Dual DPLL Digitized Clock Synchronizer

Data Sheet **AD9546**

FEATURES

- Digitized clock transport subsystem
- 9 independent UTS blocks (time stamp egress ports)
- 2 independent IUTS blocks (time stamp ingress ports)
- Dual DPLL synchronizes 1 Hz to 750 MHz physical layer clocks, providing frequency translation with jitter cleaning of noisy references
- Complies with ITU-T G.8262 and Telcordia GR-253
- Supports Telcordia GR-1244, ITU-T G.812, ITU-T G.813, ITU-T G.823, ITU-T G.824, ITU-T G.825, and ITU-T G.8273.2
- Continuous frequency monitoring and reference validation for frequency deviation as low as 50 ppb (5×10^{-8})
- Both DPLLs feature a 24-bit fractional divider with 24-bit programmable modulus
- Programmable digital loop filter bandwidth: 0.0001 Hz to 1850 Hz
- 2 independent, programmable auxiliary NCOs (1 Hz to 65,535 Hz, resolution < 1.37 pHz), suitable for IEEE 1588 Version 2 servo feedback in PTP applications
- Automatic and manual holdover and reference switchover, providing zero delay, hitless, or phase buildout operation
- Programmable priority-based reference switching with manual, automatic revertive, and automatic nonrevertive modes supported
- 5 pairs of clock output pins with each pair useable as differential LVDS/HCSL/CML or as 2 single-ended outputs (1 Hz to 500 MHz)
- 2 differential or 8 single-ended input references
- Crosspoint mux interconnects reference inputs to PLLs
- Supports embedded (modulated) input/output clock signals
- Fast DPLL locking modes
- Provides internal capability to combine the low phase noise of a crystal resonator or crystal oscillator with the frequency stability and accuracy of a TCXO or OCXO
- External EEPROM support for autonomous initialization
- Single 1.8 V power supply operation with internal regulation
- Built in temperature monitor and alarm and temperature compensation for enhanced zero delay performance

APPLICATIONS

- 5G timing transport high precision synchronization
- Global positioning system (GPS), precision time protocol (PTP) (IEEE 1588), and synchronous Ethernet (SyncE) jitter cleanup and synchronization
- Optical transport networks (OTN), synchronous digital hierarchy (SDH), and macro and small cell base stations
- Small base station clocking (baseband and radio)
- Stratum 2, Stratum 3e, and Stratum 3 holdover, jitter cleanup, and phase transient control
- JESD204B support for analog-to-digital converter (ADC) and digital-to-analog converter (DAC) clocking
- Carrier Ethernet

GENERAL DESCRIPTION

The AD9546 incorporates digitized clocking technology that efficiently transports and distributes clock signals in systems. Digitized clocking allows the design of flexible and scalable clock transport systems with well controlled phase (time) alignment. These characteristics make the AD9546 a leading choice for the design of network equipment that must meet the synchronization requirements for IEEE® 1588™ boundary clocks per ITU-T G.8273.2 Class D. Digitized clocking is also relevant in applications requiring the accurate transport of frequency and phase to multiple usage endpoints (for example, distributing synchronized system reference (SYSREF) clocks to an array of ADC channels).

The AD9546 supports existing and emerging International Telecommunications Union (ITU) standards for the delivery of frequency, phase, and time of day over service provider packet networks (ITU-T G.8262, ITU-T G.812, ITU-T G.813, ITU-T G.823, ITU-T G.824, ITU-T G.825, and ITU-T G.8273.2).

The 10 clock outputs of the AD9546 synchronize to any one of up to eight input references. The digital phase-locked loops (DPLLs) reduce timing jitter associated with the external references, and the analog phase-locked loops (APLLs) provide frequency translation with low jitter output clocks. The digitally controlled loop and holdover circuitry continuously generate a low jitter output signal, even when all reference inputs fail.

The AD9546 is available in a 48-lead LFCSP (7 mm × 7 mm) package and operates over the −40°C to +85°C temperature range.

Throughout this data sheet, a single function of a multifunction pin name may be referenced when only that function is relevant (for example, M5 for SDO/M5).

Rev. 0 Document Feedback

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 Technical Support

www.analog.com

TABLE OF CONTENTS

REVISION HISTORY

10/2020—Revision 0: Initial Version

FUNCTIONAL BLOCK DIAGRAM

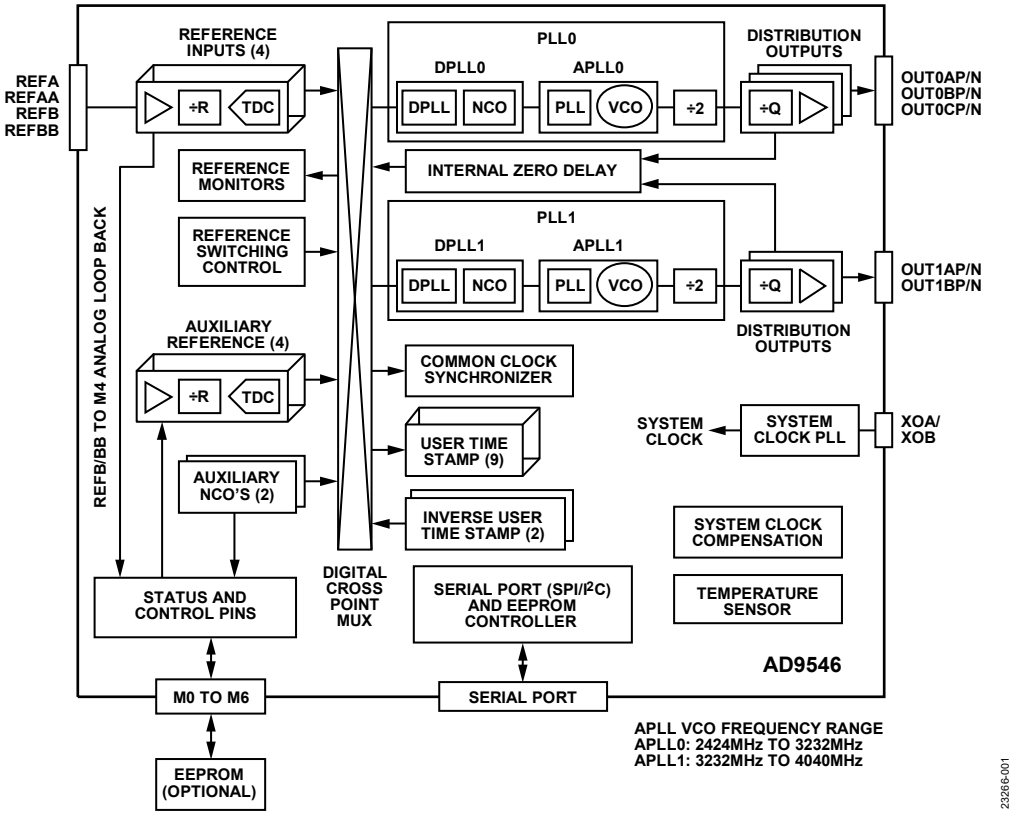


Figure 1.

SPECIFICATIONS

The minimum and maximum values apply for the full range of supply voltage and operating temperature variations. Typical values apply for VDD = 1.8 V and T_A= 25°C, unless otherwise noted.

OPERATING TEMPERATURE

Table 1.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
OPERATING TEMPERATURE					
Ambient Temperature[1]	−40		+85	°C	
Die Temperature[2]			+105	°C	Internal temperature sensor readings exceeding 105°C indicate excessive die temperature

[1] See the Thermal Resistance section for additional information.
[2] The maximum die temperature supports the performance stated in Table 2 to Table 39. The maximum operating die temperature takes precedence over the maximum ambient operating temperature. Use the internal temperature sensor to measure the AD9546 die temperature (see the Temperature Sensor section).

SUPPLY VOLTAGE

Table 2.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
SUPPLY VOLTAGE					
VDDIOA and VDDIOB	1.71	1.8	3.465	V	1.8 V, 2.5 V, and 3.3 V operation supported
VDD	1.71	1.8	1.89	V	

SUPPLY CURRENT

The minimum, typical, and maximum supply current values given in Table 3 are for the minimum, typical, and maximum supply voltage specifications in Table 2.

Table 3.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
SUPPLY CURRENT FOR TYPICAL CONFIGURATION					Uses the Typical Configuration conditions in Table 4
VDDIOx Current (I_{VDDIOx})		5	8	mA	Aggregate current for all VDDIOx pins (where x = A or B)
VDD Current (I_{VDD})		325	330	mA	Aggregate current for all VDD pins
SUPPLY CURRENT FOR LINE CARD CONFIGURATION					Uses the Line Card Configuration conditions in Table 4; see the Digitized Clocking Application section for line card information
I_{VDDIOx}		5	8	mA	Aggregate current for all VDDIOx pins (where x = A or B)
I_{VDD}		325	330	mA	Aggregate current for all VDD pins
SUPPLY CURRENT FOR TIMING CARD CONFIGURATION					Uses the Timing Card Configuration conditions in Table 4; see the Digitized Clocking Application section for timing card information
I_{VDDIOx}		5	8	mA	Aggregate current for all VDDIOx pins (where x = A or B)
I_{VDD}		325	330	mA	Aggregate current for all VDD pins
SUPPLY CURRENT FOR ALL BLOCKS RUNNING CONFIGURATION					Uses the All Blocks Running Configuration conditions in Table 4
I_{VDDIOx}		5	8	mA	Aggregate current for all VDDIOx pins (where x = A or B)
I_{VDD}		410	440	mA	Aggregate current for all VDD pins

POWER DISSIPATION

The typical values apply for VDD = 1.8 V, and the maximum values apply for VDD = 1.89 V.

Table 4.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
POWER DISSIPATION					f_{OSC} is the frequency at the XOA/XOB pins
Typical Configuration		585	760	mW	f_{OSC} = 49.152 MHz crystal; two active DPLLs; two 19.44 MHz input references in differential mode; two ac-coupled PLL Channel 0 (PLL0) current mode logic (CML) output drivers at 245.76 MHz; and two PLL Channel 1 (PLL1) CML output drivers at 156.25 MHz
Line Card Configuration		585	760	mW	f_{OSC} = 52 MHz crystal; two active DPLLs (hitless mode); five single-ended input references with two operating at 10 MHz and three with active 1 Hz reference demodulation operating at 50 MHz; five HCSL (15 mA) output drivers operating at 10 MHz; one auxiliary NCO operating at 500 Hz; two inverse user timer stampers (IUTSs) operating at 500 Hz; auxiliary DPLL enabled; see the Digitized Clocking Application section for line card information
Timing Card Configuration		585	760	mW	f_{OSC} = 52 MHz crystal; two active DPLLs (one hitless mode, one phase build-out mode); eight single-ended input references with four operating at 10 MHz and four with active 1 Hz reference demodulation operating at 50 MHz; five HCSL (15 mA) output drivers with two operating at 10 MHz and three with active 1 Hz embedded clock modulation operating at 50 MHz; two auxiliary NCOs with one operating at 500 Hz and one operating at 1 Hz; one IUTS operating at 500 Hz; auxiliary DPLL enabled; see the Digitized Clocking Application section for timing card information
All Blocks Running Configuration		745	795	mW	f_{OSC} = 49.152 MHz crystal; two active DPLLs; two 19.44 MHz input references in differential mode; three ac-coupled PLL0 HCSL output drivers at 400 MHz; and two PLL1 HCSL output drivers at 400 MHz
Full Power-Down		125		mW	Based on the Typical Configuration specification, but with Register 0x2000, Bit 0 = 1
Incremental Power Dissipation					Based on the Typical Configuration specification; the values in this section indicate the change in power due to the indicated operation relative to the Typical Configuration specification
Complete DPLL/APLL On vs. Off		160		mW	Change in dissipated power relative to the Typical Configuration specification; the powered down blocks consist of one reference input, one DPLL, one APLL, two channel dividers, and two output drivers
Input Reference On vs. Off					
Differential (AC-Coupled Mode)		20		mW	Reference frequency (f_{REF}) = 19.44 MHz (see Figure 36)
Differential (DC-Coupled Mode)		21		mW	f_{REF} = 19.44 MHz (see Figure 37)
Single-Ended		13		mW	f_{REF} = 19.44 MHz
Output Distribution Driver On vs. Off					At 156.25 MHz
15 mA Mode		28		mW	
12.5 mA Mode		23		mW	
7.5 mA Mode		15		mW	
Auxiliary DPLL On vs. Off		1		mW	
Auxiliary Numerically Controlled Oscillator (NCO) to Mx Pin On vs. Off		1		mW	Fundamental set to 50 kHz
Auxiliary Time to Digital Converter (TDC) Input from Mx Pin On vs. Off		1		mW	Input frequency (f_{IN}) = 10 MHz, auxiliary TDC rate = 200 kHz

SYSTEM CLOCK INPUTS, XOA AND XOB

Table 5.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
SYSTEM CLOCK MULTIPLIER					
Output Frequency Range	2250		2415	MHz	The frequency range of the internal voltage controlled oscillator (VCO) places limits on the choice of the system clock input frequency
Phase Frequency Detector (PFD) Rate	20		300	MHz	
SYSTEM CLOCK INPUT DIRECT PATH					The direct path requires ac coupling to the XOA and XOB pins (see Figure 31). Do not enable the frequency doubler when using the direct path
Input Frequency Range	20		300	MHz	Degraded phase noise performance typically occurs for frequencies below 50 MHz appearing at the input of the phase frequency detector of the system clock PLL; for optimal phase noise performance, the recommendation is to use the quartz crystal resonator path (instead of the direct path), a crystal resonator with a resonant frequency ≥50 MHz, the frequency doubler enabled, and the device configured to use the system clock Compensation Method 3 (see the System Clock Compensation section)
Input Duty Cycle	40		60	%	
Self Biased Common-Mode Voltage		0.75		V	Internally generated
Input Voltage					For dc-coupled, single-ended operation
High	0.9			V	
Low			0.5	V	
Differential Input Voltage Sensitivity	250			mV p-p	Minimum voltage swing required (as measured with a differential probe) across the XOA/XOB pins to ensure switching between logic states; the instantaneous voltage on either pin must not exceed 1.2 V; accommodate the single-ended input by ac grounding the complementary input; 800 mV p-p recommended for optimal jitter performance
Slew Rate for Sinusoidal Input					
Functional		>6		V/µs	Required to provide a locked and stable system clock
Operational		>31		V/µs	Required for <1 dBc phase noise degradation
System Clock Input Divider (J Divider) Frequency	100			MHz	
Input Resistance		5		kΩ	
SYSTEM CLOCK INPUT QUARTZ CRYSTAL RESONATOR PATH					The crystal resonator path includes an optional frequency doubler
Resonator Frequency Range	25		80	MHz	Fundamental mode, AT cut crystal
Crystal Motional Resistance			100	Ω	For crystal resonant frequency > 52 MHz, the motional resistance must not exceed 50 Ω and the crystal load capacitance (C_L) must not exceed 8 pF

REFERENCE INPUTS

Table 6.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
DIFFERENTIAL MODE					Differential mode specifications assume ac coupling of the input signal to the reference input pins
Frequency Range					
Sinusoidal Input			750	MHz	Lower limit dependent on input slew rate
Low Voltage Positive Emitter Coupled Logic (LVPECL) Input	1		750×10^6	Hz	Lower limit dependent on ac coupling
LVDS Input	1		500×10^6	Hz	Assumes an LVDS minimum of 494 mV p-p differential amplitude; lower limit dependent on ac coupling
Slew Rate		>4.1		V/µs	Required for proper reference monitor operation; output jitter degradation can occur for slew rates < 35 V/µs
Common-Mode Input Voltage		0.64		V	Internally generated self bias voltage
Differential Input Amplitude					Peak-to-peak differential voltage swing across pins required to ensure switching between logic levels as measured with a differential probe; instantaneous voltage on either pin must not exceed 1.3 V
f_{IN} < 500 MHz	350		2100	mV p-p	
f_{IN} = 500 MHz to 750 MHz	500		2100	mV p-p	
Differential Input Voltage Hysteresis		55	100	mV	
Input Resistance		16		kΩ	Equivalent differential input resistance
Input Pulse Width					
LVPECL	600			ps	
LVDS	900			ps	
DC-COUPLED, LVDS-COMPATIBLE MODE					Applies for dc coupling to an LVDS source
Frequency Range	1		450×10^6	Hz	
Slew Rate		>1.2		V/µs	Required for proper reference monitor operation; output jitter degradation can occur for slew rates < 35 V/µs
Common-Mode Input Voltage	1.125		1.375	V	
Differential Input Amplitude	400		1200	mV p-p	Differential voltage across pins required to ensure switching between logic levels; instantaneous voltage on either pin must not exceed the supply rails
Differential Input Voltage Hysteresis		55	100	mV	
Input Resistance		16		kΩ	
Input Pulse Width	1			ns	
SINGLE-ENDED MODE					Single-ended mode specifications assume dc coupling of the input signal to the reference input pins
1.2 V CMOS					
Frequency Range	1		500×10^6	Hz	Assumes dc-coupled input signal to the reference input pins
Input Voltage					
High, V_{IH}	0.78		1.38	V	
Low, V_{IL}			0.42	V	
Input Resistance		30		kΩ	
Slew Rate		>8		V/µs	Required for proper reference monitor operation; output jitter degradation can occur for slew rates < 35 V/µs
Input Pulse Width	900			ps	
1.8 V CMOS					
Frequency Range	1		500×10^6	Hz	Assumes dc-coupled input signal to the reference input pins

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
Input Voltage					
High, V_{IH}	1.17		2.07	V	
Low, V_{IL}			0.63	V	
Input Resistance		30		kΩ	
Slew Rate		>8		V/µs	Required for proper reference monitor operation; output jitter degradation can occur for slew rates <35 V/µs
Input Pulse Width	900			ps	
AC-Coupled					
Frequency Range	1		500×10^6	Hz	Applies to 1.2 V CMOS ac-coupled
Input Amplitude	360		1200	mV	Peak-to-peak single-ended voltage swing; instantaneous voltage must not exceed 1.3 V
Input Impedance		15		kΩ	
Input Pulse Width	900			ps	
Slew Rate		>8		V/µs	Required for proper reference monitor operation; output jitter degradation can occur for slew rates < 35 V/µs
Common-Mode Voltage		610		mV	Internally generated self bias voltage; applies to 1.2 V CMOS ac-coupled
AUXILIARY REFERENCES					Applies to the auxiliary Reference 0 (REF0) to auxiliary Reference 3 (REF3) inputs via appropriately configured Mx control pins (see the Reference Clock Input Resources section)
Frequency Range			225	MHz	
Input Voltage					
High, V_{IH}	VDDIOx – 0.4			V	
Low, V_{IL}			0.4	V	
Slew rate		>8		V/µs	Required for proper reference monitor operation; output jitter degradation can occur for slew rates < 35 V/µs; applies for VDDIOB = 1.8 V, 2.5 V, or 3.3 V
REFERENCE DEMODULATOR					f_{OUT} is the Q divider output frequency; f_{SYS} is the system clock frequency and must be in the 2250 MHz to 2415 MHz range; $f_{SYS} = f_{OSC} \times K/J$, where f_{OSC} is the frequency of the system clock oscillator connected to the XOA/XOB pins, K is the feedback divider ratio, and J is the system clock (SYSCLK) input scale factor; when the SYSCLK doubler is disabled, J is the value of the J divider (1, 2, 4, or 8); when the SYSCLK doubler is enabled, J = ½
Carrier Frequency (Synchronization Edge = 1, 2, 3)					Band 0 and Band 1 (see the Demodulator Bandwidth section)
Band 0					
DC Balanced Modulation	0.5		40	MHz	
Unbalanced Modulation	0.5		60	MHz	
Band 1					
DC Balanced Modulation	1		120	MHz	
Unbalanced Modulation	1		180	MHz	
Embedded Clock Rate	1		$f_{CAR}/8$	Hz	f_{CAR} is the unmodulated input frequency
Duty Cycle Deviation					S is the decimal value of Bits[1:0] in Register 0x0302, Register 0x0303, Register 0x0306, or Register 0x0307 (see the Demodulator Sensitivity section); B depends on the selected band, where B is $f_{SYS}/3$ for Band 0 or f_{SYS} for Band 1 (see the Demodulator Bandwidth section)
DC Balanced Modulation	(5 + S)/3B		$1/(4f_{CAR})$	sec	
Unbalanced Modulation	(5 + S)/2B		$1/(4f_{CAR})$	sec	
Polarity Detection Enabled	(5 + S)/B		$1/(4f_{CAR})$	sec	

REFERENCE TO REFERENCE COUPLING

The reference to reference coupling measurement always involves two reference inputs: a target reference input and an aggressor reference input. The device configuration comprises a frequency translation from the target reference input to an appropriate OUTxyP/OUTxyN output pair. The target reference frequency is 10 MHz, whereas the aggressor reference frequency deviates from the target reference frequency over a range of ±100 ppm in steps of 0.5 ppm. The values shown in Table 7 are the levels of the worst spur appearing at the OUTxyP/OUTxyN output pair, measured at the appropriate offset frequency relative to the aggressor reference input frequency. To prevent the DPLL from suppressing the coupling spur, the DPLL loop bandwidth is always twice the aggressor reference offset frequency.

Table 7.

Parameter	Min	Typ	Max	Unit
TARGET REFERENCE AND INPUT MODE				
REFA 1.2 V CMOS				
Aggressor Reference and Input Mode				
REFAA 1.2 V CMOS		−62		dBc
REFAA 1.8 V CMOS		−59		dBc
REFAA Single-Ended AC-Coupled		−69		dBc
REFA 1.8 V CMOS				
Aggressor Reference and Input Mode				
REFAA 1.2 V CMOS		−65		dBc
REFAA 1.8 V CMOS		−66		dBc
REFAA Single-Ended AC-Coupled		−67		dBc
REFA Single-Ended AC-Coupled				
Aggressor Reference and Input Mode				
REFAA 1.2 V CMOS		−60		dBc
REFAA 1.8 V CMOS		−58		dBc
REFAA Single-Ended AC-Coupled		−71		dBc
REFA Differential AC-Coupled				
Aggressor Reference and Input Mode				
REFB Differential AC-Coupled		−76		dBc
REFB Differential LVDS		−76		dBc
REFBB 1.2 V CMOS		−75		dBc
REFBB 1.8 V CMOS		−74		dBc
REFBB Single-Ended AC-Coupled		−76		dBc
REFA Differential LVDS				
Aggressor Reference and Input Mode				
REFB Differential AC-Coupled		−76		dBc
REFB Differential LVDS		−76		dBc
REFBB 1.2 V CMOS		−76		dBc
REFBB 1.8 V CMOS		−75		dBc
REFBB Single-Ended AC-Coupled		−68		dBc
REFAA 1.2 V CMOS				
Aggressor Reference and Input Mode				
REFB Differential AC-Coupled		−76		dBc
REFB Differential LVDS		−75		dBc
REFBB 1.2 V CMOS		−74		dBc
REFBB 1.8 V CMOS		−76		dBc
REFBB Single-Ended AC-Coupled		−76		dBc

Parameter	Min	Typ	Max	Unit
REFAA 1.8 V CMOS				
Aggressor Reference and Input Mode				
REFB Differential AC-Coupled		−76		dBc
REFB Differential LVDS		−76		dBc
REFBB 1.2 V CMOS		−76		dBc
REFBB 1.8 V CMOS		−76		dBc
REFBB Single-Ended AC-Coupled		−76		dBc
REFAA Single-Ended AC-Coupled				
Aggressor Reference and Input Mode				
REFB Differential AC-Coupled		−75		dBc
REFB Differential LVDS		−75		dBc
REFBB 1.2 V CMOS		−76		dBc
REFBB 1.8 V CMOS		−76		dBc
REFBB Single-Ended AC-Coupled		−76		dBc
REFB or REFBB Using Any Single-Ended Input Mode				
Aggressor Reference (Not REFA or REFAA)				
M0, M1, M2, or M3 with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−66		dBc
REFBB Using Any Single-Ended Input Mode				
Aggressor Reference (Not REFA or REFAA)				
M0, M1, or M2 (Not M3) with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−74		dBc
REFB and REFBB Using Any Differential Input Mode				
Aggressor Reference (Not REFA or REFAA)				
M0, M1, M2, or M3 with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−70		dBc
M0 with VDDIOx = 1.8 V or 3.3 V				
Aggressor Reference (Not REFA or REFAA)				
REFB or REFBB Using Any Input Mode; or M1, M2, or M3 with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−65		dBc
M0 with VDDIOx = 1.8 V or 3.3 V				
Aggressor Reference (Not REFA or REFAA)				
REFB or REFBB Using Any Input Mode; or M2 or M3 (Not M1) with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−70		dBc
M1 with VDDIOx = 1.8 V or 3.3 V				
Aggressor Reference (Not REFA or REFAA)				
REFB or REFBB Using Any Input Mode; or M0, M2, or M3 with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−68		dBc
M2 with VDDIOx = 1.8 V or 3.3 V				
Aggressor Reference (Not REFA or REFAA)				
REFB or REFBB Using Any Input Mode; or M1, M3, or M4 with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−67		dBc
M3 with VDDIOx = 1.8 V or 3.3 V				
Aggressor Reference (Not REFA or REFAA)				
REFB or REFBB Using Any Input Mode; or M0, M1, or M2 with VDDIOx = 1.8 V or 3.3 V; and Connected to Any Auxiliary Reference		−69		dBc
M3 with VDDI0x = 3.3 V (Not 1.8 V)				
Aggressor Reference (Not REFA, REFAA, REFB, or REFBB)				
M0, M1, or M2 with VDDIOx = 3.3 V (Not 1.8 V); and Connected to Any Auxiliary Reference		−80		dBc

REFx TO REFx INPUT TIMING SKEW

The values shown in Table 8 indicate the internal timing skew between a clock signal applied to the REFA input (base input) and the same clock signal applied to the REFAA, REFB, or REFBB input (target input) with both clock signals time aligned at the device pins. The top section of the table indicates timing skew for REFA configured as the CMOS 1.2 V base input against REFx (x = AA, B, BB) as the target input with various target input configurations. The bottom section of the table captures timing skew for REFA as the base input with input configurations other than CMOS 1.2 V against REFB CMOS 1.2 V as the target input, which establishes REFB CMOS 1.2 V as a common target for different REFA input configurations. The common target allows the user to extrapolate the skew to other target inputs when REFA is configured for other than CMOS 1.2 V.

Table 8.

Parameter	Min	Typ	Max	Unit
TIMING SKEW—REFA CMOS 1.2 V BASE REFERENCE INPUT				
Target Reference Input				
REFAA CMOS 1.2 V	−190	−70	+45	ps
REFAA CMOS 1.8 V	−95	+130	+330	ps
REFAA AC-Coupled Single-Ended	−70	+55	+215	ps
REFB CMOS 1.2 V	−155	−15	+125	ps
REFBB CMOS 1.2 V	−170	−30	+110	ps
REFB Differential, LVDS	−250	−100	+50	ps
REFB Differential, DC-Coupled	−190	+30	+270	ps
TIMING SKEW—OTHER BASE REFERENCE INPUT CONFIGURATIONS				
REFA CMOS 1.8 V Base Reference Input				
Target Reference Input				
REFB CMOS 1.2 V	−180	−5	+155	ps
REFA AC-Coupled, Single-Ended Base Reference Input				
Target Reference Input				
REFB CMOS 1.2 V	−200	−55	+95	ps
REFA Differential, LVDS Base Reference Input				
Target Reference Input				
REFB CMOS 1.2 V	40	195	340	ps
REFA Differential, DC-Coupled Base Reference Input				
Target Reference Input				
REFB CMOS 1.2 V	−145	+85	+285	ps

REFx TO AUXILIARY REFx INPUT TIMING SKEW

The values shown in Table 9 indicate the internal timing skew between a clock signal applied to the REFA input (configured for CMOS 1.2 V) and the same clock signal applied to an auxiliary REFx via an Mx pin with both clock signals time aligned at the device pins. Skew data for the various VDDIOB voltages is also captured in Table 9 because VDDIOB is the power supply for the Mx pins. Note that REFA CMOS 1.2 V is always the base input for a target Mx and auxiliary REFx combination. Skew relative to a different input configuration for REFA or to another REFx input (with various input configurations) can be inferred using Table 8 in conjunction with Table 9. Skew data appears only for M0, M1, M2, and M3 because these are the only Mx pins recommended for use with the auxiliary REFx references. The data reveals that operating with VDDIOB = 3.3 V yields lower REFx to auxiliary REFx timing skew than operating with VDDIOB = 1.8 V or 2.5 V.

Table 9.

Parameter	Min	Typ	Max	Unit
INTERNAL TIMING SKEW				
M0				
VDDIOB = 1.8 V				
Auxiliary REF0	950	1465	2040	ps
Auxiliary REF1	845	1355	1930	ps
Auxiliary REF2	715	1240	1820	ps
Auxiliary REF3	720	1250	1835	ps
VDDIOB = 2.5 V				
Auxiliary REF0	495	870	1255	ps
Auxiliary REF1	380	760	1150	ps
Auxiliary REF2	250	640	1045	ps
Auxiliary REF3	255	655	1075	ps
VDDIOB = 3.3 V				
Auxiliary REF0	330	610	925	ps
Auxiliary REF1	215	505	820	ps
Auxiliary REF2	95	385	715	ps
Auxiliary REF3	95	400	740	ps
M1				
VDDIOB = 1.8 V				
Auxiliary REF0	1000	1485	2050	ps
Auxiliary REF1	865	1355	1925	ps
Auxiliary REF2	800	1300	1885	ps
Auxiliary REF3	700	1200	1785	ps
VDDIOB = 2.5 V				
Auxiliary REF0	455	830	1265	ps
Auxiliary REF1	325	700	1135	ps
Auxiliary REF2	255	640	1085	ps
Auxiliary REF3	150	540	990	ps
VDDIOB = 3.3 V				
Auxiliary REF0	315	585	915	ps
Auxiliary REF1	185	460	795	ps
Auxiliary REF2	115	400	735	ps
Auxiliary REF3	35	310	635	ps
M2				
VDDIOB = 1.8 V				
Auxiliary REF0	1020	1515	2075	ps
Auxiliary REF1	870	1390	1970	ps
Auxiliary REF2	825	1325	1890	ps
Auxiliary REF3	725	1235	1815	ps

Parameter	Min	Typ	Max	Unit
VDDIOB = 2.5 V				
Auxiliary REF0	485	860	1290	ps
Auxiliary REF1	365	735	1160	ps
Auxiliary REF2	295	690	1135	ps
Auxiliary REF3	190	580	1025	ps
VDDIOB = 3.3 V				
Auxiliary REF0	340	620	940	ps
Auxiliary REF1	215	500	815	ps
Auxiliary REF2	130	455	820	ps
Auxiliary REF3	35	340	680	ps
M3				
VDDIOB = 1.8 V				
Auxiliary REF0	1095	1585	2135	ps
Auxiliary REF1	955	1460	2035	ps
Auxiliary REF2	860	1380	1970	ps
Auxiliary REF3	790	1310	1905	ps
VDDIOB = 2.5 V				
Auxiliary REF0	550	945	1410	ps
Auxiliary REF1	415	820	1300	ps
Auxiliary REF2	340	750	1235	ps
Auxiliary REF3	275	685	1170	ps
VDDIOB = 3.3 V				
Auxiliary REF0	370	705	1105	ps
Auxiliary REF1	285	585	960	ps
Auxiliary REF2	210	540	930	ps
Auxiliary REF3	145	485	890	ps

REFERENCE MONITORS

Table 10.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
REFERENCE MONITORS					
Reference Monitor					t_{PFD} is the nominal phase detector period; $t_{PFD} = R/f_{REF}$, where R is the frequency division factor determined by the R divider, and f_{REF} is the frequency of the active reference
Loss of Reference Detection Time		$4.9 + 0.13 \times t_{PFD}$		µs	$t_{PFD} = R/f_{REF}$ with f_{REF} in units of MHz
Frequency Out of Range Limit	5×10^{-8}		0.015	$\Delta f/f_{REF}$	Minimum deviation from f_{REF} for an unfaulted reference to become faulted; programmable with the lower limit subject to quality of the system clock (or the source of system clock compensation); Δf = frequency deviation from f_{REF}; to prevent false reference invalidation due to the inherent peak jitter of the TDC (TDC_{Jpk}), requires $f_{TDC} < \text{Threshold} \times 10^{-9}/TDC_{Jpk}$, where f_{TDC} is the frequency applied at the TDC input (see the Reference Period Threshold section for the threshold parameter); minimum = 5×10^{-8} requires $f_{TDC} < 200$ Hz
Frequency In Range Limit	$(5 \times 10^{-8}) \times (1 - SF)$		$0.015 \times (1 - SF)$	$\Delta f/f_{REF}$	Maximum deviation from f_{REF} for a faulted reference to become unfaulted; lower limit subject to quality of the system clock (or the source of system clock compensation); SF = programmable hysteresis scale factor (see Table 54); Δf = frequency deviation from f_{REF}; to prevent false reference revalidation due to TDC_{Jpk} requires $f_{TDC} < \text{Threshold} \times 10^{-9} \times SF/TDC_{Jpk}$, where f_{TDC} is the frequency applied at the TDC input (see the Hysteresis section for SF); minimum = $(5 \times 10^{-8}) \times (1 - SF)$ requires $f_{TDC} < SF \times 200$ Hz
Validation Timer	0.001		1048	sec	Programmable in 1 ms steps; maximum is $(2^{20} - 1)$ ms
Excessive Jitter Alarm Threshold	1		65,535	ns	Programmable in 1 ns steps

DISTRIBUTION CLOCK OUTPUTS

Table 11.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
DIFFERENTIAL MODE					All testing is both ac-coupled and dc-coupled
Output Frequency					Frequency range determined by driver functionality; actual frequency synthesis may be limited by the APLL VCO frequency range
CML	1		500×10^6	Hz	Terminated per Figure 40
High Speed Current Steering Logic (HCSL)	1		500×10^6	Hz	Terminated per Figure 39
Differential Output Voltage Swing					Voltage between output pins measured with output driver static; peak-to-peak differential output amplitude is twice that shown when driver is toggling and measured using a differential probe
Output Current = 7.5 mA					
HCSL	312	368	402	mV	Terminated per Figure 39
CML	257	348	408	mV	Terminated to VDD (nominal 1.8 V) per Figure 40
Output Current = 15 mA					
HCSL	631	745	809	mV	Terminated per Figure 39
CML	578	729	818	mV	Terminated to VDD (nominal 1.8 V) per Figure 40
Common-Mode Output Voltage					
Output Current = 7.5 mA					
HCSL	155	184	201	mV	Terminated per Figure 39
CML	VDD − 208	VDD − 188	VDD − 169	mV	Terminated to VDD (nominal 1.8 V) per Figure 40 (maximum common-mode voltage case occurs at the minimum amplitude)
Output Current = 15 mA					
HCSL	316	372	405	mV	Terminated per Figure 39
CML	VDD − 416	VDD − 371	VDD − 327	mV	Terminated to VDD (nominal 1.8 V) per Figure 40 (maximum common-mode voltage case occurs at the minimum amplitude)
Rise/Fall Time					Rise/fall times measured on a 50 MHz output signal (parasitic load ≈ 5 pF)
HCSL					
7.5 mA Drive Current		228	342	ps	
15 mA Drive Current		218	340	ps	
CML					
7.5 mA Drive Current		205	323	ps	
15 mA Drive Current		185	303	ps	
Duty Cycle					
HCSL					
Frequency with 7.5 mA Drive Current					
≤100 MHz	47.5	50	52.5	%	
500 MHz	43.8	50	55	%	
Frequency with 15 mA Drive Current					
≤100 MHz	47.5	50	52.5	%	
500 MHz	46.5	50	52.5	%	
CML					
Frequency with 7.5 mA Drive Current					
≤100 MHz	47.5	50	52.5	%	
500 MHz	45	50	55	%	
Frequency with 15 mA Drive Current					
≤100 MHz	47.5	50	52.5	%	
500 MHz	45	50	54	%	

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
SINGLE-ENDED MODE					
Output Frequency	1		500×10^6	Hz	Frequency range determined by driver functionality; actual frequency synthesis may be limited by the APLL VCO frequency range
Output Current = 12.5 mA					
Voltage Swing (Peak to Peak)					
HCSL Driver Mode	509	584	634	mV	Each output terminated per Figure 44 with load resistor (R_L) = 50 Ω
CML Driver Mode	456	565	644	mV	Each output terminated per Figure 44 with R_L = 50 Ω connected to VDD (nominal 1.8 V) instead of ground
Voltage Swing Midpoint					
HCSL Driver Mode	255	292	317	mV	Each output terminated per Figure 44 with R_L = 50 Ω
CML Driver Mode	VDD − 325	VDD − 291	VDD − 266	mV	Each output terminated per Figure 44 with R_L = 50 Ω connected to VDD (nominal 1.8 V) instead of ground
Output Current = 15 mA					
Voltage Swing (Peak to Peak)					
HCSL Driver Mode	645	734	796	mV	Each output terminated per Figure 44 with R_L = 50 Ω
CML Driver Mode	589	721	815	mV	Each output terminated per Figure 44 with R_L = 50 Ω connected to VDD (nominal 1.8 V) instead of ground
Voltage Swing Midpoint					
HCSL Driver Mode	322	367	398	mV	Each output terminated per Figure 44 with R_L = 50 Ω
CML Driver Mode	VDD − 411	VDD − 367	VDD − 334	mV	Each output terminated per Figure 44 with R_L = 50 Ω connected to VDD (nominal 1.8 V) instead of ground
Rise/Fall Time					Rise/fall times based on 50 MHz output signal (parasitic load ≈ 5 pF)
HCSL					
12.5 mA Drive Current		183	367	ps	
15 mA Drive Current		181	348	ps	
CML					
12.5 mA Drive Current		178	353	ps	
15 mA Drive Current		183	393	ps	
Duty Cycle					
HCSL					
Frequency with 12.5 mA Drive Current					
50 MHz	46.5	49	51.5	%	
100 MHz	43.8	48.5	53.8	%	
500 MHz	36.2	42	47.8	%	
Frequency with 15 mA Drive Current					
50 MHz	46.5	49	51.5	%	
100 MHz	46.5	49	51.3	%	
500 MHz	38	43.5	50	%	
CML					
Frequency with 12.5 mA Drive Current					
50 MHz	48.5	51	53.5	%	
100 MHz	49	51.5	54	%	
500 MHz	50	58	63	%	
Frequency with 15 mA Drive Current					
50 MHz	48.5	51	53.5	%	
100 MHz	49	51.5	54	%	
500 MHz	50	57.5	62.5	%	

OUTPUT TO OUTPUT TIMING SKEW

The values in Table 12 indicate the time offset of the target output clock edge relative to the base output clock edge. f_{REF} = 25 MHz; f_{REF} applied to a single reference input with that reference assigned to Digital Phase-Locked Loop 0 (DPLL0) and Digital Phase-Locked Loop 1 (DPLL1); DPLL0 and DPLL1 operate in zero delay (hitless mode); PLL0 and PLL1 frequency translation is such that the frequency of the OUTxyP and OUTxyN pins (f_{OUTxy}) = f_{REF} for all outputs.

Table 12.

Parameter	Min	Typ	Max	Unit
TARGET OUTPUT[1]				
OUT1A Differential CML				
Base Output OUT0A				
Differential CML	−21	+53	+123	ps
Differential HCSL	−89	−36	+14	ps
OUT1A Differential HCSL				
Base Output OUT0A				
Differential CML	3	83	174	ps
Differential HCSL	−61	−5	+50	ps
OUT1A Single-Ended CML				
Base Output OUT0A				
Differential CML	−24	+48	+109	ps
Differential HCSL	−120	−43	+25	ps
Single-Ended CML	−18	+63	+123	ps
Single-Ended HCSL	−132	−56	+8	ps
OUT1A Single-Ended HCSL				
Base Output OUT0A				
Differential CML	93	175	254	ps
Differential HCSL	16	84	190	ps
Single-Ended CML	92	191	282	ps
Single-Ended HCSL	0	77	134	ps
OUT0B or OUT0C Differential CML				
Base Output OUT0A				
Differential CML	37	60	97	ps
Differential HCSL	−67	−30	+2	ps
OUT0B or OUT0C Differential HCSL				
Base Output OUT0A				
Differential CML	53	87	118	ps
Differential HCSL	−25	−3	+22	ps
OUT0B or OUT0C Single-Ended CML				
Base Output OUT0A				
Differential CML	15	44	66	ps
Differential HCSL	−84	−48	−21	ps
Single-Ended CML	37	67	98	ps
Single-Ended HCSL	−100	−53	−24	ps
OUT0B or OUT0C Single-Ended HCSL				
Base Output OUT0A				
Differential CML	135	175	212	ps
Differential HCSL	53	80	112	ps
Single-Ended CML	162	198	273	ps
Single-Ended HCSL	46	79	104	ps

[1] OUTxy refers to Output xy where x is 0 or 1, and y is A, B, or C. See the Distribution Clock Output Drivers Overview section for more information about the output drivers.

OUTPUT TIMING SKEW BETWEEN Mx PINS AND OUTxyP AND/OR OUTxyN PINS

The timing skew between the Mx pins and the OUTxyP and/or OUTxyN pins is important when using an auxiliary NCO as a reference input to one of the DPLLs and simultaneously using an Mx pin as an output for the auxiliary NCO (see the Auxiliary NCOs section and the Status and Control Pins section). To obtain the data given in Table 13, the device configuration uses an auxiliary NCO as the reference input to DPLL0 and DPLL1, with both phase-locked loop (PLL) channels employing a 1:1 frequency translation from the auxiliary NCO to the distribution output. The auxiliary NCO output routes to one of the Mx pins that is configured as a status output. Because of the 1:1 frequency translation, the Mx pin output signal and the OUTxyP/OUTxyN output signal have the same frequency. The data in Table 13 is for the timing skew between the OUT0AP, OUT0AN, OUT1AP, or OUT1AN pin and the Mx pin carrying the auxiliary NCO output signal. Note that the auxiliary NCO exhibits ~1.25 ns quantization, and the tabulated data in Table 13 is an average of the quantization variation.

Table 13.

Parameter	Min	Typ	Max	Unit
TIMING SKEW BETWEEN Mx AND OUT0AP/OUT0AN PINS				
VDDIOA and VDDIOB Supply Voltage				
1.8 V		−5.5		ns
2.5 V		−5.15		ns
3.3 V		−5.0		ns

TIME DURATION OF DIGITAL FUNCTIONS

Table 14.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
TIME DURATION OF DIGITAL FUNCTIONS					
EEPROM to Register Download Time		10		ms	Using the Typical Configuration conditions from Table 4
Power-On Reset (POR)			25	ms	Time from power supplies > 80% to release of internal reset
Mx Pin to RESETB Rising Edge Setup Time			1	ns	Mx refers to Pin M0 through Pin CSB/M6
Mx Pin to RESETB Rising Edge Hold Time			2	ns	
Multiple Mx Pin Timing Skew			39	ns	Applies only to multibit Mx pin functions
RESETB Falling Edge to Mx Pin High-Z Time			14	ns	
Time from Release of Power-Down to Completion of System Clock PLL Calibration		170		µs	Excludes time delay associated with the user programmable system clock stability timer (50 ms default)
Time from Release of Power-Down to System Clock PLL Locked and Calibrated		172		µs	Excludes time delay associated with the user programmable system clock stability timer (50 ms default)
Time for APLLx to Calibrate and Lock		5		ms	Time from assertion of IO update (with Bit 1 = 1 in Register 0x2100 and/or Register 0x2200) to APLLx indicating calibrated and locked status; uses the Typical Configuration conditions in Table 4; prerequisites = system clock PLL locked and stable, and DPLLx phase locked
TIME FROM START OF DPLL ACTIVATION TO ACTIVE PHASE DETECTOR OUTPUT					
Untagged Operation			10	t_{PFD}	$t_{PFD} = R/f_{REF}$, where R is the frequency division factor determined by the R divider, and f_{REF} is the frequency of the active reference
Tagged Operation			10	Tag period	Tag period = (tag ratio/f_{TAG}), where f_{TAG} is either f_{REF} (for tagged reference mode) or the feedback frequency of DPLLx, $f_{FEEDBACK}$ (for all other tagged modes); the tag ratio corresponds to the selection of f_{TAG} (as f_{REF} or $f_{FEEDBACK}$)

DPLL0 AND DPLL1 SPECIFICATIONS

Table 15.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
DPLL					
Digital Phase Detector (DPD) Input Frequency Range	1		2×10^5	Hz	
Loop Filter					
Profile 0					
Bandwidth	0.0001		1850	Hz	Programmable design parameter; (f_{PFD}/bandwidth) ≥ 20
Phase Margin		70		Degrees	
Closed-Loop Peaking		1.1		dB	
Profile 1					
Bandwidth	0.0001		305	Hz	Programmable design parameter; (f_{PFD}/bandwidth) ≥ 20
Phase Margin		88.5		Degrees	
Closed-Loop Peaking			0.1	dB	In accordance with Telcordia GR-253 jitter transfer specifications
DPLL NCO DIVISION RATIO					These specifications cover limitations on the freerun tuning word (FTW0) of DPLLx; the AD9546 evaluation software frequency planning wizard sets these values automatically for the user, and the AD9546 evaluation software is available for download from the AD9546 product page at www.analog.com/AD9546; NCO division = 2^{48}/FTW0, which takes the form of INT.FRAC, where INT is the integer portion, and FRAC is the fractional portion
NCO Integer	7		13		This is the integer portion of NCO division
NCO Fraction	0.05		0.95		This is the fractional portion of NCO division

DPLL LOCK DETECTION SPECIFICATIONS

Table 16.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
PHASE LOCK DETECTOR					
Threshold Programming Range	10		$2^{24} - 1$	ps	
Threshold Resolution		1		ps	
FREQUENCY LOCK DETECTOR					
Threshold Programming Range	10		$2^{24} - 1$	ps	
Threshold Resolution		1		ps	
PHASE STEP DETECTOR					
Threshold Programming Range	100		$2^{32} - 1$	ps	Setting this value too low causes false triggers
Threshold Resolution		1		ps	

DPLL PHASE CHARACTERISTICS

Table 17.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
MAXIMUM OUTPUT PHASE PERTURBATION					Assumes a jitter free reference; satisfies Telcordia GR-1244 requirements; 0 ppm frequency difference between references; reference switch initiated via register map (see UG-1793) by faulting the active reference input
Phase Refinement Disabled					50 Hz DPLL loop bandwidth; normal phase margin mode; frequency translation = 19.44 MHz to 155.52 MHz; 49.152 MHz signal generator used for system clock source
Peak		±20	±175	ps	
Steady State					
Phase Buildout Operation		±18	±160	ps	
Hitless Operation		0		ps	
Phase Refinement Enabled					50 Hz DPLL loop bandwidth; high phase margin mode; phase refinement iterations = 4; frequency translation = 19.44 MHz to 155.52 MHz; 49.152 MHz signal generator used for system clock source
Peak		±5	±44	ps	
Steady State					
Phase Buildout Operation		±4	±43	ps	
Hitless Operation		0		ps	
PHASE SLEW LIMITER	0.001		250	µs/sec	See the AN-1420 Application Note, *Phase Buildout and Hitless Switchover with Digital Phase-Locked Loops (DPLLs)*

DPLL PROPAGATION DELAY

DPLL propagation delay is the difference in clock edge timing at the output (OUTxyP/OUTxyN) relative to the REFA input (1.2 V CMOS mode). To accommodate edge timing measurements, the DPLL is configured for internal zero delay (hitless) operation and with a 1:1 frequency translation ratio: $f_{REF} = f_{OUT} = 10$ MHz. Delay compensation is not active (see the Delay Compensation section).

Absolute delay values use Q Divider QxA as feedback for DPLLx, where x is 0 or 1. The data shows propagation delay for various output driver configurations (single-ended, differential), driver modes (HCSL, CML), and driver output current. The data also shows the propagation delay to outputs associated with a Q divider that is not the Q divider used for DPLL feedback. Also captured is the delay when using DPLL1 instead of DPLL0.

Relative delay values show the impact of using other Q dividers for DPLL feedback (relative to QxA for DPLLx, where x is 0 or 1).

For the propagation delay variation due to using a reference input mode other than 1.2 V CMOS, use Table 8.

Table 18.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
ABSOLUTE DELAY					
REFA to OUT0AP					Q0A = DPLL0 feedback; driver output = single-ended
CML at 12.5 mA	1055	1265	1430	ps	
CML at 15 mA	1065	1270	1430	ps	
HCSL at 12.5 mA	1180	1385	1550	ps	
HCSL at 15 mA	1145	1345	1515	ps	
REFA to OUT0AN	1185	1390	1565	ps	Driver output = single-ended HCSL at 15 mA
REFA to OUT0A[1]					Driver output = differential
CML at 7.5 mA	805	1000	1180	ps	
CML at 15 mA	760	955	1130	ps	
HCSL at 7.5 mA	845	1030	1200	ps	
HCSL at 15 mA	785	975	1155	ps	
REFA to OUT0B[1]	770	960	1140	ps	Driver output = differential HCSL at 15 mA
REFA to OUT0C[1]	785	970	1145	ps	Driver output = differential HCSL at 15 mA
REFA to OUT1AP	1150	1370	1565	ps	Q1A = DPLL1 feedback; driver output = single-ended HCSL at 15 mA
REFA to OUT1A[1]	810	985	1140	ps	Driver output = differential HCSL at 15 mA
REFA to OUT1B[1]	740	970	1180	ps	Driver output = differential HCSL at 15 mA
RELATIVE DELAY[2]					With respect to Q0A (PLL0) or Q1A (PLL1)
QxAA		14		ps	
QxB		9		ps	
QxBB		23		ps	
Q0C		36		ps	
Q0CC		50		ps	

[1] OUTxy refers to Output xy where x is 0 or 1, and y is A, B, or C. See the Distribution Clock Output Drivers Overview section for more information about the output drivers.
[2] Qxy denotes a specific Q divider in the clock distribution section, where x is 0 or 1, and y is AA, B, BB, C, or CC.

DPLL PROPAGATION DELAY VARIATION

The PLL channels of the AD9546 have an inherent propagation delay (see Table 18) that can be nominally offset using the skew adjustment feature in the source profiles of the device. Dynamic variations that are proportional to temperature change can be mitigated with the delay compensation mechanism of the AD9546.

A polynomial with programmable coefficients of temperature (see the Delay Compensation section) determines the response of the delay compensation circuit. An additive inverse of the polynomial function derives from a best fit algorithm applied to the uncompensated delay variation measurements. Active delay compensation uses the additive inverse polynomial in an attempt to nullify the delay variation over temperature. For a second-order polynomial, the measured data without delay compensation in Table 19 is the first-order representation for the various device configurations. A repeat of the same measurement with the inverse coefficient programmed yields the slopes of the residual delay in the specifications with delay compensation in Table 19. See Figure 25 and Figure 26 for the related plots.

The tabulated data with delay compensation in Table 19 reflects an average of several devices that are programmed with the same delay compensation coefficients. The tabulated residual errors are greater than expected for coefficients that are optimized for an individual device.

The DPLL propagation delay variation results depend on the device being configured with a 1:1 frequency translation ratio, where $f_{REF} = f_{OUT}$ = 10 MHz, the reference input configured to 1.2V CMOS, and DPLL configured to zero delay (hitless). A die temperature range of −40°C to +105°C applies for both the uncompensated and compensated results.

Table 19.

Parameter	Min	Typ	Max	Unit
OUTPUT DRIVER MODE				
Differential				
CML				
Without Delay Compensation		0.517		ps/°C
With Delay Compensation		−0.0155		ps/°C
HCSL				
Without Delay Compensation		0.624		ps/°C
With Delay Compensation		−0.0253		ps/°C
Single-Ended				
CML				
Without Delay Compensation		0.681		ps/°C
With Delay Compensation		0.0208		ps/°C
HCSL				
Without Delay Compensation		0.861		ps/°C
With Delay Compensation		0.0288		ps/°C

HOLDOVER SPECIFICATIONS

Table 20.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
HOLDOVER SPECIFICATIONS					
Initial Frequency Accuracy		±0.01	±0.1	ppb	AD9546 is configured using Device Configuration 1 from Table 37; excludes frequency drift of SYSCLK source; excludes frequency drift of input reference prior to entering holdover; 160 ms history timer; history holdoff setting = 8 decimal in Register 0x1010 and Register 0x1410; three features (bits) are Logic 1: Bit 4 in Register 0x100E and Register 0x140E, Bit 3 in Register 0x100E and Register 0x140E, and Bit 5 in Register 0x100E and Register 0x140E
Relative Frequency Accuracy Between Channels					
Cascaded DPLL Operation			0	ppb	
Noncascaded DPLL Operation		<1		ppb	f_{OSC} = 52 MHz (stable external oscillator); f_{OUT} = 155.52 MHz; f_{REF} = 38.88 MHz; DPLL loop bandwidth = 50 Hz; DPLL history accumulation timer = 1 ms; DPLL history holdoff value = 1
History Averaging Window	0.001		26,8435	sec	

ANALOG PLL (APLL0 AND APLL1) SPECIFICATIONS

Table 21.

Parameter	Min	Typ	Max	Unit
VCO FREQUENCY RANGE				
Analog PLL0 (APLL0)	2424		3232	MHz
Analog PLL1 (APLL1)	3232		4040	MHz
PFD INPUT FREQUENCY RANGE	162		350	MHz
LOOP BANDWIDTH		260		kHz
PHASE MARGIN		68		Degrees

OUTPUT CHANNEL DIVIDER SPECIFICATIONS

Table 22.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
OUTPUT PHASE ADJUST STEP SIZE	1			t_{VCO}	VCO time period (t_{VCO}) = 1/(APLLx VCO frequency), where x = 0 or 1
MODULATOR					
Carrier Frequency			$f_{VCO}/16$	Hz	The maximum value is the APLLx VCO frequency divided by 16, where x = 0 or 1; f_{VCO} is the VCO frequency
Time Deviation (from Nominal Duty Cycle of Carrier Clock)	0		$2^{16} - 1$	t_{VCO}	t_{VCO} = 1/(APLLx VCO frequency), where x = 0 or 1; the maximum value is limited to the Qxy divide ratio − 1; Qxy refers to the distribution dividers on each output, where x is either 0 (for PLL0) or 1 (for PLL1), and y is A, B, or C
Embedded Frequency	$f_{OUT}/(2^{28} - 1)$		$f_{OUT}/6$	Hz	f_{OUT} is the output frequency

TIME TO DIGITAL CONVERTER (TDC) SPECIFICATIONS

Table 23.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
TDCs					
Frequency Range			200	kHz	
Pulse Width	5			ns	
Time Stamp Jitter					
Root Mean Square (TDC_{Jrms})		5	11	ps	
Peak (TDC_{Jpk})			250	ps	
Retrigger Blackout Period	4.9			µs	Interval between successive rising edges at the TDC input
Start-Up Time		6		ms	The time required, after the system clock PLL locks, before the TDCs can generate time stamps

AUXILIARY NCO SPECIFICATIONS

Table 24.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
NCOs					Auxiliary NCO 0 and auxiliary NCO 1
Fundamental Frequency					
Range	1		65,535	Hz	
Quantization	1.27		1.37	pHz	pHz is picohertz
Phase Slew Limiter	5		2^{32}	ppb	Actual units are fractional part (ideal)/actual unit interval (UI)
Output Signal					
Pulse Width	38			ns	
Duty Cycle	45		55	%	Assumes the device is programmed to produce a nominal pulse width of 50%
Quantization			1.4	ns	

COMMON CLOCK DPLL SPECIFICATIONS

Table 25.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
PHASE LOCK DETECTOR					
Threshold Programming Range	0		65,535	ps	Programmed values < 10 ps not recommended
Threshold Resolution		1		ps	
Post Detection Delay Range	0		65,535	ms	
REFERENCE MONITOR					
Loss of Reference Detection Time		$2 \times t_{PFD}$		µs	$t_{PFD} = R/f_{REF}$, where R is the reference input divider value and f_{REF} is the reference input frequency in MHz
Input Period Deviation Limit			3.125	%	Relative to the programmed value of the reference period, t_{REF}
Reference Validation Delay		2		ms	Applies only to the common clock DPLL secondary frequency source (see the Common Clock Reference Switchover section)

COMMON CLOCK SYNCHRONIZER (CCS) SPECIFICATIONS

Table 26.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
ABSOLUTE LATENCY					Time from asserting an input/output (IO) update (Bit 0 in Register 0x000F) to the CSS indicating ready status (asserting IO update constitutes insertion of a time value; see the synchronization time input in Figure 51)
Fractional Seconds Format			1	µs	
PTP Format			3	µs	
BLACKOUT PERIOD	10			µs	The time interval following assertion of an input synchronization event (see the synchronization input in Figure 51) before asserting an IO update to insert a time value (see the synchronization time input in Figure 51)
GUARD PARAMETERS					
Maximum Magnitude Guard					
Range	0		953	ns	Minimum value applies when maximum magnitude guard is disabled; maximum value is $(2^{20} - 1) \times 2^{-40}$ sec
Resolution		909		fs	2^{-40} sec
Latency Guard					
Range	0		0.99998	sec	Minimum value applies when latency guard is disabled; maximum value is $1 - 2^{-16}$ sec
Resolution		15.3		µs	2^{-16} sec
SLEW LIMITER					
Range	0		244	µs/sec	Minimum value applies when slew limiter is disabled; Max value is $(2^{24} - 1) \times 2^{-36}$ sec
Resolution		14.6		ps/sec	2^{-36} sec/sec
TIME OFFSET					
Range	−7.63		+7.63	µs	Minimum value is -2^{-17} sec; maximum value is $(2^{31} - 1) \times 2^{-48}$ sec
Resolution		3.55		fs	2^{-48} sec
TIME SKEW					
Range	−29.8		+29.8	ns	Minimum value is -2^{-25}s; maximum value is $(2^{23} - 1) \times 2^{-48}$ sec
Resolution		3.55		fs	2^{-48} sec

USER TIME STAMPERS (UTS) SPECIFICATIONS

Table 27.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
LATENCY					
Fractional Seconds Format		4	6	µs	
PTP Format		5	7	µs	
THROUGHPUT RATE					The sample rate is shared among all UTSs that are connected to a time stamp source; data capture involves asserting an IO update and reading the UTS first in, first out (FIFO) registers as a contiguous 14-byte block; the indicated rate is the maximum continuous UTS traffic possible without overloading the UTS FIFO
SCLK = 50 MHz with Address Looping			446	kSPS	Dedicated, 100% serial port interface (SPI) utilization using address looping (loop length = 14 decimal in Register 0x0010), which eliminates the overhead of asserting an IO update prior to reading the UTS FIFO registers
SCLK = 50 MHz			324	kSPS	Dedicated, 100% SPI utilization
SCLK = 25 MHz			16	kSPS	10% SPI utilization
TIME OFFSET					
Range	−29.8		+29.8	ns	Signed 24-bit range scaled by 2^{-48} sec
Resolution		3.55		fs	2^{-48} sec
FIFO DEPTH			18	Samples	Each UTS has a one-sample buffer, which can extend the apparent FIFO depth by a few samples beyond 18 samples

INVERSE USER TIME STAMPERS (IUTS) SPECIFICATIONS

Table 28.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
FREQUENCY RANGE	0.001	2	200	kHz	IUTS frequency transport range for digitized clocking
PERIOD MONITOR THRESHOLD		±1.56		%	$\pm 2^{-6}$ relative to the average period IUTS transport frequency
THROUGHPUT RATE					Data transfer involves a 14-byte block write to the IUTS control registers followed by assertion of an IO update; the indicated rate is the maximum continuous IUTS traffic possible without causing IUTS invalidation; SPI utilization other than for IUTS operation reduces throughput
SCLK = 50 MHz			64	kSPS	20% SPI utilization allocation for IUTS operations
SCLK = 25 MHz			16	kSPS	10% SPI utilization allocation for IUTS operations
TIME OFFSET					
Range	−29.8		+29.8	ns	Signed 24-bit range scaled by 2^{-48} sec
Resolution		3.55		fs	2^{-48} sec

ANALOG LOOPBACK (ROUND TRIP DELAY) SPECIFICATIONS

The data in Table 29 reflects REFB configured for dc-coupled 1.2 V CMOS. The data for Path A captures the delay associated with the path from the REFB receiver through the loopback selection mux to the M4 pin driver and the effect of operating the M4 driver with nominal supply voltages of 1.8 V, 2.5 V, and 3.3 V. The data for Path B captures the delay associated with the reference divider. The data for Path C captures the delay associated with the reference demodulator. See Figure 56 for a depiction of the three paths (A, B, and C).

In the upper propagation delay section of Table 29, the tabulated values include variation of supply voltage from nominal and temperature variation.

In the lower propagation delay section of Table 29, the tabulated values include only variation of supply voltage from nominal. This data is useful when the user has the means to correct for delay variation due to temperature by using the tabulated temperature coefficients to apply an offset to round trip delay measurements.

The M4 output section of Table 29 specifies the output drive capability of the M4 pin driver when operating in analog loopback mode.

Table 29.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
PROPAGATION DELAY					See Figure 56 for path details
REFB TO M4 Accounting for Supply Variation from Nominal and Temperature Variation					
Path A					
VDDIOB = 1.8 V	2.09	2.54	3.00	ns	
VDDIOB = 2.5 V	1.79	2.09	2.45	ns	
VDDIOB = 3.3 V	1.63	1.91	2.22	ns	
Path B					VDDIOB = 1.8 V
R Divider > 1	2.95	3.57	4.20	ns	
R Divider = 1	2.76	3.33	3.90	ns	
Path C	4.14	5.11	6.06	ns	VDDIOB = 1.8 V
REFB to M4 Accounting for Supply Variation from Nominal					See Figure 56 for path details
Path A					
VDDIOB = 1.8 V	2.41	2.54	2.66	ns	
Temperature Coefficient		4.12		ps/°C	
VDDIOB = 2.5 V	1.98	2.09	2.20	ns	
Temperature Coefficient		2.92		ps/°C	
VDDIOB = 3.3 V	1.81	1.91	2.02	ns	
Temperature Coefficient		2.33		ps/°C	
Path B					VDDIOB = 1.8 V
R Divider > 1	3.32	3.57	3.80	ns	
Temperature Coefficient		4.82		ps/°C	
R Divider = 1	3.12	3.33	3.52	ns	
Temperature Coefficient		4.73		ps/°C	
Path C	4.75	5.11	5.45	ns	VDDIOB = 1.8 V
Temperature Coefficient		7.60		ps/°C	
M4 OUTPUT SPECIFICATION					Applies only for analog loopback operation; 5 pF capacitive load
Frequency range	0		125	MHz	Maximum limit applies for full output voltage swing with VDDIOB = 1.8 V, 2.5 V, or 3.3 V; output voltage swing degrades for frequencies above 125 MHz
Pulse Width High	3			ns	Minimum duration of positive half cycle that supports full output voltage swing; positive output voltage swing degrades for frequencies above 125 MHz or for pulse width < 3 ns
Pulse Width Low	3			ns	Minimum duration of negative half cycle that supports full output voltage swing; negative output voltage swing degrades for frequencies above 125 MHz or for pulse width < 3 ns

Mx TO Mx PIN OUTPUT TIMING SKEW

The data in Table 30 indicates the time offset of the target Mx pin clock edge relative to the M0 pin clock edge. Table 30 indicates the difference in output timing between Mx pins that are configured as outputs. The M0 pin and target Mx pins use the same internal status source. To determine the skew between any two Mx pins (for example, M2 to SDO/M5), subtract the values in the appropriate rows (skew from M2 to SDO/M5 is 190 − 760 = −570 ps). Note that the supply voltage on VDDIOA and VDDIOB has minimal effect (when the nominal supply voltage of each is similar) on the output timing skew.

Table 30.

Parameter	Min	Typ	Max	Unit
TIMING SKEW				
M0 to M1		160		ps
M0 to M2		190		ps
M0 to M3		630		ps
M0 to M4		1250		ps
M0 to SDO/M5		760		ps
M0 to CSB/M6		150		ps

SYSTEM CLOCK COMPENSATION SPECIFICATIONS

Table 31.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
OPEN-LOOP COMPENSATION					See the Compensation Method 1 section
Resolution		0.028		ppt	ppt is parts per trillion (10^{-12})
CLOSED-LOOP COMPENSATION (AUXILIARY DPLL)					See the Compensation Method 3 section
Phase Detector Frequency	2		200	kHz	
Loop Bandwidth	0.1		2×10^3	Hz	
Reference Monitor Threshold		5		%	

TEMPERATURE SENSOR SPECIFICATIONS

Table 32.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
TEMPERATURE					
Accuracy					T_A = −50°C to +110°C
Absolute Die Temperature		5		°C	
Relative Die Temperature		1.7		%	
Resolution		0.0078		°C	16-bit (signed) resolution
Conversion time		0.18		ms	
REPEATABILITY		±0.02		°C	T_A = 25°C
DRIFT		0.1		°C	500-hour stress test at 100°C

LOGIC INPUT SPECIFICATIONS (RESETB, M0 TO CSB/M6 PINS)

Table 33.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
RESETB					Valid for 3.3 V ≥ VDDIOA ≥ 1.8 V; internal 100 kΩ pull-up resistor
Input Voltage					
High, V_{IH}	VDDIOA − 0.4			V	
Low, V_{IL}			0.4	V	
Input Current					
High, I_{INH}		1		µA	
Low, I_{INL}		±15	±125	µA	

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
LOGIC INPUTS (M0 to CSB/M6)					Valid for 3.3 V ≥ VDDIOx ≥ 1.8 V; VDDIOA applies to the SDO/M5 pin and the SDA/M6 pin; VDDIOB applies to the M0, M1, M2, M3, and M4 pins; the M3 and M4 pins have internal 100 kΩ pull-down resistors
Input Voltage					
High, V_{IH}	VDDIOx – 0.4			V	
Low, V_{IL}			0.4	V	
Input Current, I_{INH}, I_{INL}		±15	±125	µA	

LOGIC OUTPUT SPECIFICATIONS (M0 TO CSB/M6 PINS)

Table 34.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
LOGIC OUTPUTS (M0 to CSB/M6)					Valid for 3.3 V ≥ VDDIOx ≥ 1.8 V; VDDIOA applies for the SDO/M5 and CSB/M6 pins; VDDIOB applies for the M0 to M4 pins; normal (default) output drive current setting for the M0 to CSB/M6 pins
Frequency Range			26	MHz	M4 capable of 125 MHz when configured for analog loopback (see Table 29)
Output Voltage					
High, V_{OH}	VDDIOx – 0.6			V	Load current = 10 mA
	VDDIOx – 0.2			V	Load current = 1 mA
Low, V_{OL}			0.6	V	Load current = 10 mA
			0.2	V	Load current = 1 mA
Rise/Fall Time					
Normal Drive Strength (Default)					
5 pF Capacitive Load					
VDDIOx = 1.8 V		880		ps	
VDDIOx = 2.5 V		740		ps	
VDDIOx = 3.3 V		580		ps	
10 pF Capacitive Load					
VDDIOx = 1.8 V		1210		ps	
VDDIOx = 2.5 V		1020		ps	
VDDIOx = 3.3 V		900		ps	
20 pF Capacitive Load					
VDDIOx = 1.8 V		1900		ps	
VDDIOx = 2.5 V		1230		ps	
VDDIOx = 3.3 V		970		ps	
Weak Drive Strength					
5 pF Capacitive Load					
VDDIOx = 1.8 V		1900		ps	
VDDIOx = 2.5 V		1220		ps	
VDDIOx = 3.3 V		970		ps	
10 pF Capacitive Load					
VDDIOx = 1.8 V		2730		ps	
VDDIOx = 2.5 V		1810		ps	
VDDIOx = 3.3 V		1420		ps	
20 pF Capacitive Load					
VDDIOx = 1.8 V		4040		ps	
VDDIOx = 2.5 V		2890		ps	
VDDIOx = 3.3 V		2340		ps	

SERIAL PORT SPECIFICATIONS

SPI Mode

Table 35.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
CSB					Valid for VDDIOA = 1.8 V, 2.5 V, and 3.3 V
Input Voltage					
Logic 1	VDDIOA – 0.4			V	
Logic 0			0.4	V	
Input Current					
Logic 1		1		µA	
Logic 0		1		µA	
SCLK					
Input Voltage					
Logic 1	VDDIOA – 0.4			V	
Logic 0			0.4	V	
Input Current					
Logic 1		1		µA	
Logic 0		1		µA	
SDIO					
As an Input					
Input Voltage					
Logic 1	VDDIOA – 0.4			V	
Logic 0			0.4	V	
Input Current					
Logic 1		1		µA	
Logic 0		1		µA	
As an Output					
Output Voltage					1 mA load current
Logic 1	VDDIOA – 0.2			V	
Logic 0			0.2	V	
SDO					
Output Voltage					1 mA load current
Logic 1	VDDIOA – 0.2			V	
Logic 0			0.2	V	
Leakage Current			±1	µA	SDO inactive (high impedance)
TIMING					Valid for VDDIOA = 1.8 V, 2.5 V, and 3.3 V
SCLK					
Clock Rate, $1/t_{CLK}$			50	MHz	
Pulse Width High, t_{HIGH}	5			ns	
Pulse Width Low, t_{LOW}	9			ns	
SDIO to SCLK Setup, t_{DS}	2.2			ns	
SCLK to SDIO Hold, t_{DH}	0			ns	
SCLK to Valid SDIO and SDO, t_{DV}			9	ns	
CSB to SCLK Setup, t_S	1.5			ns	
CSB to SCLK Hold, t_C	0			ns	
CSB Minimum Pulse Width High	1			t_{CLK}	

I²C Mode

Table 36.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
SDA, SCL (AS INPUTS)					Valid for VDDIOA = 1.8 V, 2.5 V, and 3.3 V
Input Voltage					
Logic 1	70			% of VDDIOA	
Logic 0			0.3 × VDDIOA	V	
Input Current	−10		+10	µA	For V_{IN} = 10% to 90% of VDDIOA
Hysteresis of Schmitt Trigger Inputs	1.5			% of VDDIOA	
SDA (AS OUTPUT)					
Output Logic 0 Voltage			0.2	V	Output current (I_{OUT}) = 3 mA
Output Fall Time from V_{IH} Minimum to V_{IL} Maximum	20 + 0.1 × C_B		250	ns	10 pF ≤ C_B ≤ 400 pF; C_B is the bus capacitance
TIMING					
SCL Clock Rate			400	kHz	
Bus Free Time Between a Stop and Start Condition, t_{BUF}	1.3			µs	
Repeated Start Condition Setup Time, $t_{SU;STA}$	0.6			µs	
Repeated Hold Time Start Condition, $t_{HD;STA}$	0.6			µs	After this period, the first clock pulse is generated
Stop Condition Setup Time, $t_{SU;STO}$	0.6			µs	
Low Period of the SCL Clock, t_{LOW}	1.3			µs	
High Period of the SCL Clock, t_{HIGH}	0.6			µs	
SCL/SDA Rise Time, t_R	20 + 0.1 × C_B		300	ns	
SCL/SDA Fall Time, t_F	20 + 0.1 × C_B		300	ns	
Data Setup Time, $t_{SU;DAT}$	100			ns	
Data Hold Time, $t_{HD;DAT}$	100			ns	
Capacitive Load, C_B			400	pF	Applies to individual bus lines

JITTER GENERATION (RANDOM JITTER)

System clock frequency doubler enabled. DPLL configured with the high phase margin loop filter. The integration bandwidth for rms jitter values is 12 kHz to 20 MHz. The effect on jitter values due to using different output driver modes is insignificant.

Device Configuration 1:
f_{OSC} = 52 MHz crystal, f_{REF} = 38.88 MHz, f_{OUT} = 155.52 MHz, DPLL bandwidth (BW_{DPLL}) = 50 Hz, phase buildout operation;
Channel 0: f_{VCO} = 2488.32 MHz; Channel 1: VCO frequency (f_{VCO}) = 3265.92 MHz, Q divider half divide enabled

Device Configuration 2:
f_{OSC} = 52 MHz crystal, f_{REF} = 30.72 MHz, f_{OUT} = 245.76 MHz, BW_{DPLL} = 50 Hz, internal zero delay operation;
Channel 0: f_{VCO} = 2457.6 MHz; Channel 1: f_{VCO} = 3686.4 MHz, Q divider half divide enabled

Device Configuration 3:
f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz temperature compensated crystal oscillator (TCXO), loop bandwidth of the auxiliary DPLL servo loop (BW_{COMP}) = 50 Hz (see the Auxiliary DPLL Loop Bandwidth section for more details), f_{REF} = 1 Hz, f_{OUT} = 491.52 MHz, BW_{DPLL} = 0.05 Hz, phase buildout operation; Channel 0: f_{VCO} = 2949.12 MHz; Channel 1: f_{VCO} = 3932.16 MHz

Device Configuration 4:
f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 125 MHz, f_{OUT} = 125 MHz, BW_{DPLL} = 0.1 Hz, phase buildout operation; Channel 0: f_{VCO} = 2500 MHz; Channel 1: f_{VCO} = 3250 MHz

Device Configuration 5:
f_{OSC} = 52 MHz crystal, f_{REF} = 25 MHz, f_{OUT} = 312.5 MHz, BW_{DPLL} = 50 Hz , phase buildout operation; Channel 0: f_{VCO} = 2500 MHz; Channel 1: f_{VCO} = 3750 MHz

Device Configuration 6:
f_{OSC} = 52 MHz crystal, f_{REF} = 155.52 MHz, MHz, f_{OUT} = (155.52 × 255/227) MHz, BW_{DPLL} = 50 Hz; Channel 0: f_{VCO} = 2620.5463 MHz; Channel 1: f_{VCO} = 3319.3586 MHz

Table 37.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
JITTER GENERATION					
Channel 0—DPLL0, APLL0					Channel 1 powered down
f_{OUT} = 155.52 MHz		223		fs	Device Configuration 1
f_{OUT} = 245.76 MHz		220		fs	Device Configuration 2
f_{OUT} = 491.52 MHz		235		fs	Device Configuration 3
f_{OUT} = 125.0 MHz		213		fs	Device Configuration 4
f_{OUT} = 312.5 MHz		217		fs	Device Configuration 5
f_{OUT} = 174.7030837 MHz		230		fs	Device Configuration 6
Channel 1—DPLL1, APLL1					Channel 0 powered down
f_{OUT} = 155.52 MHz		247		fs	Device Configuration 1
f_{OUT} = 245.76 MHz		280		fs	Device Configuration 2
f_{OUT} = 491.52 MHz		323		fs	Device Configuration 3
f_{OUT} = 125.0 MHz		243		fs	Device Configuration 4
f_{OUT} = 312.5 MHz		266		fs	Device Configuration 5
f_{OUT} = 174.7030837 MHz		264		fs	Device Configuration 6

PHASE NOISE

System clock frequency doubler enabled. DPLL configured with the high phase margin loop filter.

Device Configuration 1:
f_{OSC} = 52 MHz crystal, f_{REF} = 38.88 MHz, f_{OUT} = 155.52 MHz, BW_{DPLL} = 50 Hz, phase buildout operation; Channel 0: f_{VCO} = 2488.32 MHz; Channel 1: VCO frequency (f_{VCO}) = 3265.92 MHz, Q divider half divide enabled

Device Configuration 2:
f_{OSC} = 52 MHz crystal, f_{REF} = 30.72 MHz, f_{OUT} = 245.76 MHz, BW_{DPLL} = 50 Hz, internal zero delay operation; Channel 0: f_{VCO} = 2457.6 MHz; Channel 1: f_{VCO} = 3686.4 MHz, Q divider half divide enabled

Device Configuration 3:
f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz (see the Auxiliary DPLL Loop Bandwidth section for more details), f_{REF} = 1 Hz, f_{OUT} = 491.52 MHz, BW_{DPLL} = 0.05 Hz, phase buildout operation; Channel 0: f_{VCO} = 2949.12 MHz; Channel 1: f_{VCO} = 3932.16 MHz

Device Configuration 4:
f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 125 MHz, f_{OUT} = 125 MHz, BW_{DPLL} = 0.1 Hz, phase buildout operation; Channel 0: f_{VCO} = 2500 MHz; Channel 1: f_{VCO} = 3250 MHz

Device Configuration 5:
f_{OSC} = 52 MHz crystal, f_{REF} = 25 MHz, f_{OUT} = 312.5 MHz, BW_{DPLL} = 50 Hz , phase buildout operation; Channel 0: f_{VCO} = 2500 MHz; Channel 1: f_{VCO} = 3750 MHz

Device Configuration 6:
f_{OSC} = 52 MHz crystal, f_{REF} = 155.52 MHz, MHz, f_{OUT} = (155.52 × 255/227) MHz, BW_{DPLL} = 50 Hz; Channel 0: f_{VCO} = 2620.5463 MHz; Channel 1: f_{VCO} = 3319.3586 MHz

Table 38.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
PHASE NOISE					
Channel 0—DPLL0, APLL0					Channel 1 powered down
f_{OUT} = 155.52 MHz					Device Configuration 1
10 Hz Offset		−81		dBc/Hz	
100 Hz Offset		−98		dBc/Hz	
1 kHz Offset		−118		dBc/Hz	
10 kHz Offset		−128		dBc/Hz	
100 kHz Offset		−134		dBc/Hz	
1 MHz Offset		−144		dBc/Hz	
10 MHz Offset		−158		dBc/Hz	
Floor		−161		dBc/Hz	
f_{OUT} = 245.76 MHz					Device Configuration 2
10 Hz Offset		−77		dBc/Hz	
100 Hz Offset		−93		dBc/Hz	
1 kHz Offset		−114		dBc/Hz	
10 kHz Offset		−125		dBc/Hz	
100 kHz Offset		−130		dBc/Hz	
1 MHz Offset		−140		dBc/Hz	
10 MHz Offset		−156		dBc/Hz	
Floor		−161		dBc/Hz	
f_{OUT} = 491.52 MHz					Device Configuration 3
10 Hz Offset		−74		dBc/Hz	
100 Hz Offset		−89		dBc/Hz	
1 kHz Offset		−108		dBc/Hz	
10 kHz Offset		−119		dBc/Hz	
100 kHz Offset		−123		dBc/Hz	
1 MHz Offset		−134		dBc/Hz	
10 MHz offset		−152		dBc/Hz	
Floor		−159			

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
f_{OUT} = 125 MHz					Device Configuration 4
10 Hz Offset		−84		dBc/Hz	
100 Hz Offset		−106		dBc/Hz	
1 kHz Offset		−120		dBc/Hz	
10 kHz Offset		−131		dBc/Hz	
100 kHz Offset		−136		dBc/Hz	
1 MHz Offset		−147		dBc/Hz	
10 MHz Offset		−160		dBc/Hz	
Floor		−163		dBc/Hz	
f_{OUT} = 312.5 MHz					Device Configuration 5
10 Hz Offset		−74		dBc/Hz	
100 Hz Offset		−91		dBc/Hz	
1 kHz Offset		−112		dBc/Hz	
10 kHz Offset		−123		dBc/Hz	
100 kHz Offset		−128		dBc/Hz	
1 MHz Offset		−138		dBc/Hz	
10 MHz Offset		−154		dBc/Hz	
Floor		−161		dBc/Hz	
f_{OUT} = 174.7030837 MHz					Device Configuration 6
10 Hz Offset		−82		dBc/Hz	
100 Hz Offset		−99		dBc/Hz	
1 kHz Offset		−117		dBc/Hz	
10 kHz Offset		−127		dBc/Hz	
100 kHz Offset		−133		dBc/Hz	
1 MHz Offset		−143		dBc/Hz	
10 MHz Offset		−157		dBc/Hz	
Floor		−160		dBc/Hz	
Channel 1—DPLL1, APLL1					Channel 0 powered down
f_{OUT} = 155.52 MHz					Device Configuration 1
10 Hz Offset		−81		dBc/Hz	
100 Hz Offset		−98		dBc/Hz	
1 kHz Offset		−118		dBc/Hz	
10 kHz Offset		−128		dBc/Hz	
100 kHz Offset		−132		dBc/Hz	
1 MHz Offset		−144		dBc/Hz	
10 MHz Offset		−158		dBc/Hz	
Floor		−162		dBc/Hz	
f_{OUT} = 245.76 MHz					Device Configuration 2
10 Hz Offset		−76		dBc/Hz	
100 Hz Offset		−93		dBc/Hz	
1 kHz Offset		−114		dBc/Hz	
10 kHz Offset		−124		dBc/Hz	
100 kHz Offset		−127		dBc/Hz	
1 MHz Offset		−138		dBc/Hz	
10 MHz Offset		−156		dBc/Hz	
Floor		−161		dBc/Hz	

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
f_{OUT} = 491.52 MHz					Device Configuration 3
10 Hz Offset		−74		dBc/Hz	
100 Hz Offset		−90		dBc/Hz	
1 kHz Offset		−108		dBc/Hz	
10 kHz Offset		−118		dBc/Hz	
100 kHz Offset		−120		dBc/Hz	
1 MHz Offset		−131		dBc/Hz	
10 MHz Offset		−150		dBc/Hz	
Floor		−160		dBc/Hz	
f_{OUT} = 125 MHz					Device Configuration 4
10 Hz Offset		−83		dBc/Hz	
100 Hz Offset		−106		dBc/Hz	
1 kHz Offset		−120		dBc/Hz	
10 kHz Offset		−131		dBc/Hz	
100 kHz Offset		−135		dBc/Hz	
1 MHz Offset		−145		dBc/Hz	
10 MHz Offset		−160		dBc/Hz	
Floor		−163		dBc/Hz	
f_{OUT} = 312.5 MHz					Device configuration 5
10 Hz Offset		−73		dBc/Hz	
100 Hz Offset		−91		dBc/Hz	
1 kHz Offset		−112		dBc/Hz	
10 kHz Offset		−122		dBc/Hz	
100 kHz Offset		−125		dBc/Hz	
1 MHz Offset		−137		dBc/Hz	
10 MHz Offset		−154		dBc/Hz	
Floor		−161		dBc/Hz	
f_{OUT} = 174.7030837 MHz					Device Configuration 6
10 Hz Offset		−77		dBc/Hz	
100 Hz Offset		−99		dBc/Hz	
1 kHz Offset		−117		dBc/Hz	
10 kHz Offset		−127		dBc/Hz	
100 kHz Offset		−131		dBc/Hz	
1 MHz Offset		−142		dBc/Hz	
10 MHz Offset		−158		dBc/Hz	
Floor		−161		dBc/Hz	

ABSOLUTE MAXIMUM RATINGS

Table 39.

Parameter	Rating
1.8 V Supply Voltage (VDD)	2 V
Output Drivers (OUT0AP, OUT0AN, OUT0BP, OUT0BN, OUT0CP, OUT0CN, OUT1AP, OUT1AN, OUT1BP, and OUT1BN Pins)	2 V
Input/Output Supply Voltage (VDDIOA and VDDIOB)	3.6 V
Input Voltage Range (XOA, XOB, REFA, REFAA, REFB, and REFBB Pins)	−0.5 V to VDD + 0.5 V
Digital Input Voltage Range	
SDO/M5, SCLK/SCL, SDIO/SDA, and CSB/M6 Pins	−0.5 V to VDDIOA + 0.5 V
M0, M1, M2, M3, and M4 Pins	−0.5 V to VDDIOB + 0.5 V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering 10 sec)	300°C

Stresses at or above those listed under Absolute Maximum Ratings may cause permanent damage to the product. This is a stress rating only; functional operation of the product at these or any other conditions above those indicated in the operational section of this specification is not implied. Operation beyond the maximum operating conditions for extended periods may affect product reliability.

THERMAL RESISTANCE

Thermal performance is directly linked to printed circuit board (PCB) design and operating environment. Careful attention to PCB thermal design is required.

θ_{JA} is the junction to ambient thermal resistance, 0.0 m/sec airflow per JEDEC JESD51-2 (still air).

θ_{JMA} is the junction to ambient thermal resistance, 1.0 m/sec airflow or 2.5 m/sec airflow per JEDEC JESD51-6 (moving air).

θ_{JC} is the junction to case thermal resistance (die to heat sink) per MIL-STD 883, Method 1012.1.

Values of θ_{JA} are for package comparison and PCB design considerations. θ_{JA} provides for a first-order approximation of T_J per the following equation:

$$T_J = T_A + (\theta_{JA} \times PD)$$

where:
T_A is the ambient temperature (°C).
PD is the power dissipation in watts.

Values of θ_{JC} are for package comparison and PCB design considerations when an external heat sink is required.

Table 40. Thermal Resistance

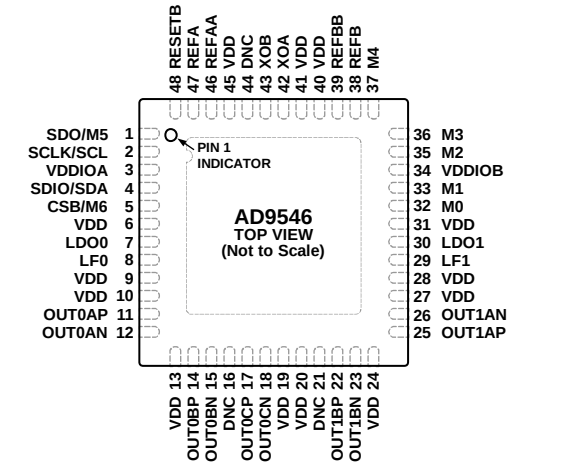
Package Type	θ_{JA}	θ_{JMA}[1]	θ_{JC}	Unit
CP-48-13[2, 3]	23.1	18.4, 17.1	0.7	°C/W

[1] θ_{JMA} is 18.4°C/W at 1.0 m/sec airflow and 17.1°C/W at 2.5 m/sec airflow.
[2] Thermal characteristics derived using a JEDEC51-7 plus JEDEC51-5 2S2P test board. The exposed pad on the bottom of the package must be soldered to ground to achieve the specified thermal performance.
[3] Results are from simulations. The PCB is a JEDEC multilayer type. Thermal performance for actual applications requires careful inspection of the conditions in the application to determine if they are similar to those assumed in these calculations.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS



NOTES
1. EXPOSED PAD. THE EXPOSED PAD IS THE GROUND CONNECTION ON THE CHIP. THE EXPOSED PAD MUST BE SOLDERED TO THE ANALOG GROUND OF THE PCB TO ENSURE PROPER FUNCTIONALITY AND FOR HEAT DISSIPATION, NOISE, AND MECHANICAL STRENGTH BENEFITS.
2. DNC = DO NOT CONNECT. DO NOT CONNECT TO THIS PIN.

23266-002

Figure 2. Pin Configuration

Table 41. Pin Function Descriptions

Pin No.	Mnemonic	Input/ Output	Pin Type	Description
1	SDO/M5	Output	CMOS	Serial Data Output (SDO). This pin is for reading serial data in 4-wire SPI mode. Changes to the VDDIOA supply voltage affect the V_{IH} and V_{OH} values for this pin. Configurable Input/Output (M5). This pin is a status and control pin when the device is not in 4-wire SPI mode.
2	SCLK/SCL	Input	CMOS	Serial Programming Clock Pin in SPI Mode (SCLK). Changes to the VDDIOA supply voltage affect the V_{IH} and V_{OH} values for this pin. Serial Clock Pin in I^2C Mode (SCL). Changes to the VDDIOA supply voltage affect the V_{IH} and V_{OH} values for this pin.
3	VDDIOA	Input	Power	Serial Port Power Supply. The valid supply voltage is 1.8 V, 2.5 V, or 3.3 V. The VDDIOA pin can be connected to the VDD supply bus if 1.8 V operation is desired.
4	SDIO/SDA	Input/ output	CMOS	Serial Data Input/Output in SPI Mode (SDIO). Write data to this pin in 4-wire SPI mode. This pin has no internal pull-up or pull-down resistor. Changes to the VDDIOA supply voltage affect the V_{IH} and V_{OH} values for this pin. Serial Data Pin in I^2C Mode (SDA).
5	CSB/M6	Input/ output	CMOS	Chip Select in SPI Mode (CSB). Active low input. Maintain a Logic 0 level on this pin when programming the device in SPI mode. This pin has an internal 10 kΩ pull-up resistor. Changes to the VDDIOA supply voltage affect the V_{IH} and V_{OH} values for this pin. Configurable Input/Output (M6). This pin is a status and control pin when the device is not in SPI mode.
6, 9, 10, 13, 19, 20, 24, 27, 28, 31, 40, 41, 45	VDD	Input	Power	1.8 V Power Supply.
7	LDO0	Input	LDO bypass	APLL0 Loop Filter Voltage Regulator. Connect a 0.22 µF capacitor from the LDO0 pin to ground. LDO0 is the ac ground reference for the integrated APLL0 loop filter.
8	LF0	Input/ output	Loop filter for APLL0	Loop Filter Node for APLL0. Connect a 3.9 nF capacitor from the LF0 pin to Pin 7 (LDO0).
11	OUT0AP	Output	HCSL, LVDS, CML	PLL0 Output 0A.

Pin No.	Mnemonic	Input/ Output	Pin Type	Description
12	OUT0AN	Output	HCSL, LVDS, CML	PLL0 Complementary Output 0A.
14	OUT0BP	Output	HCSL, LVDS, CML	PLL0 Output 0B.
15	OUT0BN	Output	HCSL, LVDS, CML	PLL0 Complementary Output 0B.
16, 21, 44	DNC	DNC	No connect	Do Not Connect. Leave the DNC pins floating.
17	OUT0CP	Output	HCSL, LVDS, CML	PLL0 Output 0C.
18	OUT0CN	Output	HCSL, LVDS, CML	PLL0 Complementary Output 0C.
22	OUT1BP	Output	HCSL, LVDS, CML	PLL1 Output 1B.
23	OUT1BN	Output	HCSL, LVDS, CML	PLL1 Complementary Output 1B.
25	OUT1AP	Output	HCSL, LVDS, CML	PLL1 Output 1A.
26	OUT1AN	Input/ Output	HCSL, LVDS, CML	PLL1 Complementary Output 1A.
29	LF1	Input/ output	Loop filter for APLL1	Loop Filter Node for APLL1. Connect a 3.9 nF capacitor from the LF1 pin to Pin 30 (LDO1).
30	LDO1	Input	LDO bypass	APLL1 Loop Filter Voltage Regulator. Connect a 0.22 µF capacitor from the LDO1 pin to ground. LDO1 is the ac ground reference for the integrated APLL1 loop filter.
32, 33, 35, 36, 37	M0, M1, M2, M3, M4	Input/ output	CMOS	Configurable Input/Output Pins. Mx are status and control pins. Changes to the VDDIOB supply voltage affect the V_{IH} and V_{OH} values for these Mx pins. M3 and M4 have internal 100 kΩ pull-down resistors. M0, M1, and M2 do not have internal resistors.
34	VDDIOB	Input	Power	Mx Pin Power Supply. The VDDIOB power supply powers the digital section that controls the M0 to M4 pins. Valid supply voltages are 1.8 V, 2.5 V, or 3.3 V. The VDDIOB pin can be connected to the VDD supply bus if 1.8 V operation is desired.
38	REFB	Input	1.8 V single-ended or differential input	Reference B Input. This internally biased REFB input is typically ac-coupled. When configured in this manner, REFB can accept any differential signal with a single-ended swing up to the VDD power supply. If dc-coupled, the REFB input can be LVDS or single-ended 1.8 V CMOS.
39	REFBB	Input	1.8 V single-ended or differential input	Reference BB Input or Complementary Reference B Input. If REFB is in differential mode, the REFB complementary signal is on the REFBB pin. No connection is necessary to the REFBB pin if REFB is a single-ended input and REFBB is not used.
42	XOA	Input	Differential input	System Clock Input. XOA contains internal dc biasing and must be ac-coupled with a 0.1 µF capacitor except when using a crystal. When a crystal is used, connect the crystal across XOA and XOB. A single-ended CMOS input is also an option, but it can produce spurious spectral content when the duty cycle is not 50%. When using XOA as a single-ended input, connect a 0.1 µF capacitor from XOB to ground.
43	XOB	Input	Differential input	Complementary System Clock Input. XOB is the complementary signal to XOA. XOB contains internal dc biasing and must be ac-coupled with a 0.1 µF capacitor except when using a crystal. When a crystal is used, connect the crystal across XOA and XOB.
46	REFAA	Input	1.8 V single-ended or differential input	Reference AA input or Complementary REFA Input. If REFA is in differential mode, the REFA complementary signal is on the REFAA pin. No connection is necessary to the REFAA pin if REFA is a single-ended input and REFAA is not used. If dc-coupled, the REFAA input is single-ended 1.8 V CMOS.
47	REFA	Input	1.8 V single-ended or differential input	Reference A Input. The internally biased REFA input is typically ac-coupled. When REFA is configured in this manner, REFA can accept any differential signal with a single-ended swing up to the VDD power supply. If dc-coupled, the input can be LVDS or single-ended 1.8 V CMOS.

Pin No.	Mnemonic	Input/ Output	Pin Type	Description
48	RESETB	Input	CMOS logic	Active Low Chip Reset. The RESETB pin has an internal 100 kΩ pull-up resistor. When asserted, the chip goes into reset. Changes to the VDDIOA supply voltage affect the V_{IH} values for RESETB.
	EPAD	Output	Exposed pad	Exposed Pad. The exposed pad is the ground connection on the chip. The exposed pad must be soldered to the analog ground of the PCB to ensure proper functionality and for heat dissipation, noise, and mechanical strength benefits.

TYPICAL PERFORMANCE CHARACTERISTICS

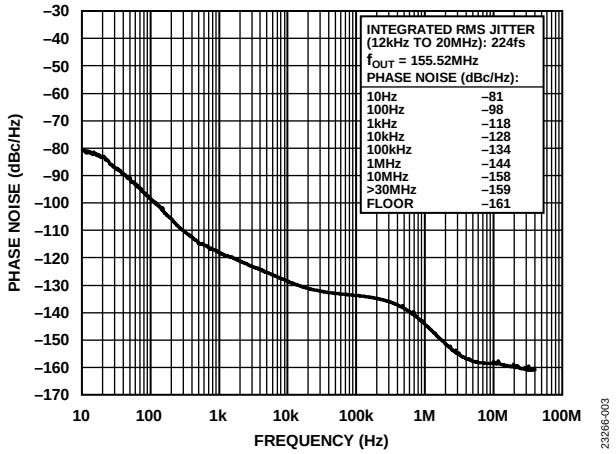


Figure 3. Absolute Phase Noise (PLL0, Device Configuration 1, HCSL Mode, f_{REF} = 38.88 MHz, f_{OUT} = 155.52 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth)

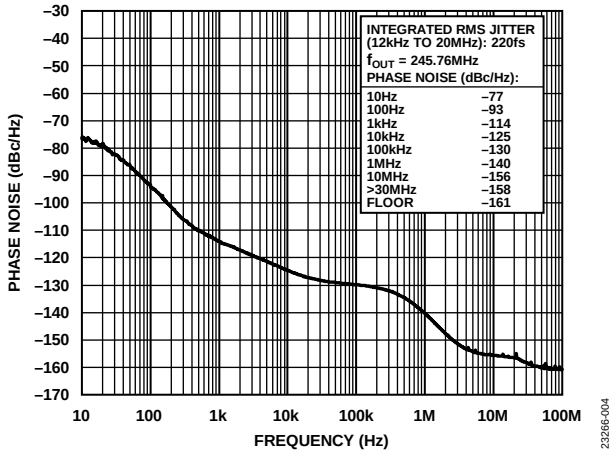


Figure 4. Absolute Phase Noise (PLL0, Device Configuration 2, HCSL Mode, f_{REF} = 30.72 MHz, f_{OUT} = 245.76 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth)

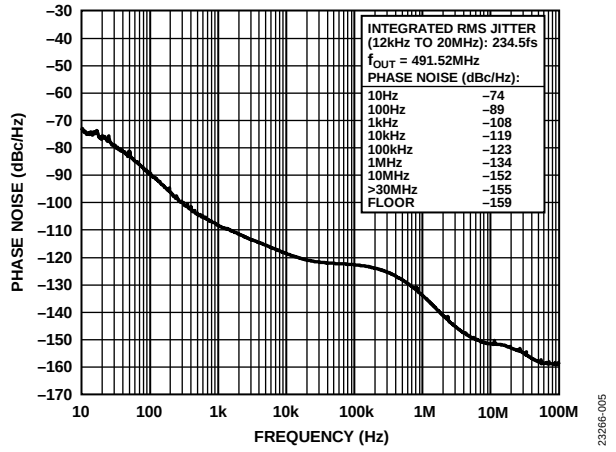


Figure 5. Absolute Phase Noise (PLL0, Device Configuration 3, HCSL Mode, f_{REF} = 1 Hz, f_{OUT} = 491.52 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 0.05 Hz DPLL Bandwidth)

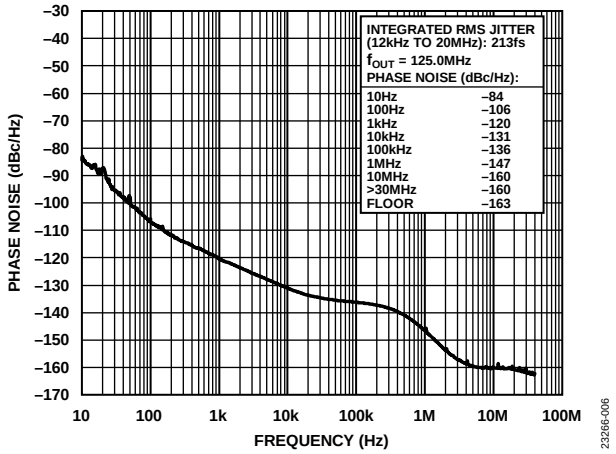


Figure 6. Absolute Phase Noise (PLL0, Device Configuration 4, HCSL Mode, f_{REF} = 125 MHz, f_{OUT} = 125.0 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 0.1 Hz DPLL Bandwidth, Phase Buildout Mode)

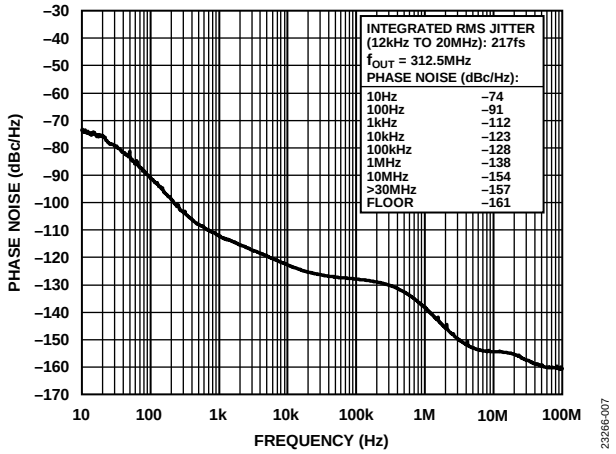


Figure 7. Absolute Phase Noise (PLL0, Device Configuration 5, HCSL Mode, f_{REF} = 25 MHz, f_{OUT} = 312.5 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth, Phase Buildout Mode)

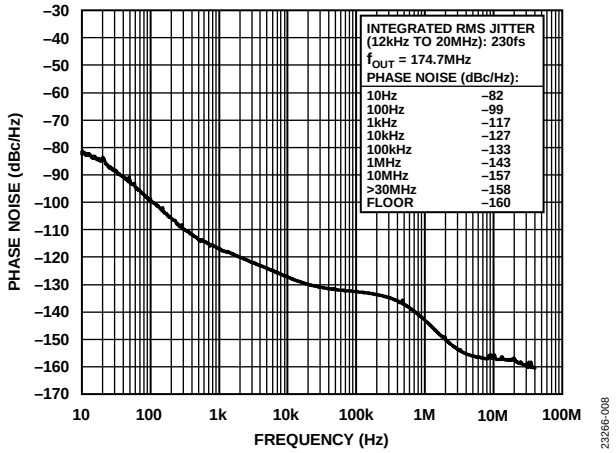


Figure 8. Absolute Phase Noise (PLL0, Device Configuration 6, HCSL Mode, f_{REF} = 155.52 MHz, f_{OUT} = 174.7 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth, Phase Buildout Mode)

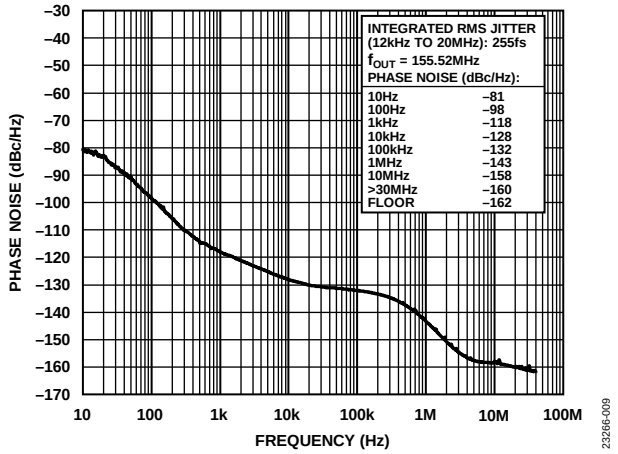


Figure 9. Absolute Phase Noise (PLL1, Device Configuration 1, HCSL Mode, f_{REF} = 38.88 MHz, f_{OUT} = 155.52 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth)

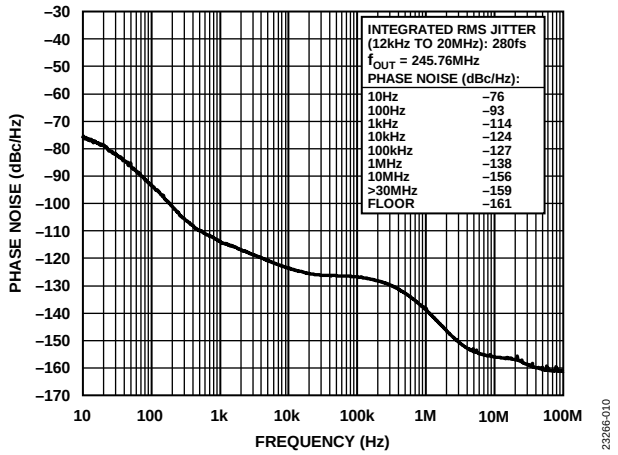


Figure 10. Absolute Phase Noise (PLL1, Device Configuration 2, HCSL Mode, f_{REF} = 30.72 MHz, f_{OUT} = 245.76 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth)

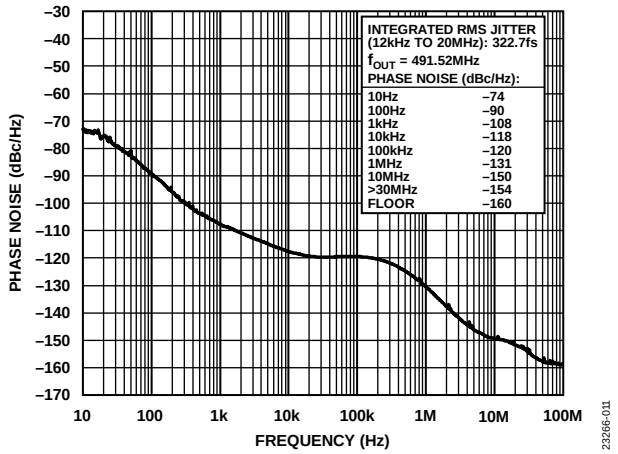


Figure 11. Absolute Phase Noise (PLL1, Device Configuration 3, HCSL Mode, f_{REF} = 1 Hz, f_{OUT} = 491.52 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 0.05 Hz DPLL Bandwidth)

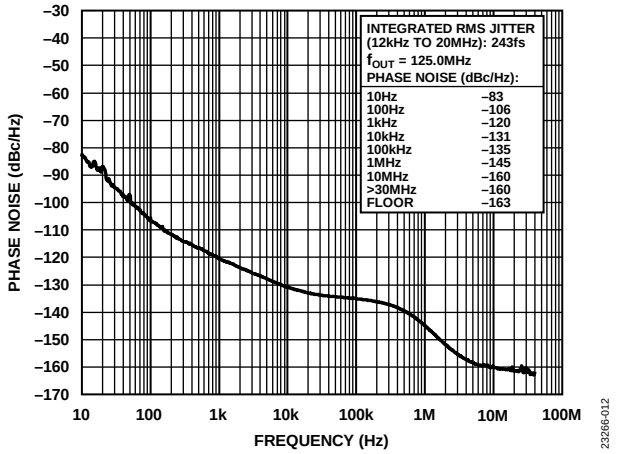


Figure 12. Absolute Phase Noise (PLL1, Device Configuration 4, HCSL Mode, f_{REF} = 125 MHz, f_{OUT} = 125 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 0.1 Hz DPLL Bandwidth, Phase Buildout Mode)

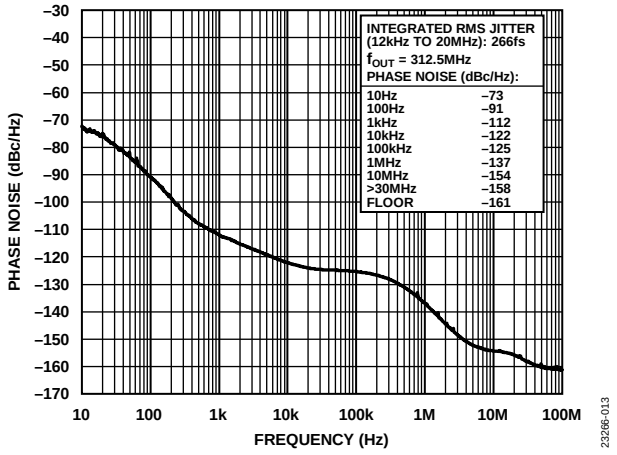


Figure 13. Absolute Phase Noise (PLL1, Device Configuration 5, HCSL Mode, f_{REF} = 25 MHz, f_{OUT} = 312.5 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth, Phase Buildout Mode)

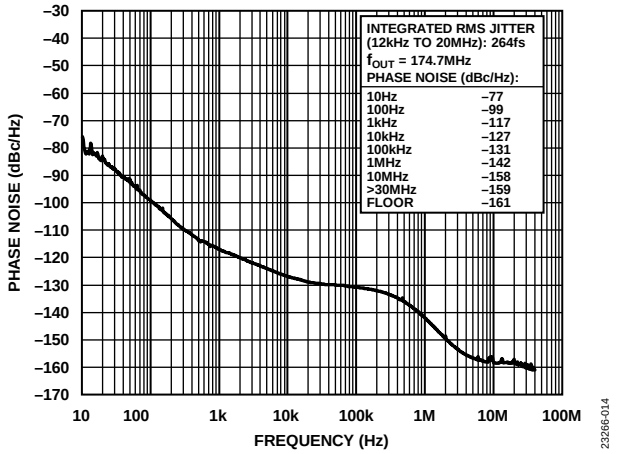


Figure 14. Absolute Phase Noise (PLL1, Device Configuration 6, HCSL Mode, f_{REF} = 155.52 MHz, f_{OUT} = 174.7 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL Bandwidth, Phase Buildout Mode)

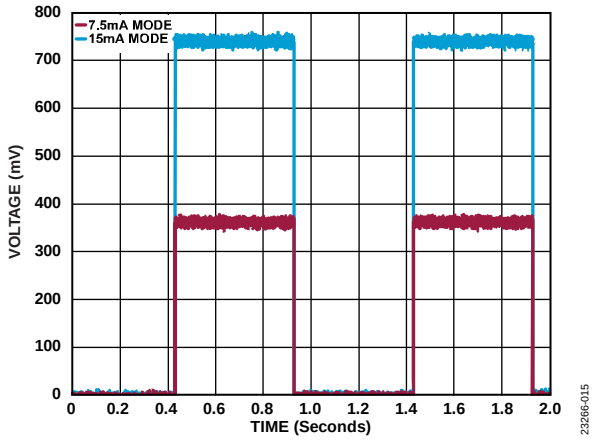


Figure 15. DC-Coupled, Single-Ended, 1 Hz Output Waveforms Using HCSL 7.5 mA and 15 mA Mode Terminated 50 Ω to Ground per Figure 46; Slew Rate ≈ 7 V/ns for 15 mA Mode; ~3.5 V/ns for 7.5 mA Mode

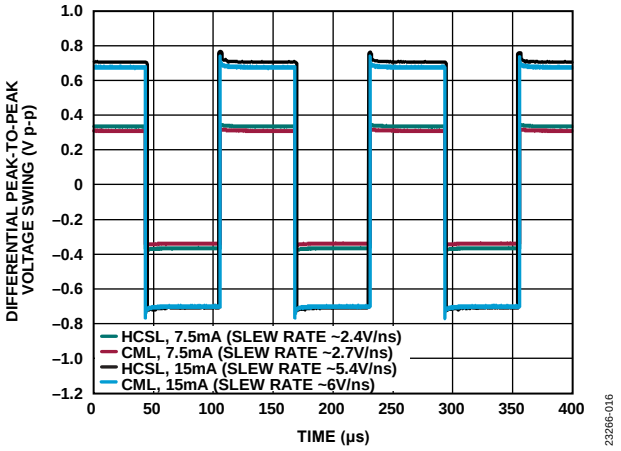


Figure 16. 8 kHz Output Waveforms for Various Driver Settings; HCSL Drivers Terminated 50 Ω to Ground per Figure 39; CML Drivers Terminated 50 Ω to 1.8 V per Figure 40

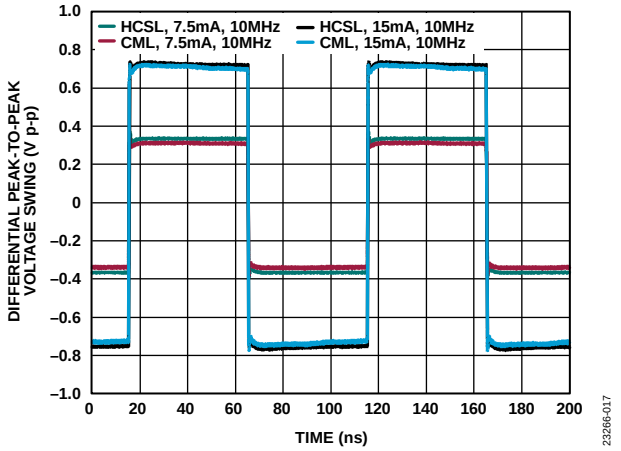


Figure 17. 10 MHz Output Waveforms for Various Driver Settings; HCSL Drivers Terminated 50 Ω to Ground per Figure 39; CML Drivers Terminated 50 Ω to 1.8 V per Figure 40

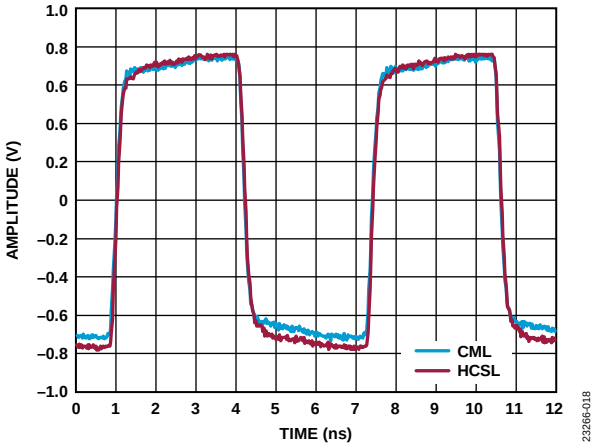


Figure 18. 156.25 MHz Output Waveform (15 mA Driver Setting); HCSL Drivers Terminated 50 Ω to Ground per Figure 39; CML Drivers Terminated 50 Ω to 1.8 V per Figure 40

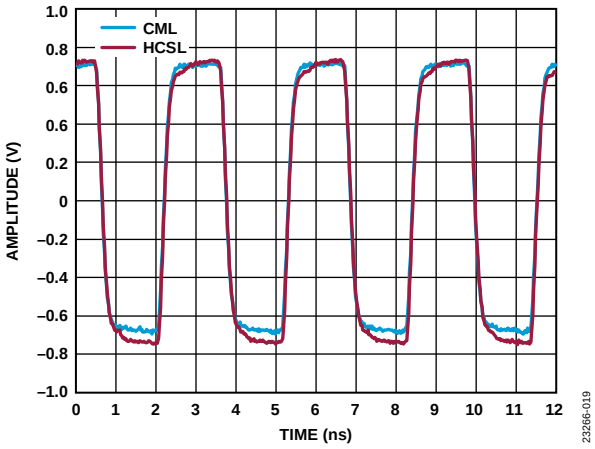


Figure 19. 312.5 × 66 MHz/64 MHz Output Waveform (15 mA Driver Setting); HCSL Drivers Terminated 50 Ω to Ground per Figure 39; CML Drivers Terminated 50 Ω to 1.8 V per Figure 40

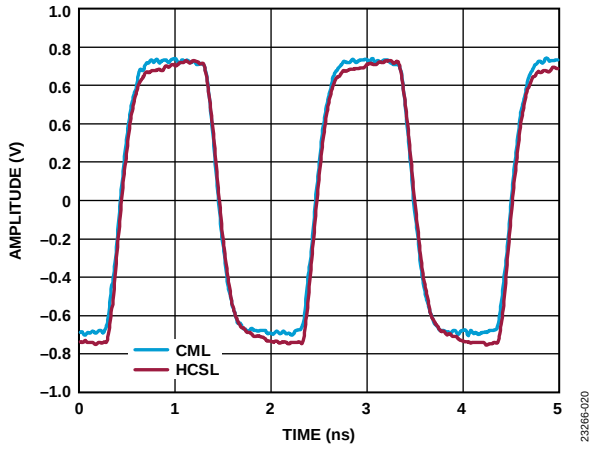


Figure 20. 491.52 MHz Output Waveform (15 mA Driver Setting); HCSL Drivers Terminated 50 Ω to Ground per Figure 39; CML Drivers Terminated 50 Ω to 1.8 V per Figure 40

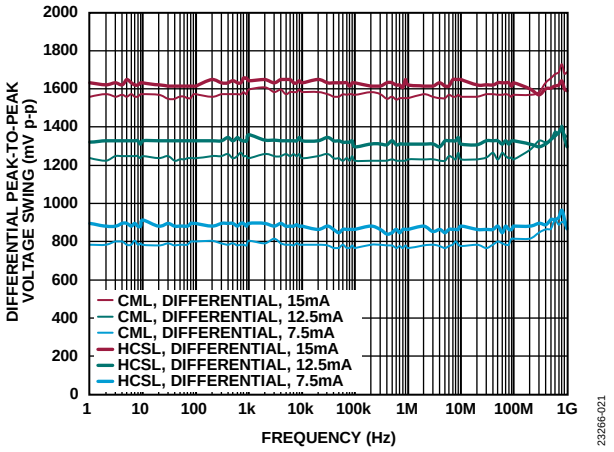


Figure 21. Differential Peak-to-Peak Voltage Swing vs. Frequency; HCSL Drivers Terminated 50 Ω to Ground per Figure 39; CML Drivers Terminated 50 Ω to 1.8 V per Figure 40

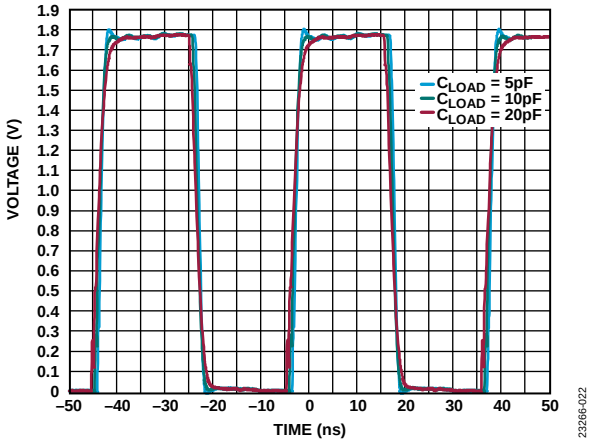


Figure 22. Mx Pin Waveforms for Various Load Conditions; Normal Output Drive Strength (Default); VDDIOx = 1.8 V

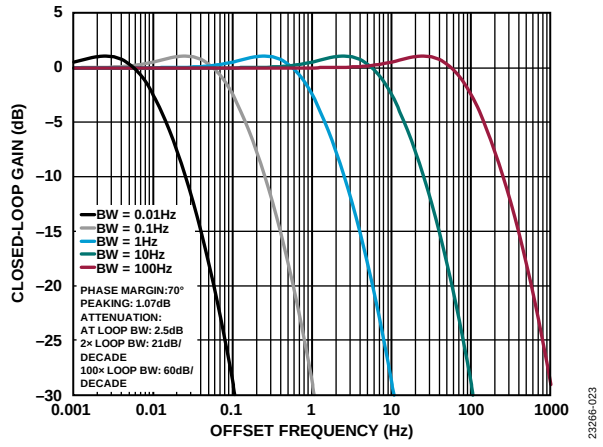


Figure 23. DPLL Closed-Loop Transfer Function Nominal Phase Margin Loop Filter Setting

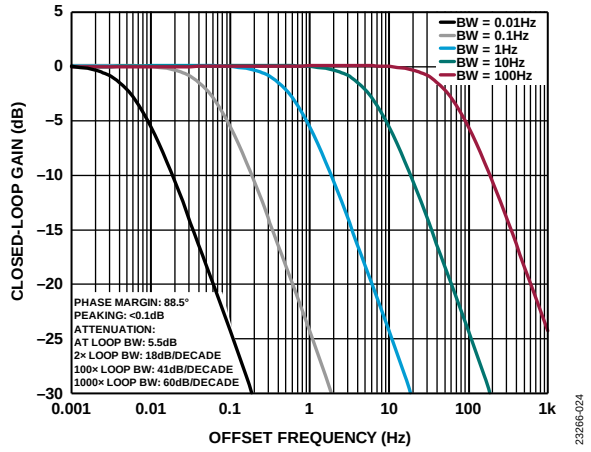


Figure 24. DPLL Closed-Loop Transfer Function High Phase Margin Loop Filter Setting

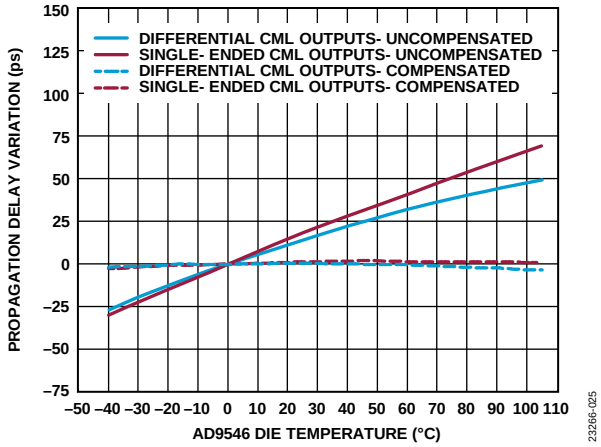


Figure 25. Propagation Delay Variation vs. AD9546 Die Temperature; CML Output Mode; 1.2 V CMOS Reference Input Mode; f_{SYS} = 2392 MHz; Internal Zero Delay (Hitless) Mode

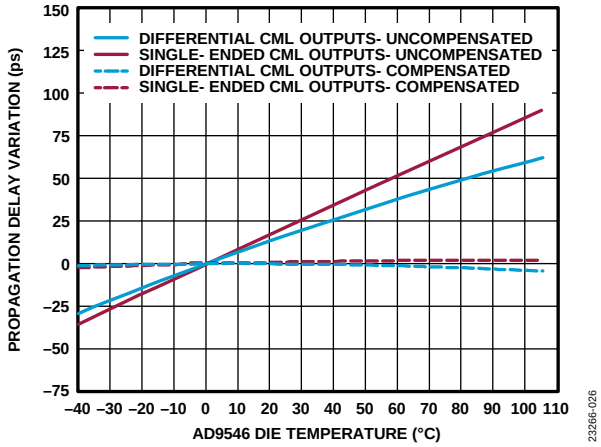


Figure 26. Propagation Delay Variation vs. AD9546 Die Temperature; HCSL Output Mode; 1.2 V CMOS Reference Input Mode; f_{SYS} = 2392 MHz; Internal Zero Delay (Hitless) Mode

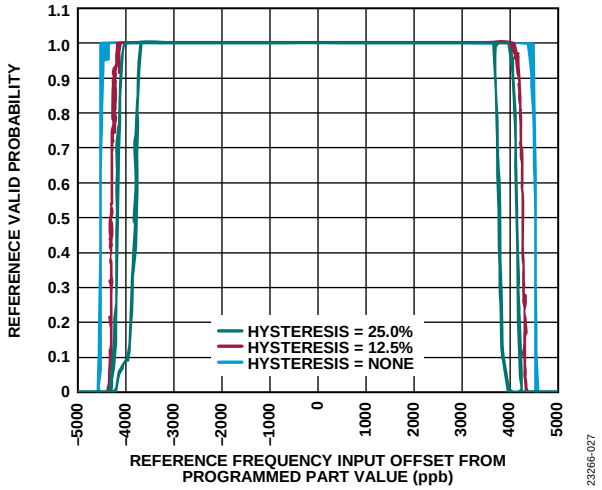


Figure 27. Reference Valid Probability vs. Reference Frequency Input Offset from Programmed Part Value; Tolerance = 4.6 ppm

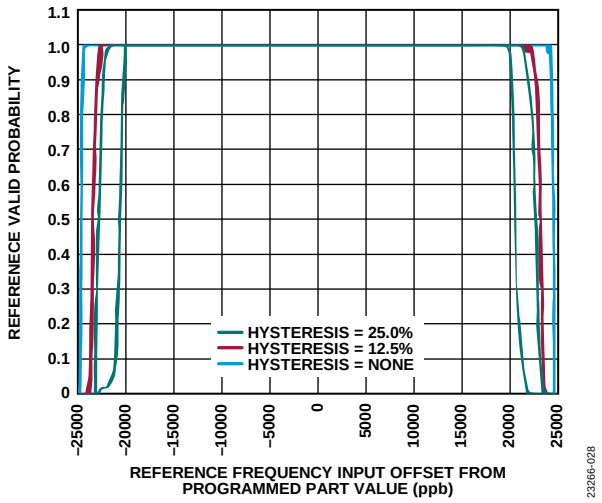


Figure 28. Reference Valid Probability vs. Reference Frequency Input Offset from Programmed Part Value; Tolerance = 25 ppm

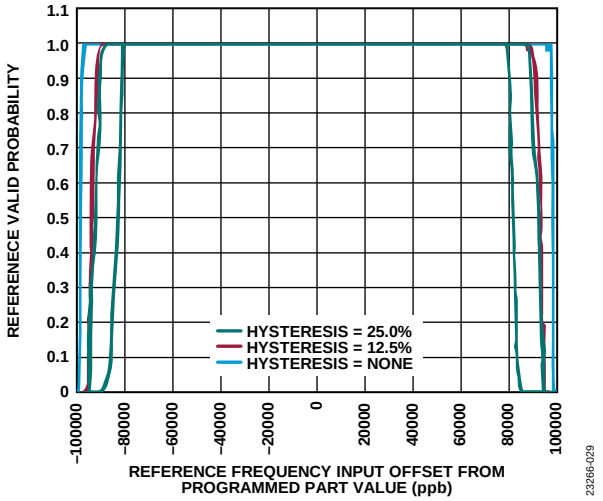


Figure 29. Reference Valid Probability vs. Reference Frequency Input Offset from Programmed Part Value; Tolerance = 100 ppm

TERMINOLOGY

Zero Delay
Zero delay is a PLL-based clock architecture that establishes zero phase offset between the output clock signal and input clock signal. A zero delay architecture requires an Integer N type PLL, by definition, because there must be a harmonic relationship between the input and output frequencies to preserve a static phase relationship between input and output edges. Note that zero delay provides a zero input to output phase offset in the static (steady state) sense only. That is, transient events may cause the PLL to slew its output phase, which temporarily breaks the zero delay condition (for example, when the PLL is in the process of phase or frequency acquisition). An example of a transient event is when a multiple input PLL switches from one input reference signal to another.

Hitless Switchover
Hitless switchover applies to PLLs that can switch between multiple input clock signals (usually operating at the same frequency). Generally, the multiple input clock signals are not phase aligned. Therefore, switching from one to another can create a substantial phase transient in the output clock signal. Hitless switchover limits the phase transient at the output by allowing the output clock signal to slew its phase in a prescribed manner to accommodate the phase of the new reference signal (typically by limiting the output frequency deviation to some prescribed maximum value). A PLL employing hitless switchover capability requires the output/input frequency ratio to be an integer greater than or equal to one. Otherwise, the output period constitutes phase in excess of 360° relative to the input period, which creates unresolvable phase ambiguity.

Note that the output phase transient limitation imposed by hitless switchover also applies any time the PLL is in the process of phase or frequency acquisition (that is, it is not necessarily limited to a reference clock switchover). A phase transient limitation exists in hitless operation because the primary enabler for hitless operation is the use of a narrow loop bandwidth, which governs the transient characteristics of the PLL in general.

Phase Buildout Switchover
Phase buildout switchover applies to PLLs that can switch between multiple input clock signals (usually operating at the same frequency). Generally, the multiple input clock signals are not phase aligned. Therefore, switching from one to another can create a substantial phase transient in the output clock signal. Phase buildout capability enables the PLL to absorb the phase difference between the original and the switched to reference clock signals, thereby preventing a phase disturbance in the output clock signal (unlike hitless switchover, which gradually slews the output phase to the new input phase). Because phase buildout operation prevents a disturbance in the phase of the output clock signal, there is no guarantee of phase alignment between the input and output clock signals. Unlike hitless switchover, phase buildout places no restriction on the output/input frequency ratio.

For more information, see the AN-1420 Application Note.

Time Base
Time base is a frequency source, the period of which denotes the passage of time.

Time Scale
Time scale is the accumulation of the periods of a time base. A time scale allows identifying a particular instant as having a particular time value. The starting point of a time scale (its epoch) is arbitrary.

Epoch
Epoch is the starting time associated with a time scale. For example, the epoch of the UTC time scale is midnight, January 1, 1970.

Time Stamp
In the context of this data sheet, time stamp is a digital (numeric) time value with an arbitrary epoch suitable for internal device usage.

Time Code
In the context of this data sheet, time code is a time stamp with a defined epoch suitable for general usage external to the device.

THEORY OF OPERATION

Figure 1 summarizes the fundamental building blocks of the AD9546. The core of the device comprises two independent PLLs, PLL0 and PLL1, as well as proprietary resources enabling the digital transport of clock signals (digitized clocking) between multiple AD9546 devices over a single digital link. Digitized clocking streamlines the design of systems that rely on the transport of time scales between spatially separated devices. The digitized clocking functionality comprises a common clock DPLL (CCDPLL), a common clock Synchronizer (CCS), nine user time stampers (UTS 0 to UTS 8), and two inverse user time stampers (IUTS 0 and IUTS 1).

The AD9546 provides a high degree of backward compatibility (legacy) with the AD9545, but with enhancements to support digitized clocking. The user activates most enhancements individually via register programming. Activating certain enhancements may affect legacy functionality and override settings in the registers associated with legacy functionality. In such cases, text describing a specific enhancement in this data sheet expressly states any impact on legacy functionality or register overrides.

The AD9546 synthesizes its system clock via a system clock PLL that upconverts the frequency applied to the system clock input pins, XOA and XOB. The system clock input pins accept frequencies from 20 MHz to 300 MHz from an external clock source. Alternatively, the user can connect a crystal resonator (25 MHz to 80 MHz) directly across the XOA and XOB pins. The output of the system clock PLL is the primary time base for time keeping functions within the AD9546.

The AD9546 supports up to eight reference input clock signals (via the REFA, REFAA, REFB, and REFBB pins and four appropriately configured Mx pins feeding auxiliary REF0, auxiliary REF1, auxiliary REF2, and auxiliary REF3). Each reference input has a dedicated reference divider as well as a reference monitor to detect when the reference signal is lost or compromised. Furthermore, when the AD9546 detects a faulty reference, it can automatically switch to another (valid) reference with the user-programmable reference selection priority.

The user also has access to two independent auxiliary NCOs. These NCOs provide for internally generated frequencies up to 65,535 Hz with a tuning resolution of 2^{-40} Hz (approximately 1 pHz).

The DPLL portion of each PLL channel includes a programmable digital loop filter with extremely low loop bandwidth capability. Low loop bandwidth capability enables a significant reduction of jitter transfer from the reference input to the output. Each DPLL provides precise frequency translation by means of a 48-bit NCO capable of generating clock signals in the range of 162 MHz to 350 MHz. Furthermore, both DPLLs enable support of manual or automatic transition to holdover operation.

Holdover operation constitutes the device generating an output frequency synthesized from the system clock (as opposed to translating a frequency from a reference input to an output). During holdover operation, the AD9546 continuously provides an output clock signal while the system clock is present. The AD9546 is also capable of setting the holdover output frequency based on a time average of the output frequency history prior to the transition to holdover operation.

The output signal from the DPLL routes to the input of the APLL, which upconverts the signal to a range of 2.424 GHz to 3.232 GHz (APLL0) or 3.232 GHz to 4.040 GHz (APLL1). A high frequency divider (÷2) follows the APLL VCO output providing a factor of 2 frequency reduction prior to delivery to the clock distribution section.

The clock distribution section consists of a bank of dividers (Q) and output drivers dedicated to each output pin. The PLL0 channel has six Q divider/driver units, whereas the PLL1 channel has four Q divider/driver units. Each Q divider has 32-bit programmable division depth and programmable phase offset control. The drivers operate up to 500 MHz and are configurable as a single-ended or differential output with adjustable output current (7.5 mA, 12.5 mA, or 15 mA).

The AD9546 includes an integrated temperature sensor and system clock compensation circuitry, which allows the device to adjust automatically for frequency drift errors associated with the system clock source.

INPUT/OUTPUT TERMINATION RECOMMENDATIONS

SYSTEM CLOCK INPUTS

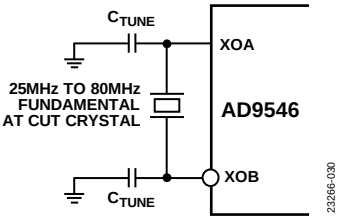


Figure 30. System Clock Input (XOA/XOB) Using the Crystal Path ($C_{TUNE} = 2 \times (C_{LOAD} - C_{STRAY})$, Where Typical $C_{STRAY} = 2$ pF to 5 pF); see the Crystal Path Section for Capacitor Definitions)

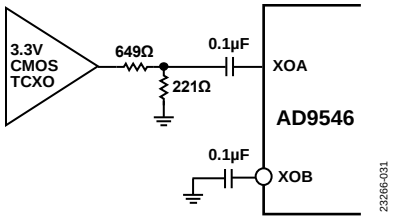


Figure 31. System Clock Input (XOA, XOB) Using the Direct Path with a TCXO or Oven Controlled Crystal Oscillator (OCXO) Having a 3.3 V CMOS Output

REFERENCE CLOCK INPUTS

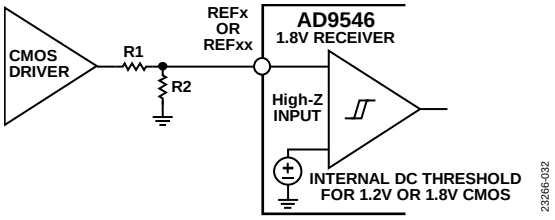


Figure 32. Single-Ended DC-Coupled Mode, 1.2 V or 1.8 V CMOS

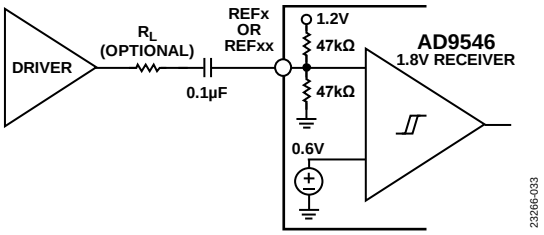


Figure 33. Single-Ended AC-Coupled Interface Mode

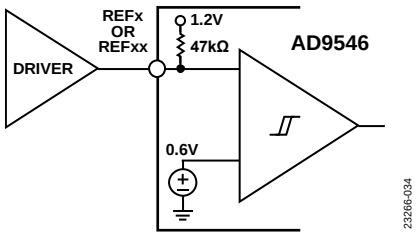


Figure 34. Single-Ended Internal Pull-Up Resistor Mode

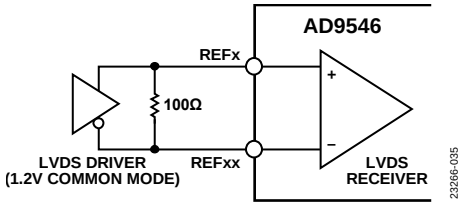


Figure 35. Differential LVDS Input Mode

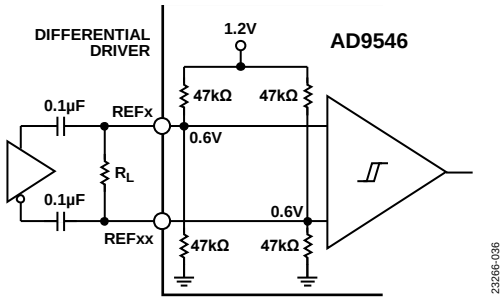


Figure 36. Differential AC-Coupled Mode ($R_L = 100\ \Omega$ Is Recommended, Except For HCSL)

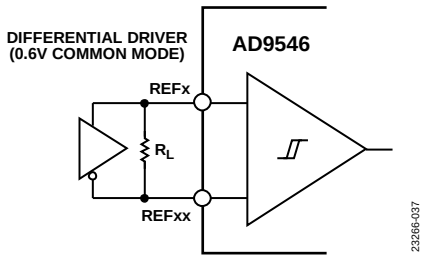


Figure 37. Differential DC-Coupled Mode

CLOCK OUTPUTS

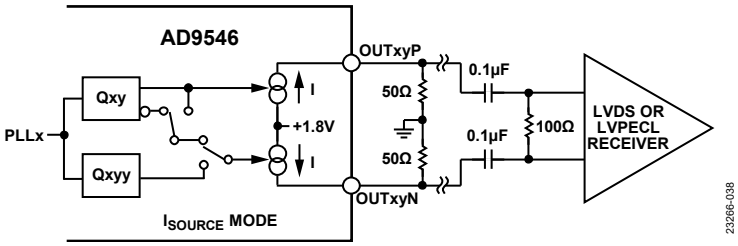


Figure 38. LVDS-Compatible Output Swing, AC-Coupled (375 mV (Differential) Across 100 Ω Termination Resistor for I = 15 mA)

The output drivers of the AD9546 are capable of either sourcing or sinking current. Each pin of an output pin pair (OUTxyP/OUTxyN) has an independent current source. However, the source or sink function applies to both current sources of an output pin pair (that is, either the source current (I_{SOURCE}) or sink current (I_{SINK}) of both pin drivers). The current source or sink current of the individual driver is either on or off, depending on the output logic level of the associated Qxy or Qxyy divider (0 = off, 1 = on).

The Qxy divider has both normal and inverted logic outputs, which means when the Qxy divider is connected to both output drivers (as in Figure 38, for example), one current source is on while the other is off, allowing a differential output signal. See the Distribution Clock Output Drivers section for configuring the clock output drivers.

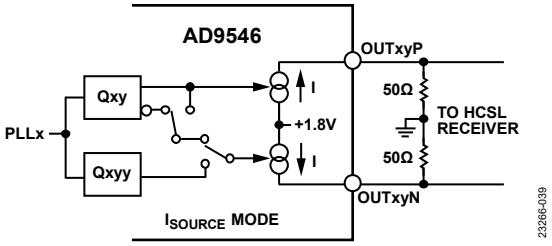


Figure 39. HCSL Output, Peak-to-Peak Voltage ≈ 750 mV per Section (I = 15 mA)

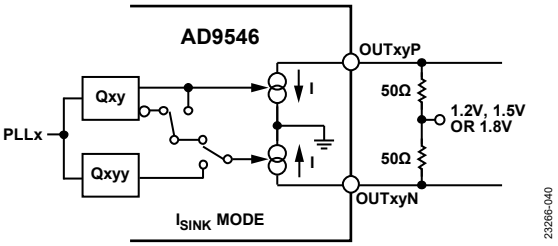


Figure 40. CML Output (I = 7.5 mA; I = 15 mA Options for 1.5 V or 1.8 V Supply)

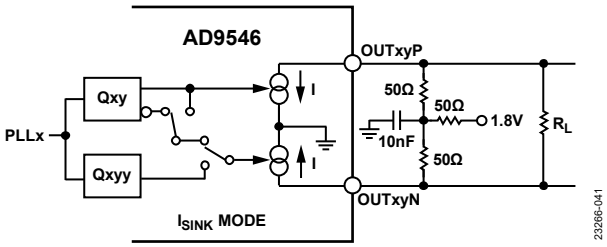


Figure 41. LVDS-Compatible Output, 1.24 V Common Mode, T Network (I = 7.5 mA; I = 15 mA with Extra 100 Ω Termination, R_L)

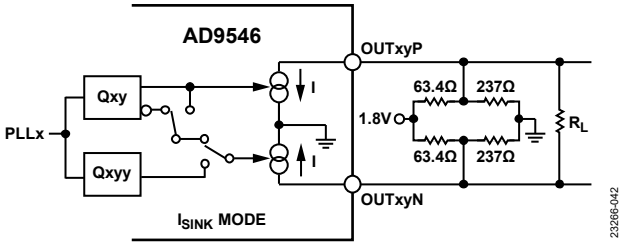


Figure 42. LVDS-Compatible Output, 1.2 V Common-Mode, Thevenin Bias Network (I = 7.5 mA; I = 15 mA with Extra 100 Ω Termination, R_L)

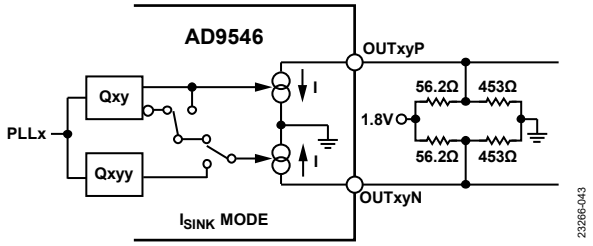


Figure 43. 2.5 V LVPECL or Double Amplitude LVDS-Compatible Boost Output, 1.5 V p-p, 1.24 V Common Mode (I = 15 mA)

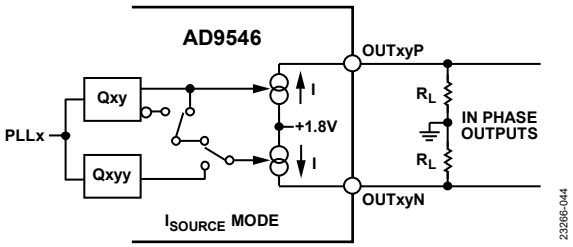


Figure 44. Single-Divider, Single-Ended Mode Providing In Phase Outputs (Current Source Mode)

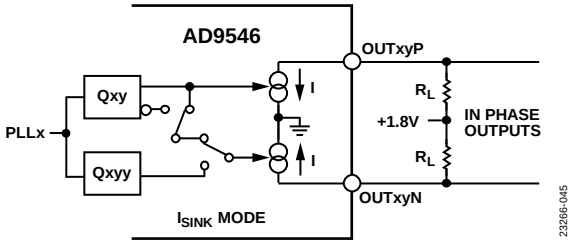


Figure 45. Single-Divider, Single-Ended Mode Providing In Phase Outputs (Current Sink Mode)

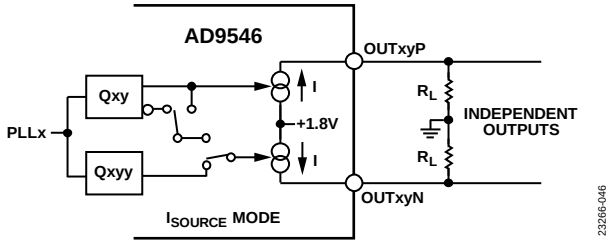


Figure 46. Dual-Divider, Single-Ended Mode Providing Independent Outputs (Current Source Mode)

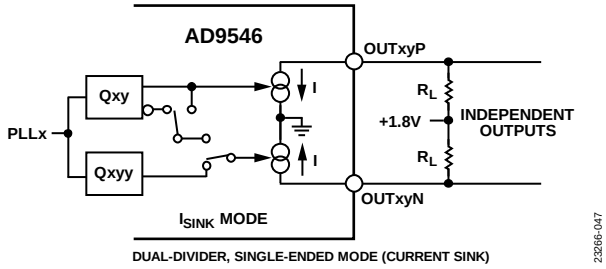


Figure 47. Dual-Divider, Single-Ended Mode Providing Independent Outputs (Current Sink Mode)

DIGITIZED CLOCKING

OVERVIEW

Digitized clocking enables precise timing transport (frequency and phase) among spatially separated devices by the exchange of numeric time codes to synchronize local clocks (assuming all devices have access to a common reference clock signal).

A digitized clocking system comprises multiple timing nodes with each node relying on a shared external time base (common clock reference). Because each node shares the same common clock reference, each node increments at the same rate, resulting in all nodes having the same time scale. Although all the nodes have the same time scale, there is no guarantee they share the same epoch. Digitized clocking relies on a user provided synchronization signal at each node to establish a common epoch, ultimately leading to all nodes being time aligned. A unique characteristic of digitized clocking is the ability to exchange numeric time codes between nodes via a digital bus, which provides a means to send multiple clocks over one digital bus (clock aggregation) instead of having to route multiple analog clock signals throughout a system. The AD9546 contains all the necessary building blocks to implement one digitized clocking node.

Figure 48 shows the individual components of a complete digitized clock node as implemented in the AD9546. These digitized clocking components include the following:

- A system clock PLL for generating a system clock frequency for internal digital timing functions (see the System Clock PLL section)
- A common clock DPLL to phase lock to a reference time base (common clock reference) and maintain a local time scale
- A physical clock converter for converting the rising edges of a physical clock to time stamps derived from the system clock
- A physical clock generator for converting time stamps derived from the common time scale to a physical clock output signal
- A CCS for creating a common time scale by assigning an epoch to the local time scale
- A UTS to convert internal time stamps to time codes based on the common time scale and provide those time codes to the user for external use
- An IUTS to convert a series of time codes based on the common time scale (a digitized clock signal) provided by the user to internal time stamps

Figure 48 shows a variety of time stamp sources. Certain digitized clocking components require the ability to connect to any one of the time stamp sources, which is the reason those components include a time stamp selection mux at their input.

SYSTEM CLOCK PLL COMPONENT

The system clock PLL typically uses an external crystal resonator as a frequency source, which tends to offer optimal overall phase noise performance. The system clock PLL synthesizes a high frequency internal system clock signal (~2.4 GHz) from the external frequency source, which provides the basic internal timing for the device. The TDCs of the device use the system clock to generate time stamps (see the Time to Digital Converter (TDC) section).

Note that the internal system clock signal is one of the two frequency sources the common clock DPLL component uses to generate the local time scale.

COMMON CLOCK DPLL COMPONENT

The purpose of the common clock DPLL is to produce a local time scale. The common clock DPLL uses the system clock for internal timing, but the phase locks to a reference time base, the common clock reference. The output of the common clock DPLL is the local time scale, which constitutes the accumulated period of cycles of the common clock reference and provides an internal sense of time for the digitized clocking components.

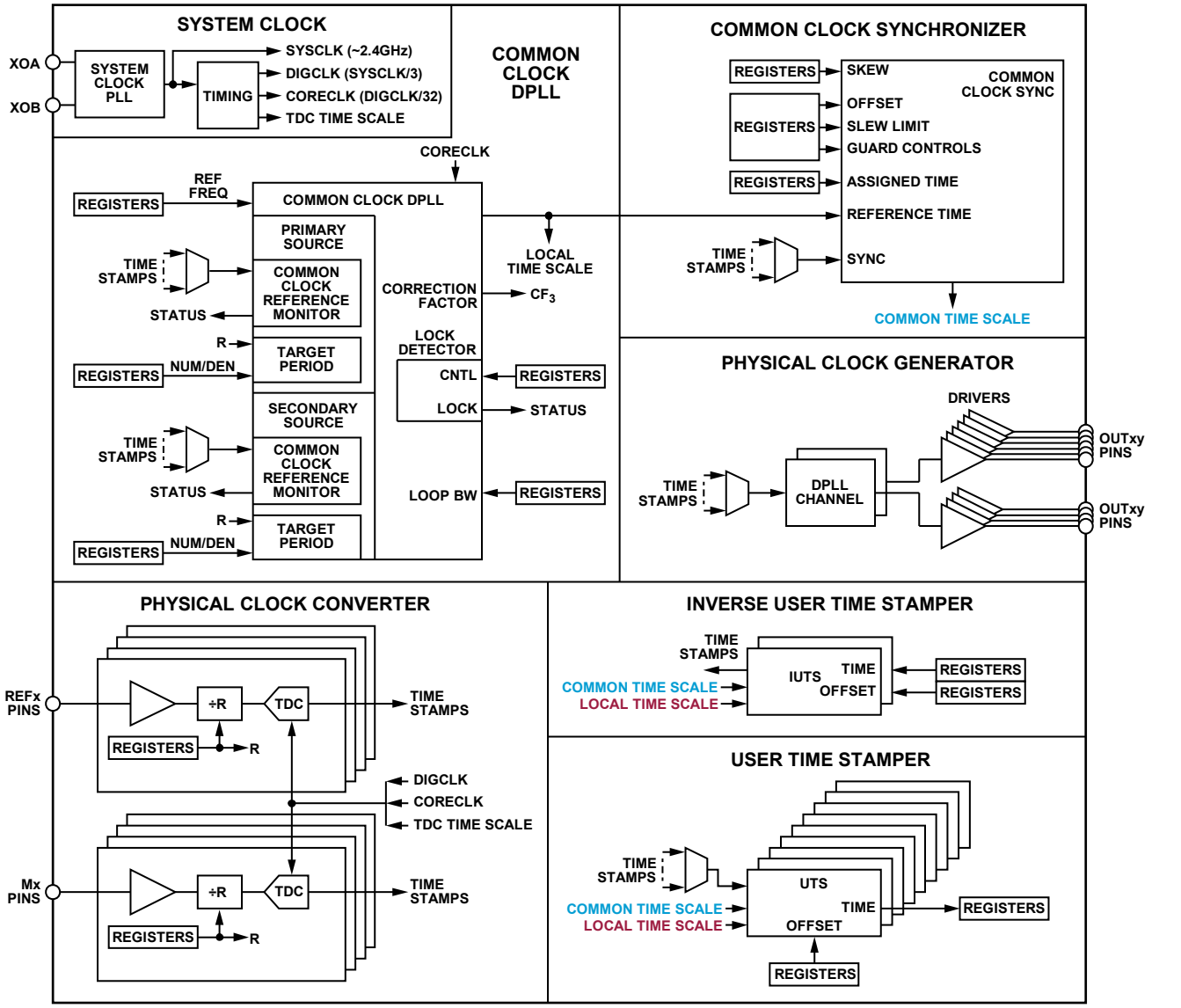
The inherent closed-loop architecture of the common clock DPLL results in the generation of an error signal, Correction Factor 3 (CF_3), that effectively compensates for system clock fluctuations. Note the CF_3 signal is available to the system clock compensation functional block of the AD9546 (see the Compensation Method 3 section).

PHYSICAL CLOCK CONVERTER

The physical clock converters provide an input to receive an external clock signal, provide optional integer frequency division, and convert rising clock edges to numeric time stamps. The time stamps are then available for use by those digitized clocking components that require a time stamp source.

PHYSICAL CLOCK GENERATOR

The physical clock generator makes use of the AD9546 DPLL channels and output distribution drivers to produce an analog clock output signal from time stamps. In a digitized clocking application, the time stamps derive from the common time scale. Because the DPLL can accept those time stamps as its reference input, the output of the physical clock generator consists of analog clock signals.



NOTES
1. DIGCLK IS DIGITAL CLOCK.
2. NUM/DEN ARE NUMERATOR/DENOMINATOR.

Figure 48. Digitized Clock Components

COMMON CLOCK SYNCHRONIZER COMPONENT

The common clock synchronizer provides a precision method for assigning an epoch to the local time scale yielding the common time scale. Essentially, the user provides a synchronization trigger event, the rising edge of an external analog input signal, which the device time stamps. The time stamp serves as an internal trigger event. Immediately following the trigger event (and before the next trigger event), the user programs the device with a time code corresponding to the time associated with the trigger event and asserts an IO update, which aligns the common time scale with the programmed time code. In this way, the common time scale carries the proper time. By carrying out the synchronization process across all nodes (with a distributed common clock as the time base for every node) and properly accounting for time delay in the system, the common time scale of every node increments at the same rate and carries the same epoch. Thus, all nodes are time aligned to a high degree of precision.

USER TIME STAMPER (UTS) AND INVERSE USER TIMER STAMPER (IUTS) COMPONENTS

A typical digitized clocking system can use one AD9546 as a master time unit with various nodes synchronized to the master. Thus, a means of transporting numeric time codes between nodes is essential for transporting clocks and correcting time errors throughout the system. Such is the role of the UTS and IUTS. The AD9546 possesses nine UTS units and two IUTS units.

A UTS converts internal time stamps to time codes (numeric time values based on the common time scale) and provides the user with a means to read the corresponding time codes. Because time codes relate to the common time scale, the UTS provides a means to export a digitized clock signal to remote digitized clocking nodes.

An IUTS provides the user with a means to send a digital clock signal to a node (a digital clock signal being a continuous series of uniformly increasing numeric time codes). An IUTS converts the digital clock signal to time stamps (based on the common time scale), which can be distributed internally as needed.

A DIGITIZED CLOCKING NODE EXAMPLE

Figure 49 shows the interaction of the various digitized clocking components as applied to a single node in a hypothetical digitized clocking system. The analog input signals consist of the common clock (CC), a synchronization source (SS), and a local user clock (LUC). The local user clock is any analog clock signal intended for transport to another node or for local use as a local analog clock output signal.

The three analog input clock signals route through their respective physical clock converters, which convert the analog clock signals to time stamps. The common clock time stamps route to the common clock DPLL, which is responsible for regulating the local time scale. The synchronization source time stamps route to the common clock synchronizer, which allows the user to translate the local time scale to a common time scale with a known epoch. The local user clock time stamps route to a UTS, allowing an external processor to capture local user clock time codes related to the common time scale.

The local user clock time codes captured by the external processor exist as a digitized clock signal. Thus, the processor can transport the digitized clock signal in the form of data packets to remote nodes.

Alternatively, the processor can route the local user clock time codes from a local UTS to a local IUTS, which converts the local user clock time codes back into time stamps. The time stamps are then available to a physical clock generator for conversion to an analog output signal. This basic example of translating input clock edges to numeric values (time codes) and reconstituting an analog clock signal from the time codes within a single chip demonstrates the concept of digitized clocking.

The local user clock time codes can be transported to a remote node (via the processor) and then converted to an analog signal at the far end. Assuming the remote node shares the same common clock (a requirement for digitized clocking), and the processor at the remote node properly synchronizes the remote common time scale, then it is possible to replicate the local user clock at remote nodes in the native time domain of the local user clock.

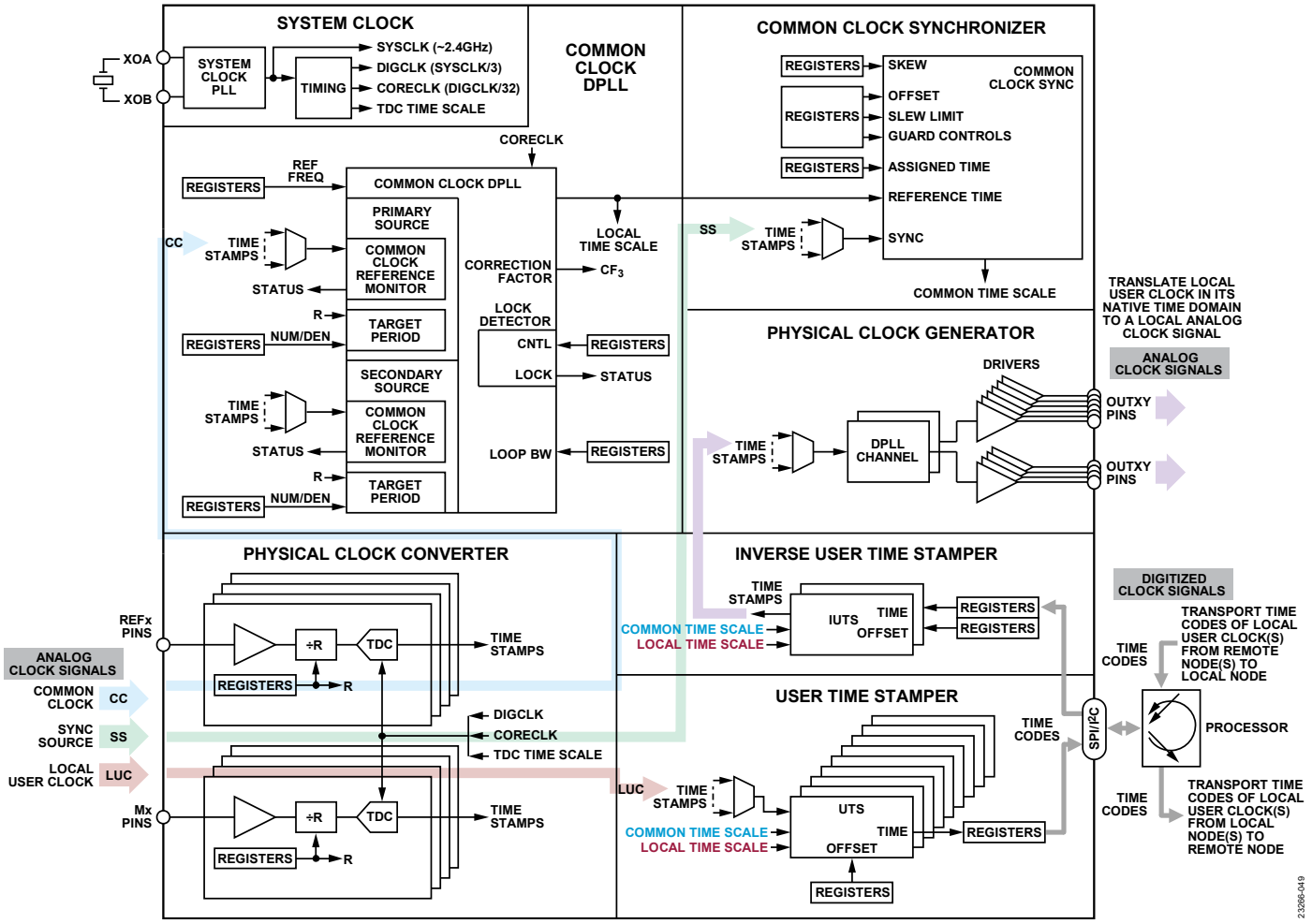


Figure 49. Digitized Clocking Node

COMMON CLOCK DPLL (CCDPLL)

OVERVIEW

The CCDPLL embodies most of the hardware associated with the auxiliary DPLL (see the Compensation Method 3 subsection of the System Clock Compensation section). However, the CCDPLL incorporates additional enhancements to the auxiliary DPLL hardware to meet the needs of a digitized clocking system. Figure 50 shows a functional block diagram of the CCDPLL.

COMMON CLOCK REFERENCES

The CCDPLL supports two independent time stamp sources as common clock reference (CCR) frequency inputs. CCR0 is the primary frequency reference, whereas CCR1 is the secondary frequency reference. The primary and secondary distinction matters with respect to the handling of the CCDPLL of automatic switching between references (see the Common Clock Reference Switchover section).

When employing only a single CCR, the user must use CCR0. That is, assigning a reference to CCR1 with no reference assigned to CCR0 is an invalid condition.

It is best practice to use REFB, REFBB, or both input pins for the common clock DPLL reference frequency because the REFB and REFBB inputs are the only reference inputs that provide access to the analog loopback function (see the Analog Clock Loopback section).

CCR Enable and CCR Frequency Source

Each CCR has an enable bit (Bit 7 of Register 0x0D10 and/or Register 0x0D20 for CCR0 and CCR1, respectively). Logic 0 (default) disables the CCR, whereas Logic 1 enables it. When operating the device with both CCRs, the enable bit gives the user a simple mechanism to force a reference switchover (see the Common Clock Reference Switchover section) by disabling the active CCR.

The user can independently assign each CCR to a specific time stamp source (via Bits[4:0] of Register 0x0D10 and Register 0x0D20 for CCR0 and CCR1, respectively). The default value, 0x31, is the null selection. That is, the associated CCR has no time stamp source.

For CCR0, programming a nondefault value for the enable or time stamp source selection settings results in the CCR0 time stamp source selection overriding the reference TDC selection of the auxiliary DPLL (see the Compensation Method 3 subsection of the System Clock Compensation section).

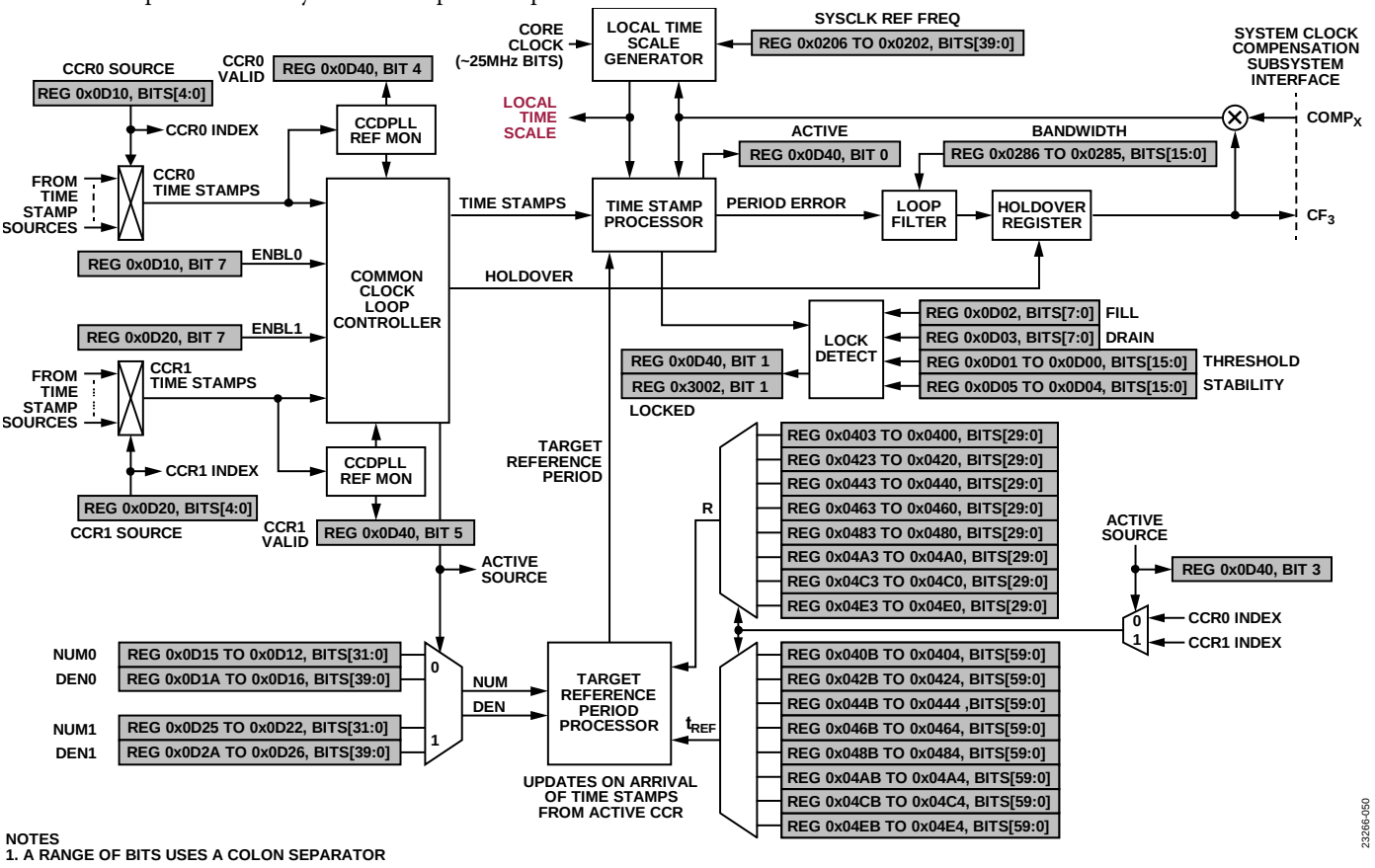


Figure 50. Common Clock DPLL Block Diagram

COMMON CLOCK REFERENCE MONITOR

The reference monitor for CCR0 uses the reference monitor hardware of the legacy auxiliary DPLL. Because the CCDPLL supports reference redundancy, however, it requires a second reference monitor for CCR1. Thus, the CCDPLL implements a duplicate reference monitor for CCR1. The CCR1 reference monitor includes a validation delay feature (see the Common Clock Reference Switchover section).

Each CCR reference monitor continuously observes the post R divider period (t_{CCR}) of each CCR source, where $t_{CCR} = R \times t_{REF}$ (see the Reference Monitor Controls section of the Reference Monitor section for details on t_{REF}). The reference monitor considers a CCR valid when the period of the specified source of the CCR is within ±3.125% of t_{CCR}. Conversely, a CCR is invalid when the CCR reference monitor fails to detect the next rising edge event within two t_{CCR} periods, or the user clears the enable bit associated with the CCR.

Each CCR reference monitor provides a status bit to indicate that the CCR is valid (Logic 1), via Bit 4 and Bit 5 of Register 0x0D40, for CCR0 and CCR1, respectively. The status of the CCR reference monitors (valid and invalid) are also part of the IRQ mechanism (see Bits[7:4] of Register 0x301D and the Interrupt Request (IRQ) section). The status bits (Bit 4 and Bit 5 of Register 0x0D40) are undefined for legacy auxiliary DPLL operation. Furthermore, when using the common clock DPLL, the legacy auxiliary DPLL reference status bit (Bit 2 of Register 0x3002) is undefined.

The CCDPLL uses the status of the CCR reference monitors to coordinate automatic reference switchover as described in the Common Clock Reference Switchover section.

CCDPLL LOCK DETECTOR

By default, the CCDPLL uses the same lock detector as the legacy auxiliary DPLL (see the Auxiliary DPLL Reference Monitor Status section of the System Clock Compensation section). However, the auxiliary DPLL lock detector has fixed parameters that are not adjustable by the user. Thus, the default lock detector is not well suited to digitized clocking applications. To resolve this issue, the CCDPLL has access to a specialized lock detector with programmable features, the CCDPLL lock detector. The general functionality of the CCDPLL lock detector is identical to the phase lock detector employed in the channel DPLLs (DPLL0 and DPLL1). Refer to the DPLL Phase Lock Detector section of the Digital PLL (DPLL) section for details.

The phase lock threshold, phase lock fill, and phase lock drain parameters of the CCDPLL lock detector appear in different registers than those associated with DPLL0 and DPLL1: Register 0x0D00 to Register 0x0D01, Register 0x0D02, and Register 0x0D03 for threshold, fill, and drain, respectively. Furthermore, the common clock DPLL uses the legacy auxiliary DPLL lock detector by default. To enable the CCDPLL lock detector, the user must program a nonzero value for the phase lock threshold parameter. Otherwise, the phase lock fill and drain parameters have no function, because they are meaningless in the context of the legacy auxiliary DPLL.

When the user enables the CCDPLL lock detector, a stability timer becomes available for use. The stability timer allows the user to specify a fixed delay for indicating a locked status after the phase lock detector makes a transition from an unlocked to a locked state. The time delay provides for a more robust indication of lock, because it allows additional time for lock and unlock chatter to resolve. The stability timer uses an unsigned 16-bit value (units of milliseconds) via Register 0x0D04 to Register 0x0D05. Programming a nonzero value enables the stability timer. Otherwise, the CCDPLL lock detector bypasses the stability timer.

The user has access to the status of the CCDPLL lock detector via Bit 1 of Register 0x0D40 or Bit 1 of Register 0x3002. Logic 1 indicates locked, and Logic 0 indicates unlocked.

CCDPLL LOOP FILTER

The CCDPLL loop filter and the auxiliary DPLL loop filter use the same loop filter hardware. As such, the user can program the bandwidth of the loop filter as described in the Auxiliary DPLL Loop Bandwidth section of the System Clock Compensation section.

COMMON CLOCK REFERENCE (CCR) PERIOD DECLARATION

Because a CCR is a time stamp source, the user programs the expected reference period, t_{REF}, in the reference monitor controls for a specific source (see the Reference Monitor Controls section of the Reference Monitor section for details on t_{REF}). However, t_{REF} is subject to rounding errors, which are typically small enough to ignore.

Although t_{REF} is sufficiently precise for use by the reference monitors, it is generally not precise enough for use by the CCDPLL. The reason is the CCDPLL generates a local time scale based on integrating t_{REF}. Integrating t_{REF}, however, poses a problem because even a minuscule error accumulates over time. To remedy this problem, the CCDPLL provides a mechanism for declaring the reference period rationally as a numerator (NUM) and denominator (DEN).

For example, consider a 38.88 MHz reference frequency. The period, in rational terms, is 1/38,880,000. For this case, the numerator = 1 and the denominator = 38,880,000. The user programs the numerator and denominator values in Register 0x0D12 to Register 0x0D1A for CCR0 and Register 0x0D22 to Register 0x0D2A for CCR1.

The rational period functionality is inactive by default (that is, denominator = 0), which means the common clock DPLL uses t_{REF} strictly as the period definition by default. As such, to make use of the rational period functionality, the user must program the denominator with a nonzero value.

COMMON CLOCK REFERENCE SWITCHOVER

The CCDPLL uses an integrated controller to handle the activation and deactivation of the loop based on the status of the CCRs. Because the CCDPLL supports redundant CCRs, the controller requires a priority scheme to handle transitions from one CCR to another.

The controller gives nonrevertive priority to CCR0, meaning that after the controller activates the loop using a specific CCR, the controller continues using that CCR until the CCR invalidates. If both CCRs are operational at power-up, however, there is the risk that CCR1 may validate before CCR0, causing CCR1 to be the active reference, which is contrary to the desired nonrevertive priority assigned to CCR0. The controller solves this potential problem by adding a validation delay to the validation of CCR1. The delay ensures nonrevertive priority to CCR0, but the delay is only in effect when CCR0 is invalid (the delay is unnecessary otherwise).

The user can determine which reference is currently active via Bit 3 of Register 0x0D40. Logic 0 or Logic 1 indicates CCR0 or CCR1 is active, respectively. Bit 3 is undefined for legacy auxiliary DPLL operation.

ACTIVE STATUS

The CCDPLL provides a status bit that indicates the CCDPLL is actively tracking a reference input signal. To check the active status of the CCDPLL, use Bit 0 of Register 0x0D40, where Logic 1 indicates the common clock DPLL is active.

COMMON CLOCK SYNCHRONIZER (CCS)

OVERVIEW

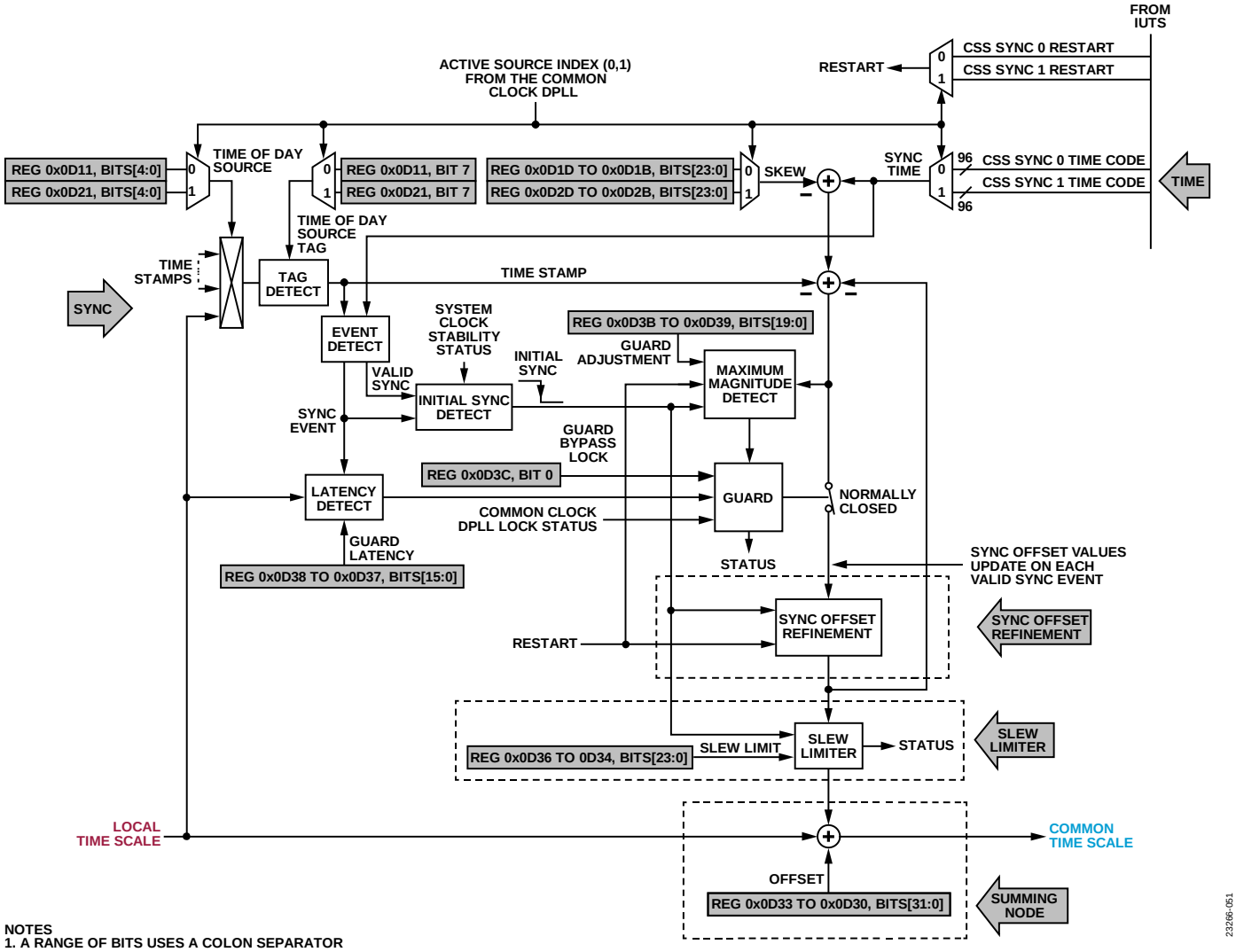


Figure 51. Common Clock Synchronizer

The CCDPLL phase locks to a physical common clock source (the common clock reference) and generates an internal local time scale. The local time scale constitutes time as the accumulation of the period of the system clock as defined by the system clock input frequency programmed by the user. The CCDPLL relies on a crystal resonator (via the XOA and XOB pins) for internal timing, which is a relatively unstable frequency source. However, the architecture of the CCDPLL allows it to phase lock to the common clock reference even with the relative instability of the crystal resonator. Thus, the local time scale is immune to instability associated with the frequency source of the system clock.

Because the CCDPLL derives its timing from the system clock but the phase locks to the common clock reference, instabilities associated with either source contribute to the loop stability of the CCDPLL. Thus, the user must tailor the loop bandwidth of the CCDPLL in accordance with the combined stability of the system clock and common clock reference frequencies. The frequency stability of the sources puts a limit on the minimum allowable loop bandwidth that yields a stable loop and minimizes the phase offset associated with the ability of the PLL to track frequency variations.

Although the local time scale is precise (frequency stable), it is likely not accurate. That is, the time indicated by the local time scale is with respect to an arbitrary starting time. Stated another way, the epoch of the local time scale is arbitrary. The purpose of the CCS is to provide a means to produce an internal common time scale with a precise epoch determined by the user. In practice, the user provides a physical trigger event (the synchronization arrow in Figure 51) along with a numeric entry of the time associated with the trigger event (the time arrow in Figure 51), which effectively assigns an epoch. The CCS uses the trigger event and numeric entry to produce a precise time shifted version of the local time scale (the common time scale) synchronized to the desired epoch. More importantly, in the context of multiple AD9546 devices in a digitized clocking system, the user can use the CCS of each device to obtain precise alignment of the common time scale across devices.

The diagram in Figure 51 is a functional representation of the CCS. The key points of interest are the time and synchronization inputs, synchronization offset refinement, the slew limiter, and the summing node with the common time scale (a time shifted version of the local time scale) as its output.

SYNCHRONIZATION SOURCE

The synchronization source constitutes the time stamp source associated with the synchronization input of Figure 51. The arrival of each time stamp from the selected synchronization source constitutes a synchronization event. In operation, when a synchronization event occurs, the user updates the synchronization time (see the Synchronization Time section) associated with that synchronization event, but before the occurrence of the next synchronization event.

Because the common clock DPLL has access to a primary and secondary (optional) reference source for producing the local time scale, the CCS also has access to a primary and secondary (optional) synchronization source. Whenever the common clock DPLL switches between primary and secondary references, the CCS automatically switches to its primary or secondary synchronization source, commensurate with the common clock DPLL (a one to one correspondence).

To assign the primary synchronization source, use Bits[4:0] of Register 0x0D11. To assign the secondary synchronization source, use Bits[4:0] of Register 0x0D21. In legacy auxiliary DPLL operation, the CCS uses the source specified in Bits[4:0] of Register 0x0D11.

If possible, the recommendation is to use REFB and/or REFBB as the synchronization source to the CCS because REFB and REFBB provide access to the analog loopback function (see the Analog Clock Loopback section).

The local time scale appears as one of the synchronization sources associated with the synchronization input in Figure 51. The user can employ the local time scale for immediate synchronization rather than using a time stamp source. Immediate synchronization is useful for forcing a synchronization trigger in applications where the user is not transporting the common time scale across multiple devices or does not require the common time scale to have a specific epoch, but wants to use one (or more) of the digitized clocking resources (a UTS, for example). In such cases, the immediate trigger is necessary to make the CCS ready for use by the digitized clocking resources. To select immediate synchronization, use 0x1E or 0x1F for Bits[4:0] associated with the primary or secondary synchronization source selection.

When the common clock DPLL experiences a reference switchover, there is a direct impact on the CCS. First, the synchronization source changes to primary or secondary in keeping with the new reference source (primary or secondary). Likewise, the synchronization time (see the Synchronization Time section) and skew (see the Adding Time Skew to the Synchronization Time section) change in keeping with the new reference source (primary or secondary). In addition, the CCS flushes any synchronization data previously captured. However, the synchronization offset presently applied by the CCS to the common time scale remains in effect. Likewise, the filtering state of the synchronization offset refinement block (see the Synchronization Offset Refinement section) and the condition of the slew limiter (see the Synchronization Slew Limiter section) also remain in effect.

Because the CCS flushes old synchronization data on a reference switchover, the CCS has no synchronization event to which it can compare the first synchronization time update. This condition causes a synchronization error and prevents the CCS from performing the intended synchronization. However, the next complete synchronization event clears the synchronization error with synchronization proceeding normally from that point onward.

TAGGED SYNCHRONIZATION

Tagged synchronization only applies when the synchronization source originates from a TDC generating tagged time stamps. Thus, tagged synchronization does not apply to immediate synchronization (that is, using the common time scale as the synchronization source).

For a source with tagged time stamps (see the Tagged Time Stamps section), the user has the option of synchronizing to only the tagged time stamps from that source. To enable the tagged synchronization feature, program a Logic 1 in Bit 7 of Register 0x0D11 for the primary synchronization source and/or Register 0x0D21 for the secondary synchronization source.

SYNCHRONIZATION TIME

The user programs the time associated with a forthcoming trigger event via a 96-bit time value written to Register 0x0F0A to Register 0x0F15. However, the user must also program Register 0x0F09, Bits[4:0] = 0x30 or 0x31 to associate the time value with the primary or secondary CCS source, respectively (see Figure 54 and Table 45 in the Inverse User Time Stamper (IUTS) section). Then, upon issuing an IO update (via Register 0x000F, Bit 0), the CCS considers the programmed value as the time associated with the previous synchronization event (via the synchronization path in Figure 51).

When the device initializes, the CCS has no synchronization event to which it can compare the first synchronization time update. Thus, the first synchronization event following initialization causes a synchronization error (see the Synchronization Guard and Ready Status sections). However, a subsequent synchronization event clears the synchronization error and synchronizes the CCS accordingly.

There are two options for the time format: fractional seconds and fractional nanoseconds (PTP format). Figure 52 shows the register arrangement for both formats. Regardless of the format, the register value for the integer portion of the seconds is the same.

For example, consider a synchronization time value of 148,264.05394857 sec. The integer seconds portion is 148,264, which, in 48-bit hexadecimal format, is 0x 0000 0002 4328.

In fractional seconds format, the fractional portion of the seconds entry (to the right of the virtual decimal point) constitutes a 48-bit number with an LSB weight of 2^{-48} sec. Using the same synchronization time value as in the previous example, the fractional seconds portion is 0.05394857, which converts to units of 2^{-48} sec as follows:

$$0.05394857 \times 2^{48} = 15,185,172,484,323.19496192$$

Round the result to the nearest integer, which yields 15,185,172,484,323, and convert to 48-bit hexadecimal format: 0x 0DCF 92CF D0E3.

In PTP format, the fractional portion of the seconds entry (to the right of the virtual decimal point) constitutes a 46-bit number (thus, the exclusion of the two fractional nanoseconds MSBs in the bottom half of Figure 52) with an LSB weight of 2^{-16} ns. Using the same fractional portion value (0.05394857), convert to units of 2^{-16} ns as follows:

$$0.05394857 \times 10^{9} \times 2^{16} = 3,535,573,483,520$$

This value in 46-bit hexadecimal format (rounded to the nearest integer, if necessary) is 0x 0337 309A 0000.

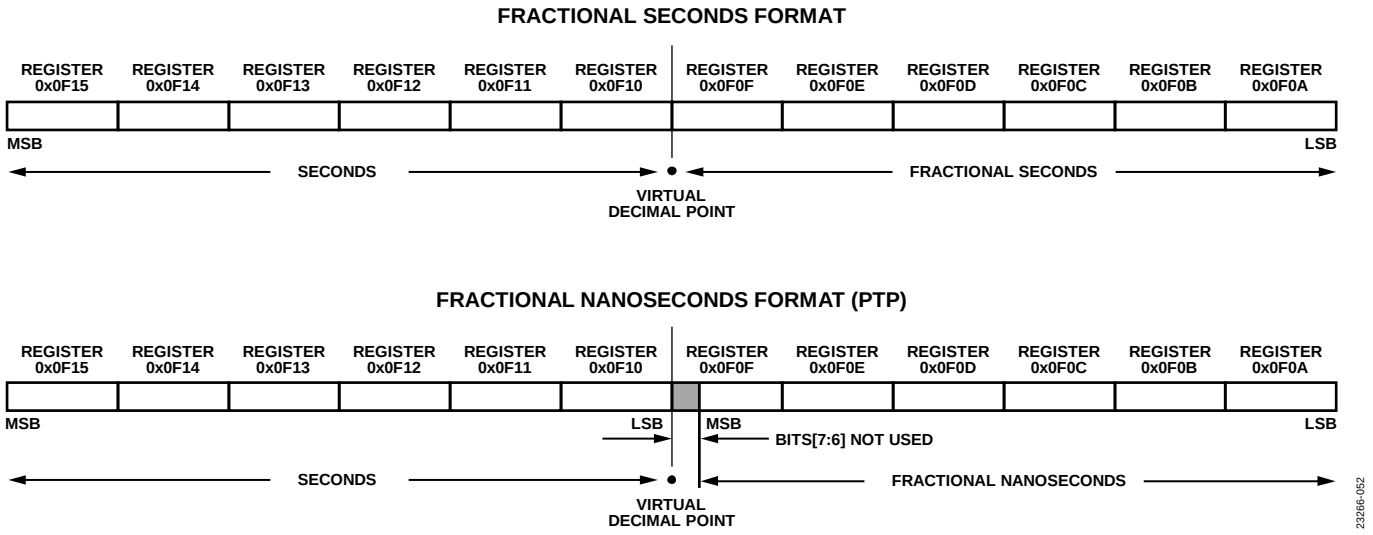


Figure 52. Time Formats

ADDING TIME SKEW TO THE SYNCHRONIZATION TIME

In a digitized clocking system with redundant common clock references, a time skew specific to each reference path may exist. The user can compensate the skew by programming the appropriate time skew for each path.

The skew programs as a 24-bit signed value. Primary skew is via Register 0x0D1B to Register 0x0D1D. Secondary skew is via Register 0x0D2B to Register 0x0D2D.

The 24-bit programmed time skew is in units of 2^{-48} sec, yielding a range of approximately ±29.8 ns. For example, to program a time skew of −5 ns (-5×10^{-9} sec), first convert to the appropriate units and round the result to the nearest integer as follows:

$$-5 \times 10^{-9} \times 2^{48} = -1{,}407{,}375$$

This resulting value in 24-bit hexadecimal format is: 0x EA 8671.

ADDING TIME OFFSET TO THE SYNCHRONIZATION TIME

In a digitized clocking system with redundant common clock references, a time skew common to both reference paths may exist. The user can compensate for the common time skew by programming the appropriate time offset value.

The time offset programs as a 32-bit signed value in Register 0x0D30 to Register 0x0D33 in units of 2^{-48} sec, yielding a range of approximately ±15.26 µs. For example, to program a time skew of 50 ns (50×10^{-9} sec), first convert to the appropriate units and round the result to the nearest integer as follows:

$$50 \times 10^{-9} \times 2^{48} = 14{,}073{,}749$$

This value in 32-bit hexadecimal format is: 0x 00D6 BF95.

SYNCHRONIZATION OFFSET REFINEMENT

In a digitized clocking application, the common clock synchronization process is typically iterative (though not a strict requirement). That is, the user applies a continuous synchronization source and continuously writes synchronization time values for each expected synchronization event. As such, each synchronization and time pair results in a specific synchronization offset value. However, a series of synchronization offset values is likely to introduce jitter onto the common time scale.

As such, the CCS employs a synchronization offset refinement block that gradually softens the impact of each new synchronization offset value. The goal is to yield an average synchronization offset over time. That is, the synchronization offset value produced by the first synchronization event (following a restart condition) propagates through the refinement block unaltered. As synchronization events continue to occur, the synchronization offset refinement block applies progressively more filtering (up to a limit) to subsequent synchronization offset values.

The user can restart the synchronization offset refinement at any time (see the Restart section).

MAXIMUM MAGNITUDE DETECTION

The CCS provides the user with the option to prevent excessive synchronization offset values from propagating into the synchronization offset refinement process. The user activates the maximum magnitude detection block by programming a nonzero guard adjustment value in Register 0x0D39 to Register 0x0D3B. A zero value (default) disables the maximum magnitude detection feature.

When activated, the maximum magnitude detector monitors each new synchronization offset value and compares it to the guard adjustment value. If a synchronization offset value exceeds the guard adjustment value, the maximum magnitude detector signals a guard event (see the Synchronization Guard section), thereby preventing the offending synchronization offset value from propagating into the synchronization offset refinement process.

The maximum magnitude detector ignores the first synchronization event following a restart or after the initial synchronization block indicates initial synchronization (see the Initial Synchronization section).

The guard adjustment value comprises a 20-bit unsigned number in units of 2^{-40} sec (~0.91 ps), yielding a maximum guard value of up to 954 ns (approximately). For example, to set the maximum magnitude detection value to 10 ns, first convert to the appropriate units and round the result to the nearest integer as follows:

$$10 \times 10^{-9} \times 2^{40} = 10{,}995$$

This value in 20-bit hexadecimal format is: 0x 0 2AF3.

LATENCY DETECTION

The normal synchronization process involves the occurrence of a synchronization event followed by the application of the synchronization time associated with the synchronization event. When a synchronization event occurs, the common clock synchronizer waits for the user to issue a synchronization time value associated with the synchronization event. The purpose of the latency detector is to allow the user to apply a time limit on how long the CCS waits for a synchronization time value before discarding the synchronization event. If a synchronization time value does not arrive within the specified period, the latency detector signals a guard event (see the Synchronization Guard section).

The user activates latency detection by writing a nonzero value to the 16-bit unsigned guard latency bit field in Register 0x0D37 to Register 0x0D38 in units of 2^{-16} sec, yielding a maximum latency guard time of slightly less than 1 sec. A zero value (default) disables the latency guard feature.

For example, to set latency detection value to 1 ms (10^{-3} sec), first convert to the appropriate units and round the result to the nearest integer as follows:

$$10^{-3} \times 2^{16} = 66$$

This value in 16-bit hexadecimal format is: 0x 0042.

SYNCHRONIZATION SLEW LIMITER

A typical digitized clocking system sometimes experiences the injection of a relatively large phase adjustment as part of the common clock synchronization process. An example is the injection of a phase jump due to switching between two CCRs having a static phase offset (see the Common Clock Reference Switchover section). Another example is a phase jump due to the injection of a phase adjustment to compensate for round trip delay in a clock signal path (see the Analog Clock Loopback section). Phase adjustments constitute instantaneous phase jumps, which equate to frequency impulses. The frequency impulses transfer to the common time scale via the action of the CCS.

To mitigate the frequency impulses resulting from phase adjustments, the CCS provides a slew limiter following the synchronization offset refinement block (see Figure 51). The slew limiter converts large phase jumps (frequency impulses) to a constant phase slope (which equates to a constant frequency offset). Thus, the action of the slew limiter puts a user defined upper bound on the frequency deviation associated with a phase adjustment. Because the slew limiter is physically situated at the output of the CCS (see Figure 51), it effectively limits the magnitude of the frequency offset injected on the common time scale as the result of a phase adjustment.

The user activates the slew limiter by writing a nonzero value to the 24-bit unsigned slew limit value in Register 0x0D34 to Register 0x0D36 in units of 2^{-36} sec/sec. For example, to constrain the frequency deviation on the common time scale to a maximum of 0.15 ppm (1.5×10^{-7}), first convert to the appropriate units and round the result to the nearest integer as follows:

$$1.5 \times 10^{-7} \times 2^{36} = 10{,}308$$

This value in 24-bit hexadecimal format is: 0x 00 2844.

When the user programs a new slew limit value, the slew limiter stalls briefly while implementing the slew limit value change. From an application perspective, the stalling behavior is inconsequential.

The user can bypass the slew limiter at any time by programming a slew limit value of zero. The slew limiter automatically bypasses for the initial application of a synchronization offset value. However, subsequent synchronization offset values are subject to slew limiting (assuming the user programs a nonzero slew limit value).

The status of the slew limiter is available via several mechanisms. Bit 6 of Register 0x0D40 is Logic 1 when the slew limiter is actively slewing and Logic 0 when it is not slewing. The status of the slew limiter is also available via an appropriately configured Mx status pin (see the Status and Control Pins section). In addition, slew limiter status is available as part of the IRQ mechanism (see the Interrupt Request (IRQ) section) via Bit 2 and Bit 3 of Register 0x301D, which indicate when the slew limiter starts and stops slewing, respectively.

RESTART

When a synchronization restart occurs, the next generated synchronization offset value propagates through the refinement block unaltered, with subsequent synchronization offset values receiving progressively more filtering over time. The user can manually restart the synchronization offset refinement process from the beginning at any time by programming Bit 0 of Register 0x0F08 to Logic 1.

The synchronization event that follows a synchronization restart is not the same as an initial synchronization event (a synchronization event following a device power-up, for example). Thus, a synchronization restart does not cause the slew limiter to bypass automatically on the first synchronization event following a synchronization restart.

SYNCHRONIZATION GUARD

Because common clock synchronization has a direct impact on the common time scale, safeguards are in place to mitigate corruption of the common time scale with invalid synchronization events. Such is the function of the guard element shown in Figure 51. Conceptually, the guard controls a normally closed switch that opens when the guard detects a guard event, thereby preventing the most recent synchronization offset value from propagating into the synchronization offset refinement process.

Assuming the initial synchronization has occurred (see the Initial Synchronization section), the following three conditions can trigger the synchronization guard to open the guard switch:

- The common clock DPLL unlocks (assuming the guard bypass lock bit is not set)
- A maximum magnitude detection event
- A latency detection event

Any one of these conditions trips the synchronization guard, which causes a synchronization error. To check the synchronization error status, use Bit 7 of Register 0x0D40, where Logic 1 indicates a synchronization error. Synchronization error status is also available via an appropriately configured Mx status pin (see the Status and Control Pins section) and via Bit 1 of Register 0x301D as part of the IRQ mechanism (see the Interrupt Request (IRQ) section).

Guard Bypass Lock

The user has the option of preventing an unlock condition of the common clock DPLL from triggering the synchronization guard via the guard bypass lock bit, Bit 0 of Register 0x0D3C. Logic 0 (default) allows an unlock condition to trigger the synchronization guard. Logic 1 prevents an unlock condition from triggering the synchronization guard.

Setting the guard bypass lock bit is useful when using the digitized clock resources of the AD9546 in a nondigitized clock application, for example, using two UTS resources to measure the phase offset between two clock signals.

INITIAL SYNCHRONIZATION

The initial synchronization detect block ensures the CCS starts in a known state. The CCS is not operational until the synchronization detect block indicates initial synchronization (the falling edge shown in Figure 51). Until initial synchronization occurs, the initial synchronization detect block holds the maximum magnitude detection block, the synchronization offset refinement block, and the slew limiter in a reset state. The initial synchronization detect block does not indicate initial synchronization until the system clock PLL is stable and a synchronization event occurs that does not trigger a guard event (see the Synchronization Guard section).

READY STATUS

The CCS provides a status indicating that the common time scale is ready to support digitized clocking resources. To check the ready status of the CCS, use Bit 2 of Register 0x0D40, where Logic 1 indicates the CCS is ready.

Ready status is also available via Bit 0 of Register 0x301D, which is part of the IRQ mechanism (see the Interrupt Request (IRQ) section) and through an appropriately configured Mx status pin (see the Status and Control Pins section).

USER TIME STAMPER (UTS)

OVERVIEW

The AD9546 has nine UTSs comprising the UTS system (see Figure 53). Each UTS in the UTS system converts internal device time stamps (based on the local time scale) to time codes based on the common time scale (see the Common Clock Synchronizer (CCS) section) and makes those time codes accessible to the user via the register map.

Because the UTS system relies on the common time scale (which originates with the CCS) the status of the CCS matters to the UTS system. As such, the UTS system is inoperative unless the CCS indicates a ready status.

The UTS system consists of nine independent UTSs with each UTS having a dedicated UTS channel control register. Time codes output by each UTS route to a readback FIFO. The FIFO contains time code samples originating from all active (enabled) UTSs. The FIFO samples include supplemental information allowing the user to discern the source of a time code sample and the status of the UTS channel associated with the time code sample. The FIFO also provides a status related to its own condition.

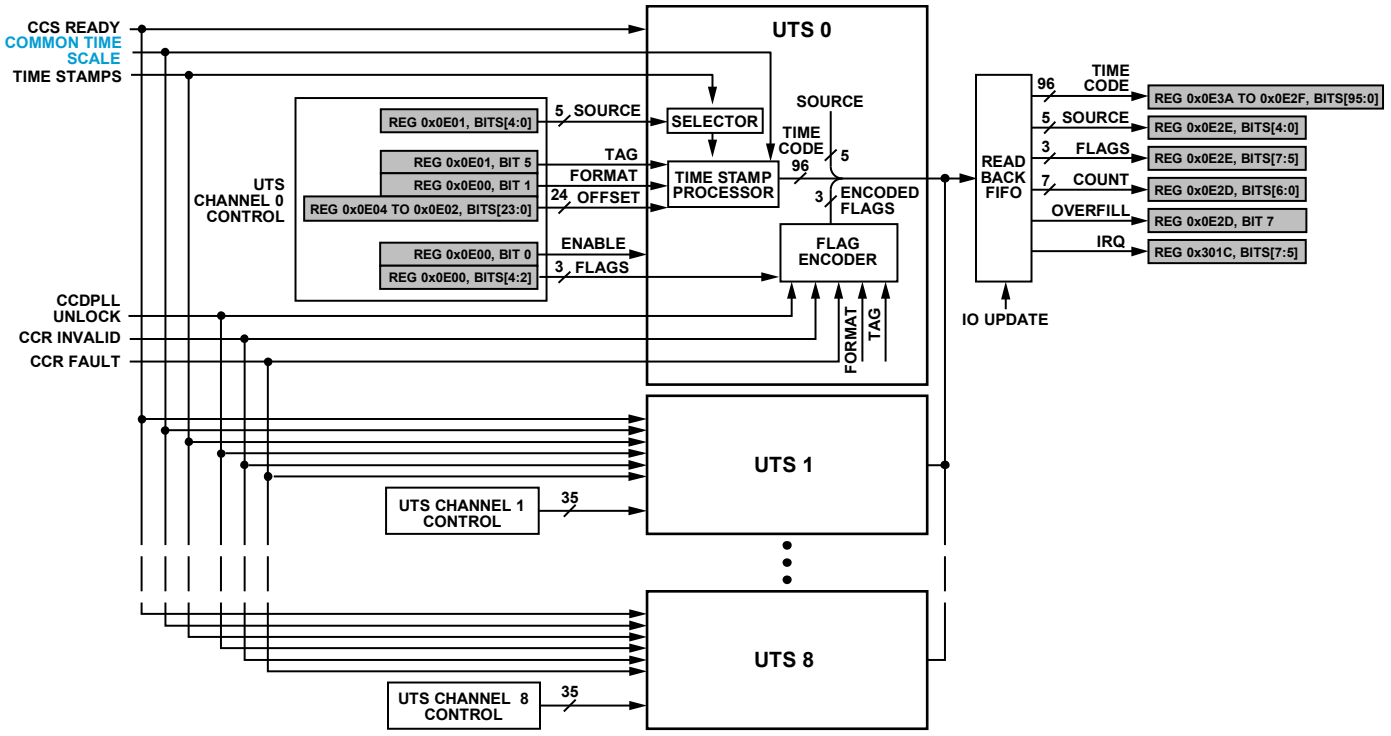
The UTS system is independent of the two user time stamp processors (see the User Time Stamp Processor (UTSP) section).

UTS CHANNEL CONTROL REGISTERS ADDRESSES

Each UTS channel has a dedicated UTS channel control register, differentiated by an integer, x (where x is 0 to 8). Each UTS Channel x control register occupies a range of register addresses. UTS 0 occupies Register 0x0E00 to Register 0x0E04. More generally, UTS x occupies the UTS 0 address range, but with an offset of x × 0x0005.

UTS CHANNEL ENABLE

Each UTS in the UTS system has an independent enable bit. The enable bit is Bit 0 in Register 0x0E00 for UTS 0, or more generally, for UTS x (where x = 0 to 8, integer), add x × 0x0005 to the register address given for UTS 0. Logic 1 enables the corresponding UTS channel. Logic 0 (default) disables the corresponding UTS, in which case it stops delivering new time code samples to the FIFO with no effect on old time code samples already delivered to the FIFO.



NOTES
1. A RANGE OF BITS USES A COLON SEPARATOR
2. ADDRESSES SHOWN FOR UTS CHANNEL CONTROL ARE SPECIFIC TO UTS 0

23266-053

Figure 53. UTS System

UTS CHANNEL TIME STAMP SOURCE

Each UTS channel has independent access to the various time stamp sources within the AD9546. The user assigns a specific time stamp source via Bits[4:0] of Register 0x0E01, or more generally, for UTS x (where x = 0 to 8, integer), add x × 0x0005 to the register addresses given for UTS 0.

The decimal value of the five source bits relates to the time stamp source per Table 42.

Table 42. Time Stamp Source Assignment Codes

Source Bit Field Value (Decimal)	Time Stamp Source
0	REFA (default)
1	REFAA
2	REFB
3	REFBB
4	DPLL0 feedback
5	DPLL1 feedback
6	Auxiliary REF0
7	Auxiliary REF1
8	Auxiliary NCO 0
9	Auxiliary NCO 1
10	Reserved
11	Auxiliary REF2
12	Auxiliary REF3
13	IUTS 0
14	IUTS 1
15 to 31	Reserved

The recommendation is to use the same time stamp source while a UTS channel remains enabled. However, the user can enable a UTS channel and assign a time stamp source simultaneously with no negative implications.

UTS CHANNEL TIME OFFSET

The user has the option of having a UTS channel apply a time offset to time stamps processed by the channel, with the time offset being relative to the common time scale in units of 2^{-48} sec. Time offset is via Bits[23:0] (signed) in Register 0x0E02 to Register 0x0E04 for UTS 0, or more generally, for UTS x (where x = 0 to 8, integer), add x × 0x0005 to the register addresses given for UTS 0.

For example, to program a time offset of −25 ns (-25×10^{-9} sec), first convert to the appropriate units and round the result to the nearest integer as follows:

$$-25 \times 10^{-9} \times 2^{48} = -7{,}036{,}874$$

This value in 24-bit hexadecimal format is: 0x 94 A036.

UTS CHANNEL TAG

The user can have a UTS channel selectively process only tagged time stamps (see the Tagged Time Stamps section) from the specified time stamp source (rather than every time stamp from that source). Tagged time stamp selection is via Bit 5 in Register 0x0E01 for UTS 0, or more generally, for UTS x (where x = 0 to 8, integer), add x × 0x0005 to the register addresses given for UTS 0. Logic 1 selects tagged time stamp processing, whereas Logic 0 (default) causes the UTS channel to process all time stamps from the specified time stamp source.

UTS CHANNEL TIME FORMAT

The user can select the output time format of each UTS channel independently via a format bit. Format selection is via Bit 1 in Register 0x0E00 for UTS 0, or more generally, for UTS x (where x = 0 to 8, integer), add x × 0x0005 to the register address given for UTS 0. Logic 0 (default) selects fractional seconds format, and Logic 1 selects PTP format.

See the Synchronization Time subsection of the Common Clock Synchronizer (CCS) section for details on fractional seconds format and PTP format.

UTS CHANNEL STATUS FLAG ASSIGNMENT

Whenever a UTS channel converts an input time stamp to a time code and delivers it to the FIFO, the UTS channel also includes a 3-bit status value as part of the time code.

The 3-bit status value is an encoding of the following five available status conditions:

- Invalid (common clock reference invalid)
- Fault (common clock reference fault)
- Unlocked (common clock DPLL unlocked)
- Tag (state of the tag bit associated with the UTS channel that processed the time stamp)
- Format (state of the format bit associated with the UTS channel that processed the time stamp)

The user can choose one of seven predefined 3-bit encoded status patterns (see Table 43). The selection is via the 3-bit flags bit field in Bits[4:2] of Register 0x0E00 for UTS 0, or more generally, for UTS x (where x = 0 to 8, integer), add x × 0x0005 to the register address given for UTS 0.

Each bit of the flags bit field encodes to a unique status bit (or logical combination of status bits). That is, each bit of the 3-bit flags bit field encodes to a specific status bit pattern based on the value of the 3-bit flags bit field.

Table 43. UTS Channel Flag Assignments

Flags Bit Field, Bits[4:2]	Bit 4 Status Encode	Bit 3 Status Encode	Bit 2 Status Encode
000 (Default)	Tag	Invalid	Unlocked
001	Tag	Fault	Unlocked
010	Format	Invalid	Unlocked
011	Format	Fault	Unlocked
100	Fault	Invalid	Unlocked
101	Format	Tag	Unlocked or invalid
110	Format	Tag	Unlocked or fault
111	Unused	Unused	Unused

UTS READBACK FIFO

The UTS system uses a FIFO to collect time code samples (up to 18) from all enabled UTS channels. The FIFO allows queuing of time codes arriving from the various UTS channels for readback by the user via the register map. Each FIFO sample consists of the following:

- 96-bit time code
- 3-bit encoded status code
- 5-bit source code

Time Code

The 96-bit time code appears in Register 0x0E2F to Register 0x0E3A, which adheres to the same functionality as Register 0x0F0A to Register 0x0F15 described in the Synchronization Time subsection of the Common Clock Synchronizer (CCS) section. Note that interpretation of the time code requires knowledge of the underlying format (fractional seconds or PTP).

There are two ways to discern the format of a time code sample. One way is to use the source code information (see the Source Code section) to determine which UTS is the source of the time code sample and then read the state of the format bit in the appropriate UTS Channel x control register (where x is 0 to 8, integer). Another way is to use one of the seven predefined encoded status patterns that includes the format status in Table 43. In this method, the encoded status code (see the Encoded Status section) contains the format information for the time code sample.

Source Code

A 5-bit source code bit field accompanies each time code. The source code appears in Bits[4:0] of Register 0x0E2E. The source code value for a given time code sample indicates the time stamp source shown in Table 42. The source code that accompanies a time code is identical to the source code assignment of the UTS channel associated with the time code (see the UTS Channel Time Stamp Source section).

The source code does not indicate the UTS channel associated with the time code, but the original source of the time stamp.

Encoded Status

A 3-bit encoded status flags bit field (Bits[7:5] of Register 0x0E2E) accompanies each time code. The meaning of each bit depends on the flags bit field of the UTS channel associated with the time code sample in accordance with Table 44.

To determine the meaning of each of the three bits, the user must be aware of which UTS channel provided the time code (see the Source Code section) and the state of the 3-bit flags bit field associated with that UTS channel (see the UTS Channel Status Flag Assignment section).

Table 44. UTS FIFO Readback Encoded Status Bits

Bits[4:2][1]	Bit 7 Encoded Status	Bit 6 Encoded Status	Bit 5 Encoded Status
000	Tag	Invalid	Unlocked
001	Tag	Fault	Unlocked
010	Format	Invalid	Unlocked
011	Format	Fault	Unlocked
100	Fault	Invalid	Unlocked
101	Format	Tag	Unlocked or invalid
110	Format	Tag	Unlocked or fault
111	Unused	Unused	Unused

[1] Bits[4:2] in the UTS Channel x control register associated with the time code.

UTS FIFO Status

Because the FIFO effectively works in the background collecting UTS time codes, the FIFO provides the user with a continuous status of the current state. The status information is essential to keep the user aware of how frequently to read FIFO samples or if sample data is lost due to an overflow condition. There are two categories of status information: FIFO overfill (a single status bit) and FIFO count (7-bit field).

FIFO Overfill

The FIFO overfill bit in Bit 7 of Register 0x0E2D indicates that the FIFO has exceeded the capacity, where Logic 1 indicates overfill. When the FIFO asserts the overfill bit, any time codes arriving at the FIFO input are lost.

The state of the FIFO overfill bit appears as part of the IRQ mechanism (see the Interrupt Request (IRQ) section) via Bit 7 of Register 0x301C. The overfill status of the FIFO is also available via an appropriately configured Mx status pin (see the Status and Control Pins section).

The FIFO overfill bit and Mx pin status (if configured) clear when the user reads a sample from the FIFO.

FIFO Count

The FIFO provides sample count information via the 7-bit FIFO status count bit field in Bits[6:0] of Register 0x0E2D. During normal operation (that is, overfill = 0), the FIFO status count bit field value indicates the number of time code samples currently in the FIFO. The user has access to the not empty status of the FIFO (that is, when the value of Bits[6:0] of Register 0x0E2D is greater than zero) via an appropriately configured Mx status pin. However, when FIFO overfill = 1, the FIFO status count bit field value indicates the number of lost time code samples. That is, the meaning of the value in the FIFO status count bit field changes based on the status of the FIFO overfill bit.

A transition of the FIFO status count bit field from a value of zero to a nonzero value is an indication that the FIFO is not empty. That is, the FIFO was empty (FIFO status count = 0) but now the FIFO has one (or more) time code samples (FIFO status count ≠ 0) available for reading by the user. The zero to nonzero transition appears as a status flag as part of the IRQ mechanism (see the Interrupt Request (IRQ) section) via Bit 5 of Register 0x301C. The not empty status of the FIFO is also available via an appropriately configured Mx status pin (see the Status and Control Pins section).

The Mx pin status indication of FIFO not empty remains asserted until the FIFO becomes empty (FIFO status count = 0).

Reading Back FIFO Data

The FIFO captures time code samples (up to 18) and status information associated with those time code samples from all the enabled UTS channels. Because each FIFO sample consists of a time code and status information, reading time code samples from the FIFO requires a sequence of operations. When the FIFO receives a new UTS sample, it flags an IRQ via Bit 6 of Register 0x301C.

First, the user asserts an IO update (Bit 0 in Register 0x000F), which latches the FIFO overfill and FIFO count status bits into the register map. If the user reads the previous time code from the register map prior to asserting an IO update, the FIFO shifts the contents and the next available FIFO sample overwrites the previous contents in the register map upon an IO update assertion. The FIFO does not shift the contents unless the user had previously read the time code from the register map prior to the assertion of an IO update. Thus, the user is free to read the current FIFO time code multiple times as long as the user does not assert an IO update.

Next, the user reads the FIFO status. If the FIFO count is zero, the FIFO is empty and there is no need to continue. If the FIFO count is nonzero and the FIFO overfill bit is Logic 0, the time code value in the register map is ready for reading. However, if the FIFO overfill bit is Logic 1, the FIFO is in an overflow condition and time code samples have been lost. In an overflow condition, the FIFO count value indicates how many samples were lost. The FIFO count value has a limit of 127. Therefore, there is no way to determine if more than 127 samples were lost. Upon reaching an overflow condition, the FIFO does not accept new time code samples. However, the contents remaining in the FIFO are valid but may be too old to be useful.

Finally, assuming the FIFO status is not empty and not in an overflow condition, the user may read the time code as well as the source code and encoded status bits associated with the current FIFO sample. The user can read the existing time code and status information repeatedly, provided the IO update remains unasserted. Assertion of an IO update overwrites the register map contents but only if the user reads the time code in the register map prior to asserting an IO update.

INVERSE USER TIME STAMPER (IUTS)

OVERVIEW

The AD9546 has two IUTSs: IUTS 0 and IUTS 1 (see Figure 54). As the name implies, an IUTS performs the inverse function of a UTS: accepting time codes based on the common time scale provided by the user via the register map and converting those time codes to internal device time stamps based on the local time scale (see the Common Clock Synchronizer (CCS) section). Because the IUTSs rely on the common time scale (which originates with the CCS), the status of the CCS matters for the proper function of an IUTS. As such, each IUTS is inoperative unless the CCS indicates ready status.

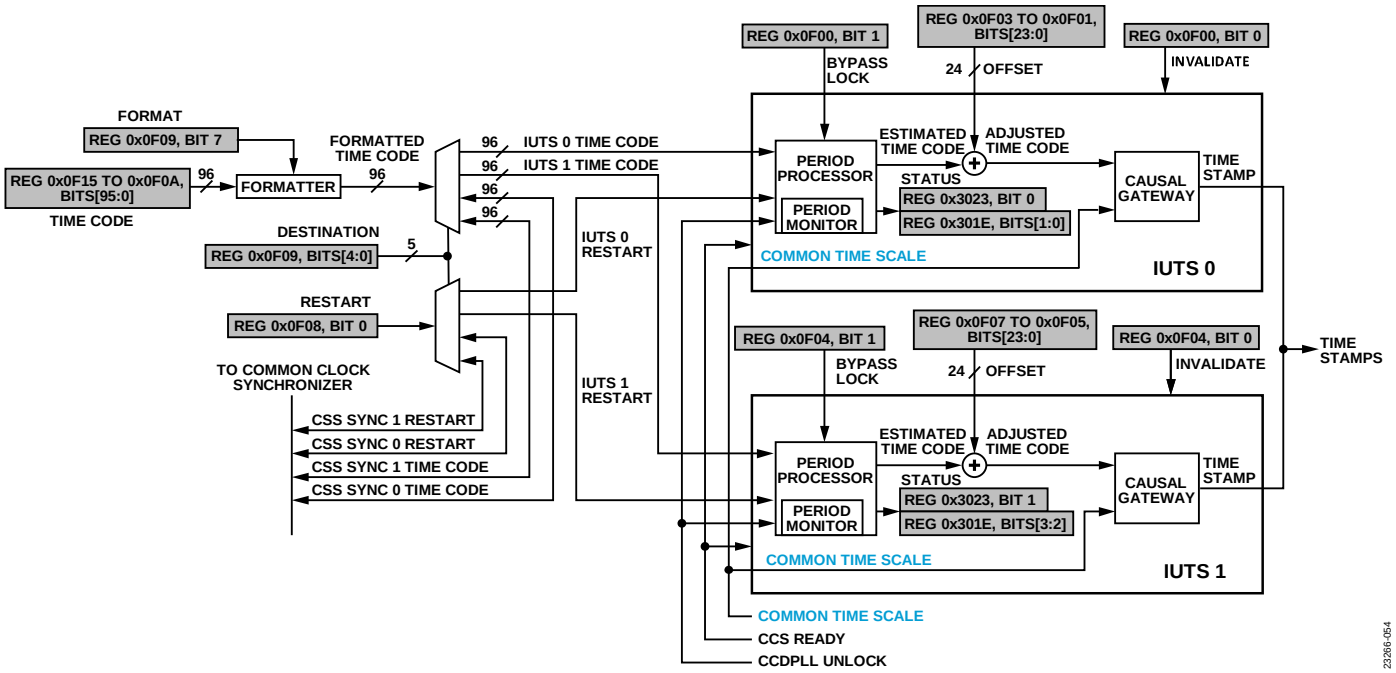
The intended operation of an IUTS is to translate a digitized clock signal (provided by the user in the form of a periodic sequence of time codes written to the register map) to an internal digitized clock signal for use by other resources within the AD9546 (for example, as the input to one of the DPLL channels). In a real-world, typical digitized clocking application, the user reads time codes from a UTS and delivers those time codes to an IUTS, which constitutes the transport of a digitized clock signal (see Figure 49).

In operation, the user writes a time code to an IUTS and asserts an IO update. The writing of a time code and subsequent assertion of an IO update causes the IUTS to process the new time code. Repeatedly carrying out the process of writing a time code and asserting an IO update effectively constitutes a digitized clock signal.

The IUTS computes the period of successive time codes via the period processor, where the current time code minus the previous time code constitutes an instantaneous period sample. Because period samples may exhibit jitter from sample to sample, the period processor employs an averaging mechanism to filter out period jitter from the underlying digitized clock signal. The averaging mechanism allows the period processor to provide a reliable estimate of the next time code in the sequence, despite the presence of period jitter on the incoming sequence of time codes.

The IUTS does not produce an output time stamp from the estimated time code until the time indicated by the common time scale is greater than or equal to the value of the estimated time code. That is, a causal gateway prevents estimated time codes from appearing as output time stamps until they are current (in terms of the common time scale).

The period processor of the IUTS includes a period monitor that observes the period of each time code sample processed. If a period sample deviates by more than about 1.5625%, the IUTS that processed the sample indicates an invalid status (see the IUTS Status section).



NOTES
1. A RANGE OF BITS USES A COLON SEPARATOR

Figure 54. IUTS Block Diagram

PARAMETER DESTINATION SELECTION

IUTS 0 and IUTS 1 each require the following two parameters from the register map as part of normal operation: time code and restart.

The register map has one instance of the two parameters which the user designates for a specific IUTS via the 5-bit unsigned destination bit field in Bits[4:0] of Register 0x0F09. The destination bits select the desired path of the two parameters to the target resource (via the two 1 to 4 demuxes shown in Figure 54) according to Table 45. The target resources include both IUTSs as well as the CCS. For destination bit field values of 30 and 31, see the Synchronization Time section of the Common Clock Synchronizer (CCS) section for details.

Table 45. Destination Selection

Destination Bit Field Value (Decimal)	Target Resource
0 to 12	Reserved
13	IUTS 0
14	IUTS 1
15 to 29	Reserved
30	CCS Sync 0 time code
31	CCS Sync 1 time code

TIME CODE AND FORMAT

The time code and format functionality is identical to that described in the Synchronization Time section of the Common Clock Synchronizer (CCS) section. The only difference is that the user selects IUTS 0 or IUTS 1 as the target resource per Table 45 via Bits[4:0] of Register 0x0F09.

IUTS TIME OFFSET

Each IUTS allow the user to add a constant time offset value to each estimated time code generated by the IUTS period processor. The time offset programs as a 24-bit signed value in Register 0x0F01 to Register 0x0F03 for IUTS 0 and Register 0x0F05 to Register 0x0F07 for IUTS 1.

The time offset carries units of 2^{-48} sec, yielding a range of approximately ±29.8 ns. For example, to program a time offset of 5 ns (5×10^{-9} sec), first convert to the appropriate units and round the result to the nearest integer as follows:

$$5 \times 10^{-9} \times 2^{48} = 1{,}407{,}375$$

This value in 24-bit hexadecimal format is: 0x 15 798F.

IUTS STATUS

When the IUTS period processor establishes the period associated with successive time codes, it indicates a ready status (Logic 1) via Bits[1:0] of Register 0x3023, with Bit 1 and Bit 0 indicating the status of IUTS 1 and IUTS 0, respectively. However, the user can manually override the valid status of an IUTS (see the Declaring an IUTS Invalid section).

The user also has access to two IRQ status bits for each IUTS via Bits[3:0] of Register 0x301E. One bit of the pair indicates IUTS valid status, and the other bit indicates IUTS invalid status. IUTS valid status appears in Bit 0 and Bit 2 (for IUTS 0 and IUTS 1, respectively). IUTS invalid status appears in Bit 1 and Bit 3 (for IUTS 0 and IUTS 1, respectively).

The status bits exist as part of the IRQ mechanism (see the Interrupt Request (IRQ) section). IUTS status indication is also available via an appropriately configured Mx status pin (see the Status and Control Pins section).

An IUTS indicates invalid status under the following conditions:

- The user declares an IUTS as invalid (see the Declaring an IUTS section)
- The common clock DPLL indicates unlocked (unless the user activates the bypass lock option (see the IUTS Bypass Lock Option section)
- A restart (see the IUTS Restart section) accompanies an input time code
- An input time code arrives when the IUTS is busy
- The IUTS period monitor detects a period discontinuity

IUTS RESTART

The IUTS period processor uses an internal algorithm for filtering period jitter that may be present on the incoming sequence of time codes. The algorithm gradually reduces the filtering bandwidth as it collects period samples over time. In some cases, there may be a need to restart the filtering process (such as the user switching the IUTS to a new time code source, for example).

To restart the IUTS, use the restart bit in Bit 0 of Register 0x0F08. Logic 0 (default) is normal operation, whereas Logic 1 forces the IUTS to restart.

IUTS BYPASS LOCK OPTION

Normally, each IUTS requires the common clock DPLL to indicate locked status to processes time codes. The user, however, can remove this constraint for each IUTS independently via Bit 1 of Register 0x0F00 for IUTS 0 and Register 0x0F04 for IUTS 1. Logic 0 (default) means the IUTS is inoperative when the common clock DPLL indicates an unlock condition. Logic 1 causes the IUTS to disregard the lock status of the common clock DPLL.

DECLARING AN IUTS INVALID

Because the output of an IUTS is a digitized clock signal available to other digitized clocking resources, the user may need to declare an IUTS as an invalid clock source. For example, the user may be aware of a problem with the source of input time codes to the IUTS that renders the IUTS output unsuitable as a digitized clock source to downstream resources. To handle such conditions, each IUTS allows manual invalidation of an IUTS as a time stamp source.

To invalidate an IUTS, use Bit 0 of Register 0x0F00 for IUTS 0 and Register 0x0F04 for IUTS 1. Logic 0 (default) invalidates the IUTS, whereas Logic 1 allows normal IUTS operation. Note that manual invalidation of an IUTS causes the IUTS to indicate an invalid status (see the IUTS Status section).

ANALOG CLOCK LOOPBACK

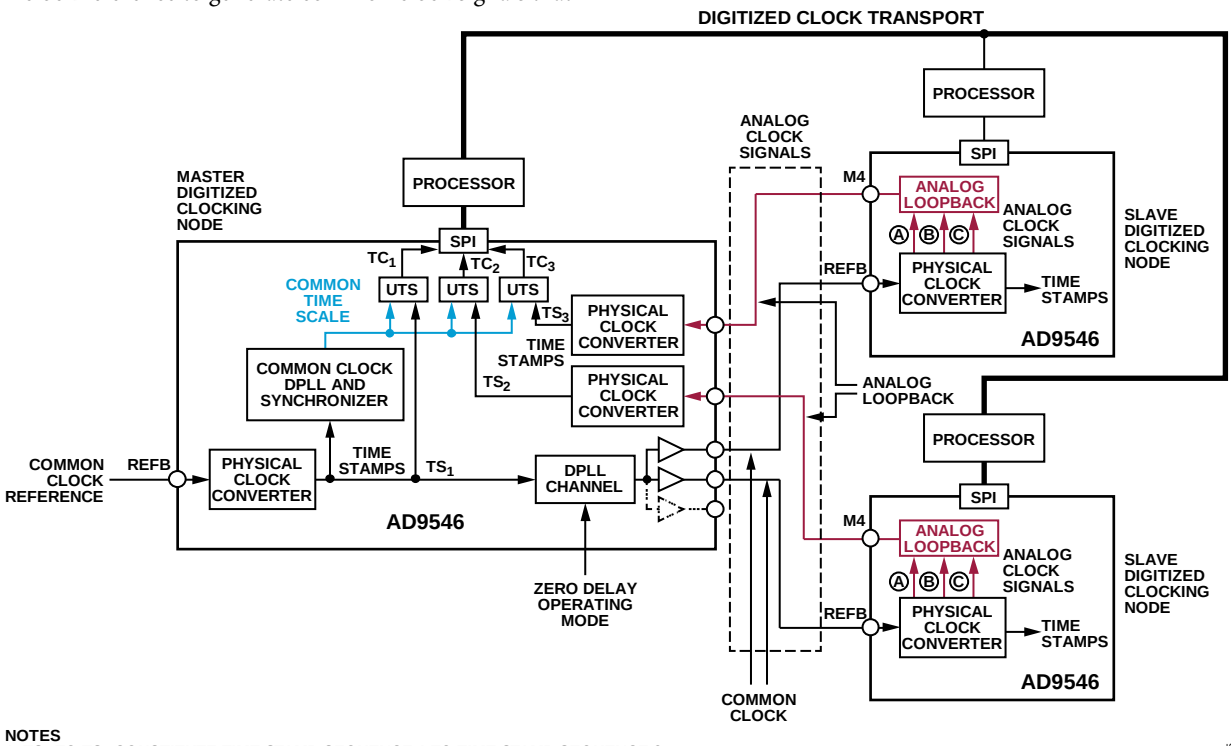
OVERVIEW

Analog loopback provides a means for designers of distributed clock systems to loop an analog clock signal back to measure the round trip delay. Knowledge of the round trip delay allows the user to estimate the timing error associated with physical connections (cables and circuit board traces, for example) and compensate for those errors in downstream nodes. The net effect is improved synchronization between nodes in a distributed clock system.

The diagram of Figure 55 shows an example of a distributed digitized clocking system comprising three AD9546 devices. On the left is the master timing node, and on the right are the slave timing nodes. Each node has a processor with a SPI connection to the AD9546 associated with the node and a digital bus interconnection between the processors of all nodes. The main purpose of the digital bus is the transport of digitized clocks between nodes. However, the digital bus enables the transport of other data between nodes, allowing node processors to share data (for example, time offset information for correcting delays between nodes).

The primary timing signal for the system is a common clock reference, which drives the REFB input of the master node. The master node generates its common time scale from the common clock reference. The master node also uses the common clock reference to generate common clock signals that route to the slave nodes. The common clocks result from routing the common clock reference through the master DPLL channel, preferably operating in zero delay mode to minimize phase (delay) offset. The slave nodes use the common clock to generate their own common time scale and output clock signals. The goal of the distributed clock system is to have all output clock signals of the slave nodes synchronous to the common clock of the slave node, which is synchronous to the common clock reference of the master.

A significant challenge in the implementation of a high precision distributed clock system is accounting for the time delay of the common clock as it propagates to each slave node. The delay arises from the physical interconnects (cables, for example) as well as active components (drivers and buffers, for example) in the signal path. Without knowledge of the amount of delay associated with each master to slave common clock path, the delay constitutes a timing error that degrades the precision of synchronization between nodes. To help remedy delay errors, the AD9546 offers high resolution time measurement capability (the UTS system) together with an analog loopback feature to provide a mechanism to quantify timing errors in the signal path. By quantifying the time errors, the user can compensate for those errors by inserting corrective time offsets via the various offset controls provided by the AD9546.



NOTES
1. TS_1 TO TS_3 CONSTITUTE TIME STAMP SEQUENCE 1 TO TIME STAMP SEQUENCE 3.
2. TC_1 TO TC_3 CONSTITUTE TIME CODE SEQUENCE 1 TO TIME CODE SEQUENCE 3.

23266-055

Figure 55. Distributed Clock System with Analog Loopback

ANALOG LOOPBACK FEATURE

Figure 55 shows the analog loopback feature of the AD9546 color coded in red. Figure 56 shows a detailed diagram of the internal structure of the analog loopback feature. The analog loopback feature uses the REFB and REFBB pins as the analog loopback input and the M4 pin as the analog loopback output. The REFB and REFBB inputs both provide loopback paths at the output of the single-ended receiver (the A path), the input to the TDC (the B path) and a replica of the reference demodulator synchronization signal (the C path). Note that the A path does not provide access to the output of the differential receiver.

The user selects the desired loopback path via Bits[2:0] of Register 0x2D02 per Table 46. The user selects the loopback path based on the operating configuration of the REFB or REFBB input.

Table 46. Analog Loopback Path Selection

Bits[2:0] (Decimal)	Loopback Path
0 (Default)	Disable loopback
1	REFB receiver output to M4
2	REFBB receiver output to M4
3	REFB divider output to M4
4	REFBB divider output to M4
5	REFB demodulator synchronization output to M4
6	REFBB demodulator synchronization output to M4
7	Reserved

The selection of a loopback path (1 decimal through 6 decimal in Table 46) substitutes the looped back analog clock signal for the normal M4 status selection associated with Register 0x0186 (or Register 0x0106; see the Status and Control Pins section for details). Furthermore, selecting a loopback path does not automatically complete the loopback path. To complete the loopback path, the user must enable the M4 output driver by programming Register 0x0106, Bit 7 = 1.

MEASURING ROUND TRIP DELAY

The analog loopback feature enables the measurement of a round trip delay. The main goal of a round trip delay measurement is to quantify the interconnect delay, which tends to be the dominant time error contributor in a distributed clock system. A round trip delay measurement assumes the following three conditions:

- The DPLL channel operates in zero delay mode
- The magnitude of the round trip delay is less than the period of consecutive time stamp events associated with TS_1, TS_2, or TS_3 (see Figure 55)
- The frequency of the common clocks is the same as the frequency of the common clock reference

Operating in zero delay mode is a not a strict requirement, but greatly simplifies the round trip delay measurement process. The second and third conditions are not strict requirements, but if violated, increase the complexity of the round trip delay computation to a level beyond the scope of this data sheet.

To measure the round trip delay, the user programs one of the UTS units of the master node to accept time stamps from the physical clock converter associated with the common clock reference, TS_1 (refer to Figure 55). The UTS converts the TS_1 time stamps to TC_1 time codes. The user also programs a second UTS unit to accept time stamps from the physical clock converter associated with the desired analog loop back path. As an example, assume the loopback path is from the lower slave node in Figure 55. In this case, the UTS converts the TS_2 time stamps to TC_2 time codes.

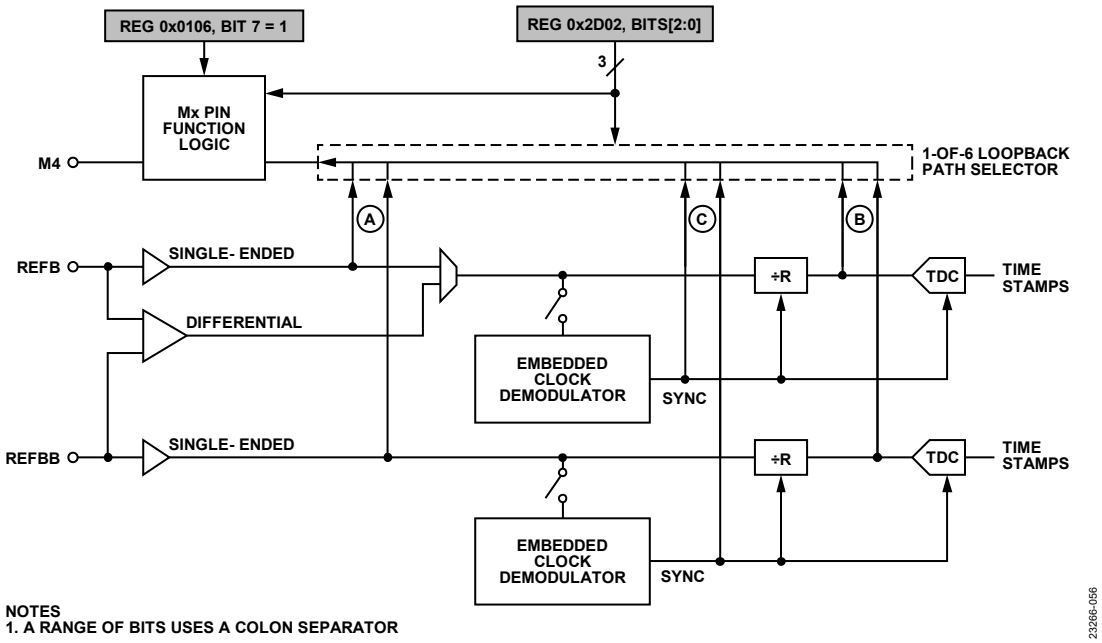


Figure 56. Analog Loopback Detail

The TC_1 and TC_2 time codes occur sequentially in pairs: $TC_1[x]$ and $TC_2[x]$, $TC_1[x + 1]$ and $TC_2[x + 1]$, and so on. Each time code pair provides a round trip delay measurement, RTD, given by

$$RTD[x] = TC_2[x] - TC_1[x]$$

Ideally, each RTD[x] is the same value, but in practice, the RTD[x] values vary slightly due to various noise contributors. Generally, the user takes an average of several RTD[x] values to yield an improved estimate of round trip delay. The round trip delay measurement includes all delays in the loopback path: interconnections and active elements (like buffers and other semiconductor components). However, Table 29 provides static delay values for sections of the internal loopback paths, which the user can subtract from the round trip delay measurement to account for the delay associated with the internal paths of the device. The resulting value, RTD_{CORR}, constitutes the corrected round trip delay, which consists of the total interconnect delay (forward and reverse paths) along with delay contributions of any other components in the path. The small error due to the difference between the mean specified component delays and the actual component delays also exists. Assuming the interconnect delay dominates, an approximation of the forward (or reverse) path interconnect delay, Δt, is

$$\Delta t = \frac{1}{2} \times RTD_{CORR}$$

With a value for RTD_{CORR} and knowledge of the mean component delays in the signal path, the user can program corrective offsets in the appropriate nodes to counteract the path delay.

EXCESSIVE ROUND TRIP DELAY

In some applications, interconnect lengths may result in a round trip delay that exceeds the period of the underlying clock period. Such a situation breaks the symmetry of the TC_1 and TC_2 time code pairs. Techniques exist to overcome excessive round trip delay, but such is beyond the scope of this data sheet.

SYSTEM CLOCK PLL

SYSTEM CLOCK PLL OVERVIEW

The system clock PLL (see Figure 57) comprises an Integer N frequency synthesizer with a fully integrated loop filter and VCO. The VCO output is the AD9546 system clock with a frequency range of 2250 MHz to 2415 MHz. The XOA and XOB pins constitute the input to the system clock PLL to which a user connects a clock source or crystal resonator.

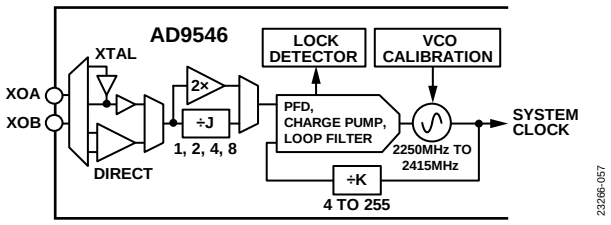


Figure 57. System Clock PLL Block Diagram

SYSTEM CLOCK INPUT FREQUENCY DECLARATION

Proper operation of the AD9546 requires the user to declare the input reference frequency to the system clock PLL by programming the 40-bit unsigned integer in Bits[39:0] of Register 0x0206 to Register 0x0202. The programmed value constitutes the nominal frequency, in units of mHz, applied to the XOA and XOB pins. The AD9546 evaluation software frequency planning wizard calculates this value for the user.

SYSTEM CLOCK SOURCE

The XOA and XOB pins serve as the input connection to the system clock PLL, giving the user access to a crystal path (see Figure 30) or a direct path (see Figure 31). Path selection is via Register 0x0201, Bit 3, where a Logic 0 (default) selects the direct path and Logic 1 selects the crystal path. The optimal reference source for the system clock input is a crystal resonator in the 50 MHz range or an ac-coupled square wave source (single-ended or differential) with an 800 mV p-p amplitude.

Crystal Path

Select the crystal path by programming Register 0x0201, Bit 3, to Logic 1. Refer to Figure 30 for connecting a crystal resonator to the XOA and XOB pins. The C_{TUNE} capacitors shown in Figure 30 relate to C_{LOAD} and C_{STRAY} as follows:

$$C_{TUNE} = 2 \times (C_{LOAD} - C_{STRAY})$$

where:
C_{TUNE} is a tuning capacitor.
C_{LOAD} is the load capacitance, per the specification of the crystal manufacturer.
C_{STRAY} is any additional parasitic capacitance.

As an example, when C_{LOAD} = 10 pF and C_{STRAY} = 2 pF to 5 pF, the value of C_{TUNE} is approximately 15 pF.

The crystal path supports crystal resonators in the 25 MHz to 80 MHz frequency range. An internal maintaining amplifier provides the negative resistance required to induce oscillation.

The internal amplifier expects an AT cut, fundamental mode crystal with a maximum motional resistance of 100 Ω for crystals up to 52 MHz, and 50 Ω for crystals up to 80 MHz. The following crystals, listed in alphabetical order, may meet these criteria:

- AVX/Kyocera CX3225SB
- ECS, Inc. ECX-32
- Epson/Toyocom TSX-3225
- Fox FX3225BS
- NDK NX3225SA
- Siward SX-3225
- Suntsu SCM10B48-49.152 MHz

Analog Devices, Inc., does not guarantee the operation of the AD9546 with these crystals, nor does Analog Devices endorse one crystal supplier over another. The AD9546 reference design uses a readily available high performance 49.152 MHz crystal with low spurious content.

Crystal resonators are subject to the specified maximum power dissipation requirement of the manufacturer. As such, some crystal resonators may require the use of a resistor to absorb some of the power delivered by the maintaining amplifier to satisfy the maximum power dissipation requirement of the crystal resonator. Figure 58 shows the proper way to connect the power limiting resistor, R_{LIMIT}, to achieve optimal performance.

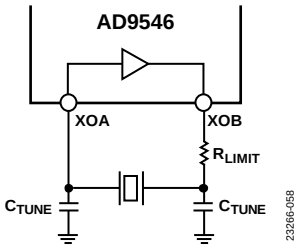


Figure 58. Power Limiting Resistor Connection

Frequency Doubler (2× Multiplier)

When a crystal resonator is the frequency source to the system clock, the user has the option of doubling the reference frequency via Bit 0 of Register 0x0201. Programming a Logic 1 doubles the input reference frequency to the system clock PLL, which improves the in-band noise of the PLL by 2 dB to 3 dB.

The frequency doubler cannot be used with the direct path input of the system clock PLL.

Direct Path

Select the direct path by programming Register 0x0201, Bit 3, to Logic 0. Refer to Figure 31 for connecting a TCXO or OCXO with a 3.3 V output to the XOA and XOB pins.

The direct path supports low frequency LVPECL, LVDS, CMOS, or sinusoidal clock sources as a reference to the system clock PLL. For a sinusoidal source, however, it is best to use a frequency of 50 MHz or greater. The low slew rate of lower frequency sinusoids tends to yield nonoptimal noise performance.

The direct path has a differential receiver with a self bias of 0.75 V dc. Generally, the presence of the bias voltage necessitates the use of ac coupling between the external source and the XOA and XOB pins. Furthermore, when using a 3.3 V CMOS oscillator as the system clock PLL reference source, in addition to ac coupling, it is important to use a voltage divider to reduce the 3.3 V swing to a maximum of 1.14 V (note that the optimal voltage swing is 800 mV p-p).

Degraded phase noise performance typically occurs when the frequency at the input of the PFD of the system clock PLL is less than 50 MHz. For optimal phase noise performance, use the quartz crystal resonator path with a crystal resonator frequency ≥ 50 MHz, the frequency doubler enabled, and the device configured to use system clock Compensation Method 3 (see the System Clock Compensation section).

PRESCALE DIVIDER

The system clock PLL includes an input prescale divider programmable for divide ratios of 1 (default), 2, 4, or 8. The purpose of the divider provides flexible frequency planning for mitigating potential spurs in the output clock signals of the AD9546. The user selects the divide ratio via Register 0x0201, Bits[2:1]. The corresponding divide value is 2^J, where J is the decimal value of the 2-bit number comprising Register 0x0201, Bits[2:1].

For example, a value of 10 (binary) in Register 0x0201 Bits[2:1], means J = 2 (decimal), making the divide ratio = 4 ($2^J = 2^2 = 4$).

FEEDBACK DIVIDER

The output of the system clock PLL constitutes the system clock frequency, f_S. The system clock frequency depends on the value of the feedback divider (K in Figure 57). The feedback divide ratio has a range of 4 to 255, which the user programs via Register 0x0200, Bits[7:0]. The programmed register value is the divide ratio. For example, a programmed value of 100 (0x64 hexadecimal) yields a divide ratio of 100.

SYSTEM CLOCK PLL OUTPUT FREQUENCY

Calculate the system clock frequency, f_S, as follows:

$$\frac{K}{J} f_S = f_{OSC} \times \frac{K}{J}$$

where:
f_{OSC} is the input frequency.
K is the feedback divide ratio.
J is the input divide ratio. J = ½ when using the 2× frequency multiplier.

The user must choose f_{OSC}, K, and J such that f_S satisfies the VCO range of 2250 MHz to 2415 MHz.

SYSTEM CLOCK PLL LOCK DETECTOR

The system clock PLL features a simple lock detector that compares the time difference between the reference and feedback clock edges. The user can check the status of the lock detector via Register 0x3001, Bit 0, where Logic 1 indicates locked and Logic 0 indicates unlocked.

SYSTEM CLOCK STABILITY TIMER

Because time processing blocks within the AD9546 depend on the system clock generating a stable frequency, the system clock PLL provides an indication of the status. The status of the system clock PLL is available to the user as well as directly to certain internal time keeping blocks.

At initial power-up, the system clock status is unknown and reported as being unstable. However, after the user programs the system clock registers and the system clock PLL VCO calibrates, the system clock PLL locks shortly thereafter.

Even though the system clock PLL may indicate a lock status, the user can introduce an additional holdoff period, in units of milliseconds (10^{-3} sec), via an unsigned 20-bit integer in Bits[19:0] of Register 0x0209 to Register 0x0207 (Register 0x0209, Bit 3 denotes the MSB). There are two special values: 0 and 1,048,575. A value of 0 causes the system clock to indicate an unstable status regardless of the actual state of the system clock (useful for debugging purposes, because the user can force an unstable status indication even though the PLL is stable). A value of 1,048,575 is invalid because it makes Register 0x3001, Bit 1 undefined when the system clock PLL is unlocked.

SYSTEM CLOCK CALIBRATION

The VCO in the system clock PLL requires a calibration sequence. At power-up, part of the initialization sequence of the device includes system clock calibration. In general, the user is not required to calibrate the system clock manually.

However, changing any of the system clock parameters that result in a new VCO frequency requires a manual system clock calibration via Register 0x2000, Bit 2, is not an autoclearing bit. Therefore, the full programming sequence is to write Logic 1, assert an IO update, write Logic 0, and assert an IO update. Furthermore, whenever the system clock PLL is in the process

of calibrating, status Bit 2 of Register 0x3001 indicates busy (Logic 1).

Bit 1 of Register 0x2000 provides an alternate means to calibrate the system clock PLL (this bit also calibrates APLL0 and APLL1).

Although the AD9546 has a variety of power-down modes, including a specific system clock power-down, it is not necessary to calibrate the system clock when recovering from any of the power-down modes. However, recovering from a power-down of the system clock necessitates calibration of the APLLs due to the loss of the system clock during its power-down.

SYSTEM CLOCK STABILITY COMPENSATION

An application requiring low DPLL loop bandwidth necessitates the use of a very stable system clock source. Using a relatively unstable system clock source like a crystal resonator or crystal oscillator, for example, in conjunction with the DPLL having a very narrow loop bandwidth (less than approximately 50 Hz) may cause the DPLL to unlock or experience intermittent unlock events. The typical solution requires the use of a very stable system clock source (an OCXO, for example). An alternate solution uses the integrated system clock compensation of the AD9546, which enables the use of a crystal resonator as the system clock source. The basic concept is the connection of a crystal resonator (25 MHz to 80 MHz) to the XOA and XOB pins per Figure 30, which capitalizes on the exceptional phase noise performance of the crystal.

However, the crystal resonator lacks the desired stability. Instead, the stability arises via the connection of a highly stable source (for example, TCXO, OCXO, or GPS) to an unused REFx or Mx input. The stable source then serves as the reference frequency to the system clock compensation feature of the device (see the System Clock Compensation section). The system clock compensation feature allows the device to benefit from both the stability of the highly stable source and the exceptional phase noise performance of the crystal resonator. Thus, the system clock compensation feature provides a mechanism to employ a low loop bandwidth DPLL (as required by the application) even though the system clock source is a relatively unstable crystal resonator.

REFERENCE CLOCK INPUT RESOURCES

The AD9546 has eight reference clock inputs as shown in Figure 59. The eight inputs comprise two groups of four:

- Four primary reference inputs (REFA, REFAA, REFB, and REFBB)
- Four auxiliary reference inputs (auxiliary REF0, auxiliary REF1, auxiliary REF2, and auxiliary REF3)

The two groups are functionally identical except for the configuration of their input receivers.

The primary reference inputs have configurable input receivers (see the Reference Receivers section) and dedicated pins. The user can configure the REFA and REFAA inputs as a single differential receiver or as two independent, single-ended receivers. The REFB and REFBB inputs can be configured likewise.

The auxiliary reference clock inputs have nonconfigurable, single-ended CMOS receivers, and rather than having dedicated pins, they are accessible only through Mx Pins (see the Status and Control Pins section).

Each reference clock input has a dedicated reference monitor, reference divider (R divider), TDC, and reference demodulator.

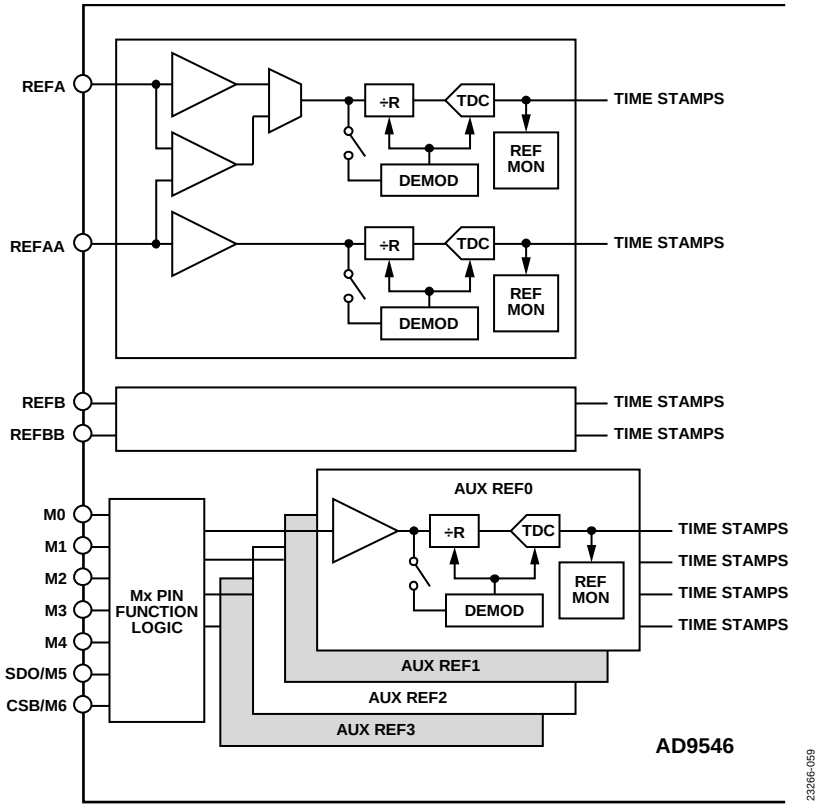


Figure 59. Reference Inputs Block Diagram

REFERENCE RECEIVERS

REFERENCE RECEIVERS OVERVIEW

The four dedicated reference clock input pins (REFA, REFAA, REFB, and REFBB) support up to four reference clock signals. Two of the four pins designate the A reference input path, and the other two pins designate the B reference input path. Both the A and B reference inputs are configurable as either one differential input or as two single-ended inputs. Each A and B reference input path has a dedicated REFx input mode bit (where x is A or B) for selecting single-ended or differential modes in Bit 0 of Register 0x0300 for REFA and Register 0x0304 for REFB. Logic 0 (default) selects single-ended mode, and Logic 1 selects differential mode. To accommodate input signals with slow rising and falling edges, both the differential and single-ended input receivers employ hysteresis. Hysteresis also ensures that a disconnected or floating input does not cause the receiver to oscillate.

The input receivers of the auxiliary references (accessible via appropriately configured Mx control pins) are not configurable. Instead, they have dc-coupled, single-ended, 1.8 V CMOS receivers.

SINGLE-ENDED MODE

When configured for single-ended operation, the input receivers accommodate either ac- or dc-coupled input signals via Bits[7:4] of Register 0x0300 and Register 0x0304 per Table 47. Bits[7:4] are only effective when single-ended mode is in effect per Bit 0 of Register 0x0300 and/or Register 0x0304.

The ac-coupled mode has an internal bias termination equivalent to a 0.6 V dc source with a 23.5 kΩ series resistance. The dc-coupled modes have a 47 kΩ resistor connected to ground. The internal pull-up mode has a 47 kΩ resistor connected to 1.2 V.

Single-Ended DC-Coupled Interface (1.2 V or 1.8 V CMOS)

Refer to Figure 32 for connecting a 1.2 V or 1.8 V CMOS driver to a reference input. Be sure to select the corresponding dc-coupled 1.2 V or 1.8 V CMOS mode per Table 47.

Single-Ended DC-Coupled Internal Pull-Up Resistor

Refer to Figure 34 for connecting an open-collector or open-drain driver to a reference input. Be sure to select the internal pull-up mode per Table 47, which provides an internal pull-up resistor (~47 kΩ) to 1.2 V.

Single-Ended AC-Coupled Interface

Refer to Figure 33 to ac couple a driver to a reference input. Be sure to select the ac-coupled interface mode per Table 47, which provides the appropriate internal bias network.

Table 47. Single-Ended Mode Bit Field Settings

Register	Bits[7:4][1]	Reference	Description
0x0300	XX00	REFA	AC-coupled 1.2 V interface (default)
	XX01	REFA	DC-coupled 1.2 V CMOS
	XX10	REFA	DC-coupled 1.8 V CMOS
	XX11	REFA	Internal pull-up resistor (disable pull-down resistor)
	00XX	REFAA	AC-coupled 1.2 V interface (default)
	01XX	REFAA	DC-coupled 1.2 V CMOS
	10XX	REFAA	DC-coupled 1.8 V CMOS
	11XX	REFAA	Internal pull-up resistor (disable pull-down resistor)
0x0304	XX00	REFB	AC-coupled 1.2 V interface (default)
	XX01	REFB	DC-coupled 1.2 V CMOS
	XX10	REFB	DC-coupled 1.8 V CMOS
	XX11	REFB	Internal pull-up resistor (disable pull-down resistor)
	00XX	REFBB	AC-coupled 1.2 V interface (default)
	01XX	REFBB	DC-coupled 1.2 V CMOS
	10XX	REFBB	DC-coupled 1.8 V CMOS
	11XX	REFBB	Internal pull-up resistor (disable pull-down resistor)

[1] X means don't care.

DIFFERENTIAL MODE

When configured for differential operation, the input receivers accommodate either ac-coupled or dc-coupled input signals per Table 48. Bits[3:2] are only effective when the corresponding differential mode is in effect.

Table 48. Differential Mode Settings

Register	Bits[3:2]	Reference	Description
0x0300	00	REFA, REFAA	AC-coupled (default)
	01	REFA, REFAA	DC-coupled
	10	REFA, REFAA	DC-coupled LVDS
	11	Unused	Unused
0x0304	00	REFB, REFBB	AC-coupled interface (default)
	01	REFB, REFBB	DC-coupled 1.2 V CMOS
	10	REFB, REFBB	DC-coupled 1.8 V CMOS
	11	Unused	Unused

The ac-coupled mode has an internal bias termination on each differential input pin, which has a Thevenin equivalent of a 0.6 V dc source in series with a 23.5 kΩ series. The dc-coupled mode expects the external driver to provide a 0.6 V common-mode bias. The dc-coupled LVDS mode expects a 1.2 V common-mode source with LVDS signal levels, as delivered by a standard LVDS driver.

For differential mode operating frequencies in excess of 10.24 MHz, use the ac-coupled differential mode and connect the reference input clock signal to the reference input pins through a series dc blocking capacitor. For differential mode operating frequencies < 10.24 MHz, use dc-coupled differential mode. For direct connection of a low frequency (<10.24 MHz) LVDS input reference source, use dc-coupled LVDS mode, which provides the necessary level shifting.

DC-Coupled LVDS (<10.24 MHz) Interface

Use Figure 35 to make a dc-coupled connection to a standard LVDS driver having a 1.2 V common-mode output bias. Be sure to select the dc-coupled LVDS mode per Table 48, which provides the appropriate internal level shifting for the input receiver. This configuration supports input frequencies up to 10.24 MHz.

Differential AC Coupling (>10.24 MHz)

For differential input signals in excess of 10.24 MHz, refer to Figure 36 to make an ac-coupled connection to an external driver. Be sure to select the ac-coupled mode per Table 48.

Differential DC Coupling (>10.24 MHz)

Use Figure 37 to make a dc-coupled connection to a differential driver having a 0.6 V common-mode output bias. Be sure to select the dc-coupled mode per Table 48.

REFERENCE DIVIDERS (R DIVIDERS)

Each reference input has a dedicated divider, R_x (where x differentiates between the eight reference inputs: A, AA, B, BB, or auxiliary REF0 to auxiliary REF3). The primary purpose of the R_x dividers is to reduce the input reference frequency (assuming it is greater than 200 kHz) to a value between 1 Hz and 200 kHz to satisfy the input frequency bounds of the TDC.

The user programs R_x via the 30-bit unsigned bit fields per Table 49. The divide ratio of R_x depends on the value, R, the user programs in the bit fields. The user must program the bit field with a value one less than the desired divide ratio. Thus, the R_x dividers provide divide ratios from 1 to 1,073,741,824.

For example, for an input reference frequency of 125 MHz and the desired R_x output frequency of 125 kHz, a divide ratio of 1000 is required. Thus, the required bit field value is 999 (0x000003E7 hexadecimal).

Table 49. Reference Divider Address Ranges

R_x Reference	Register Address Range
REFA	0x0400 to 0x0403
REFAA	0x0420 to 0x0423
REFB	0x0440 to 0x0443
REFBB	0x0460 to 0x0463
Auxiliary REF0	0x0480 to 0x0483
Auxiliary REF1	0x04A0 to 0x04A3
Auxiliary REF2	0x04C0 to 0x04C3
Auxiliary REF3	0x04E0 to 0x04E3

REFERENCE MONITOR

REFERENCE MONITOR OVERVIEW

The AD9546 has eight independent reference monitors, one for each reference input. Figure 60 shows the REFA input along with its associated reference monitor. The reference monitor diagram of Figure 60 applies identically for the REFAA, REFB, REFBB, auxiliary REF0, auxiliary REF1, auxiliary REF2, and auxiliary REF3 inputs by replacing all implied references to REFA with the other reference inputs.

The user has the option of powering down unused primary references via Register 0x2001, Bits[3:0] (see the Reference Clock Input Resources section for a definition of primary references).

Each reference input has a dedicated input divider (for example, R_A in Figure 60) with a programmable divide ratio (see the Reference Dividers (R Dividers) section). Because the reference monitor follows the R divider in the signal path, the reference monitor essentially undersamples the reference input clock signal for divide ratios greater than one.

Between the R divider and the reference monitor is a TDC (see the Time to Digital Converter (TDC) section). The TDC delivers digital (numeric) time stamps to the reference monitor, which processes the time stamps to determine the status of the input clock signal. Each reference TDC can connect to either DPLL input via the digital cross point mux (see Figure 1). Even when a reference does not connect to one of the DPLLs via the cross point mux, the corresponding reference monitor continues to observe the reference signal and generate status signals accordingly.

The reference monitor uses certain user-programmable parameters (see the Reference Monitor Controls section) to perform various monitoring functions. The reference monitor provides status indicators (see the Reference Monitor Status Indicators section) that allow the user to assess the quality of the reference input signal.

The AD9546 uses the status provided by the reference monitor to control certain internal functions (automatic reference switching, for example). The user has access to the status signals via the register map or as a physical signal via the Mx pins (see the Status and Control Pins section and Interrupt Request (IRQ) section for more details).

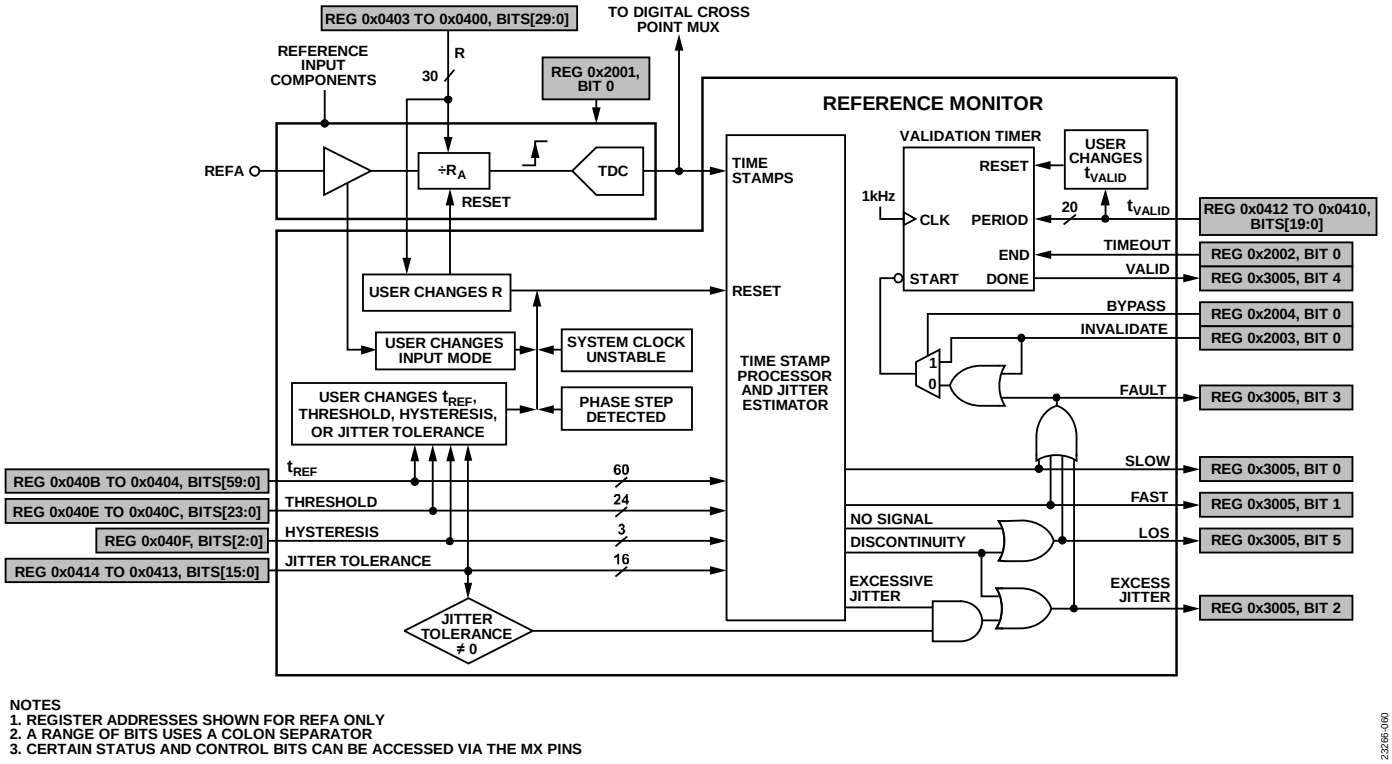


Figure 60. Reference Monitor

REFERENCE MONITOR BASE PERIOD

The reference monitor operates directly on TDC time stamps, which occur with each rising edge from the output of the R divider. Because time stamps result from rising edges at the TDC input, which occur at intervals of the input reference period multiplied by R (the reference divider ratio), the base period (t_{BASE}) of the reference monitor is R times the reference period.

REFERENCE MONITOR STATUS INDICATORS

The reference monitor associated with each reference provides six status indicators: valid, fault, LOS, slow, fast, and excess jitter. These indicators are associated with the D4, D3, D5, D0, D1, and D2 bits, respectively, but the register address depends on the reference per Table 50.

Table 50. Reference Monitor Status Register Address

Reference Input	Register Address
REFA	0x3005
REFAA	0x3006
REFB	0x3007
REFBB	0x3008
Auxiliary REF0	0x301F
Auxiliary REF1	0x3020
Auxiliary REF2	0x3021
Auxiliary REF3	0x3022

Valid

The valid status indicator is Logic 1 at the expiration of the optional validation time (see the Reference Validation section).

Fault

The fault status indicator is Logic 1 when the LOS, slow, fast, or excess jitter status is true.

Loss of Signal (LOS)

The LOS status indicator is Logic 1 if the reference monitor determines the input signal is absent, very slow (>1.15 × t_{BASE}) or a phase discontinuity occurred (see the Discontinuity Detection section).

Slow

The slow status indicator is Logic 1 if the period of the input signal is greater than the maximum period prescribed by the threshold setting (see the Reference Period Threshold section).

Fast

The fast status indicator is Logic 1 if the period of the input signal is less than the minimum period prescribed by the threshold setting (see the Reference Period Threshold section).

Excess Jitter

The excess jitter status indicator is Logic 1 if the reference monitor determines the input signal jitter exceeds the limits specified by jitter tolerance (see the Jitter Tolerance section) or a phase discontinuity occurred (see the Discontinuity Detection section).

REFERENCE MONITOR CONTROLS

The reference monitor requires certain advance information to function properly. The user provides this information by programming the appropriate parameters as described in the Reference Period (t_{REF}), Reference Period Threshold, Hysteresis, Jitter Tolerance, Validation Time (t_{VALID}), Timeout, Bypass, and Invalidate sections.

Reference Period (t_{REF})

Each reference input has a dedicated unsigned 60-bit bit field in the register address ranges shown in Table 51 for specifying t_{REF}. t_{REF} constitutes the nominal period of the associated input reference and represents time in units of attoseconds (10^{-18} sec) with an upper limit of ~1.15 sec.

For example, given the REFAA input has a nominal 2.048 MHz input signal. The corresponding period is $1/(2.048 \times 10^6)$ sec, which, when multiplied by 10^{18}, yields 488,281,250,000 attoseconds. Program Register 0x042B to Register 0x0424 with the 60-bit value corresponding to 488,281,250,000 attoseconds (0x 000 0071 AFD4 98D0 hexadecimal).

Table 51. Nominal Period Address Ranges

Reference Input	Register Address Range
REFA	0x0404 to 0x040B
REFAA	0x0424 to 0x042B
REFB	0x0444 to 0x044B
REFBB	0x0464 to 0x046B
Auxiliary REF0	0x0484 to 0x048B
Auxiliary REF1	0x04A4 to 0x04AB
Auxiliary REF2	0x04C4 to 0x04CB
Auxiliary REF3	0x04E4 to 0x04EB

Reference Period Threshold

The reference period threshold is the maximum relative deviation from t_{REF} the reference monitor uses to declare a nonfaulted reference as out of tolerance (that is, slow or fast). To specify the threshold parameter, each reference input has a dedicated register group comprising an unsigned 24-bit integer per the register address ranges shown in Table 52. The 24-bit number represents fractional units of parts per billion (ppb) up to a maximum of approximately 17 million ppb (1.7%), with the default setting yielding 100 ppm.

Table 52. Offset Limit Address Ranges

Reference Input	Register Address Range
REFA	0x040C to 0x040E
REFAA	0x042C to 0x042E
REFB	0x044C to 0x044E
REFBB	0x046C to 0x046E
Auxiliary REF0	0x048C to 0x048E
Auxiliary REF1	0x04AC to 0x04AE
Auxiliary REF2	0x04CC to 0x04CE
Auxiliary REF3	0x04EC to 0x04EE

Because the threshold parameter specifies a fractional period (δp), it relates to fractional frequency (δf) as follows:

$$\delta p = -\delta f/(1 + \delta f) \quad (1)$$

For example, consider the case where the expected maximum relative deviation of the input reference frequency for REFAA is 75×10^{-6} (75 ppm). That is, $\delta f = 75 \times 10^{-6}$.

From Equation 1, $\delta p = -74.994375 \times 10^{-6}$, or −74.994375 ppm, or −74994.375 ppb. Because the threshold parameter is an unsigned number with integer units of ppb, threshold = 74,994. Therefore, program Register 0x042C to Register 0x042E with the 24-bit value corresponding to 74,994 (0x 01 24F2 hexadecimal).

To properly verify small values of the threshold parameter, the reference monitor requires a very precise timing reference. The reference monitor uses time stamps from the TDC to assess the input signal parameters. Because the underlying precision of the TDC relates to the system clock, small threshold values require a suitably precise system clock or a suitably compensated system clock (see the System Clock Compensation section).

Hysteresis

The hysteresis parameter applies to a faulted reference, whereas the threshold parameter applies to a nonfaulted reference. The reference monitor makes fault and unfault decisions based on how reference period samples relate to t_{REF}, threshold, and hysteresis. To specify the hysteresis parameter for a particular reference, use Bits[2:0] of the register address associated with the reference per Table 53.

Table 53. Hysteresis Register Address

Reference Input	Register Address Range
REFA	0x040F
REFAA	0x042F
REFB	0x044F
REFBB	0x046F
Auxiliary REF0	0x048F
Auxiliary REF1	0x04AF
Auxiliary REF2	0x04CF
Auxiliary REF3	0x04EF

The programmed value of the hysteresis parameter defines a hysteresis scale factor (SF) per Table 54.

Table 54. Hysteresis Selection Table

Bits[2:0] (Decimal)	Scale Factor (SF)
0	0
1	0.03125
2	0.0625
3	0.125 (default)
4	0.25
5	0.5
6	0.75
7	0.875

The SF value constitutes a fractional portion of the value of threshold and defines the amount of hysteresis.

Hysteresis (ppb) = *Threshold* × *SF*

For example, given threshold = 74,994 ppb and SF = 0.0625, determine the hysteresis.

Hysteresis (ppb) = 74,994 ppb × 0.0625 = 4,687

The reference monitor compares reference period samples against the t_{REF}, threshold, and SF parameters to make fault and unfault decisions per Figure 61.

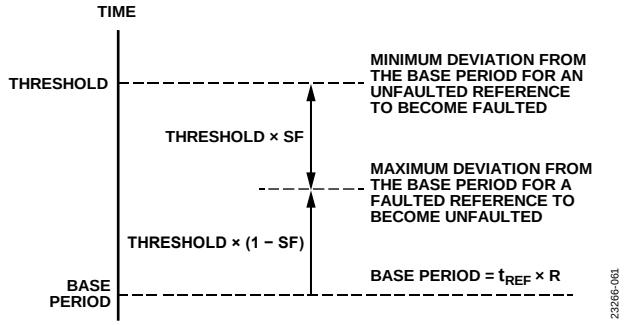


Figure 61. Reference Monitor Hysteresis

Jitter Tolerance

Use the jitter tolerance parameter to define the maximum rms jitter the reference monitor accepts before indicating an excessive jitter condition. Each reference input has a dedicated register group for entering jitter tolerance per the register address ranges shown in Table 55. The jitter tolerance parameter is an unsigned 16-bit integer with units of nanoseconds (10^{-9} sec) up to a maximum of approximately 65.5 µs.

Table 55. Jitter Tolerance Address Ranges

Reference Input	Register Address Range
REFA	0x0413 to 0x0414
REFAA	0x0433 to 0x0434
REFB	0x0453 to 0x0454
REFBB	0x0473 to 0x0474
Auxiliary REF0	0x0493 to 0x0494
Auxiliary REF1	0x04B3 to 0x04B4
Auxiliary REF2	0x04D3 to 0x04D4
Auxiliary REF3	0x04F3 to 0x04F4

For example, given an expected jitter limit of 75 ns rms, then jitter tolerance = 75 (0x004B hexadecimal).

When jitter tolerance = 0, the reference monitor does not include its estimation of jitter as part of the excess noise status decision. However, even when jitter tolerance = 0, the reference monitor continues to perform jitter estimation to continue making out of tolerance decisions.

Validation Time (t_{VALID})

In some applications, a reference signal must maintain its validity (that is, not faulted) for some minimum duration (t_{VALID}) before the system considers the reference valid. The reference monitor has a built in timer to handle such timed validation requirements. Each reference monitor has an associated unsigned 20-bit integer for assigning t_{VALID} per the register address ranges shown in Table 56.

Table 56. Validation Timer Address Ranges

Reference Input	Register Address Range
REFA	0x0410 to 0x0412
REFAA	0x0430 to 0x0432
REFB	0x0450 to 0x0452
REFBB	0x0470 to 0x0472
Auxiliary REF0	0x0490 to 0x0492
Auxiliary REF1	0x04B0 to 0x04B2
Auxiliary REF2	0x04D0 to 0x04D2
Auxiliary REF3	0x04F0 to 0x04F2

The programmed value of t_{VALID} carries units of milliseconds, up to a maximum of approximately 1048 sec (about 17.5 min). For example, a validation time of 10 min requires

$$t_{VALID} = (10 \text{ min}) \times (60 \text{ sec/min}) \times (1000 \text{ ms/sec})$$

$$= 600{,}000 \text{ ms (0x927C0 hexadecimal)}$$

The validation timer starts when the reference goes from a faulted condition to an unfaulted condition. When the validation timer expires (that is, in Figure 60, the done output of the validation timer = 1), then the reference exhibits valid status (valid = 1).

After the validation timer starts, if a faulted condition occurs, the validation timer restarts. This behavior ensures that reference validation occurs only after a reference maintains an unfaulted condition for the full duration of the validation timer.

Note that programming a new value for t_{VALID} causes the validation timer to reset. Resetting the validation timer invalidates the reference even if the reference was valid prior to changing t_{VALID}. However, even though the reference exhibits an invalid status when the validation timer is reset, the reference monitor continues to indicate the appropriate fault or unfaulted condition based on the quality of the reference input signal.

The user can force a reference to invalid status without affecting the normal operation of the reference monitor (see the Invalidate section).

Timeout

The user has the option to force the validation timer to immediately time out via Register 0x2002, Bits[7:0], where each bit in the register corresponds to a particular reference. Specifically, starting with D7 and ending with D0, the corresponding references are auxiliary REF3, auxiliary REF2, auxiliary REF1, auxiliary REF0, REFBB, REFB, REFAA, and REFA.

Programming timeout = 1 causes the validation timer to time out immediately, giving the reference valid status (valid = 1).

Bypass

In certain circumstances, a user may find it necessary to bypass the reference monitor control of the validation timer. To this end, each reference input has a dedicated control bit (bypass) via Register 0x2004, Bits[7:0], where each bit in the register corresponds to a particular reference. Specifically, starting with D7 and ending with D0, the corresponding references are auxiliary REF3, auxiliary REF2, auxiliary REF1, auxiliary REF0, REFBB, REFB, REFAA, and REFA.

The bypass function does not actually bypass the reference monitor itself, but only the behavior of valid status.

When a reference has bypass = 0 (default), the normal fault and unfault conditions prescribed by the reference monitor control the start of the validation timer, which controls the valid state. Per Figure 60, when bypass = 0, the user has the option to force an unfaulted reference to invalid status via the invalidate control mechanism (see the Invalidate section). However, the user cannot force a faulted reference to be valid via the invalidate control mechanism when bypass = 0.

When a reference has bypass = 1, starting the validation timer falls under user control via the invalidate control mechanism (see the Invalidate section). That is, programming invalidate = 0 is equivalent to the reference monitor exhibiting an unfaulted condition. Conversely, programming invalidate = 1 is equivalent to the reference monitor exhibiting a faulted condition. The validation timer responds accordingly to the imposed fault and unfault conditions.

Note that regardless of the state of the bypass or invalidate control bit, the fast, slow, LOS, and excess jitter status indicators of the reference monitor function normally and remain accessible to the user via their respective status registers (see Figure 60).

Invalidate

In certain circumstances, a user may find it necessary to force a reference from a valid state to an invalid state. To this end, each reference input has a dedicated control bit (invalidate) via Register 0x2003, Bits[7:0], where each bit in the register corresponds to a particular reference. Specifically, starting with D7 and ending with D0, the corresponding references are auxiliary REF3, auxiliary REF2, auxiliary REF1, auxiliary REF0, REFBB, REFB, REFAA, and REFA.

A scenario for using invalidate = 1 when bypass = 0 is when the reference signal must be traceable to a primary timing source, but the user has external knowledge that the source is no longer traceable. Under these circumstances, the reference signal may still satisfy the parameters for being in tolerance. Therefore, the reference monitor indicates the reference as unfaulted.

However, because the user knows the source is no longer traceable, the user can force the reference to be invalid by making invalidate = 1 (even though the reference monitor sees the reference as being in tolerance). The user can reinstate a valid status when conditions permit by making invalidate = 0. However, if the reference signal goes out of tolerance (as determined by the reference monitor) during the time that invalidate = 1, then when the user makes invalidate = 0, the reference remains invalid. That is, the user cannot force a reference with an invalid status to a valid status via the fault reference bit when bypass = 0.

A scenario for using bypass = 1 (that is bypassing reference monitor control of the validation timer) is when a reference signal is of the gapped clock variety. A reference signal employing a gapped clock is problematic for the reference monitor because it cannot reliably discern an in or out of tolerance condition. As such, an otherwise unfaulted reference can be deemed faulted from the point of view of the reference monitor. Under such conditions, the user can bypass reference monitor control of the validation timer (and thereby valid and invalid status) by making bypass = 1. With bypass = 1, the invalidate control bit allows the user to mimic the fault and unfault conditions normally generated by the reference monitor.

DISCONTINUITY DETECTION

The reference monitor detects the following discontinuities in the period of the reference signal:

- An absent reference signal or a reference period that deviates by more than 3.125% of the expected value.
- A reference period sample deviating by more than 64 times J_{EST} plus ½ of jitter tolerance, where J_{EST} is the current estimate of the jitter variance.
- Short term jitter exceeds long-term jitter by ~50%.

Detection of a discontinuity causes the discontinuity status signal (see Figure 60) to be Logic 1.

REFERENCE PERIOD JITTER ESTIMATION

Because the reference monitor uses numeric time stamps to measure the reference period, it can estimate the mean and variance of the reference signal as it observes period samples (that is, the difference between successive time stamps from the TDC). The mean is an estimate of the reference period, and the variance is an estimate of the jitter present on the reference signal. The reference monitor uses the mean and jitter variance estimate to make in tolerance or out of tolerance decisions by comparing a span of 16 times the current jitter variance estimate (±8σ) against the t_{REF}, threshold, and hysteresis parameters. This statistical approach allows the reference monitor to make tolerance decisions with a high degree of confidence in a nearly optimal minimum observation time.

REFERENCE MONITOR DECISION TIME

The reference period estimation algorithm takes into account the value of the threshold parameter specified by the user (see the Reference Monitor Controls section) and the actual jitter present on the reference signal. In general, less averaging required by the jitter estimator to bring the variance within a suitable range means a shorter decision making time. The key point is that both the value of the threshold parameter and the amount of jitter present on the input signal govern the decision time for the reference monitor to declare an out of tolerance condition. That is, the decision time is not deterministic.

Although jitter plays a role in the decision time of the reference monitor under normal operation, jitter has little effect on the decision time when the reference period is much greater than the base period or when the reference signal disappears completely. The reason is the reference monitor has advance knowledge of the expected period of the reference signal (namely, $R \times t_{REF}$). Thus, the reference monitor can rely on internal timing (rather than TDC time stamps) to keep watch for expected time stamps from the TDC. If the reference monitor fails to observe the arrival of a new time stamp after a period of $1.15 \times R \times t_{REF}$, the monitor declares an LOS.

Note that for an input signal that is slow (but not absent), after the first assertion of LOS status, the LOS status toggles on each subsequent occurrence of the expected period of the reference. Thus, the LOS bit is not a reliable indicator of the loss of the reference signal. Instead, use the fault status as an indicator of the loss of the signal.

REFERENCE VALIDATION

Some applications require a reference to be in an unfaulted condition for a prescribed period before exhibiting valid status. To accommodate these applications, the reference monitor includes a programmable validation timer. A reference transition from a faulted state to a nonfaulted state indicates a start event for the validation timer, which begins counting down. Upon expiration of the timer, valid = 1, which indicates the reference is available for use (see Figure 60).

It is important to note that the validation timer stops counting down and resets upon the occurrence of the faulted status (fault = 1). A subsequent unfault condition causes the timer to reinitiate a countdown from the programmed t_{VALID} value. Thus, the reference only attains valid status when the reference remains unfaulted for the full duration of the validation timer.

The user has the option to force the validation timer to jump to the end of its timing function by programming timeout = 1 (see the Reference Monitor Controls section and Figure 60). In this way, if a faulted reference has returned to a nonfaulted state and is awaiting validation, the user can override the timer if necessary, and immediately bring the reference to a valid status. Programming timeout = 1 has no effect on a faulted reference because a fault condition resets the validation timer.

REFERENCE MONITOR RESET

The reference monitor associated with a specific reference resets under certain conditions. A reset restarts the period and jitter estimation process for the reference monitor. The following list indicates the conditions that cause the reference monitor to reset (also shown in Figure 60):

- When the system clock indicates unstable status
- When the user changes the mode of the input receiver, the R divider value, the t_{REF} value, the Δt_{REF} value, the jitter tolerance value, or the hysteresis value
- Optionally, when the reference experiences a significant phase step as determined by the DPLL (assuming the user enables phase step detection via Bit 7 of Register 0x2105 and Register 0x2205).

The last item requires that the user program a nonzero phase step limit value. See the Phase Step Limit section for details regarding the phase step limit feature.

REFERENCE DEMODULATOR

REFERENCE DEMODULATOR OVERVIEW

The AD9546 has eight reference demodulators: one for each REFx input and one for each auxiliary REFx input (see the Reference Clock Input Resources section). The REFx receivers are configurable as single-ended or differential. The auxiliary REFx receivers are single-ended only (nonconfigurable).

When REFA is configured for differential operation, only the demodulator associated with REFA is available (the demodulator associated with REFAA is unused). The same applies for REFB.

When enabled, a demodulator recognizes modulated carrier cycles on its associated reference input clock (a modulated cycle is a phase deviation imposed on the falling edge of an input clock cycle). The demodulator is compatible with embedded clock signals generated by Analog Devices clock chips that support embedded output clock modulation (the AD9545, for example). For details on embedded clock modulation, see the Distribution Embedded Output Clock Modulation section.

Figure 62 shows a basic block diagram of the reference demodulator (legacy mode). The Letter X denotes a reference demodulator associated with a specific reference input, and M and N associate a specific register address with Demodulator X per the register legend in Figure 62.

Referring to Figure 62, the reference input clock signal passes through the reference receiver and drives the clock input of the reference divider as well as the demodulator input. When the input reference signal carries embedded clock modulation events, the demodulator provides a means of detecting them. On detection of a modulation event on its input clock, the demodulator synchronizes the associated reference divider and causes the associated reference TDC to tag the time stamp generated by the modulation event.

INTERACTION BETWEEN DEMODULATOR AND REFERENCE MONITOR

The reference monitor validates based on the input carrier signal. However, the first modulation event after enabling the reference demodulator causes a resynchronization of the R divider. The resynchronization of the R divider starts a new reference monitor validation sequence, which means the reference invalidates. The user has access to the resynchronization strobe via an appropriately configured Mx pin.

The resynchronization of the R divider is a singular event that happens when the demodulator must force the R divider to reset on grid (see the Demodulator Synchronization section regarding the modulation grid). Thus, after enabling the reference demodulator, to avoid the disruption of reference validation due to the resynchronization of the R divider, use a reference validation timer value greater than the period between modulation events (see the Validation Time (t_{VALID}) section).

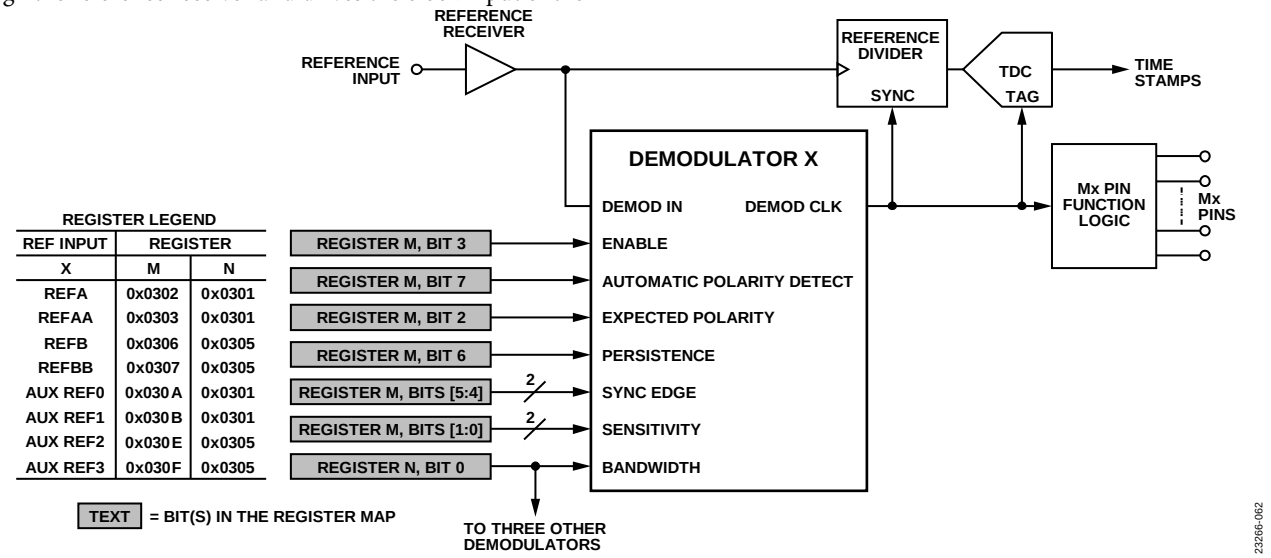


Figure 62. Reference Demodulator: Legacy Mode

MODULATED CYCLES AND MODULATION EVENTS

The reference demodulator detects the arrival of modulated carrier cycles. A modulated cycle spans one period of the input reference clock ($t_{CARRIER}$) and consists of a predefined phase deviation relative to the expected falling edge of the input clock. The definition of a modulated cycle is when the duration of the second half of a carrier cycle is lengthened or shortened by some period of time, Δt (see Figure 63). In general, Δt is constant for a given application and defines the modulation depth, where the magnitude of the modulation depth is always less than ½.

$$Modulation\ Depth = \frac{\Delta t}{t_{CARRIER}}$$

The reference demodulator uses two modulated cycles to identify a modulation event, although the second modulated cycle can be unmodulated in the case of unbalanced modulation (see Figure 63).

The demodulator performs no type of error detection on the modulated carrier cycles.

BALANCED AND UNBALANCED MODULATION

The demodulator recognizes two types of modulation events: unbalanced and balanced. Figure 63 shows generalized examples of the modulation event types and their polarity.

Unbalanced modulation occurs when the second modulated cycle of a modulation event is unmodulated. Note that unbalanced modulation only modulates the first modulated cycle of a modulation event. Therefore, an unbalanced modulation event shifts the dc offset of the carrier signal (relative to an unmodulated carrier).

Balanced modulation occurs when the second modulated cycle of a modulation event has the opposite polarity modulation of the first modulation event. For example, when the first modulation event has a period deviation of +Δt, then the second modulation event must have a period of –Δt (or vice versa). Because balanced modulation modulates the first and second modulated cycles of a modulation event in opposite fashion, balanced modulation events do not shift the dc balance of the carrier signal.

The first modulated cycle in a modulation event determines the polarity of the modulation event. A modulated cycle of –Δt equates to Logic 0, whereas a modulated cycle of +Δt equates to Logic 1.

ENABLING A DEMODULATOR

To enable a specific demodulator, use Bit 3 of Register M per the register legend in Figure 62. Logic 0 (default) disables the demodulator and holds it in a reset state. Logic 1 enables the demodulator to detect modulation events.

MODULATION POLARITY DETECTION

When Bit 7 = 1 of Register M per the register legend in Figure 62, the demodulator must automatically determine the polarity of the first modulated cycle of a modulation event (see Figure 63). However, the autodetect feature reduces the overall sensitivity of the demodulator.

The reference demodulator is most sensitive when it is aware of the modulation polarity in advance. The user can take advantage of the maximum sensitivity of the demodulator by programming Bit 2 in Register M per the register legend in Figure 62 (assuming Bit 7 = 0 in Register M per the register legend in Figure 62). The value of Bit 2 indicates the expected polarity of Δt in the first modulated cycle of a modulation event (see Figure 63). Logic 0 (default) indicates negative Δt, and Logic 1 indicates positive Δt.

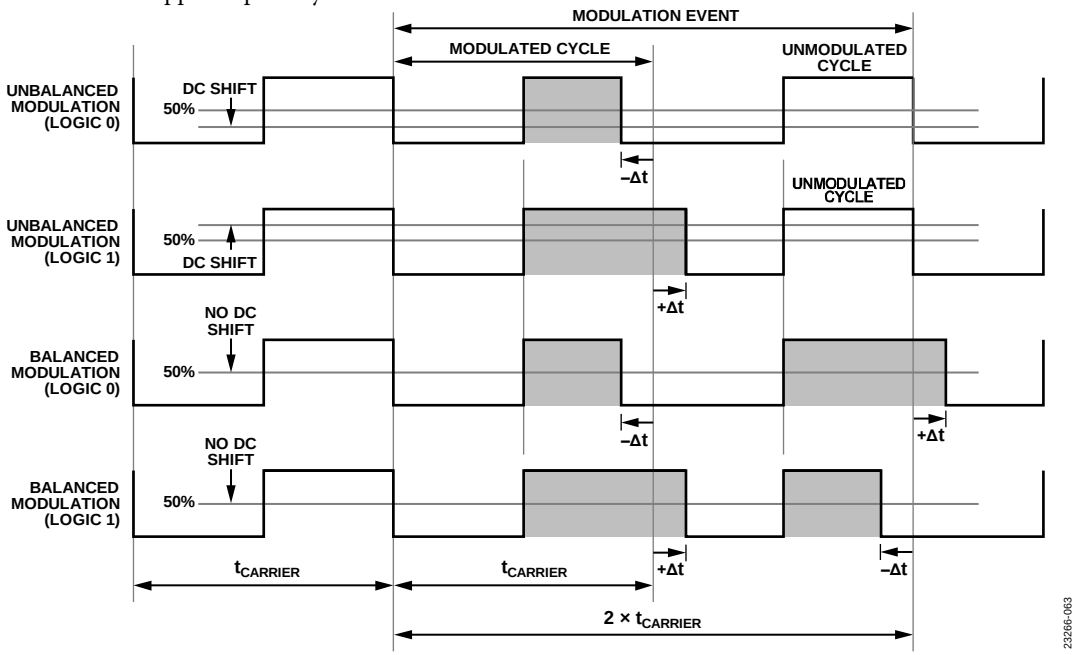


Figure 63. Modulation Events

DEMODULATOR SENSITIVITY

Modulated input signals exhibiting a small change in pulse width require more sensitivity than those that exhibit a large change. To control the demodulator sensitivity to pulse width changes, use Bits[1:0] in Register M per the register legend in Figure 62.

The demodulator defaults to its most sensitive setting, 00 (binary). Increasing the binary value decreases the sensitivity, with 11 (binary) constituting the least sensitive setting. Decreased sensitivity implies the need for more robust modulation events (that is, larger Δt in Figure 63) for reliable demodulation.

DEMODULATOR PERSISTENCE

When the demodulator detects a modulation event, it generates a single demodulator clock pulse (see Figure 62) lasting one $t_{CARRIER}$ period. Because demodulation events normally occur at intervals of t_{GRID}, the demodulator clock signal consists of single pulses occurring at the t_{GRID} rate. The demodulator clock pulse synchronizes the R divider and causes the reference TDC to tag the accompanying time stamp. Thus, tagged reference time stamps occur at the t_{GRID} rate (see the Demodulator Synchronization section).

Note that if the incoming modulation events stop occurring, the demodulator clock pulses cease and the TDC stops generating tags. Assuming the TDC tags route to one of the DPLLs, the absence of tags causes the DPLL to unlock. The persistence feature provides a mechanism for the demodulator to continue to generate demodulator clock pulses at the t_{GRID} rate when modulation events are no longer present at the input, thereby preventing the DPLL from losing lock.

The user controls the persistence functionality via Bit 6 in Register M per the register legend in Figure 62. When Bit 6 = 0, the demodulator clock signal only occurs when the demodulator detects a modulation event. When D6 = 1 (default), the demodulator learns the modulation period (which is t_{GRID}) and continues to generate demodulator clock pulses at the t_{GRID} rate. Figure 64 shows the timing of the demodulator clock signal.

The demodulator clock signal is available via an appropriately configured Mx status pin (see the Status and Control Pins section).

Note that if the demodulator loses synchronization (for example, when incoming modulation events go off grid), the demodulator requires at least three consecutive t_{GRID} intervals (with valid modulation events) to resynchronize the persistence mechanism. Because the demodulator requires three consecutive t_{GRID} intervals (with persistence enabled) for resynchronization, the occurrence of an occasional off grid modulation event does not cause the persistence mechanism to realign the demodulator output clock and resynchronize the R divider. Instead, the persistence mechanism effectively filters out the occasional occurrence of off grid modulation events.

DEMODULATOR BANDWIDTH

The demodulators operate within two overlapping frequency bands: Band 0 and Band 1. The user must select the appropriate demodulator bandwidth (based on the input carrier frequency) via Bit 0 in Register N per the register legend in Figure 62. Logic 0 selects Band 0, and Logic 1 (default) selects Band 1.

Note that demodulator bandwidth control applies to a group of four demodulators rather than an individual demodulator. Thus, the user must ensure that all demodulators of the same group that are actively demodulating have carrier frequencies within the specified frequency range.

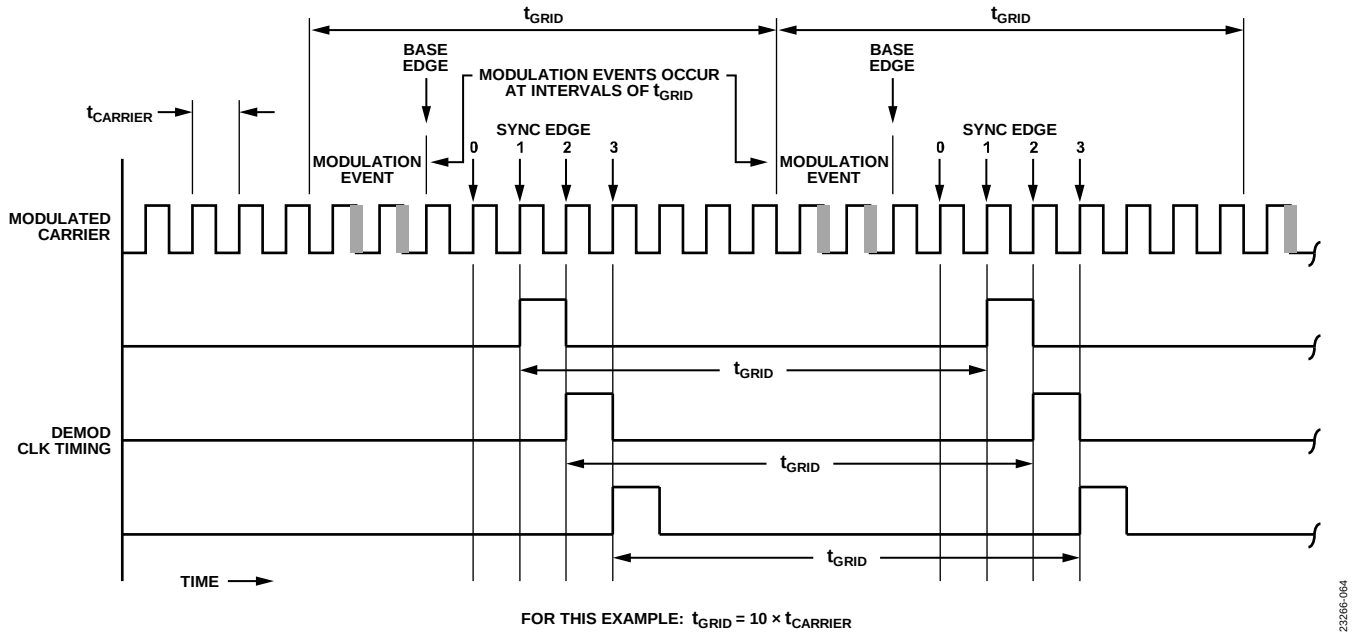


Figure 64. Demodulator Synchronization

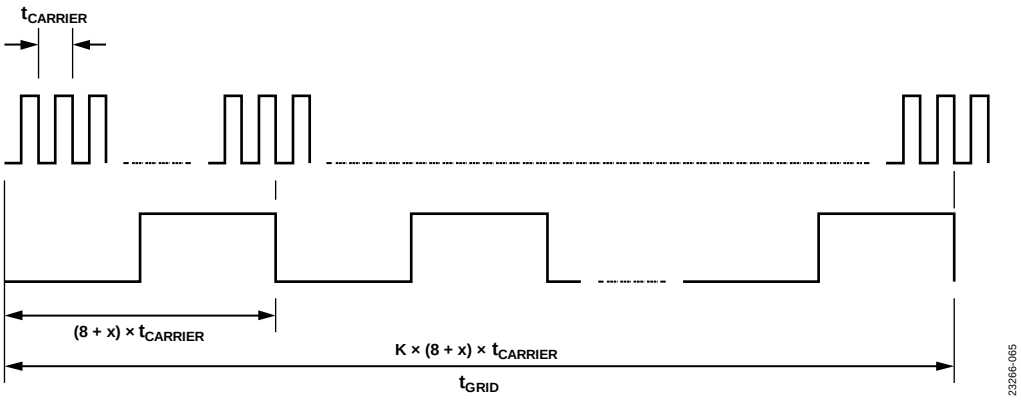


Figure 65. Grid Timing Diagram

DEMODULATOR SYNCHRONIZATION

The demodulators expect modulation events to occur at regular periodic intervals, t_{GRID}, with t_{GRID} comprising an integer number of $t_{CARRIER}$ periods (see Figure 65) such that

$$t_{GRID} = K \times (8 + x) \times t_{CARRIER}$$

where:
K is an integer $K \geq 1$.
x is an integer with $x \geq 0$.

Essentially, the value of x relates to the divide ratio, R, of the reference divider as follows:

$$R = 8 + x$$

The net effect is that the reference divider produces an output clock of period, $t_{CARRIER} \times (8 + x)$. The reference TDC time stamps the rising edge of each divider output clock cycle with K time stamps occurring in each t_{GRID} period.

When the demodulator identifies a modulation event within the carrier signal, it generates a demodulator clock pulse with timing according to Figure 64. The demodulator clock pulse synchronizes the reference divider and causes the reference TDC to tag the associated time stamp (see Figure 62), which creates a fixed timing relationship between incoming modulation events and the occurrence of tagged time stamps. Specifically, every K^{th} time stamp (see Figure 65) is a tagged time stamp.

The demodulator denotes the next rising edge of the carrier following the modulation event as the base edge. The user can set the timing of the demodulator clock pulse to occur a predefined number of $t_{CARRIER}$ periods after the base edge via the 2-bit unsigned bit field comprising Bits[5:4] in Register M per the register legend in Figure 62. The decimal value of this bit field, 0 (default), 1, 2, or 3, corresponds to the desired synchronization edge in Figure 64.

The user must program a bit field value other than the default value of zero (Bits[5:4] = 0 is not supported).

If a modulation event occurs off grid or the modulation rate changes to a new t_{GRID} period, the demodulator must resynchronize. The demodulator requires at least three $t_{CARRIER}$ periods (including the first modulated cycle) to resynchronize when Bit 6 in Register M per the register legend in Figure 62 is Logic 0 and at least eight $t_{CARRIER}$ periods when Bit 6 is Logic 1.

DISTRIBUTION CLOCK OUTPUT DRIVERS

DISTRIBUTION CLOCK OUTPUT DRIVERS OVERVIEW

The AD9546 has 10 clock output pins capable of generating up to 10 output clock signals. The 10 pins comprise two groups of clocks differentiated by the PLL channel available as the clock source driving the group. PLL0 can clock the Output 0 group, and PLL1 can clock the Output 1 group.

The Output 0 group comprises six of the 10 clock output pins with the six pins consisting of three pin pairs. Each pin pair represents an output subgroup (A, B, and C) with the pins designated as

- OUT0AP and OUT0AN
- OUT0BP and OUT0BN
- OUT0CP and OUT0CN

The Output 1 group comprises the remaining four output pins consisting of two pin pairs. Each pin pair represents an output subgroup (A and B) with the pins designated as

- OUT1AP and OUT1AN
- OUT1BP and OUT1BN

Each output driver has three programmable parameters controlling

- Output current direction (source or sink)
- Output current magnitude
- Driver configuration

The three control parameters for a given output are bits that reside in the same register with a register address specific to a specific output per Table 57.

Table 57. Output Driver Control Register Address

Output	Register Address
OUT0A	0x10D7
OUT0B	0x10D8
OUT0C	0x10D9
OUT1A	0x14D7
OUT1B	0x14D8

OUTPUT CURRENT CONTROL

Each pin associated with a subgroup pin pair (OUTxyP/OUTxyN, where x is 0 or 1 and y is A, B, or C) has an output driver and a channel divider associated with it. Both drivers of an output pin pair act as either current sources or current sinks. The user programs the source or sink option via Bit 0 of the appropriate register address in Table 57. Logic 0 (default) sets both drivers to current sink mode, and Logic 1 sets both drivers to current source (or HCSL) mode.

To control the magnitude of the current, use Bits[2:1] of the appropriate register address in Table 57. Table 58 relates the value of the two bits to the magnitude of the driver current.

Table 58. Output Driver Current

Bits[2:1] Value (Decimal)	Output Current (mA)
0	7.5 (default)
1	12.5
2	15
3	Not applicable

OUTPUT MODE CONTROL

Each output pin pair has two channel dividers and two output drivers. Figure 66 shows the connectivity between the dividers and drivers. The divider to driver connectivity depends on the selected output mode.

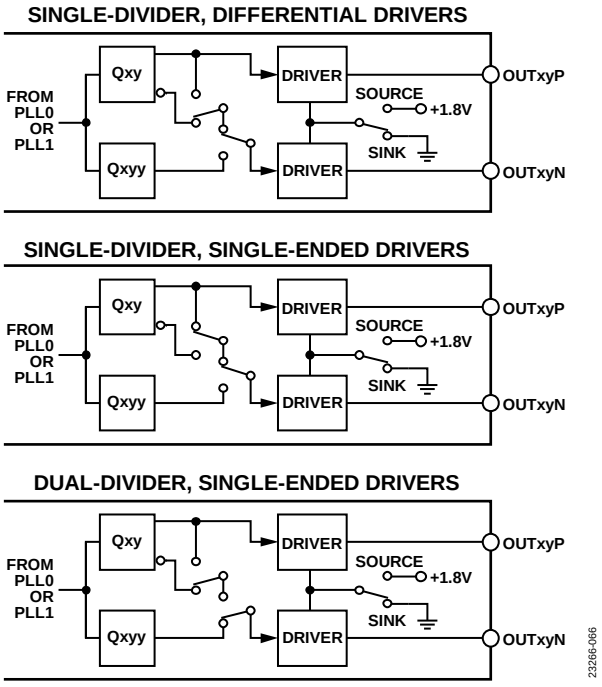


Figure 66. Output Driver Modes

The user controls the connectivity of the dividers and drivers of each subgroup pin pair via Bits[4:3] of the appropriate register address (see Table 57). The value of the two bits selects the divider to driver relationship per Table 59.

Table 59. Driver Output Mode

Bits[4:3] Value (Decimal)	Output Driver Mode
0	Single-divider, differential (default)
1	Single-divider, single-ended
2	Dual-divider, single-ended
3	Not applicable

The single divider modes use only the Qxy divider (with the Qxyy divider powered down). The dual divider mode makes use of both the Qxy and Qxyy dividers.

In single-divider differential mode, the clock signal at the OUTxyN pin phase is inverted relative to the OUTxyP pin. In single-divider single-ended mode, the clock signals at the OUTxyP and OUTxyN pins are in phase relative to one another. In dual-divider single-ended mode, the Qxy divider drives the OUTxyP pin and the Qxyy divider drives the OUTxyN pin. In this mode, each output of the subgroup can have a different frequency and phase offset.

OUTPUT DRIVER CONFIGURATIONS

Each driver receives a clock input signal from its associated channel divider (see Figure 66). The current source (or sink) of the driver is either on or off, depending on the logic level output by the divider. That is, when the divider output is Logic 0, the driver output current is 0 mA. Conversely, when the divider output is Logic 1, the drive output current is the programmed output current value. The Qxy divider has normal and inverted logic outputs, which means when the Qxy divider connects to both drivers and one current source is on, the other is off, allowing a differential output signal.

Differential HCSL Output

To drive a standard HCSL receiver, refer to Figure 39. This configuration requires programming the driver settings as follows: single-divider, differential drivers; current source; driver current = 15 mA. In this mode, the driver expects a 50 Ω termination to ground on each driver output pin. The 15 mA drive current yields 750 mV swing across each 50 Ω load.

LVDS Output

To drive an LVDS receiver using ac coupling, refer to Figure 38. This configuration requires programming the driver settings as follows: single-divider, differential drivers; current source; and driver current = 7.5 mA. In this mode, the driver expects a 50 Ω termination to ground on each driver output pin and dc blocking capacitors with a 100 Ω differential termination at the receiver. The expected termination arrangement yields 188 mV swing across the 100 Ω load but with 188 mV common mode across the output pins, and thus, the need for dc blocking capacitors to preserve the common-mode bias of 1.2 V of the receiver.

To drive an LVDS receiver directly with dc coupling, use a T network, as shown in Figure 41. This configuration requires programming the driver settings as follows: single-divider, differential drivers; current sink; and driver current = 7.5 mA. This arrangement yields a 375 mV swing across the output pins with 1.24 V common mode. When using an additional 100 Ω differential termination (R_L) at the receiver, however, ac coupling is necessary (as shown in Figure 38). The additional termination also requires programming a drive current of 15 mA. This arrangement yields a 375 mV swing across the output pins with 0.67 V common mode (thus, the ac coupling requirement to preserve the 1.2 V common-mode bias of the LVDS receiver).

To drive an LVDS receiver using a Thevenin equivalent termination, refer to Figure 42, which exhibits the equivalent of a 50 Ω pull-up resistor to 1.42 V on each output pin. This configuration requires programming the driver settings as follows: single-divider, differential drivers; current sink; and driver current = 7.5 mA. This arrangement yields a 375 mV swing across the output pins with 1.23 V common mode.

When using an additional 100 Ω differential termination (R_L) at the receiver, however, ac coupling is necessary (as shown in Figure 38), as well as programming a drive current of 15 mA. This arrangement yields 375 mV swing across the output pins with 1.05 V common mode (thus the ac coupling requirement to preserve the 1.2 V common-mode bias of the LVDS receiver).

To drive an LVDS-compatible receiver that can handle boosted signal swing, use a Thevenin equivalent termination per Figure 43, which exhibits the equivalent of a 50 Ω pull-up resistor to 1.60 V on each output pin. This configuration requires programming the driver settings as follows: single-divider, differential drivers; current sink; and driver current = 15 mA. This arrangement yields 750 mV swing across the output pins with 1.23 V common mode.

CML Output

To configure an output for CML signals, see Figure 40. When using a 50 Ω pull-up resistor to 1.2 V, this configuration requires programming the driver settings as follows: single-divider, differential drivers; current sink; and driver current = 7.5 mA. This arrangement yields a 375 mV swing across the output pins with 1.01 V common mode.

When using a 50 Ω pull-up resistor to 1.5 V or 1.8 V, program a drive current of 15 mA. This arrangement yields a 750 mV swing across the output pins with 1.125 V common mode for a 1.5 V supply and 1.425 V common mode for a 1.8 V supply.

Dual, Single-Ended, In Phase Outputs

To configure the output to produce the same signal (in phase) on each pin of an OUTxyP/OUTxyN pin pair, refer to Figure 44. This configuration requires programming the driver settings as follows: single-divider, single-ended drivers and current source. Select the driver current to yield an acceptable voltage swing based on the load resistance, R_L. The output is essentially a current source (on or off per the logic state of the Q_{xy} divider output) with an external pull-down resistor. Thus, the output signal swing is between ground and $V = I \times R_L$.

Independent, Single-Ended Outputs

To configure the output to produce independent signals on the two output pins of an OUTxyP/OUTxyN pin pair, refer to Figure 46. This configuration requires programming the driver settings as follows: dual-divider, single-ended drivers and current source. Select the driver current to yield an acceptable voltage swing based on the load resistance, R_L. Note that the output is essentially a current source (on or off per the logic state of the Qxy and Qxyy dividers) with an external pull-down resistor. Thus, the output signal swing is between ground and $V = I \times R_L$.

OUTPUT DRIVER RESET

Because driver pairs (OUT0AP and OUT0AN, for example) are configurable as differential or single-ended, when the Q dividers associated with a driver pair stop toggling (due to a divider reset, for example), the output state of the driver pair is unknown. To remedy this uncertainty, the user can force a driver pair to a known state by means of the various driver reset bits.

Bit 2 of Register 0x2101 allows the user to reset all the driver pairs of Channel 0. Bit 2 of Register 0x2201 allows the user to reset all the driver pairs of Channel 1. The user can reset individual output driver pairs via Bit 5 per the appropriate register addresses in Table 60.

Table 60. Output Driver Reset Register Address

Output	Register Address
OUT0AP/OUT0AN	0x2102
OUT0BP/OUT0BN	0x2103
OUT0CP/OUT0CN	0x2104
OUT1AP/OUT1AN	0x2202
OUT1BP/OUT1BN	0x2203

When a driver pair is in a differential configuration, resetting the driver pair forces the OUTxyP driver to a Logic 0 state and the OUTxyN driver to a Logic 1 state. When a driver pair is in a single-ended configuration, resetting the driver pair forces the OUTxyP and OUTxyN drivers to a Logic 0 state.

OUTPUT MUTING

Manual Output Muting

To mute all the drivers associated with PLL0 or PLL1, use Bit 1 in Register 0x2101 and Register 0x2201 for PLL0 and PLL1, respectively. To mute individual drivers, use Bits[3:2] per the appropriate register addresses in Table 60, where Bit 3 corresponds to the OUTxyN driver and Bit 2 corresponds to the OUTxyP driver.

Automatic Output Unmuting

Automatic unmuting works in conjunction with the autosynchronization function (see the Autosynchronization Trigger section). Part of the synchronization sequence involves synchronously unmuting the output drivers. The AD9546 provides the user options for the timing of the unmuting of the output drivers.

To select the automatic unmute conditions, use Bits[1:0] of Register 0x10DC for PLL0 and Register 0x14DC for PLL1. Table 61 shows the automatic unmute conditions invoked by these bits. The automatic unmute control applies to PLL0 and PLL1 independently.

Table 61. Autotmatic Unmute Conditions

Bits[1:0] (Decimal)	Automatic Unmute Condition
0	Immediate (default)
1	Upon activation of a hitless profile
2	Upon phase lock (hitless mode only)
3	Upon frequency lock (hitless mode only)

When Bits[1:0] = 0 (decimal), the drivers unmute immediately upon release of the synchronization request. When Bits[1:0] = 1 (decimal), the drivers do not unmute until release of the synchronization request and subsequent activation of a hitless/zero delay profile. When Bits[1:0] = 2 (decimal), the drivers do not unmute until release of the synchronization request and subsequent phase lock of the appropriate PLL channel (assuming a hitless/zero delay profile). When Bits[1:0] = 3 (decimal), the drivers do not unmute until release of the synchronization request and subsequent frequency lock of the appropriate PLL channel (assuming a hitless/zero delay profile).

The user can also selectively opt out of the automatic unmuting feature on a per output basis. For PLL0, use Bits[7:2] of Register 0x10DC, where Bits[7:2] correspond to the driver associated with OUT0CN, OUT0CP, OUT0BN, OUT0BP, OUT0AN, and OUT0AP, respectively. For PLL1, use Bits[5:2] of Register 0x14DC where Bits[5:2] correspond to the driver associated with OUT1BN, OUT1BP, OUT1AN, and OUT1AP, respectively.

After an automatic mute event occurs for a particular PLL channel, the user can clear the automatic mute state by setting Bit 4 of Register 0x2107 for PLL0 and Register 0x2207 for PLL1. However, this action is generally unnecessary, because the device automatically clears the automatic mute state whenever the user changes the state of an automatic unmute bit and issues an IO update operation.

Output Mute Retiming

If the driver output mutes immediately upon receiving a mute command, it may result in the output signal shutting off before the normal falling edge of the associated Q divider output. Early shutoff of the output signal produces a runt pulse (that is, the normal Logic 1 period of the Q divider is cut short).

By default, the AD9546 uses a retiming block to prevent the generation of a runt pulse. However, the user can bypass the retiming block via Bit 5 per the appropriate register addresses in Table 57. Bypassing the retiming block causes the output driver to mute immediately upon receiving a mute command.

DISTRIBUTION DIVIDERS (Q DIVIDERS)

DISTRIBUTION DIVIDERS OVERVIEW

Five pairs of Q dividers correspond to the five pairs of clock output drivers (see the Distribution Clock Output Drivers section). Unlike typical clock dividers, which count the rising edge (or falling edge) of its input clock, the Q dividers count both the rising and falling edges of its input clock. Dual-edge counting enables features not possible with a typical counter.

The Q dividers respond to several controllers that implement the various features of the Q dividers. Figure 67 shows a diagram of the Q dividers and associated controllers. Although not explicitly shown, the output of each Q divider routes to its corresponding output driver.

Each Q divider has independent phase offset (delay) capability as governed by a phase offset controller (see the Distribution Phase Offset Control section). In addition, a sophisticated and flexible synchronization mechanism provides synchronized output clock timing for any or all Q divider output signals (see the Distribution Output Clock Synchronization section).

A JESD204B and burst controller enables each Q divider to generate a clock signal supporting JESD204B or to generate flexible burst patterns supporting gapped clock and similar applications (see the Distribution N Shot/PRBS Output Clocking section).

Furthermore, the Qxy divider of each Q divider pair is capable of modulating (that is, time shifting) falling edges of the output clock signal. The modulation of the clock edge location supports embedded clock applications such as transporting a lower frequency clock signal over a higher frequency carrier clock (see the Distribution Embedded Output Clock Modulation section).

At power-up, the synchronization controller holds the disabled Q dividers (that is, there are no output clocks). Consequently, the user must issue a synchronization request (see the Distribution Output Clock Synchronization section) following a power-up or reset to initialize the Q dividers.

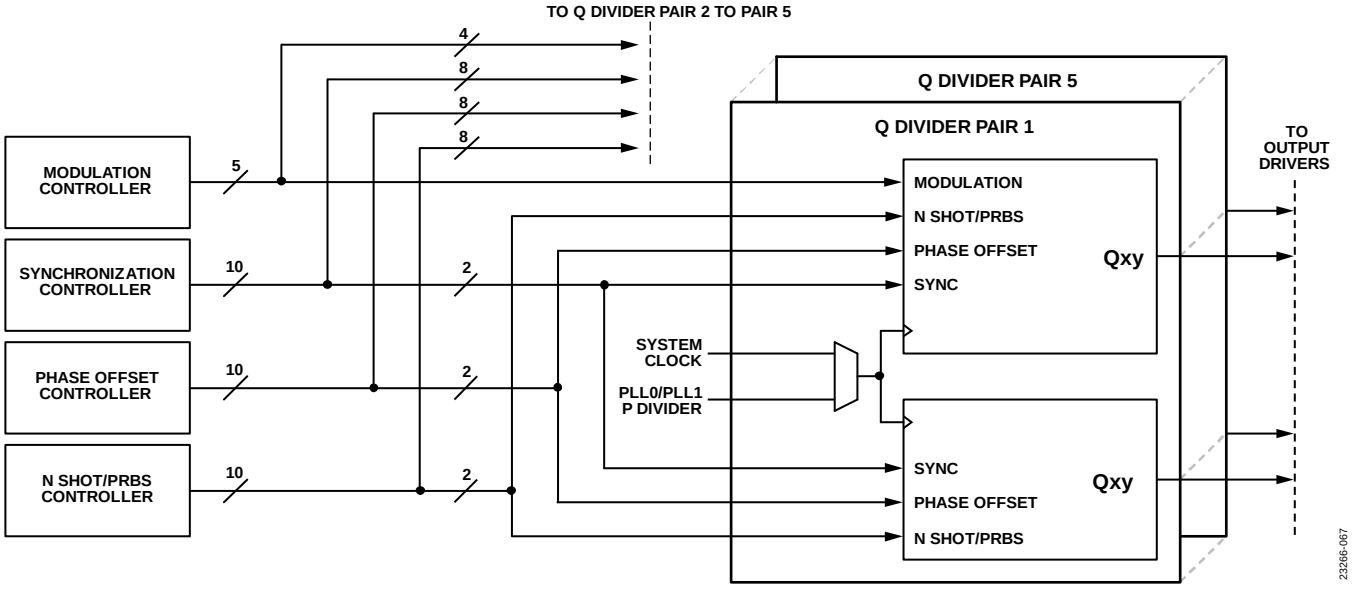


Figure 67. Functional Block Diagram of Q Dividers and Controllers

Q DIVIDER CLOCK SOURCE SELECTION

Each pair of Q dividers shares a common clock input (see Figure 67) that originate from one of two sources.

- From the output of the associated APLL channel of the Q divider.
- From the output of the system clock PLL VCO (~2.3 GHz).

The first source is the normal (default) operating mode, which provides frequency translation from a reference input to the distribution output via the corresponding PLL channel. The second source is the output frequency of the system clock PLL, which is useful for clocking certain peripheral devices (a microprocessor, for example). In this scenario, the user can employ an optional external electronically erasable programmable read only memory (EEPROM) to download a device configuration that provides an output clock signal at power up via the system clock source path (see the EEPROM Usage section).

Table 62 shows the register address and bits associated with selecting the clock source for the Q dividers. Logic 0 (default) selects the output of the corresponding APLL as the Q divider clock source, whereas Logic 1 selects the output of the system clock PLL as the Q divider clock source.

Table 62. Q Divider Clock Source Selection

Q Divider	Register Address	Bit
Q0A and Q0AA	0x10DA	D1
Q0B and Q0BB	0x10DA	D2
Q0C and Q0CC	0x10DA	D3
Q1A and Q1AA	0x14DA	D1
Q1B and Q1BB	0x14DA	D2

When the AD9546 is configured with the system clock as the clock source for a Q divider pair, use Bit 0 (system clock synchronization mask) of Register 0x10DA and Register 0x14DA to prevent those particular dividers from being resynchronized every time a synchronization trigger occurs (see the Distribution Output Clock Synchronization section). When this bit is set, any Q divider associated with a specific PLL channel and with the system clock as the clock source is immune to synchronization events. This feature is particularly useful for outputs that serve as the clock source to a microprocessor, for example, because an unmasked synchronization event disrupts the clock signal of the processor.

In some applications, when more than one Q divider pair uses the system clock as its clock source, it may be desirable to synchronize at least two of those Q divider pairs using the system clock synchronization mask bit of Register 0x10DA and Register 0x14DA. To do so, initiate a synchronization sequence (see the Distribution Output Clock Synchronization section), then program the corresponding mask system clock synchronization bit to Logic 1. To reconfigure the affected Q dividers by changing the divide ratio, for example, the user must first program the corresponding system clock synchronization mask bit to Logic 0.

INTEGER DIVISION

The Q divider divide ratio (Q) depends on a 32-bit unsigned integer in the register ranges shown in Table 63. To program the divide ratio, write the value of Q into the corresponding address range and then set the IO update bit.

The range of Q is 1 to 4,294,967,295.

For example, to establish a divide ratio of Q = 1,000,000, the required 32-bit value is 1,000,000 (0x 000F 4240 hexadecimal).

Table 63. Q Divider Divide Ratio Address Ranges

Q Divider	Register Address
Q0A	0x1100 to 0x1103
Q0AA	0x1109 to 0x110C
Q0B	0x1112 to 0x1115
Q0BB	0x111B to 0x111E
Q0C	0x1124 to 0x1127
Q0CC	0x112D to 0x1130
Q1A	0x1500 to 0x1503
Q1AA	0x1509 to 0x150C
Q1B	0x1512 to 0x1515
Q1BB	0x151B to 0x151E

A value of 0 (default) sets a divide ratio of 1, which is an unsupported divide ratio. Therefore, at power-up or after resetting the device, the user must program the Qxy divider ratio bit fields with values greater than 1 to enable the corresponding Q dividers.

Although it is possible to program a different divide ratio for each Q divider of a Q divider pair, the recommendation is to avoid doing so because a Q divider pair with differing divide ratios results in two different output frequencies and can cause cross coupling issues (due to the close physical proximity of the corresponding output drivers). The intent is for Q divider pairs to operate with the same output frequency but with the option to control the relative phase between the outputs (see the Distribution Phase Offset Control section).

HALF INTEGER DIVISION

The Q dividers are unique in that they support half integer division, a benefit of the Q dividers counting both the rising and falling edges of the input clock. To enable half integer division for the corresponding Q divider, use Bit 5 (half integer divide) of the appropriate register per Table 64.

Table 64. Q Divider Half Integer Divide Register Address

Q Divider	Register Address
Q0A	0x1108
Q0AA	0x1111
Q0B	0x111A
Q0BB	0x1123
Q0C	0x112C
Q0CC	0x1135
Q1A	0x1508
Q1AA	0x1511
Q1B	0x151A
Q1BB	0x1523

When the half integer divide bit is Logic 1, it effectively adds an extra 0.5 to the divide ratio prescribed by the corresponding 32-bit Q divider ratio. The value of the half integer divide bit does not become effective until the user sets the IO update bit.

For example, assume the Q divider ratio is 100. If the associated enable half integer divide bit is Logic 1, the total divide ratio is 100.5.

Q DIVIDER RESET

Each Q divider has a dedicated Q divider reset bit via Bits[1:0] of the registers shown in Table 65, where Bit 0 applies to the first Q divider in the row (for example, Q0A) and Bit 1 applies to the second Q divider in the row (for example, Q0AA).

The Q divider reset bits allow the user to reset the Q dividers manually in applications that do not require output clock synchronization. In most cases, however, there is no need to set these bits during normal operation because the synchronization controller automatically handles the Q divider reset function (see the Distribution Output Clock Synchronization section).

Table 65. Output Driver Reset Register Address

Q Divider	Register Address
Q0A and Q0AA	0x2102
Q0B and Q0BB	0x2103
Q0C and Q0CC	0x2104
Q1A and Q1AA	0x2202
Q1B and Q1BB	0x2203

Q DIVIDER CONSTRAINTS

Changing the divide ratio of any one of the Q dividers invokes a synchronization process (see the Autoreconfiguration Synchronization Trigger section). The act of changing the divide ratio of a single Q divider causes a disturbance of all the output clock signals of the associated PLL channel. Therefore, it is not possible to change the divide ratio of one Q divider without disturbing all the output clock signals of the associated PLL channel.

When the system clock is the selected clock source for a Q divider pair (see the Distribution Output Clock Synchronization section), an unlock event on the system clock may cause the Q divider pair to enter an invalid state. Recovery from this state requires a divider reset. Therefore, after a system clock unlock event, the user must follow these steps:

1. Calibrate the system clock (see the System Clock Calibration section).
2. Calibrate APLL0 and APLL1 (see the VCO Calibration section).
3. Reset any Q dividers (see the Q Divider Reset section) using the system clock as an input clock source.
4. Synchronize output distribution (see the Distribution Output Clock Synchronization section).

When using embedded output clock modulation to modulate the Q divider output (see the Distribution Embedded Output Clock Modulation section), a minimum Q divider divide ratio is necessary.

$$Qxy \geq 8$$

where:
x = 0 or 1.
y = A, B, or C.

When employing the N shot triggering mechanism to trigger a Q divider (see the N Shot Triggering section), a minimum Q divider divide ratio of 8 is necessary.

$$Qxy \geq 8$$

where:
x = 0 or 1.
y = A, B, or C.

HITLESS/ZERO DELAY FEEDBACK

When configured for hitless or zero delay operation, the DPLL of the affected PLL channel requires a feedback clock signal to its N divider from one of the distribution outputs. This feedback mechanism is via the hitless or zero delay feedback and synchronization block of the N shot/PRBS controller in Figure 68. Selection of the desired feedback path is via the frequency translation profiles. See the Internal Zero Delay (Hitless) Mode section in the Frequency Translation Loops section for programming details.

DISTRIBUTION PHASE OFFSET CONTROL

OUTPUT PHASE OFFSET OVERVIEW

The phase offset controller (see Figure 67) governs the application of phase offsets to the individual Q dividers. The controller implements two categories of phase offset: initial phase offset and subsequent phase offset.

An initial phase offset applies after the device is powered up or reset and the user issues a synchronization request that is completed (see the Distribution Output Clock Synchronization section). The completed synchronization request results in the establishment of the initial phase offset. Subsequent phase offsets apply for all subsequent synchronization requests (that is, any synchronization requests occurring after the initial power-up or reset synchronization request). See the Distribution Output Clock Synchronization section regarding synchronization requests.

Initial and subsequent phase offsets require the Q divider pulse width control feature to be turned off (see the Output Pulse Width Control section).

The initial phase offset depends on a 33-bit unsigned integer in the register ranges shown in Table 66. The four lower addresses for each Q divider in Table 66 carry the 32 LSBs of the 33-bit integer, and Bit 6 of the upper address carries the MSB of the 33-bit integer (Bit 7 is unused and Bits[5:0] apply to other Q divider features).

Table 66. Q Divider Phase Offset Control Address Ranges

Q Divider	Register Address
Q0A	0x1104 to 0x1108
Q0AA	0x110D to 0x1111
Q0B	0x1116 to 0x111A
Q0BB	0x111F to 0x1123
Q0C	0x1128 to 0x112C
Q0CC	0x1131 to 0x1135
Q1A	0x1504 to 0x1508
Q1AA	0x150D to 0x1511
Q1B	0x1516 to 0x151A
Q1BB	0x151F to 0x1523

INITIAL PHASE OFFSET

The initial phase offset of a Q divider is dependent on the divide ratio and state of the half integer divide bit, which together constitute the number of rising and falling input clock edges that span one period of the Q divider output. For details on the divide ratio and half integer divide, see the Distribution Dividers (Q Dividers) section.

In terms of the values defining the divide ratio of the Q divider,

$$E = (2 \times \textit{Divide Ratio}) + \textit{Half Integer Divide}$$

$$= 2 \times Q_N$$

where:
E is the total number of input edges per output period of the Q divider.
Divide Ratio is the value of the 32-bit integer stored in the corresponding Q divider register in Table 63.
Half Integer Divide is the value of the corresponding Q divider half integer divide bit (0 or 1).
Q_N is the complete divide factor (for example, 101.5) of a specific Q divider.

Thus, for $Q_N = 101.5$, $E = 203$.

The value of E relates to the phase offset angle, θ, as follows:

$$\theta = 360° \times (\textit{Phase Offset}/E)$$

where *Phase Offset* is the integer stored in the phase offset registers (the phase offset register address ranges appear in Table 66).

The phase offset angle equation implies that the programmed phase offset value must be less than E (that is, the phase offset value has a range of 0 to E − 1). Note that programming an invalid phase value results in the phase offset controller taking no action other than setting an error status in Bits[5:0] of Register 0x310E for PLL0 and Bits[3:0] of Register 0x320E for PLL1. For Register 0x310E, Bits[5:0] correspond to Q0CC, Q0C, Q0BB, Q0B, Q0AA and Q0A, respectively. For Register 0x320E, Bits[3:0] correspond to Q1BB, Q1B, Q1AA, and Q1A, respectively. The user can access the phase control error results via an appropriately configured Mx status pin (see the Status and Control Pins section). The Mx pin output signal constitutes a logical OR of the six status bits associated with PLL0 or the logical OR of the four status bits associated with PLL1.

SUBSEQUENT PHASE OFFSETS

Subsequent phase offsets involve writing a new 33-bit phase offset value to the appropriate Q divider per Table 66 and then setting the IO update bit. The magnitude of the applied phase offset is the same as described in the Initial Phase Offset section.

Unlike the initial phase offset, the phase controller implements subsequent phase offsets in stepwise fashion as a sequence of phase steps, where the 33-bit phase offset value denotes the amount of phase offset present at the termination of the sequence. The controller executes the phase steps at a rate commensurate with the Q divider output period. This mechanism makes it possible to soften the phase transient that results when applying subsequent phase offsets. The reason is the stepwise implementation of the phase offset effectively limits phase transients to some maximum amount per output cycle of the Q divider, which effectively constitutes a phase rate of change limiter. The limited rate of change of the phase replaces the relatively large instantaneous phase step of a phase adjustment with a series of small phase steps, thereby reducing the spectral content normally associated with phase adjustment.

When the Q divider is in the process of phase slewing as a part of the phase offset sequence, the phase offset controller indicates phase slew active status via Bits[5:0] of Register 0x310D and Bits[3:0] of Register 0x320D. For Register 0x310D, Bits[5:0] correspond to Q0CC, Q0C, Q0BB, Q0B, Q0AA, and Q0A, respectively. For Register 0x320D, Bits[3:0] correspond to Q1BB, Q1B, Q1AA, and Q1A, respectively.

The user can access the phase slew active results via an appropriately configured Mx status pin (see the Status and Control Pins section). The Mx pin output signal constitutes a logical OR of the six status bits associated with PLL0 or the logical OR of the four status bits associated with PLL1.

Maximum Phase Slew Step Size

During the execution of a subsequent phase offset, the user controls the maximum step size of the phase steps for a particular Q divider via Bits[2:0] per the same address given in Table 64. The value stored in Bits[2:0] establishes the maximum phase step size per Table 67 (in which E is the number of Q divider input edges per output period (see the Initial Phase Offset section) and floor(x) means to round x to the nearest integer in the direction of −∞). Keep in mind that Table 67 defines a maximum phase step size during phase slewing, but the controller may use smaller values to ensure the terminal phase value does not exceed the programmed phase offset value.

When Bits[2:0] = 7 decimal, the phase controller requires only one step to reach the desired phase offset regardless of the size of the phase offset. Therefore, the bit setting effectively deactivates the phase slewing feature. When Bits[2:0] = 7, the phase controller behaves as though the phase slew mode bit is Logic 0 (that is, lag only) even if the phase slew mode bit is Logic 1 (see the Phase Slew Mode section).

When Bits[2:0] = 0 or 1 decimal, the slewing operation is limited to half-cycles or full cycles, respectively, of the input clock. Because a 0 or 1 setting represents the smallest possible maximum phase step for most Q divider divide ratios, most choices of phase offset result in phase slewing as though the maximum phase slew step was set to 0 or 1.

Maximum phase slew step values greater than 1 decimal require the user to ensure that the maximum phase step size is greater than or equal to one input half-cycle. Otherwise, the phase slewing function is invalid, in which case the device flags an error via the control error bits described in the Initial Phase Offset section. For example, when Q_N = 5 (that is, E = 10 (see the Initial Phase Offset section)), one half-cycle constitutes 36°. As such, maximum phase slew step values of 2 or 3 are invalid, and the controller sets the appropriate error flag.

Table 67. Maximum Phase Slew Step Size

Bits[2:0] (Decimal)	Maximum Phase Step Size (Degrees)	Comment
0	360 ÷ E	One input half-cycle
1	180 ÷ E	Two input half-cycles
2	360 × floor(E/32)/E	~11.25°
3	360 × floor(E/16)/E	~22.5°
4	360 × floor(E/8)/E	~45°
5	360 × floor(E/4)/E	~90°
6	360 × floor(E/2)/E	~180°
7 (Default)	360 × (E − 1)/E	~360°

Phase Slew Mode

During the execution of a subsequent phase offset, the phase controller can slew phase in two different modes. The user selects the mode via Bit 3 per the same address given in Table 64.

When Bit 3 = 0 (default), the phase controller slews phase in the direction that reduces the output frequency during the stepwise phase adjustment sequence. Alternatively, when Bit 3 = 1, the phase controller slews phase in the direction requiring the fewest number of steps. However, when the half integer divide bit is Logic 1 and the desired phase shift is within one Q divider input half-cycle of the midpoint of an output cycle, the phase controller may choose the slightly longer direction.

When Bit 3 = 1, the output frequency may change in either direction, as required, for any given stepwise phase adjustment sequence (due to the phase controller choosing the direction of the fewest number of steps).

Output Pulse Width Control

By default, the Q divider output generates a clock signal with a 50% pulse width. However, the user has the option to control the pulse width of the Q divider output. The control granularity is one half cycle of the input clock of the Q divider. Pulse width control is via Bit 4 per the same address given in Table 64.

When Bit 4 = 0, the phase offset controller uses the programmed 33-bit phase offset as a phase offset (as described in the Initial Phase Offset section). When Bit 4 = 1, the 33-bit phase offset value defines a pulse width rather than a phase offset.

The pulse width, in percent, is given by

$$Pulse\ Width\ (\%) = 100 \times Phase\ Offset/E \quad (2)$$

where:
Pulse Width defines the portion of an output clock cycle from the rising edge to the falling edge.
Phase Offset is the integer stored in the phase offset registers. Table 66 shows the phase offset register address ranges.
E is as defined in the Initial Phase Offset section.

For example, determine the value necessary to yield a 14% pulse width given E = 7301. Solving Equation 2 for *Phase Offset* yields the following:

$$\textit{Phase Offset} = \textit{Pulse Width} \times E/100$$
$$= 14 \times 7301/100$$
$$= 1022 \text{ (nearest integer)}$$
$$= 0x\ 0\ 0000\ 03FE \text{ (hexadecimal)}$$

The user has the option to adjust the output pulse width only prior to execution of the first subsequent phase offset. After invoking a subsequent phase offset, pulse width control is no longer available. To change the output pulse width after executing the first subsequent phase offset, the user must issue a Q divider synchronization (see the Distribution Output Clock Synchronization section), at which point the Q divider is ready to accept a new initial phase offset, a new output pulse width, and subsequent phase offsets.

DISTRIBUTION N SHOT/PRBS OUTPUT CLOCKING

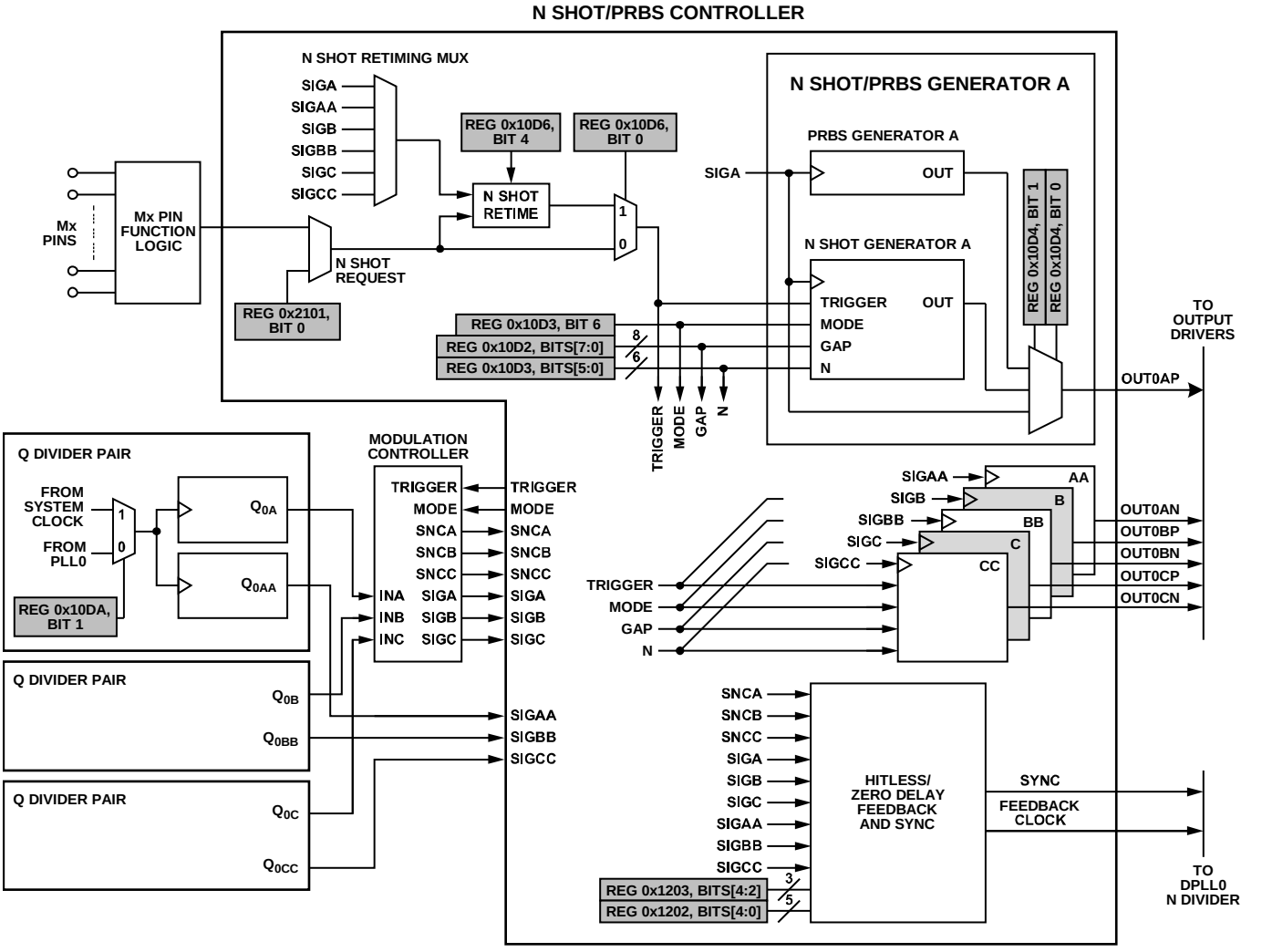
N SHOT/PRBS CLOCKING OVERVIEW

The Q dividers work in conjunction with several controllers, one of which is the N shot/pseudorandom binary sequence (PRBS) controller (see Figure 67). The N shot/PRBS controller provides the AD9546 with the capability of generating SYSREF and device clock clocks per the JESD204B standard as well as gapped and pseudorandom clock signals.

Figure 68 shows a block diagram of the N shot/PRBS clock controller, which includes a more detailed representation of the relationship between the Q dividers, the modulation controller and the N shot/PRBS controller. Although Figure 68 is specific to PLL0, it is also representative of PLL1, because there are dedicated controllers for each PLL channel.

Regarding the Q dividers, Qxy denotes a specific Q divider in the clock distribution section, whereas Q_{xy} denotes the output of divider Qxy (where x is 0 and y is A, AA, B, BB, C, or CC). For example, Q_{0A} is the output of Q divider Q0A.

Note that the Q_{0AA}, Q_{0BB}, and Q_{0CC} outputs route directly to the N shot/PRBS controller, whereas Q_{0A}, Q_{0B}, and Q_{0C} route through the modulation controller. As such, the Q_{0A}, Q_{0B}, and Q_{0C} outputs are the only outputs subject to modulation (see the Distribution Embedded Output Clock Modulation section). However, disabled (default) modulation channels have an internal bypass allowing the Q_{0A}, Q_{0B}, and Q_{0C} outputs to pass directly through the modulation controller.



NOTES
1. A RANGE OF BITS USES A COLON SEPARATOR
2. REGISTER ADDRESSES SHOWN ARE SPECIFIC TO Q DIVIDER PAIR Q0x (WHERE x = A, AA, B, BB, C, OR CC), PLL CHANNEL 0, N SHOT/PRBS GENERATOR A, AND TRANSLATION PROFILE 0.0
3. SEE THE MODULATION SYNCHRONIZATION SECTION FOR DETAILS ABOUT THE SNCx AND SIGx SIGNALS

Figure 68. Block Diagram of JESD204B/Gapped Clock Controller

Each Q divider has a dedicated pair of generators: one for generating randomized clock signals (PRBS) and another for supporting burst or gapped clock applications (N shot), as shown by the expanded detail of N shot/PRBS Generator A in Figure 68. Two bits are associated with each Q divider with one bit for the N shot function and one bit for the PRBS function per Table 68.

Table 68. Register Address and Bit Assignments for the N Shot and PRBS Functions

Q Divider	Register Address	N Shot Function Bit	PRBS Function Bit
Q0A	0x10D4	D0	D1
Q0AA	0x10D4	D2	D3
Q0B	0x10D4	D4	D5
Q0BB	0x10D4	D6	D7
Q0C	0x10D5	D0	D1
Q0CC	0x10D5	D2	D3
Q1A	0x14D4	D0	D1
Q1AA	0x14D4	D2	D3
Q1B	0x14D4	D4	D5
Q1BB	0x14D4	D6	D7

Table 69 shows how the PRBS and N shot function bit values select different output clock functions.

Table 69. N Shot/PRBS Output Clock Function Selections

PRBS Function	N Shot Function	Output Clock Function
0	0	Normal clock (Q divider)
0	1	Burst or gapped clock
1	0	Randomized clock
1	1	Not applicable

Although it is physically possible to set both function bits to Logic 1, the user must avoid doing so because Logic 1 is an undefined mode and may result in unspecified device behavior.

RANDOMIZED CLOCK (PRBS)

When the randomized clock function is in effect, the output clock signal is a pseudorandom sequence of high and low output levels generated at the rate of the associated Q divider. Figure 69 shows a representation of the output clock signal.

The spectrum of a normal clock signal (square wave) is a series of line spectra consisting of the fundamental clock frequency and its associated harmonics. The main purpose of a PRBS clock (or spread spectrum clock) is to broaden the spectral line of the fundamental while at the same time reducing the magnitude of the harmonic spectral lines.

Because the user can program more than one output of the AD9546 as a spread spectrum clock, each of the 10 PRBS generators produces a unique PRBS sequence, which prevents the possibility of correlated crosstalk between outputs enabled for PRBS clock generation. All 10 PRBS generators use 17^{th}-order generator polynomials, which means the PRBS sequence is inherently periodic over a period of $2^{17} - 1$ (131,071) Q divider output cycles.

N SHOT (JESD204B AND GAPPED CLOCKING)

N Shot Overview

The ability of the AD9546 to generate JESD204B clock signals and gapped clock signals derives from the N shot generators. Each N shot generator operates at the rate of the Q divider output. The user has control over whether the N shot generator produces a periodic output clock pattern or a single-burst pattern. In either case, triggering a pattern is via Bit 0 of Register 0x2101 for PLL0 and Register 0x2201 for PLL1. Alternatively, an external signal can serve as the N shot trigger via an Mx pin (see the Status and Control Pins section).

As shown in Figure 68, the N shot parameters (trigger, mode, gap, and N) apply to all N shot generators in a PLL channel. As such, the N shot parameters are common to all outputs in a channel that have burst or gapped clock functionality enabled (per Table 69).

Using the N shot generators imposes a minimum divide ratio of 8 on Q dividers configured for N shot functionality.

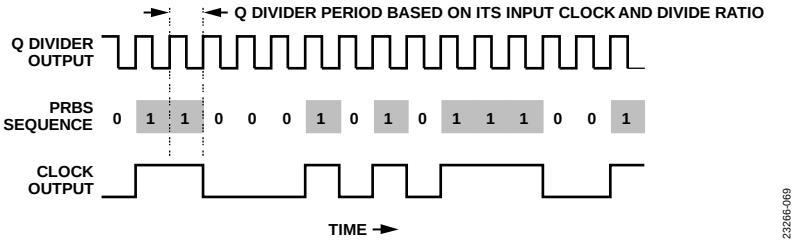


Figure 69. PRBS Clock Signal

N Shot Pattern Generation

An N shot pattern consists of a specified clock segment and gap segment. To assign the desired number of clock segment cycles, use the 6-bit unsigned integer in Bits[5:0] of Register 0x10D3 for PLL0 and Register 0x14D3 for PLL1. To assign the desired number of gap segment cycles, use the 8-bit unsigned integer in Bits[7:0] of Register 0x10D2 for PLL0 and Register 0x14D2 for PLL1.

As shown in Figure 70, the clock and gap segments are in units of the Q divider output period. The N shot generator treats a clock segment value of 0 as comprising zero clock cycles and a gap segment value of 0 as comprising 1 clock cycle. Thus, the net effect of programming a clock segment of 0 and a gap segment of 0 is no output.

The N shot generator is capable of burst or periodic gapped operation. To select burst or periodic operation, use Bit 6 of Register 0x10D3 for PLL0 and Register 0x14D3 for PLL1. When Bit 6 = 0, the N shot generators operate in burst mode (see Figure 71). When Bit 6 = 1, the N shot generators operate in periodic gapped mode (see Figure 72).

In burst operation, the rising edge of the trigger signal constitutes the trigger event. Prior to the burst, the generator is in the default state (Logic 0), which the generator holds until triggered (see the N Shot Triggering section).

When triggered, the N shot generator produces an output sequence (see the bottom trace of Figure 70). At the end of the sequence, the N shot generator remains in its default state (Logic 0).

In burst operation, the output appears as a sequence of Q divider cycles per the programmed clock segment value (in Figure 70, the clock segment is 6). The gap period is effectively an extension of the default output condition, which is the low state (in Figure 70, the gap segment is 10).

In periodic operation, the logic level of the trigger matters (rather than the rising edge as in burst mode). The N shot generator, starting from its default state (Logic 0), waits for the trigger to assume a Logic 1 state. At this point, the N shot generator begins repeatedly generating the burst pattern (clock segment pulses, then gap segment pulses), per Figure 72. The N shot generator continues generating burst patterns until the trigger assumes a Logic 0 state. The occurrence of the Logic 0 state informs the generator to stop synchronously. That is, if the generator is not in the gap portion of the burst, the generator stalls at the end of the current pulse as shown in Figure 73. Furthermore, returning the trigger signal to Logic 1 causes the pattern to resume from where it previously stopped (that is, with the absent pulses shown in Figure 73).

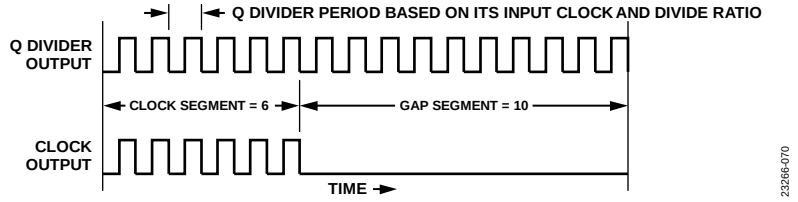


Figure 70. N Shot Plus Gap Clock Signal

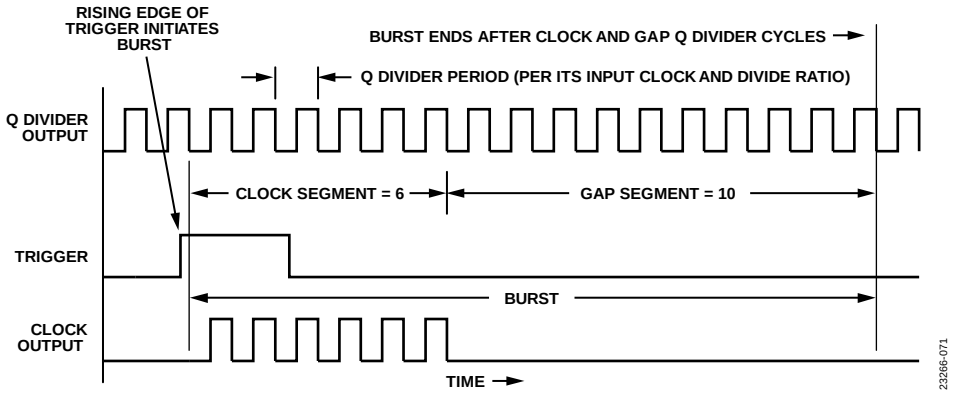


Figure 71. Burst Clock Signal

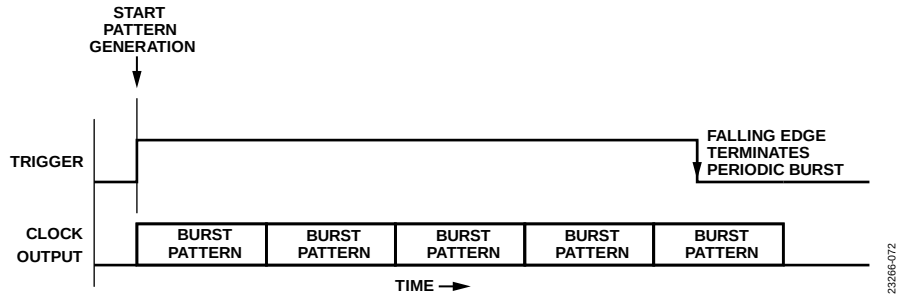


Figure 72. Periodic Gapped Clock Signal

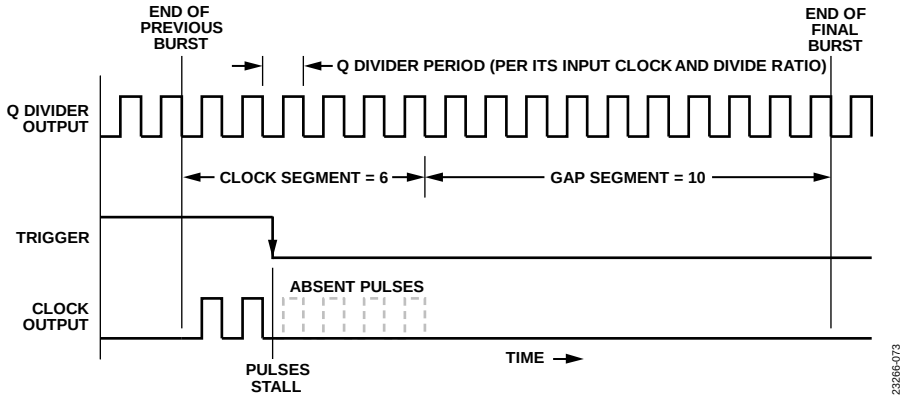


Figure 73. Stalling a Periodic Gapped Clock Signal

N Shot Triggering

The N shot trigger signal originates from one of two sources.

- Bit 0 of Register 0x2101 or Register 0x2201
- An external signal applied via an Mx pin

In either case, a Logic 1 constitutes a trigger event. In general, the user must apply Logic 1 to trigger the N shot generators and then return the trigger source to Logic 0. Otherwise, holding the trigger source in a Logic 1 state indefinitely may lead to unwanted retriggering of the N shot generators on subsequent distribution synchronization events (see the Distribution Output Clock Synchronization section).

The N shot generators respond to the trigger signal based on Bit 6 of Register 0x10D3 or Register 0x14D3. As described in the N Shot Pattern Generation section, Bit 6 makes the trigger input of the N shot generators edge or level sensitive for generating burst or periodic gapped clock signals, respectively (per Figure 71 and Figure 72).

The trigger mechanism for delivering a trigger signal to the N shot generators appears in the upper left section of Figure 68. The N shot generators support two triggering methods: direct and retimed.

To select the desired triggering method, use Bit 0 of Register 0x10D6 for PLL0 or Register 0x14D6 for PLL1. Logic 0 (default) selects direct, whereas Logic 1 selects retimed. For a description of Bit 4 of Register 0x10D6 in Figure 68, see the Modulation Trigger section.

For the direct triggering method, the trigger signal applies directly to the trigger input of the N shot generators. Thus, the trigger signal is the trigger event.

Note that in the following paragraphs, the terms slowest and fastest appear in reference to the output clock signal of the Q dividers. Slowest and fastest refers to the largest and smallest Q divider phase offset value, respectively (see the Initial Phase Offset section).

For the retimed triggering method, the trigger signal routes to the N shot retime block instead of directly to the N shot generators. The rising edge of the trigger signal initializes the retiming block such that the retiming block waits for the rising edge of the slowest Q divider enabled for N shot operation (user specified). The trigger signal, qualified by the rising edge of the slowest Q divider, constitutes a retimed trigger event. The retiming block sends a trigger signal to the N shot generators coincident with the retimed trigger event as shown in Figure 81.

The retimed trigger event occurs with a latency of three rising edges of the slowest Q divider. Furthermore, when using the retimed trigger mechanism, the associated Q dividers must have a divide ratio of at least 32. A minimum setup time of 48 Q divider input half cycles is required between the retiming output (slowest) and the subsequent N shot enabled output (fastest).

The retimed trigger is on the rising edge of the slowest N shot enabled Q divider to accommodate multiple N shot generators that are providing multiple output clocks. Using the rising edge of the slowest Q divider output as a retiming mark ensures that all N shot generators begin clocking with the fastest output as the earliest of the group, even when the Q dividers have different programmed phase offsets and regardless of when the N shot request occurs.

The AD9546 automatically selects the appropriate N shot enabled Q divider for trigger retiming, such that the output with the largest Q divider phase offset value is always the retiming clock.

DISTRIBUTION EMBEDDED OUTPUT CLOCK MODULATION

MODULATION CONTROLLER OVERVIEW

The AD9546 has the capability to embed a low frequency clock within a high frequency carrier. Referring to Figure 68, only the primary output of a Q divider pair routes to the modulation controller, whereas the secondary output bypasses the modulation controller and routes directly to the N shot/PRBS controller. Thus, only the primary distribution clock outputs support embedded clock modulation capability (for example, Output OUT0AP supports modulation, whereas Output OUT0AN does not).

Embedded clock modulation consists of changing the pulse width of the designated Q divider output clock in synchrony with a binary modulation signal to produce modulation events. A modulation event always spans two Q divider clock cycles where the first clock cycle changes duty cycle, but the second clock cycle may or may not change duty cycle (see the Balanced and Unbalanced Modulation section).

An expanded diagram of the modulation controller appears in Figure 74. Although Figure 74 is specific to PLL0, it is also representative of PLL1, because there is a dedicated modulation controller for each PLL channel.

Any one or more Q divider outputs with a single-letter subscript (for example, Q_{0A} but not Q_{0AA}) can operate as an embedded clock modulator. To enable embedded modulation, use Bit 0 of the registers shown in Table 70.

Table 70. Enable Embedded Modulation Register Address

Q Divider	Register Address
Q0A	0x10CF
Q0B	0x10D0
Q0C	0x10D1
Q1A	0x14CF
Q1B	0x14D0

Logic 1 selects the designated Q divider for embedded clock modulation, whereas Logic 0 (default) bypasses the modulation controller (via the mux in Figure 74).

Modulation control consists of two parameters: Δt and t_{MOD}. Parameter Δt defines the desired magnitude of the modulation edge variation, and t_{MOD} defines the modulation period (see Figure 75). The modulated signal consists of a time step of magnitude Δt occurring at regular intervals of period, t_{MOD}. The magnitude parameter, Δt, is common to all the modulators within a PLL channel, whereas the period parameter, t_{MOD}, is unique to each modulator.

MODULATION MAGNITUDE

To set the modulation magnitude, use the 16-bit unsigned integer (modulation step) in Register 0x10C0 to Register 0x10C1 for PLL0 and Register 0x14C0 to Register 0x14C1 for PLL1. The modulation step value carries units of one-half of the period of the input clock to the Q divider associated with the modulator, yielding the following relationship:

$$D = Modulation\ Step/(2 \times Qxy) \quad (3)$$

where:
D is the duty cycle deviation (that is, the time deviation of the modulation edge from nominal, normalized to the Q divider output period). Figure 75 shows *D* as $\Delta t/t_Q$.
Qxy is the divide ratio of the relevant Q divider (*x* is 0 or 1, and *y* is A, B or C).

Because the modulation step is common to all the modulators in a PLL channel, whereas Qxy is unique to each Q divider, D in Equation 3 is not necessarily the same for all the modulators in a PLL channel.

Given the divide ratio for Q divider Q0A is 1001, find the modulation step bit field value necessary for Modulator A to yield 5% modulation.

Modulation of 5% implies D = 0.05. Substituting the appropriate values into Equation 3 yields

$$0.05 = Modulation\ Step/(2 \times 1001)$$

Therefore,

$$Modulation\ Step = 100 \text{ (rounded to nearest integer)}$$
$$= 0x64 \text{ (hexadecimal)}$$

Given the same modulation step value as in the preceding example (modulation step = 100), find the modulation magnitude for Modulator B assuming Q0B has a divide ratio of 8025.5.

$$D = Modulation\ Step/(2 \times Qxy)$$
$$= 100/(2 \times 8025.5)$$
$$= 0.00623\ (0.623\%)$$

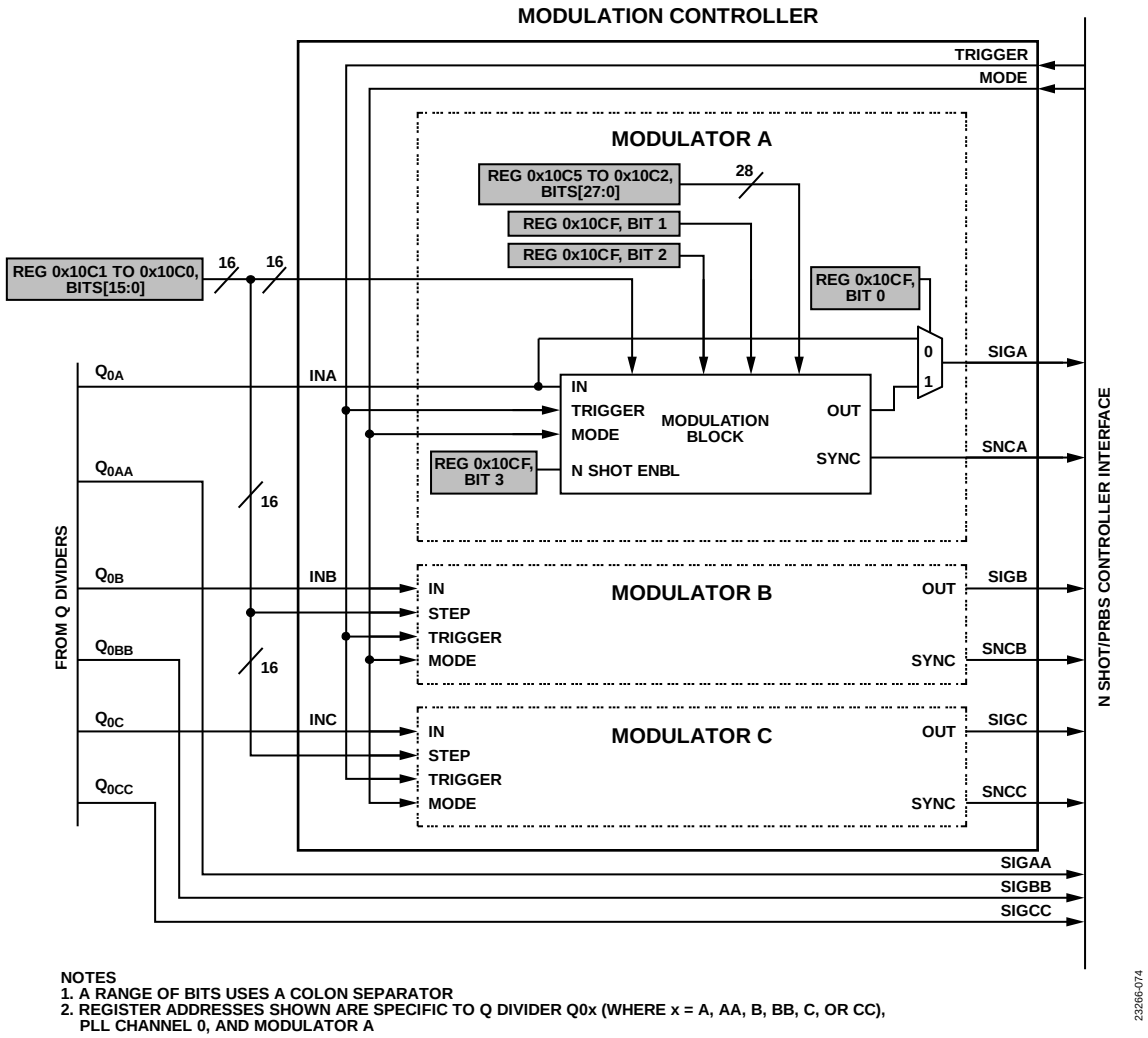


Figure 74. Block Diagram of Embedded Clock Modulation Controller

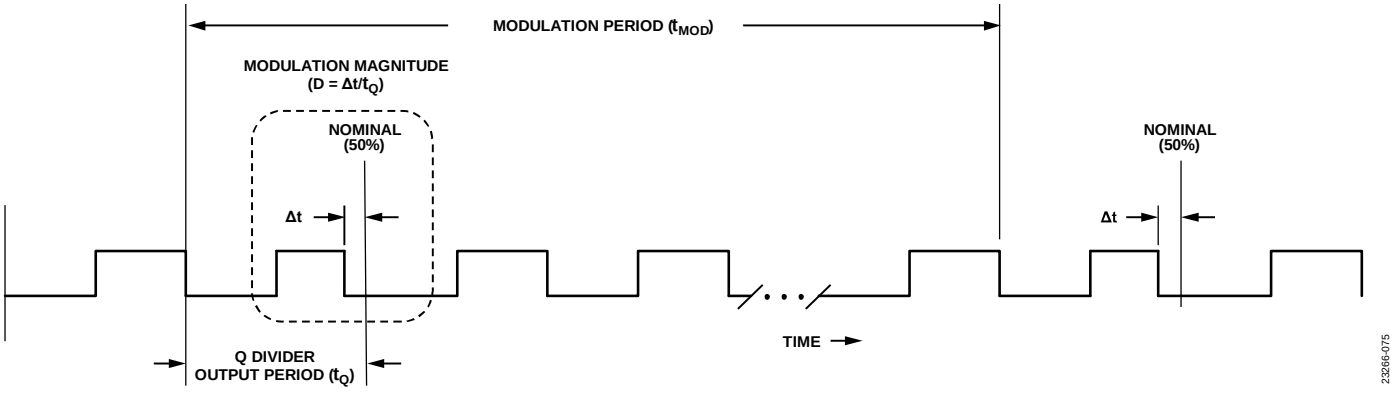


Figure 75. Modulation Magnitude and Period

MODULATION PERIOD

The modulation period is the interval between modulation events (see Figure 75). To control the modulation period parameter, use the 28-bit unsigned integer (modulation counter) in the register address ranges shown in Table 71. The modulation counter value carries units of the period of the output clock of the associated Q divider.

The modulation counter value must be greater than or equal to six.

Table 71. Modulation Period Address Ranges

Modulator	Register Address
Q0A	0x10C2 to 0x10C5
Q0B	0x10C6 to 0x10C9
Q0C	0x10CA to 0x10CD
Q1A	0x14C2 to 0x14C5
Q1B	0x14C6 to 0x14C9

The three lower addresses for each Q divider in Table 71 carry the 24 LSBs of the 28-bit integer, and Bits[3:0] of the upper address carry the four MSBs of the 28-bit integer (Bits[7:4] are unused).

The modulation period (t_{MOD}) depends on the input clock frequency (f_{IN}), the divide ratio of the associated Q divider, and the modulation counter value.

$$t_{MOD} = Modulation\ Counter \times (Qxy/f_{IN}) \quad (4)$$

Suppose the input clock to Q Divider Q0A comes from APLL0 with its VCO operating at 2.38 GHz. Because the VCO drives a divide-by-2 block prior to the Q divider clock input, the Q divider input frequency (f_{IN}) is 1.19 GHz. Find the value of the modulation counter value required to yield a modulation period (t_{MOD}) of 1 ms (0.001 sec) for a Q divide ratio of 107.5.

Substituting the appropriate values into Equation 4 yields

$$10^{-3} = Modulation\ Counter \times (107.5/(1.19 \times 10^{9}))$$

Therefore,

$$Modulation\ Counter = 11{,}070 \text{ (nearest integer)}$$

$$= 0x\ 0000\ 2B3E \text{ (hexadecimal)}$$

The desired modulation period (t_{MOD}) is 1 ms. The modulation counter value must be an integer. However, it is not always possible to produce the desired modulation period exactly. In this example, the actual t_{MOD} value is 1.0000210084 ms (per Equation 4).

BALANCED AND UNBALANCED MODULATION

Modulation consists of periodic modulation events occurring at regular intervals, t_{MOD}, with a single modulation event spanning two output clock cycles of the Q divider associated with the modulator. The modulator applies the specified pulse width variation (per the modulation step and modulation counter) to one or both clock cycles of the modulation event. Modulation of both clock cycles constitutes balanced modulation, whereas modulation of only one clock cycle constitutes unbalanced modulation.

With balanced modulation, the modulator applies opposite polarity to the modulation steps of the two consecutive cycles of the modulation event (see Figure 76). As such, balanced modulation maintains the average dc level of the waveform at its nominal 50% amplitude point.

With unbalanced modulation, the modulator applies the modulation step (Δt) to only the first pulse of the modulation event, leaving the second pulse unaltered (see Figure 77) and resulting in a waveform with a dc level slightly less than its nominal 50% amplitude point.

To select between unbalanced and balanced modulation, use Bit 2 per the register address in Table 70 for a given Q divider. Logic 0 (default) selects balanced modulation, whereas Logic 1 selects unbalanced modulation.

By default, the first pulse of a modulation event has −Δt (and the second pulse +Δt if balance modulation is in effect). However, the user can force alternate polarity, where the first pulse of a modulation event has +Δt (and the optional second pulse has −Δt). Polarity control is via Bit 1 per the register address in Table 70 for a given Q divider. Logic 0 (default) applies negative polarity to Δt for the first pulse of a modulation event, whereas Logic 1 applies positive polarity to Δt for the first pulse (see Figure 78), which causes a positive shift in dc offset in the case of unbalanced modulation.

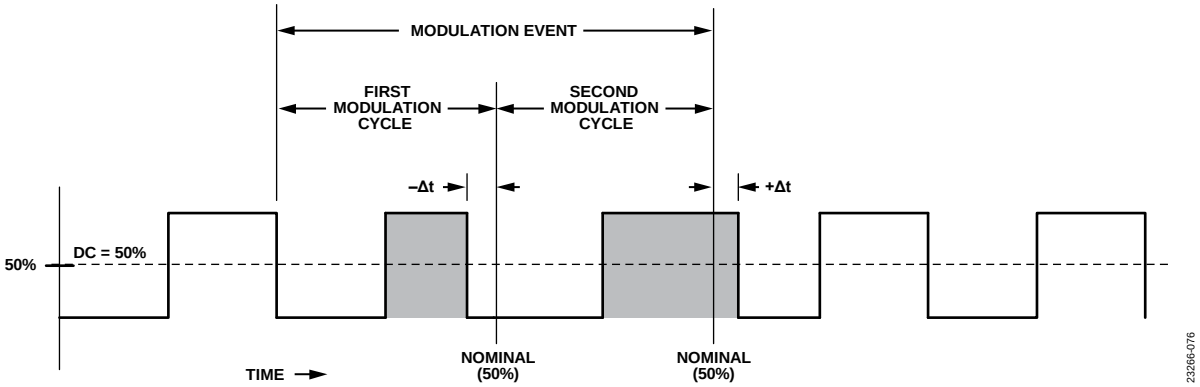


Figure 76. Balanced Modulation Waveform

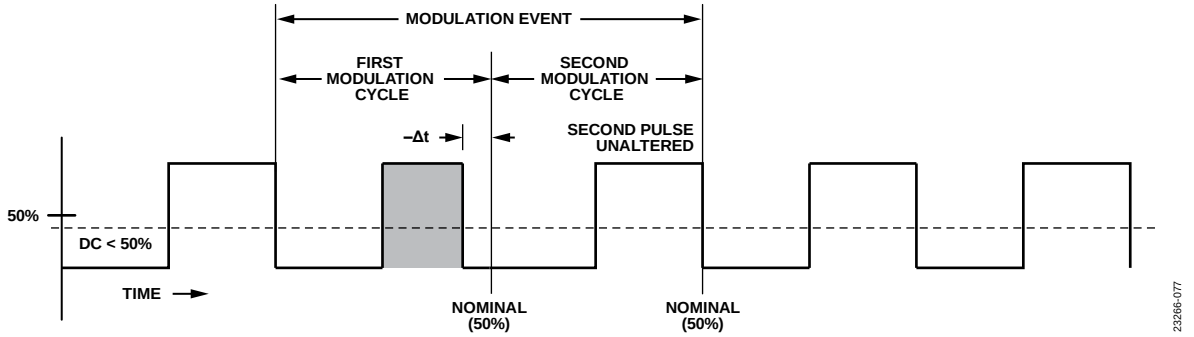


Figure 77. Unbalanced Modulation Waveform

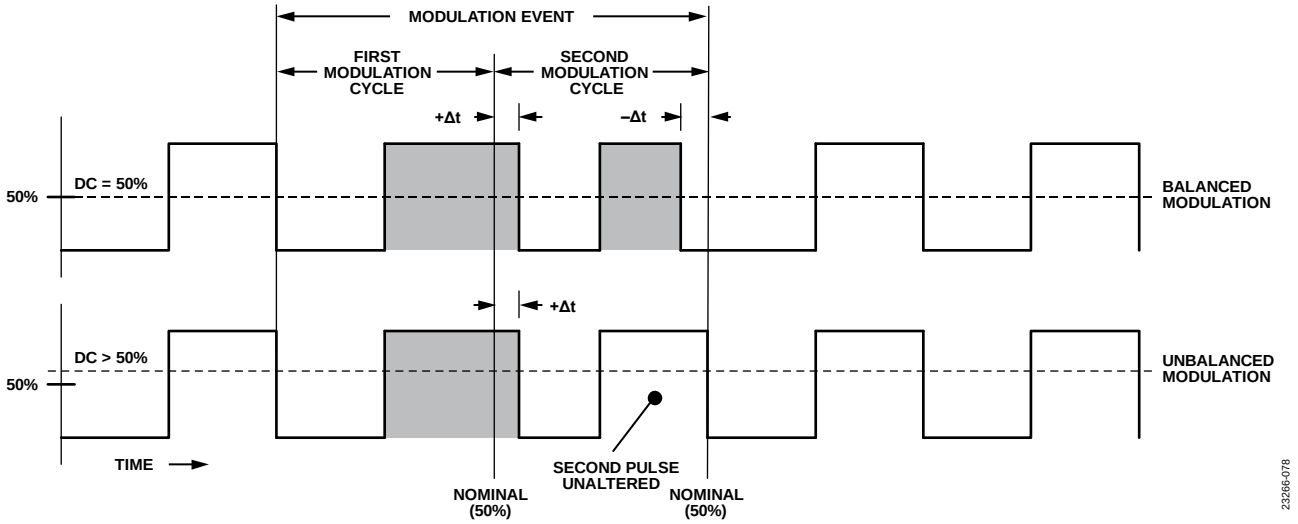


Figure 78. Alternate Modulation Polarity

MODULATION SYNCHRONIZATION

The periodic modulation events produced by the modulators are useful as timing markers for synchronization purposes. Both the balanced and unbalanced modulation schemes use two Q divider output cycles to define a modulation event (see Figure 76 and Figure 77). In the case of unbalanced modulation, the second cycle is unaltered but is still part of the modulation event. As such, the first rising edge of the Q divider output following the second modulated cycle marks the base edge for synchronization purposes (see Figure 79).

The synchronization information available from the modulators requires the use of a secondary signal path. This path is the reason the modulators have SIGx and SNCx output signals. The SIGx signals are the modulated waveforms that ultimately propagate to the distribution output pins.

The SNCx signals are internal only and carry synchronization information attached to the modulation events corresponding with the associated SIGx signal (for example, SNCA associates with SIGA). The SNCx signal is a pulse coincident with the base edge of the corresponding SIGx signal, as shown in Figure 80.

As shown in Figure 68, the SNCx signals route to the DPLL via the hitless/zero delay feedback and synchronization block of the N shot/PRBS controller. The SNCx signals provide synchronization to the N divider of the corresponding DPLL channel. The device automatically selects the appropriate SNCx signal for the Q divider chosen for feedback (when modulation is in effect for that particular Q divider). The SNCx selection depends on Bits[4:2], the 3-bit tag mode bit field in the registers associated with tag modes in the translation profiles (see the Translation Profiles section) per Table 72.

Table 72. Tag Mode Register Address

Translation Profile	Register Address
0.0	0x1203
0.1	0x1223
0.2	0x1243
0.3	0x1263
0.4	0x1283
0.5	0x12A3
1.0	0x1603
1.1	0x1623
1.2	0x1643
1.3	0x1663
1.4	0x1683
1.5	0x16A3

Because the SNCx signal identifies the modulation base edge, the user can synchronize the DPLL N divider with a specific synchronization edge per Figure 80. To specify a synchronization edge (0, 1, 2, or 3) for the N divider, use Bits[1:0] of Register 0x10CE for PLL0 and Register 0x14CE for PLL1. The decimal value of the bit pair corresponds to the desired synchronization edge in Figure 80.

The choice of synchronization edge (0, 1, 2, or 3) puts an additional constraint on the value of the modulation counter (see the Modulation Period section) as follows:

Modulation Counter ≥ *Sync Edge* + 7

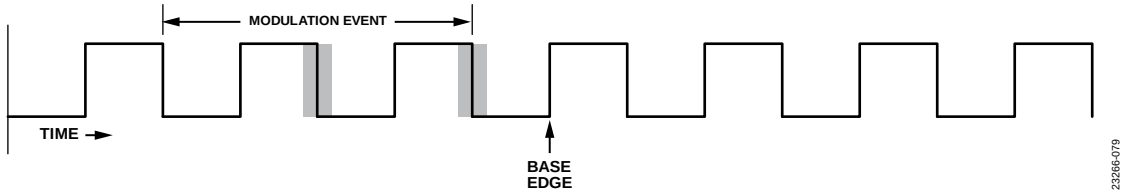


Figure 79. Modulation Synchronization Edges

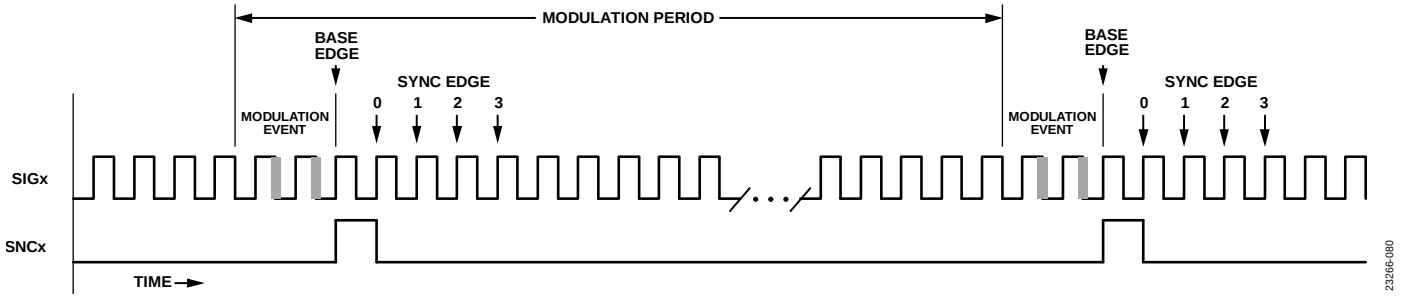


Figure 80. Modulation Synchronization

MODULATION TRIGGER

By default, when modulation is in effect for a given Q divider, the modulator applies the modulation immediately. However, it is also possible to delay modulation until triggered. To enable this feature for a given Q divider, use Bit 3 in the register address shown in Table 70. Logic 0 (default) disables the triggering feature, whereas Logic 1 enables it.

When enabled, the modulators use the trigger and mode signals originating within the N shot/PRBS controller (see Figure 68). The mode behavior differs slightly from that of the N shot/PRBS controller. Specifically, when N shot request mode (Bit 6 in Register 0x10D3 for PLL0 or Register 0x14D3 for PLL1) = 0, instead of the modulator continuously generating periodic modulation events, it generates only five periodic modulation events and then stops. Conversely, when N shot request mode = 1, the modulator continuously generates modulation events when the N shot request signal (see Figure 68) is Logic 1. This behavior allows synchronous enable or disable of modulation events without losing synchronization during modulation.

The modulators, like the N shot generators, provide a retimed trigger option (see the N Shot Triggering section). However, in the case of the modulators, retiming relates to modulation events rather than the carrier clock edges associated with the Q divider. By default, the retiming block retimes to the carrier clock edges. To select retiming to modulation events, program Bit 4 to Logic 1 in Register 0x10D6 (PLL0) or Register 0x14D6 (PLL1).

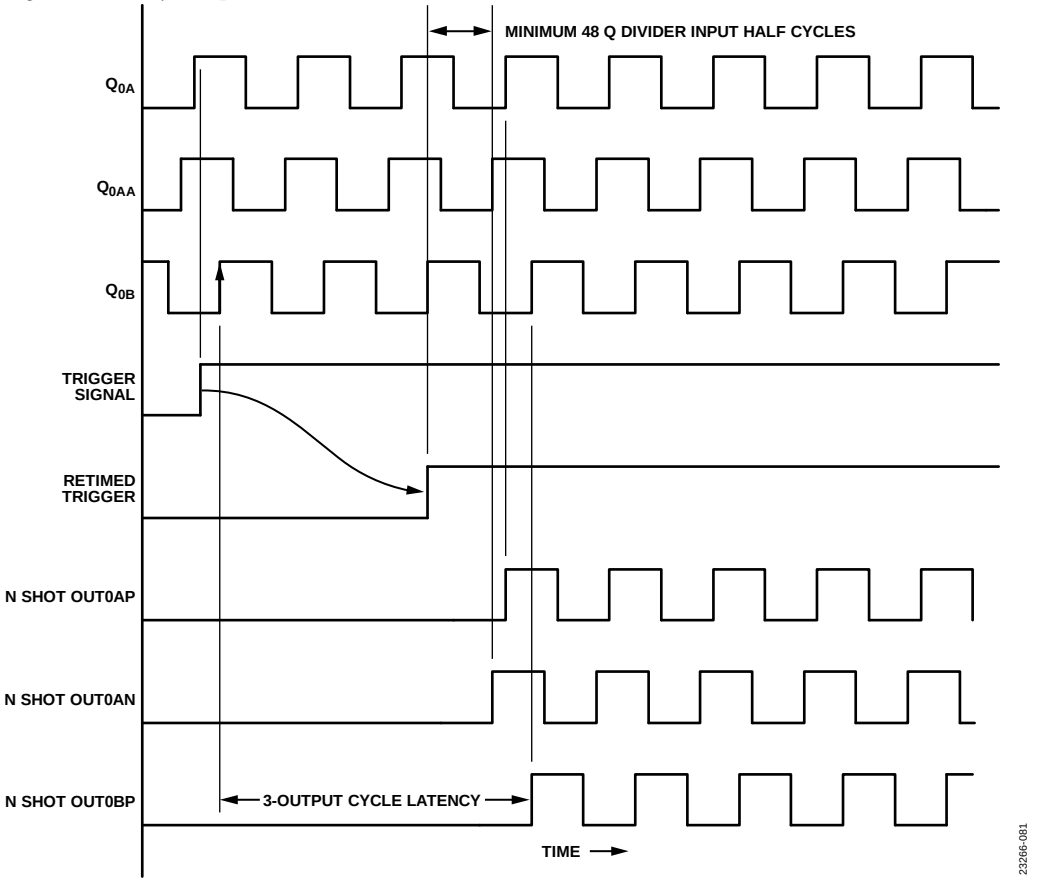


Figure 81. Trigger Retiming (with Q0AA as the Retiming Q Divider)

DISTRIBUTION OUTPUT CLOCK SYNCHRONIZATION

SYNCHRONIZATION OVERVIEW

The AD9546 has a built in synchronization controller that provides the user with a flexible and sophisticated means to synchronize its multiple output clock signals. The synchronization controller properly sequences internal events to ensure proper synchronizing of the output clocks of the device.

There are three ways to trigger the synchronization sequence.

- Manually (via Bit 3 of Register 0x2000, via Bit 3 of Register 0x2101 for PLL0 and Register 0x2201 for PLL1, via an op code in an EEPROM download, or via an Mx pin)
- Autoreconfiguration (changes to Q divider values)
- Autosynchronization

Autosynchronization is the preferred option for output clock synchronization.

A synchronization trigger event is channel specific, applying to either PLL0 or PLL1. The exception is manual synchronization, which can apply to both channels via Bit 3 of Register 0x2000. Assertion of Bit 3 does not imply synchronization of both channels relative to each other but causes a simultaneous triggering of the synchronization process for both channels.

A synchronization trigger initiates the synchronization sequence. Following a trigger event, the synchronization controller checks whether Bit 2 of Register 0x10DB or Register 0x14DB is set. If so, the synchronization controller waits for a hitless profile to become valid before muting all outputs (because reference synchronization is only viable for zero delay/hitless translation modes).

Next, the synchronization controller confirms that the output drivers are muted and then puts the P dividers and Q dividers in a reset state (for the controller to load new Q divider configuration information, for example, a new divide ratio). Then the synchronization controller again checks whether Bit 2 of Register 0x10DB or Register 0x14DB is set so the controller can align the release of the output clocks with a rising edge of the active reference. However, if Bit 2 of Register 0x10DB or Register 0x14DB bit is clear, the synchronization controller checks for release of manual synchronization (that is, a clearing of Bit 3 of Register 0x2000, or Bit 3 of Register 0x2101 (for PLL0) or Bit 3 of Register 0x2201 (for PLL1). The synchronization controller also checks that the APLL indicates calibrated and locked status before proceeding.

Next, the synchronization controller unmutes the output drivers and releases the appropriate Q dividers from their reset state. Finally, the synchronization controller returns to waiting for a subsequent synchronization trigger.

The output clock synchronization sequence initiates only when the system clock is locked and stable.

OUTPUT SIGNALS AFTER POWER-UP OR RESET

A particularly important aspect of the synchronization controller involves the control of the output clocks following a device power-up or reset. On power-up or reset, the synchronization controller waits for a synchronization trigger before activating the output clocks. Until a synchronization trigger occurs, the synchronization controller holds all the distribution Q dividers in a reset state, which prevents any clock signals from appearing at the OUTxyN and OUTxyP pins. Therefore, the user must initiate the synchronization sequence to have any clock signals appear at the output.

MANUAL SYNCHRONIZATION TRIGGER

A manual synchronization trigger occurs when the user sets one of three manual synchronization bits. Programming Bit 3 = 1 of Register 0x2000 initiates the synchronization sequence for all the AD9546 output clocks. Alternatively, the user can synchronize only those outputs associated with one of the PLL channels by programming Bit 3 = 1 of Register 0x2101 for PLL0 or Bit 3 = 1 of Register 0x2201 for PLL1. The same synchronization features available via these bits are also available via any of the Mx pins configured as an input.

Setting any of the D3 bits causes the synchronization sequence to execute up to the point just before unmuting all drivers. At that point, the synchronization controller stalls, waiting for the user to clear the appropriate manual synchronization bit. This delay allows the user to update the Q divider ratios before the controller unmutes the output drivers, which does not occur until the user clears the relevant manual synchronization bit.

AUTORECONFIGURATION SYNCHRONIZATION TRIGGER

An autoreconfiguration synchronization trigger occurs when the user changes any of the Q divider ratios and subsequently sets the IO update bit. However, the autoreconfiguration synchronization trigger applies only after the synchronization controller has already completed a synchronization sequence as the result of a power-up (or reset). After a power-up (or reset) synchronization sequence completes, any subsequent updates to the Q divider ratios constitutes an autoreconfiguration synchronization trigger.

Autoreconfiguration is a convenience feature, because it synchronizes the outputs automatically when the user changes one or more Q divider values and subsequently sets the IO update bit. This feature alleviates the need to initiate a manual synchronization trigger to change Q divider values. Be aware, however, that changing a single Q divider value affects all the clock signals of the corresponding PLL channel. Specifically, changing the value of any Q0x (where x = A, AA, B, BB, C, or CC) results in disturbing all the Output 0 clock outputs because it initiates a synchronization sequence for those outputs. Likewise, changing the value of any Q1x divider (where x = A,

AA, B, or BB) results in disturbing all the Output 1 clock outputs. These output clock disturbances result from the synchronization process. However, the disturbances are glitch free, because the synchronization sequence mutes the affected output drivers during synchronization.

The aforementioned output clock disturbances are avoidable, however, when the clock source to the Q dividers is the system clock (see Table 62) and the system clock synchronization mask bits are Logic 1 (see the Q Divider Clock Source Selection section) for a subset of the device outputs.

AUTOSYNCHRONIZATION TRIGGER

Although the AD9546 provides the user with the ability to synchronize the distribution outputs via a manual synchronization trigger, it is not a recommended action. Instead, the AD9546 provides an autosynchronization trigger feature that applies only after meeting certain conditions, which overcomes potential usability issues associated with manual trigger scenarios. The immediate autosynchronization mode (see Table 73) is an effective replacement for manual synchronization triggering.

The user selects the desired autosynchronization trigger conditions via Bits[1:0] of Register 0x10DB for PLL0 and Register 0x14DB for PLL1. Table 73 shows the selectable autosynchronization conditions.

Table 73. Autosynchronization Mode Selections

Bits[1:0] (Decimal)	Autosynchronization Mode
0	Disabled (default)
1	Immediate
2	On DPLL phase lock
3	On DPLL frequency lock

Autosynchronization enables the AD9546 to execute a distribution synchronization sequence without any user intervention. The nonzero autosynchronization selections specify conditions for generating an autosynchronization trigger.

When autosynchronization mode = 0 (default), only manual or autoreconfiguration triggers are possible. When autosynchronization mode ≠ 0, the autosynchronization trigger feature is in effect.

When autosynchronization is active (autosynchronization mode ≠ 0), nothing happens until the system clock PLL indicates a locked status. The autosynchronization mechanism then enters an autosynchronization trigger detection loop awaiting the specified autosynchronization event. When autosynchronization mode = 1, the autosynchronization event constitutes the indication of a locked status by the system clock PLL. When autosynchronization mode = 2, the autosynchronization event constitutes an indication of phase lock by the DPLL.

When autosynchronization mode = 3, the autosynchronization event constitutes an indication of frequency lock.

After the autosynchronization trigger occurs for a PLL channel, the autosynchronization trigger generation sequence terminates. Subsequent autosynchronization sequences on the same PLL channel are not possible unless the user programs Bit 0 = 1 of Register 0x2107 or Register 0x2207, or by updating Bits[1:0] of Register 0x10DB or Register 0x14DB.

REFERENCE SYNCHRONIZATION

Output clock synchronization establishes a coincident starting time among output clocks (usually on a per PLL channel basis). Output clock synchronization, however, is generally asynchronous with respect to input reference clock signal edges. The reference synchronization feature allows the user to make a rising edge of the input reference clock an additional gating item for synchronization of the output clocks. Reference synchronization establishes a small fixed phase relationship (~30 ns) between the input and output clocks (assuming the output clock frequency is an integer multiple of the input reference frequency). Establishing an initial fixed phase relationship between the input and output clocks minimizes the DPLL acquisition time by making it deterministic rather than completely random.

The reference synchronization feature requires a hitless profile be active when reference synchronization is in effect. The reference synchronization feature does not function with a phase buildout profile.

Like most of the other synchronization features, reference synchronization applies on a per PLL channel basis. To enable the reference synchronization feature, use Bit 2 (reference synchronization) of Register 0x10DB for PLL0 and Register 0x14DB for PLL1. Setting the reference synchronization bit, which is Logic 0 by default, enables reference synchronization of those outputs associated with the corresponding PLL channel.

When using reference synchronization in conjunction with autosynchronization mode (that is, autosynchronization mode = 1, 2, or 3), be sure to program the reference synchronization bit to Logic 1 and assert the I/O update bit prior to programming autosynchronization mode = 1, 2, or 3.

FREQUENCY TRANSLATION LOOPS

FREQUENCY TRANSLATION LOOPS OVERVIEW

The frequency translation capability of the AD9546 comprises two independent dual-PLL channels (Channel 0 and Channel 1). Each PLL channel comprises a DPLL followed by an APLL. Both channels have programmable reference dividers at the input and programmable channel dividers at the output. Figure 82 shows a block diagram of a single channel. REFx and REFy can each be any of the reference inputs, but not the same reference input.

The DPLL channels (DPLL0 and DPLL1) are capable of low loop bandwidths (μHz) well suited for jitter cleanup applications involving the 1 pulse per second (pps) input signals that originate from global positioning system (GPS) and global navigation satellite system (GNSS) receivers. The DPLL is of the fractional-N variety but is capable of Integer N operation when the fractional part is set to zero.

The DPLL provides an output frequency of up to approximately 400 MHz. The DPLL also plays a role in the automatic reference switching capability of the AD9546 and its ability to provide hitless and phase buildout functionality.

The APLL is of the Integer N variety with an integrated VCO and provides the high frequency upconversion capability of the cascaded DPLL/APLL pair. Unlike the two DPLLs, which are identical, the two APLLs differ slightly. Specifically, they have nonoverlapping VCO ranges with APLL0 spanning 2.424 GHz to 3.232 GHz and APLL1 spanning 3.232 GHz to 4.040 GHz.

TRANSLATION PROFILES

DPLL0 and DPLL1 each have six dedicated sections in the register map allocated for translation profiles (Translation Profile 0.0 through Translation Profile 0.5 for DPLL0 and Translation Profile 1.0 through Translation Profile 1.5 for DPLL1). Each translation profile consists of the following programmable parameters:

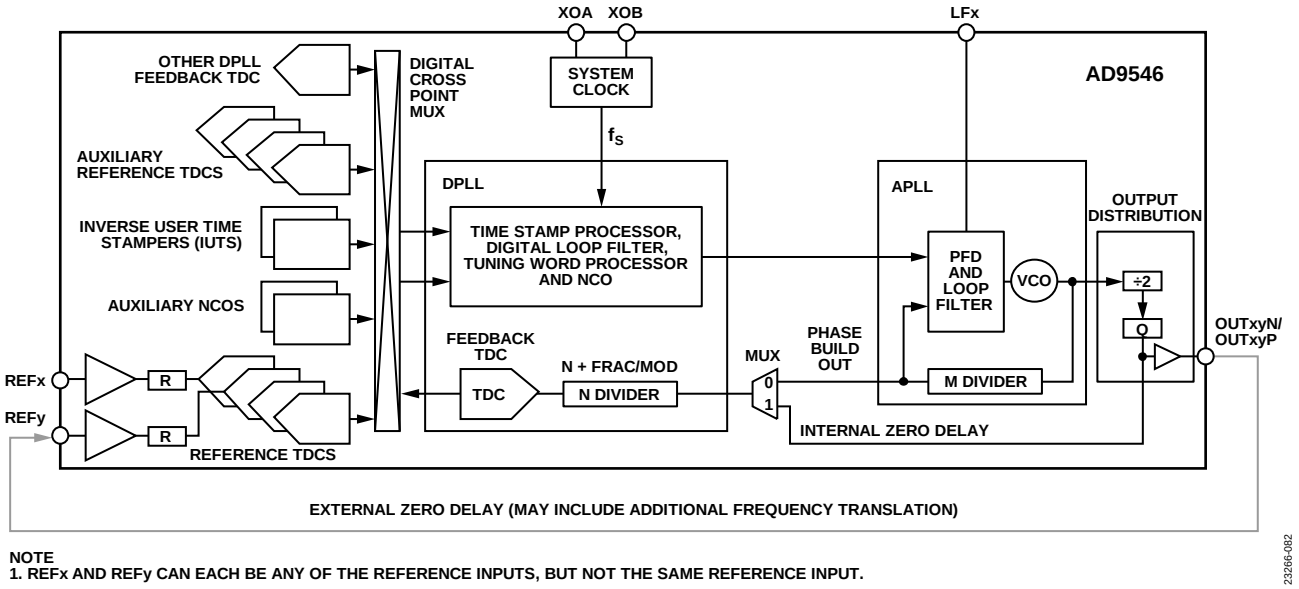
- Profile enable
- Profile priority
- Translation mode selection
- Input reference source selection
- N divider value (integer, fraction, and modulus)
- DPLL loop filter bandwidth
- Reference and feedback TDC tagging options
- DPLL fast acquisition options

Table 74 shows the translation profile address ranges.

Table 74. Translation Profile Register Address Ranges

Translation Profile	Start Address	End Address
0.0	0x1200	0x1217
0.1	0x1220	0x1237
0.2	0x1240	0x1257
0.3	0x1260	0x1277
0.4	0x1280	0x1297
0.5	0x12A0	0x12B7
1.0	0x1600	0x1617
1.1	0x1620	0x1637
1.2	0x1640	0x1657
1.3	0x1660	0x1677
1.4	0x1680	0x1697
1.5	0x16A0	0x16B7

When a given profile is in effect, the DPLL uses the parameters as specified in that profile. That is, the DPLL has no memory of the given profile when switching to another profile.



NOTE
1. REFx AND REFy CAN EACH BE ANY OF THE REFERENCE INPUTS, BUT NOT THE SAME REFERENCE INPUT.

Figure 82. Frequency Translation Loop Block Diagram

The translation profiles control only the frequency translation of the DPLL. The overall frequency translation of the channel depends on the following:

- N divider value (depending on the translation mode)
- M divider value (via the APLL0 and APLL1 controls)
- R divider value (via the REFx and auxiliary REFx controls)
- Q divider value (via the distribution controls)

Each For detailed information on the use and programming of the N, M, R, and Q divider values, see the DPLL Feedback Divider (N Divider), APLL Feedback Divider (M Divider), Reference Dividers (R Dividers), and Distribution Dividers (Q Dividers) sections.

PROFILE ENABLE

For a DPLL channel to have access to a translation profile, the user must enable the profile. To enable a translation profile, program Bit 0 to Logic 1 (default is Logic 0) in the register at the start address of the translation profile (per Table 74). Only enabled translation profiles are selectable by the DPLL. When the DPLL selects a translation profile, the selected translation profile becomes an active translation profile (see the Reference Switching section for what constitutes an active translation profile). Certain functional blocks within the AD9546 make use of the active and inactive state of a translation profile.

PROFILE PRIORITY

Profile priority works in conjunction with the reference switching capability of the AD9546 (see the Reference Switching section). The user programs a priority value via Bits[5:1] in the register at the start address of the translation profile per Table 74. When a DPLL must switch from one input reference to another, the DPLL can do so according to the 5-bit priority value programmed by the user. A priority of 0 constitutes the highest priority, and a priority of 31 constitutes the lowest priority.

INPUT REFERENCE SOURCE SELECTION

The AD9546 provides the user with several options for the reference source that supplies the input clock signal to the DPLL. These options include any one of the four REFx inputs, any of the four auxiliary REFx inputs, either of the two auxiliary NCOs, either of the two inverse user time stampers, and a special configuration that uses the feedback signal from the other DPLL channel. Each translation profile can specify a different reference source.

Source selection is via Bits[4:0] in the register at the start address of the translation profile (see Table 74) plus an offset of 1 (decimal). The decimal value associated with Bits[4:0] assigns the source, as shown in Table 75.

Table 75. Reference Source Selection

Bits[4:0]	Source
0	REFA
1	REFAA
2	REFB
3	REFBB
4	Feedback from DPLL0 (for PLL1 only)
5	Feedback from DPLL1 (for PLL0 only)
6	Auxiliary REF0
7	Auxiliary REF1
8	Auxiliary NCO 0
9	Auxiliary NCO 1
10	Not applicable
11	Auxiliary REF2
12	Auxiliary REF3
13	Inverse User Time Stamper 0
14	Inverse User Time Stamper 1
15 to 31	Not applicable

ACTIVE REFERENCE INDICATION

When a DPLL selects a translation profile, the translation profile becomes active (see the Profile Enable section), which causes the user assigned reference input (see the Input Reference Source Selection section) associated with that profile to become the time stamp source to the DPLL.

Note that when a reference becomes the time stamp source to a DPLL, the reference becomes an active reference. The user has access to the active or inactive state of a reference via an appropriately configured Mx status pin.

In summary, indication of active reference status is the result of a DPLL activating a translation profile (which contains a user assigned reference source). As such, do not confuse the active reference status with the valid reference status, which is a status condition associated with the reference source itself (not a translation profile).

TRANSLATION MODES

Overview

The translation modes govern the way the DPLL responds when it acquires a reference signal. In general, at the start of a reference acquisition, the feedback of the DPLL and reference signals do not have the same phase relationship. As such, when the DPLL begins an acquisition, it likely exhibits a significant output disturbance as the loop attempts to compensate for the phase mismatch. To deal with the output disturbance that typically results from a reference acquisition, the AD9546 offers two translation mode types: phase buildout and hitless.

A phase buildout acquisition virtually eliminates the output disturbance that results from an initial phase mismatch. During a phase buildout acquisition, the DPLL effectively measures the initial phase offset between its feedback and reference inputs and inserts the measured phase offset into the feedback path of the loop. The measured phase offset is the phase buildout offset. Insertion of the phase buildout offset into the loop effectively eliminates the initial phase difference between the feedback and reference signals, thereby minimizing the disturbance at the output of the DPLL. The insertion of the phase buildout offset implies a phase difference between the output of the DPLL and reference clock signals. This phase difference is the expected behavior for a phase buildout reference acquisition.

Note that during a phase buildout acquisition, the phase slew rate limit function does not operate on the buildout phase transition (see the Phase Slew Rate Limit section of the Digital PLL (DPLL) section for details on the phase slew rate limiter).

Unlike a phase buildout acquisition, a hitless acquisition ultimately results in a phase match of the output of the DPLL and reference clock signals. As such, a hitless translation mode is a prerequisite for zero delay operation. Given a phase mismatch between the reference of the DPLL and feedback signals at the start of an acquisition, the DPLL temporarily changes its output frequency to steer the phase difference toward zero. The AD9546 optimizes this process by assessing the initial phase relationship between the reference and feedback signals and steering toward the reference clock edge exhibiting the smallest relative phase offset.

Because a hitless acquisition undergoes frequency changes to bring about phase alignment, there can be a significant phase disturbance at the output while the loop locks to the phase of the reference (even if the initial feedback and reference frequencies are identical). The user can minimize this disturbance by placing a limit on the DPLL output frequency excursion (caused by a phase rate of change ($\delta\theta/\delta t$) at its input) via the phase slew limit feature (see the Phase Slew Rate Limit section). Furthermore, the user can place a limit on the overall output frequency excursion (due to both an input frequency offset and $\delta\theta/\delta t$) by using the frequency clamp feature (see the Tuning Word Offset Clamp section).

Even during a hitless acquisition, the AD9546 builds out the initial phase offset between the reference and feedback signals (as though the device is performing a phase buildout acquisition) to help mitigate the disturbance at the output. However, unlike the case of a phase buildout acquisition, the phase slew rate limiter is operational during a hitless acquisition (see the Phase Slew Rate Limit section for details on the phase slew rate limiter) and applies slew rate limiting to the buildout phase transition as needed.

For further information on how a PLL responds to switching from one reference to another under hitless and phase buildout operating modes, see the AN-1420 Application Note.

Translation Mode Selection

PLL0 and PLL1 can operate in any one of three modes:

- Phase buildout mode
- Internal zero delay mode (hitless)
- External zero delay mode (hitless)

The zero delay modes constitute hitless rather than phase buildout operation. For details on each mode, see the Phase Buildout Mode section, Internal Zero Delay (Hitless) Mode section, and External Zero Delay (Hitless) Mode section.

The user selects the operating mode via the translation profile registers. That is, each profile gives the user the option of selecting any one of the three operating modes for that specific profile. To select the operating mode, use Bits[1:0] in the register at the start address of the translation profile in Table 74 plus an offset of 3 (decimal). Bits[1:0] select the operating mode per Table 76.

Table 76. Frequency Translation Modes

Bits[1:0] (Decimal)	Translation Mode
0	Phase buildout
1	Internal zero delay
2	Not applicable
3	External zero delay

PHASE BUILDOUT MODE

Figure 83 shows the phase buildout configuration. In phase buildout mode, when the AD9546 switches from one input reference to another, there is virtually no phase disturbance at the output. Because the AD9546 uses time stamps generated by the reference TDCs, the difference between the time stamps from the old and new reference equates to a phase difference. Thus, the AD9546 can assess the phase offset and automatically compensate at switchover (see the Initial Phase Skew Refinement Steps section for more detail).

The frequency translation factor for phase buildout mode is

$$\frac{f_{OUTx}}{f_{REFx}} = \left(\frac{\left(N + \frac{FRAC}{MOD} \right) \times M}{2 \times R \times Q} \right)$$

where:
N is the N divider integer part.
FRAC is the N divider fractional part.
MOD is the N divider fractional modulus.
M is the divide ratio of the APLL feedback divider.
R is the divide ratio of the input reference divider.
Q is the divide ratio of the distribution divider.

However, the NCO, VCO, TDCs, and the APLL PFD inputs each have specific frequency limits. The following formulas relate the output frequency of the NCO (f_{NCO}), f_{VCO}, f_{TDC}, and f_{PFD} to f_{REFx}, f_{OUTx}, and divider values. Note that each formula has two solutions: one with respect to the input frequency (f_{REFx}) and the other with respect to the output frequency (f_{OUTx}). The user must ensure that f_{NCO}, f_{VCO}, f_{TDC}, and f_{PFD} are within their specified frequency bounds.

$$f_{TDC} = f_{REFx}/R$$
$$= (2 \times Q/(M \times (N + FRAC/MOD))) \times f_{OUTx} \quad (5)$$

where:
f_{REFx} is the reference frequency for a specific reference input, for example, REFA, REFAA, REFB, or REFBB.
f_{OUTx} is the output frequency for a specific distribution output, for example, OUT0C or OUT1A.

$$f_{NCO} = f_{PFD} = (2 \times Q/M) \times f_{OUTx}$$
$$= ((N + FRAC/MOD)/R) \times f_{REFx} \quad (6)$$

$$f_{VCO} = 2 \times Q \times f_{OUTx}$$
$$= ((N + FRAC/MOD) \times M/R) \times f_{REFx} \quad (7)$$

When the reference source is auxiliary NCO 0, auxiliary NCO 1, or the feedback from the other DPLL, then in Equation 5 through Equation 7, use R = 1.

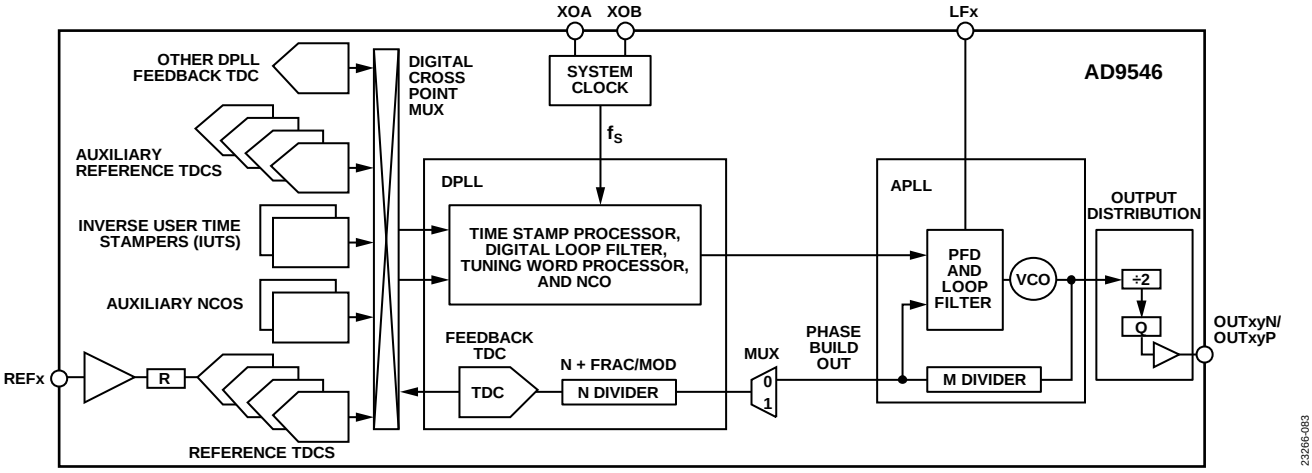


Figure 83. Phase Buildout PLL Configuration

INTERNAL ZERO DELAY (HITLESS) MODE

Figure 84 shows the internal zero delay configuration. Internal zero delay mode is a hitless mode (rather than phase buildout operating mode), in which the AD9546 gradually changes the output phase as DPLLx realigns to the phase of the new reference. The user has some control over the magnitude of the output disturbance via the phase slew limit of the AD9546 and tuning word clamp features (see the Phase Slew Rate Limit section and Tuning Word Offset Clamp section, respectively).

In internal zero delay mode, the feedback path of the PLL originates at the output of a user selected Q divider in the output distribution section. The user selects the feedback Q divider selection via the translation profile registers. To select the feedback Q divider, use Bits[4:0] (unsigned integer) in the register at the start address of the translation profile in Table 74 plus an offset of 2 (decimal).

The decimal value of Bits[4:0] selects the Q divider per the Internal Zero Delay column in Table 77. With DPLL N divider set for integer only (that is, no fractional component—a requirement for zero delay operation), the internal zero delay configuration ensures no phase offset between the reference TDC and the output of the user selected Q divider.

The frequency translation factor for internal zero delay mode is

$$\frac{f_{OUTx}}{f_{REFx}} = \frac{N}{R}$$

To provide true hitless operation, internal zero delay mode requires a constraint on the relationship between f_{REFx} and f_{OUTx}. Namely, f_{OUTx}/f_{REFx} must be an integer greater than or equal to 1.

The NCO, VCO, TDCs, and the APLL PFD inputs each have specific frequency limits. The following formulas relate f_{NCO}, f_{VCO}, f_{TDC}, and f_{PFD} to f_{REFx}, f_{OUTx}, and divider values. Note that each formula has two solutions: one with respect to the input frequency (f_{REFx}) and the other with respect to the output frequency (f_{OUTx}). The user must ensure that f_{NCO}, f_{VCO}, f_{TDC}, and f_{PFD} are within their specified frequency limits.

$$f_{TDC} = f_{REFx}/R = f_{OUTx}/N \quad (8)$$

$$f_{NCO} = f_{PFD} = (2 \times Q/M) \times f_{OUTx} = ((2 \times Q \times N)/(R \times M)) \times f_{REFx} \quad (9)$$

$$f_{VCO} = 2 \times Q \times f_{OUTx} = (2 \times Q \times N/R) \times f_{REFx} \quad (10)$$

When the reference source is auxiliary NCO 0, auxiliary NCO 1, IUTS 0, IUTS 1, or the feedback from the other DPLL, then in Equation 8 through Equation 10, use R = 1.

Table 77. Zero Delay Feedback Path Selection

Bits[4:0] (Decimal)	DPLL	Internal Zero Delay	External Zero Delay
0	0	OUT0AP	REFA
0	1	OUT1AP	REFA
1	0	OUT0AN	REFAA
1	1	OUT1AN	REFAA
2	0	OUT0BP	REFB
2	1	OUT1BP	REFB
3	0	OUT0BN	REFBB
3	1	OUT1BN	REFBB
4	0	OUT0CP	Auxiliary REF0
4	1	Not applicable	Auxiliary REF0
5	0	OUT0CN	Auxiliary REF1
5	1	Not applicable	Auxiliary REF1
6	0	Not applicable	Auxiliary REF2
6	1	Not applicable	Auxiliary REF2
7	0	Not applicable	Auxiliary REF3
7	1	Not applicable	Auxiliary REF3
8 to 31	Not applicable	Not applicable	Not applicable

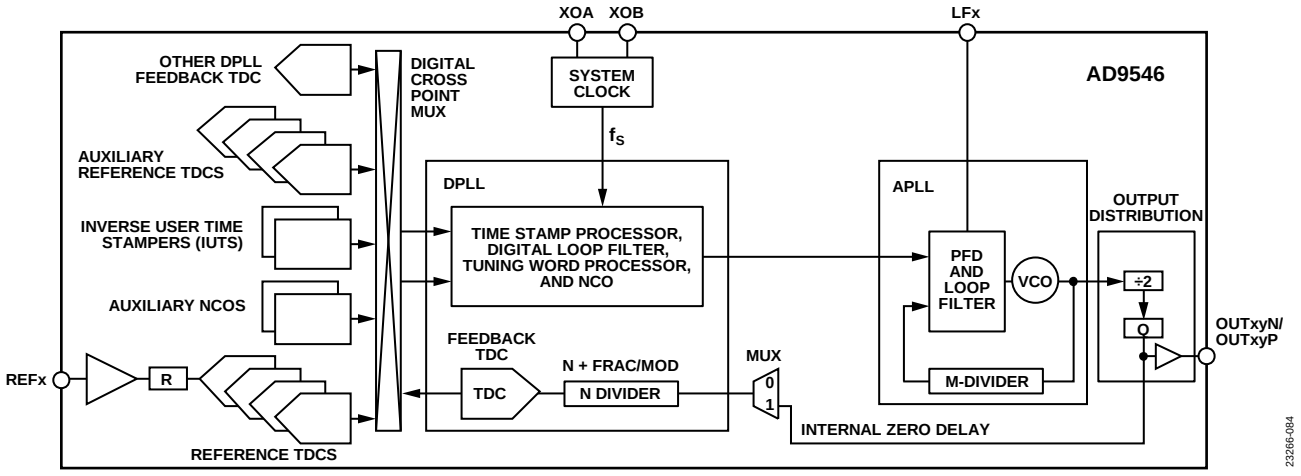


Figure 84. Internal Zero Delay PLL Configuration

The DPLL provides the user with two status bits to indicate when the DPLL enters or exits hitless operation. When the DPLL enters hitless operation, Bit 4 latches to Logic 1 in Register 0x3011 for DPLL0 or Register 0x3016 for DPLL1. Likewise, when the DPLL exits hitless operation, Bit 3 latches to Logic 1 in Register 0x3011 for DPLL0 or Register 0x3016 for DPLL1. Because these are latched status bits, to obtain visibility of subsequent DPLL state transitions into and out of hitless operation, the user must clear the corresponding bits in the IRQ map (Bit 4 and Bit 3, respectively, in Register 0x200C for DPLL0 and Register 0x2011 for DPLL1).

EXTERNAL ZERO DELAY (HITLESS) MODE

Figure 85 shows the external zero delay configuration appears in. External zero delay mode, like internal zero delay mode, is a hitless operating mode (see the Internal Zero Delay (Hitless) Mode section). As such, the same status bits associated with entering and exiting hitless operation still apply.

In external zero delay mode, the feedback path of the PLL is via an external connection from an appropriate OUTxy output to a REFx or an auxiliary REFx input. The user chooses the desired input path via the same 5-bit integer used to select the feedback Q divider in the internal zero delay configuration, but with the decimal value applying to the External Zero Delay column in Table 77. Typically, this feedback path is merely a direct connection, but the external path may also include an additional frequency translation component. To discriminate between the normal and feedback reference inputs in Figure 85, they appear as REFx and REFy, respectively. Although Figure 85 shows REFx and REFy, auxiliary REFx, and auxiliary REFy are equally valid.

The frequency translation factor from REFx (auxiliary REFx) to OUTxyP/OUTxyN for external zero delay mode is

$$\frac{f_{OUTx}}{f_{REFx}} = \frac{Ry}{Rx \times Z}$$

where:
Ry is the divide ratio of the R divider associated with REFy. REFy refers to any reference input (other than the REFx input).
Rx is the divide ratio of the R divider associated with REFx. REFx refers to any reference input (other than the REFy input).
Z is the external frequency translation factor such that

$$Z = \frac{f_{REFy}}{f_{OUTx}}$$

To provide true hitless operation, external zero delay mode requires a constraint on the relationship between f_{REFx} and f_{REFy}. Namely, f_{REFy}/f_{REFx} must be an integer greater than or equal to 1.

The NCO, VCO, TDCs, and the APLL PFD inputs each have specific frequency limits. The following formulas relate f_{NCO}, f_{VCO}, f_{TDC}, and f_{PFD} to Z, f_{REFx}, f_{OUTx}, and divider values. Note that each formula has two solutions: one with respect to the input frequency (f_{REFx}) and the other with respect to the output frequency (f_{OUTx}). The user must ensure that f_{NCO}, f_{VCO}, f_{TDC}, and f_{PFD} are within their specified frequency limits.

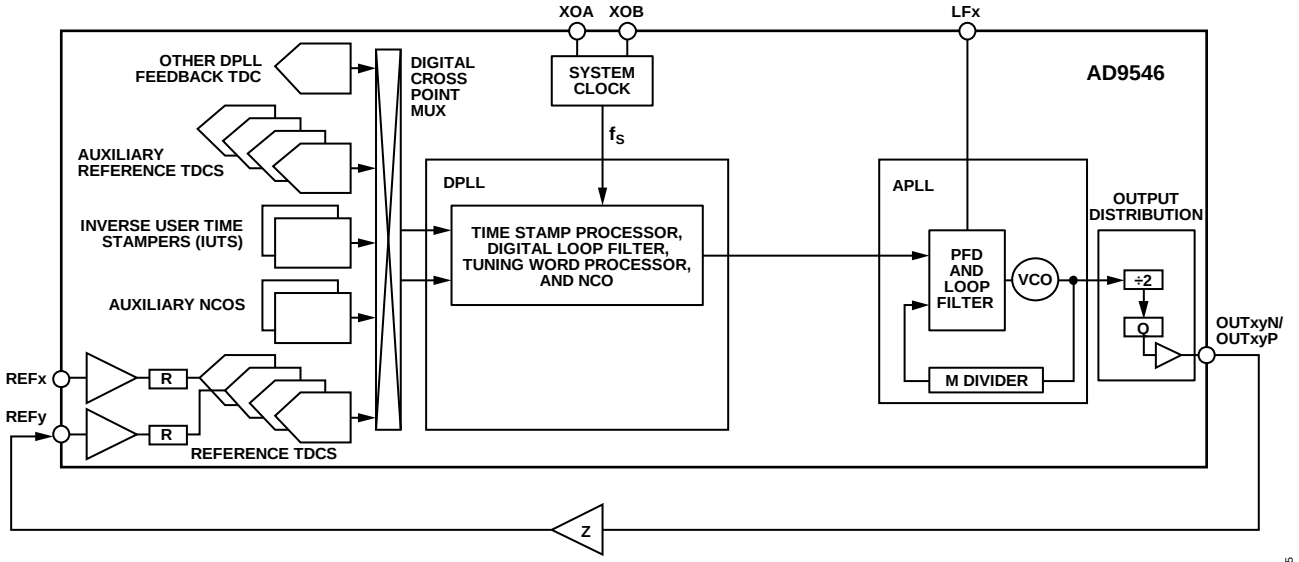
$$f_{TDC} = f_{REFx}/Rx = (Z/Ry) \times f_{OUTx} \quad (11)$$

$$f_{NCO} = f_{PFD} = ((2 \times Q \times Ry)/(Rx \times M \times Z)) \times f_{REFx} = (2 \times Q/M) \times f_{OUTx} \quad (12)$$

$$f_{VCO} = ((2 \times Q \times Ry)/(Rx \times Z)) \times f_{REFx} = 2 \times Q \times f_{OUTx} \quad (13)$$

When the reference source is auxiliary NCO 0, auxiliary NCO 1, or the feedback from the other DPLL, then in Equation 11 through Equation 13, use Rx = 1.

Although the feedback divider (N divider) does not appear in Figure 85, the N divider requires special treatment for external zero delay operation. See the DPLL Feedback Divider (N Divider) section for details.



NOTE
1. REFx AND REFy CAN EACH BE ANY OF THE REFERENCE INPUTS, BUT NOT THE SAME REFERENCE INPUT.

23266-085

Figure 85. External Zero Delay PLL Configuration

SOURCE PROFILES

SOURCE PROFILES OVERVIEW

The source profiles give the user a method to define how an input source, when selected by a DPLL channel via the active translation profile, interacts with the DPLL. The user has access to the 14 source profiles (one profile for each source) via the register map with each source profile having a start and end address per Table 78. Each source profile gives the user control of the following parameters:

- DPLL phase lock detector
- DPLL frequency lock detector
- Phase step limit
- Skew adjustment
- Initial phase skew refinement steps

Table 78. Source Name vs. Source Profile Address Range

Source Name	Start Address	End Address
REFA	0x0800	0x0811
REFAA	0x0820	0x0831
REFB	0x0840	0x0851
REFBB	0x0860	0x0871
Auxiliary NCO 0	0x0880	0x0891
Auxiliary NCO 1	0x08A0	0x08B1
DPLL0	0x08C0	0x08D1
DPLL1	0x08E0	0x08F1
IUTS 0	0x0900	0x0911
IUTS 1	0x0920	0x0931
Auxiliary REF0	0x0940	0x0951
Auxiliary REF1	0x0960	0x0971
Auxiliary REF2	0x0980	0x0991
Auxiliary REF3	0x09A0	0x09B1

A source profile links to a DPLL translation profile (see the Translation Profiles section) via Bits[4:0] of the translation profile register start address shown in Table 74 plus an offset of 1 (decimal). The selected source (per Bits[4:0] in the associated translation profile) identifies the source name in Table 78.

DPLL PHASE/FREQUENCY LOCK DETECTOR

See the DPLL Lock Detectors section for details concerning the phase and frequency detectors of the DPLL.

PHASE STEP LIMIT

Although the AD9546 can switch between multiple reference inputs, some applications use only one input and handle reference switching externally (see Figure 86). This arrangement forfeits the ability of the AD9546 to mitigate the output disturbance associated with a reference switchover, because the reference switchover is not under the control of the AD9546. However, the AD9546 offers a phase transient threshold detection feature to help identify when an external reference switchover occurs and acts accordingly.

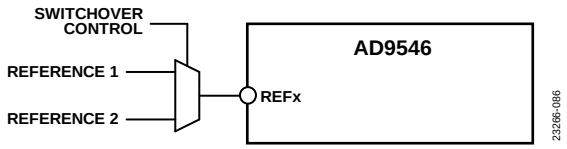


Figure 86. External Reference Switching

Phase transient threshold detection works by monitoring the output of the DPLL phase detector for phase transients but in a manner that is somewhat jitter tolerant. Otherwise, the phase transient threshold detector is prone to false positives.

To control the phase transient detector, use Bits[31:0] (unsigned) of the appropriate source profile at the start address shown in Table 78 plus an offset of 10 to 13 (decimal). Bits[31:0] constitute the phase step threshold value. A phase step threshold value of zero (default) disables the phase transient threshold detector, whereas a nonzero value serves to activate the phase transient threshold detector as well as define the desired threshold (Φ_{THRESH}) in units of picoseconds per the following equation:

$$\Phi_{THRESH} = Phase\ Step\ Threshold \times 10^{-12} \quad (14)$$

The phase transient threshold detector is not active unless the DPLL indicates frequency locked status.

Determine the value of the phase step threshold value necessary for a 12 ns limit (that is, $\Phi_{THRESH} = 12 \times 10^{-9}$). Solving Equation 14 for phase step threshold yields

$$Phase\ Step\ Threshold = \Phi_{THRESH}/10^{-12}$$
$$= (12 \times 10^{-9})/10^{-12}$$
$$= 12{,}000$$
$$= 0x\ 0000\ 2EE0\ (hexadecimal)$$

To reduce the likelihood of threshold violations induced by jitter, the user must choose Φ_{THRESH} to be at least two times the expected rms jitter (σ_{JITTER}) associated with the input reference signal.

$$\Phi_{THRESH} \geq 2 \times \sigma_{JITTER} \quad (15)$$

As such, in the previous example for a phase step threshold of 12,000, an input signal with 12 ns rms jitter is likely to produce false positives, because it violates the inequality in Equation 15. To reduce the likelihood of a false positive, the inequality in Equation 15 implies that a phase step threshold of 24,000 is a better choice. However, even with a value of 24,000, there is still a slight probability of a jitter sample exceeding $2 \times \sigma_{JITTER}$. As such, multiplying σ_{JITTER} by four to six is an even better choice.

When a phase transient occurs that exceeds the prescribed value, one or both of the following two events occur, depending on the state of Bit 7 of Register 0x2105 (for PLL0) and Register 0x2205 (for PLL1):

- The DPLL initiates a new acquisition sequence
- The reference monitor is reset

When Bit 7 is Logic 0 (default), detection of a phase step causes only the first event to occur. By initializing a new DPLL acquisition sequence, the DPLL can take advantage of the fast acquisition feature (see the DPLL Fast Acquisition Options section), assuming it is active, which is helpful for very low loop bandwidth applications. In addition, a new acquisition manages the impact of the phase step by either building out the phase or slewing to the new phase in a hitless manner.

When control Bit 7 is Logic 1, detection of a phase step causes both events to occur. Because exceeding the phase step threshold in this case implies an external switch to a new reference, resetting the reference monitor forces it to establish new reference statistics (see the Reference Monitor section).

The phase transient threshold detector provides the user with a status indicator for the occurrence of a phase step detection via Bit 0 of Register 0x3011 and Register 0x3016 (for PLL0 and PLL1, respectively). Because these status indicators are latched IRQ bits, the user must clear them via Bit 0 of Register 0x200C and Register 0x2011 (for PLL0 and PLL1, respectively) to obtain visibility of subsequent threshold violations detected by the phase step detector (see the Interrupt Request (IRQ) section).

Mitigating Phase Step Limit False Positives

When enabled, the phase transient threshold detector operates continuously while the associated reference is the active reference for the DPLL (DPLL0 or DPLL1), assuming the DPLL is frequency locked. As such, any phase disturbance at the input to the phase detector, including user induced phase adjustment per the DPLL Phase Offset Control section and the Skew Adjustment section, is subject to violating the threshold of the phase transient threshold detector. To mitigate false triggering of the phase transient threshold detector (when enabled) due to intentional phase adjustments, the user can employ the phase slew rate limiter (see the Phase Slew Rate Limit section).

The following formula relates the maximum phase slew rate (MPSR) assigned to the phase slew rate limiter necessary to prevent inadvertent triggering of the phase transient threshold detector due to a user induced phase disturbance:

$$MPSR \leq (P \times f)/7 \quad (16)$$

where:
P is the phase transient threshold detector limit (in ps).
f is the frequency (in Hz) at the input of the DPLL phase detector.

The inequality in Equation 16 ignores other contributors to phase error such as jitter, frequency offset, and propagation delay variation. Because MPSR in Equation 16 constitutes an upper limit, it is recommended to use a lower value of MPSR to provide margin (a 25% reduction is reasonable).

If the user has advance knowledge of the timing of an external event such as the switching of the reference input clock source via an external mux, rather than using the phase transient step detector, a better solution is to invalidate the associated reference manually (see the Invalidate section of the Reference Monitor section). Manual reference invalidation imposes the least impact on the steady state operation of the device. The only steady state impact is that the validation timer of the associated reference must be set to a duration that is longer (with suitable margin) than the duration between the assertion of the forced fault condition and the occurrence of the external event.

SKEW ADJUSTMENT

Skew adjustment allows the user to associate a fixed phase offset with a reference input, which is useful in applications with redundant GNSS or GPS reference sources, for example. That is, a user can have two or more GNSS/GPS sources that have identical frequency but exhibit a fixed time offset due to a mismatch between antenna cable lengths.

To activate the skew adjustment feature, use Bits[23:0] (signed) of the appropriate source profile at the start address shown in Table 78 plus an offset of 14 to 16 (decimal). Bits[23:0] constitute the phase skew value. A phase skew value of zero (default) disables the phase skew adjustment feature, whereas a nonzero value serves to activate the phase skew adjustment feature and define the desired phase skew in units of ps. The time skew associated with phase skew value is given by the following equation:

$$Time\ Skew = Phase\ Skew \times 10^{-12} \quad (17)$$

For example, determine the value of phase skew bit necessary for a time skew of −35 ns. Solving Equation 17 for phase skew yields the following:

$$Phase\ Skew = Time\ Skew/10^{-12}$$
$$= (-35 \times 10^{-9})/10^{-12}$$
$$= -35{,}000$$
$$= 0x\ FF\ 7748\ (hexadecimal)$$

INITIAL PHASE SKEW REFINEMENT STEPS

Whenever the AD9546 performs a reference switchover, it performs an initial assessment of the time offset between the old and new reference. The device does not use the phase of the new reference to determine the phase offset but compares the time stamps of the feedback TDC with the time stamps from the TDC of the new reference (see Figure 87). This comparison allows the device to insert an opposing time offset into the servo loop of the DPLL, a phase buildout operation (see the Frequency Translation Loops section).

For a phase buildout translation loop, initial phase offset insertion is an essential component of a reference switch over requiring phase buildout. However, the AD9546 also supports hitless translation profiles, for which phase buildout is not necessary. The AD9546 has the capability of initial phase offset insertion to build out the initial phase offset for a hitless switchover as well. By applying phase buildout at the start of a hitless switchover, the AD9546 can significantly reduce the time necessary to complete a hitless switchover acquisition.

Because the feedback path of the DPLL includes the loop filter, the feedback signal has the benefit of reduced jitter and an inherent resistance to change (a consequence of the typically narrow bandwidth of the loop filter). As such, when the AD9546 switches to a new reference, the phase and frequency of the old reference tends to persist in the feedback path of the DPLL. The feedback path persistence provides some time for the DPLL to compare the new reference signal to the old one.

As shown in Figure 87, the reference likely exhibits jitter. The presence of jitter implies uncertainty in the measured time offset between the feedback and reference signals. This uncertainty, in turn, leads to a potential error in the determination of the correct phase buildout value in the DPLL.

To help mitigate jitter induced errors in the assessment of the phase buildout value, the AD9546 provides a phase skew refinement feature. To activate the phase skew refinement feature, use Bits[7:0] (unsigned) of the appropriate source profile at the start address shown in Table 78 plus an offset of 17 (decimal). Bits[7:0] constitute the phase skew refinement steps value.

A phase skew refinement steps value of zero (default) disables the phase skew refinement feature. With the phase skew refinement feature disabled, the phase buildout value is an unfiltered snapshot of the reference and feedback time offset at a single sampled edge, which includes the contribution of any jitter present on the reference signal.

A nonzero phase skew refinement steps value, K, sets the number of phase samples the AD9546 analyzes as part of the phase skew refinement process. That is, instead of taking the first phase sample (jitter included) as the phase buildout value, the phase skew refinement feature processes the first K phase samples to assess the reference jitter and to determine the phase buildout value. As such, the phase skew refinement feature extends the time required to determine a phase buildout value following a reference switchover, but the extra time yields a more accurate phase buildout value.

The phase skew refinement process operates under the assumption that the feedback and reference clocks are of the same frequency, or at least very close. If they are not the same frequency, the frequency offset appears as a linear phase slew, which quickly becomes the dominant phase contributor and masks any jitter that may be present on the reference signal. Therefore, a reference switchover between references of dissimilar frequency results in degraded performance of the phase skew refinement feature.

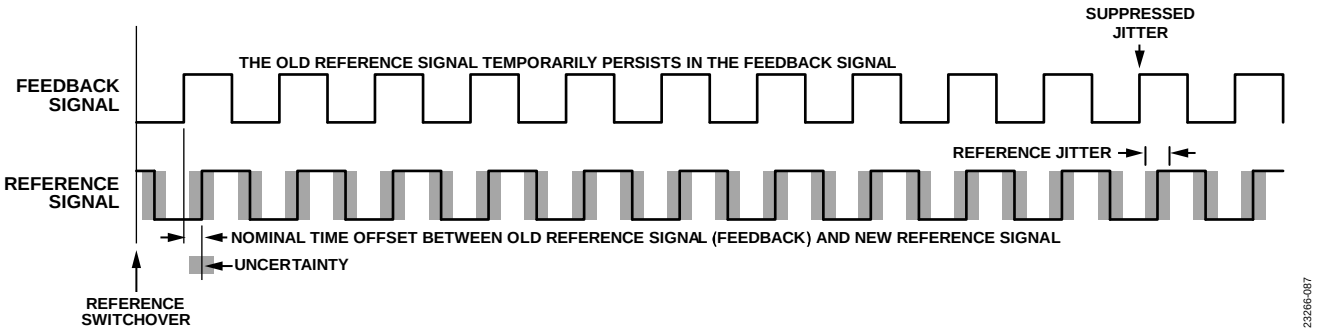


Figure 87. DPLL Reference and Feedback Signals

DIGITAL PLL (DPLL)

DPLL OVERVIEW

The DPLL is a completely digital implementation of a PLL. Figure 88 shows the fundamental building blocks of an APLL and a DPLL. An APLL typically relies on a VCO as the frequency element for generating an output signal, where the output frequency depends on an applied dc voltage. A DPLL, conversely, uses an NCO, which relies on a digital frequency tuning word (FTW) to produce the output frequency. A VCO inherently produces a timing signal because the VCO is, by definition, an oscillator, whereas the AD9546 NCO requires an external timing source, the system clock. The fundamental difference between an APLL and a DPLL is that the VCO in an APLL can tune to any frequency within its operating bandwidth, whereas the NCO in a DPLL can only tune to discrete frequencies (by virtue of the FTW).

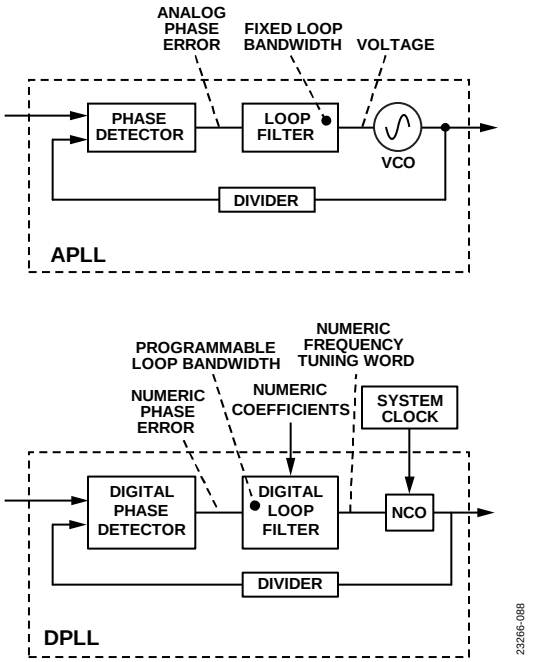


Figure 88. APLL vs. DPLL

The DPLLs in the AD9546 have a phase detector (that operates on numeric time stamps at its reference and feedback inputs rather than physical clock signals) and a digital loop filter with programmable bandwidth. The digital loop filter output yields a digital FTW (instead of a dc voltage, as in the case of an analog PLL) that produces a corresponding NCO output frequency.

DPLL LOOP CONTROLLER

The DPLL has several operating modes: freerun, holdover, and active. To ensure seamless transition between modes, the DPLL has a loop controller. The loop controller sets the appropriate DPLL operating mode based on the prevailing requirements of automatic reference switching or manual control settings.

Switchover

Switchover occurs when the loop controller switches directly from one input reference to another (see the Reference Switching section). To handle a reference switchover, the AD9546 briefly enters holdover mode, loads the DPLL parameters from the source profile associated with the new reference, and then immediately recovers.

Holdover or Freerun

Typically, the holdover or freerun state is in effect when all input references are invalid. However, the user can force the holdover mode even when one or more references are valid by programming Bit 1 to Logic 1 of Register 0x2105 (for DPLL0) or Register 0x2205 (for DPLL1). In holdover mode, the output frequency remains fixed (to the extent of the stability of the system clock). The accuracy of the AD9546 in holdover mode depends on the device programming and availability of the tuning word history.

To force freerun mode, see the Freerun Tuning Word section.

When the DPLL is in holdover or freerun mode, there is no active translation profile available (see the Frequency Translation Loops section). However, the DPLL is still attached to a translation profile by virtue of Bits[2:0] of Register 0x102A (for DPLL0) or Register 0x142A (for DPLL1). Bits[2:0] define the translation profile with a value of x, where x is 0 to 5, that associates with the DPLL when in holdover or freerun mode.

Because the DPLL attaches to the profile specified by Bits[2:0], the recommendation is for the user to program Bits[2:0] with one of the translation profiles the user has configured. For example, if the user configures Translation Profile 0, Translation Profile 2, and Translation Profile 5 for use by DPLL0, program Bits[2:0] with a decimal value of 0, 2, or 5.

Recovery from Holdover

When in holdover mode, and an enabled translation profile becomes available, the device exits holdover operation. The loop controller restores the DPLL to closed-loop operation, which begins to lock onto the selected reference, and sequences the recovery of all loop parameters based on the profile settings for the active reference.

If Bit 1 = 1 in Register 0x2105 (for DPLL0) or Register 0x2205 (for DPLL1), the device does not automatically exit holdover when a valid translation profile is available. However, automatic recovery of a valid translation profile can occur after clearing Bit 1.

DPLL FEEDBACK DIVIDER (N DIVIDER)

The DPLL feedback N divider consists of user-programmable integer part and fractional part. The fractional part, however, is only available in buildout mode (see the Frequency Translation Loops section).

Rather than offering direct programming control of the N divider, the AD9546 makes use of translation profiles (see the Translation Profiles section). Within each translation profile, the following four control elements are associated with the N divider:

- Buildout N divider integer part (N buildout)
- Hitless N divider integer part (N hitless)
- Buildout fractional multiplier (FRAC)
- Buildout fractional modulus (MOD)

N buildout and N hitless are 32-bit, unsigned integers (only Bits[29:0] are meaningful because Bits[31:30] must be zero). FRAC and MOD are 24-bit, unsigned integers.

There are 12 occurrences of the four control elements coinciding with the 12 translation profiles. Table 74 in the Frequency Translation Loops section shows the address ranges associated with the 12 translation profiles. Note that there are constraints on programming the N divider, as described in the Frequency Translation Loops section.

N Divider for Phase Buildout Mode

In phase buildout mode, the N divider divide ratio consists of an integer and fractional part, as follows:

$$N\ divider\ divide\ ratio = (N\ buildout + 1) + FRAC/MOD$$

To program the value of N buildout, use Bits[31:0] (unsigned) of the appropriate translation profile at the start address shown in Table 74 plus an offset of 12 to 15 (decimal). The programmed value of N buildout constitutes one less than the desired integer part of the divide ratio. That is, an N buildout value of 0 constitutes an integer divide ratio of 1. Therefore, N buildout provides integer divide ratios from 1 to 2^{30} (1,073,741,824).

For example, for an N divider divide ratio 1,000,000, N buildout = 999,999 (0x 000F 423F hexadecimal).

Note that changing the value of N buildout or MOD in a currently active translation profile causes the associated DPLL to reacquire its reference input signal. However, the DPLL can accommodate on-the-fly changes to the FRAC value without causing a reacquisition.

When the desired N divider divide ratio requires a fractional component, the FRAC and MOD control elements are necessary. To program the value of FRAC, use Bits[23:0] (unsigned) of the appropriate translation profile at the start address shown in Table 74 plus an offset of 16 to 18 (decimal). To program the value of MOD, use Bits[23:0] (unsigned) of the appropriate translation profile at the start address shown in Table 74 plus an offset of 19 to 21 (decimal).

For the special case of FRAC = 0, the user must also program MOD = 0 to ensure proper device functionality. Furthermore, the value of FRAC must be less than the value of MOD.

For phase buildout applications that require the N divider to be integer only, program MOD = 0.

For example, suppose the desired fractional part of the N divider divide ratio is 0.387429. Then, the corresponding fraction is FRAC/MOD = 387,429/1,000,000. Generally, the user reduces the fraction, FRAC/MOD, to its lowest terms, but in this example, the fraction is irreducible. Therefore, FRAC = 387,429 (0x 0005 E965 hexadecimal) and MOD = 1,000,000 (0x 000F 4240 hexadecimal).

N Divider in Internal Zero Delay Mode

When operating in zero delay (hitless) mode, the AD9546 behaves like an Integer N PLL. Thus, the N divider divide ratio must be an integer (that is, no fractional part). As such, the AD9546 ignores the FRAC and MOD control elements when operating in zero delay mode.

Note that zero delay operation makes use of a different integer part N divider control element than buildout operation. Thus, zero delay operation requires programming of N hitless rather than N buildout. To program the value of N hitless, use Bits[31:0] (unsigned) of the appropriate translation profile at the start address shown in Table 74 plus an offset of 8 to 11 (decimal). The programmed value of N hitless constitutes one less than the desired divide ratio. That is, an N hitless value of 0 constitutes an integer divide ratio of 1. Therefore, N hitless provides integer divide ratios from 1 to 2^{30} (1,073,741,824).

Note that operating in zero delay (hitless) mode still requires the user to program N buildout to ensure proper operation of the DPLL loop filter. However, the value of N buildout is not the same as N hitless. For internal zero delay operation, N buildout relates to N hitless per the following (rounded to the nearest integer):

$$N\ Buildout = N\ Hitless \times 2 \times Q/M$$

where:
Q is the divide ratio of the Q divider.
M is the divide ratio of the M divider (see Figure 82).

N Divider in External Zero Delay Mode

There is no internal feedback divider associated with the DPLL in external zero delay operation because the feedback path is external. Therefore, there is no requirement to program the N hitless control element. However, the user must still program N buildout to ensure proper operation of the DPLL loop filter. Program N buildout as follows (refer to Figure 85):

$$N\ Buildout = \text{round}((2 \times Q \times Ry \times f_{OUTx})/(M \times f_{REFy})) + 1$$

where:
round() is a function to round the value contained within the () to the nearest integer.
Q is the divide ratio of the Q divider.
Ry is the divide ratio of the R divider associated with the REFy input in Figure 85.
f_{OUTx} is the output frequency at the OUTxyP and/or OUTxyN output.
M is the divide ratio of the M divider.
f_{REFy} is the frequency at the REFy input.

The calculated value of N buildout cannot be less than 1 or greater than 1,073,741,824.

DPLL LOOP FILTER

The loop filter for DPLL0 and DPLL1 consists of a digital infinite impulse response (IIR) filter architecture. The digital filter is analogous to the third-order analog loop filter shown in Figure 89, where input is on the left (from the phase detector) and output is on the right (to the VCO).

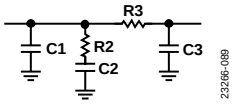


Figure 89. Third-Order Analog Loop Filter

Whereas the response of an analog loop filter is a function of physical components subject to thermal drift and aging, the response of a digital loop filter is a function of numeric coefficients and is not subject to thermal drift and aging. Figure 90 is a block diagram showing the digital loop filters and their control elements.

For completeness, Figure 90 shows the fast acquisition control block of the AD9546 as part of the loop filter control because the fast acquisition feature dynamically adjusts the loop bandwidth as part of the fast acquisition process (see the DPLL Fast Acquisition Options section).

Generally, the response characteristics of a digital filter depend solely on the numeric coefficients. However, the DPLL loop filters are special in that they require additional input parameters: a bandwidth scale factor and the numeric value of the DPLL feedback divider (see the DPLL Feedback Divider (N Divider) section for details). The DPLL automatically resets the internal state of the loop filter when the loop is not active (for example, during entry into holdover or freerun mode).

DPLL Loop Filter Base Coefficients

The response characteristic of a digital loop filter is a function of digital coefficients. In the case of the AD9546, there are base coefficients that establish a normalized frequency response characteristic. The user scales the normalized response to achieve an absolute response (see the DPLL Loop Filter Bandwidth section).

The AD9546 allows two independent sets of base coefficients, Loop Filter 0 (LF0) and Loop Filter 1 (LF1). Each set consists of four coefficients: alpha, beta, gamma, and delta, where Alpha 0 associates with LF0, Alpha 1 with LF1, and so on. Each of the four coefficients has two components: a significand and an exponent. The user has access to the coefficients via the addresses and bit ranges shown in Table 79.

Each DPLL has access to the LF0 and LF1 base coefficient sets, per Figure 90. Thus, DPLL0 and DPLL1 can use either of the base coefficient sets. The translation profiles (see the Translation Profiles section) establish which base coefficient set a DPLL uses. To select LF0 or LF1, use Bit 7 of the appropriate translation profile at the start addresses shown in Table 74 plus an offset of 3 (decimal). Logic 0 (default) selects LF0, and Logic 1 selects LF1.

Table 79. Loop Filter Base Coefficients

Coefficient	Address Range	Bit Range
Alpha 0 Significand	0x0C00 to 0x0C01	0 to 15
Alpha 0 Exponent	0x0C02	0 to 7
Beta 0 Significand	0x0C03 to 0x0C04	0 to 15
Beta 0 Exponent	0x0C05	0 to 7
Gamma 0 Significand	0x0C06 to 0x0C07	0 to 15
Gamma 0 Exponent	0x0C08	0 to 7
Delta 0 Significand	0x0C09 to 0x0C0A	0 to 15
Delta 0 Exponent	0x0C0B	0 to 7
Alpha 1 Significand	0x0C0C to 0x0C0D	0 to 15
Alpha 1 Exponent	0x0C0E	0 to 7
Beta 1 Significand	0x0C0F to 0x0C10	0 to 15
Beta 1 Exponent	0x0C11	0 to 7
Gamma 1 Significand	0x0C12 to 0x0C13	0 to 15
Gamma 1 Exponent	0x0C14	0 to 7
Delta 1 Significand	0x0C15 to 0x0C16	0 to 15
Delta 1 Exponent	0x0C17	0 to 7

The AD9546 default settings provide values for LF0 and LF1, resulting in a default response characteristic. The LF0 default values yield an open-loop response characteristic with 70° phase margin (nominal). The LF1 default values yield an open-loop response characteristic with an 88.5° phase margin (nominal). Furthermore, the LF1 default coefficients yield a nearly flat closed-loop response with <0.1 dB peaking.

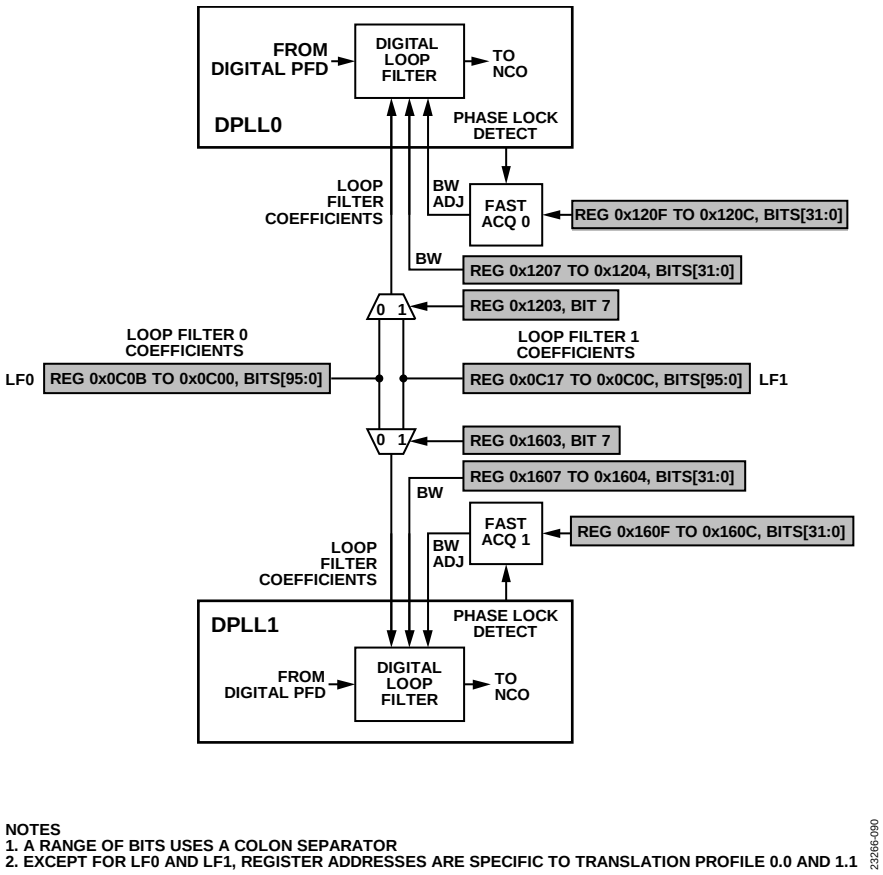


Figure 90. Digital Loop Filter Block Diagram

DPLL Loop Filter Bandwidth

Because the base coefficients establish a normalized frequency response, the loop filter requires a scale factor to set the absolute frequency response. The scale factor, which sets the bandwidth (BW), resides in the translation profiles (see the Translation Profiles section), as shown in Figure 90. The scale factor is Bits[31:0] (unsigned) residing in the appropriate translation profile at the start address shown in Table 74 plus an offset of 4 to 7 (decimal).

The frequency units associated with the scale factor depend on the normalization of the LF0 and LF1 base coefficients (see the DPLL Loop Filter Base Coefficients section). In the case of the default base coefficients, the normalized frequency units are μHz (10^{-6} Hz) for both LF0 and LF1.

To calculate the scale factor necessary to set the 0 dB open-loop bandwidth, divide the desired loop bandwidth by the normalized frequency associated with the selected set of base coefficients, LF0 or LF1. In the case of the default LF0 and LF1 values, the normalized frequency is 10^{-6} Hz.

For example, given a desired loop bandwidth of 50 Hz, the required scale factor is

$$\textit{Scale Factor} = 50\ \text{Hz}/10^{-6}\ \text{Hz}$$
$$= 50{,}000{,}000$$
$$= 0\text{x}\ 02\text{FA}\ F080\ \text{(hexadecimal)}$$

Note that the 32-bit scale factor implies a maximum DPLL loop bandwidth of nearly 4.3 kHz ($2^{32} \times 10^{-6}$ Hz). However, the presence of the NCO gain tuning word filter in the loop (see the NCO Gain Tuning Word Filter Bandwidth section) puts a constraint on the maximum DPLL loop filter bandwidth. Table 80 relates the maximum DPLL loop filter bandwidth as a function of the value of the selected NCO gain filter bandwidth. The active translation profile of the DPLL determines whether the LF0 or LF1 column applies for a given DPLL channel.

Table 80. Maximum DPLL Loop Filter Bandwidth

NCO Gain Filter Bandwidth Selection	LF0 Maximum Loop Bandwidth (Hz)	LF1 Maximum Loop Bandwidth (Hz)
0	1850	305
1	925	152.5
2	462.5	76.3
3	231.3	38.1
4	115.6	19.1
5	57.81	9.53
6	28.91	4.77
7	14.45	2.38
8	7.227	1.191
9	3.613	0.596
10	1.807	0.298
11	0.9033	0.149
12	0.4517	0.745
13	0.2258	0.037
14	0.1219	0.019
15	0.0565	0.009

DPLL NCO

The DPLL NCO requires an external clock source. In the case of the AD9546, the external clock source drives the XOA and XOB input pins, from which an integrated PLL synthesizer generates a clock signal that is approximately 2.4 GHz.

The DPLL normally operates in closed-loop fashion (that is, as a PLL), which is the normal operating mode, wherein the loop filter is the FTW source for the NCO. However, under certain conditions, the DPLL operates in an open-loop configuration. Figure 91 differentiates between the two configurations by means of a switch. A loop controller determines whether the DPLL is in closed-loop operation (switch closed) or open-loop operation (switch open), while an FTW processor determines the FTW source for the NCO.

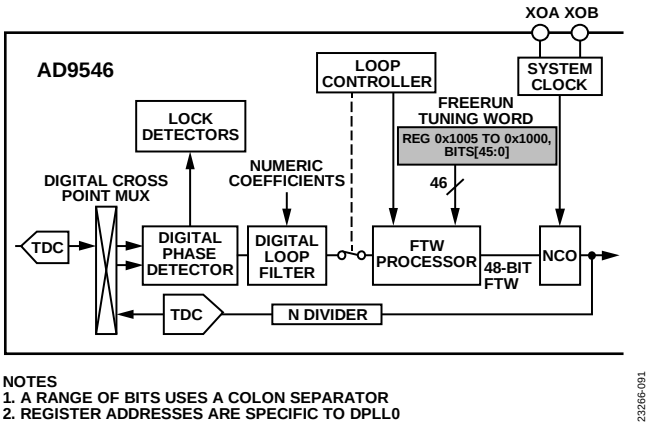


Figure 91. DPLL Block Diagram

Figure 91 also shows the lock detectors (see the DPLL Lock Detectors section). For details on the feedback divider, see the DPLL Feedback Divider (N Divider) section. For clarity, Figure 91 also shows the digital cross point mux and TDCs that feed the digital phase detector. The TDCs convert the rising edges of the input and feedback signals to numeric time stamps (see the Time to Digital Converter (TDC) section for details).

Although Figure 91 shows the N divider connected directly to the NCO output, this diagram is a simplification of the actual feedback path (see Figure 82 in the Frequency Translation Loops section). However, regarding the operation and control of the DPLL, this simplification is valid in the context of the following paragraphs.

Frequency tuning of the DPLL is by virtue of an NCO, which employs a sigma-delta modulator (SDM) architecture. The SDM has an internal integer divider that divides down the system clock frequency with the output of the divider constituting the output of the NCO. The SDM effectively modulates the modulus of this divider to produce an output frequency that is a fractionally scaled down version of the system clock frequency based on an input 48-bit FTW. Because the NCO is SDM-based, it employs noise shaping that redistributes its modulation noise away from the NCO output frequency (the APLL, which follows the DPLL, suppresses the out of band modulation noise of the SDM).

The output frequency of the NCO (f_{NCO}) depends primarily on the numeric value of the 48-bit FTW and the frequency of the system clock (f_S) per the following equation:

$$f_{NCO} = f_S \times FTW/2^{48}$$

For a given f_S and a desired f_{NCO}, compute FTW as

$$FTW = \text{round}(2^{48} \times f_{NCO}/f_S) \quad (18)$$

where round(x) is a function to round x to the nearest integer.

The NCO automatically converts the 48-bit FTW into two components: an integer part (INT) and a fractional part (FRAC). INT and FRAC relate to FTW as

$$INT = \text{floor}(2^{48}/FTW) \quad (19)$$

$$FRAC = 2^{-40} \times \text{round}(2^{40} \times ((2^{48}/FTW) - INT)) \quad (20)$$

where:
$7 \le INT \le 13$.
$0.05 \le FRAC \le 0.95$.
floor(x) is a function that leaves x unchanged if x is an integer. Otherwise, x becomes the nearest integer in the negative direction.

The constraints on INT and FRAC necessarily impose limitations on the choice of FTW. For example, let f_S = 2.30 GHz and f_{NCO} = 245.76 MHz, which yields (per Equation 18).

$$FTW = 30{,}076{,}213{,}163{,}657$$

Then, per Equation 19 and Equation 20,

$$INT = 9$$

$$FRAC = 0.358723958333030168432742357254 03$$

In this case, INT and FRAC satisfy their defined constraints.

Although the preceding example validates FTW for the given f_S and f_{NCO}, the example does not necessarily validate FTW for a

given application. That is, the preceding example assumes f_S and f_{NCO} are completely static values. However, f_S is only as stable as the oscillator or resonator at the XOA and XOB pins. Furthermore, with the DPLL locked to an input reference signal, f_{NCO} tracks variations in the reference frequency. Therefore, the user must assess variations on FTW for a given application. That is, the user must consider upper and lower FTW values, which lead to upper and lower INT and FRAC values as well.

For example, assume in the preceding example that input frequency variations cause the FTW to vary by 0.5%, leading to two FTW values that differ from 30,076,213,163,657 by 0.5%:

Lower FTW = 29,925,832,097,839

Upper FTW = 30,226,594,229,475

The upper and lower FTW values lead to the following INT and FRAC values:

INT_{UPPER} = 9

$FRAC_{UPPER}$ = 0.312163142620192957110702991 4856

INT_{LOWER} = 9

$FRAC_{LOWER}$ = 0.405752721942917560227215290069 58

In this case, the upper and lower INT values and upper and lower FRAC values satisfy the constraints on INT and FRAC.

The upper and lower INT values must be the same. Otherwise, having different values implies that the upper and lower FTW values cross an SDM integer boundary, which can lead to poor spurious performance. There are two ways to remedy this problem. The first, which is less workable, is to limit the variation on the system clock frequency and the reference input frequency. The second is to choose a new FTW value (and, by implication, a new f_{NCO} value). In either case, the goal is to constrain the variation of the FTW such that the FTW yields identical (and valid) upper and lower INT values, as well as valid upper and lower FRAC values.

The NCO applies adjustments to INT and FRAC as necessary when the system clock compensation feature is active (see the System Clock Compensation section).

NCO GAIN TUNING WORD FILTER BANDWIDTH

Although not explicitly shown in Figure 91, the NCO contains a digital low-pass filter, the NCO gain tuning word filter. This filter has a single-pole response similar to a simple resistor and capacitor low-pass filter (see Figure 92), but with a variable gain component that compensates for the nonlinear gain of the NCO. The NCO gain tuning word filter reduces frequency transients that can occur when the DPLL switches between closed-loop and open-loop operating modes (active to holdover, for example).

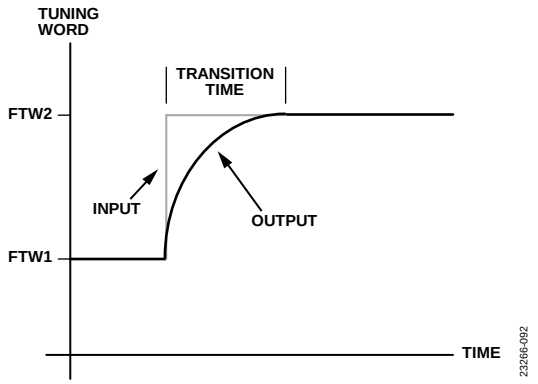


Figure 92. NCO Gain Tuning Word Filter Response

To control the filter bandwidth, use Bits[3:0] (unsigned integer) in Register 0x1009 for DPLL0 and Register 0x1409 for DPLL1. Table 81 shows how the value of Bits[3:0] relates to the bandwidth and resulting transition time.

To prevent degradation of the phase margin associated with the DPLL loop filter (see the DPLL Loop Filter section), the user must be careful to choose an NCO gain tuning word filter bandwidth from Table 81 that is at least 100 times greater than the loop bandwidth of the DPLL. This value includes any expansion of the DPLL loop filter bandwidth by the fast acquisition block, if enabled (see the DPLL Fast Acquisition Options section).

Table 81. NCO Gain Tuning Word Filter Bandwidth Selections

Bits[3:0]	3 dB Bandwidth (Hz)	Transition Time (ms)
0	248,000	0.003
1	124,000	0.006
2	62,000	0.013
3	31,000	0.026
4	15,500	0.051
5	7800	0.102
6	3900	0.204
7	1900	0.419
8	970	0.820
9	490	1.62
10	240	3.32
11	120	6.63
12	61	13.0
13	30	26.5
14	15	53.1
15	7.6	105

The NCO gain tuning word filter has implications when using the NCO as a traditional, open-loop digital frequency synthesizer. For example, when the DPLL is programmed for freerun mode (see the Freerun Tuning Word section), the DPLL operates like a traditional NCO. That is, the user can program different freerun tuning words to synthesize different frequencies. In a traditional NCO, programming a new tuning word results in an instantaneous switch from the initial frequency to the new frequency (like the input trace shown in Figure 92). However,

in the case of the DPLL, when programming different tuning words, the NCO transitions from one frequency to the next smoothly based on the programmed bandwidth of the NCO gain tuning word filter, as shown in Figure 92 (see Table 81 for the transition time to 99% of a step input).

DPLL LOCK DETECTORS

DPLL Phase Lock Detector

Each DPLL channel (DPLL0 and DPLL1) contains a completely digital phase lock detector. The user controls the threshold sensitivity and hysteresis of the phase lock detector via the source profiles (see the Source Profiles section).

The phase lock detector indicates the phase lock status via Bit 1 of Register 0x3100 for DPLL0 and Register 0x3200 for DPLL1 (Logic 0 is unlocked and Logic 1 is locked). However, because Bit 1 is dynamic in nature, the recommendation is to use the interrupt request (IRQ) mechanism for phase lock indication instead. The IRQ mechanism observes the state of Bit 1 and latches the state transitions. Specifically, Bit 0 of Register 0x3010 for DPLL0 and Register 0x3015 for DPLL1 latches a status change from phase unlocked to phase locked as a Logic 1. Likewise, Bit 1 of the same registers latches a status change from phase locked to phase unlocked as a Logic 1. Because Bit 0 and Bit 1 are latched bits, however, they may represent a condition that is no longer true. Therefore, the user must clear the phase locked and phase unlocked status via Bit 0 and Bit 1, respectively, of Register 0x200B for DPLL0 and Register 0x2010 for DPLL1. Otherwise, the user may lose indication of subsequent state transitions by the phase lock detector (see the Interrupt Request (IRQ) section).

The phase lock detector behaves in a manner analogous to water in a tub (see Figure 93). The total capacity of the tub is 4096 units, with −2048 denoting empty, 0 denoting the 50% point, and +2047 denoting full. The tub also has a safeguard to prevent overflow. Furthermore, the tub has a low water mark at −1025 and a high water mark at +1024. To change the water level, the phase lock detector adds water with a fill bucket or removes water with a drain bucket.

The user specifies the size of the fill and drain buckets via the source profiles. To specify the phase lock fill rate, use Bits[7:0] (unsigned integer) of the appropriate source profile at the start address shown in Table 78 plus an offset of 3 (decimal). To specify the phase lock drain rate use Bits[7:0] of the appropriate source profile at the start address shown in Table 78 plus an offset of 4 (decimal).

The water level in the tub is what the lock detector uses to determine the lock and unlock conditions. When the water level is below the low water mark (−1025), the lock detector indicates an unlock condition. Conversely, when the water level is above the high water mark (1024), the lock detector indicates a lock condition. When the water level is between the marks, the lock detector holds its previous condition. Figure 93 shows this concept with an overlay of an example of the instantaneous water level (vertical) vs. time (horizontal) and the resulting lock/unlock states.

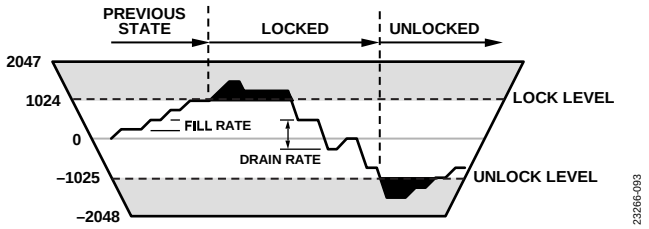


Figure 93. Lock Detector Diagram

The user has access to the 12-bit (signed) instantaneous water level value of the phase lock detector via Register 0x3109 to Register 0x310A (DPLL0) and Register 0x3209 to Register 0x320A (DPLL1). As shown in Figure 93, the pertinent water level values appear along the left side of the tub.

During any given PFD phase error sample, the lock detector either adds water with the fill bucket or removes water with the drain bucket (one or the other, but not both). The decision of whether to add or remove water depends on the phase lock threshold level specified by the user via Bits[23:0] (unsigned integer) of the appropriate source profile at the start address shown in Table 78 plus an offset of 0 to 2 (decimal). The value of Bits[23:0] is the desired threshold in picoseconds. Thus, the phase lock threshold extends from 0 ps to 16.7 µs and represents the phase error at the output of the PFD. Though the programming range supports 0 ps as a lower limit, in practice, the minimum value must be greater than 50 ps.

The phase lock detector compares the absolute value of each phase error sample at the output of the PFD to the programmed phase threshold value. If the absolute value of the phase error sample is less than or equal to the programmed phase threshold value, the detector control logic adds one fill bucket into the tub. Otherwise, the detector control logic removes one drain bucket from the tub. The magnitude of the phase error sample (polarity is ignored), relative to the phase threshold value, determines whether to fill or drain the bucket.

Regarding the fill and drain process, an exception to normal operation occurs when the phase slew limiter is active. When the phase slew limiter is actively in the limiting process, the lock detector inhibits fill events, allowing only drain events to occur.

When more filling is taking place than draining, the water level in the tub eventually rises above the high water mark (1024), which causes the lock detector to indicate lock. When more draining is taking place than filling, the water level in the tub eventually falls below the low water mark (−1024), which causes the lock detector to indicate unlock. The ability to specify the threshold level, fill rate, and drain rate enables the user to tailor the operation of the lock detector to the statistics of the timing jitter associated with the input reference signal. Note that, for debug purposes, the user can make the fill or drain rate zero to force the lock detector to indicate a lock or unlock state, respectively.

Whenever the AD9546 enters freerun or holdover mode, the DPLL phase lock detector indicates an unlocked state.

For more information on how to choose the appropriate phase lock threshold, fill rate, and drain rate values for a given application, refer to the AN-1061 Application Note, *Behavior of the AD9548 Phase and Frequency Lock Detectors in the Presence of Random Jitter*.

DPLL Frequency Lock Detector

The operation of the frequency lock detector is identical to that of the phase lock detector, with the following two exceptions:

- The fill or drain decision is based on the period deviation between the reference of the DPLL and the feedback signals instead of the phase error at the output of the PFD.
- The frequency lock detector is unaffected by the state of the phase slew limiter.

Like the phase lock detector, the user has access to the 12-bit (signed) instantaneous water level value of the frequency lock detector via Register 0x310B to Register 0x310C (DPLL0) and Register 0x320B to Register 0x320C (DPLL1). As shown in Figure 93, the pertinent water level values appear along the left side of the tub.

The frequency lock detector indicates frequency lock status via Bit 2 of Register 0x3100 for DPLL0 and Register 0x3200 for DPLL1 (Logic 0 is unlocked, and Logic 1 is locked). However, because Bit 2 is dynamic in nature, the recommendation is to use the IRQ mechanism for frequency lock indication instead. The IRQ mechanism observes the state of Bit 2 and latches the state transitions. Specifically, Bit 2 of Register 0x3010 for DPLL0 and Register 0x3015 for DPLL1 latches a status change from frequency unlocked to frequency locked as a Logic 1. Likewise, Bit 3 of the same registers latches a status change from frequency locked to frequency unlocked as a Logic 1. Because Bit 2 and Bit 3 are latched bits, however, they may represent a condition that is no longer true. Therefore, the user must clear the frequency locked and frequency unlocked status via Bit 2 and Bit 3, respectively, of Register 0x200B for DPLL0 and Register 0x2010 for DPLL1. Otherwise, the user may lose indication of subsequent state transitions by the frequency lock detector (see the Interrupt Request (IRQ) section).

The difference between the period of the signal arriving at the reference input to the DPLL and the period of the signal arriving at the feedback input to the DPLL constitutes the period error between the two signals. The period error relates to f_{REF} and the feedback frequency (f_{FB}) as

$$Period\ Error = 1/f_{FB} - 1/f_{REF}$$

For any given period error sample, the frequency lock detector either adds water with the fill bucket or removes water with the drain bucket (one or the other, but not both). The decision of whether to add or remove water depends on the frequency lock threshold specified by the user via Bits[23:0] (unsigned integer) of the appropriate source profile at the start address shown in Table 78 plus an offset of 5 to 7 (decimal). The value of Bits[23:0] is the desired frequency lock threshold in ps. Thus, the frequency lock threshold extends from 0 ps to 16.7 µs. The frequency lock threshold represents the absolute value of the period error between the reference and feedback signals at the input to the DPLL as follows:

$$Frequency\ Lock\ Threshold = |Period\ Error|/10^{-12}$$

For example, consider a nominal frequency at the reference input to the DPLL of 80 kHz. Under a stable lock condition, the frequency at the reference and feedback inputs to the DPLL are equal. To configure the frequency lock detector to make fill or drain decisions when the feedback and reference frequency at the input to the DPLL differ by 100 Hz, establish the frequency lock threshold for a 100 Hz deviation by choosing f_{REF} = 80 kHz and f_{FB} = 80.1 kHz (or 79.9 kHz).

$$Frequency\ Lock\ Threshold = |Period\ Error|/10^{-12}$$
$$= |1/f_{REF} - 1/f_{FB}|/10^{-12}$$
$$= |1/80{,}000 - 1/80{,}100|/10^{-12}$$
$$= 15{,}605 \text{ (nearest integer)}$$
$$= 0\text{x } 00\ 3CF5 \text{ (hexadecimal)}$$

For more information on how to choose the appropriate frequency lock threshold, fill rate, and drain rate values for a given application, refer to AN-1061 Application Note.

FREERUN TUNING WORD

The closed switch in Figure 91 indicates that the DPLL is operating in closed-loop mode, where the loop filter delivers FTWs in real time to the NCO. In open-loop operation, the switch is in the open position and the FTW processor provides a static FTW to the NCO. The loop controller opens or closes the switch as needed. For example, when the DPLL is in freerun mode (that is, Bit 0 = 1 of Register 0x2105 or Register 0x2205), the loop controller opens the switch and the FTW processor routes the freerun tuning word to the NCO.

In this case, the freerun tuning word establishes the NCO output frequency, f_{NCO}. The user sets the value of the freerun tuning word via Bits[45:0] (unsigned integer) in Register 0x1000 to Register 0x1005 (for DPLL0) and Register 0x1400 to Register 0x1405 (for DPLL1).

$$f_{NCO} \approx f_S \times FTW0/2^{48}$$

where:
f_{NCO} is the NCO output frequency.
f_S is the system clock frequency.
FTW0 represents the value of the 46-bit freerun tuning word.

The preceding f_{NCO} formula is an approximation (≈) because the exact NCO output frequency differs slightly (see the DPLL NCO section).

For example, given a system clock frequency of 2.3 GHz and a desired NCO output frequency of 245.76 MHz, solve the f_{NCO} equation for FTW0 to yield the following value of the freerun tuning word:

$$FTW0 = 2^{48} \times (f_{NCO}/f_S) \quad (21)$$

$$= 2^{48} \times (245.76 \times 10^6/(2.3 \times 10^9))$$

$$= 30{,}076{,}213{,}163{,}657 \text{ (nearest integer)}$$

$$= 0\text{x } 1\text{B5A AA00 7689 (hexadecimal)}$$

Although the tuning resolution of the NCO is 48 bits, the freerun tuning word is only 46 bits because, under all normal operating conditions, the two most significant bits of the freerun tuning word calculation (see Equation 21) are Logic 0. Therefore, even though the NCO tuning word is 48 bits, the freerun tuning word uses only the 46 LSBs of the NCO tuning word.

The value of freerun tuning word must satisfy the constraints imposed by the NCO SDM (see the DPLL NCO section).

DPLL FAST ACQUISITION OPTIONS

Fast Acquisition Overview

Certain applications necessitate low loop bandwidths less than 1 Hz (for example, an application where the input reference to the DPLL originates from a GPS/GNSS receiver with a 1 pulse per second output). A 1 Hz reference requires a loop bandwidth much less than 1 Hz, which can lead to frequency acquisition and phase lock times in the range of minutes to hours.

To overcome the long acquisition time imposed by a sub 1 Hz loop bandwidth, the AD9546 provides a built in fast acquisition feature that greatly reduces the lock time of the DPLL. The fast acquisition feature automatically changes the DPLL loop filter bandwidth in a controlled manner. When the fast acquisition feature is active (see the Fast Acquisition Bandwidth Control section) the fast acquisition controller uses the following features to control the fast acquisition process:

- DPLL loop bandwidth
- Fast acquisition excess bandwidth
- Fast acquisition lock settle time
- Fast acquisition timeout

The controller begins the fast acquisition process by applying a user defined excess bandwidth factor to the specified DPLL loop filter bandwidth (see the DPLL Loop Filter Bandwidth section in the DPLL Loop Filter section). The controller then successively reduces the excess bandwidth until it reaches the specified DPLL loop filter bandwidth. At each step in the bandwidth reduction process, however, the controller waits for the DPLL phase detector to indicate lock status before moving on to the next step (note the fast ACQ x functional blocks in Figure 90).

Fast Acquisition Status Indicators

While the fast acquisition controller is in the process of performing a fast acquisition, status Bit 4 is Logic 1 in Register 0x3102 (for DPLL0) and Register 0x3202 (for DPLL1). Bit 4 relates to the IRQ function (see the Interrupt Request (IRQ) section) through Bit 2 (fast acquisition started) and Bit 3 (fast acquisition completed) in Register 0x3012 (for DPLL0) and Register 0x3017 (for DPLL1). Because Bit 2 and Bit 3 are latched, the user must use Bit 2 and Bit 3, respectively, of Register 0x200D (for DPLL0) and Register 0x2012 (for DPLL1) to clear the latched state. Otherwise, the user cannot observe subsequent state transitions of Bit 4.

Fast Acquisition Bandwidth Control

The fast acquisition feature is active when the user sets the fast acquisition excess bandwidth parameter to a nonzero value. The fast acquisition excess bandwidth parameter is Bits[3:0] (unsigned integer) in the appropriate translation profile at the start address shown in Table 74 plus an offset of 22 (decimal).

When the fast acquisition excess bandwidth parameter is nonzero, the fast acquisition controller uses the fast acquisition excess bandwidth parameter value to define the maximum starting loop bandwidth (BW_0) as follows:

$$BW_0 = DPLL\ Loop\ Bandwidth \times 2^{fast\ acquisition\ excess\ bandwidth\ parameter}$$

For example, given a DPLL loop bandwidth of 0.0001 Hz (100 µHz) and a fast acquisition excess bandwidth parameter value of 9, find the value of BW_0 as follows:

$$BW_0 = DPLL\ Loop\ Bandwidth \times 2^{fast\ acquisition\ excess\ bandwidth\ parameter}$$

$$= 0.0001 \text{ Hz} \times 2^9$$

$$= 0.0512 \text{ Hz}$$

In the preceding example, when the DPLL begins the signal acquisition process, the starting loop bandwidth for the fast acquisition sequence is 512 times wider than the final value. After the DPLL acquires phase lock, the fast acquisition controller tightens the loop bandwidth by a factor of two and waits for phase lock. This bandwidth tightening process repeats until the loop bandwidth reaches the specified DPLL loop bandwidth. By using wider loop bandwidths during the signal acquisition process, the fast acquisition feature achieves frequency and phase lock much quicker than normal DPLL operation without the fast acquisition feature.

The phase lock characteristic of the loop is subject to the DPLL lock detector settings (see the DPLL Lock Detectors section).

Fast Acquisition Settling Time Control

Because each step in the fast acquisition process relies on phase lock indication to proceed to the next step, the phase lock indication must not chatter. That is, a noisy reference can cause the phase lock indicator to switch intermittently between lock and unlock until the loop settles to a stable lock condition. If the lock indicator chatters, the chatter can cause the fast acquisition controller to advance prematurely.

To help minimize the effect of a chattering phase lock indication, the fast acquisition controller allows the user to specify a lock indication settling time via Bits[2:0] (unsigned integer) in the appropriate translation profile at the start address shown in Table 74 plus an offset of 23 (decimal).

The value of Bits[2:0] tells the fast acquisition controller how long to observe a nonchanging phase lock indication before moving on to the next step. Table 82 show the available settling time durations.

Table 82. Fast Acquisition Lock Detect Settling Time

Bits[2:0]	Settle Time
0	1 ms
1	10 ms
2	50 ms
3	100 ms
4	500 ms
5	1 sec
6	10 sec
7	50 sec

During operation, if the DPLL indicates a phase unlock within the settling period, the fast acquisition controller does not advance to the next step. Instead, the controller resets the settling timer, waits for the next indication of phase lock, and invokes the settling period again. This process continues until the phase lock remains for the full duration of the prescribed settling time.

Fast Acquisition Timeout Control

To prevent the fast acquisition controller from stalling because the DPLL fails to reach a phase locked condition, the controller has a timeout mechanism. That is, at each step in the fast acquisition process, the fast acquisition controller waits for the DPLL to phase lock (and settle), but only up to a user defined maximum amount of time.

The user specifies the maximum time for the fast acquisition controller to wait for phase lock via Bits[6:4] (unsigned integer) in the appropriate translation profile at the start address shown in Table 74 plus an offset of 23 (decimal).

The available timeout values appear in Table 83. If the fast acquisition controller does not observe phase lock and settling within the prescribed timeout interval, the fast acquisition controller automatically advances to the next step in the fast acquisition process. When the fast acquisition controller reaches the final bandwidth step and the DPLL indicates phase lock within the specified settle time, the fast acquisition controller sets status Bit 5 (fast acquisition complete) of Register 0x3102 (for DPLL0) or Register 0x3202 (for DPLL1).

Table 83. Fast Acquisition Timeout

Bits[6:4]	Settle Time
0	10 ms
1	50 ms
2	100 ms
3	500 ms
4	1 sec
5	10 sec
6	50 sec
7	100 sec

Fast Acquisition Trigger

The Fast Acquisition Timeout Control section describes the timing of the fast acquisition process but not what triggers the process. The user controls what triggers the fast acquisition process via Bits[3:0] of Register 0x2106 (for DPLL0) and Register 0x2206 (for DPLL1).

- Bit 0: fast acquisition from freerun
- Bit 1: fast acquisition from holdover
- Bit 2: fast acquisition during first acquisition
- Bit 3: fast acquisition when no distribution outputs are toggling

The fast acquisition controller treats the specific restrictions implied by Bits[3:0] in logical OR fashion. That is, the controller obeys all restrictions that apply per the set bits. Note that when all four of these bits are Logic 0, no restrictions apply. That is, the fast acquisition controller is completely unrestricted, and the DPLL always uses fast acquisition (if enabled). However, making any one of these bits Logic 1 puts the fast acquisition controller into a restricted operating mode where it must obey the restrictions explicitly as follows.

Fast acquisition triggers under the following four sets of conditions:

- When Bit 0 = 1 and the DPLL enters closed-loop operation from freerun mode, fast acquisition triggers.
- When Bit 1 = 1 and the DPLL enters closed-loop operation from holdover mode, fast acquisition triggers.
- When Bit 2 = 1 and this is the first execution of the fast acquisition process, fast acquisition triggers. That is, the fast acquisition controller has not completed a fast acquisition sequence previously (more specifically, when the fast acquisition complete status bit is Logic 0). When the fast acquisition complete status bit is Logic 1, the user can clear it by writing Logic 1 to Bit 3 (autoclearing) of Register 0x2107 (for DPLL0) or Register 0x2207 (for DPLL1). Setting Bit 3 = 1 provides a mechanism to have subsequent first acquisition events.
- When Bit 3 = 1 and all clock distribution outputs of the corresponding PLL channel are not toggling, fast acquisition triggers.

As an example of fast acquisition trigger control, assume Bit 1 and Bit 2 are both Logic 1 and that DPLL enters closed-loop operation from freerun mode, which does not satisfy the first condition. However, because there is a second restriction, and the device treats the restrictions in logical OR fashion, the status of the second constraint applies. As such, if the entry of the DPLL into closed-loop operation from freerun mode happens to be the first acquisition of the DPLL, a fast acquisition is in effect (otherwise, a normal acquisition results). Next, assume the DPLL enters closed-loop operation from holdover, which satisfies the first condition. As such, the DPLL employs fast acquisition (the first acquisition restriction is immaterial in this case).

DPLL PHASE OFFSET CONTROL

In general, the feedback loop of the DPLL (like the APLL) tends to force the average phase offset between the two inputs of the digital phase detector to zero (see Figure 94). As a result, assuming Integer N operation (that is, the N divider is not fractional), the DPLL input and output signals are edge aligned (or exhibit a constant time offset due to path latency).

The AD9546 DPLLs provide the ability to impose a programmable time offset (t_{OFST}) between the input and feedback signals by means of a summing node at the feedback input of the digital phase detector. To insert a time offset, use Bits[39:0] (signed) of Register 0x1015 to Register 0x1019 (for DPLL0) and Register 0x1415 to Register 0x1419 (for DPLL1). The value of Bits[39:0] has units of ps.

Two other sources for applying a phase offset also feed the summing node (not explicitly shown in Figure 94). One is via the source profiles (see the Skew Adjustment section of the Source Profiles section), and the other is via the delay compensation feature (see the Delay Compensation section).

Because the DPLL feedback loop results in an average offset of zero between the two inputs of the digital phase detector, t_{OFST} translates to the DPLL output with the polarity shown in Figure 94. The signed nature of Bits[39:0] allows the user to advance or delay the DPLL output signal relative to its input. A positive value for t_{OFST} delays the output signal relative to the input signal. That is, the output edge occurs later. Conversely, a negative value advances the output signal relative to the input signal. Therefore, the output edge occurs earlier.

The relationship between t_{OFST} (sec) and the programmed value for the DPLL phase offset (Bits[39:0]) is

$$DPLL\ Phase\ Offset = t_{OFST}/10^{-12}$$

For example, find the value of the DPLL phase offset necessary to advance the output signal by 75 ns (that is, $t_{OFST} = -75$ ns).

$$DPLL\ Phase\ Offset = (-75 \times 10^{-9})/10^{-12}$$
$$= -75{,}000$$
$$= 0x\ FF\ FFFE\ DB08\ (hexadecimal)$$

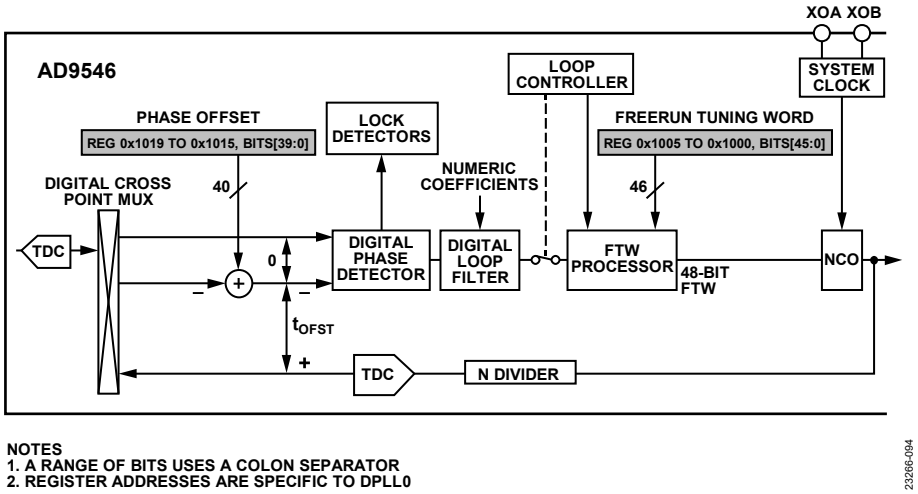


Figure 94. DPLL Phase Offset Feature

TUNING WORD OFFSET CLAMP

The DPLL contains a frequency clamp that places bounds on the frequency range of the DPLL output. The frequency clamp feature (see Figure 95) is beneficial for applications in which downstream devices cannot tolerate frequencies beyond prescribed limits.

The clamp feature uses Bits[23:0] (unsigned) in Register 0x1006 to Register 0x1008 (for DPLL0) and Register 0x1406 to Register 0x1408 (for DPLL1), which constitutes the frequency clamp value. The frequency clamp feature is always active. However, the default value of the frequency clamp is the maximum clamp value, which establishes a default frequency clamp limit of approximately ±586 kHz (for a system clock frequency of 2.4 GHz). Given a 320 MHz NCO output frequency, a 586 kHz frequency clamp equates to an offset of approximately 1800 ppm (or 0.18%).

The frequency offset, f_{CLAMP}, defines an upper and lower frequency bound as a deviation from the frequency given by the freerun tuning word of the DPLL. The relationship between f_{CLAMP}, the system clock frequency (f_S), and the frequency clamp value is as follows:

$$f_{CLAMP} = Frequency\ clamp\ value \times (f_S/2^{36})$$

For example, given a system clock frequency of 2.4 GHz and a desired frequency offset clamp limit of ±10 kHz (that is, $f_S = 2.4 \times 10^9$ and $f_{CLAMP} = 10^4$), find the frequency clamp value. Solving the f_{CLAMP} equation for the frequency clamp value,

$$Frequency\ clamp\ value = 2^{36} \times f_{CLAMP}/f_S \quad (22)$$

$$= 2^{36} \times 10^4/(2.4 \times 10^9)$$

$$= 286{,}331 \text{ (nearest integer)}$$

$$= 0x\ 04\ 5E7B \text{ (hexadecimal)}$$

In some cases, it is more useful to specify f_{CLAMP} as a fractional offset of the NCO output frequency (in percent or parts per million, for example).

For example, given a system clock frequency of 2.4 GHz, an NCO output frequency of 250 MHz and a desired frequency offset clamp limit of 25 ppm (25×10^{-6}), find the frequency clamp value. First, determine f_{CLAMP} as follows:

$$f_{CLAMP} = 25 \times 10^{-6} \times 250 \text{ MHz}$$

$$= 6250 \text{ Hz}$$

Next, substitute f_{CLAMP} into Equation 22.

$$Frequency\ Clamp\ Value = 2^{36} \times f_{CLAMP}/f_S$$

$$= 2^{36} \times 6250/(2.4 \times 10^9)$$

$$= 178{,}957 \text{ (nearest integer)}$$

$$= 0x\ 02\ BB0D \text{ (hexadecimal)}$$

When the frequency clamping circuitry is actively clamping the frequency, status Bit 1 is Logic 1 in Register 0x3102 (for DPLL0) and Register 0x3202 (for DPLL1). Bit 1 relates to the IRQ function (see the Interrupt Request (IRQ) section) through Bit 6 and Bit 7 (frequency clamp active and inactive, respectively) in Register 0x3010 (for DPLL0) and Register 0x3015 (for DPLL2). Because Bit 6 and Bit 7 are latched, the user must use Bit 6 and Bit 7, respectively, of Register 0x200B (for DPLL0) and Register 0x2010 (for DPLL1) to clear the latched state. Otherwise, the user cannot observe subsequent state transitions of Bit 1.

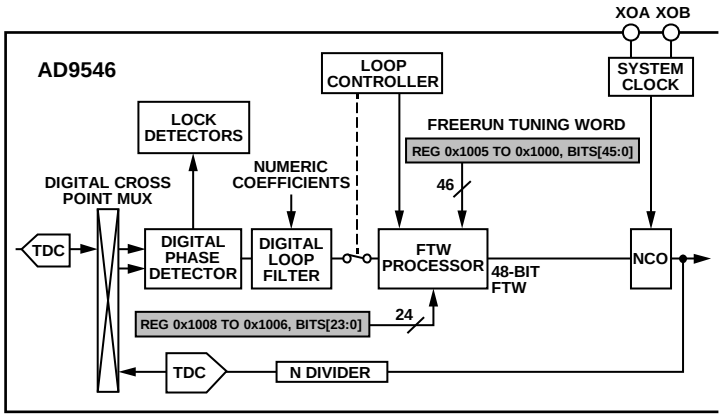


Figure 95. Tuning Word Offset Clamp Feature

PHASE SLEW RATE LIMIT

The digital phase detector uses the reference and feedback TDCs to determine the associated phase error and pass the error along to the loop filter. In addition to the loop phase error, however, other types of phase information can appear in the loop, for example, phase offsets originating from any of the following:

- Changes to the DPLL phase offset (see the DPLL Phase Offset Control section)
- Changes to the phase skew offset (see the Source Profiles section)
- From the delay compensation block (see the Delay Compensation section)
- As the result of a phase buildout operation (see the Phase Buildout Mode section)

For certain phase sources, the DPLL provides a phase slew rate limiting function (see Figure 96) that places an upper bound on the rate of change of phase passed along to the loop filter. The phase slew rate limiter mitigates the adverse effects of injecting a large phase step into the loop by converting the phase step to a phase ramp with the maximum slope set by the user.

The phase slew rate limiter operates only on phase transients introduced by the following phase sources:

- DPLL phase offset (see the DPLL Phase Offset Control section)
- Phase skew offset (see the Source Profiles section)
- Initial phase correction due to activating a hitless mode translation profile (see the Translation Modes section)

That is, the DPLL only applies phase slew limiting under two conditions. The first is a phase offset that exists at the start of a reference acquisition. The second is phase offsets introduced by the user via the DPLL phase offset feature and the phase skew offset feature.

To set the phase slew limit rate, use Bits[31:0] (unsigned) in Register 0x1011 to Register 0x1014 (for DPLL0) and Register 0x1411 to Register 0x1414 (for DPLL1), which constitutes the phase slew limit value. The DPLL phase slew limit value has units of ps/sec or 10^{-12}, but accuracy is typically limited to approximately 50 ps/sec.

The phase slew rate limit feature is active for all DPLL phase slew limit values except zero, which disables the phase slew rate limiter (disabling the phase slew rate limiter is not recommended). However, there is little reason to disable the rate limiter. Instead, programming the rate limiter to the maximum value imposes a slew rate limit corresponding to a frequency shift in excess of 4000 ppm. Most applications do not experience frequency transients exceeding 4000 ppm. The DPLL phase slew limit default value establishes a phase slew rate limit of 100,663,296 ps/sec (which equates to a maximum frequency shift of approximately 100 ppm).

The relationship between the desired phase slew rate limit, $\Delta t/t$, and the value of the DPLL phase slew limit value is as follows:

$$DPLL\ Phase\ Slew\ Limit\ Value = (\Delta t/t) \times 10^{12}$$

For example, given a desired upper bound to the rate of change of phase ($\Delta t/t$) of 0.25 ppm (2.5×10^{-7}), find the required DPLL phase slew limit value as follows:

$$DPLL\ Phase\ Slew\ Limit\ Value = (\Delta t/t) \times 10^{12}$$
$$= (2.5 \times 10^{-7}) \times 10^{12}$$
$$= 250{,}000$$
$$= 0x\ 0003\ D090\ (hexadecimal)$$

When the phase slew limiter is actively limiting the rate of change of phase, status Bit 2 (DPLL phase slew limiting) is Logic 1 of Register 0x3102 (for DPLL0) or Register 0x3202 (for DPLL1). Bit 2 relates to the IRQ function (see the Interrupt Request (IRQ) section) through Bit 4 (DPLL phase slew limiter active) and D5 (DPLL phase slew limiter inactive) in Register 0x3010 (for DPLL0) and Register 0x3015 (for DPLL1). Because Bit 4 and Bit 5 are latched, the user must use Bit 4 and Bit 5, respectively, of Register 0x200B (for DPLL0) and Register 0x2010 (for DPLL1) to clear the latched state. Otherwise, the user cannot observe subsequent state transitions of Bit 2.

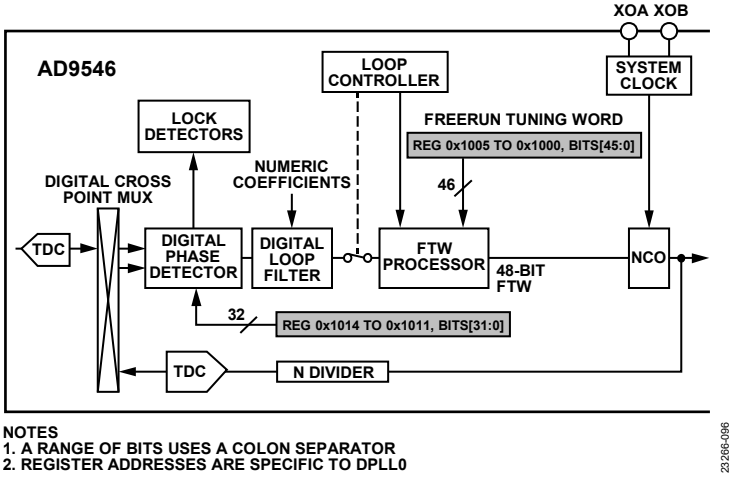


Figure 96. Phase Slew Rate Limit Feature

TUNING WORD HISTORY

The DPLLs have a tuning word processor that handles the application of tuning words to the NCO. The tuning word processor embodies several of the functional blocks shown in Figure 91, including the loop controller, FTW processor, and the switch. The NCO can receive tuning words from the following three possible sources:

- Freerun tuning word
- Digital loop filter
- Tuning word averaging processor

This section focuses on the tuning word averaging processor, which provides the following three digital outputs residing in the register map:

- DPLL tuning word history
- DPLL history available status
- DPLL history updated status

The DPLL tuning word history is a 46-bit unsigned value that resides in Register 0x3103 to Register 0x3108 (for DPLL0) and Register 0x3203 to Register 0x3208 (for DPLL1).

The DPLL history available status is available via Bit 0 of Register 0x3102 (for DPLL0) and Register 0x3202 (for DPLL1).

The DPLL history updated status is available via Bit 2 of Register 0x3011 (for DPLL0) and Register 0x3016 (for DPLL1).

The user also has access to the DPLL history available status and DPLL history updated status as a physical logic level via an appropriately configured Mx pin.

The main purpose of the averaging processor is to compute an average of tuning word samples when a DPLL translation profile initially becomes active (but after expiration of any delays specified by the delay element of the averaging processor as detailed the Averaging Processor Delay section). After the averaging processor collects enough samples to allow a valid tuning word average computation, the processor sets the DPLL history available status bit to Logic 1, indicating that the averaged tuning word history is available. If the DPLL needs to switch to holdover operation, the DPLL can use the averaged tuning word history of the averaging processor. Otherwise, the DPLL uses the last available tuning word from the loop filter or the value in the DPLL freerun tuning word, depending on the configuration of the averaging processor.

The averaging processor comprises the following three functional elements:

- Delay
- Windowed average
- Continue or reset

These functional elements respond to user input via the register map, as explained in the Averaging Processor Delay section, the Averaging Processor Windowed Average section, and the Averaging Processor Continue or Reset section.

Averaging Processor Delay

By default, as soon as a translation profile becomes active (see the Reference Switching section for what constitutes an active translation profile), the tuning word processor resets the averaging processor (and DPLL history available status bit) and the averaging processor immediately starts processing tuning words from the loop filter.

However, the user has access to two independent mechanisms to impose a delay between when a translation profile becomes active and when the averaging processor begins the tuning word averaging process: an event dependent delay and a timed delay.

By default, both mechanisms are inactive, implying no delay. Event dependent delays take priority over time delays: first, any of the three possible event dependent delay selections programmed by the user, then the timed delay programmed by the user. Until these delays expire, the tuning word processor ignores incoming tuning words.

The status of the DPLL is the basis for the event dependent delay mechanism. To invoke the event dependent delay, write a Logic 1 to any combination of the following delay history control bits:

- DPLL delay history until phase lock
- DPLL delay history until frequency lock
- DPLL delay history until not slew limiting

These bits reside in Bits[5:3] of Register 0x100E (for DPLL0) and Register 0x140E (for DPLL1). Logic 1 invokes the described delay. The DPLL delay history until phase lock bit (Bit 3) causes the averaging process to delay until the DPLL phase locks. The DPLL delay history until frequency lock bit (Bit 4) causes the averaging process to delay until the DPLL frequency locks. The DPLL delay until not slew limiting bit (Bit 5) causes the averaging process to delay until the phase slew limiter ceases slew limiting, assuming slew limiting occurs (see the Phase Slew Rate Limit section).

When more than one of the delay history control bits are Logic 1, the implementation of the delay behaves as an AND function of the selected conditions. That is, all the selected status conditions must be satisfied before the averaging process begins. The status conditions are real-time status indicators as they follow the actual state of the DPLL. However, the moment all selected status conditions are true, the averaging processor waits for the prescribed holdoff period to expire (assuming the DPLL history holdoff time is not zero) and starts the averaging process (even if any of the status conditions become false after the averaging processor starts averaging).

The user specifies the desired delay, or holdoff period, Bits[7:0] (unsigned) in Register 0x1010 (for DPLL0) and Register 0x1410 (for DPLL1). A value of zero implies no delay. Bits[7:0] constitute the DPLL history holdoff time. The delay relates to the DPLL history holdoff time value as follows:

$$Delay = DPLL\ History\ Holdoff\ Time \times (t_{ACCUM}/8)$$

where t_{ACCUM} is the time specified by the DPLL history accumulation timer value (see the Averaging Processor Windowed Average section for details on the DPLL history accumulation timer).

The delay (if enabled) is also in effect following a pause/restart trigger event (see the Averaging Processor Continue or Reset section).

Averaging Processor Windowed Average

DPLL History Accumulation Timer

In general, the averaging processor collects tuning word samples generated by the loop filter and computes an average of the tuning word samples. The user specifies an averaging window for the rolling average time span via Bits[27:0] (unsigned) of Register 0x100A to Register 0x100D (for DPLL0) and Register 0x140A to Register 0x140D (for DPLL1). Bits[27:0] constitute the DPLL history accumulation timer value. The units associated with Bits[27:0] are milliseconds allowing averaging time spans from 0.001 sec to 268,435.455 sec (~74.5 hours). The default value is 10 (decimal), yielding a default averaging window time span of 10 ms.

The time span of the averaging window relates to the value of the DPLL history accumulation timer as follows:

$$Time\ Span = DPLL\ History\ Accumulation\ Timer \times 10^{-3}$$

The recommended minimum averaging window time span is at least 10 times the minimum expected period of the signal at the input to the phase/frequency detector of the DPLL.

As an example, determine the value of the DPLL history accumulation timer necessary to provide a 15-minute windowed average. First, solve the time span equation for the DPLL history accumulation timer as follows:

$$DPLL\ History\ Accumulation\ Timer = Time\ Span \times 1000$$
$$= (15 \times 60) \times 1000$$
$$= 900{,}000$$
$$= 0x\ 00D\ BBA0\ (hexadecimal)$$

DPLL history accumulation timer = 0 is a special case. This value bypasses the averaging processor by routing tuning word samples from the digital loop filter to both the DPLL NCO and to the DPLL tuning word history. Thus, programming DPLL history accumulation timer = 0 makes it possible to monitor tuning words directly from the loop filter as they are applied to the DPLL NCO. The DPLL history available status bit (Bit 0 of Register 0x3102 for DPLL0 and Register 0x3202 for DPLL1) is Logic 0 when DPLL history accumulation timer = 0.

Changing the value of the DPLL history accumulation timer exhibits different behavior depending on the current state of the history circuit. The history processor only uses DPLL history accumulation timer during holdoff and active operation, in which case the history processor uses the new value (though there may be a delay associated with the old value before the update takes effect). The history circuit does not change states upon a change of the value of the DPLL history accumulation timer.

Updating the value of the DPLL history accumulation timer while the history circuit is actively collecting history causes future intervals to conform to the new value. If the elapsed period of the current interval exceeds that of the new interval, the next interval begins immediately. Otherwise, the current interval conforms to the new value.

However, if the history circuit is in the holdoff state (see the Averaging Processor Delay section), behavior is slightly different. Because the history circuit averages over 1/8th subintervals (see the Tuning Word Averaging Process section), the progress of the averaging circuit matters. Specifically, the number subintervals already completed remains unchanged regardless of the actual time elapsed, and future subintervals use the new period while the current subintervals continue to use the old value.

Changing the DPLL history holdoff time value (see the Averaging Processor Delay section) has no effect until the beginning of the next holdoff period (in the entirety of the period, not a subinterval). If the behavior associated with the holdoff period, as a result of updating either the history period or the holdoff value, must reflect the new value as soon as possible, the user can restore the history circuit to its initialization state via the DPLL clear history bit (see the DPLL Clear History Bit section).

Tuning Word Averaging Process

During operation, the averaging processor performs averaging by partitioning the time span defined by the value of the DPLL history accumulation timer into eight subintervals. During each subinterval, the averaging processor computes the average of the tuning word samples for that subinterval. A full tuning word average, as defined by the value of the DPLL history accumulation timer, constitutes the average of eight consecutive subinterval averages.

At the termination of each successive subinterval (given the averaging processor has set the DPLL history available status bit), the averaging processor triggers the following events:

- Update the rolling average computation
- Save the result to the DPLL tuning word history
- Update the DPLL history update status bit

As such, at the end of each subinterval, the averaging processor stores the result of the updated rolling average computation in the DPLL tuning word history. At the same time, the averaging processor sets the DPLL history updated status bit to Logic 1. The DPLL history updated status bit provides the user with a

mechanism for knowing when the tuning word history processor refreshes the DPLL tuning word history. However, the DPLL history updated bit is a latched bit. Therefore, this bit indicates a refresh of the DPLL tuning word history, but not necessarily the most recent history.

DPLL Clear History Bit

The user can clear the averaging history at any time via Bit 1 of Register 0x2107 (for DPLL0) and Register 0x2207 (for DPLL1). Logic 1 forces the averaging processor to clear the averaging circuitry, thereby erasing any previous tuning word history information, and forces the DPLL history available bit to Logic 0 (until a newly computed initial average is available).

Programming the DPLL clear history bit to Logic 1 does not clear the contents of the DPLL tuning word history. That is, the last update of the DPLL tuning word history remains intact until the processor calculates a new average and sets the DPLL history updated bit, thereby notifying the user a new average is available.

DPLL Single Sample History Bit

When the DPLL must make a switch from active closed-loop operation to holdover operation, it attempts to use the tuning word history from the averaging processor. However, if the averaging processor has not yet set the DPLL history available status bit (indicating no history is available), the DPLL uses the DPLL freerun tuning word as the tuning word for the NCO by default. Alternatively, the user can have the DPLL use the most recent tuning word from the loop filter instead. To use the most recent tuning word from the loop filter, program Bit 1 to Logic 1 of Register 0x100E (for DPLL0) and Register 0x140E (for DPLL1).

DPLL Quick Start History Bit

Normally, when the DPLL is in active closed-loop operation, the averaging processor must process tuning words for at least the amount of time prescribed by the DPLL history accumulation timer. The prescribed time is necessary to compute a full tuning word history average and to set the DPLL history available bit (thereby indicating the averaging processor has processed enough tuning words to compute an initial average). However, the user has the option to allow the averaging processor to indicate the history is available earlier by writing a Logic 1 to Bit 2 of Register 0x100E (for DPLL0) and Register 0x140E (for DPLL1). Under this condition, the averaging processor sets the DPLL history available status bit after the first two subintervals expire and updates the DPLL tuning word history. Updates of the DPLL tuning word history continue every subinterval thereafter.

During the first eight subintervals of the quick start history sequence, the second and third subintervals exhibit a two-subinterval averaging window. The fourth, fifth, sixth, and seventh subintervals exhibit a four-subinterval averaging window. Finally, the eighth subinterval exhibits an eight-subinterval averaging window, which remains the case for all subsequent subintervals.

The quick start feature is useful when the time span dictated by the DPLL history accumulation timer is very large (hours, for example). The quick start feature allows the averaging processor to make an initial tuning word average available after ¼ of the time indicated by the DPLL history accumulation timer.

The DPLL single sample history and DPLL quick start history bits are not mutually exclusive. Both can be set to provide both options simultaneously with the state of the averaging processor, dictating which option is applicable for the prevailing conditions. Following a switch from active closed-loop operation to holdover operation, the progress made by the averaging processor in computing the initial tuning word average directs the action of the tuning word processor according to the state of the DPLL single sample history bit and the DPLL quick start history bit.

DPLL Persistent History Bit

The DPLL persistent history bit works in conjunction with the control bits described in the Averaging Processor Continue or Reset section. The DPLL persistent history bit is Bit 0 of Register 0x100E (for DPLL0) and Register 0x140E (for DPLL1).

When Bit 0 = 0, the tuning word processor resets the averaging processor based on the control bits described in the Averaging Processor Continue or Reset section. When Bit 0 = 1, this setting prevents the tuning word processor from clearing the averaging processor based on the control bits described in the Averaging Processor Continue or Reset section. Instead, the averaging processor holds onto the previous computation. That is, the computation pauses or persists.

Averaging Processor Continue or Reset

By default, the averaging processor continues to compute a rolling average regardless of the state of the DPLL. The following three bits provide optional control over resetting or pausing the averaging processor based on the DPLL state:

- DPLL pause history while phase unlocked
- DPLL pause history while frequency unlocked
- DPLL pause history while phase slew limiting

These bits pause or reset the averaging process depending on the DPLL persistent history bit. When DPLL persistent history = 0, these three bits cause the averaging processor to reset. When DPLL persistent history = 1, these three bits cause the averaging processor to pause and then resume when the conditions dictate.

The DPLL pause history while phase unlocked bit is Bit 0 of Register 0x100F (for DPLL0) and Register 0x140F (for DPLL1). Programming Bit 0 = 1 pauses or resets the averaging processor while the DPLL is not phase locked.

The DPLL pause history while frequency unlocked bit is Bit 1 of Register 0x100F (for DPLL0) and Register 0x140F (for DPLL1). Programming Bit 1 = 1 pauses or resets the averaging processor while the DPLL is not frequency locked.

The DPLL pause history while phase slew limiting bit is Bit 2 of Register 0x100F (for DPLL0) and Register 0x140F (for DPLL1). Programming Bit 2 = 1 pauses or resets the averaging processor while the phase slew limiter is slew limiting (see the Phase Slew Rate Limit section).

The user can set any combination of these bits. Unlike the averaging processor delay bits, however, when more than one of these bits are Logic 1, the implementation of the delay behaves as an OR function of the selected conditions. That is, satisfying any of the selected status conditions causes the averaging processor to reset or pause.

DELAY COMPENSATION

Overview

Delay compensation provides an automatic phase offset to the DPLL to counteract temperature dependent delay variation. Figure 97 is a block diagram of the DPLL and delay compensation block. Delay compensation is only in effect while the DPLL is active. Furthermore, the response time of the delay compensation at the DPLL output is subject to the loop bandwidth of the DPLL.

Delay compensation generates the temperature dependent time offset value, Δt, based on a fifth degree polynomial, as shown in Figure 97. The independent variable of the polynomial is temperature (T in Figure 97). The five polynomial coefficients (C_1 to C_5), which the user enters via the register map, scale with corresponding powers of T. Note that there is no need for a constant term (C_0) in the delay compensation block, because the DPLL phase offset feature (see the DPLL Phase Offset Control section) provides that function. The Δt computation implied by Figure 97 is

$$\Delta t = \sum_{k=1}^{5} C_k T^k$$

The temperature (T) input into the open-loop direct compensation block originates from temperature sensor circuitry (see the Temperature Sensor section). The user has the option of choosing the temperature source: either the internal temperature sensor (default) or a user supplied temperature via the register map. When using the internal temperature sensor, the device inherently uses the appropriate scaling units. However, when using the register map option, the user must provide the proper scaling (see the External Temperature Source section for details).

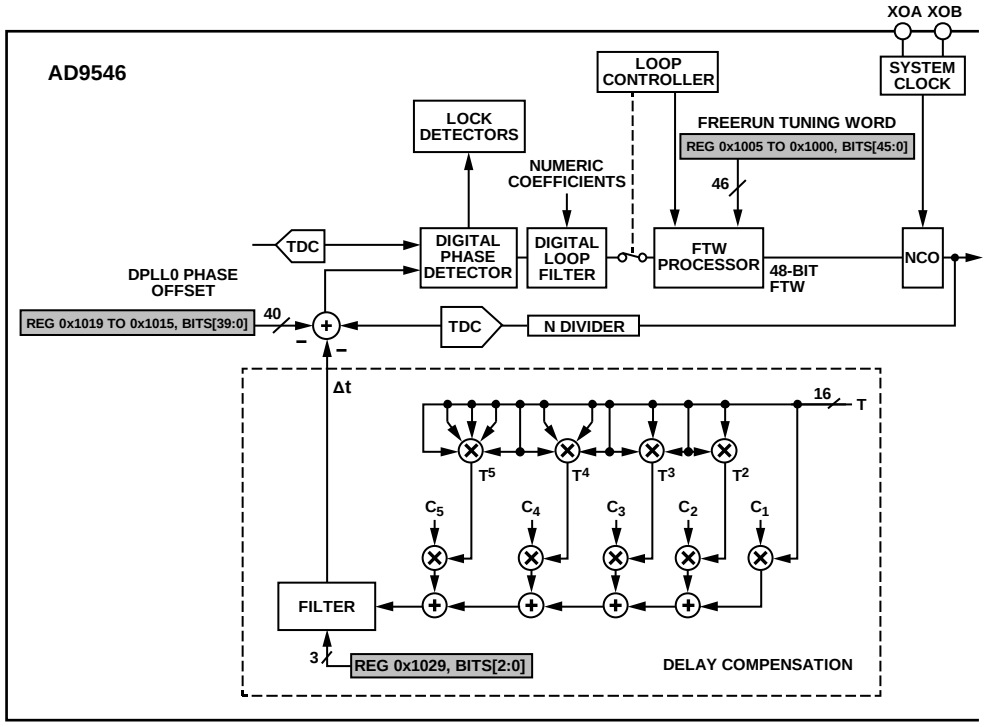


Figure 97. Delay Compensation

Delay Compensation Coefficients (C_k) Programming

Coefficient C_k (where k = 1 to 5) applies scale factors to the appropriate powers of T. The C_k coefficients carry units of sec/°C. Each coefficient comprises a significand component and an exponent component. The user programs the significand and exponent components independently in the register map. The coefficient components reside in the register map per Table 84.

Table 84. Delay Compensation Coefficient Address Ranges

		Register Address	
DPLL	Coefficient	Significand	Exponent
0	C_1	0x101A to 0x101B	0x101C
0	C_2	0x101D to 0x101E	0x101F
0	C_3	0x1020 to 0x1021	0x1022
0	C_4	0x1023 to 0x1024	0x1025
0	C_5	0x1026 to 0x1027	0x1028
1	C_1	0x141A to 0x141B	0x141C
1	C_2	0x141D to 0x141E	0x141F
1	C_3	0x1420 to 0x1421	0x1422
1	C_4	0x1423 to 0x1424	0x1425
1	C_5	0x1426 to 0x1427	0x1428

The significand and exponent values are signed (twos complement) numbers and program in exactly the same manner as the significand and exponent of the T^k values for system clock compensation (see the Compensation Method 1 section for details on converting decimal coefficient values to signed, twos complement, significand and exponent values).

For example, suppose a user makes measurements that indicate the output clock signals of the AD9546 exhibit a delay variation of −1 ns/°C. To compensate for this variation, apply a 1 ns/°C correction. In polynomial form,

$$f(T) = C_5T^5 + C_4T^4 + C_3T^3 + C_2T^2 + C_1T$$

where $f(T)$ is the correction value in units of seconds as a function of temperature (T).

Because the delay variation in this example is strictly linear with respect to temperature, all the coefficients are zero except for C_1. Thus, in this example, the compensation polynomial simplifies to the following:

$$f(T) = C_1T$$

To apply 1 ns/°C compensation, $C_1 = 10^{-9}$ sec/°C. Referring to the procedure in the Compensation Method 1 section, calculate the following for C_1:

$$C1\_ExpVal = \left\lfloor \frac{\log|C_1|}{\log 2} \right\rfloor + 1$$

$$= \left\lfloor \frac{\log|10^{-9}|}{\log 2} \right\rfloor + 1$$

$$= -29$$

Because −29 is greater than the quantization limit of −127, the C_1 exponent is

$$C_1\ Exponent = C1\_ExpVal$$

$$= -29$$

$$= 0xE3 \text{ (hexadecimal)}$$

The C_1 significand is

$$C_1\ Significand = C_1 \times 2^{15 - C1\_ExpVal}$$

$$= 10^{-9} \times 2^{15 - (-29)}$$

$$= 17{,}592 \text{ (nearest integer)}$$

$$= 0x44B8 \text{ (hexadecimal)}$$

Delay Compensation Filter

The Δt values produced by the delay compensation block depend on temperature measurements (internal or via the register map) as an input parameter. Any noise associated with those measurements is a potential noise source on Δt, which can lead to a degradation of the phase noise performance of the DPLL. To mitigate potential noise injection, the delay compensation block uses a filter that applies a smoothing function to the raw Δt values.

The phase slew limiter (see the Phase Slew Rate Limit section) does not process time variations injected into the DPLL by the delay compensation filter. That is, these variations affect the DPLL output without intervention by the phase slew limiter.

The user controls the filter bandwidth via Bits[2:0] (unsigned) of Register 0x1029 (for DPLL0) and Register 0x1429 (for DPLL1). The filter comprises a single-pole response yielding a 3 dB bandwidth per Table 85 (which also shows the transition time to 99% of a step input, associated with a step change at the input to the filter).

Table 85. Delay Compensation Filter Bandwidth

Bits[2:0]	3 dB Bandwidth (Hz)	Transition Time (ms)
000	240	3.27
001	120	6.58
010	60	13.0
011	30	26.5
100	15	53.1
101	7.6	99.5
110	3.8	199
111	1.9	398

TIME STAMP TAGGING OPTIONS

The input to the DPLL consists of time stamps arriving from two TDCs: a reference input TDC and a feedback TDC. Furthermore, the user can enable tagging of time stamps originating from those TDCs (see the Tagged Time Stamps section). The user can also enable the DPLL to operate based on tagged time stamps (rather than every time stamp) originating from the reference and/or feedback TDCs.

The use of tagged time stamps by the DPLL only applies for zero delay (hitless) mode. The DPLL ignores tagged time stamps in phase buildout mode. See the Frequency Translation Loops section for an explanation of zero delay and phase buildout modes.

The user establishes the tagged time stamp functionality of the DPLL via Bits[4:2] of the appropriate translation profile at the start address shown in Table 74 plus an offset of 3 (decimal). There are five tagged operating modes per Table 86.

Table 86. Tagged Time Stamp Operating Modes

Bits[4:2]	Tag Mode
0	No tagging (default)
1	Reference tagging only
2	Feedback tagging only
3	Combined reference and feedback tagging with equal carrier rates
4	Combined reference and feedback tagging with unequal carrier rates
5 to 7	Not applicable

With respect to tagged time stamps, the carrier rate is the rate at which the TDC generates time stamps (that is, the underlying rate of the input clock to the TDC). The tagged rate is the rate at which the TDC generates tagged time stamps. The tagged rate is always an integer submultiple of the carrier rate.

The no tagging mode is the normal operating mode of the DPLL. That is, the digital PFD processes every reference and feedback time stamp whether the time stamp is a tagged time stamp or not.

The reference tagging only mode tells the digital PFD to use tagged reference time stamps for phase alignment purposes. The user must ensure the reference is set up to provide tagged time stamps. When the reference is one of the REFx or auxiliary REFx inputs, the reference demodulator is the source of tagged time stamps (see the Reference Demodulator section). When the reference is an auxiliary NCO, the auxiliary NCO is the source of tagged time stamps (see the Auxiliary NCOs section).

The feedback tagging only mode tells the digital PFD to use tagged feedback time stamps for phase alignment purposes. The user must ensure the feedback path is set up to provide tagged time stamps via output clock modulation (see the Distribution Embedded Output Clock Modulation section).

The combined reference and feedback tagging modes tell the digital PFD to use tagged time stamps from both the reference and feedback TDCs. The user must ensure that the reference and feedback paths are set up to provide tagged time stamps. Furthermore, the user must ensure that the tag rate of the reference path matches the tag rate of the feedback path.

There are two variants of the combined reference and feedback tagging modes: equal and unequal carrier rates. The combined reference and feedback tagging modes require equal tag rates on both paths. There is no such requirement on the carrier rates for both paths. However, to function properly, the DPLL requires advance knowledge of the relationship between the reference and feedback carrier rates, which is the reason for the two variants. The user must select the appropriate variant based on whether the reference and feedback TDC carrier rates differ.

CASCADED DPLL CONFIGURATION

The cascaded DPLL configuration applies to applications that lock both DPLLs to the same reference source with both DPLLs remaining phase locked, even if all reference sources are lost. To understand cascaded configurations, first consider device operation without cascaded capability, as shown in Figure 98. As a prerequisite, the reader must be familiar with the Translation Profiles section and the Reference Switching section. Figure 98 and Figure 99 show simplistic diagrams of the DPLL showing a phase buildout configuration, but the zero delay (hitless) configuration is valid as well.

The far left side of Figure 98 shows two valid reference sources with DPLL0 and DPLL1 using Translation Profile 0.a and Translation Profile 1.z, respectively, assigned to Reference Source 1, where a and z are variables that can be any integer value from 0 to 5. Therefore, both DPLLs lock to a common reference as desired.

The middle portion of Figure 98 shows the result of losing Reference Source 1. The loss of Reference Source 1 results in both DPLLs independently searching for a new reference source, assuming automatic reference switching is in effect (see the Reference Switching section). Presumably, the user has defined translation profiles in such a way that DPLL0 finds Translation Profile 0.b and DPLL1 finds Translation Profile 1.y (where b and y are variables that can be any integer value from 0 to 5, but not the same value as the a and z variables) with both profiles assigning the same valid reference source (Reference Source 2, in this example). As such, both DPLLs again lock to a common reference as desired.

The far right hand side of Figure 98 shows the result of losing Reference Source 2, leaving no valid reference sources. In this case, both DPLLs have no recourse but to switch to holdover or freerun mode. However, because the DPLLs may not have the same FTW in effect when they switch to holdover or freerun mode, their outputs are no longer phase locked, which breaks the original requirement that both DPLLs remain phase locked when all reference sources are lost. The cascaded configuration provides a solution to this problem.

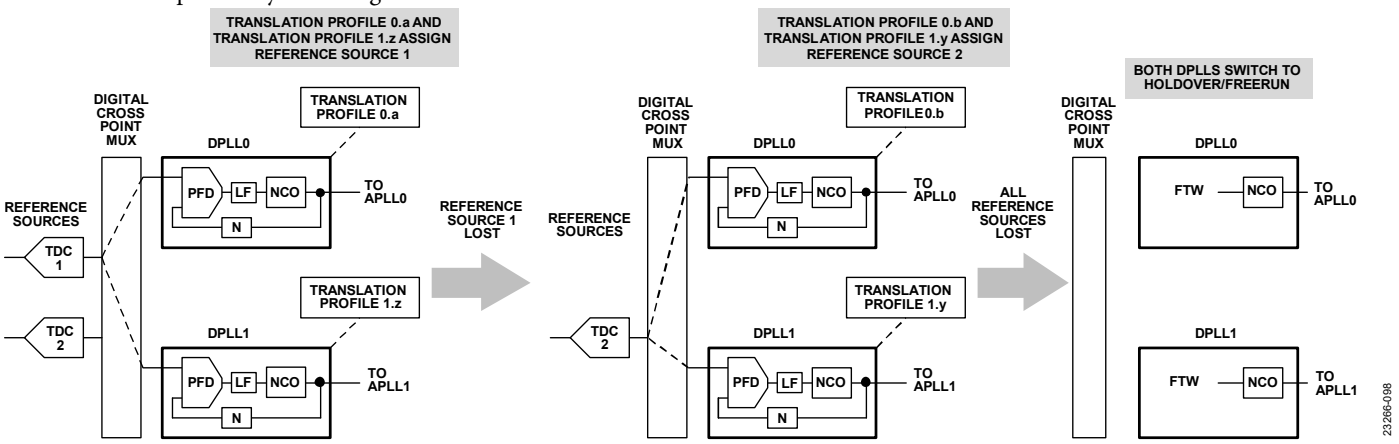


Figure 98. Noncascaded DPLL Operation

Figure 99 shows cascaded DPLL configuration and its operation. The main difference between Figure 98 and Figure 99 is that DPLL1 has Translation Profile 1.x configured to use the feedback path of DPLL0 as a reference input. Setting up Translation Profile 1.x in this way establishes the potential for cascaded DPLL operation. The fact that Translation Profile 1.x is set up to use the feedback path of DPLL0 as a reference input identifies DPLL1 as the secondary DPLL and DPLL0 as the primary DPLL in the cascaded DPLL configuration (see the right side of Figure 99).

The existence of Translation Profile 1.x is one of two conditions necessary to enable cascaded DPLL operation. The other condition is that the primary DPLL must have an assigned inactive profile. The user makes this assignment via Bits[2:0] of Register 0x102A (for DPLL0) or Register 0x142A (for DPLL1). By default, Bits[2:0] = 000 (binary), which assigns Translation Profile 0.0 (for DPLL0) or Translation Profile 1.0 (for DPLL1) as the inactive profile assignment, but the user can specify any one of the six translation profiles as the inactive profile via Bits[2:0]. Programming Bits[2:0] with a value of x, where x is 0 to 5 (decimal) selects the corresponding Translation Profile 0.x or 1.x, where x can be any integer value from 0 to 5). Figure 99 indicates Translation Profile 0.b as the inactive profile to demonstrate a nondefault example.

Consider the left side of Figure 99, which is the same as in Figure 98 except for the fact that DPLL0 has Translation Profile B assigned as its inactive profile. Like Figure 98, both DPLLs lock to Reference Source 1. The middle portion of Figure 99 applies when Reference Source 1 is lost. Like Figure 98, both DPLLs lock to Reference Source 2. When Reference Source 2 is lost (as shown in the right portion of Figure 99), however, and assuming the user has configured a translation profile (Translation Profile 1.x in this example) to use the feedback path of the primary DPLL as a reference source, the following events occur.

When DPLL0 loses its reference, it has no recourse but to switch to holdover or freerun mode because no reference sources are available. However, when Translation Profile 0.b becomes inactive, because Translation Profile B is the inactive profile index for DPLL0, the feedback path of the DPLL0 is available as a valid reference for DPLL1. Although DPLL0 switches to holdover or freerun, DPLL1 does not because it has an available reference source: the feedback path of DPLL0. DPLL1 may experience a momentary switch to holdover or freerun mode before switching to cascaded DPLL operation. The result is cascaded DPLL operation, in which the reference input of DPLL1 (the secondary DPLL) connects to the feedback path of DPLL0 (the primary DPLL). In this way, the two DPLL outputs are phase locked, which preserves the original requirement of phase locked DPLL outputs even when all reference sources are lost.

The role of the primary and secondary DPLLs is reversible because the role is dependent on how the user programs the translation profiles and assigns the inactive profile index.

For the cascaded DPLL configuration to become active, the following conditions are necessary:

- A translation profile associated with the secondary DPLL that assigns the feedback path of the primary DPLL as a reference input must exist.
- The primary DPLL must have an inactive translation profile assignment (by default, the inactive profile is Translation Profile 0 unless otherwise specified).
- The primary channel of the APLL must be calibrated and indicate locked status.
- The primary DPLL must be inactive, that is, either in freerun or holdover operation.

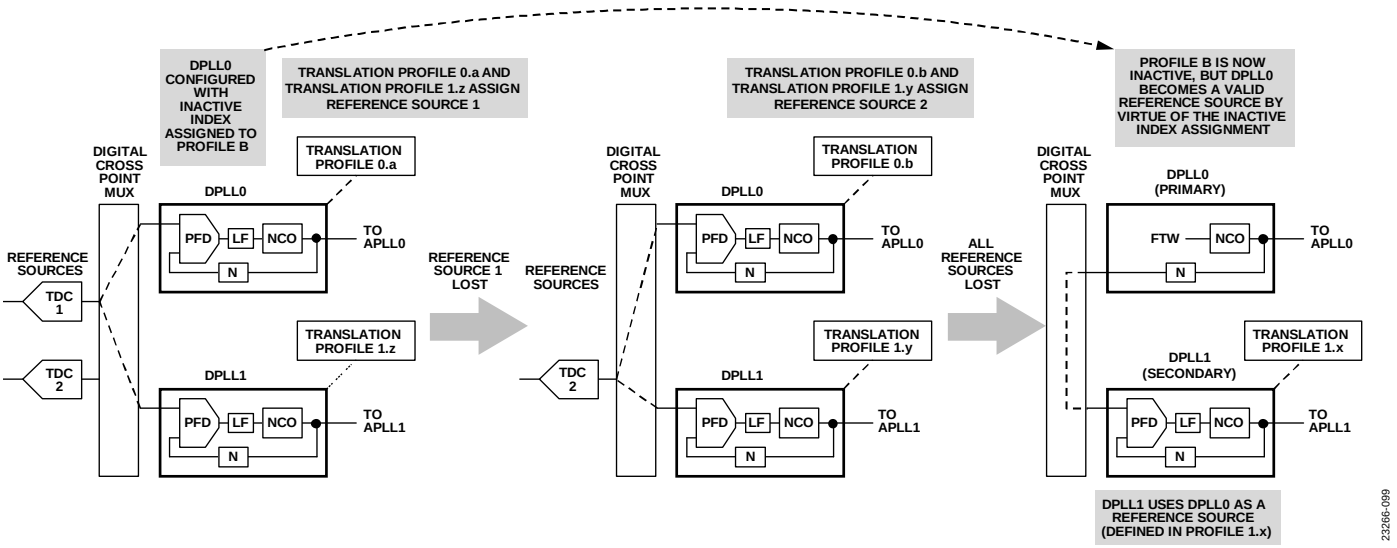


Figure 99. Cascaded DPLL Configuration

CAVEATS OF CASCADED DPLL OPERATION

The recommendation is to avoid using reference synchronization when employing cascaded DPLL operation (see the Reference Synchronization section).

When the profile targeted by the assigned inactive profile index becomes inactive, certain aspects of the target profile configuration still matter (even though the target profile is inactive), as follows:

- The translation type (phase buildout or hitless—see the Translation Modes section). For phase buildout mode, the APLL associated with the primary DPLL must be in a calibrated and locked state. For hitless (internal) mode, the APLL associated with the primary DPLL must be in a calibrated and locked state, and the selected distribution path must be active (that is, a synchronization event has occurred (see the Distribution Output Clock Synchronization section) on the channel associated with the secondary DPLL).
- The device uses the information related to the feedback configuration of the primary DPLL (even though the primary DPLL is effectively operating open loop, that is, holdover or freerun).
- The user cannot specify fractional feedback division for the primary DPLL (that is, the device ignores the fractional portion of the feedback divider when operating in the cascaded DPLL configuration). Furthermore, the user must properly program the buildout N divider value in the targeted inactive profile (even if the targeted inactive profile is a hitless (internal) profile). See the DPLL Feedback Divider (N Divider) section for details relating to the buildout N divider.
- The user cannot specify the external zero delay configuration for cascaded DPLL operation.
- The device uses the information related to time stamp tagging according to the primary DPLL. Specifically, the tagging mode defined for the feedback path of the primary DPLL determines whether time stamp tags are available on the reference input path of the secondary DPLL. Thus, the targeted inactive profile for the primary DPLL must use a value of 2, 3, or 4 as the tag mode. In addition, the translation profile for the secondary DPLL must use a value of 1, 3, or 4 as the tag mode (see Table 86 regarding the tag modes).

ANALOG PLL (APLL)

APLL OVERVIEW

Figure 100 shows a block diagram of the APLL, which is valid for both APLL0 and APLL1. The main components of the APLL comprise the following:

- An integrated VCO
- A combined PFD and charge pump
- A loop filter
- A feedback divider (M divider)

The purpose of the APLL is

- To provide a low noise output clock signal
- To upconvert the DPLL output frequency to ~3 GHz
- To suppress the spurious artifacts of the NCO

VOLTAGE CONTROLLED OSCILLATOR (VCO)

The VCO provides a spurious free, low noise RF clock signal in the 3 GHz range that feeds the output distribution section, with APLL0 as the source for the OUT0x outputs and APLL1 as the source for the OUT1x outputs. The VCOs for APLL0 and APLL1 cover nonoverlapping frequency ranges, as shown in Table 87. The output frequency of the APPL is one half of the VCO frequency due to the P divider (see Figure 100).

Table 87. APLL VCO Frequency and Gain

APLL	VCO Frequency (MHz)	VCO Gain (MHz/V)
0	2424 to 3232	60
1	3232 to 4040	100

The VCO has a frequency range that is approximately 800 MHz, which consists of many narrow, overlapping frequency bands. To ensure the VCO uses the appropriate band for a given output frequency and to accommodate proper operation over a broad temperature range, the AD9546 incorporates an automated VCO calibration feature.

VCO Calibration

To initiate VCO calibration of APLL0 or APLL1 independently, write a Logic 1 to Bit 1 of Register 0x2100 (for APLL0) and Register 0x2200 (for APLL1). Bit 0 is not autoclearing. Therefore, the user must immediately restore Bit 0 to Logic 0, because maintaining a static Logic 1 state may cause unexpected device behavior.

To calibrate both APLL channels together, instead of each channel independently, write a Logic 1 to Bit 1 of Register 0x2000. Bit 1 calibrates the system clock PLL as well as both APLLs.

One option for checking the status of the APLL VCO calibration process is via Bits[5:4] of Register 0x3100 (for APPL0) and Register 0x3200 (for APLL1). Bit 4, when Logic 1, indicates the APPL is busy calibrating. Bit 5, when Logic 1, indicates the APPL is done calibrating. Bit 4 is also available as a physical output signal via an appropriately configured Mx pin.

The other option for checking the status of the APLL VCO calibration process is via Bits[1:0] of Register 0x3014 (for APLL0) and Register 0x3019 (for APLL1). Because Bit 0 and Bit 1 represent the latched state transitions of Bit 4 and Bit 5, respectively, they may represent a condition that is no longer true. Therefore, the user must clear Bit 0 and Bit 1 via Bit 0 and Bit 1, respectively, of Register 0x200F for APLL0 and Register 0x2014 for APLL1. Otherwise, the user may lose indication of subsequent state changes of the VCO calibration process.

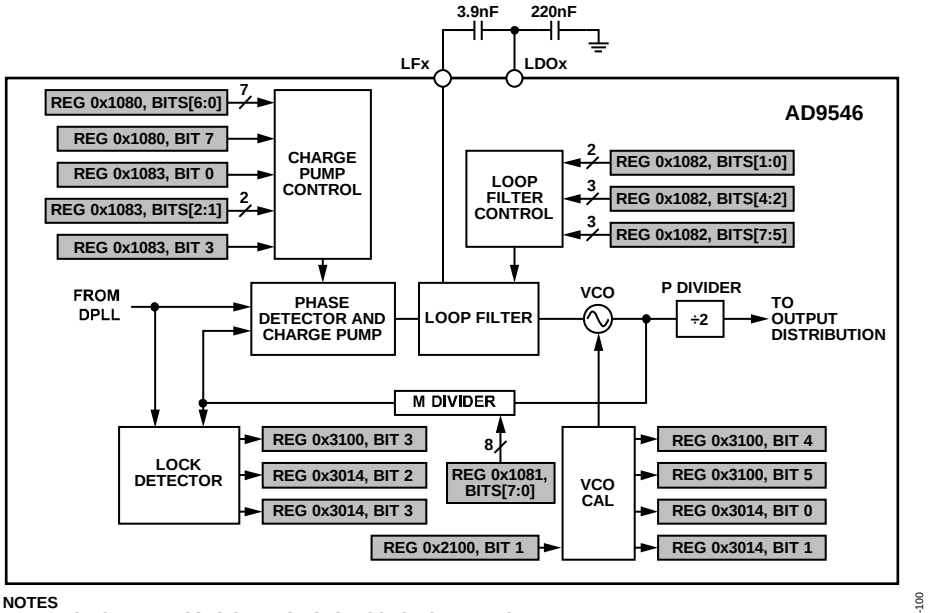


Figure 100. APLL Block Diagram

APLL FEEDBACK DIVIDER (M DIVIDER)

The user programs the APLL feedback M divider via Bits[7:0] in Register 0x1081 (for APLL0) and Register 0x1481 (for APLL1). The divide ratio of the M divider is the decimal value of Bits[7:0], yielding divide ratios from 1 to 255. A value of 0 is the same as 1.

For example, given a desired divide ratio of 27, program Bits[7:0] = 27 (decimal) or 0x1B (hexadecimal).

PHASE FREQUENCY DETECTOR (PFD)

The PFD detects the instantaneous phase error between the feedback signal from the M divider and the output from the DPLL. The phase error essentially drives the servo loop of the APLL in a manner that ultimately nulls out the phase difference between the two signals. The PFD bandwidth is wide enough to handle signals originating from the NCO of the DPLL (up to ~350 MHz nominal).

The APLLs incorporate lock detection circuitry that indicates when they achieve a frequency locked condition. Locked status indication is available via Bit 3 of Register 0x3100 (for APLL0) and Register 0x3200 (for APLL1), where Logic 1 indicates locked status. Bit 3 is also available as a physical output signal via an appropriately configured Mx pin.

The IRQ section of the register map provides latched status bits reflecting state transitions of the lock detector status via Bits[3:2] of Register 0x3014 (for APLL0) and Register 0x3019 (for APLL1). Bit 2 latches to Logic 1 on an unlocked to locked state transition, whereas Bit 3 latches to Logic 1 on a locked to unlocked state transition. Because Bit 2 and Bit 3 are latched bits, the user must clear Bit 2 and Bit 3 via Bit 2 and Bit 3, respectively, of Register 0x200F (for APLL0) and Register 0x2014 (APLL1). Otherwise, the user may lose indication of subsequent state changes of the lock detector.

CHARGE PUMP

The charge pump consists of a pair of constant current sources that deliver charge to or remove charge from the loop filter based on the output of the phase detector. The transfer of charge increases or decreases the voltage applied to the VCO, which steers the VCO frequency to match the input frequency and ultimately bring about a phase locked condition.

The charge pump has two operating modes, manual and automatic, selectable via Bit 7 of Register 0x1080 (for APLL0) and Register 0x1480 (for APLL1). Logic 1 (default) selects manual mode, whereas Logic 0 selects automatic mode.

In manual mode, the user programs the charge pump current via Bits[6:0] (unsigned) of Register 0x1080 and Register 0x1480. The actual charge pump current (I_{CP}) relates to Bits[6:0] (charge pump scale) as follows:

$$I_{CP} = Charge\ Pump\ Scale \times 8\ \mu A \quad (23)$$

I_{CP} has a range of 0 µA to 1016 µA. The default value of Bits[6:0] is 0x14 (20 decimal), which yields I_{CP} = 160 µA.

For example, given I_{CP} = 743 µA, determine the charge pump scale value. Solving Equation 23 for the charge pump scale yields

$$Charge\ Pump\ Scale = I_{CP}/8\ \mu A$$
$$= 743\ \mu A/8\ \mu A$$
$$= 93\ (\text{nearest integer})$$
$$= 0x5D\ (\text{hexadecimal})$$

In automatic mode, the charge pump current scale value is ineffective. Instead, the APLL automatically adjusts the charge pump current (I_{CP}) based on the value of the M divider according to Table 88. This automatic adjustment yields a relatively constant loop bandwidth for M divider values from 1 to 63. The loop bandwidth is ~250 kHz for APLL0 and ~300 kHz for APLL1. Note that automatic charge pump control is only valid over a subset of M divider values. Given this constraint, it is generally best practice to avoid automatic mode.

Table 88. APLL Charge Pump Current in Automatic Mode

M Divider Value	I_{CP} (µA)
1 to 63	M × 16
64 to 255	1016

Regardless of the operating mode (manual or automatic), the charge pump has a provision for applying a constant dc offset current to the output of the charge pump. Injection of an offset current overcomes some of the spectral artifacts associated with charge pump nonlinearity when the APLL is in a locked state. Generally, noise performance improves significantly when this feature is active and properly adjusted.

To enable or disable the dc offset current feature, use Bit 0 of Register 0x1083 (for APLL0) and Register 0x1483 (for APLL1). Logic 1 (default) enables this feature, whereas Logic 0 disables it. To control the polarity (positive or negative) of the offset current, use Bit 3 of Register 0x1083 (for APLL0) and Register 0x1483 (for APLL1). Logic 0 (default) is positive, whereas Logic 1 is negative. The magnitude of the offset current depends on the value of Bits[2:1] (unsigned) of Register 0x1083 (for APLL0) and Register 0x1483 (for APLL1). The value of this bit field sets the dc offset current as a fraction of I_{CP} per Table 89, but with a granularity of 8 µA. Thus, the offset current is in integer steps of 8 µA, with fractions of an 8 µA step rounded in the direction of zero (that is, −53 µA rounds to −48 µA).

Table 89. APLL Charge Pump DC Offset Current

Bits[2:1] (Decimal)	DC Offset Current (% of I_{CP})
0	50
1	25 (default)
2	12.5
3	6.25

In general, the default values of the dc offset current and charge pump scale yield optimal overall performance. For this reason, the recommendation is to use only the manual charge pump mode (refer to the Charge Pump section regarding manual charge pump mode).

APLL LOOP FILTER

The phase errors detected by the PFD drive the charge pump. The charge pump transfers charge to the loop filter in proportion to the amount of phase error. The loop filter stores the charge, which creates a dc voltage that steers the VCO frequency in a manner that nulls out the phase error.

The loop filter is a resistor and capacitor filter with a low-pass response characteristic that serves three purposes.

- Storage of charge originating from the charge pump
- Attenuation of the high frequency components of the phase error signal
- Stabilization of the feedback loop

Figure 101 shows the loop filter schematic. Three of the components (R1, R3, and C2) are programmable, allowing the user to tailor the response of the loop filter. The two external capacitors (3.9 nF and 220 nF) are connected to the LFx and LDOx pins. These two components are necessary for proper operation of the loop filter and the internal LDO, respectively.

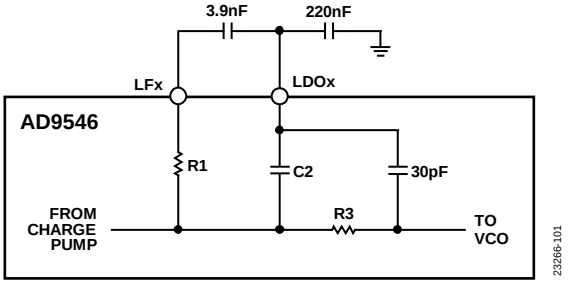


Figure 101. APLL Loop Filter

To program R1, C2 and R3 use Register 0x1082 (for APLL0) and Register 0x1482 (for APLL1). Bits[7:5] (unsigned) control R1, Bits[4:2] (unsigned) control C2, and Bits[1:0] (unsigned) control R3. Table 90 shows how the decimal value of the bit fields that select R1, C2 and R3 relate to the physical loop filter component value.

Table 90. APLL Loop Filter Component Values

Bit Field Value (Decimal)	R1 (Ω)	C2 (pF)	R3 (Ω)
0	0	8 (default)	200 (default)
1	250	24	250
2	500	40	333
3	750	56	500
4	1000	72	n/a
5	1250	88	n/a
6	1500	104	n/a
7	1750 (default)	120	n/a

In general, the default filter settings yield optimal performance. Although altering the response of the APLL loop filter is not necessary, the user can contact Analog Devices for assistance with determining alternative component values for the APLL loop filter to tailor the response to specific requirements.

REFERENCE SWITCHING

REFERENCE SWITCHING OVERVIEW

The AD9546 has a sophisticated reference switching mechanism administered by an internal controller. In general, the DPLL operates in closed-loop mode, wherein the DPLL locks to the frequency and phase of a reference input signal. However, if a reference becomes invalid, the AD9546 can automatically switch to a different reference input, depending on how the user programs translation profiles (see the Translation Profiles section). In the absence of any valid references, the AD9546 switches to one of the two open-loop operating modes: holdover or freerun. The DPLL selects freerun, holdover, or manual/automatic source selection per the DPLL mode selection flowchart of Figure 102. The three procedures in Figure 102 (holdover, translation profile validation, and priority-based translation profile selection) have separate flow charts per Figure 103, Figure 104, and Figure 105, respectively.

The DPLL has three status indicators to notify the user when the DPLL begins the switch to a new translation profile (inferring a new reference input) or when the DPLL enters freerun or holdover mode. These indicators reside in Bits[2:0] of Register 0x3101 and Register 0x3201, as follows:

- Bit 2: switching translation profiles
- Bit 1: holdover mode
- Bit 0: freerun mode

These indicators also inform corresponding IRQ status bits via Bits[7:5] of Register 0x3011 and Register 0x3016, as follows:

- Bit 7: switched translation profiles
- Bit 6: entered freerun mode
- Bit 5: entered holdover mode

Bit 7 latches to Logic 1 when the DPLL is in the process of switching to an active translation profile. Bit 6 latches to Logic 1 when the DPLL enters freerun mode. Bit 5 latches to Logic 1 when the DPLL enters holdover mode. Because Bits[7:5] are latched bits, the user must clear them via Bits[7:5] of Register 0x200C and Register 0x2011 to obtain visibility of subsequent state transitions to reference switching, freerun, or holdover (see the Interrupt Request (IRQ) section).

FORCED FREERUN MODE

Although the automatic reference switching capability of the AD9546 can cause the DPLL to switch to freerun mode, the user can force the DPLL to freerun mode by writing Logic 1 to Bit 0 of Register 0x2105 (DPLL0) or Register 0x2205 (DPLL1). In freerun mode, the DPLL operates open loop and produces a static output frequency as dictated by the 46-bit freerun tuning word in Register 0x1000 to Register 0x1005 (DPLL0) and Register 0x1400 to Register 0x1405 (DPLL1). Refer to the Freerun Tuning Word section for details.

FORCED HOLDOVER MODE

Although the automatic reference switching capability of the AD9546 can cause the DPLL to switch to holdover mode, the user can force the DPLL to holdover mode by writing Logic 1 to Bit 1 of Register 0x2105 (DPLL0) or Register 0x2205 (DPLL1). In holdover mode, like freerun mode, the DPLL operates open loop and produces a static output frequency. However, the DPLL uses one of three static tuning words dictated by the tuning word history processor (see the Tuning Word History section). Figure 103 shows a summary of the holdover tuning word selection.

Note that freerun mode overrides holdover mode (see Figure 102). Furthermore, freerun mode and holdover mode both override the manual/automatic source selection modes (see the Manual/Automatic Translation Profile Selection section).

MANUAL/AUTOMATIC TRANSLATION PROFILE SELECTION

Assuming Bits[1:0] = 1 decimal in Register 0x2105 (DPLL0) or Register 0x2205 (DPLL1), the DPLL selects a translation profile based on Bits[3:2] of Register 0x2105 (DPLL0) or Register 0x2205 (DPLL1). Bits[3:2] support four translation profile selection modes per Table 91.

Table 91. DPLL Translation Profile Select Mode Settings

Bits[3:2] (Decimal)	Description
0	Automatic, priority-based translation profile selection
1	Manual translation profile selection with fallback to priority-based translation profile selection
2	Manual translation profile selection with fallback to holdover
3	Manual only translation profile selection

Manual Only Translation Profile Selection

The manual only translation profile selection mode in Table 91 (Bits[3:2] = 3 decimal) is not a recommended operating mode because this setting includes several operational caveats. Contact Analog Devices for application support on using the manual only translation profile selection mode.

Manual Translation Profile Selection with Fallback to Holdover

In the manual translation profile selection with fallback to holdover mode (Bits[3:2] = 2 decimal), the user specifies a particular translation profile for manual operation using Bits[6:4] of Register 0x2105 (DPLL0) or 0x2205 (DPLL1). The decimal value, y, of Bits[6:4] identifies the desired translation profile, x.y (y is 0 to 5 and x is 0 or 1 for DPLL0 or DPLL1, respectively). If the specified translation profile is not valid, the DPLL switches to holdover operation (open loop). However, depending on the status of the tuning word history processor (see the Tuning Word History section), the DPLL may select freerun operation instead of holdover.

Manual Translation Profile Selection with Fallback to Priority-Based Translation Profile Selection

In the manual translation profile selection with fallback to priority-based translation profile selection mode (Bits[3:2] = 1 decimal), the user specifies a particular translation profile for manual operation in the same manner as described in the Manual Translation Profile Selection with Fallback to Holdover section. However, if the selected profile is not valid, rather than switching to holdover mode, the DPLL attempts to use the priority-based selection process (see the Automatic, Priority-Based Translation Profile Selection section for a description of priority-based selection). If the DPLL is unable to find a translation profile per the priority selection process, the DPLL switches to holdover or freerun operation per Figure 103.

Automatic, Priority-Based Translation Profile Selection

The priority associated with a particular translation profile depends on the programmed value of Bits[5:1] of the following register addresses: 0x1200, 0x1220, 0x1240, 0x1260, 0x1280, and 0x12A0 for DPLL0 and 0x1600, 0x1620, 0x1640, 0x1660, 0x1680, and 0x16A0 for DPLL1. A lower value of Bits[5:1] means a higher priority of the associated input reference source. That is, 31 is the lowest (least favored) selection priority, and 0 is the highest (most favored) selection priority.

In the automatic, priority-based translation profile selection mode (Bits[3:2] = 0 decimal), the DPLL relies solely on the automatic priority selection process shown in Figure 105 for choosing a translation profile. However, if the DPLL is unable to find a translation profile per the priority selection process, the DPLL switches to holdover or freerun operation per Figure 103.

As shown in Figure 105, the DPLL selects the most favored translation profile when the current profile becomes invalid.

Translation Profile Validation

As shown in Figure 102, the manual and automatic translation profile selection modes require validation of the selected translation profile. Figure 104 shows a flowchart of the translation profile validation process.

Part of the translation profile validation process requires verification of the validity of the reference source associated with the translation profile. Because the reference source associated with a translation profile can be any of the following five different source types (per Table 75), determining what constitutes a valid reference depends on the source type:

- REFA, REFAA, REFB, or REFBB reference input
- Auxiliary REF0, auxiliary REF1, auxiliary REF2, or Auxiliary REF3
- IUTS0 or IUTS1
- Auxiliary NCO 0 or NCO 1
- DPLL0/DPLL1 feedback path

Figure 104 shows the source dependent validation decisions in the reference source validation portion of the flowchart.

Revertive and Nonrevertive Reference Switching

Revertive reference switching means that if the reference associated with a translation profile (Translation Profile 0, for example) becomes invalid and the DPLL switches to a new translation profile (Translation Profile 1, for example), and the reference associated with Translation Profile 0 becomes valid again, the DPLL switches back (reverts) to Translation Profile 0, even though the reference associated with Translation Profile 1 is valid.

Nonrevertive reference switching means that if the reference associated with a translation profile (Translation Profile 0.0, for example) becomes invalid and the DPLL switches to a new translation profile (Translation Profile 0.1, for example), the DPLL continues to use Translation Profile 0.1 even if the reference associated with Translation Profile 0.0 becomes valid again. See the Automatic, Priority-Based Translation Profile Selection section on how to assign a priority value to a translation profile.

The automatic, priority-based translation profile selection process provides for revertive and nonrevertive reference switching using the difference between the priority of the current translation profile and the priority of the previous switched from translation profile to make the revertive or nonrevertive decision. Specifically, if the priority value of the current translation profile is at least 8 more than the priority value of the switched from translation profile, the current translation profile has significantly lower priority than the switched from translation profile. Therefore, the DPLL reverts to the switched from translation profile as soon as its reference source becomes valid. Alternatively, if the priority value of the current translation profile is 0 to 7 more than the priority value of the switched from translation profile, the current translation profile has only marginally lower priority than the switched from translation profile. Therefore, the DPLL does not revert, but remains with the current translation profile until its reference source becomes invalid (or the user selects a new translation profile manually).

In summary, the priority separation between the current translation profile and the previously switched from translation profile determines whether reference switching is revertive or nonrevertive. Specifically, revertive switching requires the priority value of the switched from translation profile to be at least 8 lower than the priority value of the current translation profile. Otherwise, nonrevertive reference switching applies. That is, a priority separation of 8 marks the boundary between revertive and nonrevertive reference switching.

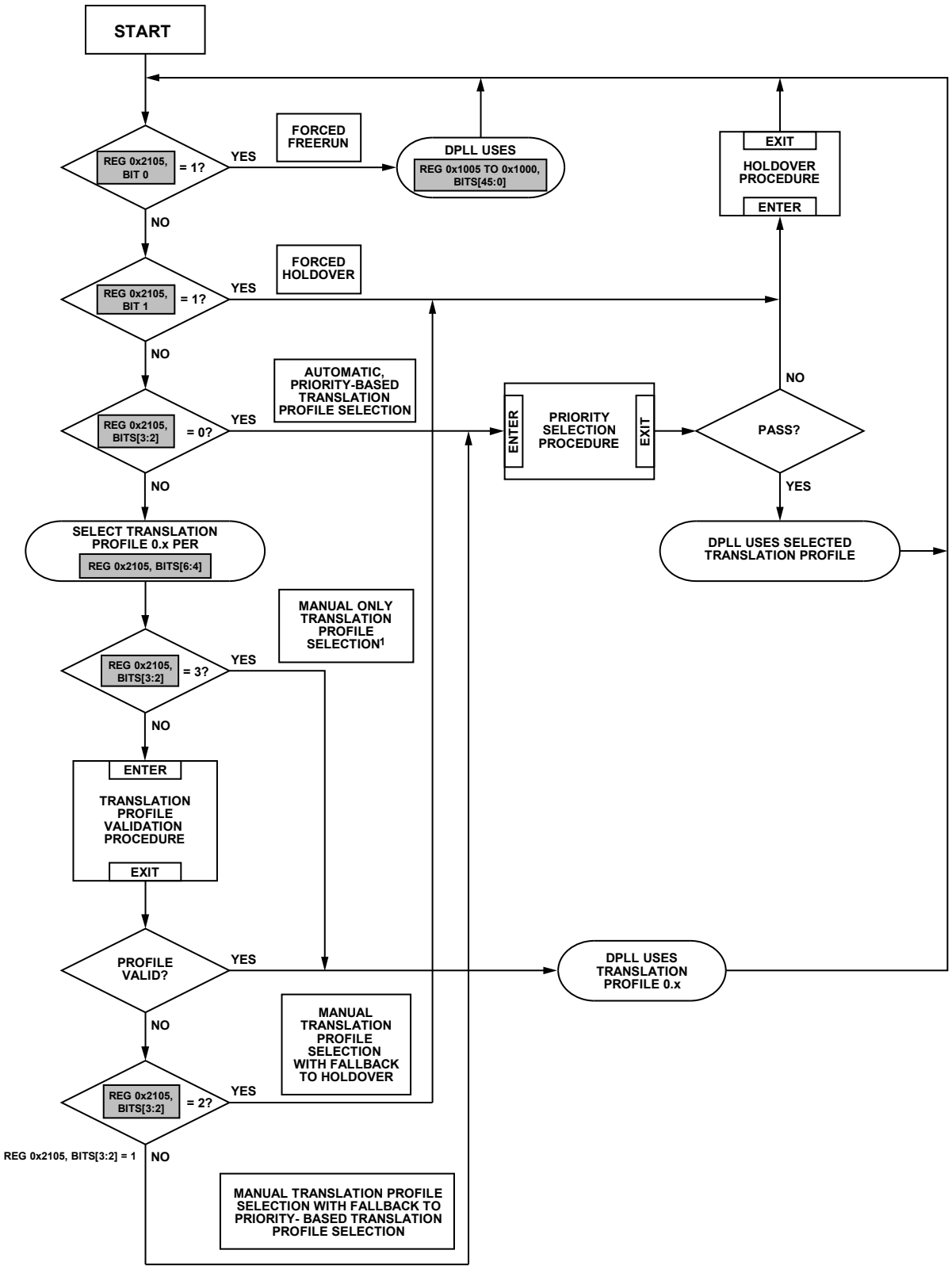
Active Profile

When the DPLL selects a specific translation profile, whether manually or automatically, the selected profile becomes active. When a translation profile is active, the corresponding bit is Logic 1 in Register 0x3009 (DPLL0) or Register 0x300A (DPLL1). Bits[5:0] associate with Translation Profile x.5 through Translation Profile x.0, respectively, where x = 0 or 1.

The IRQ status section of the register map indicates Logic 0 to Logic 1 transitions of Bits[5:0] of Register 0x3013 (DPLL0) and Register 0x3018 (DPLL1). Because the IRQ bits are latched bits, the user must clear them via Bits[5:0] of Register 0x200E (DPLL0) and Register 0x2013 (DPLL1) to obtain visibility of subsequent Logic 0 to Logic 1 transitions of Bits[5:0] in Register 0x3009 (DPLL0) and Register 0x300A (DPLL1) (see the Interrupt Request (IRQ) section).

An active translation profile is not the same as a valid translation profile. That is, a profile can be valid without being active.

When all translation profiles for a particular DPLL are inactive, the user can find the most recently active profile via Bits[6:4] of Register 0x3101 (DPLL0) and Register 0x3201 (DPLL1). The value of Bits[6:4] corresponds to the most recently active profile. If a translation profile is currently active (as indicated by Bits[5:0] in Register 0x3009 and Register 0x300A), the value of Bits[6:4] in Register 0x3101 (DPLL0) and Register 0x3201 (DPLL1) identify the currently active profile.



[1]FOR DEBUG ONLY. THE DPLL USES TRANSLATION PROFILE 0.x WITHOUT VALIDATING THE SOURCE. THEREFORE, THE DPLL IS NOT GUARANTEED TO FUNCTION NORMALLY. CONTACT ANALOG DEVICES FOR DETAILS.

NOTES
1. A RANGE OF BITS USES A COLON SEPARATOR
2. REGISTER ADDRESSES ARE SPECIFIC TO DPLL0

23266-102

Figure 102. DPLL Mode Selection Flowchart

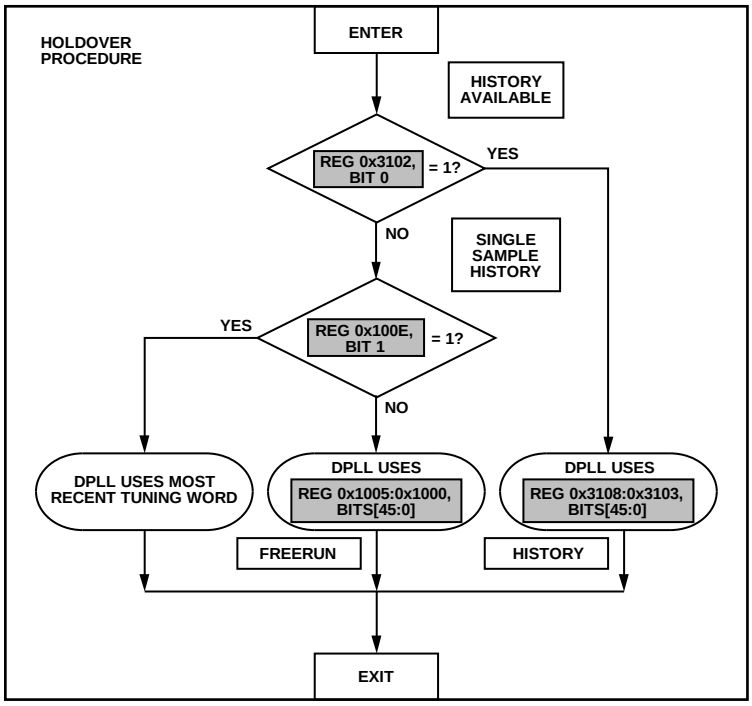
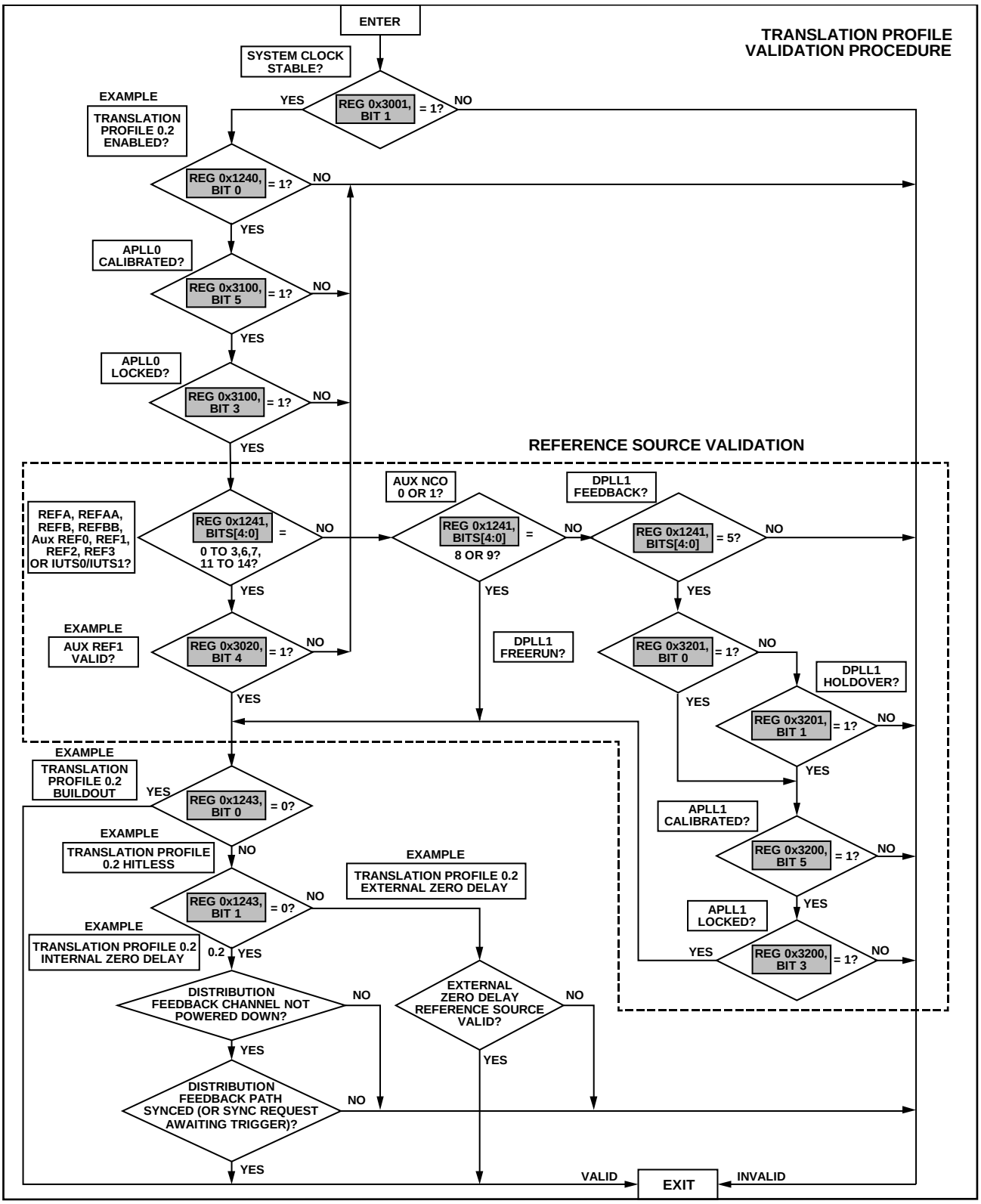


Figure 103. Holdover Flowchart



NOTES
1. A RANGE OF ADDRESSES OR BITS USES A COLON SEPARATOR
2. REGISTER ADDRESSES ARE SPECIFIC TO DPLL0, TRANSLATION PROFILE 0.2 (EXAMPLE), AND AUXILIARY REF1 (EXAMPLE)

23266-104

Figure 104. Flowchart for Translation Profile Validation

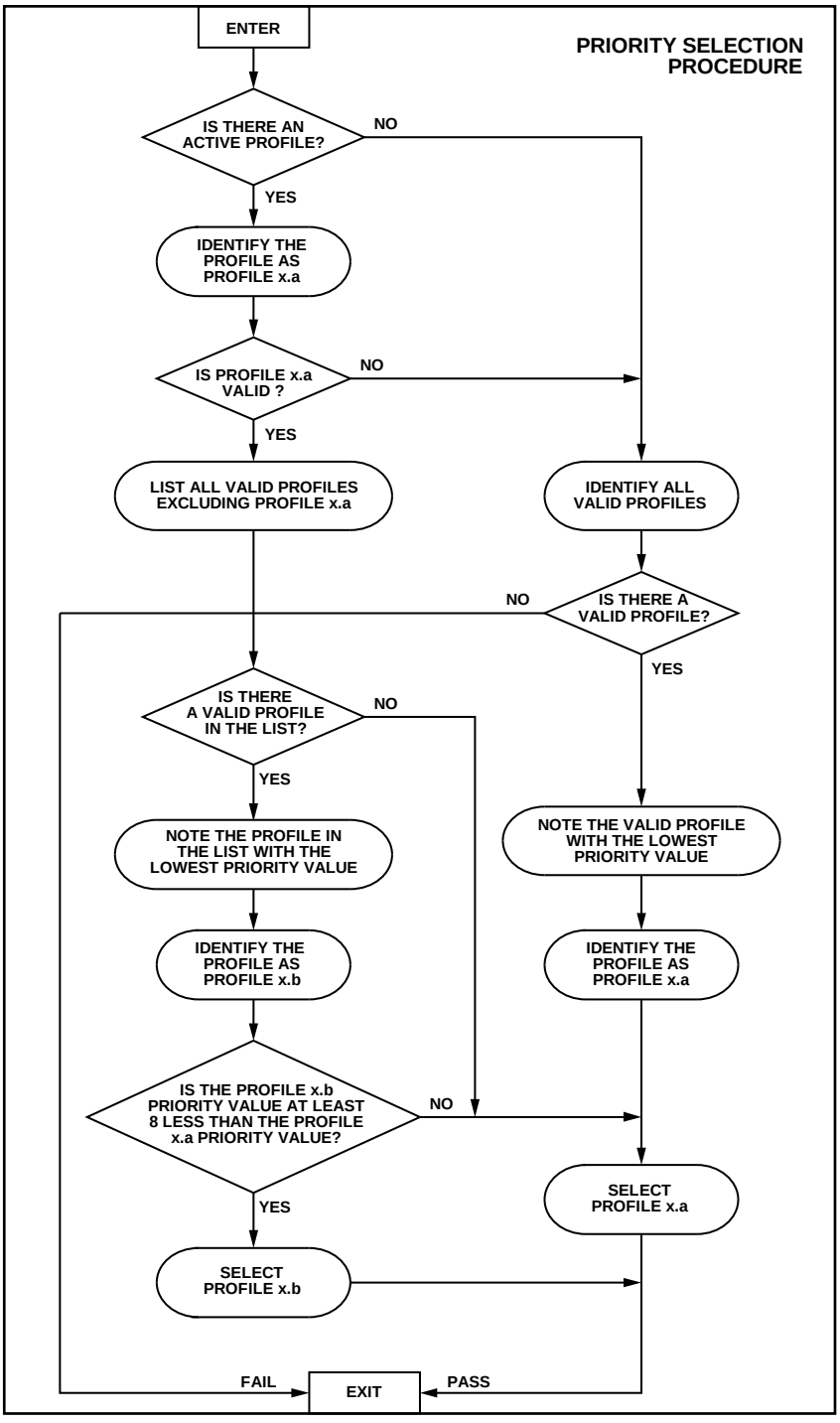


Figure 105. Priority-Based Translation Profile Selection Flowchart

TIME TO DIGITAL CONVERTER (TDC)

TDCs are at the core of the DPLL technology in the AD9546. The purpose of a TDC is to continuously observe a clock signal, note the occurrence of an edge transition, and generate a digital time stamp identifying when the edge occurred.

Because a TDC effectively digitizes rising edge clock events by means of time stamps, it enables digital (that is, numeric) processing of clock signals. In a conventional PLL, for example, a PFD circuit measures the relative phase of the reference and feedback signals and provides an analog representation of the phase difference. Instead of an analog representation of the phase difference, the AD9546 uses a pair of TDCs (one in the reference path and one in the feedback path) to generate reference and feedback time stamps, constituting a numeric representation of the phase difference. With phase information in numeric form, the AD9546 implements the entire PLL function digitally by employing a numeric PFD, digital loop filter and an NCO.

Figure 106 shows a simplified diagram of a TDC. The TDC has a time base clock (which gives the TDC its sense of time), an input clock (the signal in question), and an output consisting of numeric time stamps.

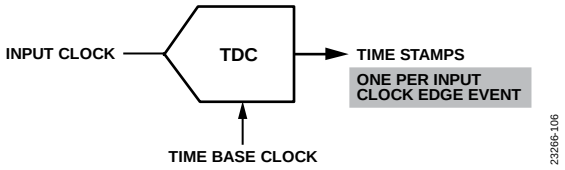


Figure 106. TDC Diagram

The AD9546 contains 10 TDCs.

- Four at the output of the reference dividers associated with the four reference inputs
- Four at the output of the reference dividers associated with the auxiliary reference inputs
- Two associated with the two DPLLs at the output of the feedback divider

Each TDC produces digital time stamps identifying each rising edge of its input clock. That is, the TDCs generate time stamps at a rate commensurate with their respective input clock period. The user must, however, ensure the TDC input clock rate is within the specified range of 1 Hz to approximately 200 kHz.

The TDC time stamps derive from an internal time base that originates with the device system clock. As such, the TDCs benefit from the ability of the AD9546 to provide system clock compensation, assuming the user activates system clock compensation via Register 0x0280 to Register 0x0282 (see the System Clock Compensation section).

TIME STAMPS

TIME STAMPS OVERVIEW

The AD9546 has 14 time stamp sources.

- REFA TDC
- REFAA TDC
- REFB TDC
- REFBB TDC
- DPLL0 feedback TDC
- DPLL1 feedback TDC
- Auxiliary REF0 TDC
- Auxiliary REF1 TDC
- Auxiliary REF2 TDC
- Auxiliary REF3 TDC
- Auxiliary NCO 0
- Auxiliary NCO 1
- IUTS 0
- IUTS 1

Most of the time stamp sources in the AD9546 are TDCs. The Auxiliary NCOs and IUTSs also generate time stamps (see the Auxiliary NCOs and Inverse User Time Stamper (IUTS) sections), but without using a TDC.

In general, the time stamp sources require advance knowledge of the period of the applied input signal. For example, the REFA TDC requires the user to enter the nominal period of the REFA input signal. The user programs the nominal input period via the register map.

Time stamps produced by the time stamp sources are not readily accessible by the user. Instead, the user accesses time stamps from the various time stamp sources by means of a user time stamp processor (see the User Time Stamp Processor (UTSP) section and User Time Stamper (UTS) section).

DIGITAL CROSSPOINT MUX

The digital cross point mux is an interconnect matrix allowing an array of time stamp destinations access to the various time stamp sources within the AD9546. However, not every time stamp source can connect to every time stamp destination. Table 92 summarizes the interconnections of the various time stamp sources to potential destinations, where a yes indicates a valid connection and a blank cell means invalid connection.

Table 92. Time Stamp Source to Destination Matrix

	Time Stamp Destination					
Time Stamp Source	**DPLL0 Reference Input**	**DPLL1 Reference Input**	**SYSCLK Compensation PLL Reference Input**	**Time Skew Measurement Input**	**User Time Stamp Processor Input**	**User Time Stamper (UTSx) Input**
REFA TDC	Yes	Yes	Yes	Yes	Yes	Yes
REFAA TDC	Yes	Yes	Yes	Yes	Yes	Yes
REFB TDC	Yes	Yes	Yes	Yes	Yes	Yes
REFBB TDC	Yes	Yes	Yes	Yes	Yes	Yes
DPLL0 Feedback TDC		Yes		Yes	Yes	Yes
DPLL1 Feedback TDC	Yes			Yes	Yes	Yes
Auxiliary REF0 TDC	Yes	Yes	Yes	Yes	Yes	Yes
Auxiliary REF1 TDC	Yes	Yes	Yes	Yes	Yes	Yes
Auxiliary REF2 TDC	Yes	Yes	Yes	Yes	Yes	Yes
Auxiliary REF3 TDC	Yes	Yes	Yes	Yes	Yes	Yes
Auxiliary NCO 0	Yes	Yes		Yes	Yes	Yes
Auxiliary NCO 1	Yes	Yes		Yes	Yes	Yes
IUTS 0	Yes	Yes		Yes	Yes	Yes
IUTS 1	Yes	Yes		Yes	Yes	Yes

TAGGED TIME STAMPS

A tagged time stamp is a time stamp with a unique marker, or tag, attached to it. Tagged time stamps allow the AD9546 to use a lower frequency clock embedded within a higher frequency clock. For example, by tagging every 10th edge of a clock signal, there is a carrier signal (that is, the original clock signal) and a divide by 10 version of the carrier embedded within the signal resulting from the tagged time stamps.

The various time stamp sources within the AD9546 can generate tagged time stamps (the auxiliary NCOs, for example). Likewise, certain time stamp destinations within the AD9546 are capable of processing only tagged time stamps while ignoring untagged time stamps (the user time stamp processors, for example, described in the User Time Stamp Processor (UTSP) section). However, the two DPLLs are the most important of the time stamp destinations capable of using tagged time stamps (see the Time Stamp Tagging Options section).

Tagged Auxiliary NCO Time Stamps

The auxiliary NCOs generate time stamps coincident with the expiration of each period defined by the programmed auxiliary NCO frequency (that is, at the rollover point of the phase accumulator). For example, an NCO frequency of 100 Hz results in the generation of time stamps at a 100 Hz rate with each time stamp 10 ms greater than the previous one.

The time stamp generator gives the user the option of tagging auxiliary NCO time stamps via Bits[15:0] in Register 0x280B to Register 0x280C (for Auxiliary NCO 0) and Register 0x284B to Register 0x284C (for Auxiliary NCO 1).

The value, N, represented by Bits[15:0] (tag ratio) results in every N^{th} time stamp being tagged. For example, Bits[15:0] = 1 decimal means every other time stamp is tagged. Similarly, a decimal value of 2 means every third time stamp is tagged. Likewise, a decimal value of 3 means every fourth time stamp is tagged. This pattern continues to 65,535. A decimal value of 0 (default) disables time stamp tagging (no tags).

The user can also shift the time position of the tagged flags Bits[15:0] in Register 0x280D to Register 0x280E (for Auxiliary NCO 0) and Register 0x284D to Register 0x284E (for Auxiliary NCO 1). Bits[15:0] are an autoclearing bit field. The value, K, represented by Bits[15:0], is a signed integer (twos complement) indicating a number of fundamental periods of the auxiliary NCO. Thus, for K = −3, the time stamp processor tags time stamps three time stamps earlier than the N^{th} time stamp locations indicated by the tag ratio described previously. K is effectively a phase shift of the tagged time stamp period, where K < N.

Note that when the auxiliary NCO is the reference to the DPLL for an active translation profile (see the Translation Profiles section), shifting the time position of a tag is not valid and must be avoided. When using the tag shift feature, be sure that any translation profiles using the auxiliary NCO as a DPLL reference are not enabled, or if they are enabled, that the associated DPLL is in freerun or holdover mode.

Tagged Reference Time Stamps

Tags associated with time stamps originating from the reference TDCs are the result of demodulating an embedded clock (see the Reference Demodulator section). Synchronization events associated with demodulation generate tagged time stamps. The user can program the DPLL to use the tagged time stamps from a reference TDC.

Tagged Feedback Time Stamps

Tagged time stamps originating from the DPLL feedback path are the result of synchronization events associated with the modulation controller (see the Modulation section). The user can program the DPLL to use the tagged time stamps from a feedback TDC.

USER TIME STAMP PROCESSOR (UTSP)

UTSP OVERVIEW

The user can view time stamps from the various time stamp sources (see the Time Stamps section) via the UTSPs. The AD9546 has two UTSPs with their respective output results available via the register map, as shown in Figure 107. The UTSPs convert time stamps originating from a given time stamp source (see Table 92) to a time scale meaningful to the user. The time scale conversion is necessary because the time stamp sources use an arbitrary time scale internal to the device, which is meaningless to the user.

The user selects the desired time scale source for a UTSP via Bits[7:6] of Register 0x2A12 (for UTSP 0) and Register 0x2A13 (for UTSP 1). The value of Bits[7:6] selects the desired time scale per Table 93. Note that either one of the UTSPs can use either one of the auxiliary NCO time scales.

Table 93. UTSP Time Scale Source

Bits[7:6] (Decimal)	Description
0	Auxiliary NCO 0 time scale (default)
1	Common time scale
2	Auxiliary NCO 1 time scale
3	Common time scale

To attach a particular time stamp source to a UTSP, use Bits[4:0] (unsigned) of Register 0x2A12 (for UTSP 0) and Register 0x2A13 (for UTSP 1). The value of Bits[4:0] select the desired time stamp source per Table 94.

Table 94. UTSP Input Time Stamp Source

Bits[4:0] (Decimal)	Description
0	REFA TDC
1	REFAA TDC
2	REFB TDC
3	REFBB TDC
4	DPLL0 feedback TDC
5	DPLL1 feedback TDC
6	Auxiliary REF0 TDC (default for UTSP 0)
7	Auxiliary REF1 TDC (default for UTSP 1)
8	Auxiliary NCO 0 time scale
9	Auxiliary NCO 1 time scale
10	Auxiliary REF0 TDC
11	Auxiliary REF2 TDC
12	Auxiliary REF3 TDC
13	IUTS 0
14	IUTS 1
15 to 31	Not applicable

READING USER TIME STAMPS

The converted input time stamps are available at the output of the UTSP via Bits[79:0] in Register 0x3A14 to Register 0x3A1D (for UTSP 0) and Register 0x3A20 to Register 0x3A29 (for UTSP 1). However, the user must assert an IO update operation (see the IO Update section) prior to reading Bits[79:0]. The UTSPs perform time stamp conversions as the time stamps arrive from their prescribed time stamp sources. As such, the UTSPs are event driven. Because of this event driven process, the user cannot be certain that the data residing in Bits[79:0] is indeed relevant. There are two methods for mitigating this problem.

The first method is to read Bits[79:0] periodically. However, to avoid missing time stamps, the user must read Bits[79:0] at a rate greater than the expected rate of the time stamp source assigned to the input of the UTSP. Because consecutive time stamps arriving from a time source cannot have the same value (by definition), it guarantees that a time stamp reading different from the last constitutes a new time stamp.

The second method is to use the IRQ feature set of the AD9546 (see the Interrupt Request (IRQ) section). Whenever a UTSP completes a time stamp conversion, the corresponding status Bit 2 (for UTSP 0) or Bit 3 (for UTSP 1) of Register 0x300F (assuming the status bit is unmasked) is set. Upon the status bit becoming Logic 1, assert an IO update operation, and read Bits[79:0] (see Figure 107).

The event driven nature of the UTSP means that the user may miss one or more user time stamps if multiple time stamps occurred before the user reads Bits[79:0]. Bits[7:0] in Register 0x3A1E (for UTSP 0) and Register 0x3A2A (for UTSP 1) provide a way for the user to check for any missed input time stamps. When Bits[7:0] indicate a value of zero, then the current user time stamp is the latest one in a contiguous series. When Bits[7:0] indicate a nonzero value, then the user missed the indicated number of time stamps. Therefore, the current time stamp constitutes the first one in a new series of time stamps. Missed time stamps are lost and are not recoverable.

In operation, the user must assert an IO update operation, which refreshes Bits[79:0] and Bits[7:0] with the most recent values. After asserting an IO update operation, the recommendation is to read Bits[79:0] immediately followed by reading Bits[7:0]. Doing so enables immediate assessment of the value of Bits[7:0] as an integral part of capturing the time stamp value. Bits[7:0] clear automatically when read.

Because the user specifies the expected input period for a TDC via the register map, the TDC knows approximately how much time must elapse between time stamp events. If an event happens far outside the expected event period, the TDC flags the time stamp associated with that event as an error via Bit 0 of Register 0x3A1F (for UTSP 0) and Register 0x3A2B (for UTSP 1). In general, the user must discard a time stamp when Bit 0 is Logic 1.

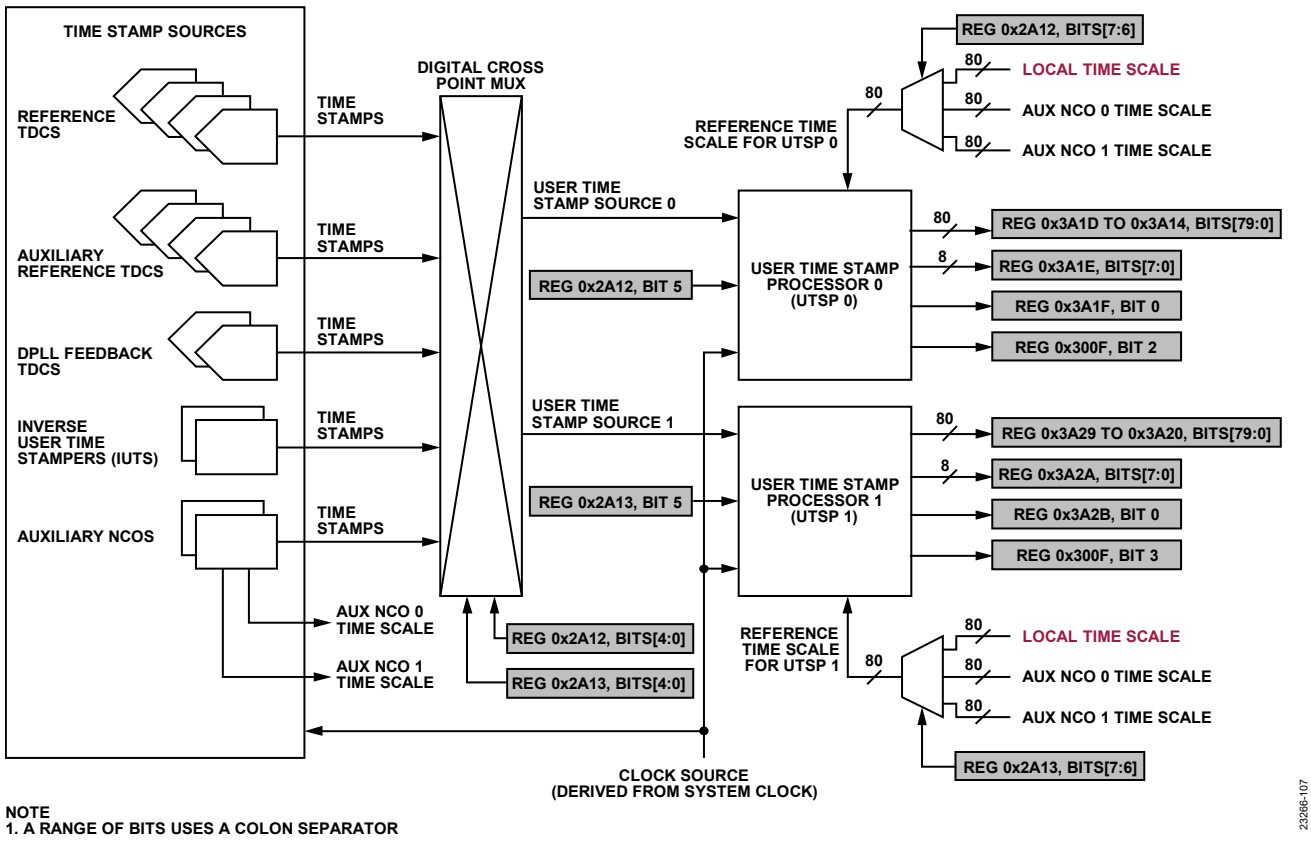


Figure 107. User Time Stamp Processors

INTERPRETING USER TIME STAMPS

The outputs of the UTSPs comprise 80-bit words with the least significant bit weighted as 2^{-40}. The units associated with a time stamp converted by a UTSP depends on the selected time scale. Specifically, the 80-bit user time stamp consists of a 40-bit integer part (the 40 leftmost bits) and a 40-bit fractional part (the 40 rightmost bits), as shown in Figure 108. Each count of the integer part represents one period of the selected time scale. The 40 fractional bits represent the fractional part of one period of the selected time scale with a resolution of 2^{-40}.

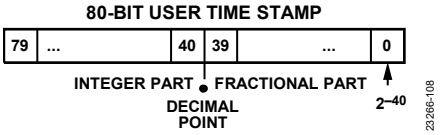


Figure 108. User Time Stamp

For example, assume the 80-bit word is the hexadecimal number, 0x 0000 0000 B10E 04F0 07C1, in which case the decimal equivalent is 194,673,770,497,985.

Because the scale of the fractional part is 2^{-40}, the time stamp value (rounded to 10 decimal places) is

$$\textit{Time Stamp Value} = 194{,}673{,}770{,}497{,}985 \times 2^{-40}$$

$$= 177.054762842076343076769262552 26$$

$$= 177.0547628421$$

This time stamp value represents periods of the selected time scale.

For example, suppose the selected time scale is auxiliary NCO 0, which the user had programmed with a frequency of 10052.630025 Hz (corresponding to a period of 99.476455167760936272992897696939 µs). Thus, the time associated with the example time stamp (177.0547628421) is the time stamp value multiplied by the period of selected time scale (99.476455167760936272992897696939 µs), which is 0.017612780178 sec (rounded to the nearest ps). That is, the time stamp in this example indicates a time of approximately 17.6 ms.

Time stamps have a meaningful resolution of approximately 1 ps. Thus, digits in a time stamp calculation representing less than 1 ps must be ignored.

In a digitized clocking application, the AD9546 produces a local time scale. Because the local time scale advances at a rate commensurate with external time keeping (rather than based on periods of a programmed frequency as with the auxiliary NCOs), the local time scale is better suited for relating time stamps to external time.

TAGGED TIME STAMPS

The UTSPs can operate on tagged time stamps from the various time stamp sources shown in Figure 107. When the selected time stamp source is generating tagged time stamps, the user can program the UTSP to only pay attention to the tagged time stamps, while ignoring untagged time stamps. To enable tagged time stamp processing, program Bit 5 of Register 0x2A12 (for UTSP 0) and Register 0x2A13 (for UTSP 1) to Logic 1.

UTSP WITH SYSTEM CLOCK COMPENSATION

The AD9546 allows the user to apply system clock compensation (see the System Clock Compensation section) to the time stamp sources as well as to the auxiliary NCOs. This poses a potential problem when the user selects an auxiliary NCO as a time scale for a UTSP. Specifically, when the user elects to apply system clock compensation to a time stamp source and that time stamp source is the input to a UTSP, then the user must apply the same system clock compensation to the auxiliary NCO selected as the time scale for that UTSP. Otherwise, the time stamps that are output by the UTSP are in error.

TIMING SKEW MEASUREMENTS USING TWO TDCs

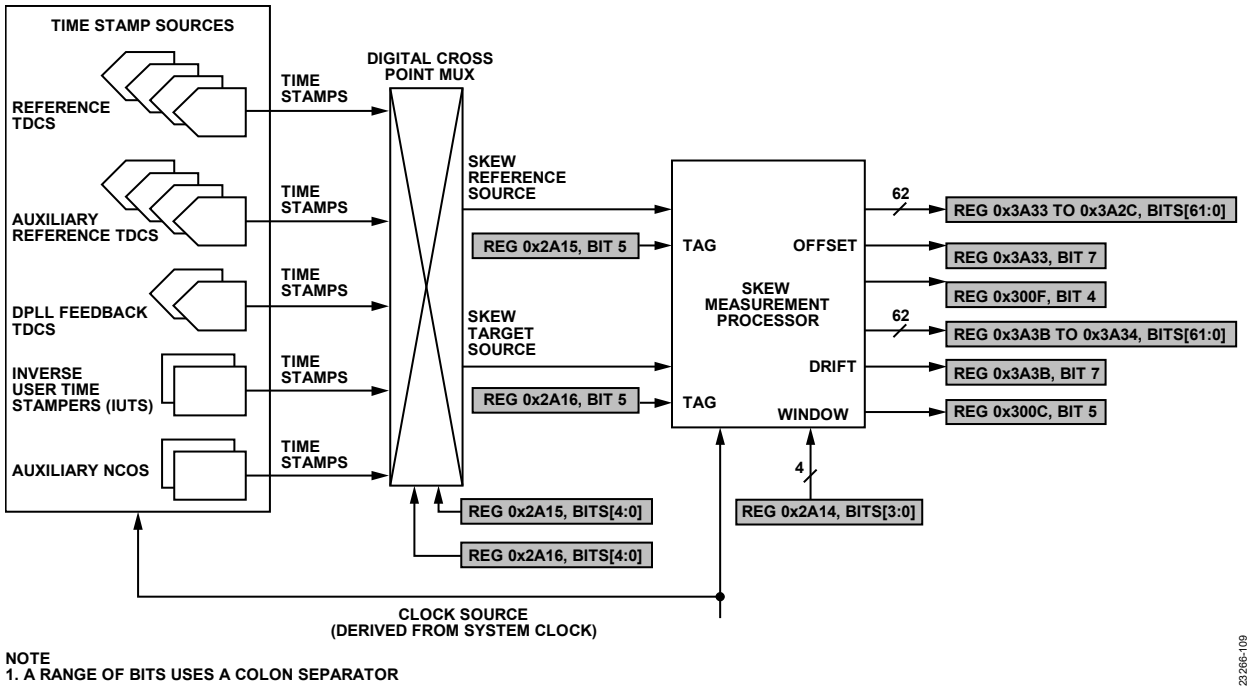


Figure 109. Skew Measurement Block Diagram

The UTSP (see the User Time Stamp Processor (UTSP) section) allows the user to read time stamps originating from various time stamp sources—the time stamps indicating the occurrence of a rising edge at the signal source. Sometimes, however, it is desirable to know the relative phase offset between the rising edges of two different signal sources rather than the corresponding time associated with each edge. To this end, the AD9546 provides the ability to measure and report the time difference (timing skew) between two time stamps (usually, but not exclusively, from different sources). The time difference between time stamps relates directly to phase offset.

Time skew (phase offset) only has meaning when two clock signals operate at the same frequency. Otherwise, the phase offset is not constant but drifts over time. Figure 109 shows a block diagram of the skew measurement processor.

Time skew measurements involve a reference time stamp source and a target time stamp source. The user assigns the reference time stamp source via Bits[4:0] of Register 0x2A15 and the target time stamp source via Bits[4:0] of Register 0x2A16. The decimal value of Bits[4:0] selects the time stamp source per Table 94, except that the default selection is REFA for both the reference time stamp source and target time stamp source.

Unlike the results from the UTSP, which constitute individual measurements, the offset and skew measurements from the skew measurement processor constitute an averaged group of sequential measurements. The user selects the number, N, of samples averaged per measurement via Bits[3:0] of Register 0x2A14 according to Table 95.

Table 95. Skew Measurement Window Settings

Bits[3:0]	N for Skew Offset	N for Skew Drift
0	2	2
1	4	4
2	8	8
3	16	16
4	32	16
5	64	16
6	128	16
7	256	16
8	512	16
9	1024	16
10	2048	16
11	4096	16
12	8192	16
13	16384	16
14	32768	16

The selected skew window size bit field value sets the number of averaging samples for both skew offset and skew drift measurements. However, a given selection may yield a different number of averaging samples for offset measurements vs. drift measurements, as shown in Table 95.

The skew offset and drift measurements constitute a low-pass filtered sequence of N samples. That is, the skew measurement processor smooths out (that is, averages) the measurement variation over N samples. After an N sample average is established, subsequent measurements include the next sample and the preceding N − 1 samples.

As such, the processor requires a minimum of N samples to compute the desired average, after which the processor continues to update the average one sample at a time. The skew measurement processor provides the user with an indication that it has captured the first block of N samples by setting Bit 7 of Register 0x3A33 (for an offset measurement) or Register 0x3A3B (for a drift measurement).

Bit 7 is Logic 0 for the first N − 1 measurements in a sequence and Logic 1 for the N^{th} measurement. As such, when the user reads the result of a skew offset or skew drift measurement, the value is a fully averaged result if Bit 7 (associated with the measurement type: offset or drift) is Logic 1. When Bit 7 is Logic 0, offset or drift readings captured exhibit incomplete averaging but are still useable. However, the variance of the results is greater than for a fully averaged measurement.

Unlike the results from the UTSPs, the skew measurement processor results are in absolute units of picoseconds. To access the time skew measurement processor results, use the skew offset and skew drift bit fields (Register 0x3A2C to Register 0x3A33 and Register 0x3A34 to Register 0x3A3B, respectively).

The result of a skew offset measurement appears in Bits[61:0] (signed, twos complement) of Register 0x3A2C to Register 0x3A33 and notification that a measurement is ready appears as an IRQ in Bit 4 of Register 0x300F. The decimal value of Bits[61:0] constitute time in units of 2^{-16} ps (a range of approximately ±35 sec). Skew offset measurements assume the reference and target sources are of nearly the same frequency. That is, skew is meaningless for two sources with differing frequency, because the skew constantly drifts in such a case. The skew measurement processor measures skew relative to the assigned skew reference source (the processor computes a coarse average of the reference period). The skew measurement processor determines the polarity of the skew offset measurement by considering a time window spanning ±½ unit interval (UI), where 1 UI is the measured period of the reference source. A negative value means time stamps from the target source lead time stamps from the reference source by as much as ½ UI, whereas a positive value means time stamps from the target source lag time stamps from the reference source by as much as ½ UI. Note that jitter can affect the polarity of the skew measurement result.

For example, consider a 62-bit skew offset reading of 0x 3FFF FFF8 A673 24B5 (hexadecimal), which is −31,567,174,475 decimal. To find the skew offset time, scale the decimal value by 2^{-16} ps, which results in −481,676.8566131591796875 ps. The negative sign implies the target source leads the reference source by approximately 482 ns.

The result of a skew drift measurement appears in Bits[61:0] (signed, twos complement) of Register 0x3A34 to Register 0x3A3B. The decimal value of Bits[61:0] has units of 2^{-16} picoseconds per unit interval (ps/UI). That is, the value is relative to the measured period of the reference source. Skew drift is a measure of the rate at which the skew offset varies cycle by cycle.

For example, consider two signals with a nominal frequency of 100 Hz. A skew drift reading of 0x 3FFF FFF8 A673 24B5 (hexadecimal) is −31,567,174,475 decimal. To find the skew drift, scale the decimal value by 2^{-16}, which results in −481,676.8566131591796875 ps/UI. Because the reference period is 10 ms (1/100 Hz), the result indicates the target source period is decreasing 482 ns every 10 ms, implying a frequency offset of −48.2 ppm (−482 ns/10 ms = -48.2×10^{-6}).

Two signals with the same frequency have no drift and must yield a drift measurement of zero. However, a frequency difference that is too great can exceed the limits of the skew processor, which is $1/16^{th}$ UI. Upon reaching this limit, the skew measurement processor sets the status Bit 5 of Register 0x300C to Logic 1.

TAGGED SKEW MEASUREMENT TIME STAMPS

The skew measurement processor can operate on tagged time stamps from the various time stamp sources shown in Figure 109. When the selected skew reference and/or skew measurement time stamp source is generating tagged time stamps, the user can program the skew measurement processor to only pay attention to the tagged time stamps, while ignoring untagged time stamps. To enable tagged skew reference time stamp processing, program Bit 5 of Register 0x2A15 to Logic 1. To enable tagged skew target time stamp processing program Bit 5 of Register 0x2A16 to Logic 1.

AUXILIARY NCOS

AUXILIARY NCO OVERVIEW

The AD9546 has two auxiliary NCOs as shown in Figure 111. The NCOs generate time stamps and/or physical output signals based on a very precise programmable frequency specified by the user. The auxiliary NCOs also provide a time scale to the UTSPs (see the User Time Stamp Processor (UTSP) section). The NCOs have frequency tuning resolution of 2^{-40} Hz, or approximately 1 pHz.

AUXILIARY NCO FREQUENCY

The user programs the output frequency of the auxiliary NCOs (f_{AUX}) by assigning a center frequency and an offset frequency. By default, the offset frequency value is zero, which means the default output frequency is the center frequency. Figure 110 details the relationship between the center frequency (56 bits) and offset frequency (32 bits) register values.

The auxiliary NCO is effectively a direct digital synthesizer (DDS) with a phase accumulator. The DDS accumulates a given phase value, θ, at a prescribed rate, f_{ACCUM}.

$$f_{ACCUM} = f_S/96$$

where f_S is the system clock frequency (see the System Clock PLL section). The auxiliary NCO calculates θ such that the accumulator rolls over with a period of precisely $1/f_{AUX}$.

Auxiliary NCO Center Frequency

Center frequency assignment is via Bits[55:0] (unsigned) of Register 0x2800 to Register 0x2806 (for auxiliary NCO 0) and Register 0x2840 to Register 0x2846 (for auxiliary NCO 1). Bits[55:0] carry units of 2−40 Hz. Thus, the center frequency value of Bits[55:0] relates to the center frequency, f_{CENTER}, as

$$f_{CENTER} = Center\ Frequency\ Value \times 2^{-40}$$

For example, given a desired center frequency of 10 kHz, find the required 56-bit register value.

$$Center\ Frequency\ Value = f_{CENTER} \times 2^{40}$$
$$= 10^4 \times 2^{40}$$
$$= 10{,}995{,}116{,}277{,}760{,}000$$
$$= 0x\ 27\ 1000\ 0000\ 0000\ (hexadecimal)$$

The center frequency upper limit is 65,536 Hz ($2^{56} \times 2^{-40}$ Hz).

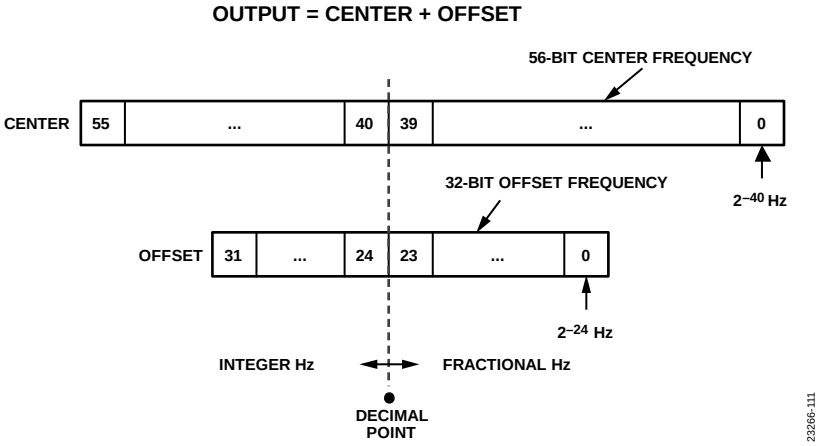


Figure 110. Auxiliary NCO Frequency Tuning

Auxiliary NCO Offset Frequency

Offset frequency assignment is via Bits[31:0] (unsigned) of Register 0x2807 to Register 0x280A (for auxiliary NCO 0) and Register 0x2847 to Register 0x284A (for auxiliary NCO 1). Bits[31:0] carry units of 2^{-24} Hz. The offset frequency value of Bits[31:0] relates to the offset frequency, f_{OFFSET}, as

$$f_{OFFSET} = Offset\ Frequency\ Value \times 2^{-24}$$

For example, given a desired offset frequency of 100 Hz, find the required 32-bit register value.

$$Offset\ Frequency\ Value = f_{OFFSET} \times 2^{24}$$
$$= 10^2 \times 2^{24}$$
$$= 1{,}677{,}721{,}600$$
$$= 0x\ 6400\ 0000\ (hexadecimal)$$

The offset frequency upper limit is 256 Hz ($2^{32} \times 2^{-24}$ Hz).

The main purpose for the offset frequency is to give an external processor or controller that is limited to 32-bit processing capability the ability of rapidly changing the output frequency. For example, it is cumbersome for a 32-bit machine to compute the 56-bit center frequency and then deliver the computation to the register map.

The 32-bit offset frequency option alleviates this problem because the user can initially program an appropriate 56-bit center frequency and then use 32-bit offset values to change the output frequency. The 32-bit offset value allows a 32-bit machine to operate in its native 32-bit environment and facilitates rapid changing of the output frequency. The offset values offer less precise tuning than the center frequency (2^{-24} Hz vs. 2^{-40} Hz resolution).

Depending on the tuning range for a given application, the user can use a combination of the center and offset values to minimize the number of 8-bit registers required to cover the range, yet still yield 2^{-40} Hz frequency resolution. For example, the user can select a center and offset value that allows the offset to remain fixed over the tuning range yet constrains tuning changes of the center value to the lower 8, 16, 24, or 32 bits of the center frequency value. Using only the LSBs of the center frequency allows 32-bit (or less) register operations while still maintaining 2^{-40} Hz frequency tuning resolution.

AUXILIARY NCO PHASE OFFSET

The user can adjust the phase of the auxiliary NCO via Bits[39:0] (signed, twos complement) in Register 0x2814 to Register 0x2818 (for auxiliary NCO 0) and Register 0x2854 to Register 0x2858 (for auxiliary NCO 1). The specific functionality of the phase adjustment depends on the value of Bit 0 of Register 0x280F (for auxiliary NCO 0) and Register 0x284F (for auxiliary NCO 1), as explained in the Absolute Phase Offset section and the Relative Phase Offset section.

When the user programs Bits[39:0] and asserts the IO update bit, the AD9546 automatically imposes a specified upper limit on how fast the phase can change (phase slew). See the Auxiliary NCO Phase Slew Limit section for details.

Absolute Phase Offset

Absolute phase offset adjustment is in effect when Bit 0 = 0 of Register 0x280F (for auxiliary NCO 0) and Register 0x284F (for auxiliary NCO 1). For absolute phase offset adjustments, the phase offset bits, Bits[39:0], constitute a signed absolute time offset with 1 ps resolution. The signed absolute time offset allows an offset range of approximately ±0.55 sec ($\pm 10^{-12}$ sec $\times 2^{39}$). However, the user must limit programmed offset value to ±1 UI (that is, plus or minus one period of the auxiliary NCO).

Note that programming a phase offset value in excess of the ±1 UI limit causes the device to set the status in Bit 5 or Bit 7 in Register 0x3002 corresponding to auxiliary NCO 0 or auxiliary NCO 1, respectively.

For example, given a programmed auxiliary NCO 0 frequency of 10 kHz (per the center frequency and offset frequency settings) and a desired phase offset of −15.5°, determine the necessary value of Bits[39:0] in Register 0x2814 to Register 0x2818, the absolute time offset.

First, ensure the phase offset is less than ±360° to satisfy the ±1 UI maximum phase offset requirement. The desired phase offset of −15.5° is within the ±1 UI limit.

Next, determine the time, t, associated with a −15.5° phase offset.

$$t = (-15.5°/360°)/(10\ kHz)$$
$$= -4.3055555556\ \mu s\ (to\ 10\ decimal\ places)$$

Finally, convert the value of t to units of ps.

$$Bits[39:0] = t/(10^{-12}\ sec)$$
$$= (-4.3055555556 \times 10^{-6}\ s)/(10^{-12}\ sec)$$
$$= -4{,}305{,}556\ (nearest\ integer)$$
$$= 0x\ FF\ FFBE\ 4D6C\ (40\text{-}bit\ hexadecimal)$$

Relative Phase Offset

Relative phase offset adjustment is in effect when Bit 0 = 1 of Register 0x280F (for auxiliary NCO 0) and Register 0x284F (for auxiliary NCO 1). For relative phase offset adjustments, the phase offset bits, Bits[39:0], constitute a signed relative offset specified as a fraction of the period of the auxiliary NCO. Thus, Bits[39:0] cover a range from −½ UI to 1 LSB less than +½ UI.

For example, given a programmed auxiliary NCO 0 frequency of 10 kHz (per the center frequency and offset frequency settings) and a desired phase offset of −15.5°, determine the necessary value of Bits[39:0] in Register 0x2814 to Register 0x2818, the relative time offset.

First, the phase offset must be within ±180° to satisfy the ±½ UI maximum offset requirement. The desired phase offset of −15.5° is within the ±½ UI limit.

Next, convert −15.5° to fractional UI.

$$Fractional\ UI = -15.5°/360°$$

$$= -0.0430555556 \text{ (to 10 decimal places)}$$

Because Bits[39:0], the phase offset bits, constitute a signed, twos complement 40-bit number spanning 1 UI, calculate the value of Bits[39:0] as follows:

$$Bits[39:0] = -0.0430555556 \times 2^{40}$$

$$= -47,340,083,974 \text{ (nearest integer)}$$

$$= 0x\ F4\ FA4F\ A4FA \text{ (40-bit hexadecimal)}$$

AUXILIARY NCO PHASE SLEW LIMIT

When the user requests the application of a phase offset on one of the auxiliary NCOs, the AD9546 does not necessarily apply the full phase offset all at once. Instead, Bits[31:0] (unsigned) of Register 0x2810 to Register 0x2813 (for auxiliary NCO 0) and Register 0x2850 to Register 0x2853 (for auxiliary NCO 1) control whether the phase offset is immediate or slew limited.

When the phase slew limit bits, Bits[31:0], have a decimal value of zero, phase offset is immediate. That is, the auxiliary NCO applies the specified phase offset value all at once. However, when Bits[31:0] are nonzero, phase offset is slew limited. That is, the auxiliary NCO gradually introduces phase changes by regulating the rate at which the phase changes occur to prevent exceeding the limit imposed by the value of Bits[31:0]. The auxiliary NCO keeps track of the total phase change and terminates the phase slew limiting process when the total phase change satisfies the value established by the auxiliary NCO phase offset parameters (see the Auxiliary NCO Phase Offset section).

Bits[31:0] carry units of 2^{-36} UI/UI, where UI is the unit interval of the associated auxiliary NCO (1 UI is the reciprocal of the programmed frequency of the auxiliary NCO). That is, the user specifies the maximum rate of change of phase as fractions of a cycle per cycle. Note the user must program Bits[31:0] prior to invoking a phase offset.

For example, assume the programmed frequency of the auxiliary NCO is 60 Hz (therefore, 1 UI = 1/60 sec) and the desired limit for the rate of change of phase is 1 µs/sec (10^{-6} sec/sec). Note that sec/sec and UI/UI are fundamentally equivalent. Therefore, 10^{-6} sec/sec is 10^{-6} UI/UI. Because Bits[31:0] are in units of 2^{-36} UI/UI, the required value for Bits[31:0] is $10^{-6}/2^{-36}$ = 68,719 (rounded to the nearest integer) or 0x 0001 0C6F hexadecimal. Assuming the user programs Bits[31:0] to 68,719 (decimal), when the user invokes the phase offset specified by the auxiliary NCO phase offset parameters (see the Auxiliary NCO Phase Offset section), the AD9546 automatically ensures that the rate of change of phase does not exceed the specified value of 1 µs per second.

When Bits[31:0] are nonzero and while the auxiliary NCO is in the process of limiting the phase slew rate, the device sets a status bit in Register 0x3002: Bit 4 for auxiliary NCO 0 or Bit 6 for auxiliary NCO 1.

ELAPSED TIME ADJUSTMENT

As shown in Figure 111, the user can read the auxiliary NCO time scale via Bits[79:0] of Register 0x3A00 to Register 0x3A09 (for auxiliary NCO 0) and Register 0x3A0A to Register 0x3A13 (for auxiliary NCO 1). The format of the time scale is a 40-bit integer part (most significant bits) and a 40-bit fractional part (least significant bits). The integer part keeps track of elapsed periods of the auxiliary NCO. The fractional part effectively represents the phase of the accumulator within the current cycle period.

The contents of the time scale registers indicate the elapsed time up to the moment of the most recent IO update.

The user has the option of adjusting the 40-bit integer portion of the time scale register based on Bits[39:0] of Register 0x2819 to Register 0x281D (for auxiliary NCO 0) and Register 0x2859 to Register 0x285D (for auxiliary NCO 1). The specific functionality of the adjustment depends on the value of Bit 1 of Register 0x280F (for auxiliary NCO 0) and Register 0x284F (for auxiliary NCO 1), as explained in the Direct Elapsed Time Adjustment section and the Incremental Elapsed Time Adjustment section.

An application benefiting from elapsed time adjustment is time of day indication, where the user adjusts the integer part of the time scale register to match UTC time. Such an application assumes programming the auxiliary NCO to 1 Hz, which equates to a period of 1 sec.

Direct Elapsed Time Adjustment

Direct elapsed time adjustment is in effect when Bit 1 = 0 of Register 0x280F (for auxiliary NCO 0) and Register 0x284F (for auxiliary NCO 1). For direct elapsed time adjustments, Bits[39:0] constitute an unsigned integer that directly replaces the corresponding 40-bit integer portion of the auxiliary NCO time scale register.

Incremental Elapsed Time Adjustment

Incremental elapsed time adjustment is in effect when Bit 1 = 1 of Register 0x280F (for auxiliary NCO 0) and Register 0x284F (for auxiliary NCO 1). For incremental elapsed time adjustments, the elapsed time adjustment bits (Bits[39:0] in Register 0x281D to Register 0x2819 for auxiliary NCO 0 and Register 0x285D to Register 0x2859 for auxiliary NCO 1), constitute a signed, twos complement integer that increments or decrements the 40-bit integer portion of the auxiliary NCO time scale register relative to its current value.

AUXILIARY NCO TIME STAMPS

The auxiliary NCO generates internal time stamps marking the rollover of its phase accumulator. Because the accumulator updates with a quantized period of $1/f_{ACCUM}$, the accumulator rarely rolls over to exactly zero (that is, most of the time, the accumulator rolls over with a remainder). Therefore, the auxiliary NCO performs the appropriate interpolation to

ensure that the internally generated time stamps precisely mark the zero-crossing point of the accumulator.

The internal time stamps generated by the auxiliary NCO relate to the internal time scale of the AD9546.

Because the internal time stamps truly mark the rollover period of the auxiliary NCO accumulator as well as relate to the internal time scale, the time stamps can serve as an input reference source to either of the DPLLs. This feature allows the AD9546 to have a completely self contained reference source to the DPLLs (that is, no external reference source required).

Because the auxiliary NCO time stamps are an available reference source, the user has access to them through the UTSPs (see the User Time Stamp Processor (UTSP) section). For example, suppose the user programs auxiliary NCO 0 for f_{AUX} = 5 kHz. To read the time stamps from auxiliary NCO 0, the user can program UTSP 0 to use auxiliary NCO 0 as both a time stamp source and as a time scale for UTSP 0. Then, reading the first time stamp yields a reference value related to the time scale of auxiliary NCO 0. However, the next time stamp has a value of 1 unit greater than the first, because the time scale has units of 200 µs (per f_{AUX}). Unfortunately, this method of time stamp conversion by UTSP 0 is not conducive to external use.

However, by relating the time stamps to an appropriate time scale, the time stamp conversion can become meaningful. To do so, program auxiliary NCO 1 for f_{AUX} = 1 Hz. Because f_{AUX} = 1 Hz the time scale for auxiliary NCO 1 carries units of 1 sec. Next, program UTSP 0 to use the time scale from auxiliary NCO 1 instead of the time scale from auxiliary NCO 0 (but keep auxiliary NCO 0 as the time stamp source to UTSP 0). Then, reading the first time stamp yields a reference value (related to the time scale of auxiliary NCO 1) and the next time stamp corresponds to a value precisely 200 µs later than the first with subpicosecond precision.

AUXILIARY NCO PULSE OUTPUT

In addition to generating time stamps, the auxiliary NCOs also produce a physical output signal (see Figure 111). The output signal is effectively a single pulse commensurate with the rollover of the internal phase accumulator of the auxiliary NCO. The auxiliary NCO quantizes the pulse timing such that the peak-to-peak pulse jitter is approximately 1.4 ns.

The user can route the output pulse signal to any one of the Mx pins by defining the desired Mx pin as an output and selecting the desired auxiliary NCO as the output source (see the Status and Control Pins section).

The user programs the duration of the pulse via a significand (S) value and exponent (E) value. To program S, use Bits[3:0] of Register 0x281E (for auxiliary NCO 0) and Register 0x285E (for auxiliary NCO 1). To program E, use Bits[7:4] of the same registers.

$$Pulse\ Width = (1/f_{ACCUM}) \times (1 + S \times 2^{(E+5)})$$

For example, given f_S = 2.4 GHz, S = 6, and E = 2, then

$$Pulse\ Width = (1/f_{ACCUM}) \times (1 + S \times 2^{(E+5)})$$
$$= (96/f_S) \times (1 + 6 \times 2^{(2+5)})$$
$$= (96/(2.4 \times 10^9)) \times (1 + 6 \times 2^{(2+5)})$$
$$= (96/(2.4 \times 10^9)) \times (1 + 6 \times 2^7)$$
$$= 30.76\ \mu s$$

However, given a desired pulse duration (t_P), one must determine the appropriate E and S values for a given system clock PLL VCO frequency (f_S). To do so, let

$$K = (t_P \times f_S)/3072 - 1/32$$

Next, determine E as

$$E = \text{ceil}(\log(K/15)/\log(2))$$

where:
$0 \le E \le 15$.
E = ceil(x) means $E = x$ if x is an integer. Otherwise, E is the nearest integer to x in the positive direction.

Finally, determine S as

$$S = \text{round}(K/2^E)$$

where:
$0 \le S \le 1$.
round(x) means round x to the nearest integer.

When the computation of E or S yields a result outside the specified range, use the appropriate limit. For example, if the computation of E yields a result of –2, then use E = 0.

The K, E, and S values for t_P = 100 µs and f_S = 2.4 GHz are K = 78.09375, E = 3, and S = 10.

E = 3 and S = 10 are within their respective range limits. Because E and S must be integers, the resulting pulse duration is quantized. Given the K, E, and S results for a desired pulse period of t_P = 100 µs, the actual pulse period is

$$t_P = (96/f_S) \times (1 + S \times 2^{(E+5)})$$
$$= 102.44\ \mu s$$

Because the output frequency (f_{AUX}) of the auxiliary NCO and the output pulse width (t_P) are independently programmable parameters, the user must ensure

$$t_P \le 1/f_{AUX}$$

Although the user has the option of generating pulses commensurate with tagged time stamps (see the Tagged Auxiliary NCO Time Stamps section), the $t_P \le 1/f_{AUX}$ limit on t_P still applies (that is, based on f_{AUX} and not the tag frequency).

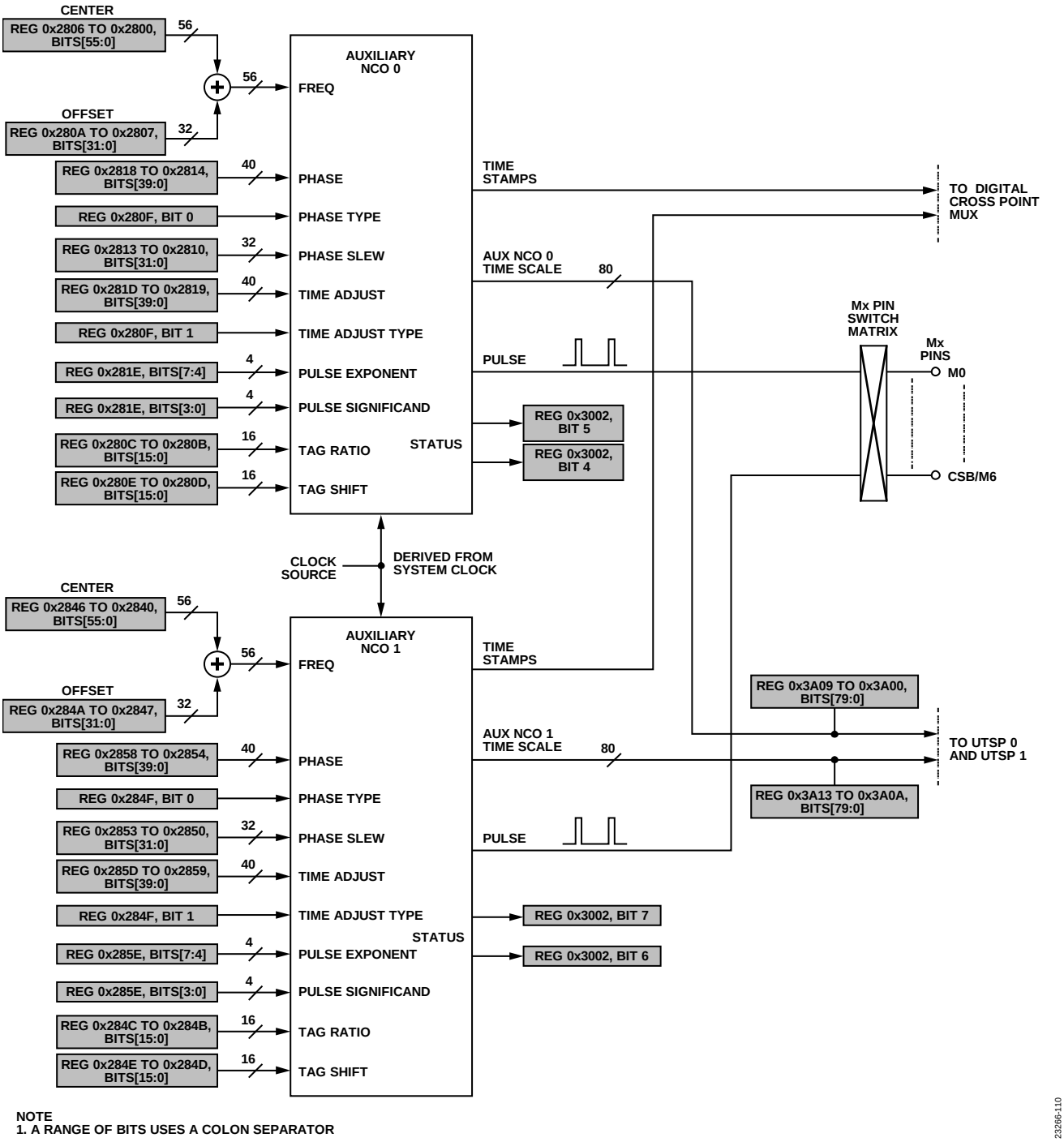


Figure 111. Auxiliary NCO Block Diagram

TEMPERATURE SENSOR

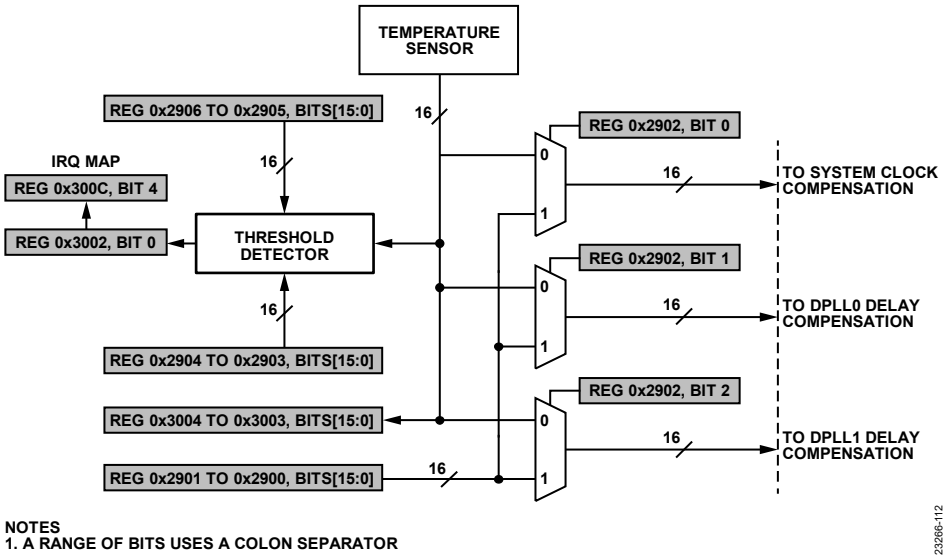


Figure 112. Temperature Sensor Block Diagram

TEMPERATURE SENSOR OVERVIEW

Figure 112 shows a block diagram of the temperature sensor and supporting circuitry. Three AD9546 functional blocks have the capability of using temperature values.

- System clock compensation
- DPLL0 delay compensation
- DPLL1 delay compensation

Each of these functional blocks can receive temperature values from two sources.

- The internal temperature sensor (default)
- Bits[15:0] of Register 0x2900 to Register 0x2901

TEMPERATURE SOURCE SELECTION

The user selects the desired temperature source for each of the temperature compensation functional blocks via Bits[2:0] of Register 0x2902. Bit 0 controls the routing of the temperature source to the system clock compensation block (see the Compensation Method 1 section). Bit 1 and Bit 2 control the routing of the temperature source to the delay compensation block of DPLL0 and DPLL1, respectively, (see the Delay Compensation section). For Bits[2:0], Logic 0 (default) selects the internal temperature sensor, whereas Logic 1 selects the external temperature source (see the External Temperature Source section).

INTERNAL TEMPERATURE SENSOR

The AD9546 has an internal temperature sensor with a digital output (see Figure 112). The sensor reflects the temperature of the silicon die, which the user can read via Bits[15:0] of Register 0x3003 to Register 0x3004. The temperature sensor indicates temperature as a 16-bit signed (twos complement) number scaled by 2^{-7}, as shown in Figure 113. This format provides a range of ±256°C with 0.0078°C resolution.

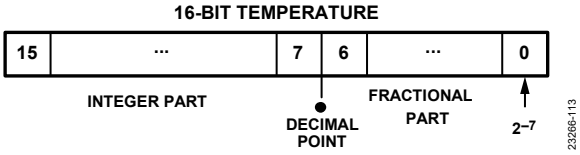


Figure 113. Temperature Format

The sensor samples at approximately 25 MHz and performs a 2^{12}-sample average, providing updates at a rate of approximately 6.1 kHz. The intent of the temperature sensor is for tracking temperature variation rather than providing a measure of absolute temperature.

Given internal temperature = 0x118D (4493 decimal), find the corresponding temperature, T.

$$T = 4493 \times 2^{-7}\text{°C}$$

$$= 35.1015625\text{°C}$$

Given internal temperature = 0xF833 (−1997 decimal),

$$T = -1997 \times 2^{-7}\text{°C}$$

$$= -15.6015625\text{°C}$$

Temperature Alarm Feature

The temperature sensor routes to a threshold detector. The threshold detector detects when temperature readings are outside the range specified by a high temperature threshold and a low temperature threshold. The high temperature threshold resides in Bits[15:0] of Register 0x2905 to Register 0x2906. The low temperature threshold resides in Bits[15:0] of Register 0x2903 to Register 0x2904. The high and low temperature thresholds use the temperature format shown in Figure 113.

If the output value of the temperature sensor falls outside the specified limits, status Bit 0 of Register 0x3002 assumes a Logic 1 state (otherwise, it is Logic 0). Status Bit 0 allows the user to check the status of the threshold detector in real time.

Status Bit 0 of Register 0x3002 is the source for Status Bit 4 of Register 0x300C. Status Bit 4 allows the user to set up automatic notification of a temperature threshold detection event via the IRQ mechanism (see the Interrupt Request (IRQ) section).

EXTERNAL TEMPERATURE SOURCE

The external temperature source constitutes a temperature value programmed into Bits[15:0] of Register 0x2900 to Register 0x2901. Although the internal temperature sensor properly scales temperature values for use by the internal temperature compensation functional blocks automatically, the user must scale the external temperature values properly.

The external temperature bit field represents temperature in units of °C × 2^{-7} (see Figure 113). This format yields a temperature range of ±256°C with 0.0078°C resolution.

For example, to program an external temperature value of 95°C,

$$\textit{External Temperature} = 95/2^{-7}$$
$$= 12{,}160$$
$$= 0x2F80 \text{ (hexadecimal)}$$

SYSTEM CLOCK COMPENSATION

SYSTEM CLOCK COMPENSATION OVERVIEW

The NCOs and the TDCs of the AD9546 derive their timekeeping from the system clock (see the System Clock PLL section). Therefore, the frequency accuracy of any of the NCOs relates directly to the accuracy of the system clock. Likewise, an inferred frequency based on the difference between successive TDC time stamps is subject to the accuracy of the system clock. Therefore, the stability of the system clock is crucial to the accuracy of the NCOs and TDCs within the AD9546.

The stability of the system clock, in turn, relates directly to the stability of the system clock source. The system clock source is the frequency source driving the XOA and XOB pins. As such, an ideally stable system clock source is desirable. In practice, however, the system clock source suffers from frequency instability caused by aging, variations in temperature, and similar physical factors. Any frequency instability introduced by the system clock source translates to a frequency instability in the NCOs and TDCs.

Because the NCOs and TDCs are fundamentally numeric (digital) in nature, it is possible to tune the NCOs and TDCs numerically to counteract the system clock instability. That is, with a known frequency error associated with the system clock source, the user can apply a corresponding correction (numerically) to the NCOs and TDCs, which is the underlying concept of system clock compensation.

The system clock compensation operates on fractional frequency error (FFE) rather than absolute frequency error. For a given nominal frequency, f_0, the FFE of a deviated frequency, f, is

$$FFE = (f - f_0)/f_0$$

or

$$FFE = f/f_0 - 1$$

In the case of TDCs, the difference between successive time stamps is the period of the underlying frequency. The difference between successive time stamps gives rise to the concept of fractional period error (FPE).

$$FPE = (p - p_0)/p_0$$

or

$$FPE = p/p_0 - 1$$

where:
p_0 is the nominal period.
p is the deviated period.

Because frequency relates to period as f = 1/p, FFE and FPE relate as follows:

$$FFE = -FPE/(FPE + 1)$$

or

$$FPE = -FFE/(FFE + 1)$$

In the context of system clock compensation, consider a given system clock frequency error expressed in terms of FFE. Applying an FFE correction factor to the NCOs and TDCs compensates for the FFE error of the system clock source. FFE as a correction factor to the NCOs and TDCs compensates for the FFE of the system clock source. The AD9546 has the option of applying system clock compensation via two methods: open-loop method and closed-loop method.

Open-Loop Method

Figure 114 shows the open-loop method, with the system clock source driving the system clock PLL, which in turn serves as the clock source for a representative NCO within the AD9546. Although Figure 114 shows an NCO, this section also applies to a TDC.

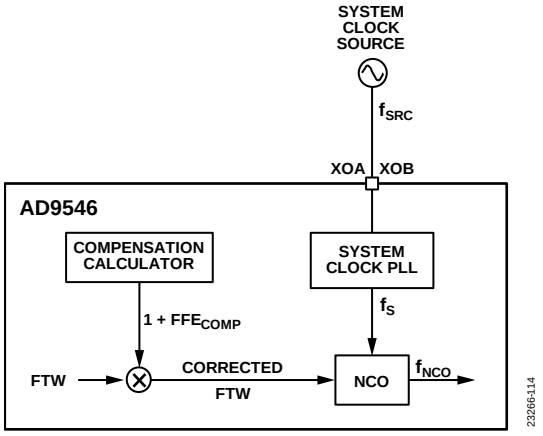


Figure 114. Open-Loop Method

The system clock source provides the primary frequency, f_{SRC}, with a nominal value of f_0. The system clock PLL, which is the clock source to the NCO, multiplies f_{SRC} by a constant, K_{PLL} (that is, $f_S = K_{PLL} \times f_{SRC}$). Assuming the FFE for system clock compensation (FFE_{COMP}) = 0 in Figure 114, the NCO produces an output frequency, f_{NCO}, proportional to the applied numeric FTW (that is $f_{NCO} = f_S \times FTW \times K_{NCO}$, where K_{NCO} is the proportionality constant). Therefore, express f_{NCO} in terms of f_{SRC} as

$$f_{NCO} = f_{SRC} \times K_{PLL} \times FTW \times K_{NCO}$$

The ideal (error free) f_{NCO} is then

$$f_{NCO\_IDEAL} = f_0 \times K_{PLL} \times FTW \times K_{NCO}$$

where f_0 is the nominal frequency of the system clock source. If the system clock source experiences a fractional error, FFE,

$$f_{NCO} = f_0 \times (1 + FFE) \times K_{PLL} \times FTW \times K_{NCO}$$

Apply a fractional correction, FFE_{COMP}, to compensate for FFE as follows:

$$f_{NCO} = f_0 \times (1 + FFE) \times (1 + FFE_{COMP}) \times K_{PLL} \times FTW \times K_{NCO}$$

If $FFE_{COMP} = -FFE$,

$$f_{NCO} = f_0 \times (1 + FFE) \times (1 - FFE) \times K_{PLL} \times FTW \times K_{NCO} \quad (24)$$

The term, (1 + FFE) × (1 − FFE), simplifies to 1 − FFE^2. Given that $FFE^2 = FFE_{COMP} \times FFE$, if both FFE_{COMP} and FFE are less than 1 ppm (10^{-6}), a reasonable assumption, then FFE^2 is less than 10^{-12} (one part in a trillion). For FFE values < 1 ppm, the value of FFE^2 is exceedingly small. Under such conditions, FFE^2 is practically zero, and Equation 24 simplifies to the following:

$$f_{NCO} = f_0 \times K_{PLL} \times FTW \times K_{NCO}$$

This expression for f_{NCO} is the same as f_{NCO_IDEAL}. Therefore, applying the fractional correction term, FFE_{COMP}, nullifies the FFE associated with the system clock source.

The preceding example demonstrates the viability of compensating for the system clock FFE. However, FFE compensation is of little value without a means of predicting FFE so that the appropriate correction can be applied. In this regard, the open-loop method relies on two assumptions.

- Temperature variation dominates the FFE of the system clock source
- The FFE vs. temperature (T) behavior of the system clock source takes the form of an x-order polynomial

In the case of the AD9546, x = 5. Therefore, the latter assumption is expressible as

$$FFE_{SRC} = c_5T^5 + c_4T^4 + c_3T^3 + c_2T^2 + c_1T^1 + c_0 \quad (25)$$

where the coefficients, c_x, define the shape of the FFE_{SRC} curve. c_0 constitutes a static offset, whereas the remaining coefficients relate to powers of T. Note also, that any particular power of T can be artificially eliminated by assigning its corresponding coefficient to zero. For example, reduce the polynomial to second order by making the c_5, c_4, and c_3 coefficients in Equation 25 equal to zero. The open-loop method assumes advance knowledge of the c_x values, which the user programs into the appropriate AD9546 register map locations.

The overall concept of the open-loop method is to program the AD9546 with the appropriate coefficients, c_x, such that the polynomial closely models the temperature dependent behavior of the system clock source. The AD9546 has a built in compensation calculator to implement the FFE polynomial (see the Compensation Method 1 section for more details). The AD9546 provides the means to apply the correction factor generated by the compensation calculator to the NCOs and TDCs designated by the user.

Closed-Loop Method

The closed-loop method, unlike the open-loop method, requires no advance knowledge of the FFE behavior of the system clock source. Instead, the closed-loop method relies on a DPLL and the availability of a very stable external frequency source (GPS, for example) as a reference input to the DPLL (see Figure 115).

In operation, assuming the DPLL is locked and stable, the FTW applied to the NCO is relatively constant, especially under the assumption that the frequency of both the stable frequency source and the system clock source are constant. However, the frequency of the stable frequency source is constant, by definition, which is a necessary condition of the closed-loop method. As such, f_{NCO} is constant because the feedback loop of the DPLL ensures $f_{NCO} = N \times f_{REF}$. Therefore, if f_{SRC} changes (due to temperature, for example), the DPLL feedback loop causes FTW to change in a manner that maintains a constant f_{NCO}. Because the FFE of the stable source is zero (the source is error free, presumably), then any FFE associated with f_{NCO} must be attributable to the FFE of the system clock source. That is, the FFE of the system clock source translates deterministically to a deviation of FTW (from its nominal value, FTW_{NOM}).

In the closed-loop method, FTW variations lead to a correction factor, 1 + FFE_{COMP}. The FTW analyzer shown in Figure 115 is an integral component of the AD9546. Applying the correction factor to other designated NCOs and TDCs in the AD9546 is the basis of the closed-loop method.

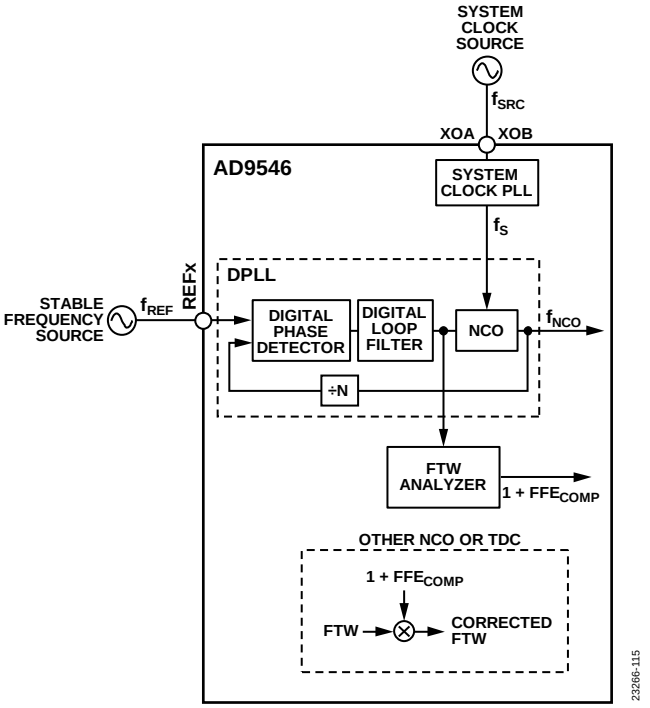


Figure 115. Closed-Loop Method

The dynamic and automatic compensation capability of the closed-loop method typically offers improved performance over the open-loop method but requires the availability of a stable frequency source. In general, the stable frequency source must be at least 10 times more stable than the system clock source for the closed-loop method to be viable.

The AD9546 employs two forms of the closed-loop method. The first form uses one of the DPLL channels (DPLL0 or DPLL1) as the DPLL shown in Figure 115. The second form uses an independent, special purpose DPLL (the auxiliary DPLL) as the DPLL shown in Figure 115. The latter form allows the user to implement closed-loop system clock compensation without sacrificing one of the DPLL channels.

COMPENSATION METHOD 1

Compensation Method 1 employs the open-loop method of system clock compensation (see the Open-Loop Method section). As shown in Figure 116, Compensation Method 1 takes the polynomial coefficients (C_x) and a temperature value (T) to compute a compensation factor (CF_1). Compensation Method 1 differs slightly from the open-loop method by the inclusion of a digital filter (see the Digital Filter section).

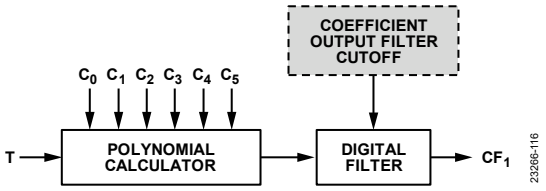


Figure 116. Compensation Method Number 1

The coefficient values, C_0 to C_5, are programmed into the register map by the user (see the Programming the Coefficients (C_x) for Compensation Method 1 section). The temperature value comes from the internal temperature sensor or a temperature register (see the Temperature Sensor section for details). The CF_1 compensation factor applies to the designated NCOs and TDCs to correct for frequency variations of the system clock source. CF_1 embodies the factor, 1 + FFE_{COMP}, shown in Figure 114.

How to Derive Coefficients from Frequency Data

To compensate for the frequency variation of the system clock source, knowledge of the frequency vs. temperature profile is a necessity. Generally, the temperature profile of the source is a table of temperature values with corresponding values of the source frequency. Table 96 shows a hypothetical example of a 25 MHz source in the two leftmost columns.

Because the AD9546 implements compensation in terms of fractional error, it is necessary to convert the frequencies in Table 96 to FFE values (refer to the System Clock Compensation Overview section). Table 96 shows the resulting FFE values. However, Compensation Method 1 generates CF_1 as a time error correction rather than a frequency error correction. Therefore, the user must convert the FFE data to FPE data (refer to the System Clock Compensation Overview section for the conversion formula). Table 96 shows the resulting FPE values. From the calculated FPE values, find the coefficients (C_x) that best fit the FPE data, which generally involves mathematical regression techniques using specialized mathematical software tools.

Applying such regression techniques to the hypothetical FPE values in Table 96 yields the following coefficients:

$C_0 = 1.0046936 \times 10^{-3}$

$C_1 = -2.8927765 \times 10^{-6}$

$C_2 = -1.9110192 \times 10^{-7}$

$C_3 = 2.2493907 \times 10^{-9}$

$C_4 = 2.7943570 \times 10^{-11}$

$C_5 = -2.4817310 \times 10^{-13}$

Table 96. Example 25 MHz Source Data

Temperature (°C)	Frequency (MHz)	FFE (×10^{-6})	FPE (×10^{-6})
−20	24.9757265855	−970.93658	971.88021
−10	24.9745666538	−1017.33385	1018.36987
0	24.9751062576	−995.74970	996.74220
10	24.9758628851	−965.48459	966.41766
20	24.9780535472	−877.85811	878.62943
30	24.9789426612	−842.29355	843.00361
40	24.9809631193	−761.47523	762.05552
50	24.9810049826	−759.80070	760.37843
60	24.9800392641	−798.42944	799.06744
70	24.9777091418	−891.63433	892.43005
80	24.9742106356	−1031.57458	1032.63982

Figure 117 shows the FPE vs. temperature curve resulting from these coefficients. The trace is the predicted FPE vs. temperature curve based on the coefficients, and the solid points are the temperature vs. FPE data from Table 96.

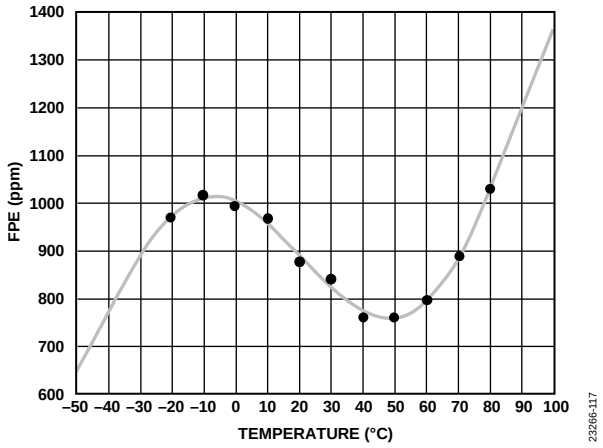


Figure 117. FPE vs. Temperature

Given a set of coefficients like C_0 to C_5, the user must convert the numeric values to suitable register contents (see the Programming the Coefficients (Cx) for Compensation Method 1 section for details).

Programming the Coefficients (C_x) for Compensation Method 1

Compensation Method 1 makes use of six coefficients, C_x, where the index, x, ranges from 0 to 5. Coefficient C_0 is represented by a 40-bit (signed) coefficient value (CV) residing in Register 0x0289 to Register 0x028D.

The relationship between the value of CV and the value of C_0 is

$$CV = C_0/2^{-45}$$

The value of CV has no units and is proportional to the system clock period.

For example, convert $C_0 = 1.0046936 \times 10^{-3}$ to its corresponding register value (C_{0_RegVal}).

$$C_{0\_RegVal} = C_0/2^{-45}$$
$$= 1.0046936 \times 10^{-3}/2^{-45}$$
$$= 35{,}349{,}513{,}305 \text{ (nearest integer)}$$
$$= 0x\ 08\ 3AFE\ C459 \text{ (hexadecimal)}$$

Coefficient C_1 to Coefficient C_5 are represented by employing two components: a 16-bit (signed) significand and an 8-bit (signed) exponent. The significand and exponent components reside in the register map per Table 97.

Table 97. System Clock Compensation Coefficients

	Register Address	
Coefficient	**Significand**	**Exponent**
C_1	0x028E to 0x028F	0x0290
C_2	0x0291 to 0x0292	0x0293
C_3	0x0294 to 0x0295	0x0296
C_4	0x0297 to 0x0298	0x0299
C_5	0x029A to 0x029B	0x029C

C_1 through C_5 apply scale factors to corresponding powers of T. The C_1 through C_5 coefficients have the following format:

$$C_x = S_x \times 2^{E_x}$$

where:
x is the coefficient index.
S_x is the significand component.
E_x is the exponent component.

In practice, C_x is a base 10 number, as are S_x and E_x. However, the AD9546 requires S_x and E_x to be signed integers. The following procedure explains how to convert base 10 C_x values to the necessary signed integers corresponding to the significand register value (C_{x_S}) and the exponent register value (C_{x_E}).

There are three cases for C_x.

- The trivial case ($C_x = 0$)
- The case where C_x is less than the quantization limit
- The case where C_x is greater than the quantization limit

For the first two cases (that is, $C_x = 0$ or C_x is less than the quantization limit),

$$C_{x\_S} = 0 \text{ (0x0000 hexadecimal)}$$

$$C_{x\_E} = 0 \text{ (0x00 hexadecimal)}$$

To test if C_x is less than the quantization limit, check whether the following inequality is true:

$$\left\lfloor \frac{\log|C_x|}{\log 2} \right\rfloor + 1 < -127 \quad (26)$$

where:
log(x) is the logarithm of x (for example, ln(x), $\log_2(x)$, $\log_{10}(x)$).
|x| is the absolute value of x.
⌊x⌋ is the notation for signifying the nearest integer to x in the direction of −∞.

If C_x is nonzero and the inequality in Equation 26 is false, then

$$E = \left\lfloor \frac{\log|C_x|}{\log 2} \right\rfloor + 1$$

$$C_{x\_E} = E$$

$$C_{x\_S} = \text{round}(C_x \times 2^{15-E})$$

Using C_1 as an example, convert $C_1 = -2.8927765 \times 10^{-6}$ to its corresponding register values. First, check that C_1 is greater than the quantization limit.

$$E = \left\lfloor \frac{\log|C_1|}{\log 2} \right\rfloor + 1$$

$$= \left\lfloor \frac{\log|-2.8927765 \times 10^{-6}|}{\log 2} \right\rfloor + 1$$

$$= -18$$

This value is greater than the quantization limit of −127. Therefore,

$C_{1\_E} = E = -18 = 0xEE$ (hexadecimal)

$C_{1\_S} = \text{round}(C_1 \times 2^{15-E})$

$= \text{round}(-2.8927765 \times 10^{-6} \times 2^{15-(-18)})$

$= -24{,}849$

$= 0x9EEF$ (hexadecimal)

Using C_2 as an example, convert $C_2 = -1.9110192 \times 10^{-7}$ to its corresponding register values. First, check that C_2 is greater than the quantization limit.

$$E = \left\lfloor \frac{\log|C_2|}{\log 2} \right\rfloor + 1$$

$$= \left\lfloor \frac{\log|-1.9110192 \times 10^{-7}|}{\log 2} \right\rfloor + 1$$

$$= -22$$

This value is greater than the quantization limit of −127. Therefore,

$C_{2\_E} = E = -22 = 0xEA$ (hexadecimal)

$C_{2\_S} = \text{round}(C_2 \times 2^{15-E})$

$= \text{round}(-1.9110192 \times 10^{-7} \times 2^{15-(-22)})$

$= -26{,}265$

$= 0x9967$ (hexadecimal)

Using C_3 as an example, convert $C_3 = 2.2493907 \times 10^{-9}$ to its corresponding register values. First, check that C_3 is greater than the quantization limit.

$$E = \left\lfloor \frac{\log|C_3|}{\log 2} \right\rfloor + 1$$

$$= \left\lfloor \frac{\log|2.2493907 \times 10^{-9}|}{\log 2} \right\rfloor + 1$$

$$= -28$$

This value is greater than the quantization limit of −127. Therefore,

$C_{3\_E} = E = -28 = 0xE4$ (hexadecimal)

$C_{3\_S} = \text{round}(C_3 \times 2^{15-E})$

$= \text{round}(2.2493907 \times 10^{-9} \times 2^{15-(-28)})$

$= 19{,}786$

$= 0x4D4A$ (hexadecimal)

Using C_4 as an example, convert $C_4 = 2.7943570 \times 10^{-11}$ to its corresponding register values. First, check that C_4 is greater than the quantization limit.

$$E = \left\lfloor \frac{\log|C_4|}{\log 2} \right\rfloor + 1$$

$$= \left\lfloor \frac{\log|2.7943570 \times 10^{-11}|}{\log 2} \right\rfloor + 1$$

$$= -35$$

This value is greater than the quantization limit of −127. Therefore,

$C_{4\_E} = E = -35 = 0xDD$ (hexadecimal)

$C_{4\_S} = \text{round}(C_4 \times 2^{15-E})$

$= \text{round}(2.7943570 \times 10^{-11} \times 2^{15-(-35)})$

$= 31{,}462$

$= 0x7AE6$ (hexadecimal)

Using C_5 as an example, convert $C_5 = -2.4817310 \times 10^{-13}$ to its corresponding register values. First, check that C_5 is greater than the quantization limit.

$$E = \left\lfloor \frac{\log|C_5|}{\log 2} \right\rfloor + 1$$

$$= \left\lfloor \frac{\log|-2.4817310 \times 10^{-13}|}{\log 2} \right\rfloor + 1$$

$$= -41$$

This value is greater than the quantization limit of −127. Therefore,

$C_{5\_E} = E = -41 = 0xD7$ (hexadecimal)

$C_{5\_S} = \text{round}(C_5 \times 2^{15-E})$

$= \text{round}(-2.4817310 \times 10^{-13} \times 2^{15-(-41)})$

$= -17{,}883$

$= 0xBA25$ (hexadecimal)

Use the following formulas to verify the register value computations, C_x', closely match the desired coefficient value:

$$C_0' = C_{0\_RegVal} \times 2^{-45} \quad (27)$$

$$C'_{x\,|\,x\neq 0} = C_{x\_S} \times 2^{C_{x\_E}-15} \quad (28)$$

Note that there may be a slight difference between the values produced by Equation 27 and Equation 28 and the original coefficient values (C_x) due to quantization effects.

An example of coefficient verification follows:

$C_0' = 35{,}349{,}513{,}305 \times 2^{-45}$

$= 1.004694 \times 10^{-3}$ (compare: $C_0 = 1.0046936 \times 10^{-3}$)

$C_1' = -24{,}849 \times 2^{-18-15}$

$= -2.892804 \times 10^{-6}$ (compare: $C_1 = -2.8927765 \times 10^{-6}$)

$C_2' = -26{,}265 \times 2^{-22-15}$

$= -1.911030 \times 10^{-14}$ (compare: $C_2 = -1.9110192 \times 10^{-7}$)

$C_3' = 19{,}786 \times 2^{-28-15}$

$= 2.249408 \times 10^{-9}$ (compare: $C_3 = 2.2493907 \times 10^{-9}$)

$C_4' = 31{,}462 \times 2^{-35-15}$

$= 2.794387 \times 10^{-11}$ (compare: $C_4 = 2.7943570 \times 10^{-11}$)

$C_5' = -17{,}883 \times 2^{-41-15}$

$= -2.481765 \times 10^{-13}$ (compare: $C_5 = -2.4817310 \times 10^{-13}$)

Digital Filter

The CF_1 values produced by the polynomial calculator block in Figure 116 depend on temperature measurements (internal or external) as an input parameter. Any noise associated with those measurements is a potential noise source on CF_1, which can lead to a degradation of phase noise performance. To mitigate potential noise injection by the compensation mechanism, Compensation Method 1 employs a filter that applies a smoothing function to the raw CF_1 values.

The user controls the filter bandwidth via Bits[2:0] of Register 0x0288 according to Table 98. The filter comprises two collocated poles that yield a 3 dB bandwidth per Table 98.

Table 98. Digital Filter Bandwidth

Bits[2:0] (Binary)	3 dB Bandwidth (Hz)
000	156
001	78
010	39
011	20
100	10
101	5
110	2
111	1

The AD9546 automatically bypasses the digital filter under the following conditions:

- The system clock is unstable.
- The difference between the input and output of the digital filter exceeds ~125 ppm.

None of these conditions occurs during normal operation. Thus, the filter is nearly always present.

COMPENSATION METHOD 2

Compensation Method 2 employs the closed-loop method of system clock compensation (see the Closed-Loop Method section). As shown in Figure 118, Compensation Method 2 makes use of one of the DPLLx channels (where x is 0 or 1). The user must specify which DPLL to use via Bit 0 of Register 0x0287. Logic 0 (default) selects DPLL0, and Logic 1 selects DPLL1 as the source of CF_2. CF_2 embodies the factor, $1 + FFE_{COMP}$, in Figure 115.

Compensation Method 2 generates the correction factor, CF_2, which the user can apply to other NCOs and TDCs within the AD9546. However, a compensation factor, $COMP_x$, can be applied to the NCO of DPLLx in Figure 118. Because it is possible for the $COMP_x$ factor to include CF_2 (see the Integrated Compensation Subsystem section for details), the user must ensure that the NCO of the DPLL channel selected for Compensation Method 2 is not itself a recipient of CF_2 via $COMP_x$.

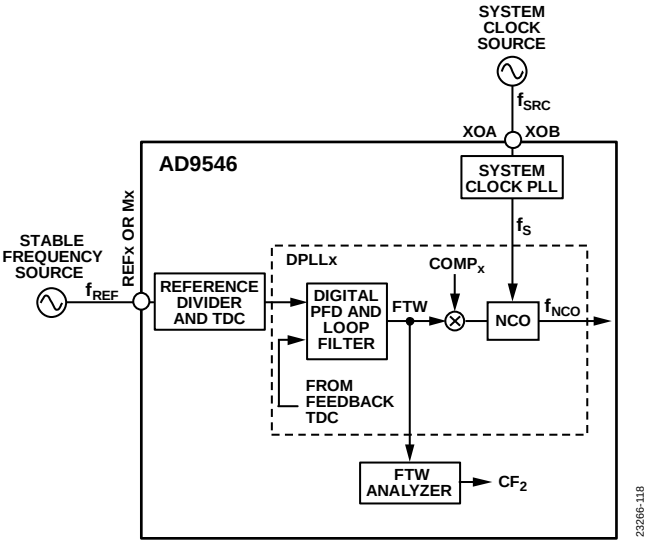


Figure 118. Compensation Method Number 2

COMPENSATION METHOD 3

Compensation Method 3 is unavailable in digitized clocking applications (see the Common Clock DPLL section).

Compensation Method 3 employs the closed-loop method of system clock compensation (see the Closed-Loop Method section). As shown in Figure 119, Compensation Method 3 produces correction factor CF_3, where CF_3 embodies the factor, $1 + FFE_{COMP}$, in Figure 115. Compensation Method 3 and Compensation Method 2 are similar, but Compensation Method 3 uses the auxiliary DPLL rather than one of the DPLL channels. Because Compensation Method 3 relies on the availability of a stable external clock source applied to one of the REFx inputs or one of the auxiliary REFx inputs (via an Mx pin), the user must assign the TDC associated with the external clock source to the auxiliary DPLL via Bits[4:0] of Register 0x0284. Table 99 shows the assignment codes for the various TDCs.

Table 99. Auxiliary DPLL TDC Assignment Codes

Bits[4:0] (Decimal)	TDC Description
0	REFA TDC (default)
1	REFAA TDC
2	REFB TDC
3	REFB TDC
4 to 5	Unused
6	Auxiliary REF0 TDC
7	Auxiliary REF1 TDC
8 to 10	Unused
11	Auxiliary REF2 TDC
12	Auxiliary REF3 TDC
13 to 15	Unused

The auxiliary DPLL can itself receive compensation from other sources via $COMP_x$ in Figure 119 (see the Integrated Compensation Subsystem section for details). Unlike Compensation Method 2, the CF_3 compensation factor does not include the contribution of $COMP_x$. That is, CF_3 comprises only the residual error to represent the frequency deviation associated with the system clock source only.

Assuming the auxiliary DPLL is locked and has fully settled, the period of the stable reference must exactly match the programmed reference period stored in the register associated with the selected reference (see Figure 119). Any deviation is an indication of error associated with the system clock period and leads to a corresponding CF_3 value. There are two error sources: stability error and accuracy error.

Stability error involves deviation from the mean value over time (the mean corresponding to a specific operating condition like 25°C, for example). As temperature varies over time, the frequency of the oscillator serving as the source for the system clock varies in response to the temperature changes. Given the prerequisite that the stable reference is significantly more stable than the system clock source, the dominant variations appearing in the feedback path are those of the system clock. Because the system clock variations are the dominant contributor to CF_3, CF_3 reflects the corrections necessary to compensate for the temperature induced stability errors on the system clock.

Accuracy error is the difference between the defined nominal frequency and the actual frequency. The defined nominal frequency derives from programmed reference period stored in the register associated with the selected reference. The actual frequency is f_{REF}, the true frequency of the stable reference (see Figure 119). The accuracy error is the difference between the programmed reference period and the actual nominal period of the stable reference ($1/f_{REF}$).

Because CF_3 is a composite of both the stability and accuracy error sources, the programmed reference period directly affects the accuracy error component of CF_3. Thus, accurate entry of the stable reference clock period is crucial, because the accuracy error propagates to any $COMP_x$ involving CF_3 (see the Integrated Compensation Subsystem section).

Auxiliary DPLL Loop Bandwidth

The user controls the loop bandwidth of the auxiliary DPLL servo loop (BW_{COMP}) via Bits[15:0] (unsigned) in Register 0x0285 to Register 0x0286. Note that BW_{COMP} is 0.1 Hz scaled by the 16-bit decimal value of Bits[15:0].

For example, given a desired value of BW_{COMP} = 247.63 Hz, determine the appropriate value of Bits[15:0] as follows:

$$\begin{aligned} Bits[15:0] &= BW_{COMP}/0.1 \\ &= 247.63/0.1 \text{ Hz} \\ &= 2476 \text{ (nearest integer)} \\ &= 0x\ 09AC \text{ (hexadecimal)} \end{aligned}$$

The programmed bandwidth affects how long it takes the auxiliary DPLL to lock to the stable reference: a lower bandwidth means it takes the auxiliary DPLL longer to lock. The programmed bandwidth also affects the ability of the auxiliary DPLL to track variations in the frequency (f_S) of the system clock. Specifically, a relatively stable, slowly varying system clock frequency allows the use of a low loop bandwidth, whereas a system clock exhibiting more rapid frequency variations requires a wider loop bandwidth. For example, using an OCXO as the system clock reference allows the use of a 0.1 Hz loop bandwidth, whereas a standard crystal oscillator necessitates a loop bandwidth in excess of 50 Hz.

Auxiliary DPLL Reference Monitor Status

The auxiliary DPLL has a dedicated reference monitor to provide indication that the reference period is within limits. The reference monitor relies on the defined reference period (user input via the appropriate registers).

The auxiliary DPLL reference monitor status is available in real time via Bit 2 (auxiliary DPLL reference status) of Register 0x3002. The auxiliary DPLL reference fault and unfault states are Logic 1 and Logic 0, respectively.

The auxiliary DPLL reference monitor status is also available via the IRQ mechanism (see the Interrupt Request (IRQ) section) by means of Bit 2 and Bit 3 of Register 0x300C. Bit 2 and Bit 3 provide latched information regarding state transitions of the auxiliary DPLL reference status bit. Bit 2 latches to Logic 1 when the auxiliary DPLL reference status bit transitions from Logic 0 to Logic 1, whereas Bit 3 latches to Logic 1 when the auxiliary DPLL reference status bit transitions from Logic 1 to Logic 0.

Because Bit 2 and Bit 3 represent the latched state transitions of the auxiliary DPLL reference status bit, they may represent a condition that is no longer true. Therefore, the user must clear Bit 2 and Bit 3 via Bit 2 and Bit 3, respectively, of Register 0x2007. Otherwise, the user may lose indication of subsequent state changes of the auxiliary DPLL reference status bit.

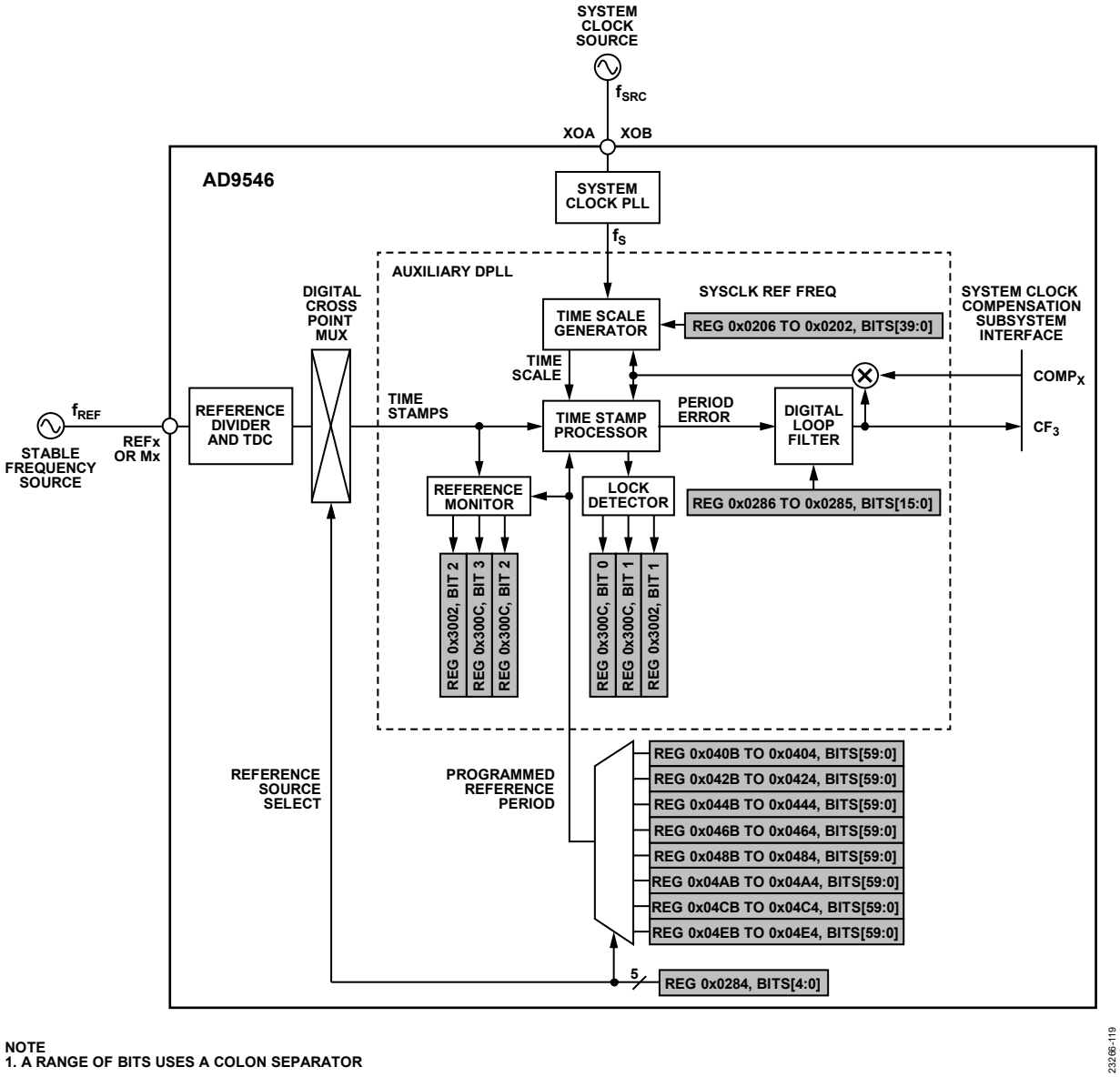


Figure 119. Compensation Method 3

Auxiliary DPLL Lock Detector

The auxiliary DPLL has a built in lock detector for indicating lock status. The lock and unlock thresholds for the lock detector are fixed and not user programmable.

The auxiliary DPLL lock detector status is available in real time via status Bit 1 (auxiliary DPLL lock status) of Register 0x3002. The auxiliary DPLL lock and unlock states are Logic 1 and Logic 0, respectively.

The auxiliary DPLL lock detector status is also available via the IRQ mechanism (see the Interrupt Request (IRQ) section) by means of Bit 0 and Bit 1 of Register 0x300C. Bit 0 and Bit 1 provide latched information regarding state transitions of the auxiliary DPLL lock status bit. Bit 0 latches to Logic 1 when the auxiliary DPLL lock status bit transitions from Logic 0 to Logic 1, whereas Bit 1 latches to Logic 1 when the auxiliary DPLL lock status bit transitions from Logic 1 to Logic 0.

Because Bit 0 and Bit 1 represent the latched state transitions of the auxiliary DPLL lock status bit, they may represent a condition that is no longer true. Therefore, the user must clear Bit 0 and Bit 1 via Bit 0 and Bit 1, respectively, of Register 0x2007. Otherwise, the user may lose indication of subsequent state changes of the auxiliary DPLL lock status bit.

Auxiliary DPLL Holdover

Under normal conditions, the auxiliary DPLL continuously updates CF_3 computations with each rising edge event to the TDC associated with the stable reference. Under certain conditions, however, the auxiliary DPLL can enter a holdover state. For example,

- The user programs auxiliary DPLL bandwidth = 0
- The reference monitor indicates a fault

While in the holdover state, the auxiliary DPLL is effectively in an open-loop condition (that is, the NCO output is static). As such, the auxiliary DPLL holds the CF_3 value that existed just prior to entering the holdover state. Therefore, it is not possible to ascertain further stability information until the loop resumes normal operation. When the condition that forced holdover no longer exists, however, the auxiliary DPLL returns to closed-loop operation and resumes updating CF_3 (that is, when the lock detector indicates a lock condition).

INTEGRATED COMPENSATION SUBSYSTEM

The three compensation methods comprise the integrated compensation subsystem of the AD9546. The compensation subsystem allows the user to employ any combination of the three compensation methods to any of the NCOs and TDCs within the AD9546 (barring a few exceptions per the Compensation Assignment Guidelines section). Figure 120 shows a simplified block diagram of the compensation subsystem.

In Figure 120, note the implied feedback path between DPLL0/DPLL1 and Compensation Method 2 and between the auxiliary DPLL and Compensation Method 3. The reason for the implied feedback is that the indicated compensation destination (the auxiliary DPLL, for example) can receive compensation from any of the compensation sources via the combiner and distributor. Thus, the compensation destination can potentially apply self compensation, which leads to an undesired positive feedback loop. The user must avoid such loops (see the Compensation Assignment Guidelines section).

For digitized clocking applications (see the Digitized Clocking section), the common clock DPLL replaces the auxiliary DPLL. Thus, in a digitized clocking application, CF_3 originates from the common clock DPLL. The implication is that, when applying system clock compensation in a digitized clocking application, the burden of stability falls on the common clock reference to the common clock DPLL.

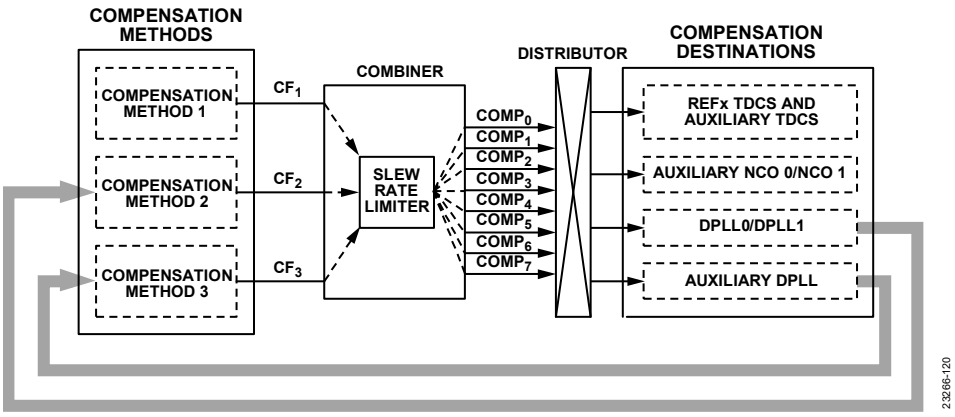


Figure 120. Integrated Compensation Subsystem

Each compensation method produces its own compensation factors (CF_1, CF_2, and CF_3). The combiner translates the three CF_x factors to eight possible combined compensation factors, $COMP_x$, according to Table 100. The index, x, of $COMP_x$ is the 3-bit value programmed in the associated compensation destination bit field per Figure 121.

Table 100. Composite Compensation Factor ($COMP_x$)

$COMP_x$ Index (x)	Compensation Factor Index	Combiner Output
0	Not applicable	0
1	1	CF_1
2	2	CF_2
3	1, 2	$CF_1 \times CF_2$
4	3	CF_3
5	1, 3	$CF_1 \times CF_3$
6	2, 3	$CF_2 \times CF_3$
7	1, 2, 3	$CF_1 \times CF_2 \times CF_3$

The distributor gives the user the ability to connect any $COMP_x$ output from the combiner to any compensation destination. The AD9546 implements the distributor through six 3-bit bit fields, where each bit field corresponds to a specific compensation destination per Figure 121. The user assigns a $COMP_x$ factor to a specific destination by writing a 3-bit value to the corresponding compensation destination bit field. See the System Clock Compensation Programming Registers section for a register programming example.

Any of the $COMP_x$ selections (except for $COMP_0$) constitute a real-time sequence of compensation values originating from the various compensation sources. This real-time sequence continuously applies to the associated compensation destination. Because the $COMP_X$ outputs respond to the various stimuli associated with the compensation sources, $COMP_x$ tends to vary over time, causing the associated compensation destination to vary accordingly. However, if the frequency variation vs. time slope is too large, the slope can appear as an unwanted noise source to compensation destinations employing closed-loop compensation (such as DPLL0, DPLL1, and the auxiliary DPLL).

To mitigate the potential noise injection caused by $COMP_x$ exhibiting an excessive frequency variation vs. time slope, the combiner employs programmable rate limiting via a slew rate limiter. The slew rate limiter effectively prevents the rate of change of frequency originating from compensation sources from exceeding a preset limit.

The user controls rate limiting via Bits[2:0] of Register 0x0283 according to Table 101. The rate limit values are in units of parts per million per second (ppm/sec or 10^{-6}/sec) and represent the maximum rate of change of $COMP_x$ that occurs given the value of Bits[2:0]. Because rate limiting applies globally to $COMP_1$ through $COMP_7$, the user must base the rate limit selection on the stability requirements of the most frequency sensitive compensation destination.

Table 101. Compensation Rate Limiting

Bits[2:0] (Binary)	Rate Limit (ppm/sec)
000	None
001	0.715
010	1.430
011	2.860
100	5.720
101	11.44
110	22.88
111	45.76

Compensation Assignment Guidelines

When using a UTSP (see the User Time Stamp Processor (UTSP) section) with an auxiliary NCO selected as the time scale, both the auxiliary NCO and the UTSP time stamp source must use the same $COMP_x$ factor. That is, the user must program identical values in the bit field of the REFx or auxiliary REFx TDC compensation destination and in the bit field of the auxiliary NCO 0 or auxiliary NCO 1 compensation destination (whichever is the selected time scale for the UTSP). See Figure 121 for the compensation destination bit fields.

To avoid a positive feedback loop in the integrated compensation subsystem, do not assign $COMP_x$ sources involving CF_2 in Table 100 to the following compensation destinations:

- DPLL0 NCO compensation destination when Bit 0 of Register 0x0287 is Logic 0
- DPLL1 NCO compensation destination when Bit 0 of Register 0x0287 is Logic 1

Likewise, do not assign $COMP_x$ sources involving CF_3 in Table 100 to the auxiliary DPLL (the reason Bit 6 of Register 0x280 is shaded out in Figure 121).

SYSTEM CLOCK COMPENSATION PROGRAMMING REGISTERS

System clock compensation comprises three compensation methods and six compensation destinations. Three 8-bit registers constitute the function of the distributor in Figure 120. The three registers function as a connection matrix with six 3-bit bit fields distributed over the three registers (see Figure 121).

Each 3-bit bit field associates with one of the following six compensation destinations:

- Auxiliary DPLL
- REFx TDCs and auxiliary REFx TDCs
- Auxiliary NCO 0
- Auxiliary NCO 1
- DPLL0
- DPLL1

The value of the 3-bit bit field for a given destination establishes the $COMP_x$ factor applied to that destination (see the Integrated Compensation Subsystem section for details).

The compensation methods relate to columns in the bit field matrix of Figure 121, whereas the compensation destinations relate to rows. Organizing the compensation bit field matrix in this way enables any compensation destination to receive any one of the eight possible $COMP_x$ signals (see Figure 120). Certain compensation destinations have constraints on which compensation methods apply (see the Compensation Assignment Guidelines section).

For example, to provide system clock compensation to Auxiliary NCO 0 using a combination of Compensation Method 1 and Compensation Method 3, follow these steps:

1. Determine the appropriate compensation destination bit field per Figure 121. In this case, Bits[2:0] of Register 0x0281 apply because the compensation destination is auxiliary NCO 0.
2. Determine the 3-bit code to write to Bits[2:0] of Register 0x0281 per the leftmost column of Table 100 corresponding to the appropriate $COMP_x$. In this case, the code is 5 (101 binary) because it uses CF_1 and CF_3 per the middle column of Table 100. In conclusion, by programming Bits[2:0] = 5 in Register 0x0281, auxiliary NCO 0 receives both Compensation Method 1 and Compensation Method 3.

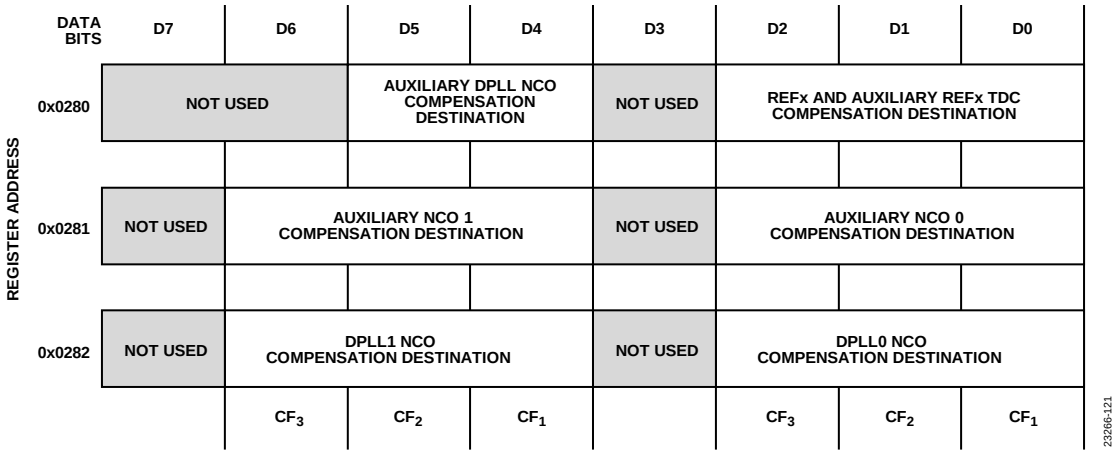


Figure 121. System Clock Compensation Registers

STATUS AND CONTROL PINS

STATUS AND CONTROL PINS OVERVIEW

The AD9546 features seven independently configurable digital CMOS status or control pins (M0 to CSB/M6, hereafter referred to as Mx). Configuring an Mx pin as a status pin causes that pin to be an output. Conversely, configuring an Mx pin as a control pin causes that pin to be an input. During power-up or reset, the Mx pins temporarily become inputs and only allow the device to autoconfigure. Figure 122 is a block diagram of the Mx pin functionality.

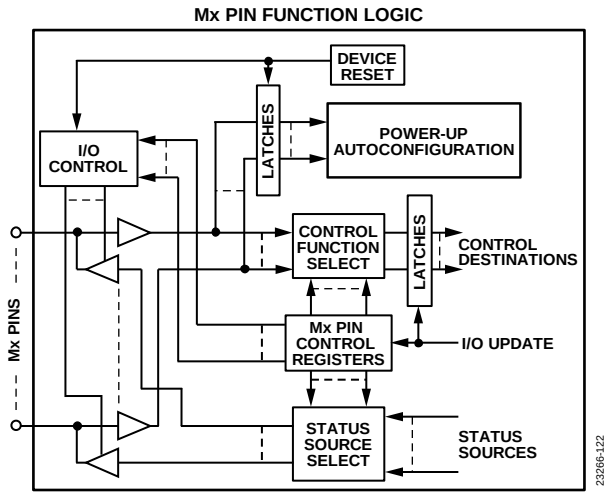


Figure 122. Mx Pin Logic

The Mx pin control logic uses special register write detection logic to prevent the Mx pins from behaving unpredictably when the Mx pin function changes, especially when changing mode from input to output or vice versa.

When an Mx pin functions as an output, it continues operating according to the prior function, even after the user programs the corresponding registers. However, assertion of an IO update causes the corresponding pins to switch to the new function according to the newly programmed register contents. Changing from one output function to another output function on an Mx pin does not require special timing to avoid input/output contention on the pin.

When an Mx pin functions as an input, programming a specific Mx pin function register causes all the Mx pin control functions to latch their values. Assertion of an IO update switches to the newly programmed pin function, at which time normal behavior resumes. When switching from one input function to another input function on the same pin, the logic state driven at the input to the pin can change freely during the interval between writing the new function to the corresponding register and asserting an IO update.

When switching the operation of an Mx pin from an input to an output function, the recommendation is that the external drive source become high impedance during the interval between writing the new function and asserting the IO update bit.

When switching the operation of an Mx pin from an output to an input, the recommendation is as follows. First, program the Mx pin input function to no operation and assert the IO update bit. This configuration avoids input/output contention on the Mx pin or other undesired behavior because, prior to the assertion of the IO update bit, the device continues to drive the Mx pin. Following the assertion of the IO update bit, the device releases the Mx pin but ignores the logic level on the pin due to the programmed no operation function. The recommendation is to avoid using a high impedance source on an Mx pin configured as an input because doing so may cause excessive internal current consumption. Second, drive the Mx pin with Logic 0 or Logic 1 via the desired external source and program the associated Mx pin register from no operation to the desired function.

MULTIFUNCTION PINS AT RESET OR POWER-UP

At power-up or in response to a reset operation, the Mx pins enter a special operating mode. For a brief interval following a power-up or reset operation, the Mx pins function only as inputs (the internal drivers enter a high impedance state during a power-up/reset operation). During this brief interval, the device latches the logic levels at the Mx inputs and uses this information to autoconfigure the device accordingly. The Mx pins remain high-Z until either an EEPROM operation occurs, in which case M1 or M2 become an I^2C master, or the user (or EEPROM) programs them to be outputs.

If the user does not connect external pull-up or pull-down resistors to the Mx pins, the M3 and M4 pins have internal pull-down resistors to ensure a predictable start-up configuration. In the absence of external resistors, the internal pull-down resistors ensure that the device starts up with the serial port in SPI mode and without automatically loading data from an external EEPROM (see Table 102). Although the M0, M1, M2, SDO/M5, and CSB/M6 pins are high impedance at startup, connect external 100 kΩ pull-down or pull-up resistors to these pins to ensure a deterministic start-up condition.

Table 102 shows the Mx pin start-up conditions. M0, M1, and M2 do not appear in Table 102 because these pins have no explicit function during a power-up or reset operation.

Table 102. Mx Pin Function at Startup or Reset

Mx Pin	Start-Up/Reset Function	Logic 1	Logic 0
M3	EEPROM load function	Load from EEPROM	Do not load from EEPROM (default)
M4	Serial port function	I^2C mode	SPI mode (default)
SDO/M5	I^2C address offset	See Table 103	See Table 103
CSB/M6	I^2C address offset	See Table 103	See Table 103

When the start-up conditions select the serial port to be I²C mode (that is, M4 is Logic 1 at startup), the SDO/M5 and CSB/M6 pins determine the I²C port device address offset, per Table 103. The logic levels in Table 103 only apply during a power-up or reset operation.

Table 103. I²C Device Address Offset

CSB/M6	SDO/M5	M4	Address Offset
X[1]	X[1]	0	Not applicable
0	0	1	1001000 (0x48)
0	1	1	1001001 (0x49)
1	0	1	1001010 (0x4A)
1	1	1	1001011 (0x4B)

[1] X means don't care.

DEFINING AN Mx PIN FOR STATUS OR CONTROL

To define an Mx pin as a status or control pin, use Bit 7 in the register address associated with the Mx pin per Table 104. Logic 0 (default) defines the associated Mx pin as a control pin, whereas Logic 1 defines the associated Mx pin as a status pin.

Table 104. Mx Pin Mode Register Addresses

Mx Pin	Register Address
M0	0x0102
M1	0x0103
M2	0x0104
M3	0x0105
M4	0x0106
SDO/M5	0x0107
CSB/M6	0x0108

The status or control function for an Mx pin depends on an 8-bit code stored in the Mx pin function register addresses associated with a specific Mx pin per Table 105.

Because the Mx pin mode and code reside in different registers, the following programming procedure is necessary. First, write the desired mode via Bit 7 of the appropriate mode register (see Table 104) with Bits[6:0] = 0. Then, write the desired code to the corresponding code register (see Table 105). Finally, assert an IO update. When programming more than one Mx pin, it is acceptable to wait until all the desired Mx pin mode and code registers are programmed before asserting an IO update.

Table 105. Mx Pin Status or Control Code Register Addresses

Mx Pin	Register Address
M0	0x0182
M1	0x0183
M2	0x0184
M3	0x0185
M4	0x0186
SDO/M5	0x0187
CSB/M6	0x0188

A 2-bit modifier augments the status or control functionality of an Mx pin. Each Mx pin has a dedicated modifier bit pair per Table 106. The modifier bits affect the associated Mx pin differently depending on the Mx pin operating mode: status or control. The specific meaning of the 2-bit code appears in Table 107 for status functions and in Table 108 for control functions.

Table 106. Modifier Bits of Mx Pin Functionality

Register Address	[D7:D6]	[D5:D4]	[D3:D2]	[D1:D0]
0x0100	M3	M2	M1	M0
0x0101	Unused	M6	M5	M4

STATUS FUNCTIONALITY

Configuring an Mx pin for status functionality gives the user access to specific internal device status and IRQ functions in the form of a hardware pin that produces a logic signal.

The drive strength of an Mx pin when configured for status functionality is programmable via Bits[6:0] of Register 0x0109, where Bits[6:0] correspond to M6 to M0, respectively). Logic 0 (default) selects normal drive strength (~6 mA) and Logic 1 selects weak drive strength (~3 mA).

When configured as a status pin, the modifier bits affect the status functionality per Table 107. The PMOS open-drain mode requires an external pull-down resistor. The NMOS open-drain mode requires an external pull-up resistor. The open-drain modes enable the implementation of wire-OR'ed functionality of multiple Mx status pins (including Mx status pins across multiple AD9546 devices or other compatible devices—for example, to implement an IRQ bus).

Table 107. Mx Status Pin 2-Bit Modifier Codes from Table 106

Code	Mode	Description
00	CMOS, active high	Default. Output is Logic 0 for a deasserted internal function and Logic 1 for an asserted internal function.
01	CMOS, active low	Output is Logic 1 for deasserted internal function and Logic 0 for an asserted internal function.
10	PMOS, open drain	Output is high impedance for a deasserted internal function and active high for an asserted internal function.
11	NMOS, open drain	Output is high impedance for a deasserted internal function and active low for an asserted internal function.

CONTROL FUNCTIONALITY

Configuring an Mx pin for control functionality gives the user control of specific internal device functions via an external hardware logic signal. When configured as a control pin, the modifier bits affect the control functionality per Table 108.

Table 108. Mx Control Pin 2-Bit Modifier Codes

Code	Boolean	Description
00	AND	Logical AND the associated Mx control pin with the other Mx control pins assigned to the same control function.
01	NOT AND	Invert the logical state of the associated Mx control pin and AND it with the other Mx control pins assigned to the same control function.
10	OR	Logical OR the associated Mx control pin with the other Mx control pins assigned to the same control function.
11	NOT OR	Invert the logical state of the associated Mx control pin and OR it with the other Mx control pins assigned to the same control function.

The Boolean functionality of aggregated Mx control pins follows a hierarchy whereby logical OR operations occur before logical AND operations. The OR and NOT OR operations are collectively grouped into a single result. A logical AND is then performed using that result and the remaining AND and NOT AND operations.

Consider a case where the M0, M2, M3, and CSB/M6 pins are configured for control functionality and are assigned to the IO update control function. That is, Bits[7:0] = 0x01 in Register 0x0182, Register 0x0184, Register 0x0185, and Register 0x0188.

In addition, M0 is assigned for AND operation (00), M2 for NOT OR (11) operation, M3 for NOT AND (01) operation, and M6 for OR (10) operation (that is, the 2-bit codes according to Table 108 in the appropriate bit positions of Register 0x0100 and Register 0x0101 per Table 106).

With these settings, the IO update function behaves according to the following Boolean expression:

((NOT *M2*) OR *M6*) AND *M0* AND (NOT *M3*)

In this case, an IO update occurs when M0 is Logic 1 and M3 is Logic 0, and either M2 is Logic 0 or M6 is Logic 1.

When an Mx control pin acts on a control function individually (rather than as part of a group, per the previous example), the Boolean functionality of the codes in Table 108 reduces to two possibilities. Namely, Code 00 and Code 10 specify a Boolean true (the Mx pin logic state applies to the corresponding control function directly), whereas Code 01 and Code 11 specify a Boolean false (the Mx pin logic state applies to the corresponding control function with a logical inversion).

INTERRUPT REQUEST (IRQ)

IRQ OVERVIEW

The AD9546 monitors certain internal device events, potentially allowing them to trigger an IRQ event. Three groups of registers (see Figure 123) control the IRQ functionality within the AD9546.

- IRQ status registers (Register 0x300B through Register 0x301E)
- IRQ mask registers (Register 0x010C through Register 0x011F)
- IRQ clear registers (Register 0x2006 through Register 0x2019)

The IRQ logic can indicate an IRQ event status result for any specific device events via the logical OR of the status of all the IRQ status bits. In addition, the IRQ logic offers IRQ event status results for select groups of specific IRQ events, namely, the PLL0 IRQs, PLL1 IRQs, and common IRQs (see Figure 123).

The PLL0 IRQ group includes all device events associated with DPLL0 and APLL0. The PLL1 IRQ group includes all device events associated with DPLL1 and APLL1. The common IRQ group includes events not associated with PLL0 or PLL1 (like the system clock, EEPROM, reference monitors, and others).

IRQ STATUS BITS

The IRQ status bits maintain a record of specific IRQ events. The occurrence of a specific device event results in the setting and latching of the corresponding bit in the IRQ status registers (assuming the user enables those specific IRQ events (see the IRQ Mask Bits section)).

IRQ MASK BITS

The IRQ mask bits comprise a bit for bit correspondence with the IRQ status. Writing a Logic 1 to a mask bit allows (unmasks) the corresponding specific device event to propagate to the IRQ status bits. A Logic 0 (default) prevents (masks) the corresponding specific device event to propagate to the IRQ status bits.

The presence of the IRQ mask allows the user to select certain device events for generating an IRQ event, while ignoring (masking) all other specific device events from contributing to an IRQ (PLL0 IRQ, PLL1 IRQ, common IRQ, or any IRQ signal in Figure 123).

The default state of the IRQ mask bits is Logic 0. Therefore, the device is not capable of generating an IRQ until the user populates the IRQ mask with a Logic 1 to unmask the desired IRQ events. Writing a Logic 1 to an IRQ mask bit may result in immediate indication of an IRQ if the corresponding IRQ event is already asserted (that is, the device previously registered the corresponding IRQ event).

IRQ CLEAR BITS

The IRQ clear bits comprise a bit for bit correspondence with the IRQ status bits. Writing a Logic 1 to an IRQ clear bit forces the corresponding IRQ event to Logic 0, thereby clearing that specific IRQ event. The IRQ clear registers are autoclearing. Therefore, after writing a Logic 1 to any IRQ clear bit in Register 0x2006 to Register 0x2019, the device automatically restores the IRQ clear bit to Logic 0. The IRQ remains asserted until the user clears all of the IRQ clear bits associated with PLL0 IRQ, PLL1 IRQ, common IRQ, or any IRQ signal shown in Figure 123).

Although it is not recommended, in certain applications, it may be desirable to clear an entire IRQ group all at one time. Register 0x2005 provides four bits for clearing IRQ groups. Bit 0 clears all IRQ clear bits. Bit 1 clears the IRQ clear bits associated with the common IRQ bits. Bit 2 clears the IRQ clear bits associated with the PLL0 IRQ bits. Bit 3 clears the IRQ clear bits associated with the PLL1 IRQ bits.

Alternatively, the user can program any of the multifunction pins as an input for clearing an IRQ group, which allows clearing an IRQ group with an external logic signal rather than by writing to Register 0x2005 (see Figure 123).

The recommendation for clearing IRQ events is to first service the specific IRQ event (as needed) and then clear the specific IRQ for the given IRQ event. Clearing IRQ groups via Register 0x2005 or via an Mx pin requires great care. Clearing an IRQ group all at one time may result in the unintentional clearing of one or more asserted IRQ events. Clearing asserted IRQ events eliminates the record of the associated device events, subsequently erasing any history of those events having occurred.

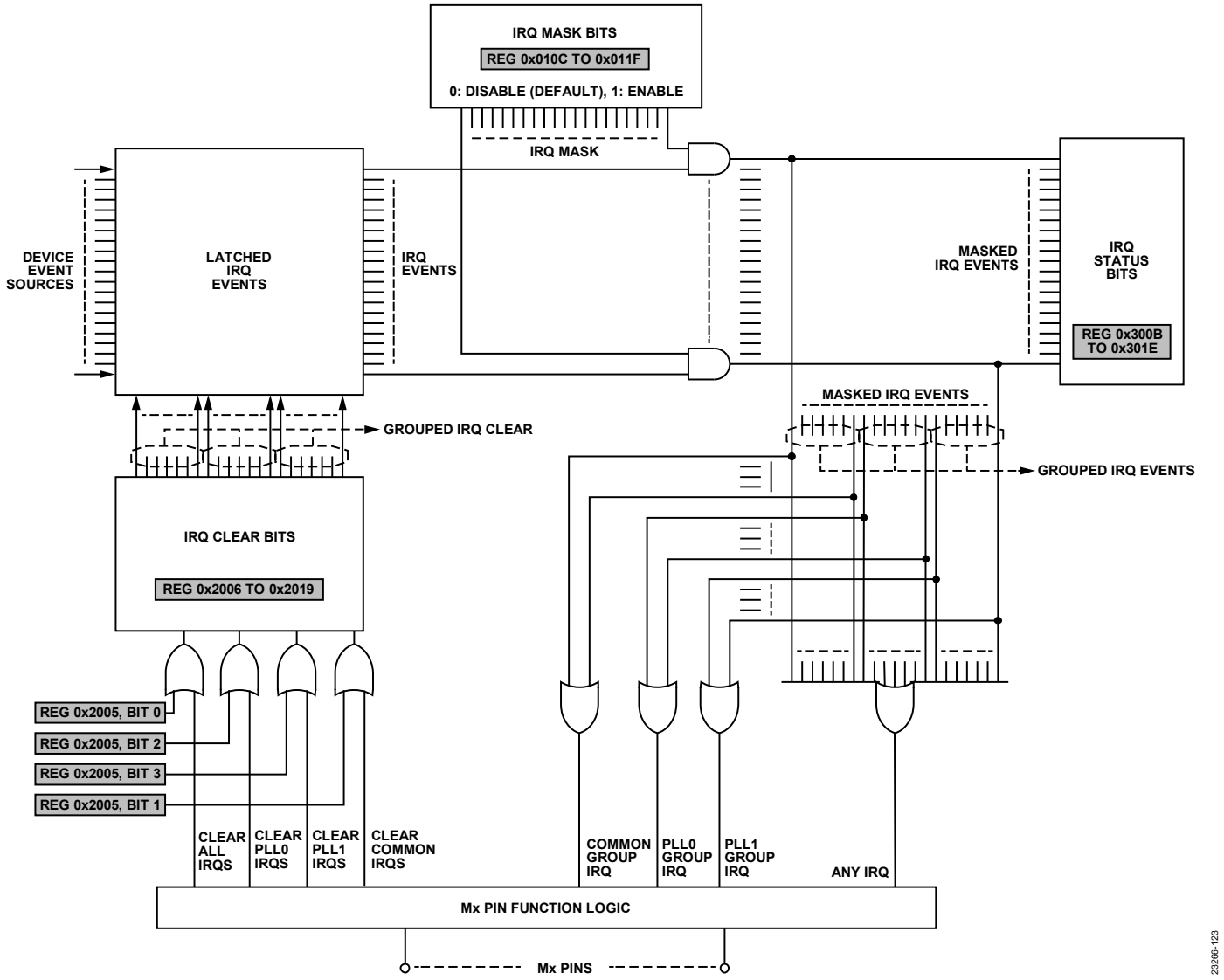


Figure 123. IRQ System Diagram

WATCHDOG TIMER

The watchdog timer is a general-purpose programmable timer capable of triggering a specific IRQ event (see Figure 124). The timer relies on the system clock, however. Therefore, the system clock must be present and locked for the watchdog timer to be functional. The bit fields associated with the watchdog timer reside in the Mx pin status and control function section of the register map.

The user sets the period of the watchdog timer via Bits[15:0] in Register 0x010A and Register 0x010B. The 16-bit unsigned timeout value carries units of milliseconds and provides a range of 1 ms to 65.535 sec.

A value of zero (default) disables the watchdog timer. The relative accuracy of the timer is approximately 0.1% with an uncertainty of 0.5 ms.

Updating the watchdog timer value automatically clears the timer, ensuring a correct timeout period (per the new value) starting from the moment of the update.

The 16-bit watchdog timer value relates to the timeout period as follows:

$$Watchdog\ Timer\ Value = Timeout\ Period \times 10^3$$

To determine the watchdog timer value necessary for a timeout period of 10 sec,

$$Watchdog\ Timer\ Value = Timeout\ Period \times 10^3$$
$$= 10 \times 10^3$$
$$= 10{,}000$$
$$= 0x2710\ (hexadecimal)$$

If enabled, the timer runs continuously and generates a timeout IRQ event when the timeout period expires. The user has access to the watchdog timer status as an IRQ bit (see the Interrupt Request (IRQ) section) via Bit 2 of Register 0x300B.

Because Bit 2 of Register 0x300B is a latched IRQ bit, the user must clear it via Bit 2 of Register 0x2006 to obtain visibility of subsequent watchdog timeout events. The user also has access to the watchdog timeout status by assigning the watchdog timer to an Mx status pin. In this case, the timeout event of the watchdog timer is a pulse spanning 96 system clock periods (approximately 40 ns).

There are two ways to reset the watchdog timer, thereby preventing it from indicating a timeout event. The first method is by writing a Logic 1 to Bit 7 (an autoclearing bit) in Register 0x2005. Alternatively, the user can program any of the Mx pins as a control pin to reset the watchdog timer, which allows the user to reset the timer using a hardware pin rather than using the register map.

There are two typical cases for employing the watchdog timer, both of which assume the watchdog timer output appears at the output of an appropriately configured Mx status pin. The first case is for an external device (an FPGA or microcontroller, for example) to monitor the watchdog timer output, using the output as a signal to carry out periodic housekeeping functions. The second case is to have the watchdog timer output connected to the external device such that the assertion of the watchdog output resets the external device. In this way, under normal operation, the external device repeatedly resets the watchdog timer by either writing Logic 1 to the clear watchdog bit or by asserting an Mx control pin configured for clearing the watchdog. Therefore, as long as the external device keeps resetting the watchdog timer before the timer times out, the watchdog timer does not generate an output signal. As such, the watchdog timer does not reset the external device. However, if the external device fails to reset the watchdog timer before the timeout period expires, the watchdog timer eventually times out, resetting the external device via the appropriately configured Mx status pin.

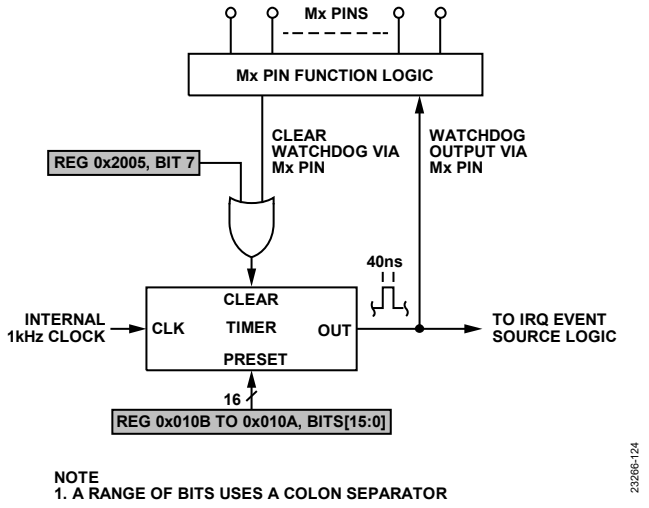


Figure 124. Watchdog Timer

EEPROM USAGE

EEPROM OVERVIEW

The AD9546 supports an external, I²C-compatible, EEPROM with dedicated access. With some restrictions, the AD9546 also supports multidevice access to a single external EEPROM on a shared I²C common bus. The AD9546 has an on-chip I²C master to interface to the EEPROM through the Mx pins.

Because the default register settings of the AD9546 do not define a specific frequency translation, the user must factory program the EEPROM content before it can be downloaded to the register map (either automatically or manually). If desired, the user can store custom device configurations by manually forcing an upload to the EEPROM via the register map.

EEPROM CONTROLLER GENERAL OPERATION

EEPROM Controller

The EEPROM controller governs all aspects of communication with the EEPROM. Because the I²C interface uses a 100 kHz (normal mode) or 400 kHz (fast mode) communication link, the controller runs synchronous to an on-chip generated clock source suitable for use as the I²C serial clock. The on-chip oscillator enables asynchronously, immediately on a request for activation of the controller. Upon startup, the oscillator notifies the controller of its availability, and the controller activates. After the requested controller operation is complete, the controller disables the clock generator and returns to an idle state.

EEPROM Download

An EEPROM download transfers contents from the EEPROM to the AD9546 programming registers and invokes specific actions per the instructions stored in the EEPROM (see Table 109). Automatic downloading is the most common method for initiating an EEPROM download sequence, which initiates at power-up of the AD9546, provided Pin M3 is Logic 1 at power-up (see the Multifunction Pins at Reset or Power-Up section). Alternatively, instead of cycling power to the AD9546 to initiate an EEPROM download, the user can force the RESETB pin to Logic 0, force Pin M3 to Logic 1, then return the RESETB pin to Logic 1, and remove the drive source from Pin M3.

The user can also request an EEPROM download on demand (that is, without resetting or cycling power to the AD9546) by writing a Logic 1 to Bit 0 of Register 0x2E03.

Bit 0 does not require an IO update. Writing a Logic 1 to this bit immediately triggers a download sequence.

The EEPROM controller sets Bit 1 of Register 0x3000 to Logic 1 while the download sequence is in progress as an indication to the user that the controller is busy.

EEPROM Upload

To store the AD9546 register contents in the EEPROM, the user must write a Logic 1 to Bit 0 of Register 0x2E02. Bit 0 does not require an IO update. Writing a Logic 1 to this bit immediately triggers an upload sequence.

The AD9546 has the equivalent of a write protect feature in that the user must write a Logic 1 to Bit 0 of Register 0x2E00 prior to requesting an upload to the EEPROM. Attempting to upload to the EEPROM without first setting Bit 0 results in a fault indication (that is, the AD9546 asserts Bit 1 of Register 0x300B).

A prerequisite to an EEPROM upload is the existence of an upload sequence stored in the 15-byte EEPROM sequence section of the register map (Register 0x2E10 to Register 0x2E1E). That is, the user must store a series of upload instructions (see the EEPROM Instruction Set section) in the EEPROM sequence section of the register map prior to executing an EEPROM upload.

The EEPROM controller performs an upload sequence by reading the instructions stored in the EEPROM sequence section of the register map byte by byte, and executing them in order. That is, the data stored in the EEPROM sequence section of the register map are instructions to the EEPROM controller on what to store in the EEPROM (including operational commands and AD9546 register data).

The EEPROM controller sets Bit 0 of Register 0x3000 to Logic 1 while the upload sequence is in progress as an indication to the user that the controller is busy.

Because the EEPROM sequence section of the register map is only 15 bytes, the sequence typically cannot hold enough instructions to upload a complete set of AD9546 data to the EEPROM. Therefore, most upload sequences necessitate that the user upload a series of subsequences. For example, to accomplish a required upload sequence consisting of 20 bytes of instructions, perform the following procedure:

1. Write the first 14 instructions to the EEPROM sequence registers with the 15th instruction being a pause instruction (see Table 109).
2. Initiate an EEPROM upload by writing Logic 1 to Bit 0 of Register 0x2E02. When the EEPROM controller reaches the pause instruction, it suspends the upload process and waits for another assertion of Bit 0 of Register 0x2E02.
3. While the controller pauses, write the remaining six bytes of the upload sequence into the EEPROM sequence registers, followed by an end of data instruction (see Table 109).
4. Initiate an EEPROM upload by writing Logic 1 to Bit 0 of Register 0x2E02. When the EEPROM controller reaches the end of data instruction, the upload process terminates.

The preceding procedure is an example of an upload sequence consisting of two subsequences. Most upload sequences require more than two subsequences. However, the procedure is the same. Specifically, partition a long sequence into several subsequences by using the pause instruction at the end of each subsequence and the end of data instruction at the end of the final subsequence.

EEPROM Checksum

When the EEPROM controller encounters an end of data instruction (see Table 109) during an upload sequence, it computes a 32-bit cyclic redundancy check (CRC) checksum and appends it to the stored data in the EEPROM. Similarly, when the EEPROM controller executes a download sequence, it computes a checksum on-the-fly. At the end of a download sequence, the EEPROM controller compares the newly computed checksum to the one stored in the EEPROM. If the two checksums do not match, the EEPROM controller asserts Bit 3 of Register 0x3000.

To minimize the possibility of downloading a corrupted EEPROM data set, the user can execute a checksum test by asserting the Bit 2 of Register 0x2E00, which causes the EEPROM controller to execute a download sequence, but without transferring data to the AD9546 registers. The controller still computes an on-the-fly checksum, performs the checksum comparison, and asserts Bit 3 of Register 0x3000 if the checksums do not match. Therefore, after the device completes the download sequence and deasserts Bit 1 of Register 0x3000, the user can check that Bit 3 of Register 0x3000 = 0 to verify that the test passed. However, even if the test fails, device operation is unaffected because there was no transfer of data to the AD9546 registers.

EEPROM Header

The EEPROM controller adds a header to stored data that carries information related to the AD9546, such as

- Vendor ID
- Chip type
- Product ID
- Chip revision

At the beginning of an EEPROM download sequence, the EEPROM controller compares the stored header values to the values in the corresponding registers of the AD9546. If the controller detects a mismatch, it asserts Bit 2 of Register 0x3000 and terminates the download.

EEPROM INSTRUCTION SET

The EEPROM controller relies on a combination of instructions and data. An instruction consists of a single byte (eight bits). Some instructions require subsequent bytes of payload data. That is, some instructions are self contained operations, whereas others are directions on how to process subsequent payload data. Table 109 shows a summary of the EEPROM controller instructions.

When the controller downloads the EEPROM contents to the AD9546 registers, it does so in a linear fashion, stepping through the instructions stored in the EEPROM. However, when the controller uploads to the EEPROM, the sequence is a nonlinear combination of various parts of the register map and computed data that the controller calculates on the fly.

Table 109. EEPROM Controller Instruction Set Summary

Instruction Code (Hexadecimal)	Response	Comments
0x00 to 0x7F	Register transfer	Requires a 2-byte register address suffix
0x80	IO update	Assert an IO update during download
0x81 to 0x8F	Not applicable	Undefined
0x90	Calibrate all PLLs	Calibrate the system clock PLL, APLL0, and APLL1 during download
0x91	Calibrate the system clock PLL	Calibrate only the system clock PLL during download
0x92	Calibrate APLL0 and APLL1	Calibrate only APLL0 and APLL1 during download
0x93	Calibrate APLL0	Calibrate only APLL0 during download
0x94	Calibrate APLL1	Calibrate only APLL1 during download
0x95 to 0x97	Not applicable	Reserved/unused
0x98	Force freerun	Force DPLL0 and DPLL1 to freerun during download
0x99	Force DPLL0 freerun	Force only DPLL0 to freerun during download
0x9A	Force DPLL1 freerun	Force only DPLL1 to freerun during download
0x9B to 0x9F	Not applicable	Reserved/unused
0xA0	Synchronize outputs	Synchronize all distribution outputs during download
0xA1	Synchronize Channel 0	Synchronize only Channel 0 distribution outputs during download
0xA2	Synchronize Channel 1	Synchronize only Channel 1 distribution outputs during download
0xA3 to 0xAF	Not applicable	Reserved/unused
0xB0	Clear condition	Apply Condition 0 and reset the condition map
0xB1 to 0xBF	Set condition	Apply Condition 1 to Condition 15, respectively
0xC0 to 0xFD	Not applicable	Undefined
0xFE	Pause	Pause the EEPROM upload sequence
0xFF	End of data	Marks the end of the instruction sequence

Register Transfer Instructions (0x00 to 0x7F)

Instructions with a hexadecimal value from 0x00 through 0x7F prescribe a register transfer operation. Register transfer instructions require a 2-byte suffix, which constitutes the starting address of the AD9546 register targeted for transfer (where the first byte to follow the data instruction is the most significant byte of the register address). When the EEPROM controller encounters a data instruction, it knows to interpret the next two bytes as the register map target address.

The value of the register transfer instruction encodes the payload length (number of bytes). That is, the EEPROM controller knows how many register bytes to transfer to and from the indicated register by adding 1 to the instruction value. For example, Data Instruction 0x1A has a decimal value of 26. Therefore, the controller knows to transfer 27 bytes (one more than the value of the instruction) to the target register for a load operation or from the target register for a save operation.

IO Update Instruction (0x80)

When the EEPROM controller encounters an IO update instruction during an upload sequence, it stores the instruction in EEPROM. When encountered during a download sequence, however, the EEPROM controller initiates an IO update event, which is equivalent to the user asserting Bit 0 of Register 0x000F (IO update).

Device Action Instructions (0x90 to 0xA2)

When the EEPROM controller encounters a device action instruction during an upload sequence, it stores the instruction in EEPROM. When encountered during a download sequence, however, the EEPROM controller executes the specified action per Table 109.

Conditional Instructions (0xB0 to 0xBF)

The conditional instructions allow conditional execution of EEPROM instructions during a download sequence. During an upload sequence, however, they are stored as is and have no effect on the upload process.

Conditional processing makes use of four elements:

- The conditional instruction
- The condition value
- The condition ID
- The condition map

Conditional Instruction

When encountering a conditional instruction during an upload sequence, the EEPROM controller stores the instruction in the EEPROM. When the EEPROM controller detects a conditional instruction during a download sequence, the instruction affects the condition map and the outcome of the conditional processing.

Condition Value

The condition value has a one to one correspondence to the conditional instruction. Specifically, the condition value is the value of the conditional instruction minus 0xB0. Therefore, condition values have a range of 0 to 15. The EEPROM controller uses condition values in conjunction with the condition map, whereas the user uses a condition value to populate the EEPROM load condition bit field of the register map with a condition ID.

Condition ID

The condition ID is the value stored in Bits[3:0] of Register 0x2E01. The EEPROM controller uses the condition ID in conjunction with the condition map to determine which instructions to execute or ignore during a download sequence.

Condition Map

The condition map is a table maintained by the EEPROM controller consisting of a list of condition values. When the EEPROM controller encounters a conditional instruction during a download sequence, it determines the corresponding condition value of the instruction (0 to 15). If the condition value is nonzero, the EEPROM controller places the condition value in the condition map. Conversely, if the condition value is zero, the controller clears the condition map and applies Condition 0 (the absence of conditions). Condition 0 causes all subsequent instructions to execute unconditionally (until the controller encounters a new conditional instruction that causes conditional processing).

Conditional Processing

While executing a download sequence, the EEPROM controller executes or skips instructions depending on the condition ID and the contents of the condition map (except for the conditional and end of data instructions, which always execute unconditionally).

If the condition map is empty or the condition ID is zero, all instructions execute unconditionally during download. However, if the condition ID is nonzero and the condition map contains a condition value matching the condition ID, the EEPROM controller executes the subsequent instructions. Alternatively, if the condition ID is nonzero but the condition map does not contain a condition value matching the condition ID, the EEPROM controller skips instructions until it encounters a conditional instruction with a condition value of zero or a condition value matching the condition ID.

The condition map allows multiple conditions to exist at any given moment. This multiconditional processing mechanism enables the user to have one download instruction sequence with many possible outcomes, depending on the value of the condition ID and the order in which the controller encounters conditional instructions. Table 110 shows an example of the use of conditional processing.

Table 110. Example Conditional Processing Sequence

Instruction	Operation
0x00 to 0x7F	A sequence of register transfer instructions that execute unconditionally
0xB1	Apply Condition 1
0x00 to 0x7F	A sequence of register transfer instructions that execute only if the condition ID is 1
0xB2	Apply Condition 2
0xB3	Apply Condition 3
0x00 to 0x7F	A sequence of register transfer instructions that execute only if the condition ID is 1, 2, or 3
0x91	Calibrate the system clock PLL
0xB0	Clear condition map
0x80	IO update
0xFF	Terminate sequence

Pause Instruction (0xFE)

The EEPROM controller only recognizes the pause instruction during an upload sequence. Upon encountering a pause instruction, the EEPROM controller enters an idle state, but preserves the current value of the EEPROM address pointer.

One use of the pause instruction is for saving multiple, yet distinct, values of the same AD9546 register, which is useful for sequencing power-up conditions.

The pause instruction is also useful for executing an upload sequence requiring more space than is available in the EEPROM sequence registers (see the EEPROM Upload section).

End of Data Instruction (0xFF)

When the EEPROM controller encounters an end of data instruction during an upload sequence, it stores the instruction in EEPROM along with the computed checksum, clears the EEPROM address pointer, and then enters an idle state. When encountered during a download sequence, however, the EEPROM controller clears the EEPROM address pointer, verifies the checksum, and then enters an idle state.

During EEPROM downloads, condition instructions always execute unconditionally.

MULTIDEVICE SUPPORT

Multidevice support enables multiple AD9546 devices to share the contents of a single EEPROM. There are two levels of multidevice support. Level 1 supports a configuration where multiple AD9546 devices share a single EEPROM through a dedicated I²C bus. Level 2 supports a configuration where multiple AD9546 devices share a single EEPROM connected to a common I²C bus that includes other I²C master devices. Figure 125 and Figure 126 show the Level 1 and Level 2 configurations, respectively. Only the I²C functions of the multifunction pins are shown.

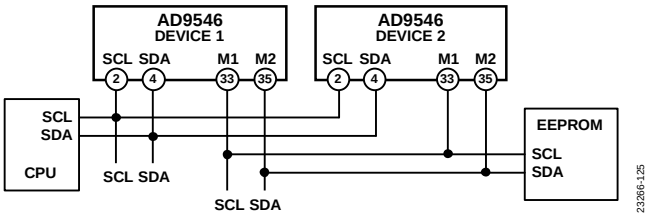


Figure 125. Level 1 Multidevice Configuration

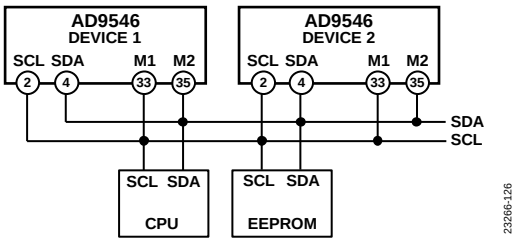


Figure 126. Level 2 Multidevice Configuration

Multidevice Bus Arbitration

The EEPROM controller implements bus arbitration by continuously monitoring the SDA and SCL bus signals for start and stop conditions. The controller can determine whether the bus is idle or busy. If the bus is busy, the EEPROM controller delays its pending I²C transfer until a stop condition indicates that the bus is available.

Bus arbitration is essential in cases where two I²C master devices simultaneously attempt an I²C transfer. For example, if one I²C master detects that SDA is Logic 0 when it is intended to be Logic 1, the master assumes that another I²C master is active and immediately terminates its own attempt to transfer data. Similarly, if one I²C master detects that SCL is Logic 0 prior to entering a start state, the master assumes that another I²C master is active and stalls its own attempt to drive the bus.

In either case, the prevailing I²C master completes its current transaction before releasing the bus. Because the postponed I²C master continuously monitors the bus for a stop condition, it attempts to seize the bus and carry out the postponed transaction on detection of such a stop condition.

The EEPROM controller includes an arbitration timer to optimize the bus arbitration process. Specifically, when the EEPROM controller postpones an I²C transfer as a result of detecting bus contention, the arbitration timer starts. If the EEPROM controller fails to detect a stop condition within 255 SCL cycles, it attempts to force another transaction. If the bus is still busy, the EEPROM controller restarts the arbitration timer, and the process continues until the EEPROM controller eventually completes the pending transaction.

Multidevice Configuration Example

Consider two AD9546 devices (Device 1 and Device 2) that share a single EEPROM and assume both devices have a common PLL0 configuration but differing PLL1 configurations.

A template for an EEPROM sequence that accomplishes this configuration is shown in Table 111.

The sequence relies on conditional processing to differentiate between Device 1 and Device 2. Therefore, the user must program the condition ID of both devices prior to executing an EEPROM download. Specifically, the user must program Bits[3:0] of Register 0x2E01 in Device 1 with a condition ID of 1 and in Device 2 with a condition ID of 2.

Table 111. Template for a Multidevice EEPROM Sequence

Instructions	Comment
0x00 to 0x7F	A sequence of register transfer instructions associated with the PLL0 configuration common to both devices
0xB1	Apply Condition 1
0x00 to 0x7F	A sequence of register transfer instructions associated with the PLL1 configuration specific to Device 1
0xB0	Clear the condition map
0xB2	Apply Condition 2
0x00 to 0x7F	A sequence of register transfer instructions associated with the PLL1 configuration specific to Device 2
0xB0	Clear the condition map
0x80	IO update
0xFF	End of sequence

APPLICATIONS INFORMATION

DIGITIZED CLOCKING APPLICATION

The diagram in Figure 127 shows a simplified example of a digitized clocking system. The system contains three digitized clocking nodes, with one of the nodes functioning as the master node and the other two nodes functioning as slave nodes (Slave Node 0 and Slave Node 1). The master node receives a common clock reference signal that serves as the common clock for the entire system.

A reference clock can be applied at an input to one of the slave nodes. The goal is to distribute the reference clock to each of the nodes using data transport over a common digital communication link rather than via analog clock signals. That is, to transport the reference clock in frequency and phase from the selected node to the other nodes over a data link rather than routing the reference clock throughout the system in the form of an analog signal. Digitized clocking technology enables this ability to transport clock signals over a digital bus (see the Digitized Clocking section for details).

The master node generates the common clock as a reference clock to the slave nodes. The common clock is the time base shared by all nodes, a fundamental digitized clocking requirement. The master node also sends synchronization information to the slave nodes. Synchronization comprises two components: a physical synchronization event signal (embedded within the common clock signal in Figure 127) and a synchronizing code sent over the digital communication bus (corresponding to the time associated with the synchronization event signal). This two-pronged synchronization approach results in all slave nodes sharing a common epoch with the master node, with all nodes having a common (synchronized) time scale.

For example, suppose the digitized clocking system has a distributable reference clock applied to Slave Node 1. Slave Node 1 digitizes the reference clock by creating time codes (digital word representations of the reference clock rising edges) derived from the common time scale. The time codes are a proxy for the physical reference clock signal. Using digitized clocking, Slave Node 1 sends the time codes to the master node over the digital bus. Likewise, the master node distributes the time codes to Slave Node 0 (and back to the Slave Node 1, if required) over the same digital bus. Then, using digitized clocking, the receiving nodes convert the time codes back into a physical clock signal with the same frequency and phase as the reference clock and aligned to within ±100 ps.

For digitized clocking applications, the recommendation is to operate the AD9546 in SPI mode, rather than I^2C mode, because SPI supports a much higher serial transfer rate than I^2C (50 MHz for SPI vs. 400 kHz for I^2C).

A key aspect of any synchronized clocking system is the timing error caused by the propagation delay of clock signals sent between the master and slave nodes. To properly align the common time scale across all nodes, the system must measure and compensate for the propagation delay between nodes. The recommended method for quantifying the propagation delay is measuring round trip delay in real time, which allows continuous assessment and correction of delay that may arise from temperature variations in the system. The AD9546 readily accommodates assessment of round trip delay via its analog loopback feature (see the Analog Clock Loopback section) and time skew measurement processor (see the application example in Figure 129 and the Clock Propagation Delay Measurement section).

An example of a digitized clocking system appears in Figure 128, which includes round trip delay measurement capability. The example system comprises a timing card and a pair of line cards interconnected via a backplane. The timing card and line cards rely on digitized clocking to transport clock signals as time codes via the AD9546 SPI port. Although not explicitly shown in Figure 128, the AD9546 analog loopback feature for round trip delay measurement requires the use of REFB or REFBB as the loopback input and the M4 pin as the loopback return.

The timing card employs an IEEE 1588 servo for transporting timing over ethernet (the traffic labels in the diagram). The servo controls a PLL with the output of the PLL providing the common clock for the system. The PLL produces three identical clock signals with coincident output clock edges that serve as the common clock input to each of the cards. This arrangement, in combination with the AD9546 analog loopback capability, allows the timing card to measure the propagation delay between the timing card and each line card. The measured propagation delay can then be compensated via offsets applied to the digitized clocking paths.

The line cards handle Ethernet traffic via a physical layer (PHY) with each line card serving multiple PHYs. Figure 128 shows only one PHY with a connection to a UTS/IUTS pair on the AD9546 for simplification. However, each PHY has access to a dedicated UTS/IUTS pair on the AD9546. In this way, any one of the PHYs can provide a received clock to the AD9546 for digitization via a UTS. The digitized received clock is transported over the digital bus to the timing card, where the timing card uses the digitized receive clock as a SyncE clock for the IEEE 1588 servo. The timing card, in turn, generates a digitized clock for transport back to the line cards over the digital bus. The line cards convert the digitized clock back to an analog clock for use by the PHYs for clocking outgoing traffic.

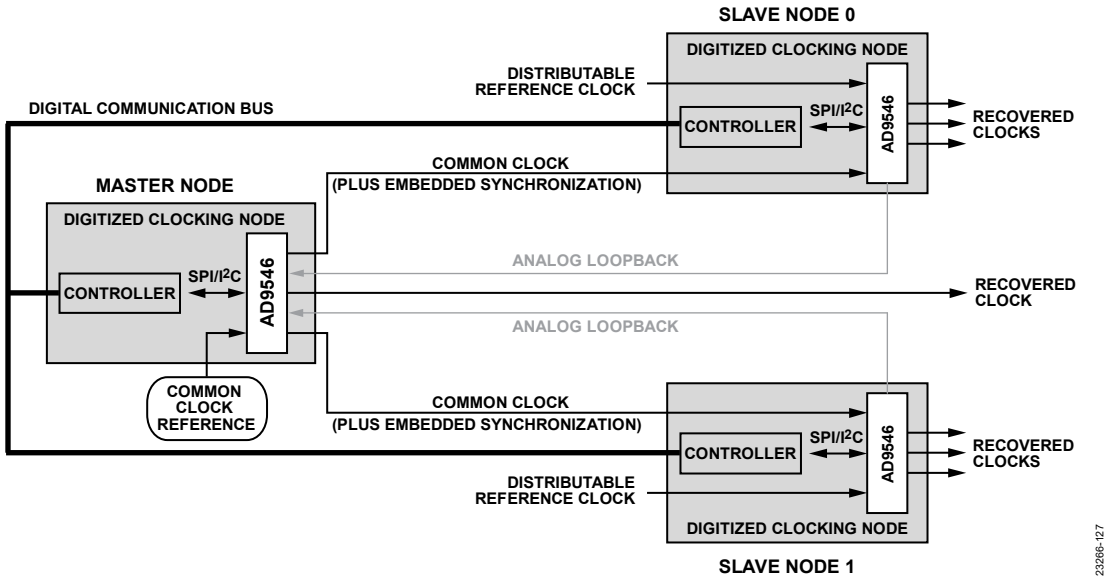


Figure 127. Digitized Clocking Application

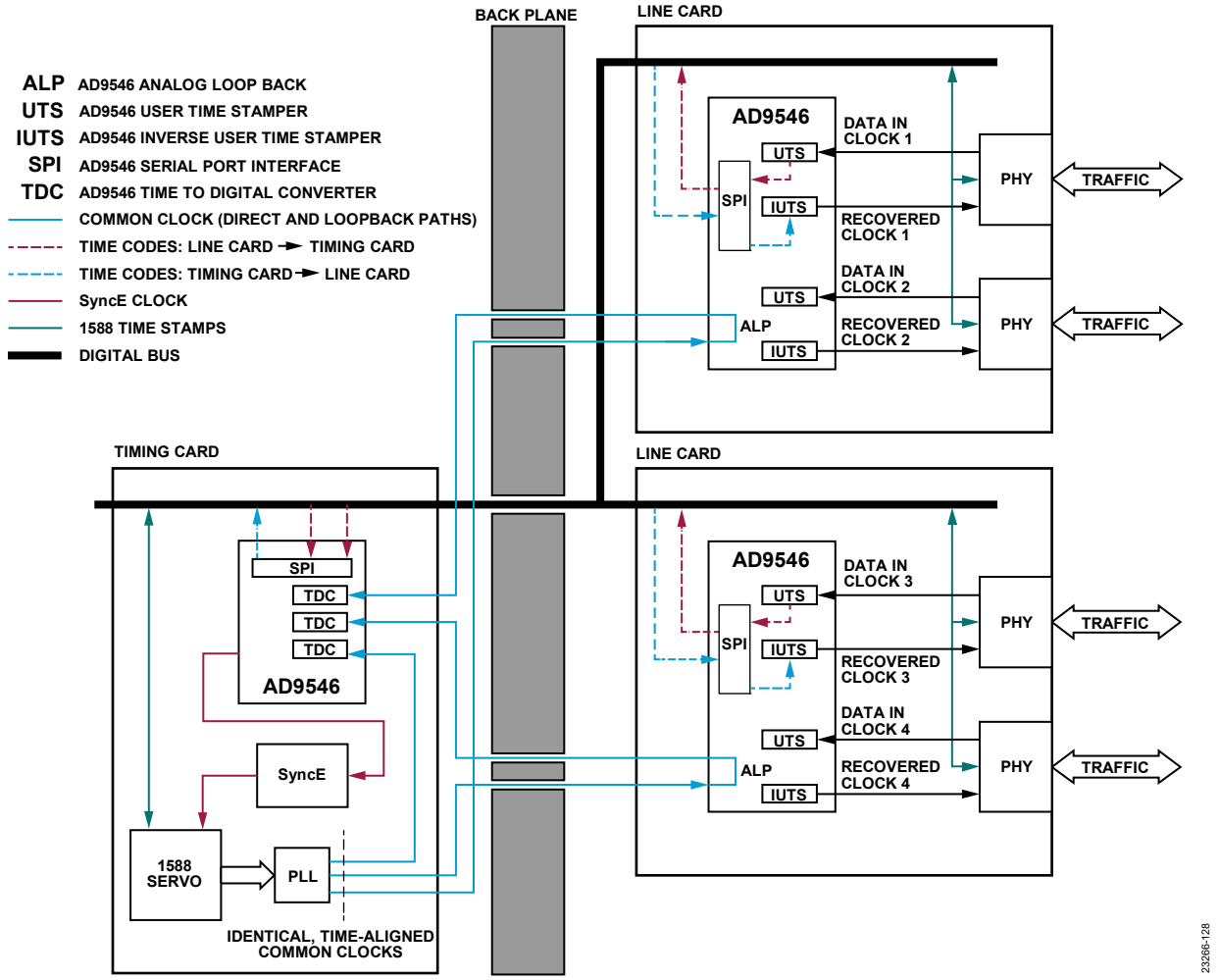


Figure 128. Digitized Clocking System Example

CLOCK PROPAGATION DELAY MEASUREMENT

Consider an application comprising a clock source (in) driving additional components that introduce propagation delay (t_P) to the final output (out) signal (see Figure 129). The additional components can be, for example, a combination of physical interconnects (cables and PCB traces), clock buffers or other elements that introduce propagation delay.

In applications requiring quantification of the propagation delay between two clock signals, the TDCs in the AD9546 provide a means to measure the delay and make the result available in digital form via serial port registers. Figure 129 shows a simplified propagation delay measurement setup using the AD9546.

To measure propagation delay, the user connects the in and out signals to any two of the eight independent reference clock inputs of the AD9546. The reference clock inputs comprise the REFx pins or one of the Mx control pins configured as an input to one of the four auxiliary references (auxiliary REFx). Then, using the skew measurement processor (see the Timing Skew Measurements Using Two TDCs section), the user can read the measurement results via the serial port, which indicates the value of t_P with approximately 1 ps resolution.

The measured value of t_P can then be used by the controller to adjust the timing of the clock source to compensate for the delay through the clock components. By periodically measuring t_P and compensating accordingly, the user can correct for time dependent changes in t_P that may arise from temperature variations.

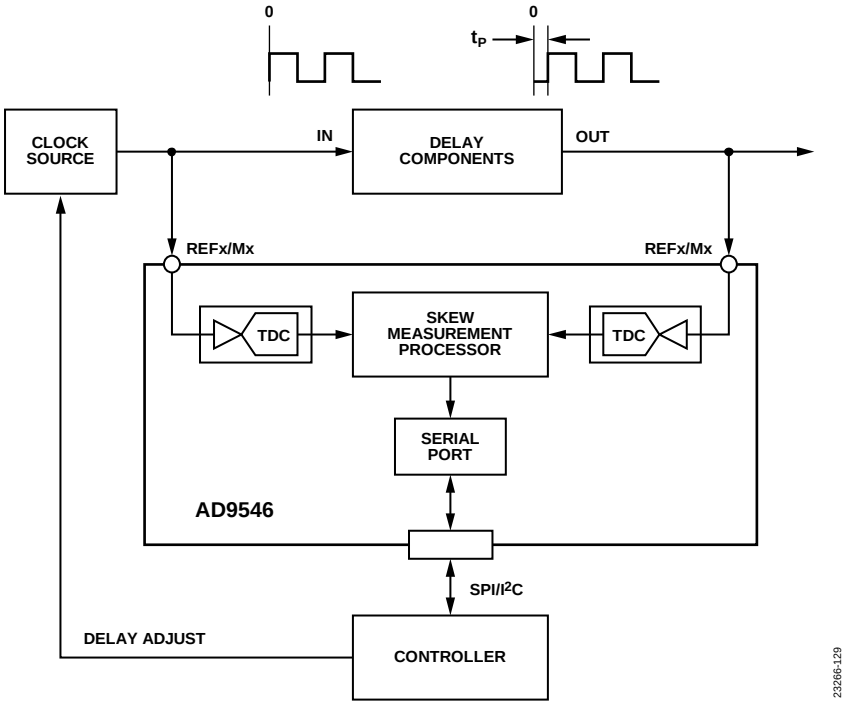


Figure 129. Propagation Delay Measurement Application

TIME STAMP MEASUREMENT

The AD9546 is particularly well suited for applications requiring the measurement of clock frequencies and clock timing offsets. Measurement of frequency and time offset is made possible by digitizing (time stamping) input clock edges using TDCs. Each of the four primary reference inputs (REFA, REFAA, REFB, REFBB) and four auxiliary reference inputs (auxiliary REF0, auxiliary REF1, auxiliary REF2, and auxiliary REF3) of the AD9546 has a dedicated TDC (see Figure 130).

The AD9546 embodies a user time stamp capturing system that gives the user access to time stamps generated by the TDCs (see the User Time Stamper (UTS) section). The UTS system embodies nine time stamp capturing units (UTS 0 to UTS 8), making it possible to capture time stamps from all eight input time stamp sources (REFx and auxiliary REFx) concurrently.

The UTS system stores captured time stamps in the UTS FIFO, which the user accesses via the serial port (see the UTS Readback FIFO section). The FIFO has limited capacity, which means if the time stamp arrival rate to the FIFO exceeds the FIFO extraction rate (on average), the FIFO eventually overflows. Because a FIFO overflow causes the FIFO to stop accepting time stamps from the UTSs, the user must extract the FIFO contents at an average rate that exceeds the average time stamp arrival rate to the FIFO (otherwise, time stamps are lost). Thus, the serial port read rate puts an upper limit on the time stamp arrival rate to the FIFO.

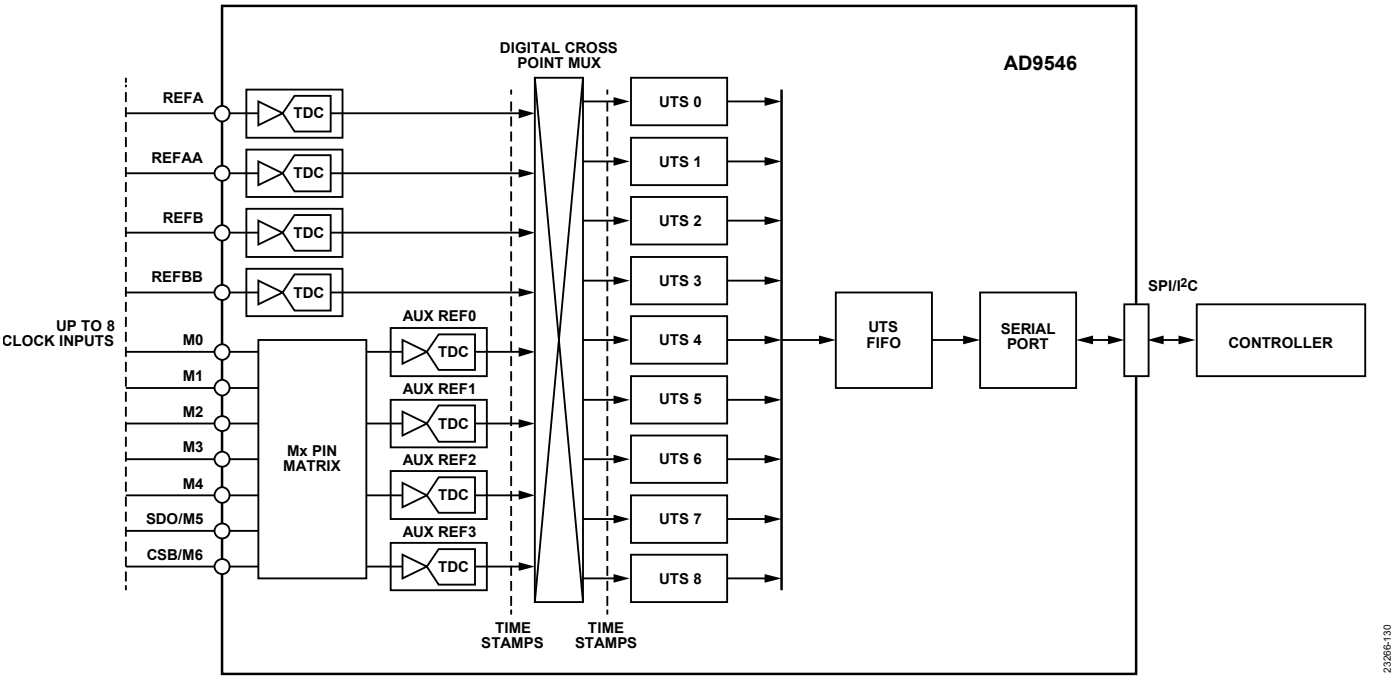


Figure 130. Time Stamp Measurement Application

IEEE 1588 SERVO

The AD9546 provides the jitter cleaning, phase shifting, and synchronization necessary for an IEEE 1588 servo and software stack in a Telecom-BC (T-BC) with hybrid support application (see Figure 131). Figure 131 represents a superset of possible connection scenarios for implementing a variety of IEEE 1588 applications rather than a specific IEEE 1588 application. The diagram in Figure 131 supports IEEE 1588 master and slave clock operation. The dashed box in Figure 131 constitutes external circuitry that handles precision time protocol (PTP) packets, software to implement IEEE 1588, and time of day support. The integrated features of the AD9546 simplify the task of synchronizing clocks via PTP. Furthermore, the Mx pins of the AD9546 provide the option to implement assisted partial timing support (APTS) via a GPS receiver providing a precision 1 pps source.

Figure 131 implies the use of the AD9546 system clock compensation feature (see the Compensation Method 3 section in the System Clock Compensation section). Specifically, the reference input serves as a frequency stable source to the system clock compensation block (for example, an OXCO or TCXO). The frequency stable source allows the AD9546 to use a crystal resonator as the system clock source (XOA and XOB pins) and compensate for the inherent crystal frequency instability, thereby maintaining the overall timing precision of the system.

PLL0 uses the 125 MHz SyncE clock as a reference input to generate two filtered, 125 MHz SyncE clocks. PLL1 uses one of the auxiliary NCOs of the AD9546 as a reference input to generate two arbitrary output clocks with stability commensurate with the REF IN source. The system clock compensation system ensures frequency stability of the NCOs associated with the DPLLs and auxiliary NCOs.

The performance and feature set of the AD9546 make it ideal for enabling the latest IEEE 1588 features, as well as the latest ITU-T packet synchronization related standards for 4G and 5G wireless networks.

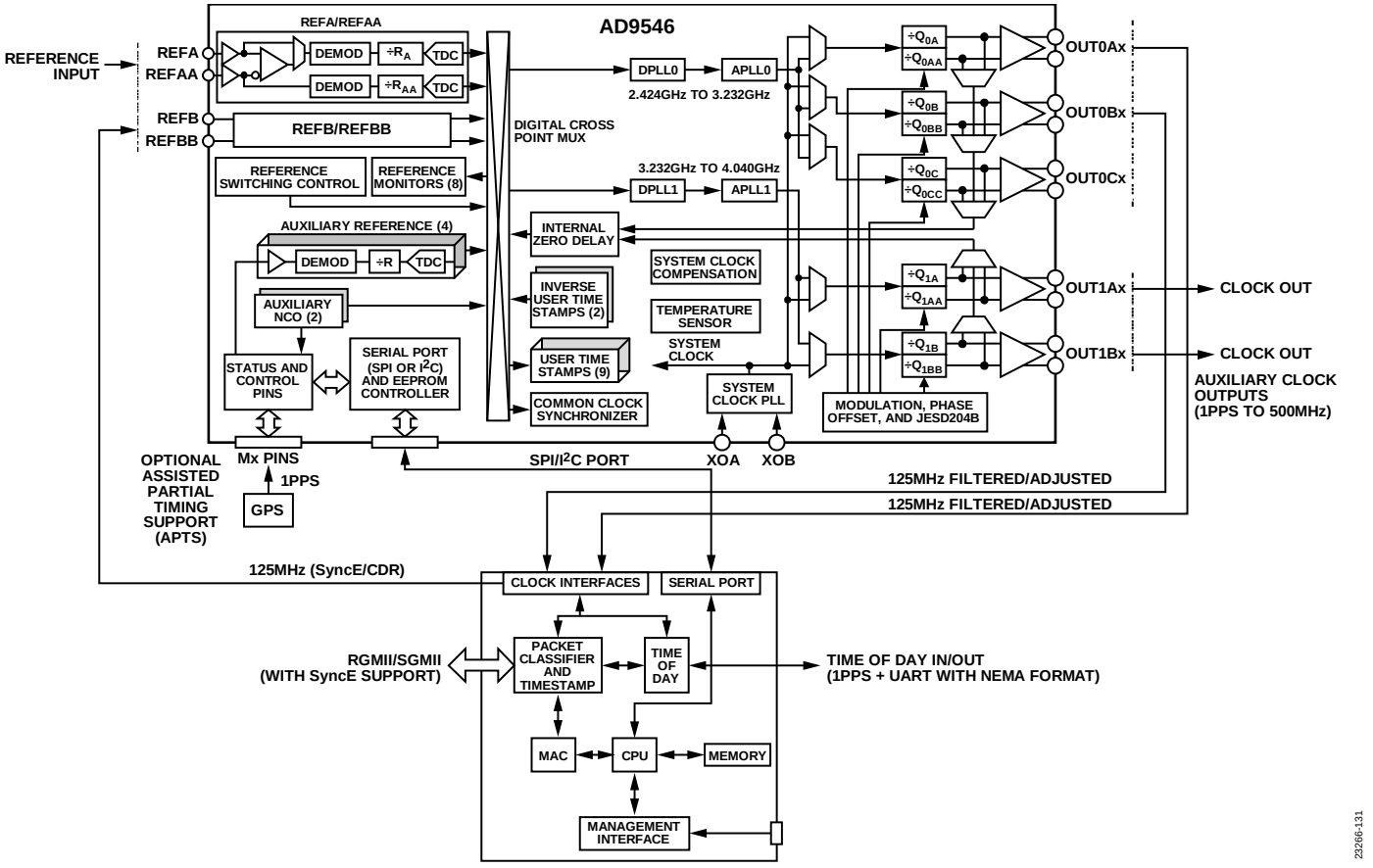


Figure 131. Using an AD9546 in an IEEE 1588 Version 2 Application

INITIALIZATION SEQUENCE

The suggested initialization sequence for powering on and programming the AD9546 is shown in Figure 132 to Figure 134.

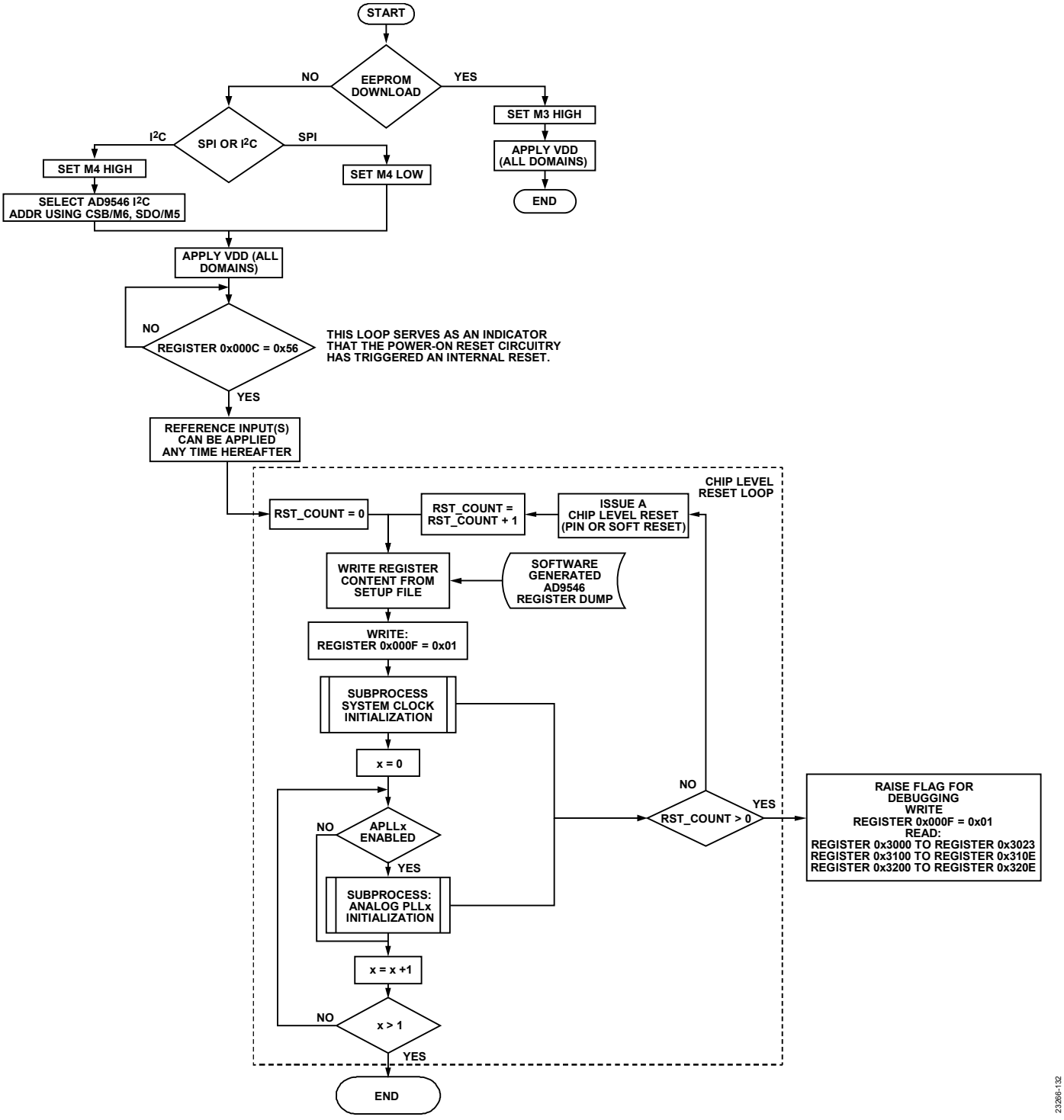


Figure 132. Programming Sequence Flowchart

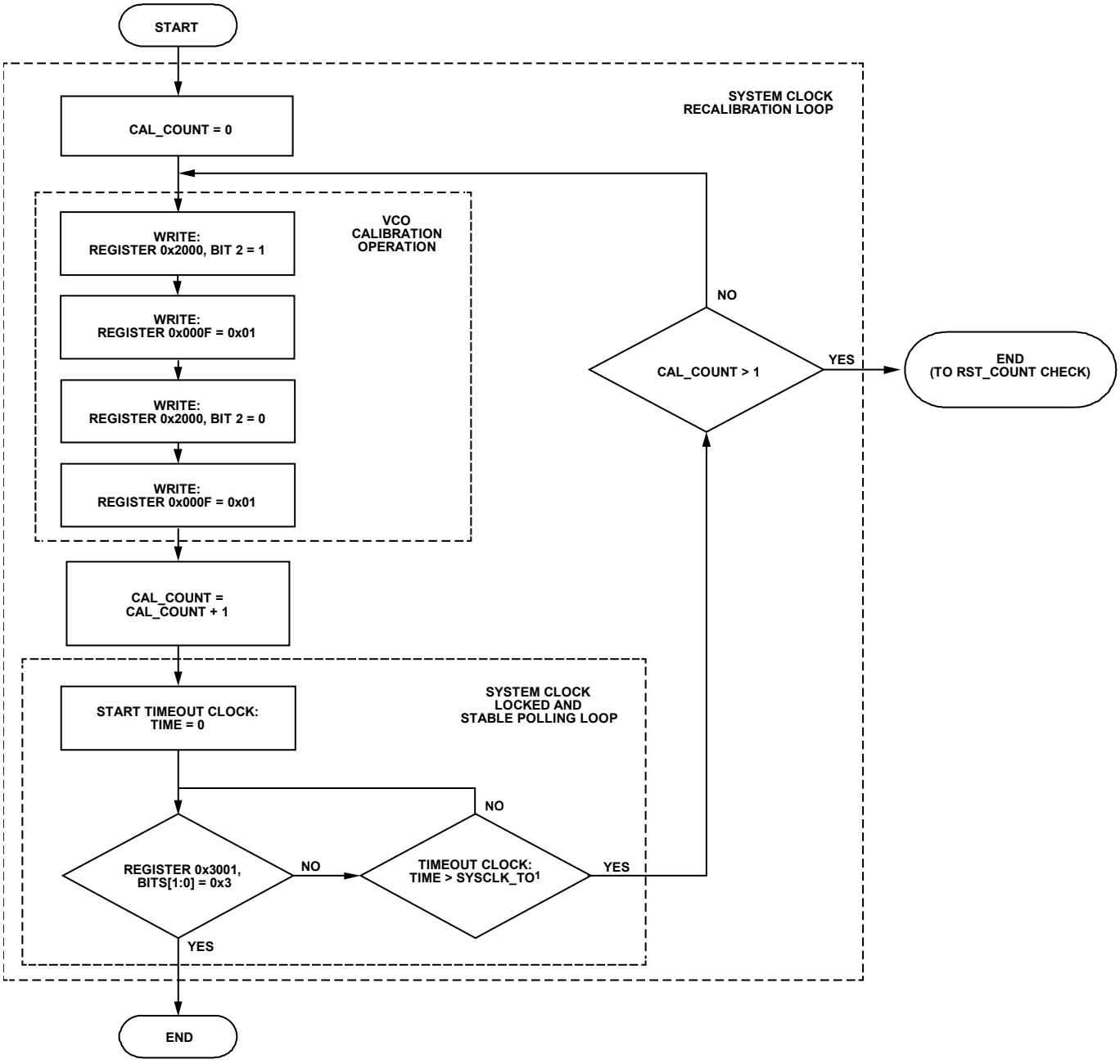
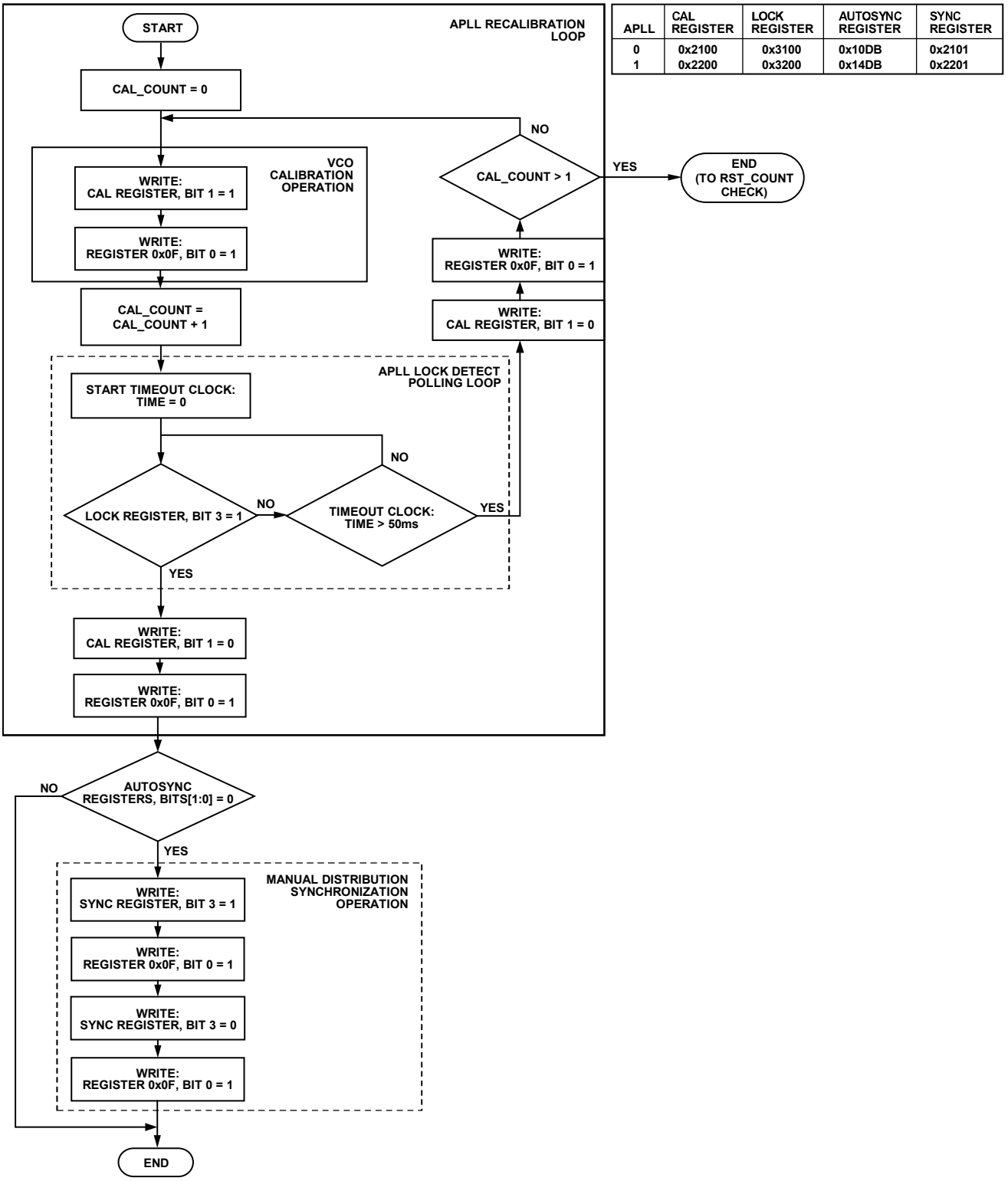


Figure 133. System Clock Initialization Subprocess



APLL	CAL REGISTER	LOCK REGISTER	AUTOSYNC REGISTER	SYNC REGISTER
0	0x2100	0x3100	0x10DB	0x2101
1	0x2200	0x3200	0x14DB	0x2201

23266-134

Figure 134. APLL Initialization Subprocess

SERIAL CONTROL PORT

SERIAL CONTROL PORT OVERVIEW

The AD9546 serial control port is a flexible, synchronous serial communications port that provides a convenient interface to many industry-standard microcontrollers and microprocessors. The AD9546 serial control port is compatible with most synchronous transfer formats, including I²C, Motorola SPI, and Intel SSR protocols. The serial control port allows read/write access to the AD9546 register map.

The AD9546 uses the Analog Devices unified SPI protocol (see the Analog Devices Serial Control Interface Standard). The unified SPI protocol guarantees that all new Analog Devices products using the unified protocol have consistent serial port characteristics. The SPI port configuration is programmable via Register 0x0000.

Unified SPI differs from the SPI port found on older Analog Devices products, such as the AD9557 and AD9558, in the following ways:

- Unified SPI does not have byte counts. A transfer is terminated when the CSB signal goes high. The W1 and W0 bits in the traditional SPI become the A12 and A13 bits of the register address (see Table 113). This format is much like streaming mode in the traditional SPI.
- The address ascension bit (Register 0x0000) controls whether register addresses are automatically incremented or decremented regardless of the LSB/MSB first setting. In a traditional SPI, LSB first mode dictated auto-incrementing and MSB first mode dictated auto-decrementing of the register address.
- The first 16 register addresses of devices that adhere to the unified serial port have a consistent structure.

SPI/I²C PORT SELECTION

Although the AD9546 supports both the SPI and I²C serial port protocols, only one is active following power-up (as determined by the M4 pin during the start-up sequence). The only way to change the serial port protocol is to reset (or power cycle) the device. See Table 103 for the I²C address assignments.

SPI SERIAL PORT OPERATION

Pin Descriptions

The serial clock (SCLK) signal serves as the serial shift clock. This pin is an input. SCLK synchronizes serial control port read and write operations. The rising edge SCLK registers write data bits, and the falling edge registers read data bits. The SCLK line supports a maximum clock rate of 50 MHz.

The user can configure the SPI port hardware configuration and data format via the Register 0x0000. The hardware configurations are 3-wire bidirectional (default) or 4-wire unidirectional mode and the data formats are MSB first (default) or LSB first. The 3-wire mode uses the serial data input/output (SDIO) signal for transferring data in both directions. The 4-wire mode uses SDIO for transferring data to the AD9546, and SDO for transferring data from the AD9546. The output drive strength of the SDO and SDIO pins is selectable via Bit 7 of Register 0x0109.

The chip select (CSB) signal is an active low control that gates read and write operations. Assertion (active low) of CSB initiates a write or read operation to the AD9546 SPI port. The user can transfer any number of data bytes in a continuous stream. The register address is automatically incremented or decremented based on the setting of the address ascension bit (Register 0x0000). The user must deassert CSB following the last byte transferred, thereby ending the stream mode. CSB is internally connected to a 10 kΩ pull-up resistor. When CSB is high, SDIO and SDO enter a high impedance state.

Implementation Specific Details

The Analog Devices Serial Control Interface Standard provides a detailed description of the unified SPI protocol and covers items such as timing, command format, and addressing. The unified SPI protocol defines the following device specific items:

- Analog Devices unified SPI protocol revision: 1.0
- Chip type: 0x5
- Product ID: 0x0121
- Physical layer: 3-wire and 4-wire supported and 1.5 V, 1.8 V, and 2.5 V operation supported
- Optional single-byte instruction mode: not supported
- Data link: not used
- Control: not used

Communication Cycle—Instruction Plus Data

The unified SPI protocol consists of a two-part communication cycle. The first part is a 16-bit instruction word coincident with the first 16 SCLK rising edges. The second part is the payload, the bits of which are coincident with SCLK rising edges. The instruction word provides the AD9546 serial control port with information regarding the payload. The instruction word includes the R/$\overline{W}$ bit that indicates the direction of the payload transfer (that is, a read or write operation). The instruction word also indicates the starting register address of the first payload byte.

IO Update

In general, there are two sets of serial port registers: buffer registers and active registers. When the SPI port receives data, it deserializes the data, which collects in the buffer registers. In most cases, the output of the buffer registers route to the input of the active registers rather than directly to the intended circuit blocks within the device. As such, when the user writes SPI data, the intended functional blocks within the device are unaffected by the newly written data.

To make the buffer register data effective, the user must perform an IO update operation, which transfers the contents of all buffer registers to the active registers. The output of the active registers causes the desired effect upon the functional blocks within the device.

There are two methods available for issuing an IO update operation:

1. Writing Logic 1 to Bit 0 of Register 0x000F
2. Configuring an Mx pin as a physical IO update pin

In the case of the first method, Bit 0 is an autoclearing bit. In the case of the second method, applying Logic 1 to the configured Mx pin asserts the IO update operation.

The user can program as many register bits as desired before executing the IO update operation. Then, upon assertion of the IO update operation, the all the buffer register contents transfer to their active register counterparts.

Write

When the instruction word indicates a write operation, the payload is written into the serial control port buffer of the AD9546. Data bits are registered on the rising edge of SCLK. Generally, it does not matter what data is written to blank registers. However, it is customary to use 0s. The user must verify that all reserved registers within a specific range have a default value of 0x00. However, Analog Devices makes every effort to avoid having reserved registers with nonzero default values.

Read

If the instruction word indicates a read operation, the next N × 8 SCLK cycles clock out the data starting from the address specified in the instruction word, where N is the number of data bytes to read. Read data appears on the appropriate data pin (SDIO or SDO) on the falling edge of SCLK. The user must latch the read data on the rising edge of SCLK. The internal SPI control logic does not skip over blank registers during a readback operation.

A readback operation takes data from either the serial control port buffer registers or the active registers, as determined by Register 0x0001, Bit 5.

SPI Instruction Word (16 Bits)

The MSB of the 16-bit instruction word is R/$\overline{W}$, which indicates whether the ensuing operation is read or write. The next 15 bits are the register address (A14 to A0), which indicates the starting register address of the read/write operation (see Table 113). The SPI controller ignores A14, treating it as Logic 0, because the AD9546 has no register addresses requiring more than a 14-bit address word.

SPI MSB First and LSB First Transfers

The AD9546 instruction word and payload can be transferred MSB first or LSB first. The default for the AD9546 is MSB first. To invoke LSB first mode, write a Logic 1 to Register 0x0000, Bit 6. Immediately after invoking LSB first mode, subsequent serial control port operations are LSB first.

Address Ascension

If the address ascension bit (Register 0x0000, Bit 5) is Logic 0, serial control port register addressing decrements from the specified starting address toward Address 0x0000. If the address ascension bit (Register 0x0000, Bit 5) is Logic 1, serial control port register addressing increments from the starting address toward Address 0x3A3B. Reserved addresses are not skipped during multibyte input/output operations. Therefore, write the default value to a reserved register and Logic 0s to unmapped registers. It is more efficient to issue a new write command than to write the default value to more than two consecutive reserved (or unmapped) registers.

Table 112. Streaming Mode (No Addresses Skipped)

Address Ascension	Stop Sequence
Increment	0x0000 to 0x3A3B
Decrement	0x3A3B to 0x0000

Table 113. Serial Control Port, 16-Bit Instruction Word

MSB															LSB
I15	**I14**	**I13**	**I12**	**I11**	**I10**	**I9**	**I8**	**I7**	**I6**	**I5**	**I4**	**I3**	**I2**	**I1**	**I0**
R/$\overline{W}$	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

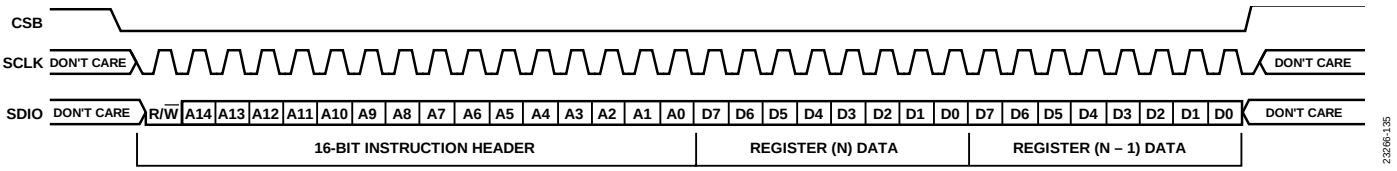


Figure 135. Serial Control Port Write—MSB First, Address Decrement, Two Bytes of Data

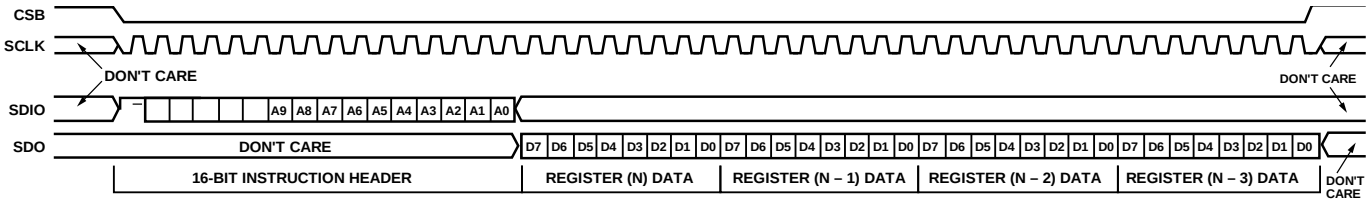


Figure 136. Serial Control Port Read—MSB First, Address Decrement, Four Bytes of Data

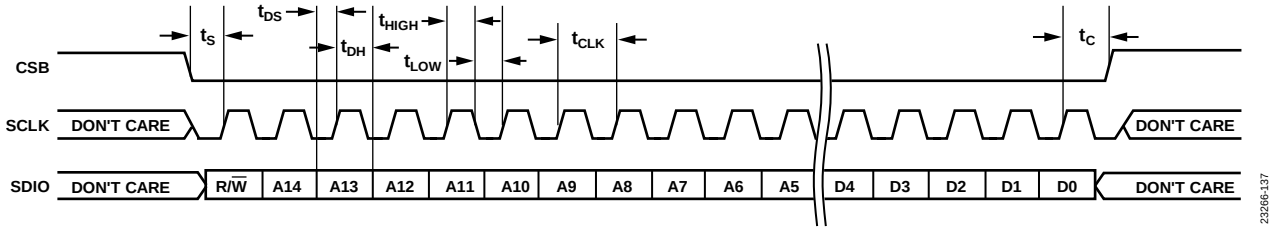


Figure 137. Timing Diagram for Serial Control Port Write—MSB First

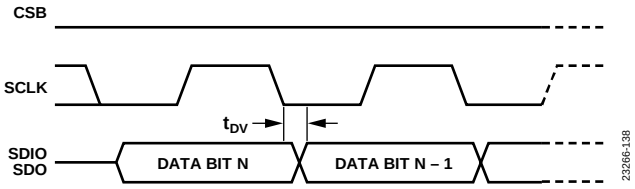


Figure 138. Timing Diagram for Serial Control Port Register Read—MSB First

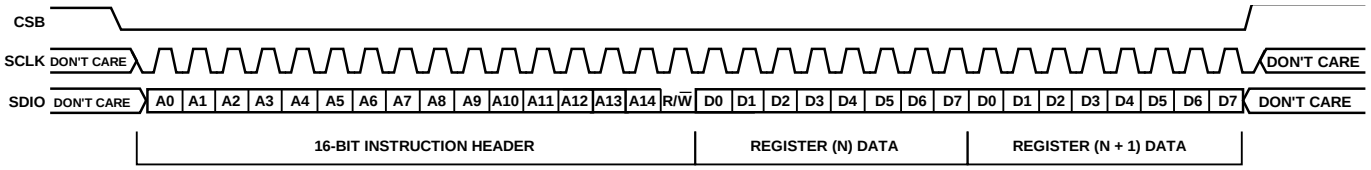


Figure 139. Serial Control Port Write—LSB First, Address Increment, Two Bytes of Data

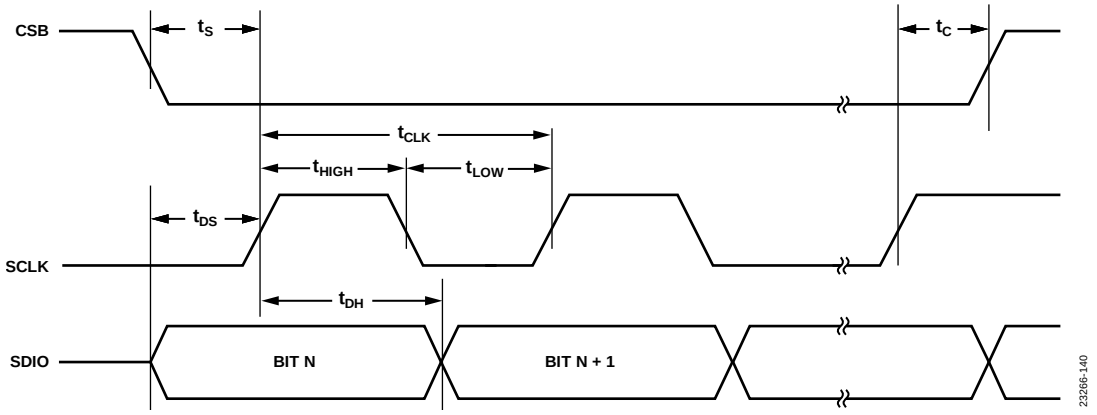


Figure 140. Serial Control Port Timing—Write

Table 114. Serial Control Port Timing

Parameter	Description
t_{DS}	Setup time between data and the rising edge of SCLK
t_{DH}	Hold time between data and the rising edge of SCLK
t_{CLK}	Period of the clock
t_S	Setup time between the CSB falling edge and the SCLK rising edge (start of the communication cycle)
t_C	Setup time between the SCLK rising edge and CSB rising edge (end of the communication cycle)
t_{HIGH}	Minimum period that SCLK is in a logic high state
t_{LOW}	Minimum period that SCLK is in a logic low state
t_{DV}	SCLK to valid SDIO (see Figure 138)

I²C SERIAL PORT OPERATION

The I²C interface is popular because it requires only two pins and easily supports multiple devices on the same bus. Its main disadvantage is its maximum programming speed of 400 kbps. The AD9546 I²C port supports the 400 kHz fast mode as well as the 100 kHz standard mode.

To support 1.5 V, 1.8 V, and 2.5 V I²C operation, the AD9546 does not strictly adhere to every requirement in the original I²C specification. For instance, it does not support specifications such as slew rate limiting and glitch filtering. Therefore, the AD9546 is I²C compatible, but not necessarily fully I²C compliant.

The AD9546 I²C port consists of a serial data line (SDA) and a serial clock line (SCL). In an I²C bus system, the AD9546 connects to the serial bus (data bus SDA and clock bus SCL) as a slave device. That is, the AD9546 does not generate an I²C clock. The AD9546 uses direct 16-bit memory addressing rather than 8-bit memory addressing, which is more common.

The AD9546 allows up to four unique slave devices to occupy the I²C bus via a 7-bit slave address transmitted as part of an I²C packet. Only the device with a matching slave address responds to subsequent I²C commands. Table 103 lists the supported device slave addresses.

I²C Bus Characteristics

Table 115 shows a summary of the various I²C abbreviations.

Table 115. I²C Bus Abbreviation Definitions

Abbreviation	Definition
S	Start
Sr	Repeated start
P	Stop
A	Acknowledge
$\overline{A}$	Nonacknowledge
$\overline{W}$	Write
R	Read

Figure 141 shows an example of valid data transfer. One clock pulse is required for each data bit transferred. The data on the SDA line must be stable during the high period of the clock. The high or low state of the data line can change only when the clock signal on the SCL line is low.

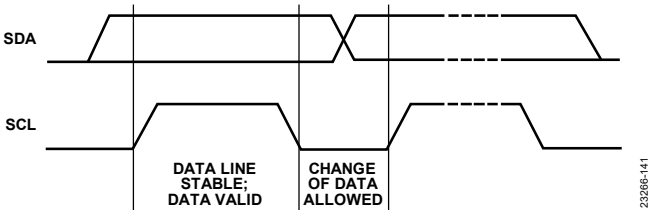


Figure 141. Valid Bit Transfer

Figure 142 shows start and stop functionality. The start condition is a high to low transition on the SDA line while SCL is high. The master always generates the start condition to initialize a data transfer. The stop condition is a low to high transition on the SDA line while SCL is high.

The master always generates the stop condition to terminate a data transfer. The SDA line must always transfer eight bits (one byte). Each byte must be followed by an acknowledge bit. Bytes are sent MSB first.

The acknowledge bit (A) is the ninth bit attached to any 8-bit data byte. An acknowledge bit is always generated by the receiver to inform the transmitter that the byte has been received. Acknowledgement consists of pulling the SDA line low during the ninth clock pulse after each 8-bit data byte.

The nonacknowledge bit ($\overline{A}$) is the ninth bit attached to any 8-bit data byte. A nonacknowledge bit is always generated by the receiver to inform the transmitter that the byte has not been received. Nonacknowledgement consists of leaving the SDA line high during the ninth clock pulse after each 8-bit data byte. After issuing a nonacknowledge bit, the AD9546 I²C state machine goes into an idle state.

Data Transfer Process

The master initiates a data transfer by asserting a start condition, which indicates that a data stream follows. All I²C slave devices connected to the serial bus respond to the start condition.

The master then sends an 8-bit address byte over the SDA line, consisting of a 7-bit slave address (MSB first) plus an R/$\overline{W}$ bit. This bit determines the direction of the data transfer, that is, whether data is written to or read from the slave device (Logic 0 indicates write, and Logic 1 indicates read).

The peripheral whose address corresponds to the transmitted address responds by sending an acknowledge bit. All other devices on the bus remain idle while the selected device waits for data to be read from or written to it. If the R/$\overline{W}$ bit is Logic 0, the master (transmitter) writes to the slave device (receiver). If the R/$\overline{W}$ bit is Logic 1, the master (receiver) reads from the slave device (transmitter). The read and write formats for these commands appear in the Data Transfer Format section.

Data is then sent over the serial bus in the format of nine clock pulses, one data byte (eight bits) from either master (write mode) or slave (read mode), followed by an acknowledge bit from the receiving device. The protocol allows a data transfer to consist of any number of bytes (that is, the payload size is unrestricted). In write mode, the first two data bytes immediately after the slave address byte are the internal memory (control registers) address bytes (the higher address byte first). This addressing scheme gives a memory address of up to $2^{16} - 1 = 65,535$. The data bytes after these two memory address bytes are register data written to or read from the control registers. In read mode, the data bytes following the slave address byte consist of register data written to or read from the control registers.

When all the data bytes are read or written, stop conditions are established. In write mode, the master device (transmitter) asserts a stop condition to end the data transfer during the clock pulse following the acknowledge bit for the last data byte from the slave device (receiver).

In read mode, the master device (receiver) receives the last data byte from the slave device (transmitter) but does not pull SDA low during the ninth clock pulse (a nonacknowledge bit). By receiving the nonacknowledge bit, the slave device knows that the data transfer is finished and enters idle mode.

The master device then takes the data line low during the low period before the 10th clock pulse, and high during the 10th clock pulse to assert a stop condition.

A start condition can be used instead of a stop condition. Furthermore, a start or stop condition can occur at any time, and partially transferred bytes are discarded.

Data Transfer Format

The write byte format is used to write a register address to the RAM starting from the specified RAM address (see Table 116). The send byte format is used to set up the register address for subsequent reads (see Table 117). The receive byte format is used to read the data byte(s) from RAM starting from the current address (see Table 118). The read byte format is the combined format of the send byte and the receive byte (see Table 119).

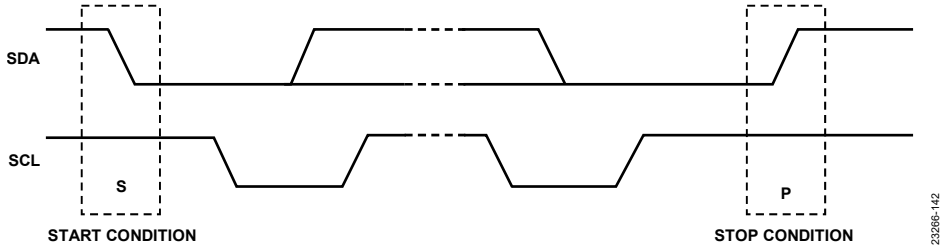


Figure 142. Start and Stop Conditions

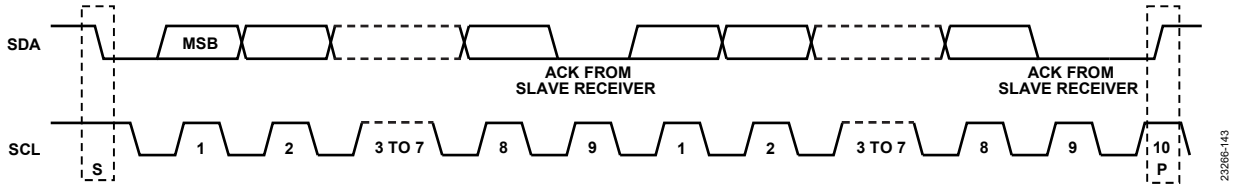


Figure 143. Acknowledge Bit

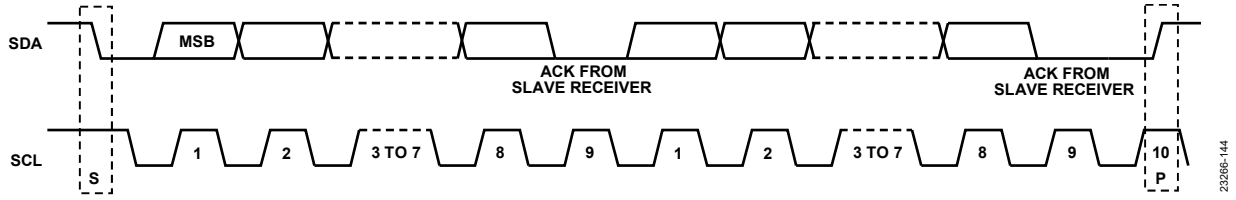


Figure 144. Data Transfer Process (Master Write Mode, 2-Byte Transfer)

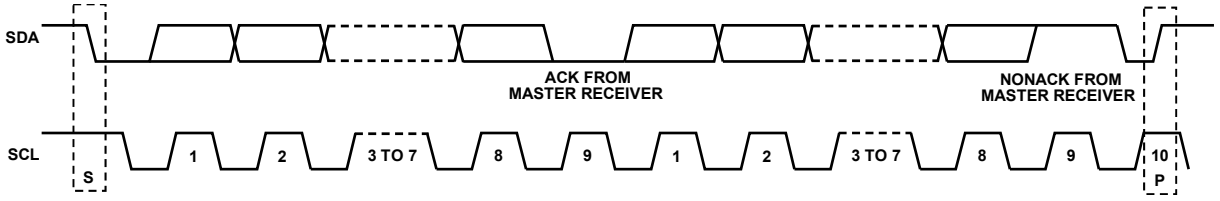


Figure 145. Data Transfer Process (Master Read Mode, 2-Byte Transfer), First Acknowledge from Slave

Table 116. Write Byte Format

S	Slave address	$\overline{W}$	A	RAM address high byte	A	RAM address low byte	A	RAM Data 0	A	RAM Data 1	A	RAM Data 2	A	P

Table 117. Send Byte Format

S	Slave address	$\overline{W}$	A	RAM address high byte	A	RAM address low byte	A	P

Table 118. Receive Byte Format

S	Slave address	R	A	RAM Data 0	A	RAM Data 1	A	RAM Data 2	$\overline{A}$	P

Table 119. Read Byte Format

S	Slave address	$\overline{W}$	A	RAM address high byte	A	RAM address low byte	A	Sr	Slave address	R	A	RAM Data 0	A	RAM Data 1	A	RAM Data 2	$\overline{A}$	P

I²C Serial Port Timing

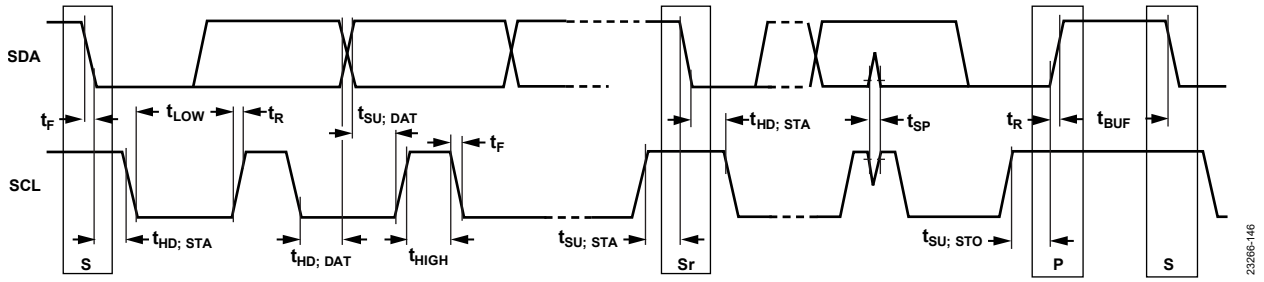


Figure 146. I²C Serial Port Timing

Table 120. I²C Timing Definitions

Parameter	Description
t_{BUF}	Bus free time between stop and start conditions
$t_{HD;\ STA}$	Repeated hold time start condition
$t_{SU;\ STA}$	Repeated start condition setup time
$t_{SU;\ STO}$	Stop condition setup time
$t_{HD;\ DAT}$	Data hold time
$t_{SU;\ DAT}$	Data setup time
t_{LOW}	SCL clock low period
t_{HIGH}	SCL clock high period
t_R	Minimum/maximum receive SCL and SDA rise time
t_F	Minimum/maximum receive SCL and SDA fall time
t_{SP}	Pulse width of voltage spikes that must be suppressed by the input filter

OUTLINE DIMENSIONS

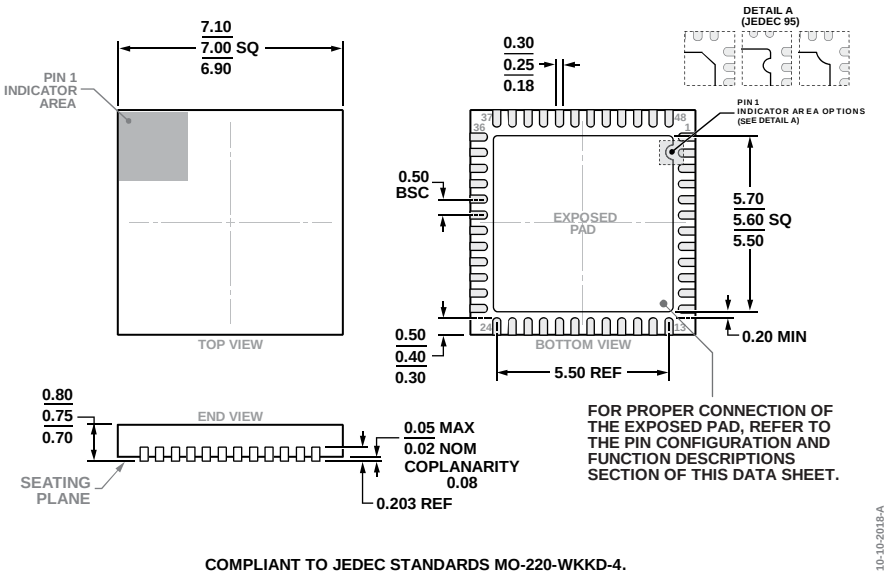


Figure 147. 48-Lead Lead Frame Chip Scale Package [LFCSP]
7 mm × 7 mm Body and 0.75 mm Package Height
(CP-48-13)
Dimensions shown in millimeters

ORDERING GUIDE

Model[1]	Temperature Range	Package Description	Package Option
AD9546BCPZ	−40°C to +85°C	48-Lead Lead Frame Chip Scale Package [LFCSP)	CP-48-13
AD9546BCPZ-REEL7	−40°C to +85°C	48-Lead Lead Frame Chip Scale Package [LFCSP]	CP-48-13
AD9546/PCBZ	−40°C to +85°C	Evaluation Board	

[1] Z = RoHS Compliant Part.

I²C refers to a communications protocol originally developed by Philips Semiconductors (now NXP Semiconductors).

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Analog Devices Inc.:

AD9546BCPZ AD9546BCPZ-REEL7