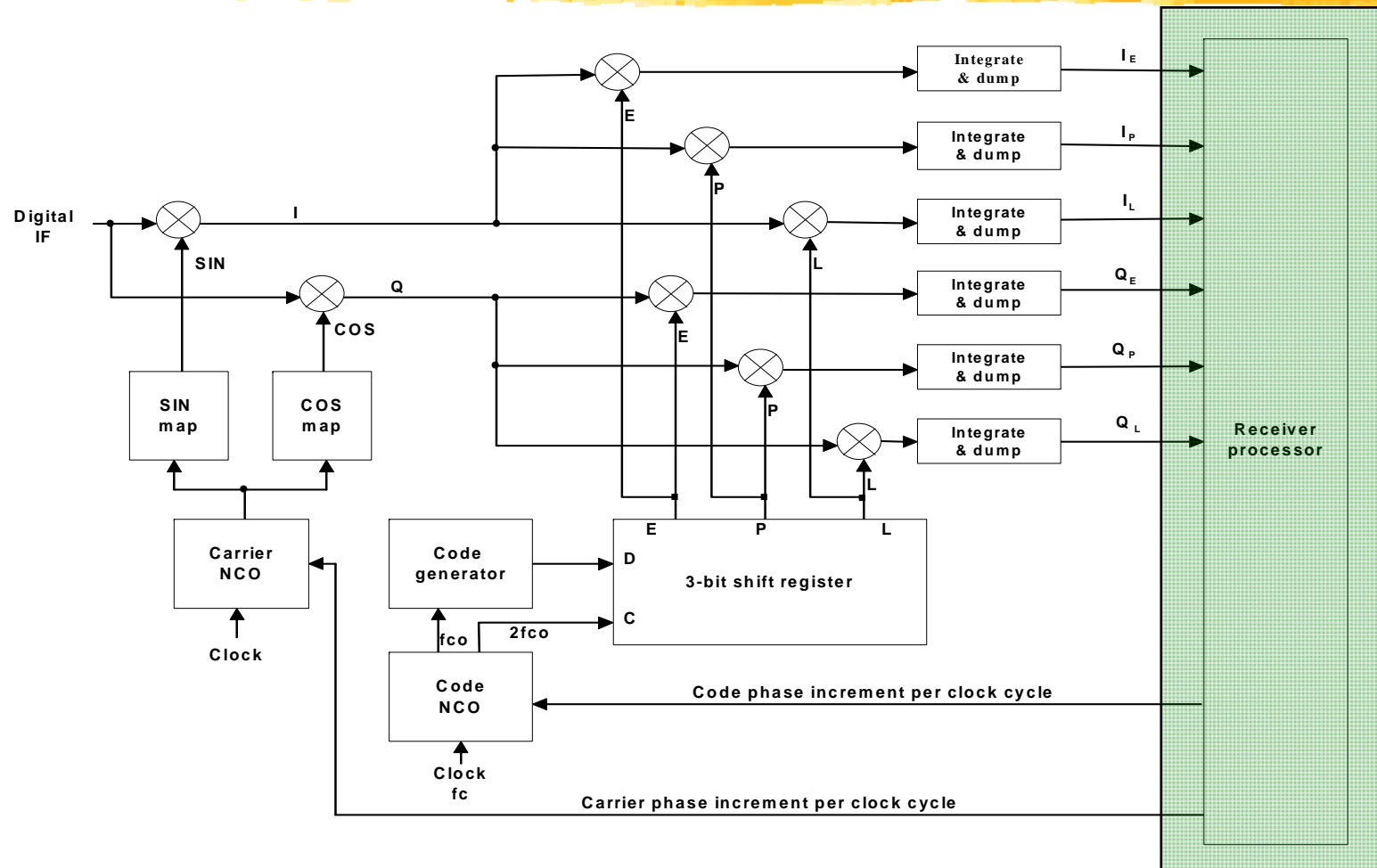
Phase alignment of predetection integrate and dump intervals with SV data transition boundaries

The figure below illustrates the phase alignment needed to prevent the predetection integrate and dump intervals from integrating across an SV data transition boundary. The start and stop boundaries for these integrate and dump functions should not straddle the data bit transition boundaries because each time the SV data bits change sign, the signs of the subsequent integrated I and Q data change. If the boundary is straddled, the result of the predetection integration for that interval will be degraded. Usually during initial searches the receiver does not know where the SV data bit transition boundaries are located. Then, the performance degradation has to be accepted until the bit synchronization process locates them. As shown in the figure below, the SV data transition boundary usually does not align with the receiver's 20 milliseconds clock boundary, which will hereafter be called the fundamental time frame (FTF). The phase offset is shown as "bit sync phase skew," because the SV bit synchronization process initially determines this phase offset. In general, the bit sync phase skew is different for every SV being tracked. The bit sync phase skew changes as the range to the SV changes. The receiver design should accommodate these data bit phase skews if an optimal predetection integration time of 20 ms is to be used during steady state tracking. However, some designers choose to keep the predetection integrate and dump time short (suboptimal at 1 to 5 ms), accepting the increased squaring loss and the degradation due to occasional data transition straddle in exchange for the hardware and software design simplifications of ignoring these boundaries

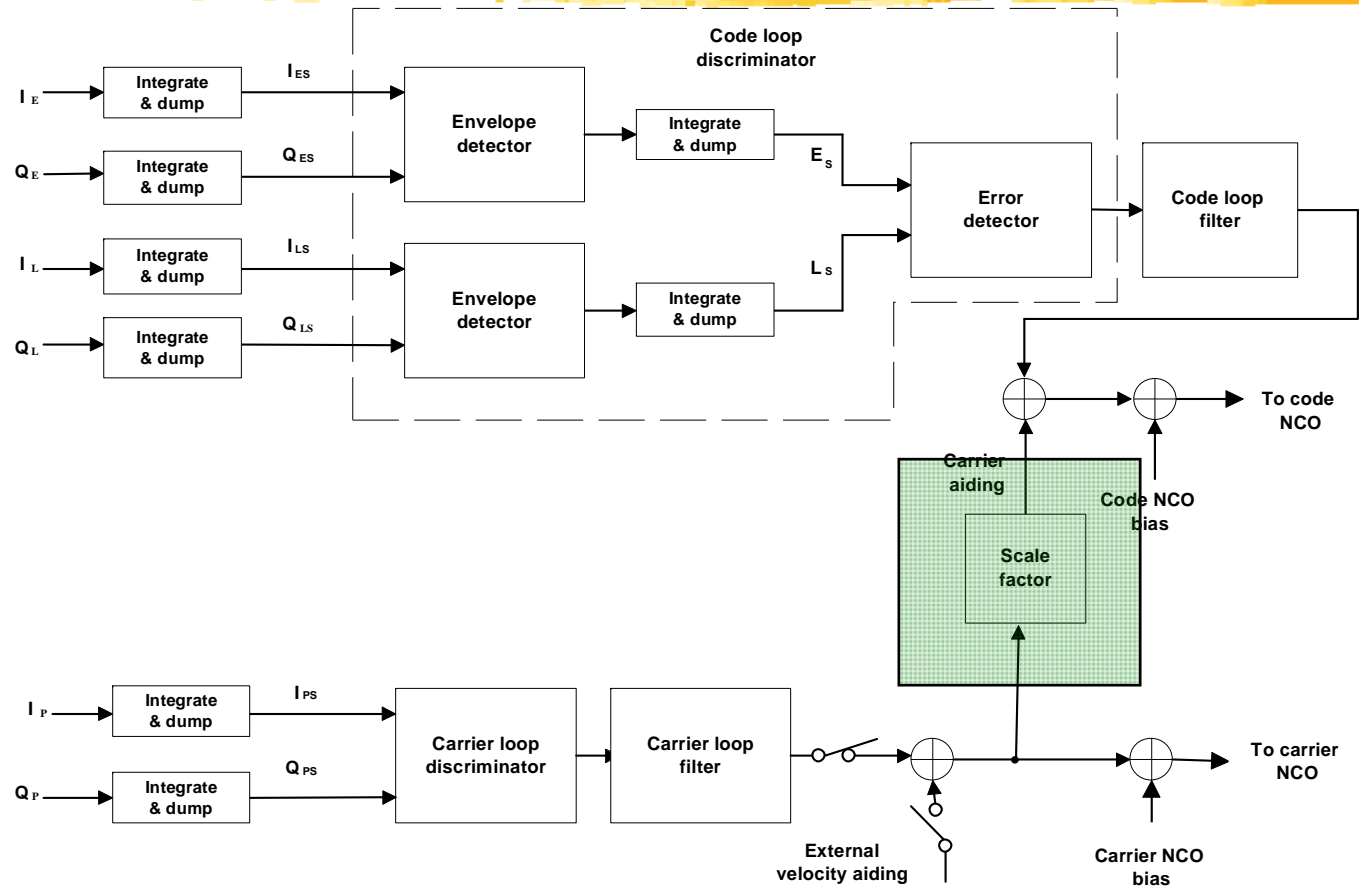
Phase alignment of predetection integrate and dump intervals with SV data transition boundaries



Digital receiver channel block diagram – receiver processor code/carrier loops



Carrier aiding of code loop: scale factors for carrier aided loop



Carrier aiding of code loop

In the previous figure depicting the baseband processor code and carrier tracking loops, the carrier loop filter output is adjusted by a scale factor and added to the code loop filter output as aiding. This is called a carrier-aided code loop. The scale factor is required because the Doppler effect on the signal is inversely proportional to the wavelength of the signal. Therefore, for the same relative velocity between the SV and the GPS receiver, the Doppler on the code chipping rate is much smaller than the Doppler on the L-band carrier. The scale factor which compensates for this difference in frequency is given by:

$$\text{Scale Factor} = \frac{R_c}{f_L}$$

where:

- R_c = code chipping rate (chips/s)
 - = R_0 for P(Y)-code = 10.23×10^6 chips/s
 - = $R_0/10$ for C/A-code = 1.023×10^6 chips/s
- f_L = L-band carrier (Hz)
 - = $154 f_0$ for L1 = $154 \times 10.23 \times 10^6$ Hz = 1575.42 MHz
 - = $120 f_0$ for L2 = $120 \times 10.23 \times 10^6$ Hz = 1227.60 MHz

Carrier aiding of code loop

- Note in previous block diagram that unbiased carrier Doppler is scaled and added to unbiased code Doppler
 - Scale adjusts Doppler effect on carrier frequency for code aiding as follows:

$$\text{Scale factor} = \frac{R_c}{f_L}$$

- where: R_c = code chipping rate (chips/s)
- f_L = L-band carrier (Hz)

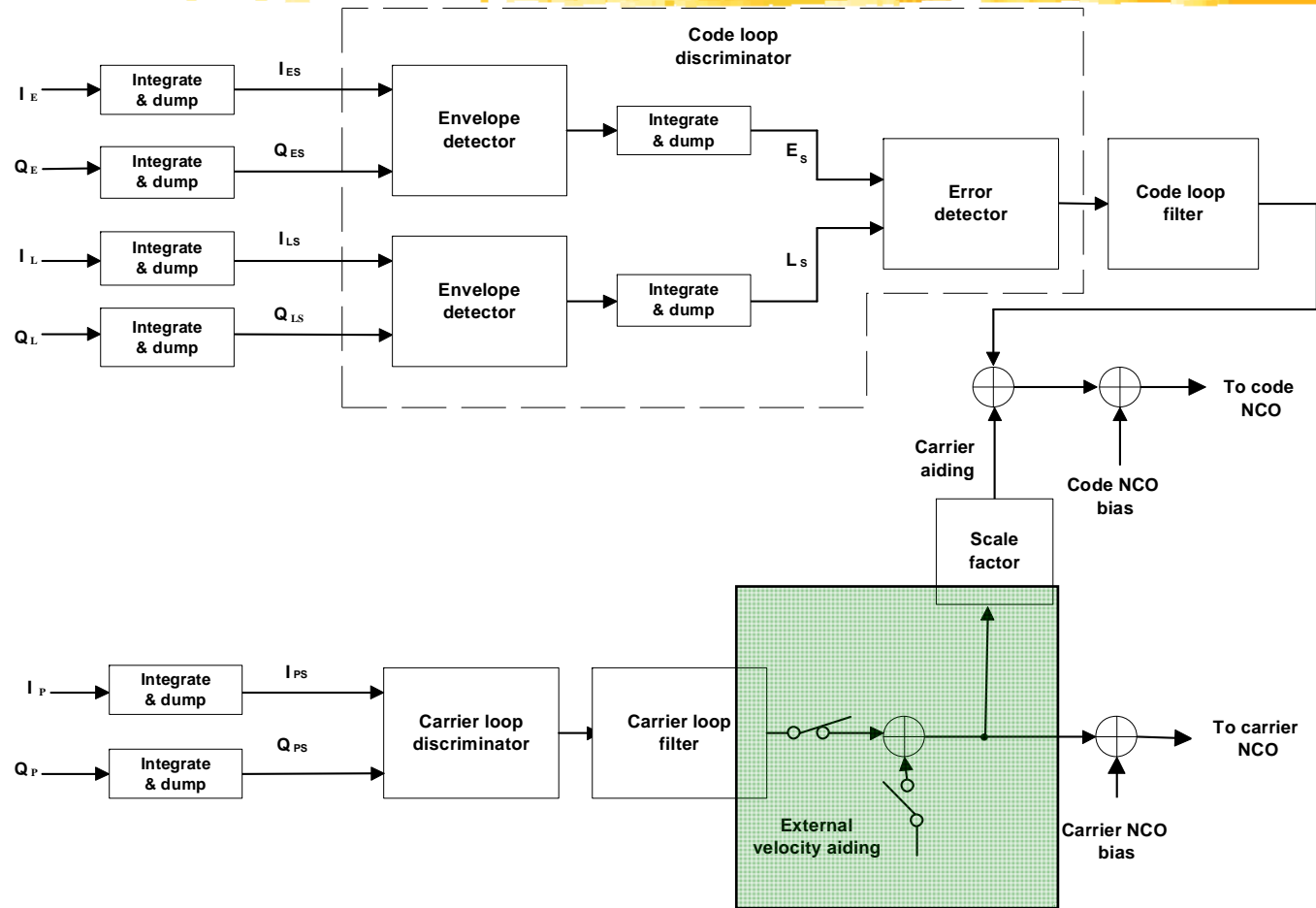
Scale factors for carrier aided code

The carrier loop output should always provide aiding to the code loop because the carrier loop jitter is about three orders of magnitude less noisy than the code loop for the same loop filter noise bandwidth and thus more accurate. The carrier loop aiding removes virtually all of the line-of-sight dynamics from the code loop, so the code loop filter order can be made smaller, the predetection integration time can be made longer and the code loop bandwidth can be made much narrower than for the unaided case, thereby increasing the code loop tracking threshold and reducing the noise in the code loop measurements. Since both the code and carrier loops must maintain track, there is nothing lost in tracking performance by using carrier aiding for an unaided GPS receiver even though the carrier loop is the weakest link.

Scale factors for carrier aided code

Carrier frequency f_L (Hz)	Code Rate R_c (chips/s)	Scale Factor
$L1 = 154 f_0$	$C/A = R_0/10$	$1/1540 =$ 0.00064935
$L1 = 154 f_0$	$P(Y) = R_0$	$1/154 =$ 0.00649350
$L2 = 120 f_0$	$P(Y) = R_0$	$1/120 =$ 0.00833333

Carrier aiding of code loop: external aiding



External aiding

As shown in the generic baseband processor code and carrier tracking loops block diagram, external velocity aiding from an inertial measurement unit (IMU) can be provided to the receiver channel in closed carrier loop operation. The switch, shown in the unaided position, must be closed when external velocity aiding is applied. The external rate aiding must be converted into line-of-sight velocity aiding with respect to the GPS satellite. The lever arm effects on the aiding must be computed with respect to the GPS antenna phase center, which requires a knowledge of the vehicle attitude and the location of the antenna phase center with respect to the navigation center of the external source of velocity aiding. For closed carrier loop operation, the aiding must be very precise and have little or no latency or the tracking loop must be delay compensated for the latency. If open carrier loop aiding is implemented, less precise external velocity aiding is required, but there can be no delta range measurements available from the receiver so it is a short term, weak signal hold-on strategy. In this weak signal hold-on case, the output of the carrier loop filter must be set to zero and there is no need to process the prompt correlator signals in the carrier loop discriminator, but these signals are still used for C/N_0 computation, etc.

The benefits of external velocity aiding are that the tracking loop dynamics are removed by the external aiding to the extent that the aiding provides "true" line-of-sight velocity into the tracking loop. This permits the tracking loop filter noise bandwidth to be made more narrow, the predetection integration time can be made longer and, typically, the order of the loop filter to be lower, than would be the case for the unaided loop which would have to track through the maximum expected dynamic stress. Reducing the noise bandwidth and increasing the predetection integration time improves the tracking threshold which improves the weak signal hold-on characteristic for situations such as antenna gain roll-off or RF interference (jamming). Reducing the order of the carrier loop filter simplifies the computational burden. In the case of the carrier loop for a military GPS receiver, it would typically be third order if unaided and second order if aided. The benefit of the second order carrier loop is that it is unconditionally stable for all bandwidths. Whereas, the third order loop can become unstable under extremely low signal to noise ratio conditions.

The carrier loop output should always provide aiding to the code loop regardless of external aiding. This is because the carrier loop jitter is about three orders of magnitude less noisy than the code loop for the same loop filter noise bandwidth and thus more accurate. The carrier loop aiding removes virtually all of the line of sight dynamics from the code loop, so the code loop filter order can be made smaller, the predetection integration time can be made longer and the code loop bandwidth can be made much narrower than for the unaided case, thereby increasing the code loop tracking threshold and reducing the noise in the code loop measurements. Since both the code and carrier loops must maintain track, there is nothing lost in tracking performance by using carrier aiding for an unaided GPS receiver even though the carrier loop is the weakest link.

External aiding: where and how it is injected in each tracking loop

- Inertial measurement unit (IMU) - tightly coupled
 - Convert into line-of-sight velocity aiding with respect to the SV
 - Compute lever arm effects between IMU navigation center and receiver antenna phase center
 - Minimize latency or provide delay compensation in carrier tracking loop
- Open carrier loop aiding (code loop aiding)
 - Requires less precise IMU velocity aiding
 - No delta range measurements or SV data available
 - Use only as short term weak signal hold-on strategy

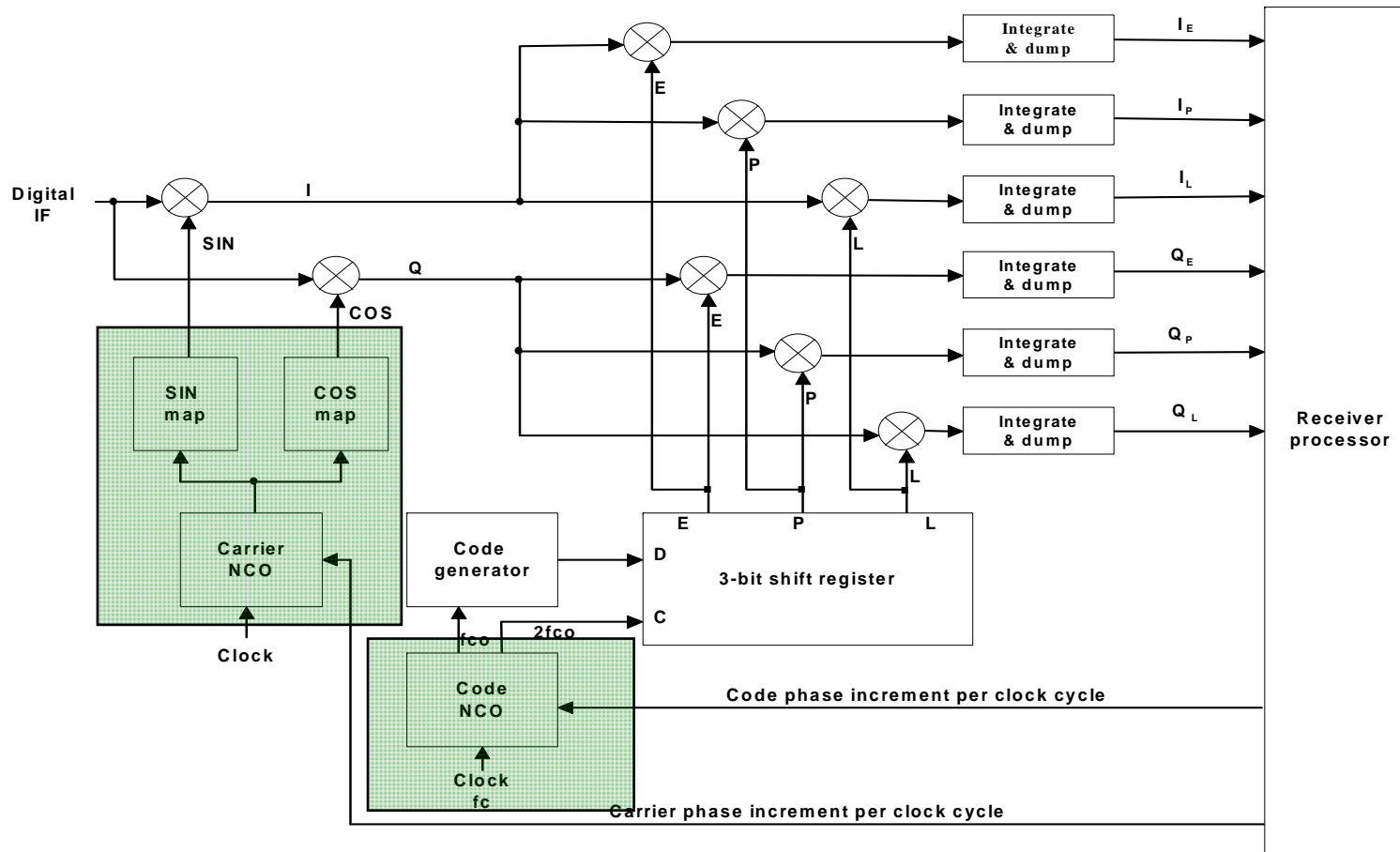
Digital frequency synthesizer block diagram and output waveforms

- Numerically controlled oscillator (NCO)
- Digital frequency synthesizer block diagram
- Digital frequency synthesizer output waveforms
- Digital frequency synthesizer phase diagram
- Mapping NCO output to COS and SIN outputs


Numerically controlled oscillator (NCO)

- NCO is simply an accumulator (increment counter)
- Receiver processor sends commands to NCO (usually at 50 - 100 Hz rate)
 - Based on microprocessor's estimate of phase/frequency tracking errors
 - Initialized with phase state
 - Controlled with phase-rate (phase increment per clock)
- NCO increments phase estimate by: $\text{phase} + \text{phase-rate} \times \text{elapsed time}$
- For carrier wipe-off, NCO produces cosine and sine of phase estimate using lookup table

Digital receiver channel – carrier frequency synthesizer and code NCO

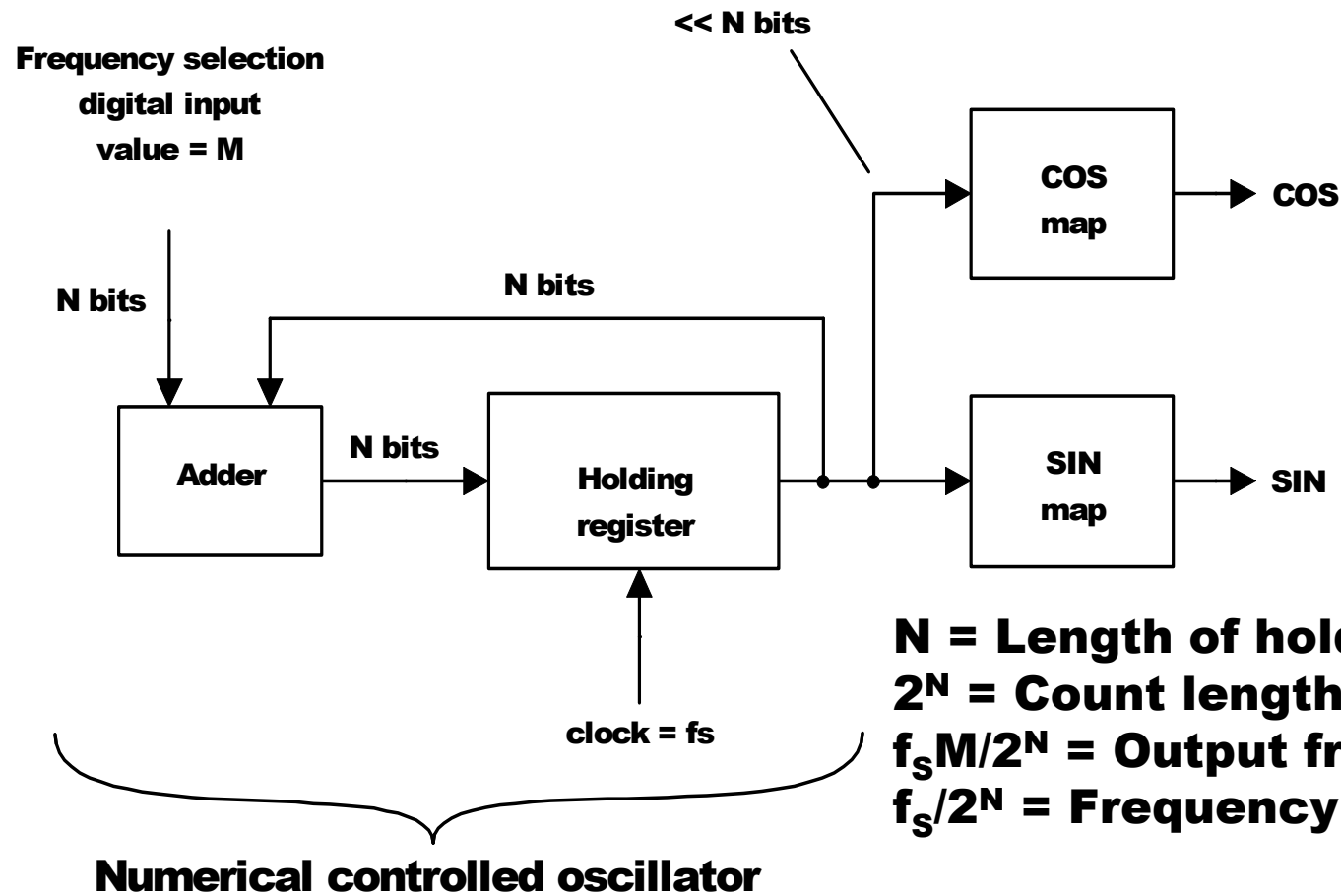


Digital frequency synthesizer block diagram

In this generic design example, both the carrier and code tracking loops use an NCO. One replica carrier cycle and one replica code cycle are completed each time the NCO overflows. A block diagram of the carrier loop NCO and its sine and cosine mapping functions are shown in the figure below 

The figure depicts an expanded block diagram of the digital frequency synthesizer used in the carrier tracking loop of the generic digital receiver channel. It consists of a numerical controlled oscillator (NCO) and a cosine/sine mapping function. The typical number of bits for the carrier NCO is 32. This provides extremely high frequency resolution of $f_s/2^N$, where f_s is the clock frequency and N is the number of bits. The NCO is used in both the code and carrier hardware synthesis functions. There are some commercial GPS receivers which do not implement a true code loop NCO. Instead, they simply operate the code generator at the nominal chipping rate from a simple divider. When the drift of the replica code is detected by the code tracking loop, then clock pulses are added or swallowed as required to re-center it. This results in very noisy pseudorange quantization. Instead of a fine adjustment of the code chipping rate of the NCO, the phase adjustment is in fractions of a chip as determined by the clock divider circuit. For example, if the clock divider circuit is 50, the adjustment is 1/50th of a chip, which is about 6 meters of noise for the C/A-code

Digital frequency synthesizer block diagram

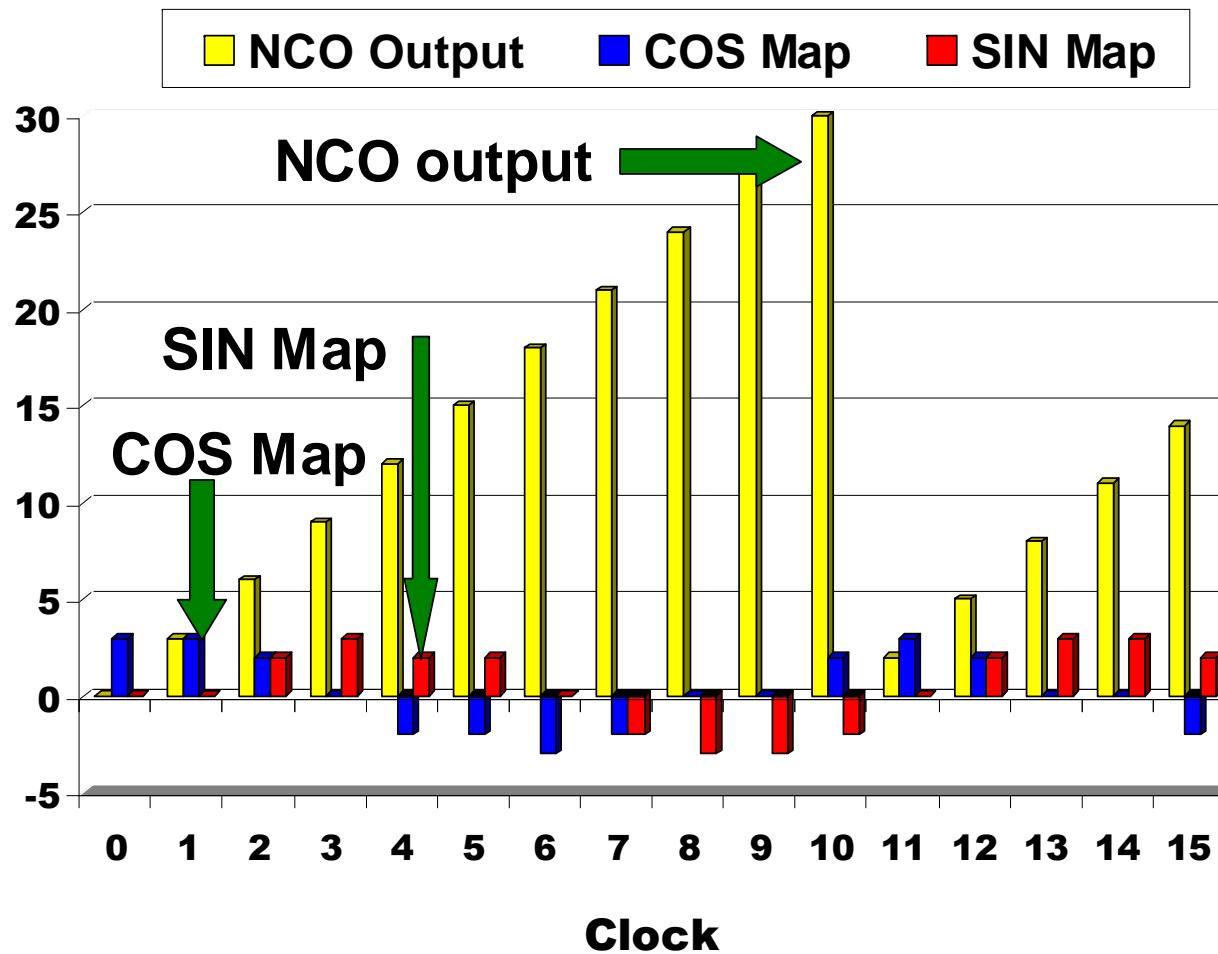


N = Length of holding register
 2^N = Count length
 $f_s M / 2^N$ = Output frequency
 $f_s / 2^N$ = Frequency resolution

Digital frequency synthesizer output values

The figure below depicts the values of the staircase output of a simple 5-bit NCO with the cosine output and sine output of the carrier frequency synthesizer shown as a result of mapping the top. In the example the increment M is 3. The map function example shown corresponds to a 3-bit quantization map shown in a following figure. If the A/D quantization is only one bit, then it would be necessary only to map the sign bit of the holding register. Note that every overflow of the holding register corresponds to one carrier cycle. The frequency of this staircase output is increased by increasing the digital value of M , which is called the carrier phase increment per clock cycle in the generic digital receiver channel block diagram. The frequency is decreased by decreasing the digital value of M . The carrier NCO bias that is added in the generic baseband processor code and carrier tracking loops block diagram is the value which most closely sets the NCO to the zero Doppler carrier frequency (at the IF) of the SV being tracked. For the code NCO, only the staircase overflow output is needed. For the example shown in the generic digital receiver channel block diagram, two outputs are used. One, at twice the frequency of the code generator rate, provides $\frac{1}{2}$ -chip separation between the three code phases. This output is used by the 3-bit shift register. The output from the most significant bit of the NCO provides the chipping rate into the replica code generator. The code NCO bias shown in the generic baseband processor code and carrier tracking loops block diagram is the value which causes the NCO to run at the zero Doppler code chipping rate (10.23 Mchips/s for P(Y)-code and 1.023 Mchips/s for C/A-code). Note that the phase state of the NCO at any time represents the fractional part of the replica code phase. For a 32-bit holding register, this provides nanometers of code range resolution compared to the crude quantization capability of the fixed divider technique.

Digital frequency synthesizer output waveforms



Digital frequency synthesizer design

The figure and table below describe the techniques step-by-step for designing a digital frequency synthesizer. The example shown is for $J = 3$ -bits, where J = the amplitude quantization used by the receiver A/D converter used at the IF of the receiver. $2^J = K$ values are computed. If more than one bit is used, a table look-up is usually provided for the sine and cosine maps. Note that only one quarter of a cycle need be stored in only one map if the correct logic is used to determine which quadrant the holding register output is in for the respective sine and cosine output functions.

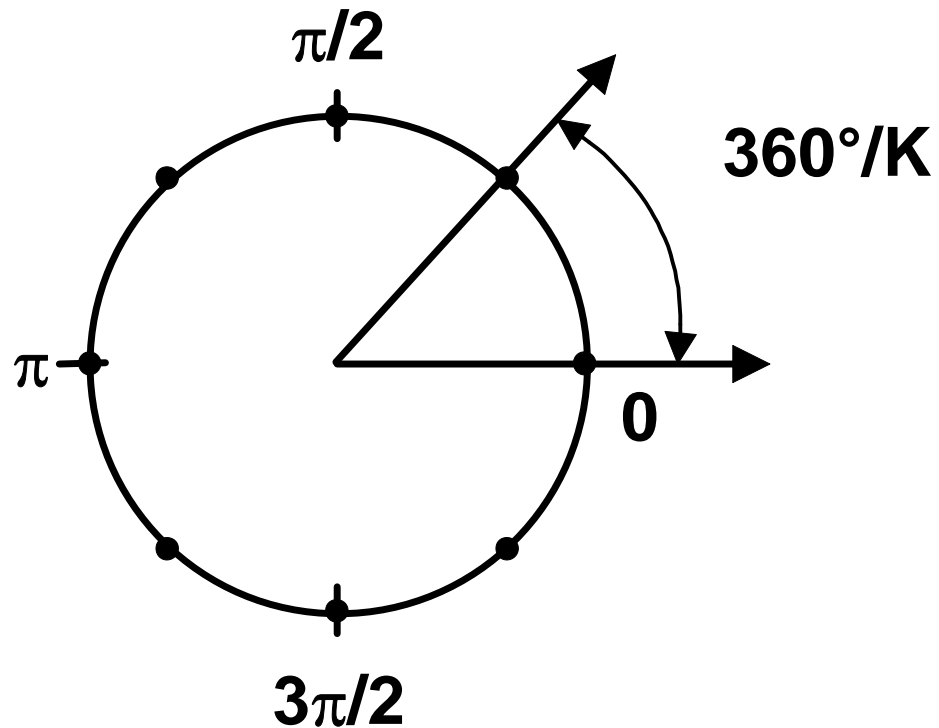
Notes:

1. The number of bits, J , is determined for the sin and cos outputs. The phase plane of 360° is subdivided into $2^J = K$ phase points.
2. K values are computed for each waveform, one value per phase point. Each value represents the amplitude of the waveform to be generated at that phase point. The upper J bits of the holding register are used to determine the address of the waveform amplitude.
3. Rate at which phase plane is traversed determines the frequency of the output waveform.
4. The upper bound of the amplitude error is:
$$e_{MAX} = 2\pi/K$$
5. The approximate amplitude error is: $e = e_{MAX}\cos\Phi(t)$
where $\cos\Phi(t)$ is the phase angle.

Digital frequency synthesizer design

- Design example: $J=3$ bits
 - Where J = receiver A/D converter amplitude quantization (more bits if ADC is non-linear)
 - $2^J = K$ values (of sin and cos) are computed usually by table look-up (only $\frac{1}{4}$ cycle plus quadrant needed)
 - Rate at which phase plane is traversed determines frequency (see chart below)
 - Upper bound amplitude error: $e_{MAX} = 2\pi/K$
 - Approximate amplitude error: $e = e_{MAX} \cos \Phi(t)$
 - Where $\Phi(t)$ is the actual phase angle.

Digital frequency synthesizer phase diagram



Mapping NCO output to COS and SIN outputs

The table below illustrates the technique for mapping the NCO into the digital cosine and sine functions. The example shown is for $J = 3$ -bits, where $J =$ the amplitude quantization used by the receiver A/D converter at the IF of the receiver. $2^J = K = 8$ values are computed. Note that only one quarter of a cycle need be stored in only one map if the correct logic is used to determine which quadrant the holding register output is in for the respective cosine and sine output functions.

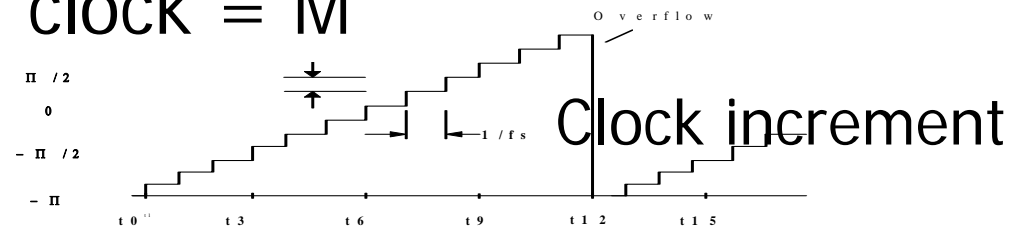
Mapping NCO output to COS and SIN outputs

Radians	Holding register	COS map	SIN map
0	000...	011...	000...
$\pi/4$	001...	010...	010...
$\pi/2$	010...	000...	011...
$3\pi/4$	011...	110...	010...
π	100...	111...	000...
$5\pi/4$	101...	110...	110...
$3\pi/2$	110...	000...	111...
$7\pi/4$	111...	010...	110...

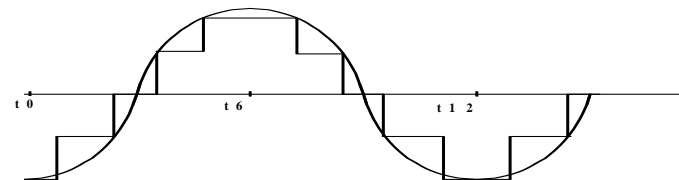
Digital frequency synthesizer output waveforms for J=3 bits

Phase increment per clock = M

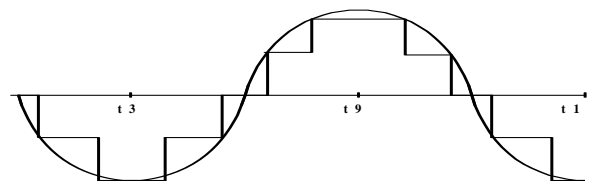
NCO output



COS map output



SIN map output



Session III



Session III

Code tracking loop design



- Generic GPS receiver code tracking loop block diagram
- Approximation techniques for computing I and Q signal envelopes
- Common delay lock loop (DLL) discriminators
- Comparison of DLL discriminators S-curves
- Code correlation process for three different replica code phases
- Correlation for replica codes: 1/2 chip early, 1/4 chip early, aligned and 1/4 chip late
- Code discriminator output versus replica code offset

Code tracking loops

The description of the code tracking loop has been spread out over several block diagrams. The code tracking loop consists of the carrier wipe-off hardware, code wide-off hardware, the predetection integration hardware (some additional predetection integration may take place in the software), the baseband software which contains the code loop discriminator and the code loop filter in the receiver processor, and finally, the replica code generation hardware which contains the code NCO, code generator and the 3-bit shift register. All of the component parts of the code tracking loop are shown in the next figure.

The code tracking loop is usually called a delay lock loop (DLL). The most popular DLL is the non-coherent type that uses the envelopes of I and Q; i.e., by squaring these components and adding them, then taking the square root of the result. Another popular non-coherent DLL avoids the square root process by using the sum of the squares only (the signal power) as an input. The most popular discriminator is the early minus late version which forms an error based on difference in the early and late envelopes (or power) obtained from the early and late correlators. However, there is another DLL discriminator, called the dot product discriminator, which uses the envelopes from all three correlators (early, prompt, late).

If the receiver is in phase lock, then there is essentially no power in the Q components of the early, prompt and late signals, so the code loop discriminator can be operated with only the I components. There is growing interest in this type of code tracking loop because it produces the least noisy code or pseudorange measurements. However, this makes the receiver extremely vulnerable if there are cycle slips during PLL operation and FLL operation is not compatible with coherent tracking. Therefore, coherent code loop tracking requires very sophisticated carrier tracking techniques to remain stable.

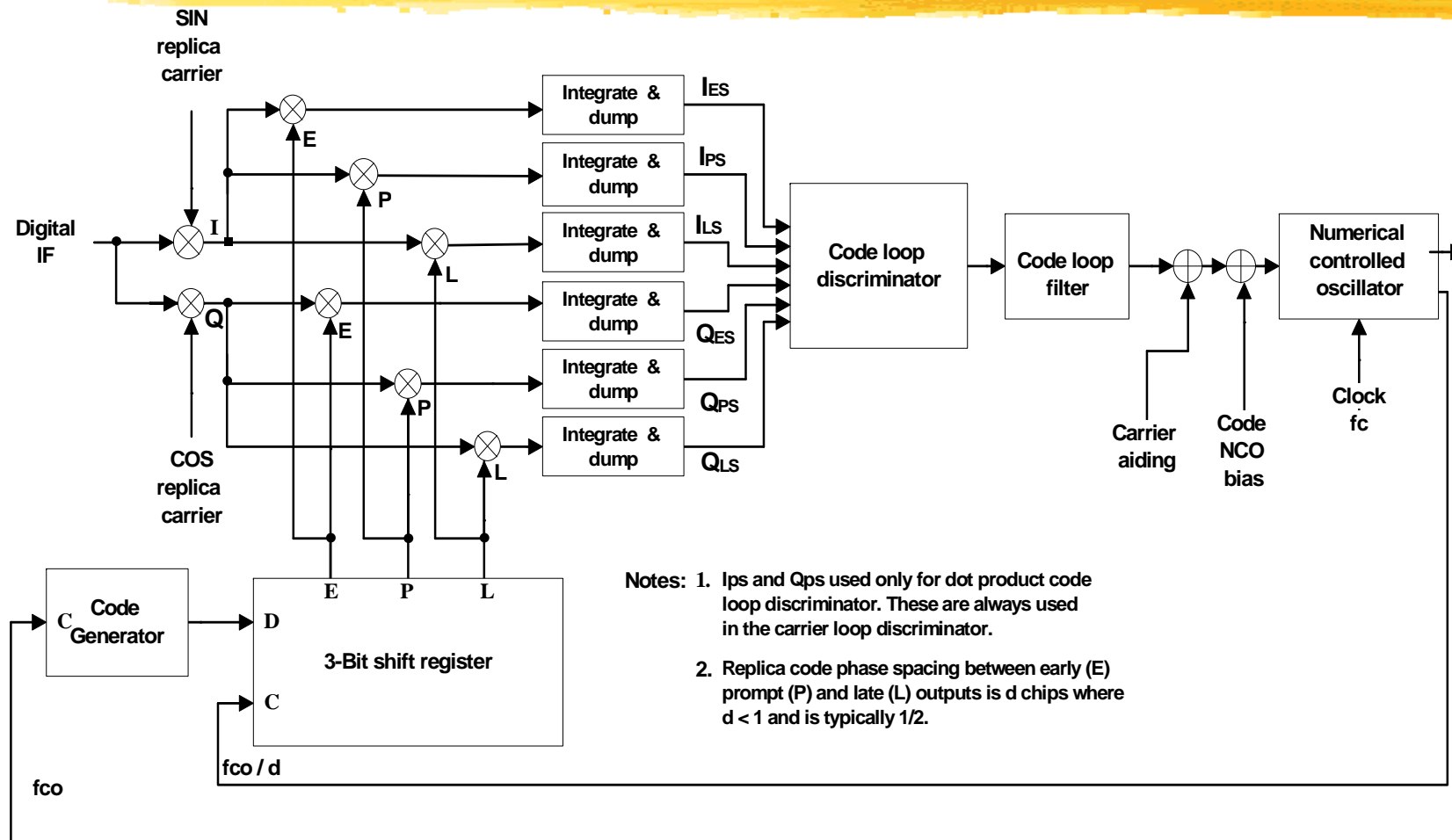
Generic GPS receiver code tracking loop block diagram

- Carrier tracking loop description spread out over several block diagrams
 - Replica code generation hardware
 - Carrier then code wipe-off hardware
 - Predetection integration hardware
 - Baseband software
 - Code discriminator and filter
- Discriminator defines code loop type
 - Non-coherent or coherent

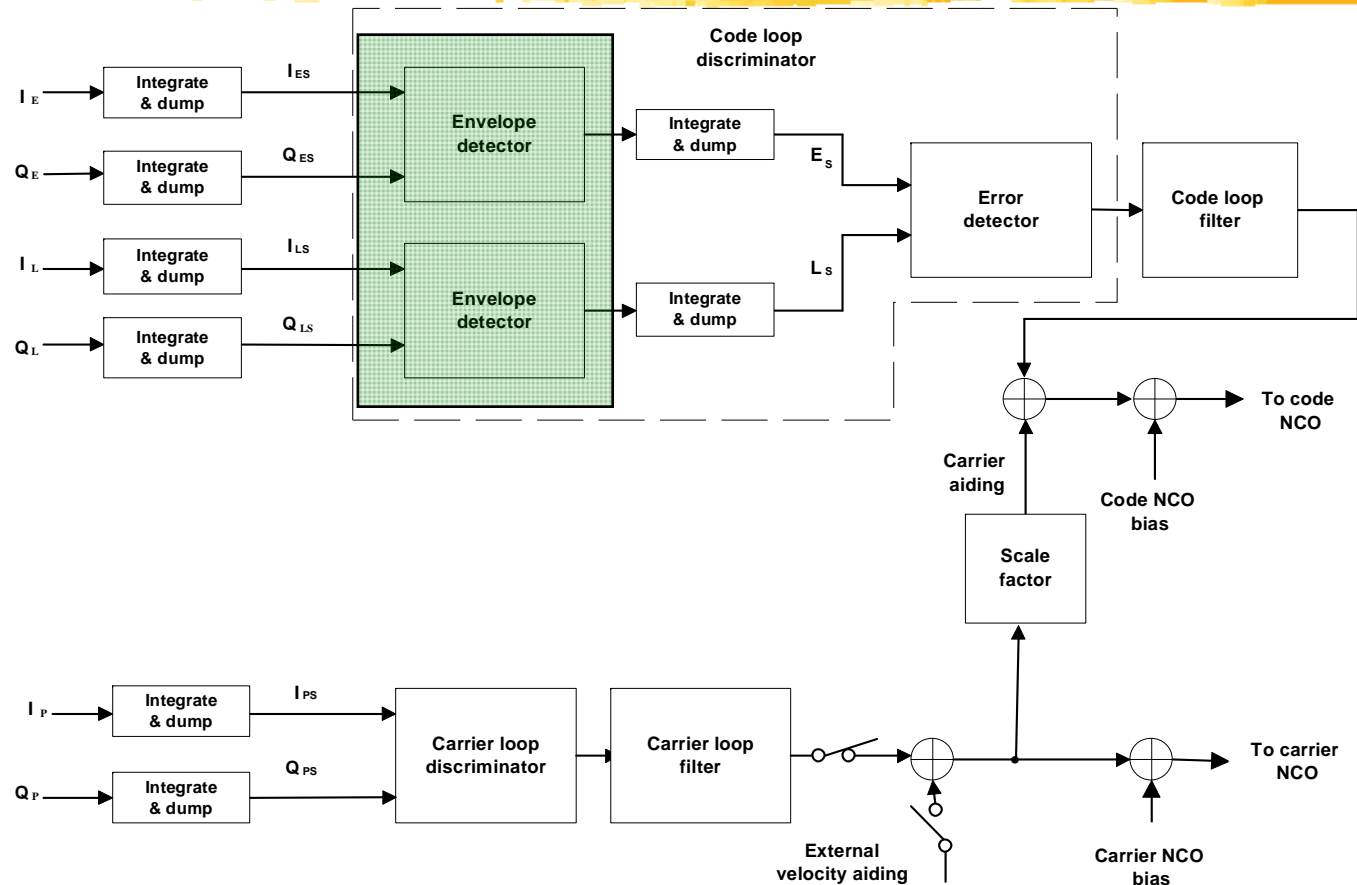
Generic GPS receiver code tracking loop block diagram

The figure below illustrates the block diagram of only the GPS receiver code tracking loop. The design of the programmable predetection integrators, the code loop discriminator and the code loop filter characterizes the receiver code tracking loop. These three functions determine the most important two performance characteristics of the receiver code loop design: the code loop thermal noise error and the maximum line-of-sight dynamic stress threshold. Even though the carrier tracking loop is the weak link in terms of the receiver's dynamic stress threshold, it would be disastrous to attempt to aid the carrier loop with the code loop output. This is because, unaided, the code loop thermal noise is about three orders of magnitude larger than the carrier loop thermal noise.

Generic GPS receiver code tracking loop block diagram



Approximation techniques for computing I and Q signal envelopes



Approximation techniques for computing I and Q signal envelopes

The JPL approximation to the signal envelope defined exactly as:

$$A_{ENV} = \sqrt{I^2 + Q^2}$$

is:

$$A_{JPL} = X + 1/8 Y \quad \text{if } X \geq 3Y$$

$$A_{JPL} = 7/8 X + 1/2 Y \quad \text{if } X < 3Y$$

$$\text{where: } X = \text{MAX} (|I|, |Q|)$$

$$Y = \text{MIN} (|I|, |Q|)$$

The Robertson approximation is:

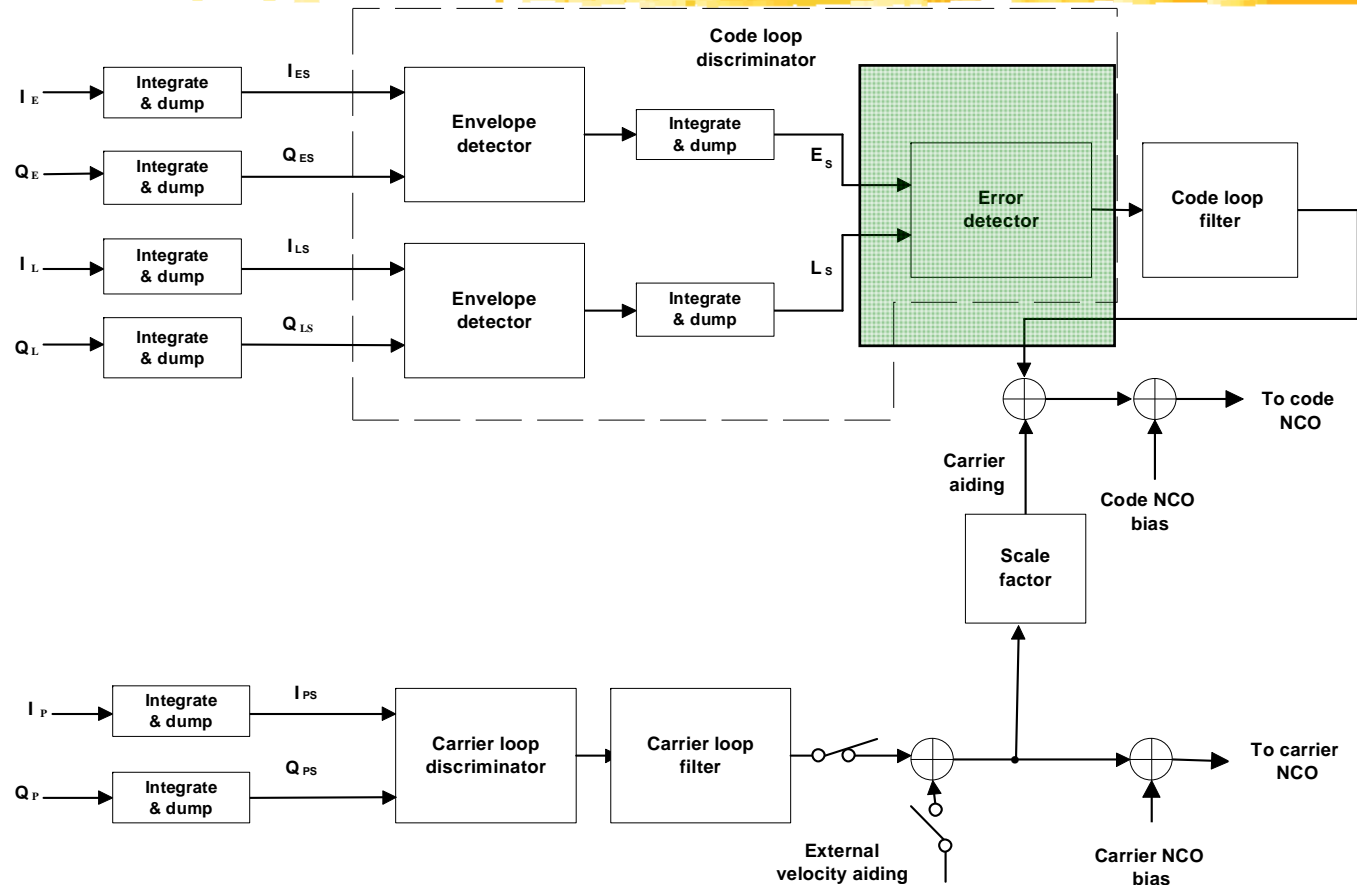
$$A_{RBN} = \text{MAX} (|I| + 1/2 |Q|, |Q| + 1/2 |I|)$$

The JPL approximation is the most accurate, but has the most computational burden. It is recommended for use during closed loop tracking, while the Robertson approximation is recommended for use during open loop search.

Approximation techniques for computing I and Q signal envelopes

- JPL approximation of $A_{ENV} = \sqrt{I^2 + Q^2}$
(most accurate, used in track):
 - $A_{JPL} = X + 1/8Y$ if $X \geq 3Y$
 - $A_{JPL} = 7/8X + 1/2Y$ if $X < 3Y$
 - where: $X = \text{MAX}(|I|, |Q|)$; $Y = \text{MIN}(|I|, |Q|)$
- Robertson approximation (used in search):
 - $A_{RBN} = \text{MAX}(|I| + 1/2|Q|, |Q| + 1/2|I|)$

Common delay lock loop (DLL) discriminators



Common delay lock loop discriminators

The table below summarizes three GPS receiver noncoherent delay lock loop (DLL) discriminators and their characteristics. The fourth DLL discriminator shown is a normalized version of the third discriminator. The normalization removes the amplitude sensitivity of this DLL discriminator which improves performance under pulse type RF interference. The other two DLL discriminators also can be normalized in a similar manner.

Note: As shown for every code loop discriminator, the power or the envelope values may be summed to reduce the iteration rate of the code loop filter as compared to that of the carrier loop filter when the code loop is aided by the carrier loop. Note that this does not increase the predetection integration time for the code loop. However the code loop NCO must be updated every time the carrier loop NCO is updated even though the code loop filter output has not been updated. The last code loop filter output is combined with the current value of carrier aiding.

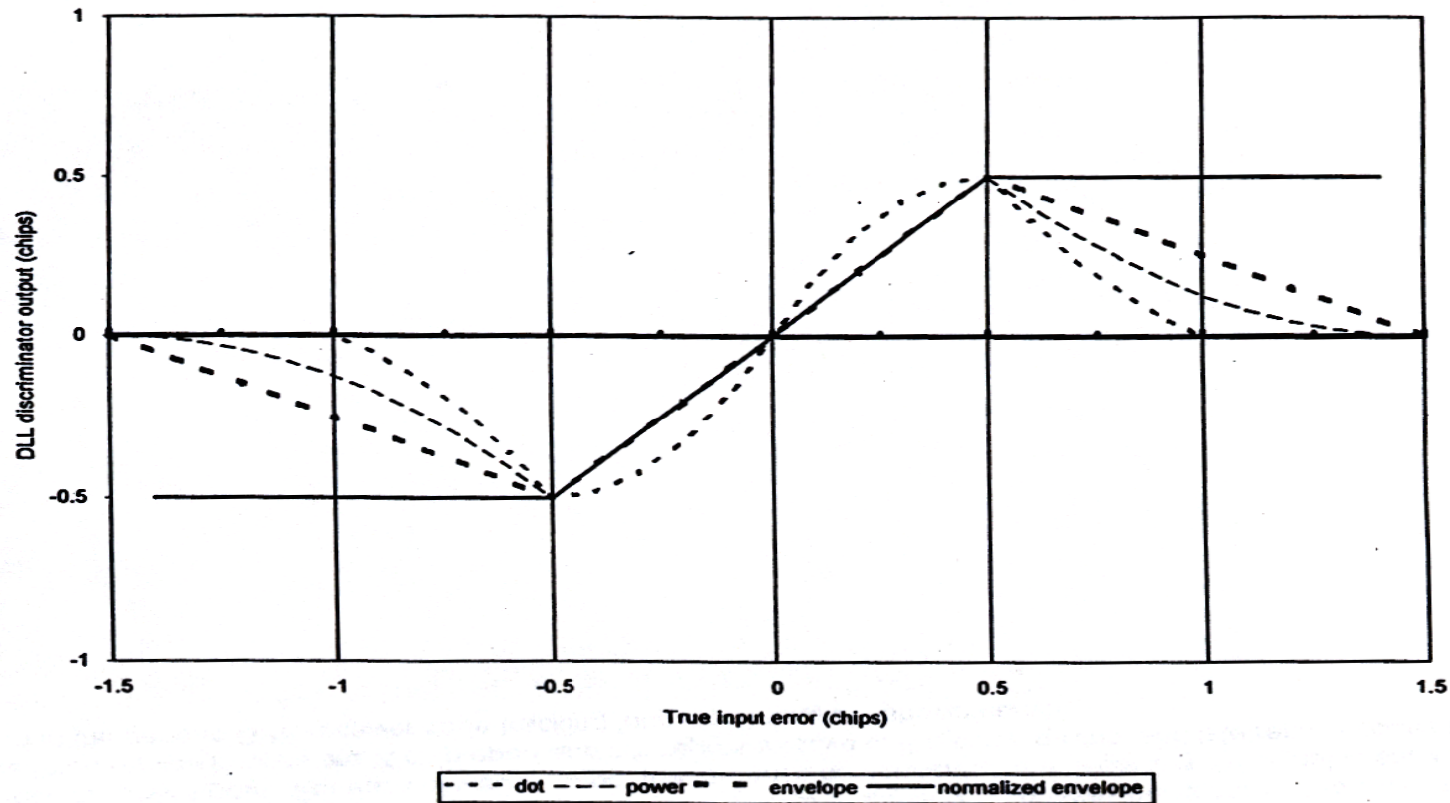
Common DLL discriminators

Discriminator algorithm	Characteristics
$(I_{ES} - I_{LS}) I_{PS} + (Q_{ES} - Q_{LS}) Q_{PS}$	<p>Dot product power. This is the only DLL discriminator which uses all three correlators and this results in the lowest baseband computational load. For ½ chip correlator spacing, it produces nearly true error output within ± ½ chip of input error.</p>
$\frac{1}{2} [(I_{ES}^2 + Q_{ES}^2) - (I_{LS}^2 + Q_{LS}^2)]$	<p>Early minus late power. Moderate computational load. Essentially the same DLL discriminator error performance as early minus late envelope within ± ½ chip of input error.</p>
$\frac{1}{2} [\sqrt{I_{ES}^2 + Q_{ES}^2} - \sqrt{I_{LS}^2 + Q_{LS}^2}]$	<p>Early minus late envelope. Higher computational load. For ½ chip correlator spacing, produces good tracking error within ± ½ chip of input error.</p>
$\frac{1}{2} \frac{[\sqrt{I_{ES}^2 + Q_{ES}^2} - \sqrt{I_{LS}^2 + Q_{LS}^2}]}{[\sqrt{I_{ES}^2 + Q_{ES}^2} + \sqrt{I_{LS}^2 + Q_{LS}^2}]}$	<p>Early minus late envelope normalized by the early plus late envelope to remove amplitude sensitivity. Highest computational load. For ½ chip correlator spacing, produces good tracking error within less than ± 1.5 chip of input error. Becomes unstable (divide by zero) at ± 1.5 chip input error.</p>

Comparisons of DLL discriminators S-curves

Each discriminator produces an error signal (or S-curve) that is proportional to the timing error between the incoming signal code and the receiver's code replica (for small errors).

Comparison of DLL discriminators S-curves



Coherent DLL Discriminator

For operation in benign or carrier aided environments, the carrier tracking loop can reliably and consistently be operated in phase lock, so the DLL can be operated coherently. When the carrier tracking loop is in phase lock, this means that most of the signal energy is in the in-phase (I) components and the quadrature (Q) components contain nearly zero signal energy. Therefore, it is logical that there could be a DLL mode in which only I components are used. The most accurate DLL algorithm dots the early minus late I components with the prompt component as: $(I_{ES} - I_{LS})(I_{PS})$. The reason it is the most accurate is that the noise in the Q components is not included because the Q components are not used since the Q components contribute little or no signal to the DLL process.

Many commercial receivers assume a benign operational environment and do not even synthesize an early or late Q component. In fact they synthesize a composite early minus late signal by correlating the incoming signal with an early minus late replica code. This saves components and achieves operational accuracy at the expense of robustness of tracking in challenging environments.

Coherent DLL discriminator

- Carrier tracking loop must be in phase lock to support coherent DLL operation
- In this case all Q_{ES} signals are nearly zero, so only the I_{ES} signals are used
- This is the most precise (lowest noise) but also the most fragile DLL tracking mode
- Coherent DLL discriminator algorithm:

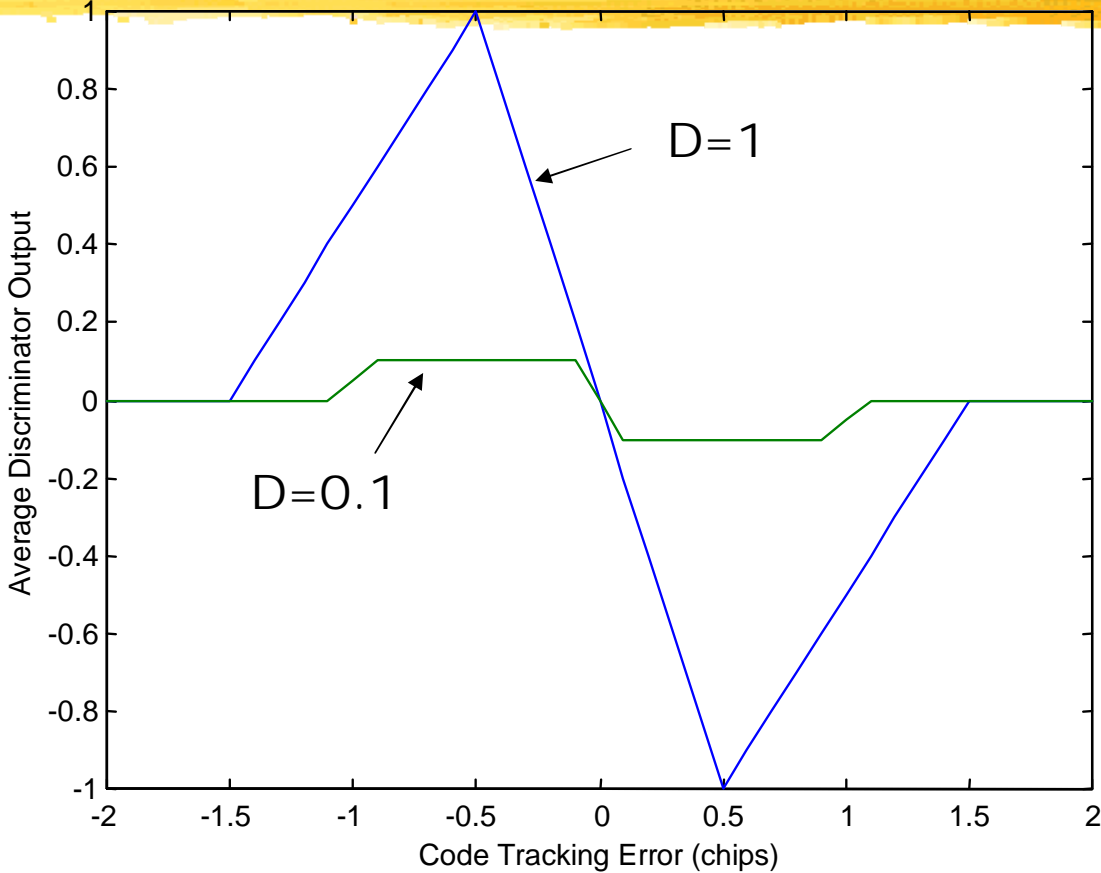
$$(I_{ES} - I_{LS})(I_{PS})$$

Coherent DLL S-curve

The coherent DLL S-curve shows the conventional plot where the early and late correlator spacing are 1-chip apart ($D=1$) as well as the plot where the early and late correlator spacing is only 1/10-chip apart ($D=0.1$). The $D=1$ code loop operational mode is usually called the "wide correlator" mode. It is the most robust DLL operation but is most vulnerable to multi-path error, especially with the GPS C/A code. The $D=0.1$ code loop operational mode is usually called the "narrow correlator" mode. It is the most accurate and is especially effective at suppressing multipath signals because the correlation region between the early and late correlators is much narrower, so a considerable amount of multipath energy is not included. In order to effectively operate in a "narrow correlator" DLL mode, the receiver front end and subsequent C/A code signal processing must be wideband (say 18 MHz instead of the minimum 2 MHz front end C/A code bandwidth prior to signal detection) so as to include several harmonics of the C/A code. Also, the used of carrier aided code helps to remove dynamic stress from the code tracking loop, thereby enabling the DLL to operate in the fragile "narrow correlator" mode with a much narrower code loop filter bandwidth. In other words, there are other things in the receiver design that must be modified in order to support a "narrow correlator" design. For this reason, these design features are included only in high precision GNSS receivers.

In the discriminator error plots, the linear region for small errors corresponds to early and late correlators "straddling" the maximum correlation peak. When the error is zero, this corresponds to the exact phase match of the replica code with respect to the incoming SV code.

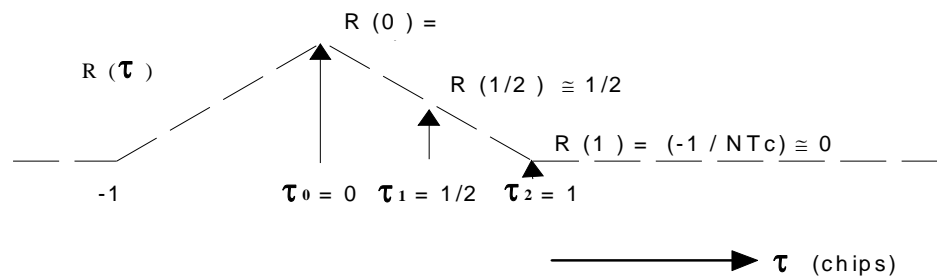
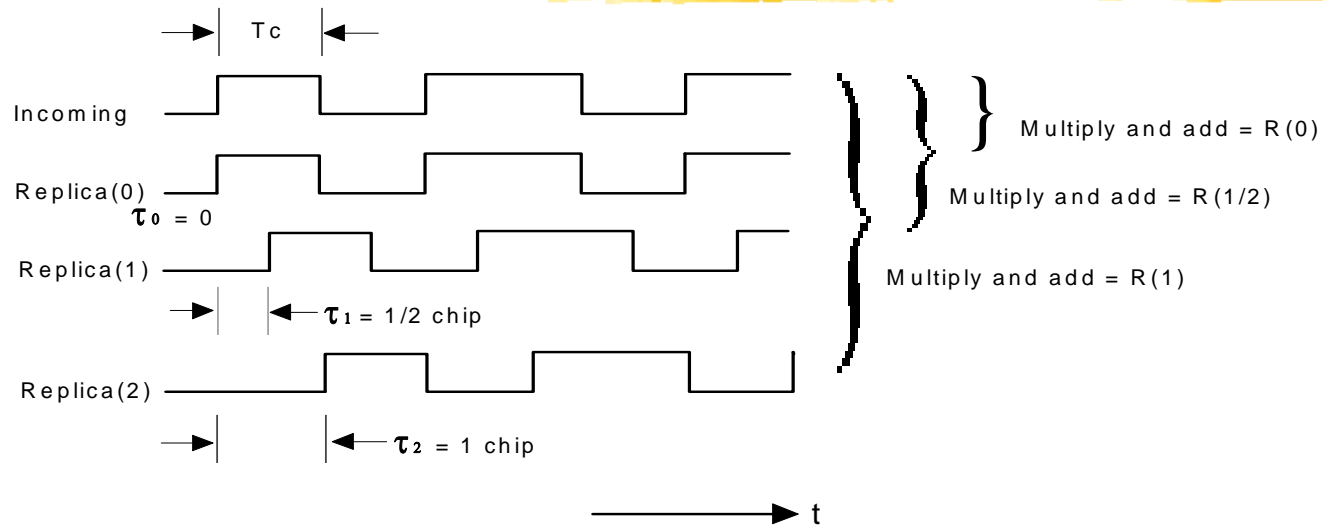
Coherent DLL S-curve



Code correlation examples for three different replica code phases

The figure below illustrates the envelopes that result for three different replica code phases being correlated simultaneously with the same incoming SV signal. For ease of visualization, the in-phase component of the incoming SV signal is shown without noise. The three replica phases are $\frac{1}{2}$ chip apart and are representative of the early, prompt and late replica codes that are synthesized in the generic GPS receiver code tracking loop block diagram shown earlier.

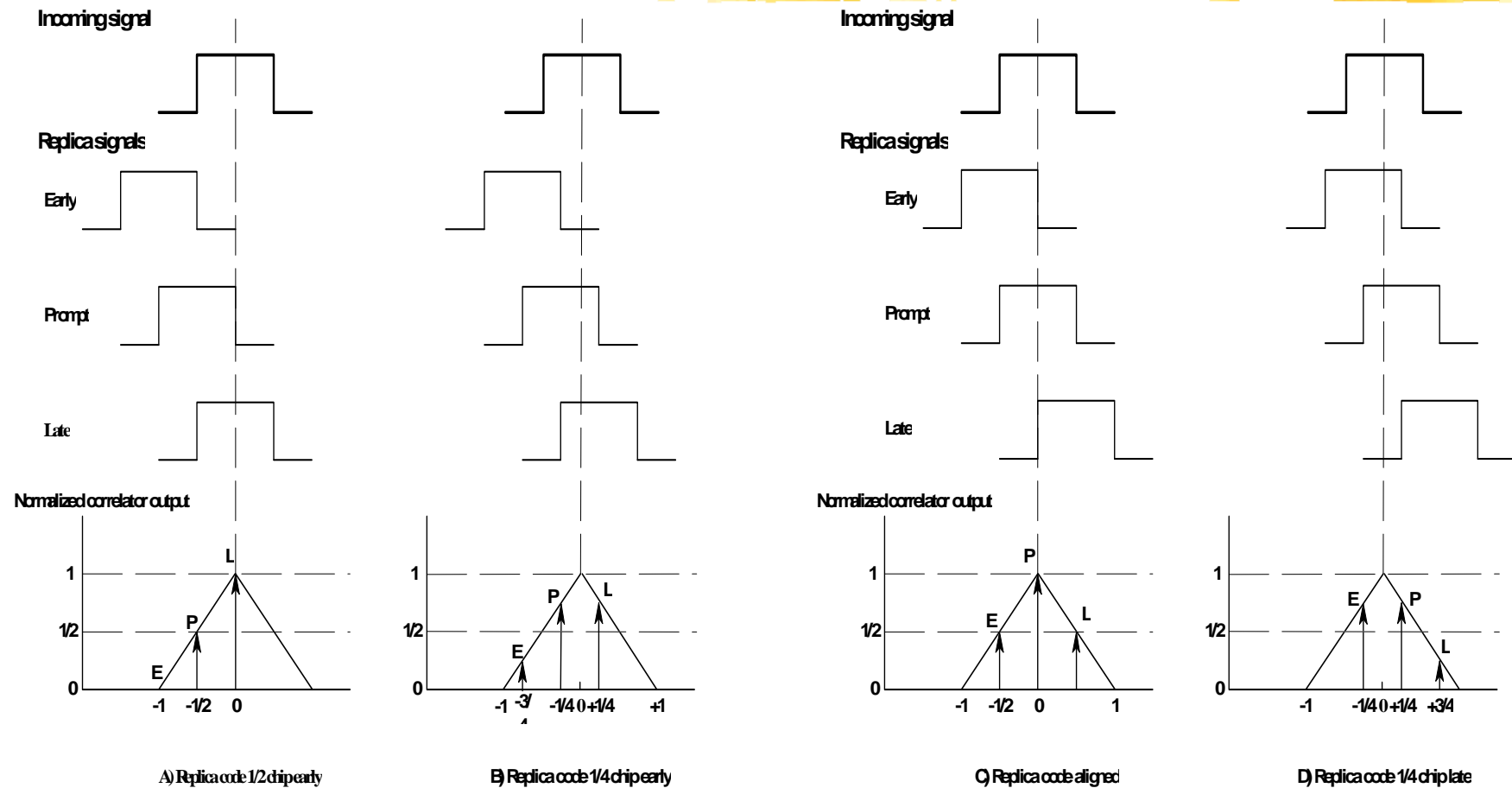
Code correlation process for three different replica code phases



Code correlation examples for replica code phase offsets of 1/2 chip early, 1/4 chip early, aligned and 1/4 chip late with respect to incoming signal

The four figures below (a), (b), (c), and (d), illustrate how the early, prompt and late envelopes change as the phases of the replica code signals are advanced with respect to the incoming SV signal. For ease of visualization, only one chip of the continuous PN signal is shown and the incoming SV signal is shown without noise. The next figure illustrates the normalized early minus late envelope discriminator error output signals corresponding to the four replica code offsets (a), (b), (c) and (d). The closed code loop operation becomes apparent as a result of studying these replica code phase changes, the envelopes that they produce, and the resulting error output generated by the early minus late envelope code discriminator. If the replica code is aligned, then the early and late envelopes are equal in amplitude and no error is generated by the discriminator. If the replica code is misaligned, then the early and late envelopes are unequal by an amount that is proportional to the amount of error (within the limits of the correlation interval). The code discriminator senses the amount of error in the replica code and the direction (early or late) from the difference in the amplitudes of the early and late envelopes. This error is filtered and then applied to the code loop NCO where the output frequency is increased or decreased as necessary to correct the replica code generator phase with respect to the incoming SV signal code phase.

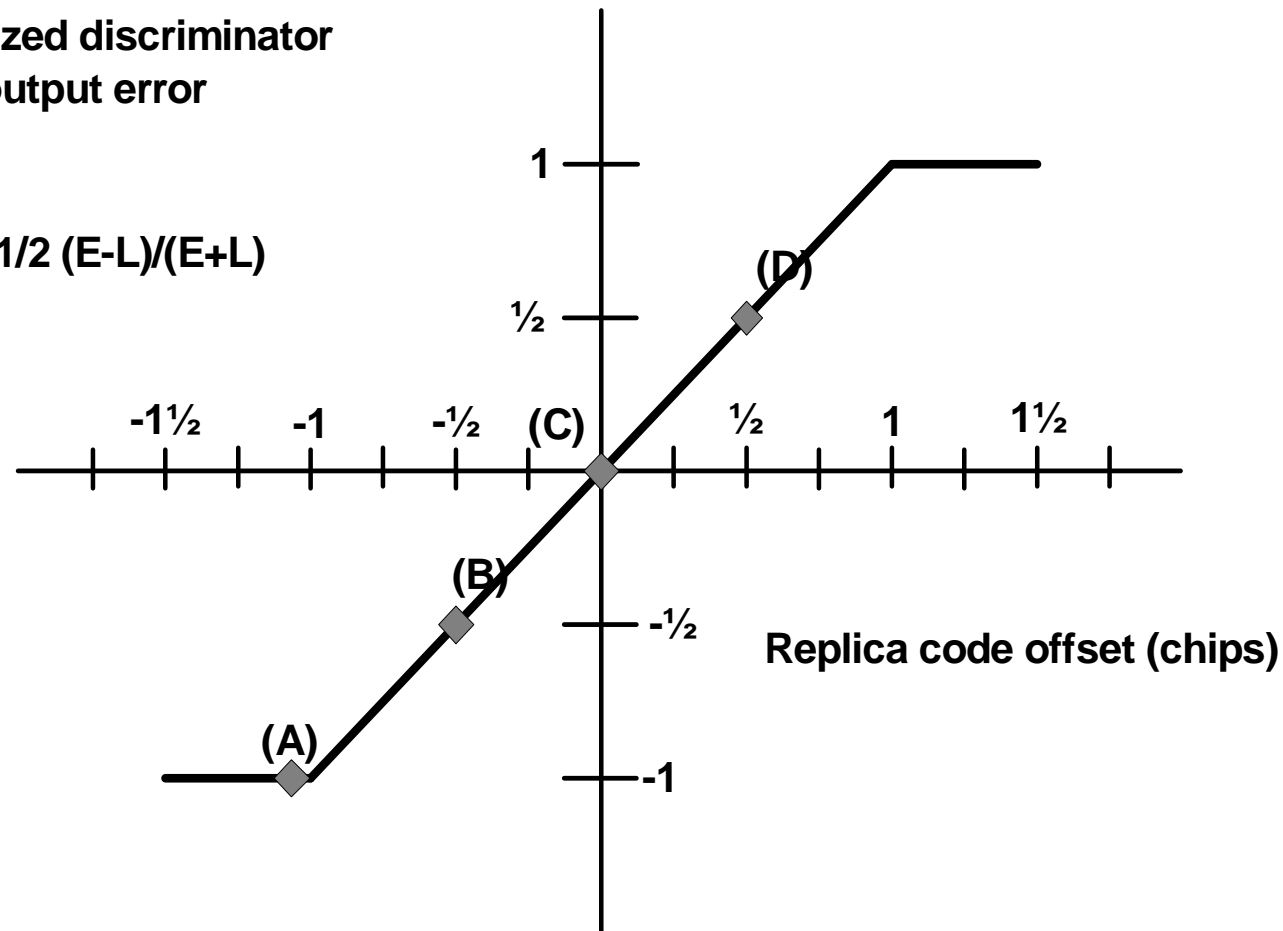
Correlation for replica codes: 1/2 chip early, 1/4 chip early, aligned and 1/4 chip late



Code discriminator output versus replica code offset

Normalized discriminator output error

$$\frac{1}{2} \frac{E-L}{E+L}$$



Session IV



Session IV

Carrier tracking loop design



- Generic GPS receiver carrier tracking loop block diagram
- Phase lock loops
- I,Q diagram depicting true PLL phase error
- Frequency lock loops
- I,Q diagram depicting true frequency error

Generic GPS receiver carrier tracking loop block diagram

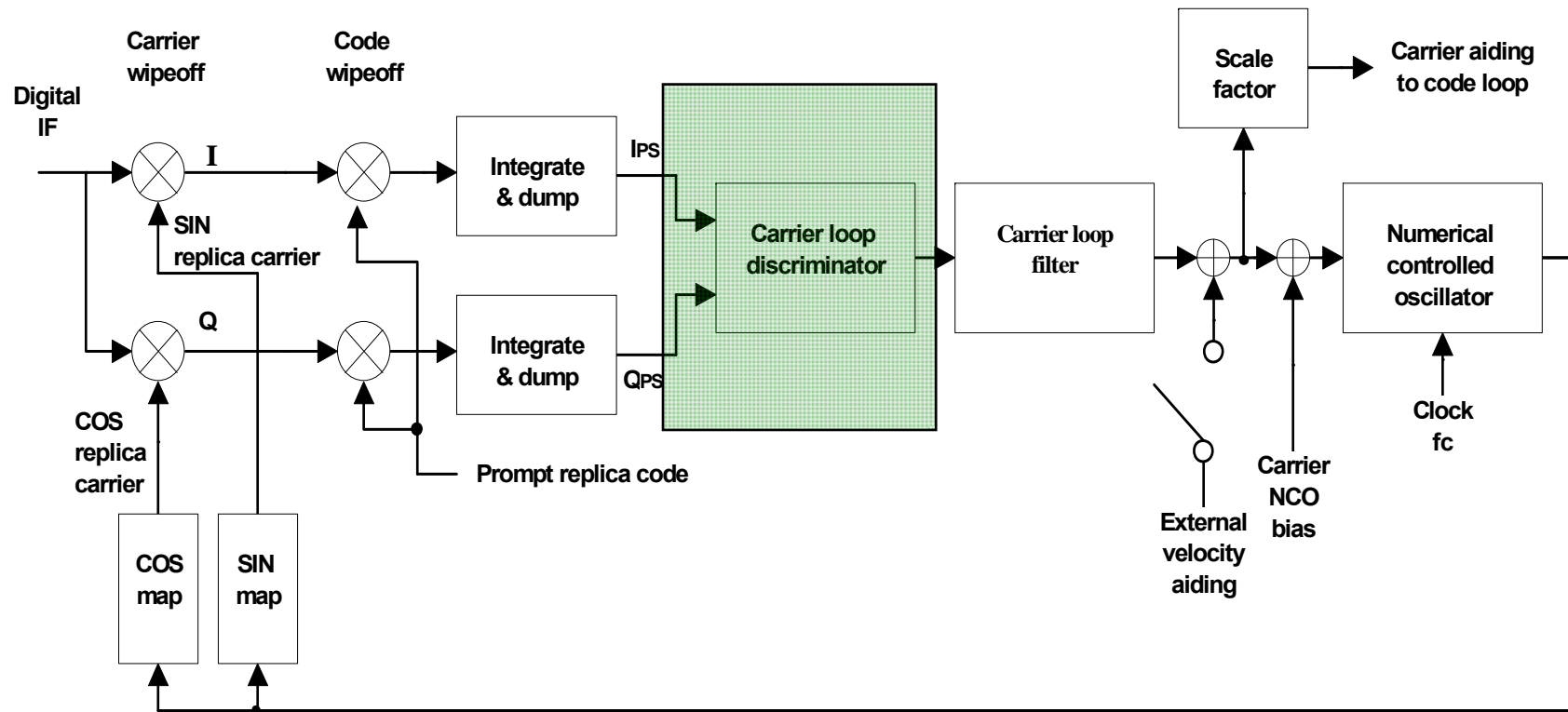
- Carrier tracking loop description spread out over several block diagrams
 - Carrier synthesis hardware
 - Carrier then code wipe-off hardware
 - Predetection integration hardware
 - Baseband software
 - Carrier loop discriminator and filter
- Discriminator defines carrier loop type
 - Phase or frequency lock loop (PLL or FLL)

Carrier tracking loops

The description of the carrier tracking loop has been spread out over several block diagrams. These consist of the carrier wipe-off hardware, the code wipe-off hardware, the predetection integration hardware (some additional predetection integration may take place in the software), the baseband software which consists of the carrier loop discriminator and the carrier loop filter and finally the carrier synthesis hardware, which consists of the carrier NCO and the sine and cosine map functions. All of the component parts of the carrier tracking loop are shown in the next figure.

The carrier loop discriminator defines the type of carrier tracking loop as a PLL, a Costas PLL (which is a PLL discriminator that tolerates the presence of data modulation on the baseband signal), or a frequency lock loop (FLL). The PLL and the Costas loops are the most accurate but are more sensitive to dynamic stress than the FLL. The PLL and Costas loop discriminators produce phase errors at their outputs. The FLL discriminator produces a frequency error. Because of this, there is also a difference in the architecture of the loop filter, described in Part II. There is an additional integration in the FLL versus the PLL filter for the same loop filter order.

Generic GPS receiver carrier tracking loop block diagram



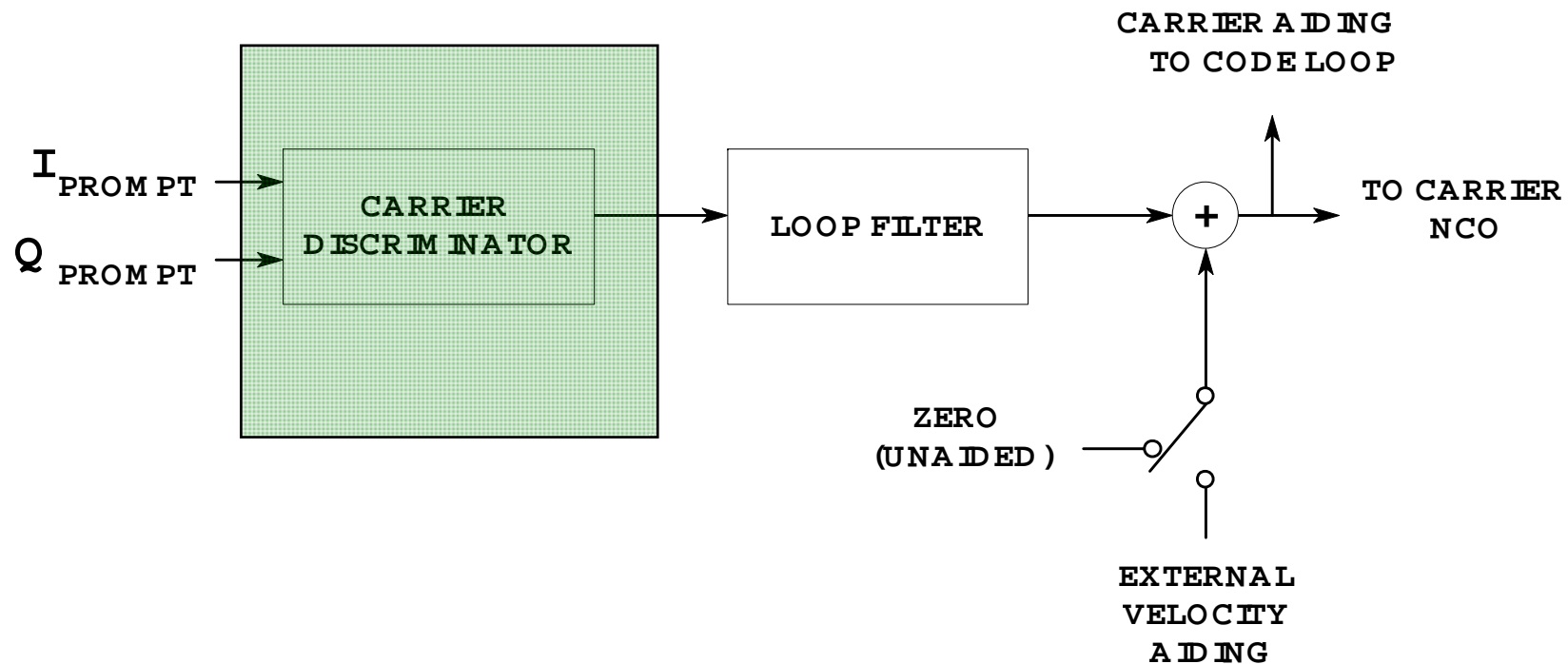
Baseband components of GPS receiver carrier tracking loop

The figure below illustrates the baseband components of the GPS receiver carrier tracking loop. The programmable designs of the carrier predetection integrators, the carrier loop discriminators and the carrier loop filters characterize the receiver carrier tracking loop. These three functions determine the two most important performance characteristics of the receiver carrier loop design: the carrier loop thermal noise error and the maximum line-of-sight dynamic stress threshold. Since the carrier tracking loop is always the weak link in a stand alone GPS receiver, its threshold characterizes the unaided GPS receiver performance.

The carrier loop discriminator defines the type of tracking loop as a pure phase lock loop (PLL) for dataless carrier tracking, a Costas-like PLL (which is a PLL type discriminator which tolerates the presence of data modulation on the baseband signal) or a frequency lock loop (FLL). The PLL is the most accurate but is more sensitive to dynamic stress than the FLL. The PLL discriminator produces a phase error. The FLL discriminator produces a frequency error. Because of this, there is also a difference in the architecture of the loop filter, described later.

There is a paradox that the GPS receiver designer must solve in the design of the predetection integration, discriminator and loop filter functions of the carrier tracking loops. To tolerate dynamic stress, the predetection integration time should be short, the discriminator should be an FLL and the carrier loop filter bandwidth should be wide. However, for the carrier Doppler phase measurements to be accurate (have low noise), the predetection integration time should be long, the discriminator should be a PLL and the carrier loop filter noise bandwidth should be narrow. In practice, some compromise must be made to resolve this paradox. A well-designed GPS receiver will close its carrier tracking loops with short predetection integration times, using an FLL and a wide band carrier loop filter. Then it will systematically transition into a Costas PLL with its predetection bandwidth and carrier tracking loop bandwidth set as narrow as the maximum anticipated dynamics permits.

Baseband components of GPS carrier tracking loop (PLL, FLL)



Designing PLL carrier tracking loop discriminators



- Phase lock loop (PLL) discriminators
- Common phase lock loop discriminators
- Comparison of PLL discriminators
- I,Q diagram depicting true PLL phase error

Common PLL discriminators

The table below summarizes five GPS receiver PLL discriminators, their output phase errors and their characteristics. The first four discriminators are insensitive to data transitions. The fourth one is the optimal one when data transitions are present. The first two are classical Costas discriminators. These were the first (analog) discriminators to be successfully used to implement PLLs with data modulation present. The last one is a pure PLL discriminator; i.e., it can operate only when there is no data modulation on the carrier. This is the four-quadrant ATAN function (ATAN2) PLL discriminator that remains linear over the full input error range of ± 180 degrees, whereas, the two-quadrant ATAN Costas discriminator remains linear over half of the input error range (± 90 degrees).

Note that digital technology now makes the optimal arctangent discriminators practical. However, the others are still used even though they are suboptimal owing to their simplicity.

Common phase lock loop discriminators

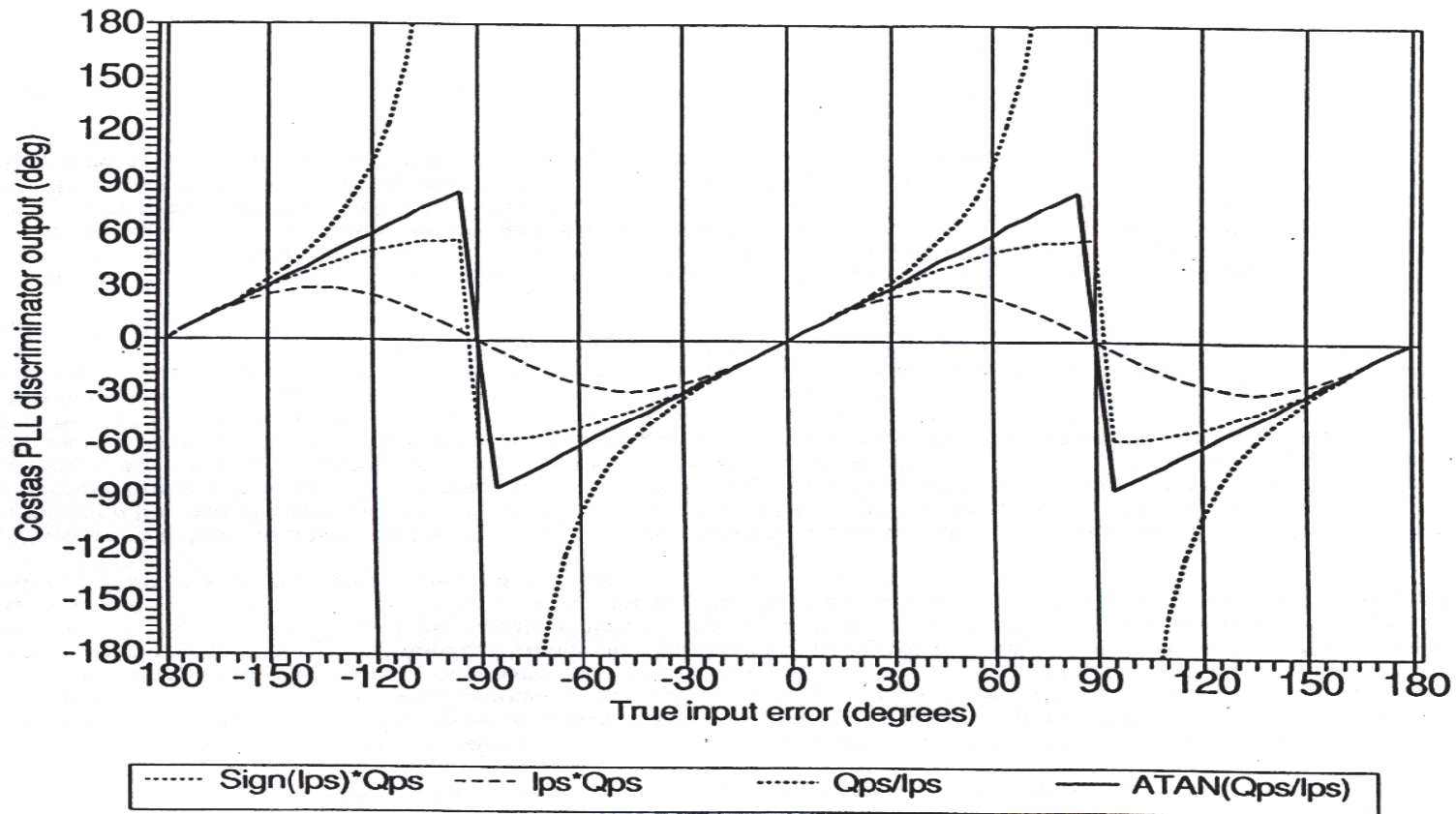
Discriminator algorithm	Output phase error	Characteristics (First four are data insensitive. Last is pure PLL)
Sign (I_{PS})*Q_{PS}	sin (φ)	Decision directed Costas. Near optimal at high SNR. Slope proportional to signal amplitude A. Least computational burden.
I_{PS}*Q_{PS}	sin (2φ)	Costas. Near optimal at low SNR. Slope proportional to signal amplitude squared A^2 . Moderate computational burden.
Q_{PS} / I_{PS}	tan (φ)	Suboptimal but good at high and low SNR. Slope not signal amplitude dependent. Higher computational burden and must check for divide by zero error near ± 90 degrees.
ATAN (Q_{PS}/I_{PS})	φ	Two-quadrant arctangent. Optimal (maximum likelihood estimator) at high and low SNR. Slope not signal amplitude dependent. Highest computational burden.
ATAN2 (Q_{PS}, I_{PS})	φ	Four-quadrant arctangent. Optimal (maximum likelihood estimator) at high and low SNR. Slope not signal amplitude dependent. Highest computational burden.

Comparison of PLL discriminators

The input/output relationships of the four PLL discriminators that are insensitive to data transitions in the previous table are plotted below. The true input error in degrees is plotted as the abscissa and the discriminator output error is plotted as the ordinate. Note that the output of all these discriminators repeat every 180 degrees (1/2 cycle). The ATAN 2 four-quadrant pure PLL discriminator input/output characteristic looks like the two-quadrant ATAN discriminator input/output characteristic, except that it repeats every 360 degrees (cycle). Therefore both its input and output range are doubled. Because the input error range is double the ATAN 2 discriminator, the pure PLL discriminator has 6 dB more noise error tolerance.

The ideal input/output relationship is linear. Therefore, only the ATAN discriminator is optimal. All of the others are simply approximations to minimize the computational burden. As a result, their discriminator performance is suboptimal. Their advantages and limitations are summarized in the table below.

Comparison of PLL discriminators



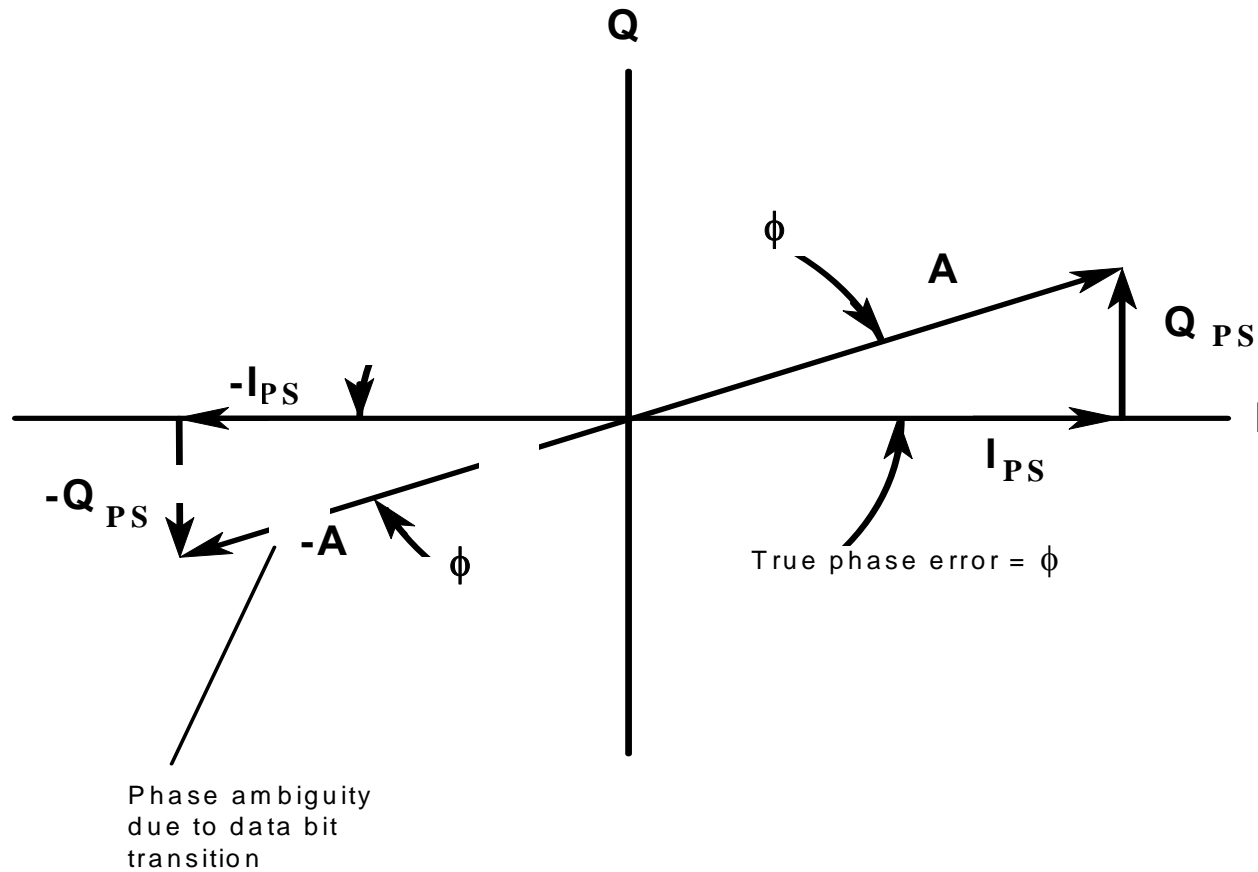
I, Q phasor diagram depicting true phase error between replica and incoming carrier phase

Referring to the carrier wipe-off process in the carrier tracking loop block diagram of the generic GPS receiver carrier tracking loop, and assuming that the carrier loop is in phase lock, the replica sine function is in-phase with the incoming SV carrier signal (converted to IF). This results in a \sin^2 product at the I output, which, when averaged, produces the maximum I_{PS} amplitude. The replica cosine function is 90 degrees out-of-phase with the incoming SV carrier. This results in a $(\cosine)(sine)$ product at the Q output, which, when averaged, produces the minimum Q_{PS} amplitude. For this reason, I_{PS} will be near its maximum (and will flip 180 degrees each time the data bit changes sign) and Q_{PS} will be near its minimum (and will also flip 180 degrees each time the data bit changes sign). These PLL characteristics are illustrated in the figure below where the phasor, A, (the vector sum of I_{PS} and Q_{PS}) tends to remain aligned with the I-axis and switches 180 degrees during each data bit reversal.

It is straightforward to detect the bits in the SV data message stream using a PLL discriminator that is insensitive to data modulation, hereafter called a Costas PLL. The I_{PS} samples are simply accumulated for one data bit interval and the sign of the result is the data bit. Since there is a 180-degree phase ambiguity with a Costas PLL, the detected data bit stream may be normal or inverted. This ambiguity is resolved during the frame synchronization process by comparing the known preamble at the beginning of each subframe both ways (normal and inverted) with the bit stream. If a match is found with the preamble pattern inverted, the bit stream is inverted and the subframe synchronization is confirmed by parity checks on the telemetry (TLM) and handover word (HOW). Otherwise, the bit stream is normal. Once the phase ambiguity is resolved, it remains resolved until the PLL loses phase lock or slips cycles. If this happens, the ambiguity must be resolved again. The 180-degree ambiguity of the Costas PLL can be resolved by referring to the phase detection result of the data bit demodulation. If the data bit phase is normal, then the carrier Doppler phase indicated by the Costas PLL is correct. If the data bit phase is inverted, then the carrier Doppler phase indicated by the Costas PLL phase can be corrected by adding 180 degrees.

Costas PLLs as well as conventional PLLs are sensitive to dynamic stress, but produce the most accurate velocity measurements. They also provide the most error-free data demodulation. Therefore, this is the desired steady state tracking mode of the GPS receiver carrier tracking loop. However, a well designed GPS receiver carrier tracking loop will close the loop with a more robust FLL operated at wideband. Then it will gradually reduce the carrier tracking loop bandwidth and transition into a wideband PLL operation. Finally, it will narrow the PLL bandwidth to the steady state mode of operation. If dynamic stress causes the PLL to lose lock, the receiver will detect this with a sensitive phase lock detector and transition back to the FLL. The PLL closure process is then repeated.

PLL I, Q phasor diagram: Phase error between replica carrier and incoming carrier



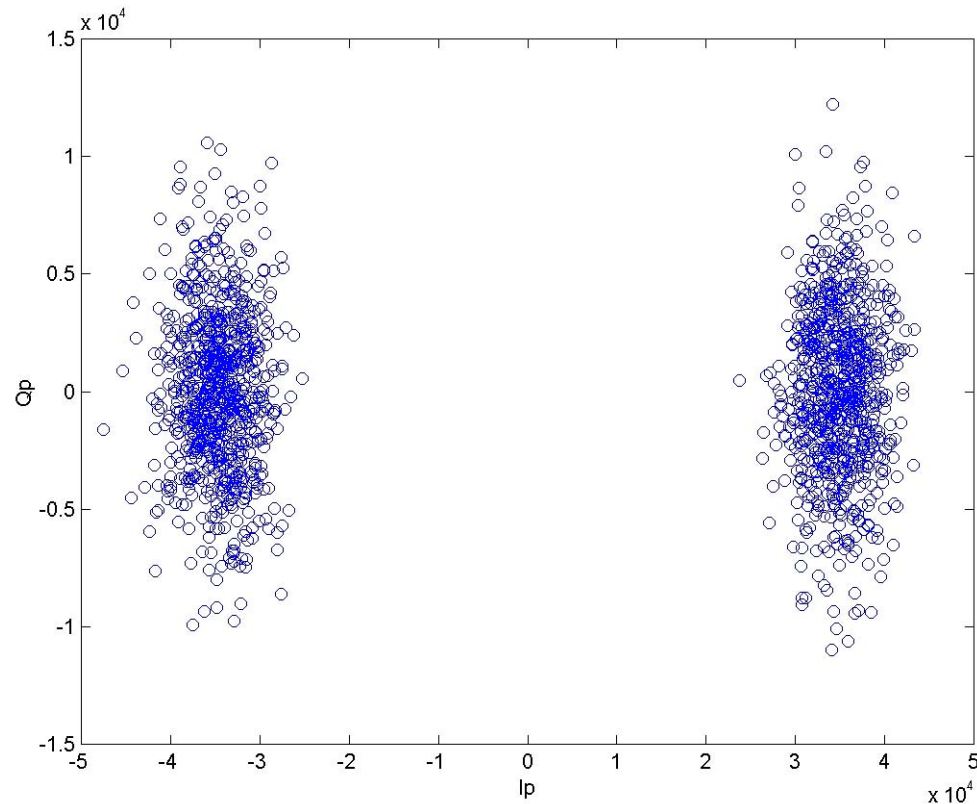
I,Q “fuzz ball” of PLL signal with data plus noise

Below is a photograph of an oscilloscope monitoring the vector sum of the prompt I and Q signals (Q on the vertical axis input and I on the horizontal axis input). These signals are monitored just prior to the PLL discriminator stage while the carrier loop is in phase lock. The oscilloscope displays the vector sum of the noisy I and Q signals as being essentially stationary along the I-axis and switching 180 degrees every time the navigation message data bits change sign.

When the carrier loop has just closed, these “fuzz balls” rotate at a rate proportional to the frequency difference between the replica carrier and the incoming SV carrier frequency.

When the carrier loop achieves frequency lock, these “fuzz balls” remain stationary but at some arbitrary phase angle with respect to the I-axis.

I,Q “fuzz ball” of PLL signal with data plus noise



Phase lock loops

A phase lock loop (PLL) in the GPS receiver replicates and tracks the SV carrier phase. Since there is data modulation present after the carrier and code wipe-off processes, a PLL discriminator that is insensitive to data transitions must normally be used. This is usually termed a Costas loop since Costas was the first to devise such a discriminator. If there were no 50 Hz data modulation on the GPS signal, the carrier tracking loop could use a pure PLL discriminator. For example, a P(Y)-code receiver could implement a pure PLL discriminator for use in the L2 carrier tracking mode if the Control Segment turns off data modulation. Although this mode is specified as a possibility, it is unlikely to be activated. This mode is specified in ICD-GPS-200 [1] because pure PLL operation permits the predetection integration time to increase beyond the 20-milliseconds limitation imposed by the 50 Hz navigation data. This reduces the squaring loss. If a four quadrant arctangent PLL discriminator is used, then the linear phase detection range is doubled thereby improving the carrier signal tracking threshold by 6 dB.

Since this is such a significant improvement, many techniques have been studied to exploit dataless carrier tracking. One example is a short-term pure PLL mode called data wipeoff. The GPS receiver typically acquires a complete copy of the full navigation message after 25 iterations of the five subframes (12.5 minutes). The receiver then can compute the navigation message sequence until the GPS Control Segment uploads a new message or until the SV changes the message. Until the message changes significantly, the GPS receiver can perform data wipeoff of each bit of the incoming 50 Hz navigation data message and use a pure PLL discriminator. The receiver baseband processing function does this by reversing the sign of the integrated in-phase components (I_{ES} , I_{PS} , I_{LS}) in accordance with a consistent algorithm. For example, if I_{PS} has a predetection integration time of 5-ms, then there are four samples of I_{PS} between each SV data bit transition that will have the same sign. This sign will be the sign of the data bit known by the receiver a-priori for that data interval. Each 5 millisecond sample may fluctuate in sign due to noise. If the known data bit for this interval is a "0" then the data wipeoff process does nothing to all four samples. If the known data bit for this interval is a "1" then the sign is reversed on all four samples

Pure phase lock loops

- Pure PLL discriminator provides 6 dB improvement but cannot be used with C/A or P(Y) code
 - 50 Hz data requires data insensitive PLL discriminator
 - Pure PLL can be used for data-less L2 P(Y) code, but this SV mode not likely to be turned on by Control Segment
- Data wipeoff can provide short term pure PLL mode – a hold on by your teeth mode (HOBYT)
 - Receiver reads full navigation message after 25 iterations of the five subframes
 - Requires 12.5 minutes - thereafter, data wipeoff works until SV message changes
 - Alternative: navigation message provided by external source

Designing FLL carrier tracking loop discriminators



- Frequency lock loop (FLL) discriminators
- Common FLL discriminators
- Comparison of FLL discriminators
- I,Q diagram depicting true frequency error

Frequency lock loops (FLLs)

FLLs replicate the approximate frequency of the incoming SV. For this reason, they are also called automatic frequency control (AFC) loops. The FLLs of GPS receivers must be insensitive to 180-degree reversals in the I and Q signals. Therefore, the sample times of the I and Q signals should not straddle the data bit transitions. During initial signal acquisition, when the receiver does not know where the data transition boundaries are, it is usually easier to maintain frequency lock than phase lock with the SV signal while performing bit synchronization. This is because the FLL discriminators are less sensitive to situations where some of the I and Q signals do straddle the data bit transitions, especially when the predetection integration times are small compared to the data bit transition intervals.

Frequency lock loops (FLLs)

- FLLs replicate frequency of SV carrier
- Also called automatic frequency control (AFC) loops
- GPS FLLs must be insensitive to 180-degree reversals in I and Q signals
- Sample times of I and Q signals should not straddle data bit transitions
- Receiver does not know phase of data transition boundaries during initial signal acquisition - FLL is less sensitive than PLL

Common frequency lock loop discriminators

The table below summarizes several GPS receiver FLL discriminators, their output frequency errors and their characteristics.

Note that the integrated and dumped prompt samples I_{PS1} and Q_{PS1} are the samples taken at time t_1 , just prior to the samples I_{PS2} and Q_{PS2} taken at a later time t_2 . These two adjacent samples should be within the same data bit interval. The next pair of samples are taken starting $(t_2 - t_1)$ seconds after t_2 , etc.

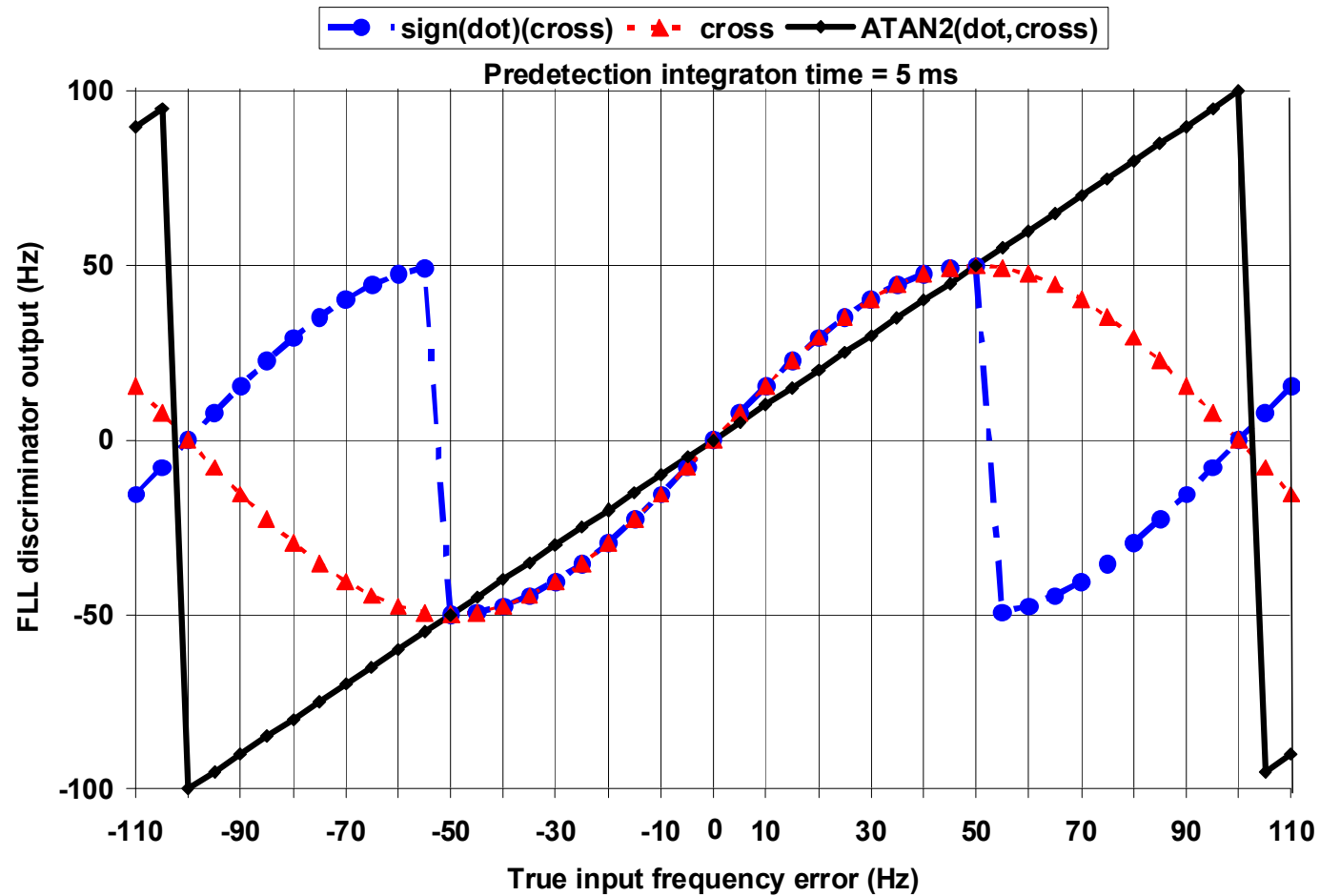
Common frequency lock loop discriminators

Discriminator algorithm	Output frequency error	Characteristics
$\frac{\text{sign}(\dot{d}t) \text{ cross}}{t_2 - t_1}$ <p>where: $\dot{d}t = I_{PS1} * I_{PS2} + Q_{PS1} * Q_{PS2}$ $\text{cross} = I_{PS1} * Q_{PS2} - I_{PS2} * Q_{PS1}$</p>	$\frac{\sin [2(\phi_2 - \phi_1)]}{t_2 - t_1}$	Near optimal at high SNR. Slope proportional to signal amplitude A. Moderate computational burden.
$\frac{\text{cross}}{t_2 - t_1}$	$\frac{\sin (\phi_2 - \phi_1)}{t_2 - t_1}$	Near optimal at low SNR. Slope proportional to signal amplitude squared A ² . Least computational burden.
$\frac{\text{ATAN2}(\text{cross}, \dot{d}t)}{(t_2 - t_1) 360}$	$\frac{\phi_2 - \phi_1}{(t_2 - t_1) 360}$	Four-quadrant arctangent. Maximum likelihood estimator. Optimal at high and low SNR. Slope not signal amplitude dependent. Highest computational burden.

Comparison of frequency lock loop discriminators

The two figures below compare the frequency error outputs of each of these discriminators assuming no noise in the I_{PS} and Q_{PS} samples. The (a) figure illustrates that the frequency pull-in range with a 200 Hz predetection bandwidth is twice the range of the (b) figure with 100 Hz. Adjacent pairs of samples ($t_2 - t_1$) are taken every 5 milliseconds and 10 milliseconds, respectively. Note in both figures that the single-sided frequency pull-in ranges of the cross and ATAN2 (cross, dot) FLL discriminators are equal to half the predetection bandwidths. The sign(dot)*cross FLL discriminator frequency pull-in ranges are only one-fourth of the predetection bandwidths. Also note that the sign(dot)*cross and the cross FLL discriminator outputs, whose outputs are sine functions scaled by the sample time interval ($t_2 - t_1$) in their denominators, would more accurately approximate true frequency error if they were divided by four. The ATAN2 (cross, dot) discriminator, whose output is a linear function scaled by $(t_2 - t_1) \cdot 360$ in the denominator, produces a true representation of the input frequency error within its pull-in range. The amplitudes of the discriminator outputs are reduced (their slopes tend to flatten) and they tend to start rounding off near the limits of their pull-in range as the thermal noise levels increase.

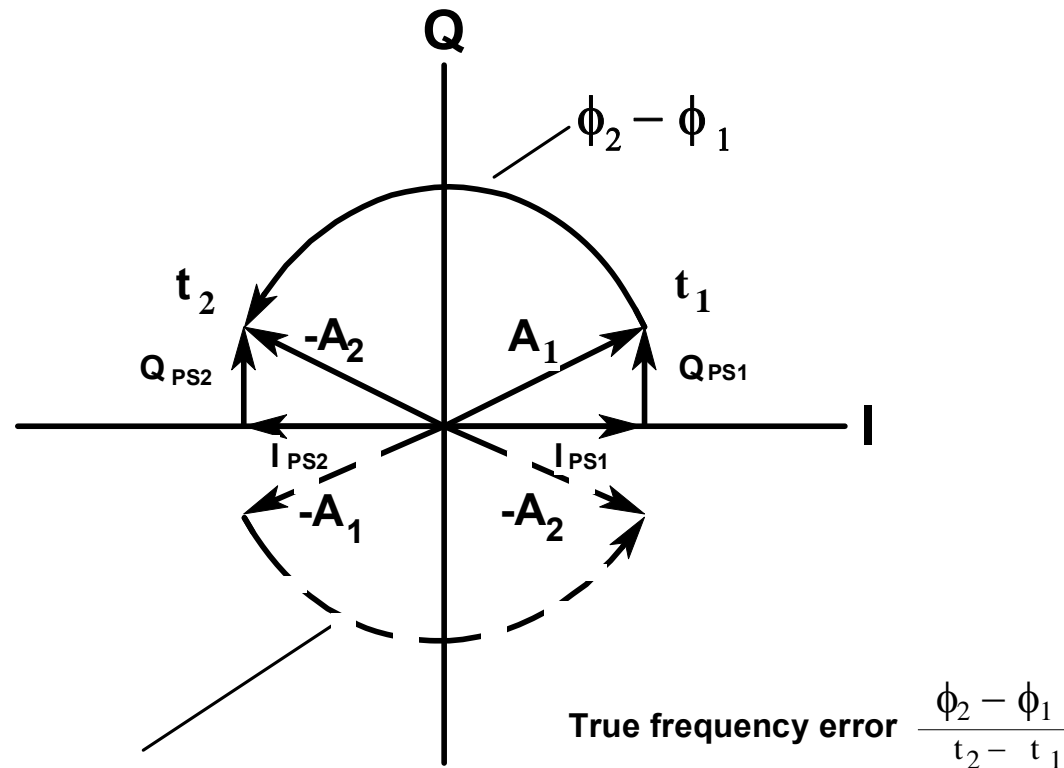
Comparison of FLL discriminators



I, Q phasor diagram depicting true frequency

The I, Q phasor diagram in the figure below depicts the change in phase, $\theta_2 - \theta_1$, between two adjacent samples of I_{PS} and Q_{PS} at times t_1 and t_2 . This phase change over a fixed time interval is proportional to the frequency error in the carrier tracking loop. The figure also illustrates that there is no frequency ambiguity in the GPS receiver FLL discriminator because of data transitions provided that the adjacent I and Q samples are taken within the same data bit interval. Note that the phasor, A, which is the vector sum of I_{PS} and Q_{PS} , remains constant and rotates at a rate directly proportional to the frequency error when the loop is in frequency lock. It is possible to demodulate the SV data bit stream in FLL. Detecting the data transitions is more complicated in FLL and the bit error rate is higher than in PLL. This is because detecting the change in sign of the phasor in the differential FLL is more complicated and noisier than detecting the sign of the integrated I_{PS} in a PLL.

FLL I, Q phasor diagram: frequency error between replica carrier and incoming carrier



No frequency ambiguity due to data bit transition (unless samples are split)

Session V



Session V

Code and carrier loops filter design

- Loop filter overview
- First, second, and third order analog loop filters
- Loop filter design characteristics
- Replacing analog with digital integrators
- First, second, and third order digital loop filters
- Block diagrams of two FLL-assisted PLL filters
- Loop filter parameter design example

Loop filter overview

The objective of the loop filter is to reduce noise in order to produce an accurate estimate of the original signal at its output. As shown in the receiver block diagram, the loop filter's output signal is mixed with the original signal (plus noise) to produce an error signal which is fed back into the filter's input in a closed loop process. There are many design approaches to digital filters. The design approach described here draws on existing knowledge of analog loop filters, then adapts these into digital implementations.

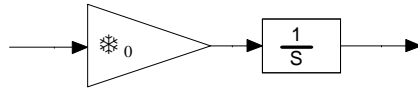
Loop filter overview

- Objective of loop filter is to reduce noise
 - Produce accurate estimate of original signal (without noise) at its output
- Loop filter's output signal used as an error signal that is fed back to the input stages (carrier replica and wipeoff, code replica and correlation) in a closed loop process
- Digital filter design approach draws on existing knowledge of analog loop filters, then adapts these into digital implementations

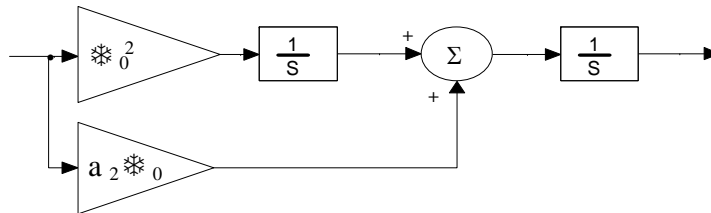
First, second and third order analog loop filters

The figure below shows block diagrams of first, second and third order analog filters. Analog integrators are represented by $1/s$, the Laplace transform of the time domain integration function. The number of integrators determine the loop filter order. The first order loop filter has one integrator, etc. The input signal is multiplied by the multiplier coefficients, then processed as shown in the figure. These multiplier coefficients and the number of integrators completely determine the loop filter's characteristics.

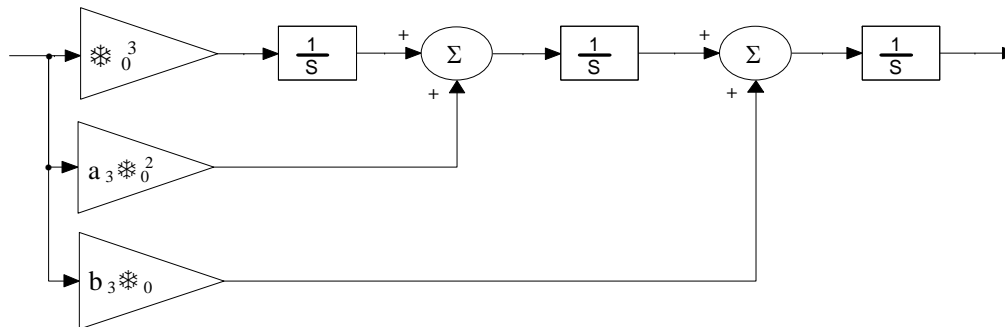
First, second, and third order analog loop filters



(a) First order analog filter.



(b) Second order analog filter.



(c) Third order analog filter.

Loop filter design characteristics

The table below summarizes the filter characteristics and provides all the information required to compute the filter coefficients for first, second or third order loop filters. Only the filter order and noise bandwidth must be determined to complete the design.

Notes: (1) The loop filter natural radian frequency, ω_0 , is computed from the value of the loop filter noise bandwidth, B_n , selected by the designer. (2) R is the line-of-sight range to the satellite. (3) The steady state error is inversely proportional to the tracking loop bandwidth and directly proportional to the n th derivative of range, where n is the loop filter order.

Loop filter design characteristics

Loop order	Noise bandwidth B_n (Hz)	Typical filter values	Steady state error	Characteristics
First	$\frac{\omega_o}{4}$	ω_o $B_n = 0.25 \omega_o$	$\frac{(dR/dt)}{\omega_o}$	Sensitive to velocity stress. Used in aided code loops. Unconditionally stable at all noise bandwidths.
Second	$\frac{\omega_o(1 + a_2^2)}{4a_2}$	ω_o^2 $a_2 \omega_o = 1.414 \omega_o$ $B_n = 0.53 \omega_o$	$\frac{(d^2R/dt^2)}{\omega_o^2}$	Sensitive to acceleration stress. Used in aided and unaided carrier loops. Unconditionally stable at all noise bandwidths.
Third	$\frac{\omega_o(a_3b_3^2 + a_3^2 - b_3)}{4(a_3b_3 - 1)}$	ω_o^3 $a_3 \omega_o^2 = 1.1 \omega_o^2$ $b_3 \omega_o = 2.4 \omega_o$ $B_n = 0.7845 \omega_o$	$\frac{(d^3R/dt^3)}{\omega_o^3}$	Sensitive to jerk stress. Used in unaided carrier loops. Remains stable at $B_n \leq 18$ Hz.

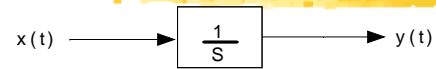
Replacing analog with digital integrators

The figure below depicts the block diagram representations of analog and digital integrators. The analog integrator of Figure (a) operates with a continuous time domain input, $x(t)$, and produces an integrated version of this input as a continuous time domain output, $y(t)$. Theoretically, $x(t)$ and $y(t)$ have infinite numerical resolution and the integration process is perfect. In reality, the resolution is limited by noise which significantly reduces the dynamic range of analog integrators. There are also problems with drift.

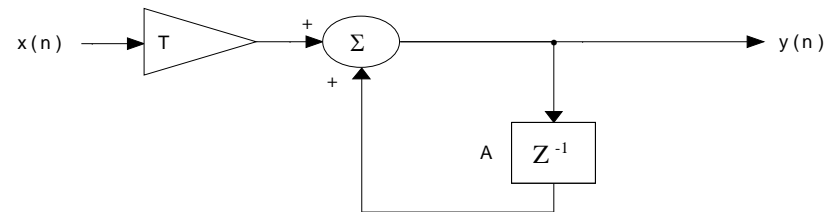
The boxcar digital integrator of Figure (b) operates with a sampled time domain input, $x(n)$, which is quantized to a finite resolution, and produces a discrete integrated output, $y(n)$. The time interval between each sample, T , represents a unit delay, z^{-1} , in the digital integrator. The digital integrator performs discrete integration perfectly with a dynamic range limited only by the number of bits used in the accumulator, A . This provides a dynamic range capability much greater than can be achieved by its analog counterpart and the digital integrator does not drift. The boxcar integrator performs the function $y(n) = T[x(n)] + A(n-1)$, where n is the discrete sample sequence number.

Figure (c) depicts a digital integrator which linearly interpolates between input samples and more closely approximates the ideal analog integrator. This is called the bilinear z-transform integrator. It performs the function $y(n) = T/2[x(n)] + A(n-1) = 1/2[A(n) + A(n-1)]$.

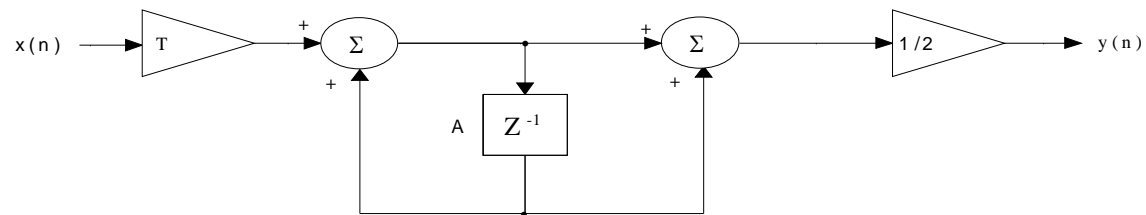
Replacing analog with digital integrators



(a) Analog integrator



(b) Digital boxcar integrator

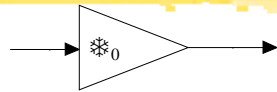


(c) Digital bilinear transform integrator

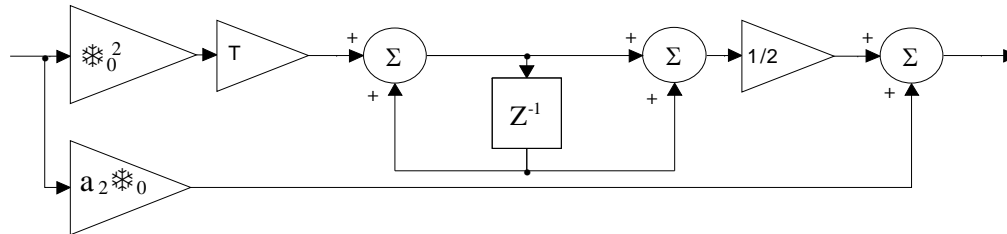
Block diagram of three digital loop filters

The digital filters depicted in the figure below result when the Laplace integrators of the analog loop filters shown in the earlier figure are replaced with bilinear z-transform integrators. The last digital integrator, which is the NCO, is not shown to emphasize the fact that the NCO is implemented separately from the programmable loop filter. The NCO is characterized as a boxcar integrator

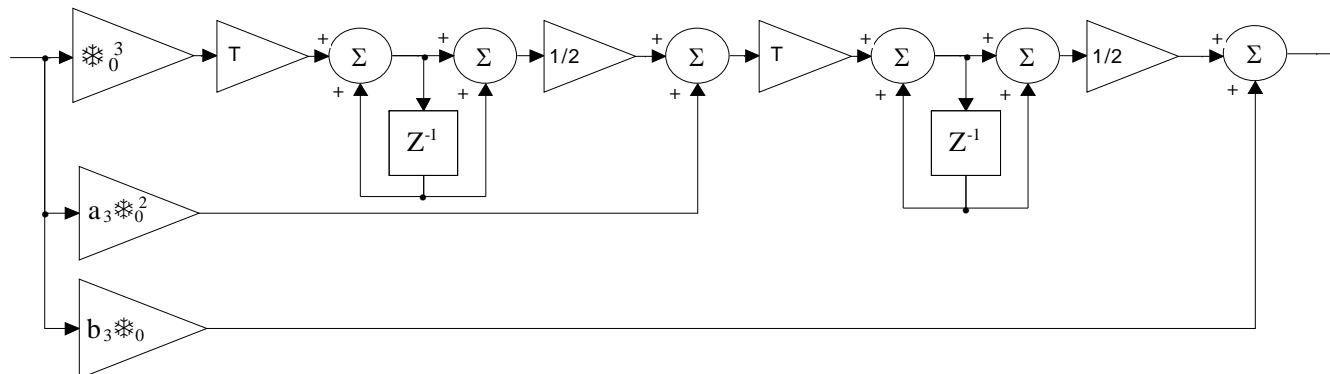
First, second, and third order digital loop filters



(a) First order digital filter.



(b) Second order digital filter.

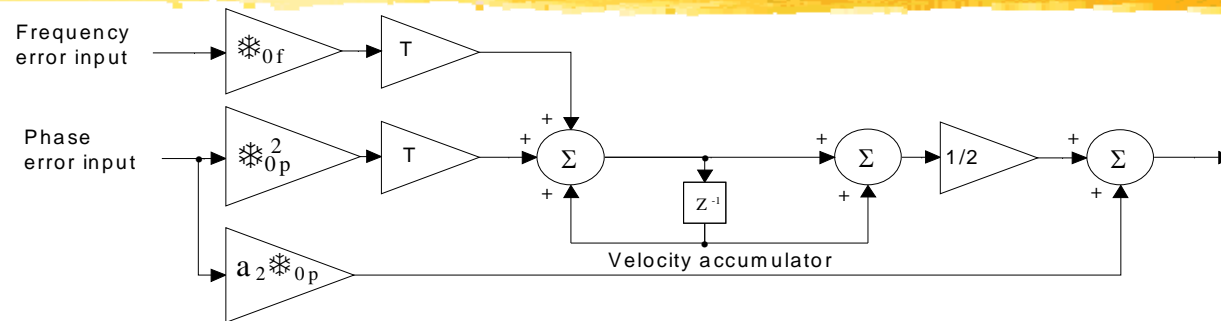


(c) Third order digital filter.

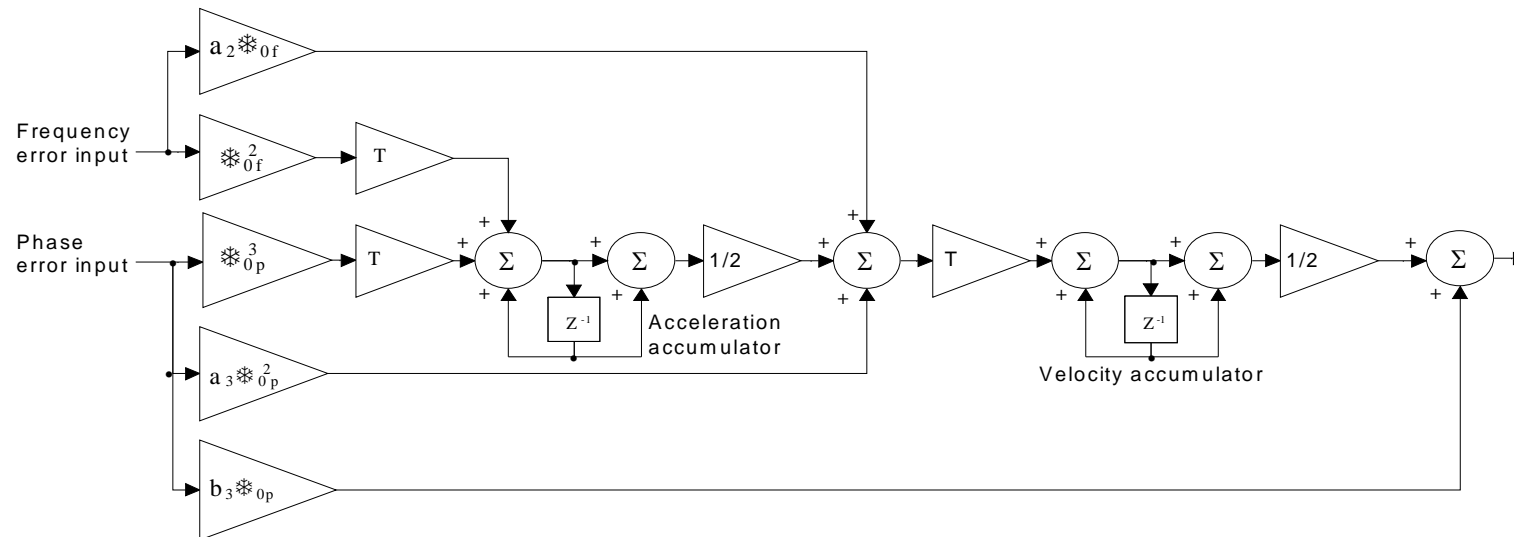
Block diagrams of two FLL-assisted PLL filters

The figure below illustrates two FLL assisted PLL loop filter designs. Figure (A) depicts a second order PLL filter with a first order FLL assist. Figure (B) depicts a third order PLL filter with a second order FLL assist. If the PLL error input is zeroed in either of these filters, the filter becomes a pure FLL. Similarly, if the FLL error input is zeroed, the filter becomes a pure PLL. The typical loop closure process is to close in pure FLL, then apply the error inputs from both discriminators as an FLL assisted PLL until phase lock is achieved, then convert to pure PLL until phase lock is lost. In general, the natural radian frequency of the FLL, ω_{of} , is different from the natural radian frequency of the PLL, ω_{op} . These natural radian frequencies are determined from the desired loop filter noise bandwidths, B_{nf} and B_{np} , respectively. The values for the second order coefficient a_2 and third order coefficients a_3 and b_3 can be determined from the table of loop filter characteristics shown earlier. These coefficients are the same for FLL, PLL or DLL applications if the loop order and the noise bandwidth, B_n , are the same. Note that the FLL coefficient insertion point into the filter is one integrator back from the PLL and DLL insertion points. This is because the FLL error is in units of Hz (change in range per unit of time), whereas the PLL and DLL errors are in units of phase (range).

Block diagrams of two FLL-assisted PLL filters



(a) Second order PLL filter with first order FLL assist.



(b) Third order PLL filter with second order FLL assist.

Loop filter parameter design example

A loop filter parameter design example will clarify the use of the equations in the loop filter characteristics table. Suppose that the receiver carrier tracking loop will be subjected to high acceleration dynamics and will not be aided by an external navigation system, but must maintain PLL operation. A third order loop is selected because it is insensitive to acceleration stress. To minimize its sensitivity to jerk stress, the noise bandwidth, B_n , is chosen to be the widest possible consistent with stability. The table indicates that $B_n \leq 18$ Hz is safe. This limitation has been determined through extensive Monte Carlo simulations and is related to the maximum predetection integration time (which is typically the same as the reciprocal of the carrier loop iteration rate). If $B_n = 18$ Hz, then $\sigma_o = B_n/0.7845 = 22.94$ radians/s.

The three multipliers shown in Figure(c) of the digital loop filters block diagram (the third order filter) are computed as follows: $\omega_p^3 = 12079.214$, $a_3 \omega_p^2 = 1.1 \omega_p^2 = 579.098$, $b_3 \omega_p = 2.4 \omega_p = 55.067$.

If the loop iteration rate is 200 Hz, then $T = 0.005$ seconds for use in the digital integrators. This completes the third order filter parameter design. The remainder of the loop filter design is the implementation of the digital integrator accumulators to ensure that they will never overflow; i.e., that they have adequate dynamic range. The use of floating point arithmetic in modern microprocessors with built-in floating point hardware greatly simplifies this part of the design process.

Loop filter parameter design example

- Assume selection of third order loop because it is insensitive to acceleration stress
 - Minimize sensitivity to jerk stress by choosing widest possible noise bandwidth, consistent with stability: $B_n = 18$ Hz
- For noise bandwidth, $B_n = 18$ Hz
 - $\omega_0 = B_n/0.7845 = 22.94455$ radians/s
 - Multipliers: $\omega_0^3 = 12079.214$, $a_3 \omega_0^2 = 1.1 \omega_0^2 = 579.098$, $b_3 \omega_0 = 2.4 \omega_0 = 55.067$
 - For 200 Hz loop iteration rate: $T = 0.005$ s

Session VI - Code



Session VI - Code

Extracting measurements from code and carrier loops

- Extracting measurements from code loop
 - Pseudorange definition
 - Satellite transmit time relationship to code phase
 - Pseudorange measurement
 - Relationship between PRN code generator and code accumulator
 - Measurement time skew
 - Code (time) accumulator
 - Maintaining the code accumulator
 - Obtaining a measurement from the code accumulator
 - Obtaining transmit time from C/A code
 - Synchronizing code accumulator to replica code generator
 - Adding a code setter to code generator
 - C/A-code setup
 - Obtaining transmit time from the C/A-code
 - GPS C/A-code timing relationships
 - GPS navigation message
 - Example of bit sync error in C/A-code

Pseudorange definition

Contrary to popular belief, the natural measurements of a GPS receiver are not pseudorange or delta pseudorange. This section describes these natural measurements of a GPS receiver and describes how they may be converted into pseudorange, delta pseudorange and integrated Doppler. The natural measurements are replica code phase and replica carrier Doppler phase (if the GPS receiver is in phase lock with the satellite carrier signal) or replica carrier Doppler frequency (if the receiver is in frequency lock with the satellite carrier signal). The replica code phase can be converted into satellite transmit time which can be used to compute the pseudorange measurement. The replica carrier Doppler phase or frequency can be converted into delta pseudorange. The replica carrier Doppler phase measurements can be converted into the so-called integrated carrier Doppler phase measurements used for ultra-precise static and kinematic surveying or positioning.

The most important concept presented in this section is the measurement relationship between the replica code phase state in the GPS receiver and the satellite transmit time. This relationship is unambiguous for P(Y)-code, but can be ambiguous for C/A-code. Every C/A-code GPS receiver is vulnerable to this ambiguity problem and, under weak signal acquisition conditions, the ambiguity will occur. When the ambiguity does occur in a C/A-code GPS receiver, it causes serious range measurement errors, which, in turn, result in severe navigation position errors.

The definition of pseudorange to SV_i where i is the PRN number is as follows:

$$PR_i(n) = c [T_R(n) - T_{Ti}(n)] \quad (\text{meters})$$

where: c = GPS propagation constant = 2.99792458×10^8 (meters/second)
 $T_R(n)$ = receive time corresponding to epoch n of the GPS receiver's clock (seconds)
 $T_{Ti}(n)$ = transmit time based on the SV_i clock (seconds).

Pseudorange definition

■ Pseudorange to SV_i where i is PRN number:

- $PR_i(n) = c [T_R(n) - T_{Ti}(n)]$ (meters)
where: $c =$ GPS propagation constant
 $= 2.99792458 \times 10^8$ (m/s)

- $T_R(n) =$ receive time at epoch n of the GPS receiver's clock (seconds)

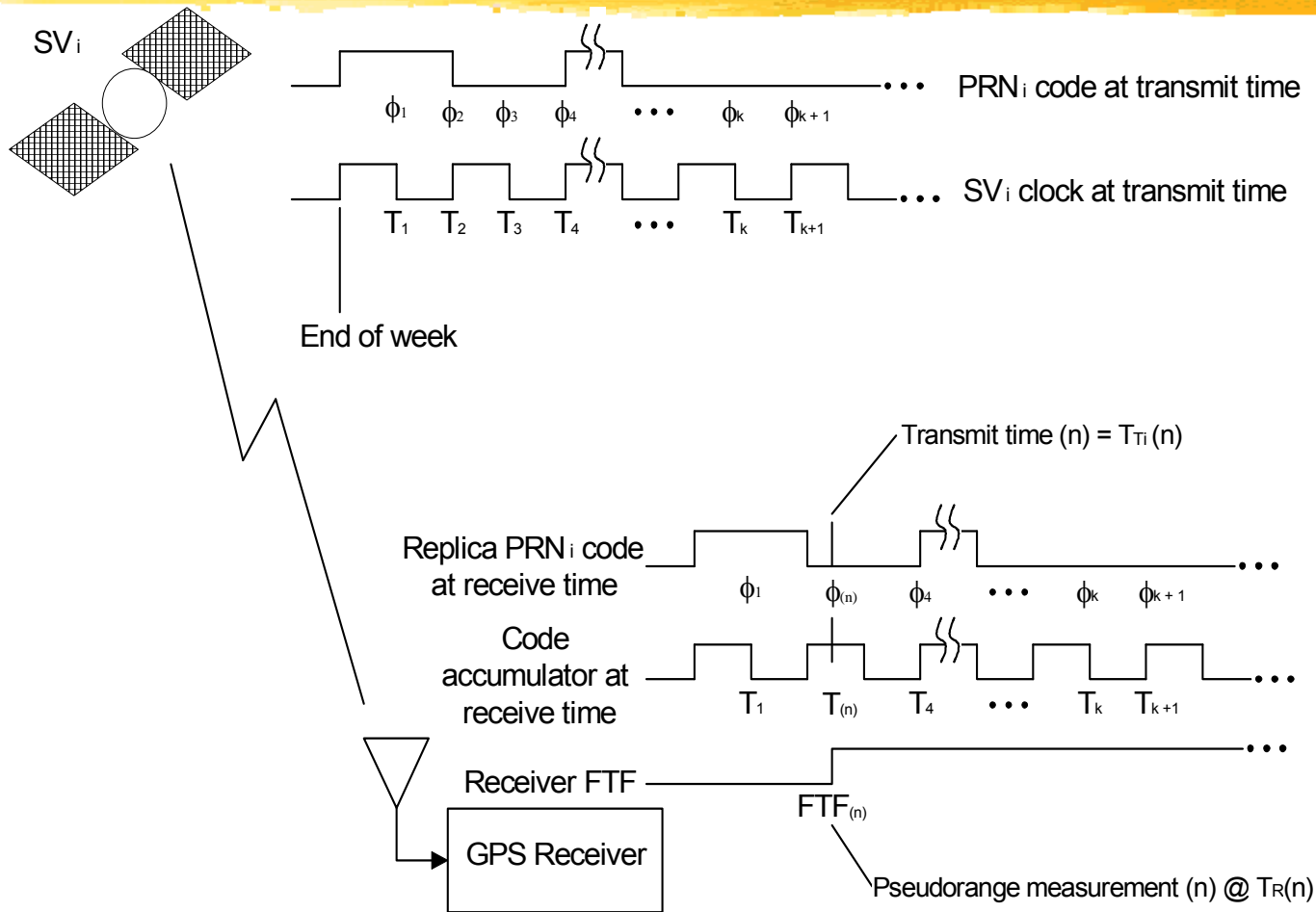
- $T_{Ti}(n) =$ transmit time based on SV_i clock (seconds) of measurement at epoch n

Satellite transmit time relationship to code phase

The figure below depicts the GPS satellite SV_i transmitting its pseudo random noise code PRN_i starting at the end of the GS week. Corresponding to each chip of the PRN_i code is a linear SV_i clock time. When this signal reaches the GPS receiver, the transmit time, $T_{Ti}(n)$, is the SV_i time corresponding to the PRN code state that is being replicated at receiver epoch n . The pseudorange derived from this measurement corresponds to a particular receive time epoch (epoch n) in the GPS receiver. Every epoch in the PRN code that is transmitted by SV_i is precisely aligned to the GPS time of week as maintained inside SV_i 's time-keeping hardware. When this transmitted PRN code reaches the user GPS receiver which is successfully correlating its replica code with it, the phase offset of the replica code with respect to the beginning of the GPS week represents the transmit time of SV_i . Therefore, it can measure the transmit time of SV_i by using its own linear time measurement of this replica code offset.

Typically, the GPS receiver will take a set of measurements at the same receiver time epoch. This is why the receive time, $T_R(n)$, is not identified with any particular SV PRN number in the pseudorange equation. When the GPS receiver schedules a set of measurements, it does this based on its own internal clock which contains a bias error with respect to true GPS time. Eventually the navigation process learns this bias error as a by-product of the GPS navigation solution. The SV_i transmit time also contains a bias error with respect to the true GPS time, although the Control Segment ensures that this is maintained at less than one millisecond of bias error. This corrections transmitted to the receiver by SV_i as clock correction parameters via the navigation message. However neither of these corrections is required to compute the pseudorange measurement in the pseudorange equation. These corrections (and several others) are determined and applied by the navigation process to determine the true measured range to SV_i .

Satellite transmit time relationship to code phase



Pseudorange measurement

From the equation given for the definition of pseudorange, it can be concluded that if the receiver baseband process can extract the SV transmit time from the code tracking loop, then it can provide a pseudorange measurement. The precise transmit time measurement for SV_i is equivalent to its code phase offset with respect to the beginning of the GPS week. There is a one-to-one relationship between the SV_i replica code phase and the GPS time as maintained by the SV_i clock (which becomes ultra precise when the clock correction terms a_0 , a_1 , and a_2 , in the navigation message are applied by the navigation process in the GPS receiver). Thus, for every fractional and integer chip advancement in the code phase of the PRN code generator since the initial (reset) state at the beginning of the week, there is a corresponding fractional and integer chip advancement in the GPS time. The fractional and integer chip code phase will hereafter be called the code state. The receiver baseband process time keeper, which contains the GPS time corresponding to this code state, will hereafter be called the code accumulator.

The replica code state corresponds to the receiver's best estimate of the SV transmit time. The receiver baseband process knows the code state because it sets the initial states during the search process and keeps track of the changes in the code state thereafter. The receiver baseband code tracking loop process keeps track of the GPS transmit time corresponding to the phase state of the code NCO and the replica PRN code generator state after each code NCO update. It does this by discrete integration of every code phase increment over the interval of time since the last NCO update and adds this number to the code accumulator. The combination of the replica code generator state (integer code state) and the code NCO state (fraction code state) is the replica code state. Since the code phase states of the PRN code generator are pseudo random, it would be impractical to read the code phase state of the PRN code generator and then attempt to convert this nonlinear code state into a linear GPS time state, say, by a table look-up. There are too many possible code states, especially for the P-code generator. A very practical way to maintain the GPS time in a GPS receiver is to use a separate code accumulator in the GPS receiver baseband process and to synchronize this accumulator to the replica PRN code generator phase state.

Pseudorange measurement



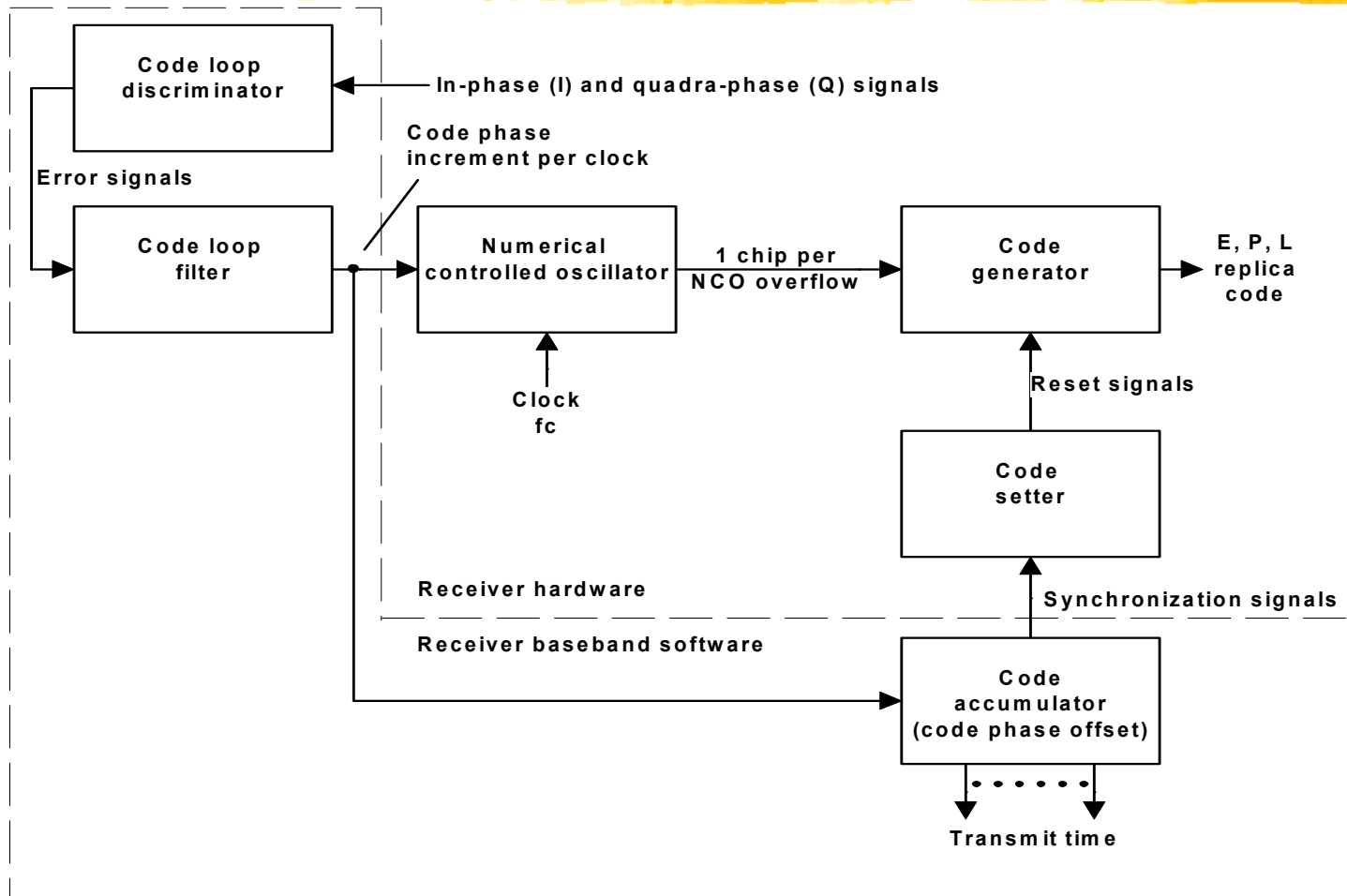
- Replica code state corresponds to receiver's best estimate of SV transmit time
 - Receiver knows replica code state because initial states are set during search process and it keeps track of code changes
 - Integer and fraction of chip replica code phase defined as code state
 - Receiver time keeper containing GPS time corresponding to replica code state defined as code accumulator

Relationship between PRN code generator and code accumulator

The figure below (derived from the code tracking loop shown earlier) illustrates the high level block diagram relationship between the replica PRN code generator and the code accumulator (which was not included in the code tracking loop block diagram) in the code tracking loop of one GPS receiver channel. A typical GPS navigation measurement incorporation rate is once per second. A typical GPS receiver fundamental time frame (FTF) for scheduling measurements is 20 milliseconds, which is the same as the 50 Hz navigation message data period. The receiver baseband process schedule for updating the code and carrier NCOs is usually some integer subset of the FTF, such as 20, 10, 5, 2 or 1 milliseconds. Assuming that the FTF is 20 milliseconds, the receiver measurement process maintains a monotone counter (call it the FTF counter) in 20-millisecond increments derived from the receiver's reference oscillator. The FTF counter is set to zero at power up, counts up, rolls over, counts up, etc. Assuming that the navigation measurement incorporation rate is one second, the navigation process will schedule measurements to be extracted from the code and carrier tracking loops every fifth FTF; i.e., based on the receiver's time epochs. When the receiver baseband process extracts the measurements from the code and carrier tracking loops, it time tags the measurements with the FTF count. The navigation process assigns and maintains a GPS receive time corresponding to the FTF count. The receive time initialization can be the first SV's transmit time plus a nominal propagation time of say, 76 milliseconds, if the navigation process does not know the GPS time accurately. This will set the initial receive time accurate to within about 20 milliseconds.

When a pseudorange measurement is scheduled on FTF(n), the receiver baseband code tracking loop process extracts the SV_i transmit time from its code accumulator and propagates this time forward to FTF(n). The result is the SV_i transmit time with a measurement resolution of 2^{-N} of a code chip, where N is the number of bits in the code NCO adder. If the code NCO adder uses a 32-bit register, this measurement resolution is less than a quarter of a nanochip, which makes the code measurement quantization noise negligible. The receiver baseband process can compute the pseudorange measurement from the SV_i transmit time using Equation 22 and time tag the measurement with FTF(n) before sending the result to the navigation process. However, the navigation process needs the (corrected) SV_i transmit time to compute the location of SV_i when it transmitted the measurement. Hence, the best measurement to send to the GPS navigation process is the (uncorrected) SV transmit time along with the FTF time tag. The navigation process applies the clock correction (including relativity correction), uses the corrected SV_i transmit time to compute the SV_i position, then computes the pseudorange plus other corrections before measurement incorporation.

Relationship between PRN code generator and code accumulator

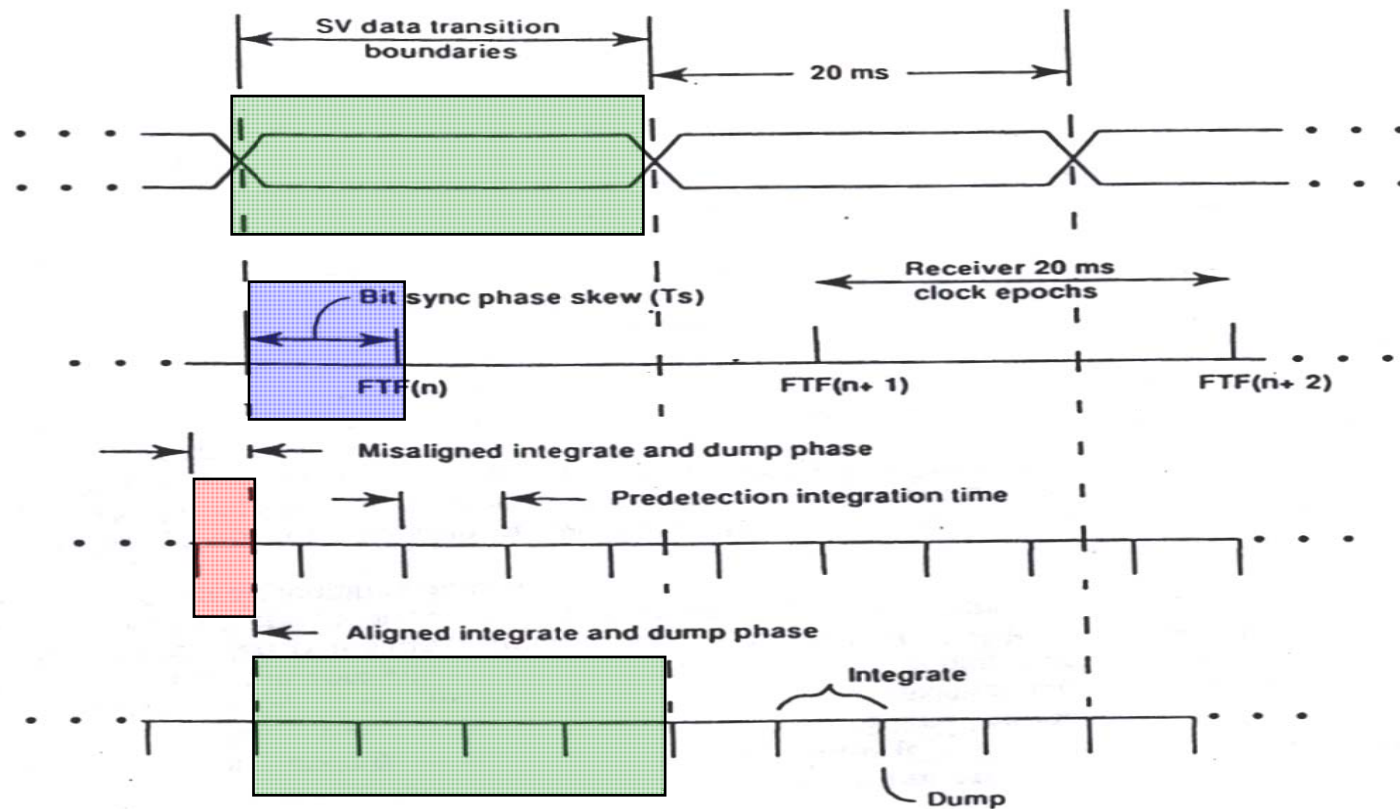


Measurement time skew

The figure below illustrates the bit sync phase skew, T_s , which exists between the SV data transition boundaries and the receiver 20-millisecond clock epochs, i.e., the FTFs. The Control Segment ensures that every SV transmits every epoch within 1 millisecond of true GPS time, i.e., the SV clocks are aligned to within 1 millisecond of true GPS time. Therefore, all of the SV data transition boundaries are approximately aligned to true GPS time at transmit time. However, at the GPS receiver the SV data transition boundaries are, in general, skewed with respect to each other and with respect to the receiver's FTF boundary. This is because the SVs are at different ranges with respect to the user GPS receiver. To achieve optimum performance, the user GPS receiver must adjust the phases of its integrate and dump boundaries in order to avoid integrating across the SV's data bit transition boundaries. The time skew, T_s , is different for each SV being tracked and it also changes with time because the range to the SVs change with time. Therefore, the epochs from each replica code generator, such as the C/A-code 1-millisecond epochs, are skewed with respect to each other and to the FTF. As a result, the integrate and dump times and the updates to the code and carrier NCOs are performed on a changing skewed time phase with respect to the FTF time phase, but the receiver baseband process learns and controls this time skew in discreet phase increments. The code accumulator is normally updated on the skewed time schedule that matches the code NCO update schedule. Therefore, if all of the GPS receiver measurements of a multiple channel GPS receiver are to be made on the same FTF, the contents of the code accumulator, when extracted for purposes of obtaining a measurement, must be propagated forward by the amount of the time skew between the code NCO update events and the FTF.

There are basically three ways to deal with this time skew problem: (1) Ignore it and keep the predetection integration time smaller than or equal to 5 ms (a potential degradation of 1 out of 4 integration samples); (2) Design the predetection integrate and dump hardware so that not only is the duration variable based on the desired integrate time, but also the dump phase is individually adjustable in each channel by the baseband process to match its respective SV data transition boundary within $\frac{1}{2}$ to 1 ms receiver clock accuracy. This will require a double buffer and an interrupt from each channel at the end of each dump; (3) Design the integrate and dump hardware for a fixed interval of $\frac{1}{2}$ to 1 ms. Provide buffering of all results compatible with the desired interrupt rate. Perform the de-skewing process and the remaining predetection integrate and dump process in the baseband processor.

Measurement time skew



Code accumulator

Although there are many code accumulator time-keeping conventions that would work, the following convention is convenient for setting the initial code generator and NCO phase states [8]. Three counters, Z, X1, and P, are maintained as the code accumulator. The Z counter (19 bits) accumulates in GPS time increments of 1.5 seconds, then is reset one count short of the maximum Z-count of 1 week = 403,200. Hence, the maximum Z count is 403,199. The X1 counter (24 bits) accumulates in GPS time increments of integer P chips, then is reset one count short of the maximum X1 count of 1.5 seconds = 15,345,000. Hence, the maximum X1 count is 15,344,999. The P counter accumulates in GPS time increments of fractions of a P chip, then rolls over one count short of one P chip. The P counter is the same length as the code NCO adder. A typical length is 32 bits.

Code (time) accumulator

■ Z-counter (19 bits)

- Accumulates in GPS time increments of 1.5 s, then resets one count short of 1-week = 403,200. Max = 403,199

■ X1-counter (24 bits)

- Accumulates in GPS time increments of integer P chips, then resets one count short of 1.5 s = 15,345,000. Max = 15,344,999

■ P-counter (same as NCO bits)

- Accumulates in GPS time increments of fractions of one P-chip

Maintaining the code accumulator

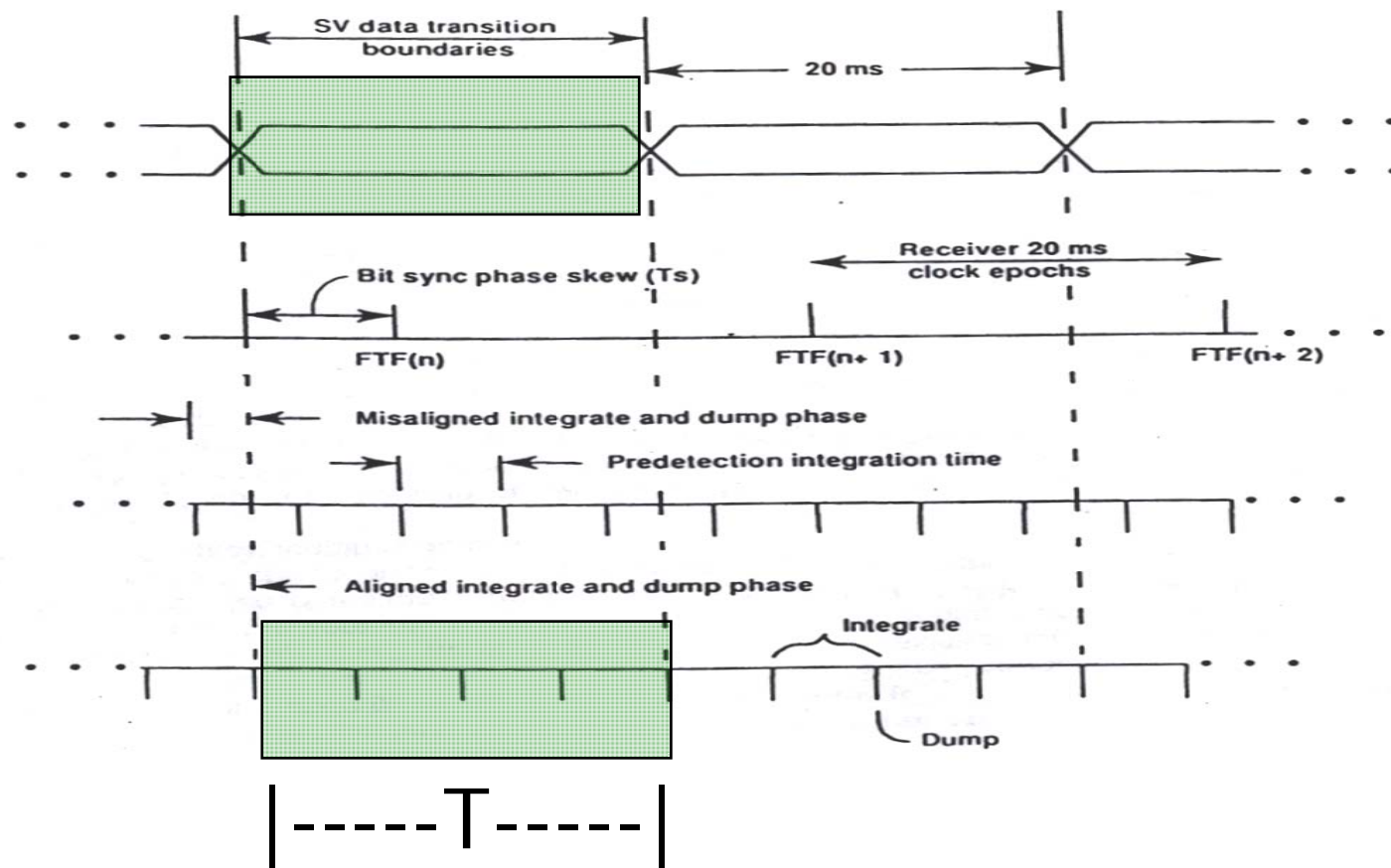
Note in the code tracking loop block diagram shown earlier that the code NCO synthesizes a code clock rate that corresponds to the replica code chipping rate including the effects of Doppler on the code. The P counter in the code accumulator tracks the fractional part of the code phase state. This is the same as the fraction contained in the code NCO (remember that the code accumulator overflows once for each advancement of one P code chip or for each 1/10 of a C/A-code chip).

Assuming that the code NCO and code accumulator are updated every T seconds, the algorithm for maintaining the code accumulator is shown below. The definition of $\Delta\phi_{co}$ contains two components, the code NCO bias and the code loop filter Doppler correction plus other small corrections). The code NCO bias (see code tracking loop block diagram) is the phase increment per clock that accounts for the "marching of time" in the P-code replica code generator. When applied to the P replica code generator, this is 10.23×10^6 chips/s. When applied to the C/A replica code generator, this is 1.023×10^6 chips/s. The code loop filter Doppler correction (and any other corrections required) is the combination of carrier aiding and code loop filter output. This combined output corrects the P replica code generator for Doppler (and a small order effect due to changes in ionospheric delay plus any bias error due to the reference oscillator) referenced to the P-code chipping rate. Usually the code generator provides the divide-by-ten function for the C/A-code generator if both P and C/A-codes are generated. This is the correct factor for the chipping rate and the code Doppler/ionospheric delay components. If only C/A code is to be synthesized, then the NCO bias can be set to 1/10th the rate and the value in the code accumulator is treated as a fraction of a C/A code chip. Also, the overflow criteria in the P algorithm must be adjusted by a factor of 10 larger.

Maintaining the code accumulator

- $P_{temp} = P + f_c \Delta\phi_{CO} T$
- P = fractional part of P_{temp} (chips)
- $X_{temp} = (X1 + \text{whole part of } P_{temp}) / 15,345,000$
- $X1$ = remainder of X_{temp} (chips)
- Z = remainder of $[(Z + \text{whole part of } X_{temp}) / 403,200]$
(1.5 seconds)
 - where: P_{temp} = temporary P register
 - f_c = code NCO clock frequency (Hz)
 - $\Delta\phi_{CO}$ = code NCO phase increment per clock cycle
 - = code NCO bias + loop filter Doppler correction, etc.
 - T = time between code NCO updates (seconds)

Maintaining the code accumulator with 20 ms PIT



Obtaining a measurement from the code accumulator

To obtain a measurement, the code accumulator must be propagated to the nearest FTF(n). This results in the set of measurements $P_i(n)$, $X1_i(n)$ and $Z_i(n)$ for SV_i . When converted to time units of seconds, the result is $T_{Ti}(n)$, the transmit time of SV_i at the receiver time epoch n . This is done very much like the algorithm for maintaining the code accumulator except the time T is replaced with the skew time, T_s . Note that the code accumulator is NOT updated when a transmit time measurement is taken.

$$\begin{aligned}
 P_{temp} &= P + f_c \cdot \Delta_{co} T_s \\
 P_i(n) &= \text{fractional part of } P_{temp} && \text{(chips)} \\
 X_{temp} &= (X1 + \text{whole part of } P_{temp}) / 15,345,000 \\
 X1_i(n) &= \text{remainder of } X_{temp} && \text{(chips)} \\
 Z_i(n) &= \text{remainder of } [(Z + \text{whole part of } X_{temp}) / 403,200] && \text{(1.5 sec)}
 \end{aligned}$$

Note that there is no error due to the measurement propagation process for the code accumulator measurements because the code NCO is running at a constant rate, Δ_{co} per clock, during the propagation interval. The following equation converts the code accumulator measurements to the SV_i transmit time, $T_{Ti}(n)$.

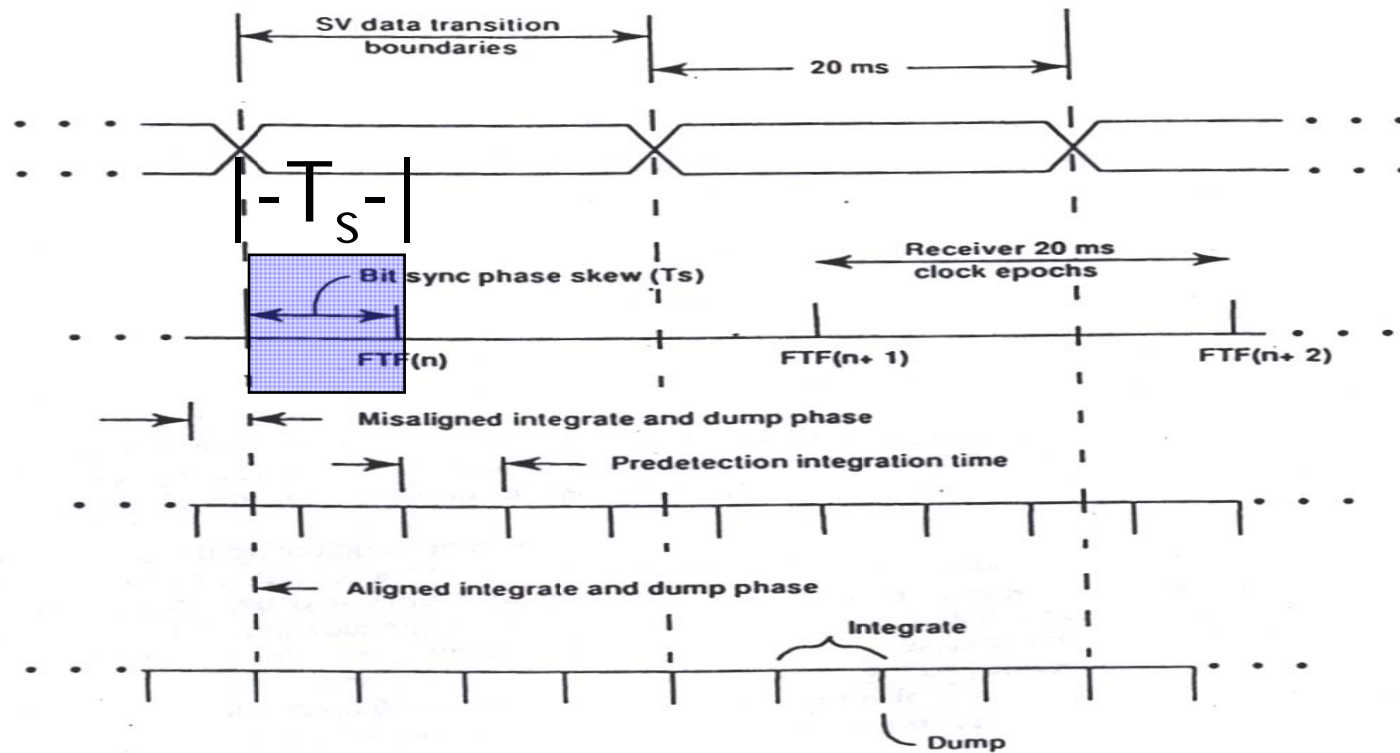
$$T_{Ti}(n) = [P_i(n) + X1_i(n)] / (10.23 \times 10^6) + Z_i(n) * 1.5 \text{ (sec)}$$

Since the number of fixed point bits is so large, the use of double precision floating point numbers is recommended after the measurement extraction.

Obtaining a measurement from the code accumulator

- $P_{temp} = P + f_c \Delta\phi_{CO} T_s$
- $P_i(n) =$ fractional part of P_{temp} (chips)
- $X_{temp} = (X1 + \text{whole part of } P_{temp}) / 15,345,000$
- $X1_i(n) =$ remainder of X_{temp} (chips)
- $Z_i(n) =$ remainder of $[(Z + \text{whole part of } X_{temp}) / 403,200]$ (1.5 seconds)
- $T_{Ti}(n) = [P_i(n) + X1_i(n)] / (10.23 \times 10^6) + Z_i(n) * 1.5$ (seconds)
- Note: code accumulator NOT changed

Obtaining a measurement from the code accumulator



Synchronizing the code accumulator to the C/A-code generator

Synchronizing the code accumulator to the C/A-code generator is the most complicated part of the pseudorange measurement process. This is because the count sequences taking place in the code generator shift registers are PN sequences, while the count sequence taking place in the code accumulator is a linear sequence. Fortunately, there are predictable reset timing events in the PN shift registers that permit them to be synchronized to the code accumulator. The first thought might be to design the code generator shift registers such that they contain the linear counters which are synchronized by the hardware and read by the receiver baseband process. However, the phase states of the code generators must be controlled by the receiver baseband process. So it is a better design to use code setters in the hardware and maintain the code accumulator in the receiver baseband processor. By far the most simple case is the C/A-code generator setup.

Synchronizing code accumulator to replica code generator

- Most complicated part of process:
Synchronizing code accumulator to C/A-code generator
 - Count sequences in code generator shift registers are pseudo random
 - Count sequences taking place in code accumulator are linear
- Reset timing events in PN shift registers are predictable

Adding a code setter to code generator

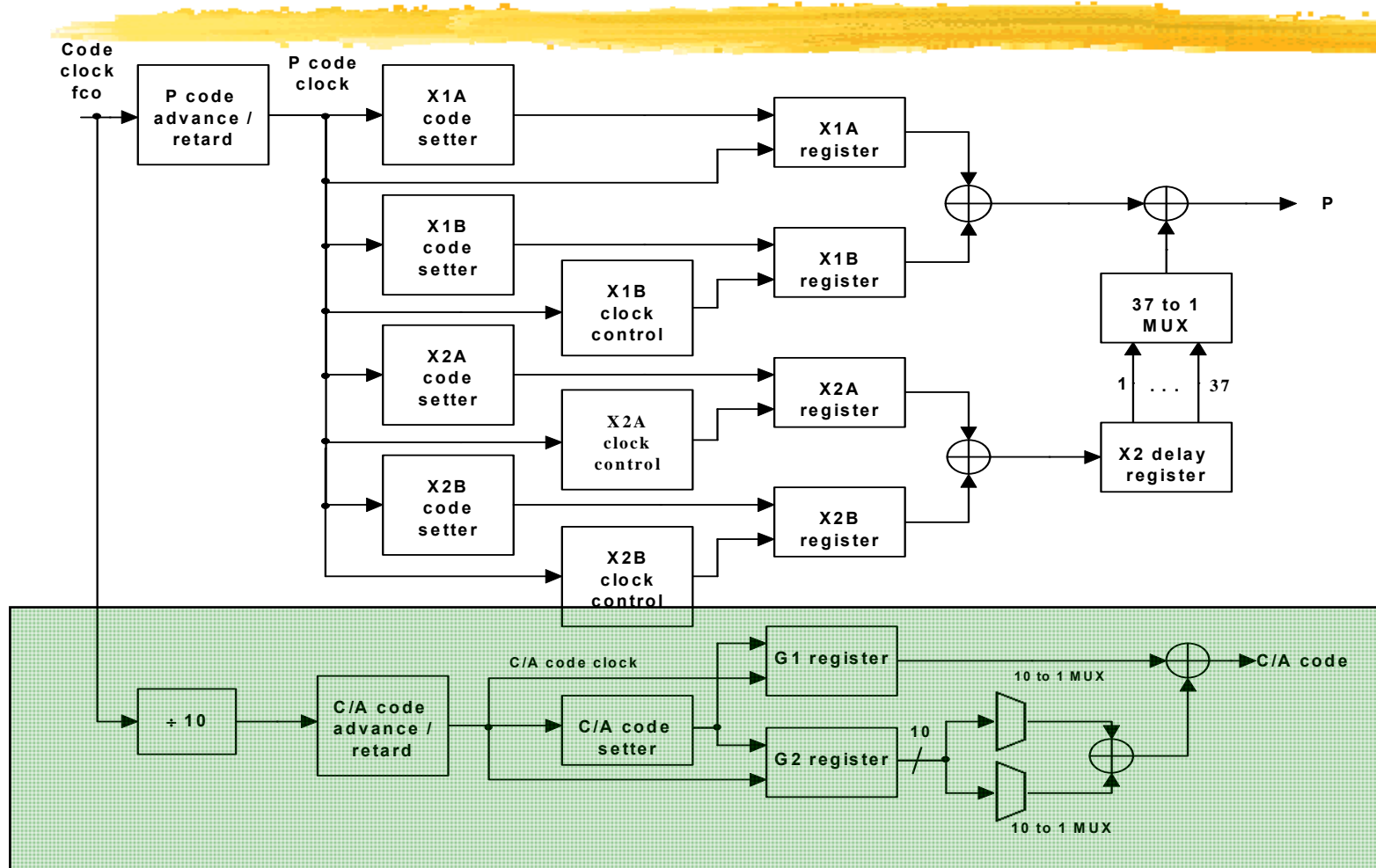
The figure below illustrates a high-level block diagram of the P and C/A-code generators including their code setters. The detailed block diagram of the C/A-code generator was illustrated earlier excluding the code setter. Recall that the C/A-code generator consists of two 10-bit linear feedback shift registers called the G1 and G2 register. The C/A-code setup for this scheme requires only one 10-bit code setter to initialize both the G1 and G2 registers in the C/A-code generator. The reason is that the linear time delay corresponding to the phase states of the G1 and G2 registers are the same for every SV for the same GPS time of week. It is the tap combination on the G2 register (or equivalently, the delay added to the G2 register) in combination with the G1 register which determines the PRN number. In this scheme, the C/A-code setter is capable of setting the G1 and G2 registers to their initial states and to their midpoint states. Since it requires only 1 ms for the C/A-code generator to complete one PRN period, this code setter design example holds the maximum delay to 500 microseconds or less to achieve the correct replica code phase state.

This code setter design has a delay count range of 0 to 511 (1/2 C/A-code epoch). When the delay value is entered and advanced until it rolls over, the code setter generates a reset signal to the G1 and G2 registers. Thereafter, the C/A-code generator is synchronized to the code accumulator.

This simple code setter principle is as follows: the replica C/A code phase state (delay) corresponding to a future time epoch in the receiver C/A-code generator is computed. This delay also corresponds to the next time the C/A-code generator resets. The delay is placed into the code setter at the precise clock epoch used for the computation. The code setter is advanced at the same rate as the C/A-code generator. When the code setter rolls over, the C/A-code generator is reset to its initial state. Thereafter, it is synchronized.

Another C/A-code setter technique that sets the state instantly is to provide a code setter register for both G1 and G2 that stores the total 10-bit state. In the baseband processor, store 1023 pre-computed 10-bit values for every state of the G1 and G2 registers. Use the delay computation to perform a table look-up of these two states and transfer the actual values into the two code setter registers. There must be a transfer signal generated at the future epoch corresponding to the computation of the C/A-code state.

Adding a code setter to code generator



C/A-code setup

Using the code setter hardware, the C/A-code setup process works as follows. In accordance with a future time delay equal to a fixed number of code NCO reference clock cycles later, the code accumulator value for that future time is loaded into the code setter. This value matches the desired C/A-code time after the scheduled time delay. The value for the code setter is computed just as though the 1023 state C/A-code generator had the same linear counting properties as a 1023-bit counter. The code setter begins counting after the scheduled time delay, starting with the loaded count value. The code setter sets the G1 and G2 registers when the count rolls over. If the value sent from the code accumulator was greater than 511, the C/A-code setter resets the G1 and G2 registers to their initial states. If the value sent from the code accumulator was less than or equal to 511, the code setter sets the G1 and G2 registers to their halfway points. As a result, the C/A replica code generator phase state matches the code accumulator GPS time state when the code setter rolls over and is synchronized to the code accumulator thereafter. When the receiver is tracking the SV after initialization, the code setter process can be repeated as often as desired without altering the C/A replica code generator phase state, because both the code accumulator and the code generator are ultimately synchronized by the same reference clock, the code NCO clock. If the receiver is in the search process, the C/A-code advance/retard feature shown in the previous figure provides the capability to add clock cycles or swallow clock cycles in 1/2-chip increments. The code accumulator keeps track of these changes. If the receiver can predict the satellite transmit time to within a few chips during the search process, it can use the code setter to perform a direct C/A-code search. This condition is satisfied if the receiver has previously acquired four or more satellites and its navigation solution has converged. Ordinarily, all 1023 C/A-code chips are searched. The algorithm for the code accumulator output to the C/A-code setter is as follows.

$$G = \text{remainder of } [(\text{whole part of } \{X1/10\}) / 1023]$$

where: G = future scheduled C/A-code time value sent to the code setter
 $X1$ = future scheduled GPS time of week in P chips ($0 \leq X1 \leq 15,344,999$)

Commercial C/A-code receivers without code NCOs propagate the code generator at the nominal chipping rate between code loop updates, tolerating the quantization error and code Doppler error between updates. Instead of the code NCO, a counter with a fractional chip advance/retard capability is used to adjust the phase of the C/A replica code generator in coarse phase increments. This results in a very large quantization noise and noisy pseudorange measurements.

C/A-code setup



- Algorithm for code accumulator output to C/A-code setter:
 - $G = \text{remainder of } [(\text{whole part of } \{X1/10\}) / 1023]$
where:
 - $G = \text{future scheduled C/A-code time value sent to the code setter}$
 - $X1 = \text{future scheduled GPS time of week in P chips}$
($0 \leq X1 \leq 15,344,999$)

Obtaining transmit time from the C/A-code

A C/A-code receiver obtains transmit time from its code accumulator in the same manner as a P(Y)-code receiver. The difference is the manner in which the C/A-code accumulator is initialized. The replica C/A-code generator produces an epoch every ms; i.e., at a rate of 1 KHz. In order to obtain alignment with the SV 50 Hz data bit transition edges, these 1 KHz replica C/A-code epochs are divided by 20. However, the phase of the output of this divide-by-20 circuit must be set to match the SV data bit transition. If the receiver is tracking the SV in C/A-code, then by definition, the C/A-epoch will match the 50 Hz data bit transitions. The only problem is, which of the 20 possible C/A-code epochs in one data bit interval is the one which corresponds to the data bit transition phase? Each receiver channel determines this phase by a process called bit synchronization. Once bit synchronization is successfully accomplished, then the receiver can read the Handover Word (HOW) at the beginning of each subframe in the SV navigation message and use it to set its code accumulator to the correct Z-count at exactly the right C/A-code epoch.

The replica 20 ms epoch that is synthesized by the replica C/A-code generator epochs and the bit-synchronized divide-by-20 circuit is skewed in phase with respect to the receiver 50 Hz clock, called the FTF (fundamental time frame). After the code accumulator has been set using the HOW, the transmit time is obtained using the algorithms defined earlier. If the receiver is a C/A-code only receiver, the P-counter in the code accumulator can be maintained in fractions of a C/A-chip instead of fractions of a P-chip (which is equal to 1/10 C/A-chip). It should be obvious how the algorithms for maintaining the code accumulator and extracting code measurements would have to be modified to reflect that the C/A-code chip is one-tenth of the rate of the P-code chip.

Obtaining transmit time from the C/A-code



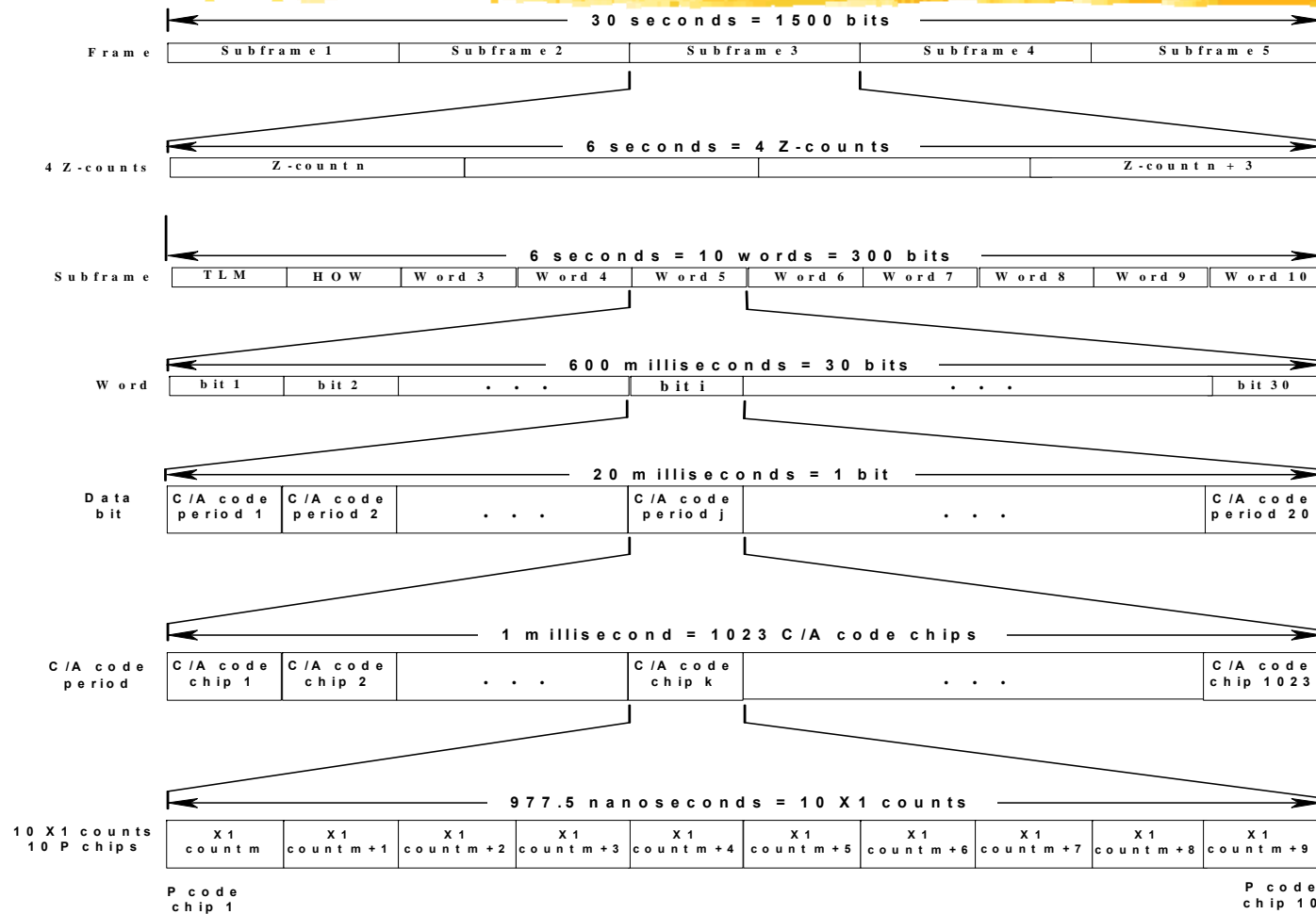
- C/A-code obtains transmit time from code accumulator in same manner as P(Y)-code
 - Difference is initialization of accumulator
 - Receiver reads Handover Word (HOW) after bit synchronization
 - Sets its code accumulator to correct Z-count at exactly right C/A-code epoch

GPS C/A-code timing relationships

The figure below illustrates the GPS timing relationships which enable a C/A-code receiver to determine the true GPS transmit time. The C/A-code repeats every 1 millisecond and is therefore ambiguous every 1 millisecond of GPS time. There is a handover word (HOW) at the beginning of every one of the five subframes of the satellite navigation message. The HOW contains the Z-count of the first data bit transition boundary at the beginning of the next subframe. This is the first data bit of the Telemetry Message (TLM) that precedes every HOW. The beginning of this 20-millisecond data bit is synchronized with the beginning of one of the satellite's C/A-code 1-millisecond periods, but there are 20 C/A-code periods in every data bit period. At this subframe epoch the X1 register has just produced a carry to the Z-count, so the X1 count is zero. The C/A-code ambiguity is resolved by setting the Z-count to the HOW value and the X1 count to zero at the beginning of the next subframe.

The Z-count and X1 count will be correct if the GPS receiver has determined its bit synchronization to within 1 millisecond or better accuracy. This level of accuracy will perfectly align the 1-millisecond C/A-code epoch with the 20-millisecond data bit transition point of the first bit in the following TLM word. Therefore, the C/A-code transmit time will be unambiguous and correct. If the bit synchronization process makes an error in the alignment of the 1 millisecond replica C/A-code epoch with the 20-millisecond data bit epoch, then the X1 count will be off by some integer multiple of 1 millisecond. If the receiver attempts a handover to P-code and fails, the typical strategy is to try the handover again with ± 1 millisecond changes in the value used for X1, then ± 2 milliseconds trials, etc., before attempting to redetermine bit synchronization. (This process can take six seconds or longer and prevents processing of GPS measurements until complete).

GPS C/A-code timing relationships



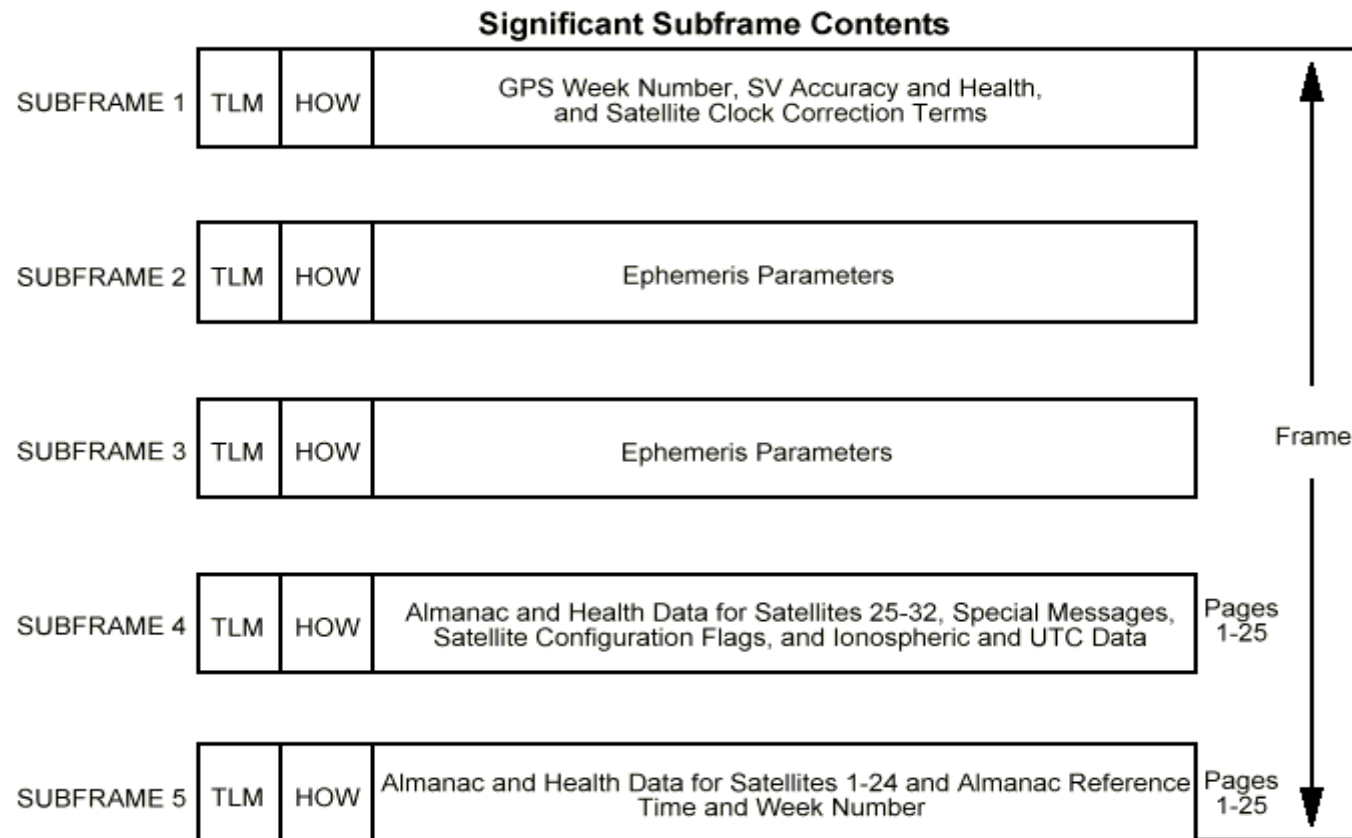
GPS navigation message data format

The GPS navigation message data format is illustrated below. It consists of five subframes each containing 300 bits. At 50 bits per second, each subframe requires 6 seconds to read. The first three subframes contain the ephemeris and clock correction information necessary to precisely locate the satellite and correct the satellite's clock so that it is possible to navigate after successfully reading these subframes. Since it is possible that this reading process could begin at subframe four, then it could take up to 30 seconds to incorporate a satellite's measurement precisely even though the receiver has acquired the signal and is obtaining measurements from the satellite. It can use the less precise almanac data which has been stored in nonvolatile memory from a previous operation, but the norm is to declare "first fix" using the broadcast ephemeris and clock correction data.

The last two subframes are permuted 25 times to multiplex in a very large amount of data including the almanac and health of every satellite on orbit plus an ionospheric correction model for single frequency users. It requires 12.5 minutes of continuous reading to demodulate all 25 of these subframe four and five permutations.

The C/A-code receiver has an additional need to read the handover word (HOW) located at the beginning of each subframe. Prior to reading the HOW, it must obtain bit and frame synchronization with the navigation message. This process can take six seconds or longer depending on the signal quality. The HOW contains the Z-count (truncated to 6 seconds LSB) referenced to the beginning of the first data bit in the following six second subframe. Since the C/A-code 1 ms epochs are synchronous with the 20 ms data bit transitions and since the satellite clock X1 count overflows every 1.5 seconds into the Z-count, then all values of the C/A-code time phase, the X1 time phase and the first two bits of the Z-count are zero at these epochs. The receiver does not have to wait six seconds after reading the HOW to set the transmit time of the satellite in his receiver code accumulator. It simply takes the HOW and backs it up to the very near future epoch at which the code accumulator is to be initialized and thereafter maintained. Until the bit synchronization, frame synchronization and HOW reading process is completed, it is not possible to incorporate C/A-code measurements unambiguously. Since the period of the P(Y)-code is one week, there is no problem with ambiguity. However, there is a serious problem finding it during search unless the user time, position and velocity are known reasonably well or, more likely, the C/A-code successfully handed over into P(Y)-code. An important point is that all C/A-code receivers will fail the bit-sync process at some degraded level of C/No. So if there is signal obscuration or attenuation enough to cause this failure, they will process incorrect pseudorange measurements.

GPS navigation message format



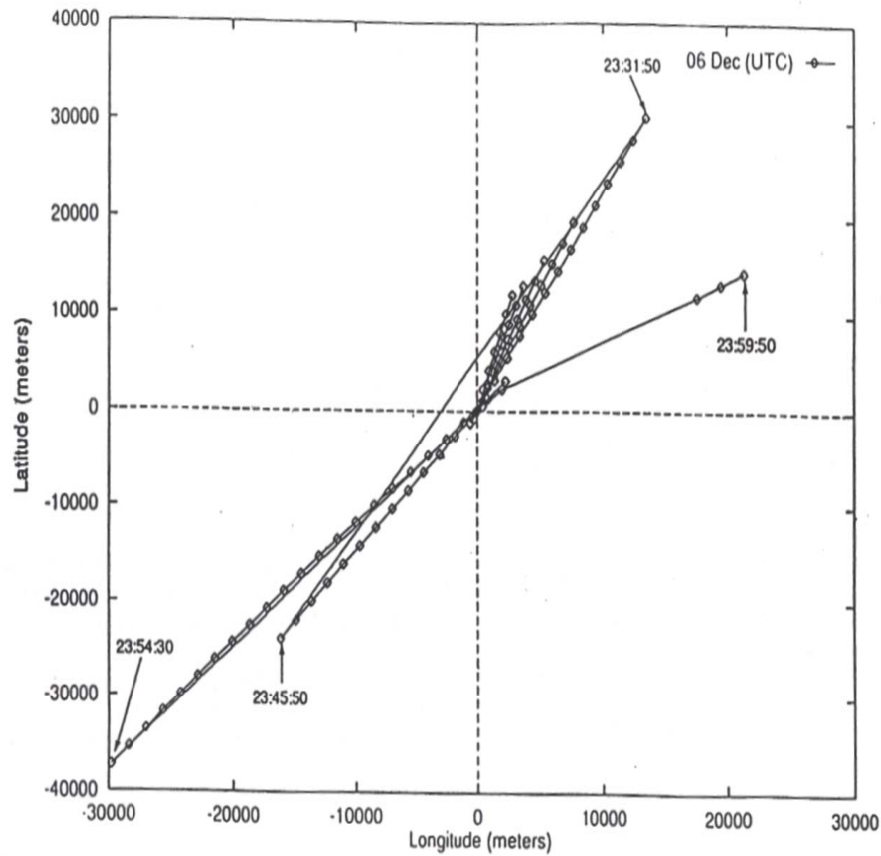
Each subframe is 300 bits (6 s @ 50 bps). Entire message repeats every 12.5 min (5 subframes × 300 bits/subframe × 25 pages = 37500 bits/message)

Example of bit sync error in C/A-code measurement

A successful handover to P-code verifies the bit synchronization process. However, if the GPS receiver is a C/A-code receiver, the verification for correct bit synchronization is more difficult. This verification task must be performed by the navigation process. Since 1 millisecond of GPS time error is equivalent to about 300,000 meters of pseudorange error, the navigation error can be quite serious. In the unlikely case that every channel makes the identical bit sync error, the navigation position error washes out of the position solution into the time bias solution and the GPS time is in error by 1 millisecond. The typical bit sync error manifestations in the navigation solution are unrealistic local level velocity and elevation computations. The latitude and longitude computations are also unrealistic, but there is usually no boundary condition for comparison. However, the velocity and elevation computations can be compared to acceptable boundary conditions.

The figure below illustrates what happens when a C/A-code receiver makes a bit synchronization error. The position error is extremely large. Usually the altitude error is the largest. As a result of the large position error, the local level velocity error is always very large. In fact, performing reasonableness checks on altitude and velocity are the two most effective methods for a C/A-code receiver to verify that the bit synchronization process has been successful. The bit synchronization process is a statistical process that is dependent on C/N_0 . It will occasionally be incorrect. It will be incorrect almost every time the C/N_0 drops below the bit synchronization design threshold. This causes serious navigation integrity problems for C/A-code receivers under conditions of signal attenuation or RF interference. This problem is compounded if there is no design provision to adapt the bit synchronization process for poor C/N_0 conditions and/or for the navigation process to check for bit synchronization errors.

Example of bit sync error in C/A-code



Session VI - Carrier



Session VI - Carrier

Extracting measurements from code and carrier loops



- Extracting measurements from carrier loop
 - Maintaining the carrier accumulator
 - Obtaining a measurement from the carrier accumulator
 - Delta pseudorange measurement
 - Data demodulation
 - Bit error performance

Maintaining the carrier accumulator

Similar to the code accumulator for tracking the uncorrected transmit time, a carrier accumulator is used to track the uncorrected carrier Doppler phase measurement, sometimes called the integrated carrier Doppler phase. The carrier accumulator is updated after each carrier loop output to the carrier NCO using the following algorithm.

$$\begin{aligned}\Phi_{\text{temp}} &= \Phi_{\text{CA}} + f_c \Delta\phi_{\text{CA}} T \\ \Phi_{\text{CA}} &= \text{fractional part of } \Phi_{\text{temp}} \text{ (cycles)} \\ N_{\text{CA}} &= N_{\text{CA}} + \text{whole part of } \Phi_{\text{temp}} \text{ (cycles)}\end{aligned}$$

where:

- Φ_{temp} = temporary Φ_{CA} register
- f_c = carrier NCO clock frequency (Hz)
- $\Delta\phi_{\text{CA}}$ = carrier NCO carrier Doppler phase increment per clock cycle
= carrier loop filter velocity correction + carrier loop velocity aiding (if any)
- T = time between carrier NCO updates (seconds)
- N_{CA} = integer number of carrier Doppler phase cycles since some starting point.

The fractional part of the carrier accumulator, Φ_{CA} , is initialized to the same state as the carrier NCO at the beginning of the search process, which is typically zero. The integer number of carrier Doppler phase cycles, N_{CA} , is ambiguous. Since only differential measurements are taken from this register, the ambiguity does not matter. The carrier integer accumulator is usually set to zero when the carrier loop is first closed following a successful search operation. Note that the "marching of time" carrier NCO bias is not included in the carrier accumulator because it is simply a bias term. Since only differential measurements are extracted from the carrier accumulator, this bias term, if included, would simply cancel out because it is a constant. The counter rolls over when the Doppler cycle count exceeds the count capacity or underflows if the Doppler count is in the reverse direction and drops below the zero count. The differential measurement comes out correct if the counter capacity is large enough to ensure that this happens no more than once between any set of differential measurements extracted from the carrier accumulator.

Maintaining the carrier accumulator

■ Carrier accumulator updated as follows:

- $\Phi_{\text{temp}} = \Phi_{\text{CA}} + f_c \Delta\phi_{\text{CA}} T$
- $\Phi_{\text{CA}} =$ fractional part of Φ_{temp} (cycles)
- $N_{\text{CA}} = N_{\text{CA}} +$ whole part of Φ_{temp} (cycles)

where:

- $\Phi_{\text{temp}} =$ temporary Φ_{CA} register
- $f_c =$ carrier NCO clock frequency (Hz)
- $\Delta\phi_{\text{CA}} =$ carrier NCO carrier Doppler phase increment per clock epoch = carrier loop filter velocity correction + carrier loop velocity aiding (if any)
- $T =$ time between carrier NCO updates (seconds)
- $N_{\text{CA}} =$ integer number of carrier Doppler phase cycles since some arbitrary starting point

Obtaining a measurement from the carrier accumulator

To extract a carrier Doppler phase measurement, $N_{CAi}(n)$, $\Phi_{CAi}(n)$, for SV_i corresponding to the carrier accumulator, it must be propagated forward to the nearest FTF(n) by the skew time, T_s , similar to the technique used in the code tracking loop.

$$\begin{aligned}\Phi_{temp} &= \Phi_{CA} + f_c \Delta\phi_{CA} T_s \\ \Phi_{CAi}(n) &= \text{fractional part of } \Phi_{temp} \text{ (cycles)} \\ N_{CAi}(n) &= N_{CA} + \text{whole part of } \Phi_{temp} \text{ (cycles)}\end{aligned}$$

Note that there is no error due to the measurement propagation process for the carrier Doppler phase measurement because the carrier NCO is running at a constant rate, $\Delta\phi_{CA}$ per clock, during the propagation interval.

Obtaining a measurement from the carrier accumulator

- The natural measurement obtained from the carrier accumulator associated with SV_i is an ambiguous integer number of cycles, N_{CAi} , and an unambiguous fraction of a cycle (phase), Φ_{CAi} , defined at some epoch time
- Called an integrated carrier Doppler phase measurement
- To obtain integrated carrier Doppler phase measurement, $N_{CAi}(n)$, $\Phi_{CAi}(n)$, for SV_i corresponding to carrier accumulator, propagate forward to nearest FTF(n) by skew time, T_s :
 - $\Phi_{temp} = \Phi_{CA} + f_c \Delta\phi_{CA} T_s$
 - $\Phi_{CAi}(n) = \text{fractional part of } \Phi_{temp} \quad (\text{cycles})$
 - $N_{CAi}(n) = N_{CA} + \text{whole part of } \Phi_{temp} \quad (\text{cycles})$

Delta pseudorange measurement

The carrier measurements are probably the most misunderstood and mid-modeled measurements in the navigation portion of a GNSS receiver. Contrary to popular opinion, the carrier phase measurement is an ambiguous range measurement, not a velocity measurement. The best use of the delta pseudorange measurement is to take integrated carrier Doppler phase measurements every time a pseudorange measurement is taken, then form the pseudo change in range (delta pseudorange) between every two measurements, so that there is no loss of carrier Doppler phase data between measurements. The navigation process can determine a very accurate change in position from the corrected measurements and an excellent average velocity between pseudorange measurements.

Some receivers take delta pseudorange measurements in a very short time interval, then divide them by that time interval to approximate a velocity measurement. The shorter the time interval the more noise is in the velocity measurement. This, in turn, is incorrectly modeled in the navigation filter, making the velocity measurement accuracy much worse. Much better to utilize all of the integrated carrier Doppler phase measurements between pseudorange measurements and to correctly model these extremely precise measurements as a pseudo change in range (about three orders of magnitude lower noise than in the pseudorange measurements).

Delta pseudorange measurement

- Define delta pseudorange measurement, $DPR_i(n)$, two integrated carrier Doppler measurements, $ICD_i(n-K)$ and $ICD_i(n)$, taken at FTF $(n-K)$ and FTF (n)
 - FTF $(n-K)$ occurs some integer number of FTFs earlier than FTF (n) , ideally at PR measurements
 - $ICD_i(n-K) = N_{CAi}(n-K), \Phi_{CAi}(n-K)$
 - $ICD_i(n) = N_{CAi}(n), \Phi_{CAi}(n)$
 - $DPR_i(n) = [ICD(n) - ICD(n-K)]\lambda$ (meters)
 - $\lambda = 0.1903$ m/cycle at L1 (and 0.2442 m/cycle at L2)

Data demodulation

Data bits are estimated from 20 ms correlator outputs as:

$$\hat{\mathbf{b}}_k = \mathbf{sign}(I_{P_k})$$

This is ordinary BPSK data demodulation. The well-known expression for BPSK bit error rate is:

$$P_b = \frac{1}{2} \operatorname{erfc}\left(\frac{E_b}{N_0}\right)$$

The ratio of bit energy E_b to noise power density N_0 may be related to P/N_0 using:

$$E_b = P/R_b$$

↑
data rate (50 Hz for GPS)

Bit error performance

