Quad Input, 10-Output, Dual DPLL/IEEE 1588 1 pps Synchronizer and Jitter Cleaner

Data Sheet

AD9545

FEATURES

- 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: 10^{-4} Hz to 1850 Hz
- 2 independent, programmable auxiliary NCOs (1 Hz to 65,535 Hz, resolution < 1.37×10^{-12} Hz), 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 4 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

- Global positioning system (GPS), 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, including 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
- Cable infrastructures
- Carrier Ethernet

GENERAL DESCRIPTION

The AD9545 supports existing and emerging International Telecommunications Union (ITU) standards for the delivery of frequency, phase, and time of day over service provider packet networks, including ITU-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 AD9545 are synchronized to any one of up to four input references. The digital phase-locked loops (DPLLs) reduce timing jitter associated with the external references. The digitally controlled loop and holdover circuitry continuously generate a low jitter output signal, even when all reference inputs fail.

The AD9545 is available in a 48-lead LFCSP (7 mm × 7 mm) package and operates over the −40°C to +85°C temperature range.

Note that throughout this data sheet, multifunction pins, such as SDO/M5, are referred to either by the entire pin name or by a single function of the pin, for example, M5, when only that function is relevant.

Rev. A
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

4/2018—Rev. 0 to Rev. A

Changes to Features Section, Applications Section, and General Description Section 1
Changes to Figure 1 7
Changes to Table 3 8
Changes to Table 4 10
Changes to Table 5 11
Changes to Table 6 12
Change to Table 7 13
Change to Table 9 14
Change to Table 10 15
Changes to Table 12 16
Changes to Table 18 17
Changes to Table 19 18
Changes to Table 20 19
Changes to Table 21 and Table 22 20
Changes to Table 23 21
Changes to Thermal Resistance Section and Table 25 24
Changes to Table 26 25
Changes to Typical Performance Characteristics Section 27
Changes to Terminology Section 31
Moved Theory of Operation Section 32
Changes to Theory of Operation Section 32
Moved Input/Output Termination Recommendations Section 33
Changes to Input/Output Termination Recommendations Section 33
Added Figure 39 and Figure 40; Renumbered Sequentially 35
Added System Clock PLL Overview Heading and Figure 42 36
Changes to System Clock Input Frequency Declaration Section, System Clock Source Section, Crystal Path Section, Direct Path Section 36
Added System Clock Calibration Section 37
Changes to 2× Frequency Multiplyer Section, Prescale Divider Section, Feedback Divider Section, and System Clock PLL Lock Detector Section 37
Deleted System Clock Input Termination Recommendations Section 37
Added System Clock Stability Compensation Section 38
Added Reference Clock Input Recievers Section, Table 27, and Table 28; Renumbered Sequentially 39
Added Reference Dividers (R-Dividers) Section 41
Added Reference Monitor Section and Figure 43 42
Added Table 29 and Figure 44 43
Added Table 30 44
Added Reference Demodulator Section and Figure 45 47
Added Figure 46 48
Added Distribution Clock Output Drivers Section, Table 31, Table 32, and Table 33 50
Added Table 34 52
Added Distribution Dividers (Q Dividers) Section 53
Added Figure 47 54
Added Distribution Phase Offset Control Section 56
Added Table 35 57
Added Distribution N-Shot/PRBS Output Clocking Section and Figure 48 59
Added Table 36 and Figure 49 60
Added Figure 50, Figure 51, and Figure 52 61
Added Figure 53 62
Added Distribution Embedded Output Clock Modulation Section and Figure 54 63
Added Figure 55 64
Added Figure 56 and Figure 57 65
Added Figure 58, Figure 59, and Figure 60 66
Added Figure 61 67
Added Distribution Output Clock Synchronization Section 68
Added Table 37 69
Added Frequency Translation Loops Section and Figure 62 70
Added Table 38 71
Added Table 39 and Figure 63 72
Added Figure 64 73
Added Table 40 74
Added Figure 65 75
Added Source Profiles Section and Figure 66 76
Added Figure 67 78
Moved Digital PLL (DPLL) Section 79
Changes to Digital PLL (DPLL) Section 79
Added Figure 69 and Figure 70 81
Added Table 41 82
Added Figure 71 83
Added Table 42 and Figure 72 84
Added Table 43 and Table 44 87
Added Figure 74 88
Added Figure 75 and Figure 76 89
Added Figure 77 90
Added Figure 78 94
Added Table 45 95
Added Cascaded DPLL Configuration Section and Figure 79 97
Added Figure 80 98
Added Analog PLL (APLL) Section, Table 46, and Figure 81 100
Added Table 47 and Table 48 101
Added Figure 82 and Table 49 102
Added Reference Switching Section 103
Added Figure 83 104
Added Figure 84 and Table 50 105
Added Figure 85 107
Added Figure 86 108
Added Time-to-Digital Converter (TDC) Section and Figure 87 109
Added Timestamps Section and Table 51 110
Added User Access to Timestamps Section 112
Added Figure 88 and Figure 89 113
Added Timing Skew Measurements Using Two TDCs Section, Figure 90, and Table 52 115
Added Auxilary TDCs Section and Figure 91 117
Added Figure 92 118
Added Auxilary NCOs Section, Figure 93, and Figure 94 119

6/2017—Revision 0: Initial Version

FUNCTIONAL BLOCK DIAGRAM

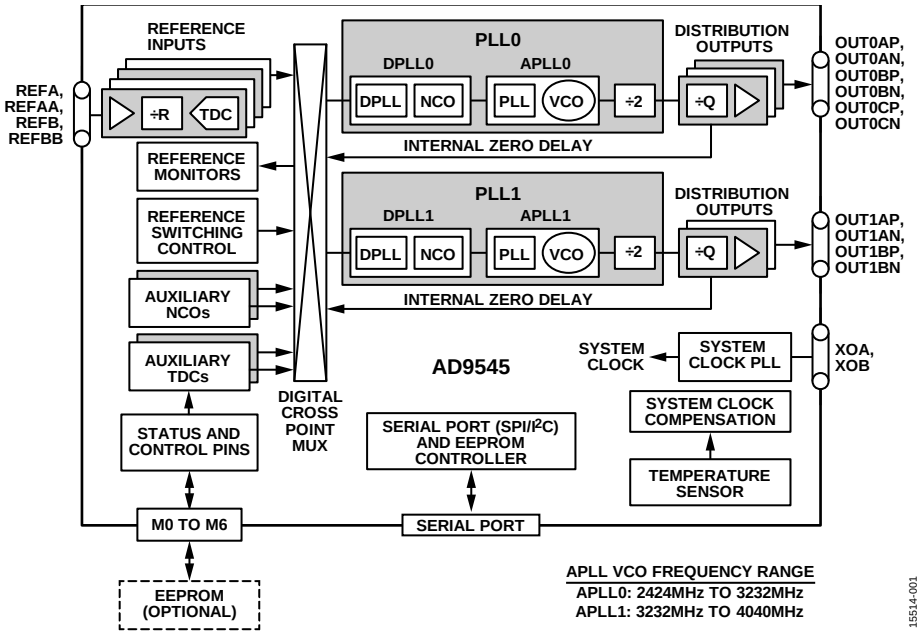


Figure 1.

SPECIFICATIONS

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

SUPPLY VOLTAGE

Table 1.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
SUPPLY VOLTAGE					
VDDIOA, 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 maximum supply voltage values given in Table 1 are the basis for the maximum supply current specifications. The typical supply voltage values given in Table 1 are the basis for the typical supply current specifications. The minimum supply voltage values given in Table 1 are the basis for the minimum supply current specifications.

Table 2.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
SUPPLY CURRENT FOR TYPICAL CONFIGURATION					The Typical Configuration specification in Table 3 is the basis for the values shown in this section
I_{VDDIOx}		5	8	mA	Aggregate current for all VDDIOx pins (where x = A or B)
I_{VDD}	260	310	355	mA	Aggregate current for all VDD pins
SUPPLY CURRENT FOR ALL BLOCKS RUNNING CONFIGURATION					The All Blocks Running condition in Table 3 is the basis for the values shown in this section
I_{VDDIOx}		5	8	mA	Aggregate current for all VDDIOx pins (where x = A or B)
I_{VDD}	321	390	430	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 3.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
POWER DISSIPATION					
Typical Configuration	445	560	671	mW	System clock = 49.152 MHz crystal; two DPLLs active; two 19.44 MHz input references in differential mode; two ac-coupled PLL0 CML output drivers at 245.76 MHz; and two PLL1 CML output drivers at 156.25 MHz
All Blocks Running	548	700	813	mW	System clock = 49.152 MHz crystal; two DPLLs active; 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 with the power down all bit set to Logic 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/Off		200		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

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
Incremental Power Dissipation					
Complete DPLL/APLL On/Off		200		mW	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; the blocks, when powered down, consist of one reference input, one DPLL, one APLL, two channel dividers, and two output drivers
Input Reference On/Off					
Differential (AC-Coupled Mode)		20		mW	f_{REF} = 19.44 MHz (see Figure 29)
Differential (DC-Coupled Mode)		21		mW	f_{REF} = 19.44 MHz (see Figure 30)
Single-Ended		13		mW	f_{REF} = 19.44 MHz
Output Distribution Driver On/Off					At 156.25 MHz
15 mA Mode		30		mW	
12 mA Mode		23		mW	
7.5 mA Mode		15		mW	
Auxiliary DPLL On/Off		1		mW	
Auxiliary Numerically Controlled Oscillator (NCO) to Mx Pin On/Off		1		mW	Fundamental set to 50 kHz
Auxiliary Time-To-Digital Converters (TDC) Input from Mx Pin On/Off		1		mW	Input frequency = 10 MHz, auxiliary TDC rate = 200 kHz

SYSTEM CLOCK INPUTS, XOA AND XOB

Table 4.

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 REFERENCE INPUT PATH					System clock input must be ac-coupled
Input Frequency Range					
System Clock Input Doubler					
Disabled	20		300	MHz	Support of oven controlled crystal oscillators (OCXOs) < 20 MHz is possible using the auxiliary DPLL for system clock frequency compensation
Enabled	16		150	MHz	
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	50			V/µs	Minimum input slew rate for device operation; oscillators with square wave outputs are recommended if not using a crystal
System Clock Input Divider (J Divider) Frequency	100			MHz	
System Clock Input Doubler Duty Cycle					Tolerable duty cycle variation on the system clock input when using the frequency doubler
20 MHz to 150 MHz	43	50	57	%	
16 MHz to 20 MHz	47	50	53	%	
Input Resistance		5		kΩ	
QUARTZ CRYSTAL RESONATOR PATH					
Resonator Frequency Range	25		60	MHz	Fundamental mode, AT cut crystal
Maximum Crystal Motional Resistance			100	Ω	A maximum motional resistance of 50 Ω, and maximum C_{LOAD} of 8 pF is strongly recommended for crystals >52 MHz

REFERENCE INPUTS

Table 5.

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 for Sinusoidal Input	20			V/µs	Minimum input slew rate for device operation; jitter degradation may 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	
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
Frequency Range					
1.2 V AC-Coupled	1		500×10^6	Hz	Lower limit dependent on ac coupling
1.2 V and 1.8 V Complementary Metal Oxide Semiconductor (CMOS)	1		500×10^6	Hz	CMOS specifications assume dc coupling of the input signal to the reference input pins
1.2 V AC-Coupled Common-Mode Voltage		610		mV	Internally generated self-bias voltage
Input Amplitude (Single-Ended, AC-Coupled Mode)	360		1200	mV p-p	Peak-to-peak single-ended voltage swing; instantaneous voltage must not exceed 1.3 V

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
1.2 V and 1.8 V CMOS Input Voltage					
High, V_{IH}	0.65 × V_{REF}		1.15 × V_{REF}	V	V_{REF} is determined by operating mode of the CMOS input receiver, 1.2 V or 1.8 V
Low, V_{IL}			0.35 × V_{REF}	V	
Input Resistance					
DC-Coupled Single-Ended Mode		30		kΩ	
AC-Coupled Single-Ended Mode		15		kΩ	
Input Pulse Width	900			ps	
REFERENCE DEMODULATOR					
Carrier Frequency (Sync Edge = 1, 2, 3)					
Band 0					
DC Balanced Modulation	0.5		30	MHz	
Unbalanced Modulation	0.5		45	MHz	
Band 1					
DC Balanced Modulation	1		90	MHz	
Unbalanced Modulation	1		135	MHz	
Embedded Clock Rate	1		f_{OUT}/6	Hz	f_{OUT} is the nominal output frequency of the output with the embedded clock
Duty Cycle Deviation					t_{SYS} = 1/(f_{SYS}); where f_{SYS} is the system clock frequency. f_{SYS} must be in the range of 2250 MHz to 2415 MHz. f_{SYS} = f_{OSC} × 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 SYSCLK input scale factor. When the SYSCLK doubler is disable, J is the value of the J-divider (1, 2, 4, or 8). When the SYSCLK doubler is enabled, J = 1/2.
DC Balanced Modulation	t_{SYS} × 5/3		1/(4 × f_{OUT})	sec	f_{OUT} is the output frequency
Unbalanced Modulation	t_{SYS} × 5/2		1/(4 × f_{OUT})	sec	
Polarity Detection Enabled	t_{SYS} × 5		1/(4 × f_{OUT})	sec	

REFERENCE MONITORS

Table 6.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
REFERENCE MONITORS					
Reference Monitor					
Loss of Reference Detection Time		4.9 + 0.13 × t_{PFD}		µs	t_{PFD} is the nominal phase detector period, 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 in MHz.
Frequency Out of Range Limits	5 × 10^{-8}		0.015	Δf/f_{REF}	f_{REF} is the reference input frequency and Δf is the frequency deviation relative to f_{REF}; programmable with the lower bound subject to quality of the system clock (or the source of system clock compensation)
Validation Timer	0.001		1048	sec	Programmable in 1 ms increments
Excess Jitter Alarm Threshold	1		65535	ns	Programmable in 1 ns increments

DPLL PHASE CHARACTERISTICS

Table 7.

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 the AD9545 Register Map Reference Manual) 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	±140	ps	
Steady State					
Phase Buildout Operation		±18	±125	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	±40	ps	
Steady State					
Phase Buildout Operation		±4	±35	ps	
Hitless Operation		0		ps	
PHASE SLEW LIMITER	0.001		250	µs/sec	See the AN-1420 Application Note

DISTRIBUTION CLOCK OUTPUTS

Table 8.

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
Common Mode Logic (CML)	1		500×10^6	Hz	Terminated per Figure 33
High Speed Current Steering Logic (HCSL)	1		500×10^6	Hz	Terminated per Figure 32
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 32
CML	257	348	408	mV	Terminated to VDD (nominal 1.8 V) per Figure 33
Output Current = 15 mA					
HCSL	631	745	809	mV	Terminated per Figure 32
CML	578	729	818	mV	Terminated to VDD (nominal 1.8 V) per Figure 33
Common-Mode Output Voltage					
Output Current = 7.5 mA					
HCSL	155	184	201	mV	Terminated per Figure 32
CML	VDD − 208	VDD − 188	VDD − 169	mV	Terminated to VDD (nominal 1.8 V) per Figure 33 (maximum common-mode voltage case occurs at the minimum amplitude)
Output Current = 15 mA					
HCSL	316	372	405	mV	Terminated per Figure 32
CML	VDD − 416	VDD − 371	VDD − 327	mV	Terminated to VDD (nominal 1.8 V) per Figure 33 (maximum common-mode voltage case occurs at the minimum amplitude)

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 mA					
Voltage Swing (Peak-to-Peak)					
HCSL Driver Mode	509	584	634	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$
CML Driver Mode	456	565	644	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$ connected to VDD (nominal 1.8 V) instead of GND
Voltage Swing Midpoint					
HCSL Driver Mode	255	292	317	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$
CML Driver Mode	VDD − 325	VDD − 291	VDD − 266	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$ connected to VDD (nominal 1.8 V) instead of GND
Output Current = 15 mA					
Voltage Swing (Peak-to-Peak)					
HCSL Driver Mode	645	734	796	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$
CML Driver Mode	589	721	815	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$ connected to VDD (nominal 1.8 V) instead of GND
Voltage Swing Midpoint					
HCSL Driver Mode	322	367	398	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$
CML Driver Mode	VDD − 411	VDD − 367	VDD − 334	mV	Each output terminated per Figure 37 with $R_L = 50\ \Omega$ connected to VDD (nominal 1.8 V) instead of GND

TIME DURATION OF DIGITAL FUNCTIONS

Table 9.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
TIME DURATION OF DIGITAL FUNCTIONS					
EEPROM to Register Download Time		10		ms	Using the Typical Configuration specification from Table 3
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 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 START OF DPLL ACTIVATION TO ACTIVE PHASE DETECTOR OUTPUT					
Untagged Operation			10	t_{PFD}	t_{PFD} is the nominal phase detector period given by 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 $f_{FEEDBACK}$ (for all other tagged modes); the tag ratio corresponds to the selection of f_{TAG}

DIGITAL PLL (DPLL0, DPLL1) SPECIFICATIONS

Table 10.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
DIGITAL PLL					
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
DIGITAL PLL NCO Division Ratio					These specifications cover limitations on the DPLLx frequency tuning word (FTW0); the AD9545 evaluation software frequency planning wizard sets these values automatically for the user, and the AD9545 evaluation software is available for download from the AD9545 product page at www.analog.com/AD9545; 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

DIGITAL PLL LOCK DETECTION SPECIFICATIONS

Table 11.

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	

HOLDOVER SPECIFICATIONS

Table 12.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
HOLDOVER SPECIFICATIONS					
Initial Frequency Accuracy		±0.01	±0.1	ppb	AD9545 is configured using Configuration 1 from Table 22; excludes frequency drift of system clock (SYSCLK) source; excludes frequency drift of input reference prior to entering holdover; 160 ms history timer; history hold off setting of 8; three features (bits) are enabled: DPLLx delay history frequency lock (Bit 4 in Register 0x100E and Register 0x140E), DPLLx delay history phase lock (Bit 3 in Register 0x100E and Register 0x140E), and DPLLx delay history until not slew limiting (Bit 5 in Register 0x100E and Register 0x140E)
Relative Frequency Accuracy Between Channels					
Cascaded Operation		0		ppb	
History Averaging Window	0.001		268435	sec	

ANALOG PLL (APLL0, APLL1) SPECIFICATIONS

Table 13.

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

OUTPUT CHANNEL DIVIDER SPECIFICATIONS

Table 14.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
OUTPUT PHASE ADJUST STEP SIZE	1			t_{VCO}	t_{VCO} = 1/(APLLx VCO frequency), where x = 0, 1
MODULATOR					
Carrier Frequency			$f_{VCO} \div 16$	Hz	The maximum value is the APLL0/1 VCO frequency divided by 16
Time Deviation (from Nominal Duty Cycle of Carrier Clock)	0		$2^{16} - 1$	t_{VCO}	t_{VCO} = 1/(APLLx VCO frequency), where x = 0, 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	$f_{OUT} \div (2^{28} - 1)$		$f_{OUT} \div 6$	Hz	f_{OUT} is the output frequency

AUXILIARY CIRCUIT SPECIFICATIONS

Table 15.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
TIME TO DIGITAL CONVERTERS (TDCs)					
Periodic Operation					Applicable to all TDCs contained within the AD9545
Frequency Range	1		2×10^5	Hz	
Timestamp Jitter (RMS)		20		ps	System clock source = 52 MHz crystal
NCOs					These NCOs are called AUXNCO0 and AUXNCO1 in the register map and evaluation software
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	

SYSTEM CLOCK COMPENSATION SPECIFICATIONS

Table 16.

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

TEMPERATURE SENSOR SPECIFICATIONS

Table 17.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
TEMPERATURE					
Accuracy					
Absolute		5		°C	T_A = −50°C to +110°C
Relative		1.7		%	T_A = −50°C to +110°C
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

SERIAL PORT SPECIFICATIONS

Serial Port Interface (SPI) Mode

Table 18.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
CSB					Valid for VDDIOA = 1.8 V, 2.5 V, and 3.3 V
Input Logic 1 Voltage	VDDIOA – 0.4			V	
Input Logic 0 Voltage			0.4	V	

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
Input Logic 1 Current		1		µA	
Input Logic 0 Current		1		µA	
SCLK					
Input Logic 1 Voltage	VDDIOA – 0.4			V	
Input Logic 0 Voltage			0.4	V	
Input Logic 1 Current		1		µA	
Input Logic 0 Current		1		µA	
SDIO					
As an Input					
Input Logic 1 Voltage	VDDIOA – 0.4			V	
Input Logic 0 Voltage			0.4	V	
Input Logic 1 Current		1		µA	
Input Logic 0 Current		1		µA	
As an Output					
Output Logic 1 Voltage	VDDIOA – 0.2			V	1 mA load current
Output Logic 0 Voltage			0.2	V	1 mA load current
SDO					
Output Logic 1 Voltage	VDDIOA – 0.2			V	1 mA load current
Output Logic 0 Voltage			0.2	V	1 mA load current
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 19.

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 Logic 1 Voltage	70			% of VDDIOA	
Input Logic 0 Voltage			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	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
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	

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
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 for Each Bus Line, C_B			400	pF	

LOGIC INPUT SPECIFICATIONS (RESETB, M0 TO M6)

Table 20.

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	
LOGIC INPUTS (M0 to M6)					Valid for 3.3 V ≥ VDDIOx ≥ 1.8 V; VDDIOA applies to the M5 pin and the 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
Frequency Range			51	MHz	
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 M6)

Table 21.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
LOGIC OUTPUTS (M0 to M6)					Valid for 3.3 V ≥ VDDIOx ≥ 1.8 V; VDDIOA applies for the M5 and M6 pins; VDDIOB applies for M0 to M4; normal (default) output drive current setting for M0 through M6
Frequency Range			26	MHz	
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

JITTER GENERATION (RANDOM JITTER)

Table 22.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
JITTER GENERATION					System clock doubler enabled; high phase margin mode enabled; there is not a significant jitter difference between driver modes
Channel 0—DPLL0, APLL0					Channel 1 powered down
RMS Jitter (12 kHz to 20 MHz)					
Configuration 1—155.52 MHz		223		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 38.88 MHz, f_{VCO} = 2488.32 MHz, f_{OUT} = 155.52 MHz, DPLL BW = 50 Hz, phase buildout operation
Configuration 2—245.76 MHz		220		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 30.72 MHz, f_{VCO} = 2457.6 MHz, f_{OUT} = 245.76 MHz, DPLL BW = 50 Hz, internal zero delay operation
Configuration 3—491.52 MHz		235		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz temperature compensated crystal oscillator (TCXO), BW_{COMP} = 50 Hz, f_{REF} = 1 Hz, f_{VCO} = 2949.12 MHz, f_{OUT} = 491.52 MHz, DPLL BW = 50 mHz, phase buildout operation
Configuration 4—125 MHz		213		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 125 MHz, f_{VCO} = 2500 MHz, f_{OUT} = 125 MHz, DPLL BW = 0.1 Hz, phase buildout operation
Configuration 5—312.5 MHz		217		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 25 MHz, f_{VCO} = 2500 MHz, f_{OUT} = 312.5 MHz, DPLL BW = 50 Hz , phase buildout operation
Configuration 6—174.7030837 MHz		230		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 155.52 MHz, f_{VCO} = 2620.5463 MHz, f_{OUT} = (155.52 × 255/227) MHz, DPLL BW = 50 Hz
Channel 1—DPLL1, APLL1					Channel 0 powered down
RMS Jitter (12 kHz to 20 MHz)					
Configuration 1—155.52 MHz		247		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 38.88 MHz, f_{VCO} = 3265.92 MHz, f_{OUT} = 155.52 MHz, DPLL BW = 50 Hz, phase buildout operation, half divide enabled
Configuration 2—245.76 MHz		280		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 30.72 MHz, f_{VCO} = 3686.4 MHz, f_{OUT} = 245.76 MHz, DPLL BW = 50 Hz, half divide enabled, internal zero delay operation
Configuration 3—491.52 MHz		323		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 1 Hz, f_{VCO} = 3932.16 MHz, f_{OUT} = 491.52 MHz, DPLL BW = 50 mHz, phase buildout operation
Configuration 4—125 MHz		243		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 125 MHz, f_{VCO} = 3250 MHz, f_{OUT} = 125 MHz, DPLL BW = 0.1 Hz, phase buildout operation
Configuration 5—312.5 MHz		266		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 25 MHz, f_{VCO} = 3750 MHz, f_{OUT} = 312.5 MHz, DPLL BW = 50 Hz, phase buildout operation
Configuration 6—174.7030837 MHz		264		fs	Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 155.52 MHz, f_{VCO} = 3319.3586 MHz, f_{OUT} = (155.52 × 255/227) MHz, DPLL BW = 50 Hz, phase buildout operation

PHASE NOISE

Table 23.

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
PHASE NOISE					System clock doubler enabled; high phase margin mode enabled; there is not a significant jitter difference between driver modes
Channel 0—DPLL0, APLL0					Channel 1 powered down
RMS Jitter (12 kHz to 20 MHz)					
Configuration 1—155.52 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 38.88 MHz, f_{VCO} = 2488.32 MHz, f_{OUT} = 155.52 MHz, DPLL BW = 50 Hz, phase buildout operation
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	
Configuration 2—245.76 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 30.72 MHz, f_{VCO} = 2457.6 MHz, f_{OUT} = 245.76 MHz, DPLL BW = 50 Hz, internal zero delay operation
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	
Configuration 3—491.52 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 1 Hz, f_{VCO} = 2949.12 MHz, f_{OUT} = 491.52 MHz, DPLL BW = 50 mHz, phase buildout operation
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			
Configuration 4—125 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 125 MHz, f_{VCO} = 2500 MHz, f_{OUT} = 125 MHz, DPLL BW = 0.1 Hz, phase buildout operation
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	

Parameter	Min	Typ	Max	Unit	Test Conditions/Comments
Configuration 5—312.5 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 25 MHz, f_{VCO} = 2500 MHz, f_{OUT} = 312.5 MHz, DPLL BW = 50 Hz, phase buildout operation
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	
Configuration 6—174.7030837 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 155.52 MHz, f_{VCO} = 2620.5463 MHz, f_{OUT} = (155.52 × 255/227) MHz, DPLL BW = 50 Hz
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
RMS Jitter (12 kHz to 20 MHz)					
Configuration 1—155.52 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 38.88 MHz, f_{VCO} = 3265.92 MHz, f_{OUT} = 155.52 MHz, DPLL BW = 50 Hz, phase buildout operation, half divide enabled
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	
Configuration 2—245.76 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 30.72 MHz, f_{VCO} = 3686.4 MHz, f_{OUT} = 245.76 MHz, DPLL BW = 50 Hz, half divide enabled; internal zero delay operation
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
Configuration 3—491.52 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 1 Hz, f_{VCO} = 3932.16 MHz, f_{OUT} = 491.52 MHz, DPLL BW = 0.5 Hz, phase buildout operation
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	
Configuration 4—125 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{COMP} = 19.2 MHz TCXO, BW_{COMP} = 50 Hz, f_{REF} = 125 MHz, f_{VCO} = 3250 MHz, f_{OUT} = 125 MHz, DPLL BW = 0.1 Hz, phase buildout operation
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	
Configuration 5—312.5 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 25 MHz, f_{VCO} = 3750 MHz, f_{OUT} = 312.5 MHz, DPLL BW = 50 Hz, phase buildout operation
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	
Configuration 6—174.7030837 MHz					Device configuration: f_{OSC} = 52 MHz crystal, f_{REF} = 155.52 MHz, f_{VCO} = 3319.3586 MHz, f_{OUT} = (155.52 × 255/227) MHz, DPLL BW = 50 Hz
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 24.

Parameter	Rating
1.8 V Supply Voltage (VDD)	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
Operating Temperature Range[1]	−40°C to +85°C
Lead Temperature (Soldering 10 sec)	300°C

[1] See the Thermal Resistance section for additional information.

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).

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

Table 25. Thermal Resistance

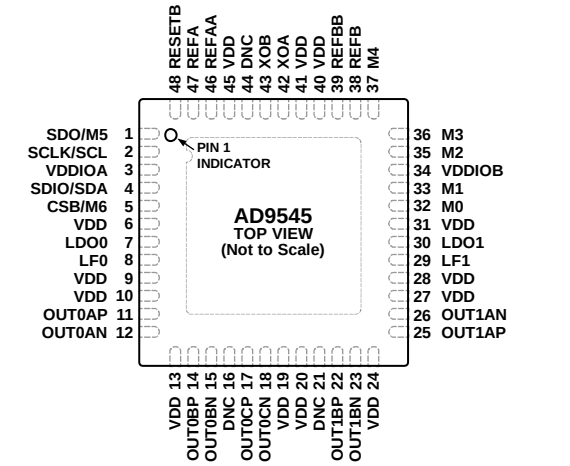
Package Type	θ_{JA}	θ_{JMA}[1]	θ_{JC}	Unit
CP-48-13[2, 3]	23.9	19.4, 18.2	1.5	°C/W

[1] θ_{JMA} is 19.4°C/W at 1.0 m/sec airflow and 18.2°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.

15514-002

Figure 2. Pin Configuration

Table 26. 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 (SCLK) Pin in SPI Mode. Changes to the VDDIOA supply voltage affect the V_{IH} and V_{OH} values for this pin. Serial Clock Pin (SCL) in I^2C Mode. 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 this pin to ground. This pin 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 this pin to Pin 7 (LDO0).
11	OUT0AP	Output	HCSL, LVDS, CML, CMOS	PLL0 Output 0A.
12	OUT0AN	Output	HCSL, LVDS, CML, CMOS	PLL0 Complementary Output 0A.
14	OUT0BP	Output	HCSL, LVDS, CML, CMOS	PLL0 Output 0B.
15	OUT0BN	Output	HCSL, LVDS, CML, CMOS	PLL0 Complementary Output 0B.

Pin No.	Mnemonic	Input/ Output	Pin Type	Description
16, 21, 44	DNC	DNC	No Connect	Do Not Connect. Leave these pins floating.
17	OUT0CP	Output	HCSL, LVDS, CML, CMOS	PLL0 Output 0C.
18	OUT0CN	Output	HCSL, LVDS, CML, CMOS	PLL0 Complementary Output 0C.
22	OUT1BP	Output	HCSL, LVDS, CML, CMOS	PLL1 Output 1B.
23	OUT1BN	Output	HCSL, LVDS, CML, CMOS	PLL1 Complementary Output 1B.
25	OUT1AP	Output	HCSL, LVDS, CML, CMOS	PLL1 Output 1A.
26	OUT1AN	Input/ Output	HCSL, LVDS, CML, CMOS	PLL1 Complementary Output 1A.
29	LF1	Input/ output	Loop filter for APLL1	Loop Filter Node for APLL1. Connect a 3.9 nF capacitor from this pin to Pin 30 (LDO1).
30	LDO1	Input	LDO bypass	APLL1 Loop Filter Voltage Regulator. Connect a 0.1 µF capacitor from this pin to ground. This pin 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. These are status and control pins. Changes to the VDDIOB supply voltage affect the V_{IH} and V_{OH} values for these 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. This 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 input is typically ac-coupled; when configured in this manner, it 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.
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 this pin. No connection is necessary to this 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 is ac-coupled with a 0.01 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. Complementary signal to XOA. XOB contains internal dc biasing and is 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 this pin. No connection is necessary to this pin if REFA is a single-ended input and REFAA is not used. If dc-coupled, the input is single-ended 1.8 V CMOS.
47	REFA	Input	1.8 V single-ended or differential input	Reference A Input. This internally biased input is typically ac-coupled; when configured in this manner, it 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.
48	RESETB	Input	1.8 V CMOS logic	Active Low Chip Reset. This 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 this pin.
	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

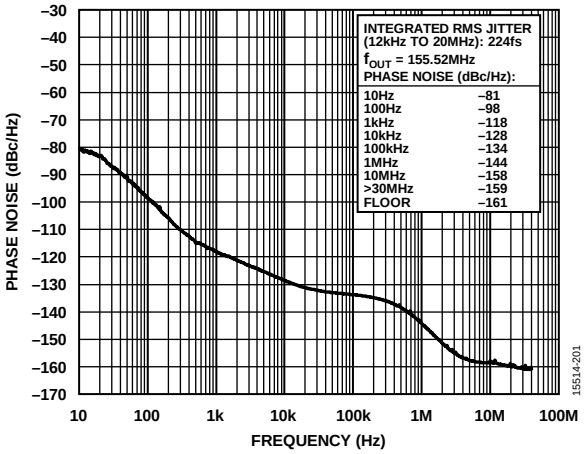


Figure 3. Absolute Phase Noise (PLL0, Configuration 1, HCSL Mode, f_{REF} = 38.88 MHz, f_{OUT} = 155.52 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW

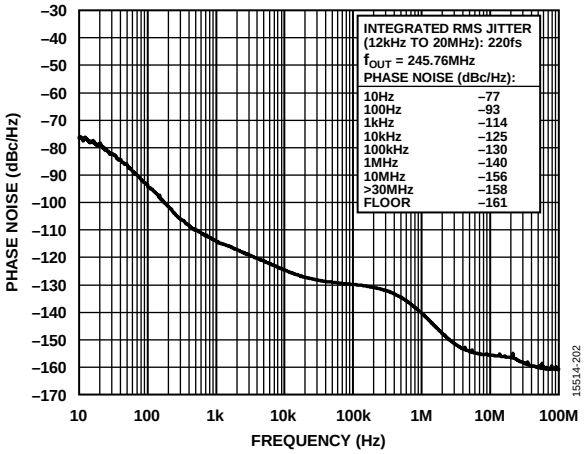


Figure 4. Absolute Phase Noise (PLL0, Configuration 2, HCSL Mode, f_{REF} = 30.72 MHz, f_{OUT} = 245.76 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW)

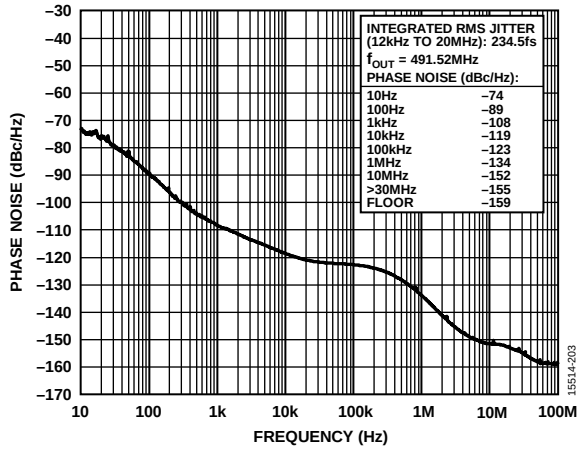


Figure 5. Absolute Phase Noise (PLL0, Configuration 3, HCSL Mode, f_{REF} = 1 Hz, f_{OUT} = 491.52 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 50 MHz DPLL BW)

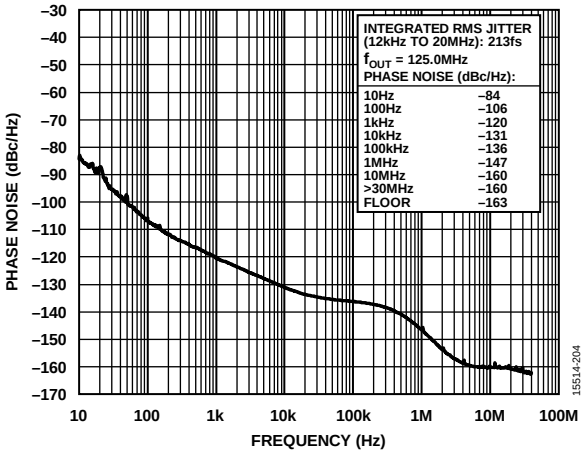


Figure 6. Absolute Phase Noise (PLL0, Configuration 4, HCSL Mode, f_{REF} = 125 MHz, f_{OUT} = 125.0 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 0.1 Hz DPLL BW, Phase Buildout Mode)

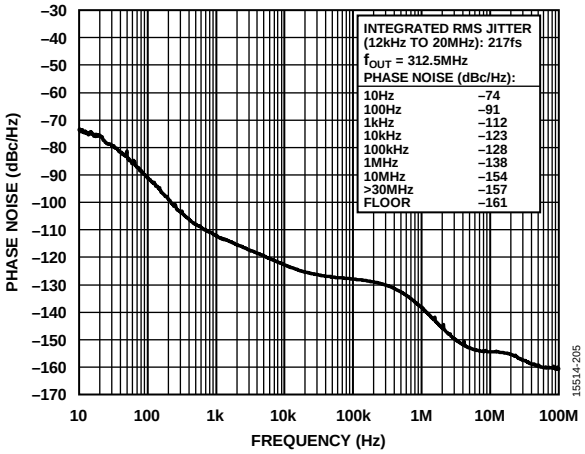


Figure 7. Absolute Phase Noise (PLL0, Configuration 5, HCSL Mode, f_{REF} = 25 MHz, f_{OUT} = 312.5 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW, Phase Buildout Mode)

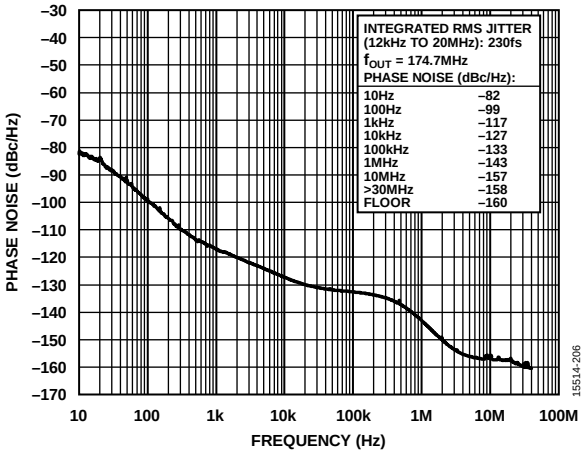


Figure 8. Absolute Phase Noise (PLL0, Configuration 6, HCSL Mode, f_{REF} = 155.52 MHz, f_{OUT} = 174.7 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW, Phase Buildout Mode)

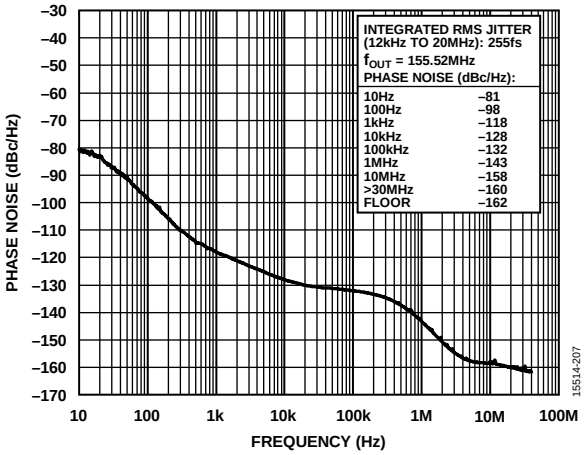


Figure 9. Absolute Phase Noise (PLL1, Configuration 1, HCSL Mode, f_{REF} = 38.88 MHz, f_{OUT} = 155.52 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW)

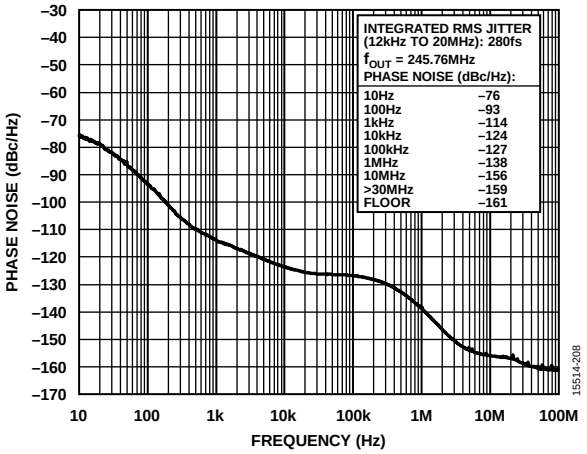


Figure 10. Absolute Phase Noise (PLL1, Configuration 2, HCSL Mode, f_{REF} = 30.72 MHz, f_{OUT} = 245.76 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW)

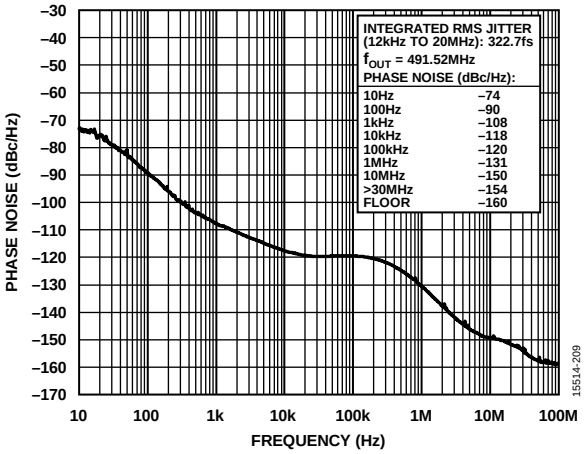


Figure 11. Absolute Phase Noise (PLL1, Configuration 3, HCSL Mode, f_{REF} = 1 Hz, f_{OUT} = 491.52 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 0.5 Hz DPLL BW)

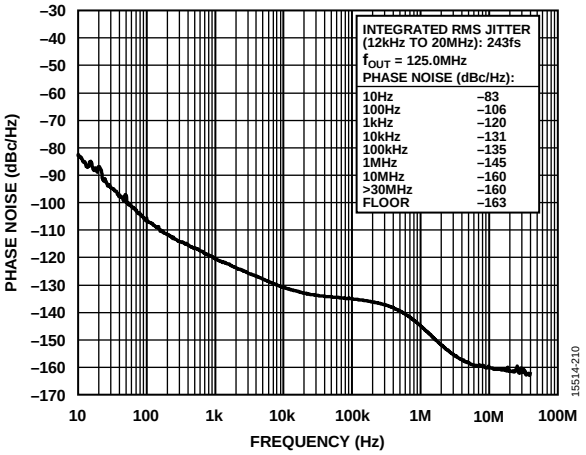


Figure 12. Absolute Phase Noise (PLL1, Configuration 4, HCSL Mode, f_{REF} = 125 MHz, f_{OUT} = 125 MHz, f_{COMP} = 19.2 MHz TCXO, f_{OSC} = 52 MHz Crystal, 0.1 Hz DPLL BW, Phase Buildout Mode)

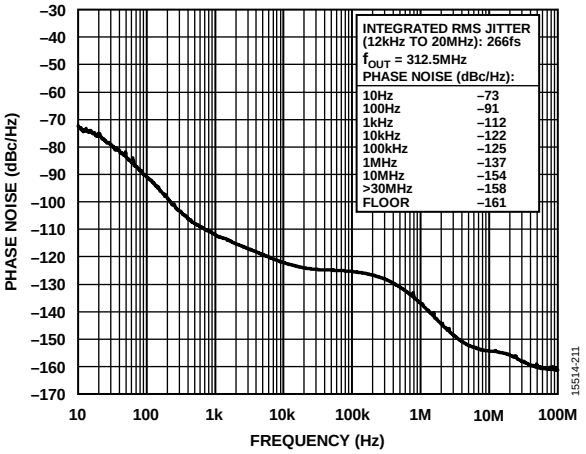


Figure 13. Absolute Phase Noise (PLL1, Configuration 5, HCSL Mode, f_{REF} = 25 MHz, f_{OUT} = 312.5 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW, Phase Buildout Mode)

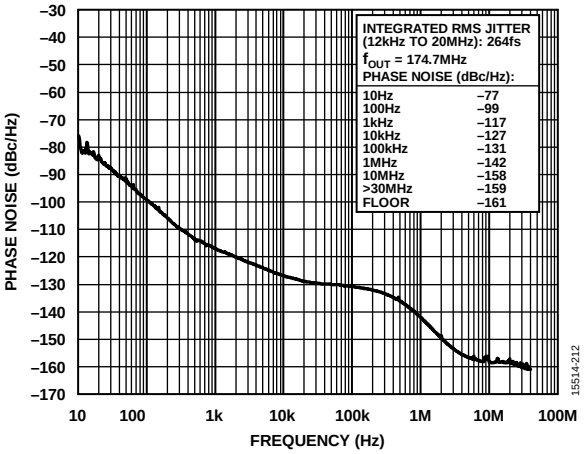


Figure 14. Absolute Phase Noise (PLL1, Configuration 6, HCSL Mode, f_{REF} = 155.52 MHz, f_{OUT} = 174.7 MHz, f_{OSC} = 52 MHz Crystal, 50 Hz DPLL BW, Phase Buildout Mode)

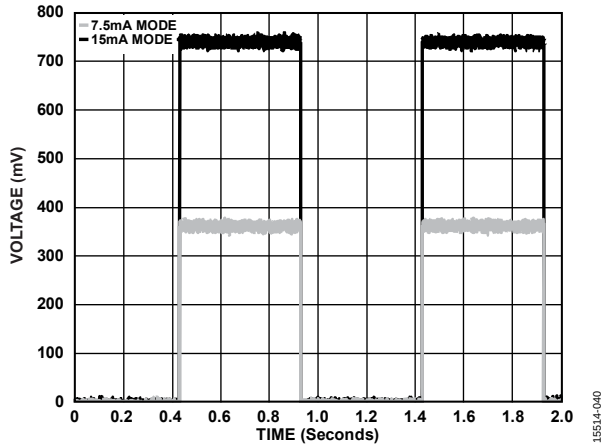


Figure 15. DC-Coupled, Single-Ended, 1 Hz Output Waveforms Using HCSL 7.5 mA and 15 mA Mode Terminated 50 Ω to GND per Figure 39; Slew Rate: ~7 V/ns for 15 mA Mode; ~3.5 V/ns for 7.5 mA Mode

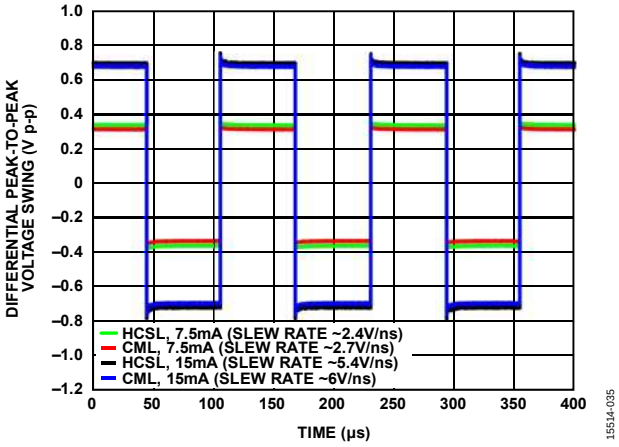


Figure 16. 8 kHz Output Waveforms for Various Driver Settings; HCSL Drivers Terminated 50 Ω to GND per Figure 32; CML Drivers Terminated 50 Ω to 1.8 V per Figure 33

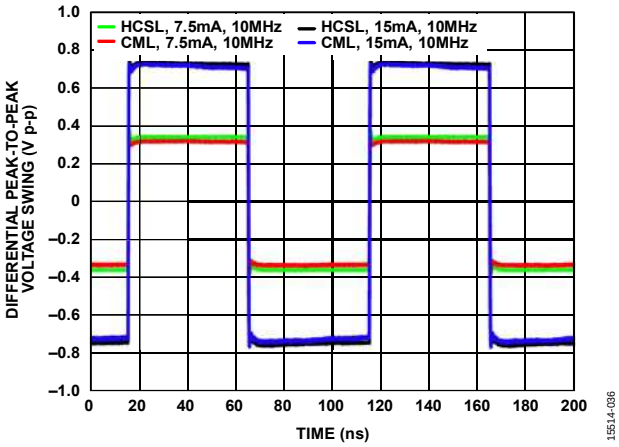


Figure 17. 10 MHz Output Waveforms for Various Driver Settings; HCSL Drivers Terminated 50 Ω to GND per Figure 32; CML Drivers Terminated 50 Ω to 1.8 V per Figure 33

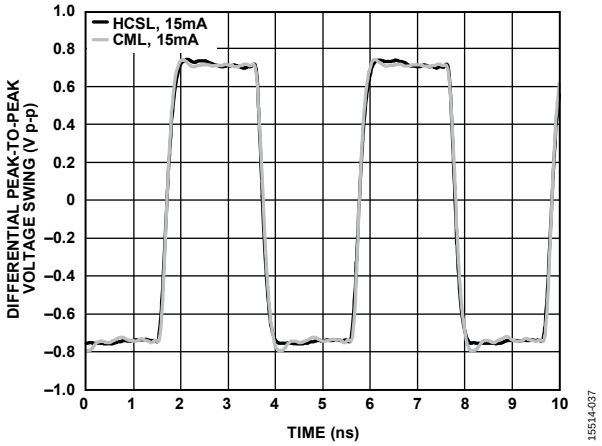


Figure 18. 245.76 MHz Output Waveform for 15 mA Driver Settings; HCSL Drivers Terminated 50 Ω to GND per Figure 32; CML Drivers Terminated 50 Ω to 1.8 V per Figure 33

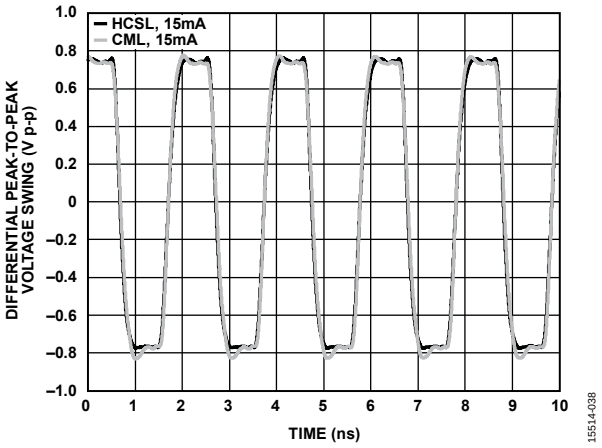


Figure 19. 491.52 MHz Output Waveform for15 mA Driver Settings; HCSL Drivers Terminated 50 Ω to GND per Figure 32; CML Drivers Terminated 50 Ω to 1.8 V per Figure 33

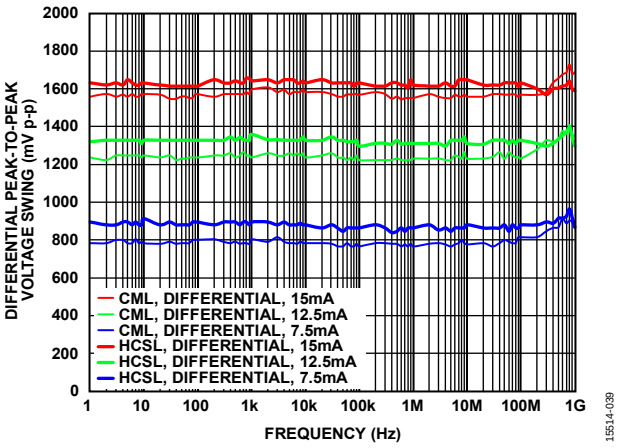


Figure 20. Differential Output Amplitude Waveforms; HCSL Drivers Terminated 50 Ω to GND per Figure 32; CML Drivers Terminated 50 Ω to 1.8 V per Figure 33

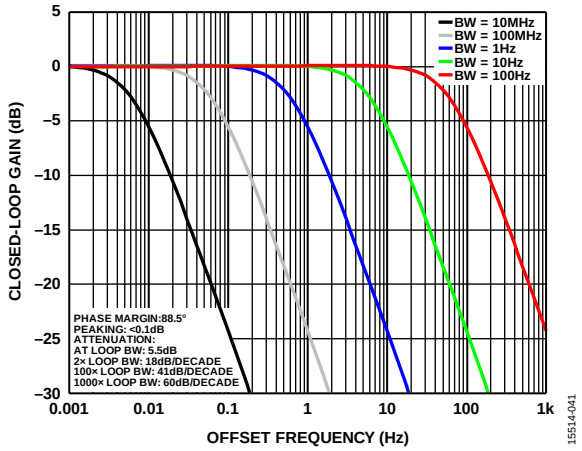


Figure 21. DPLL Closed-Loop Transfer Function Nominal Phase Margin Loop Filter Setting

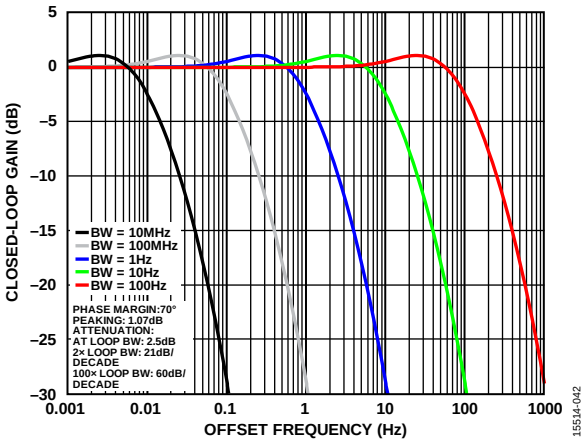


Figure 22. DPLL Closed-Loop Transfer Function High Phase Margin Loop Filter Setting

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 (PBO) Switchover

PBO 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. PBO 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 PBO 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, PBO places no restriction on the output/input frequency ratio.

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

THEORY OF OPERATION

Figure 41 summarizes the fundamental building blocks of the AD9545. The core of the device comprises two independent PLLs, PLL0 and PLL1. The primary function of the device is to accept a clock signal at one of the REFx inputs and deliver it to one (or more) of the OUTxyP and OUTxyN outputs via the two independent pathways provided by PLL0 and PLL1.

In addition, the device provides a frequency translation between the input and output by a programmed ratio, while at the same time suppressing the jitter carried by the input signal.

The AD9545 synthesizes its system clock via a system clock PLL, which 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 60 MHz) directly across the XOA and XOB pins. The AD9545 primary time base for all of its time keeping functions is the system clock.

The AD9545 supports up to four reference input clock signals with frequencies ranging from 1 Hz to 750 MHz. Each reference input has a dedicated reference monitor for detecting if the reference signal is lost or compromised. The AD9545 can detect reference faults and automatically switch to one of the other valid reference inputs with user-programmable priority.

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 AD9545 continuously provides an output clock signal while the system clock is present. The AD9545 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 or 3.232 GHz to 4.040 GHz (APLL0 or APLL1, respectively). The APLL VCO output frequency is downconverted by a factor of three 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 single-ended or differential output with adjustable output current (7.5 mA, 12 mA, or 15 mA).

The user also has access to two independent auxiliary NCOs. These provide for internally generated frequencies up to 65 kHz with a tuning resolution of 2^{-40} Hz (approximately 1 pHz). The user can also access two auxiliary TDCs, which generate high resolution time stamps associated with the rising edges of an input signal applied to one of the Mx pins.

The AD9545 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

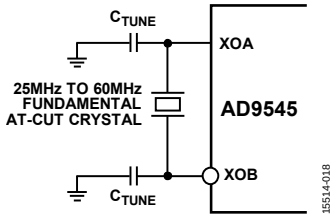


Figure 23. System Clock Input (XOA/XOB) in Crystal Mode ($C_{TUNE} = 2 \times (C_{LOAD} - C_{STRAY})$, Where Typical $C_{STRAY} = 2$ pF to 5 pF)

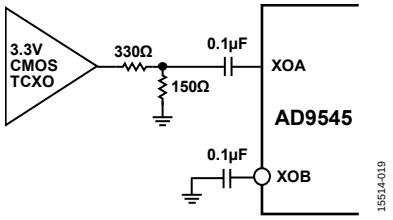


Figure 24. System Clock Input (XOA, XOB) When Using a TCXO/OCXO with 3.3 V CMOS Output

REFERENCE CLOCK INPUTS

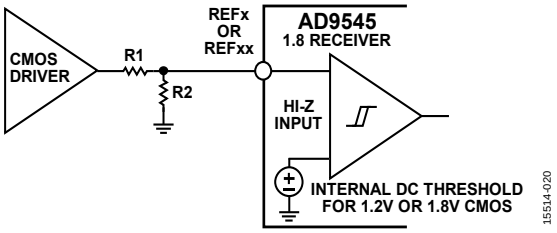


Figure 25. Single-Ended DC-Coupled Mode, 1.2 V or 1.8 V CMOS

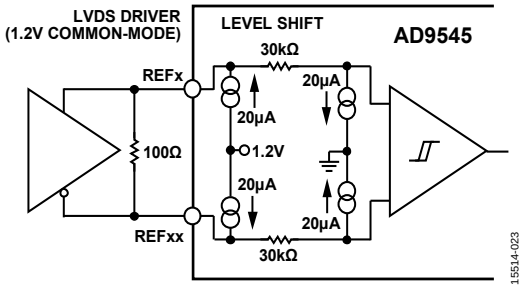


Figure 28. Differential LVDS Input Mode

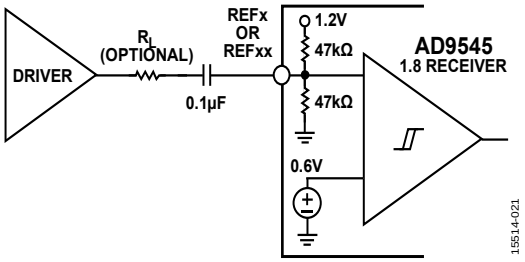


Figure 26. Single-Ended AC-Coupled Interface Mode

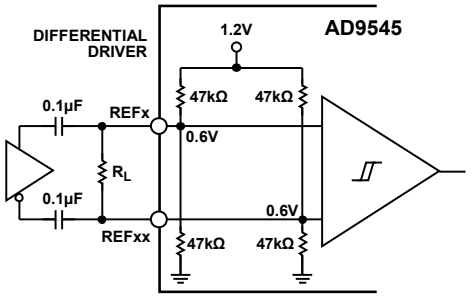


Figure 29. Differential AC-Coupled Mode (R_L = 100 Ω Is Recommended, Except For HCSL)

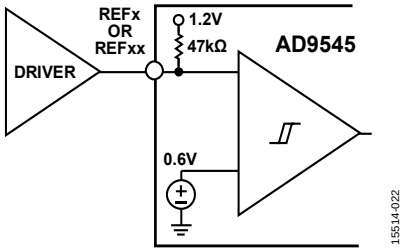


Figure 27. Single-Ended Internal Pull-Up Resistor Mode

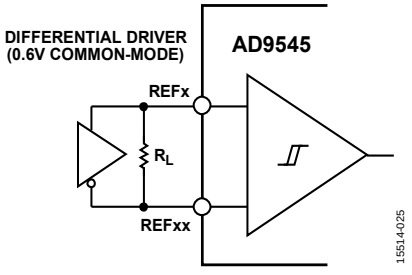


Figure 30. Differential DC-Coupled Mode

CLOCK OUTPUTS

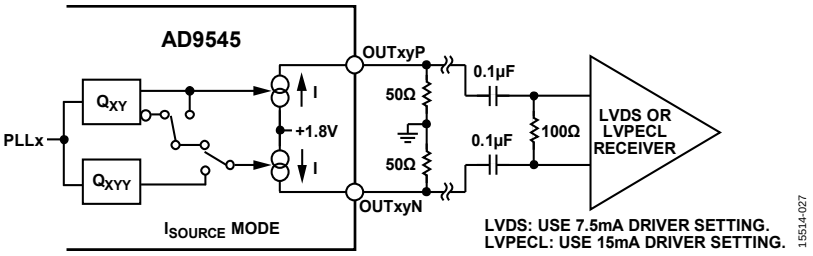


Figure 31. LVDS-Compatible Output Swing, AC-Coupled (V p-p ≈ 375 mV per Section for I = 15 mA)

The output drivers of the AD9545 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 both pin drivers source current (I_{SOURCE})or both pin drivers sink current (I_{SINK})). The current source (sink) of the individual driver is either on or off depending on the 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 31, 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.

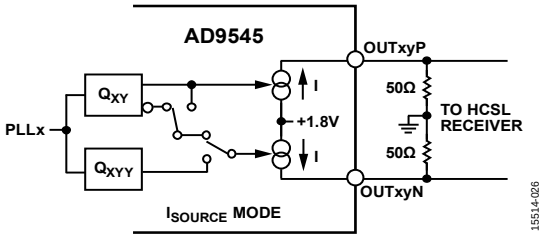


Figure 32. HCSL Output, V p-p ≈ 750 mV per Section (I = 15 mA)

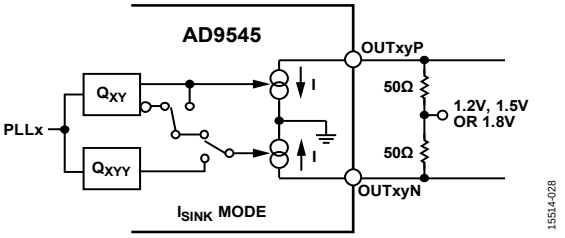


Figure 33. CML Output (I = 7.5 mA; I = 15 mA Options for 1.5 V or 1.8 V Supply)

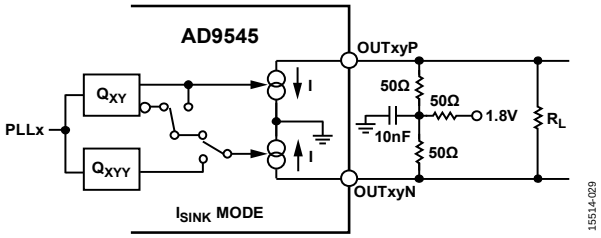


Figure 34. LVDS-Compatible Output, 1.24 V Common Mode, T Network (I = 7.5 mA; I = 15 mA with Extra 100 Ω Termination, R_L)

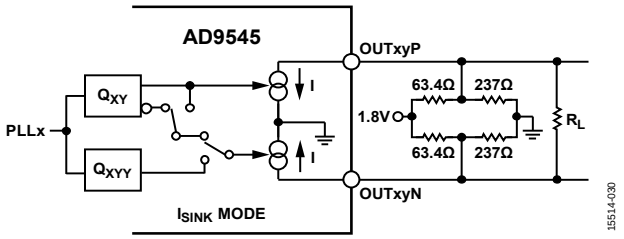


Figure 35. LVDS-Compatible Output, 1.2 V Common-Mode, Thevenin Bias Network (I = 7.5 mA; 15 mA with Extra 100 Ω Termination, R_L)

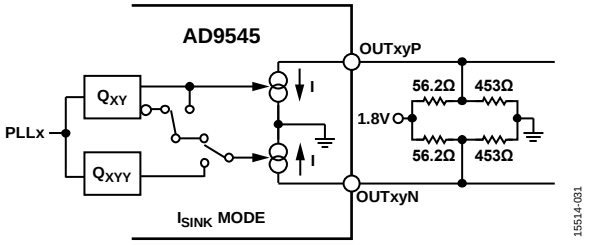


Figure 36. 2.5 V LVPECL or Double Amplitude LVDS-Compatible Boost Output, 1.5 V p-p, 1.24 V Common-Mode (I = 15 mA)

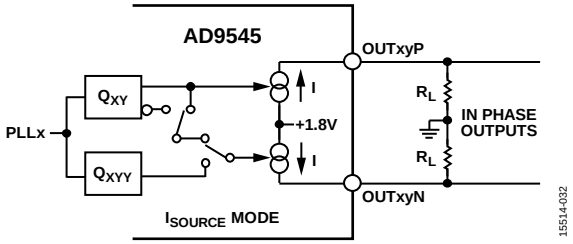


Figure 37. Single-Divider, Single-Ended Mode Providing In-Phase Outputs (Current Source Mode)

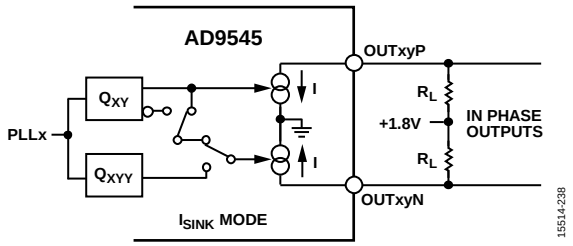


Figure 38. Single-Single-Ended Mode Providing In-Phase Outputs (Current Sink Mode)

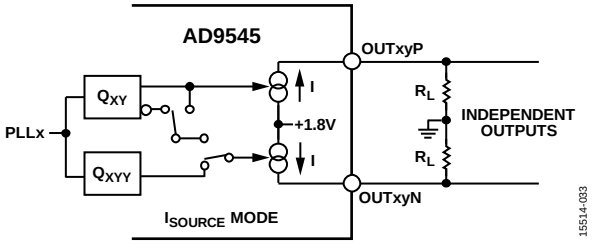


Figure 39. Dual-Divider, Single-Ended Mode Providing Independent Outputs (Current Source Mode)

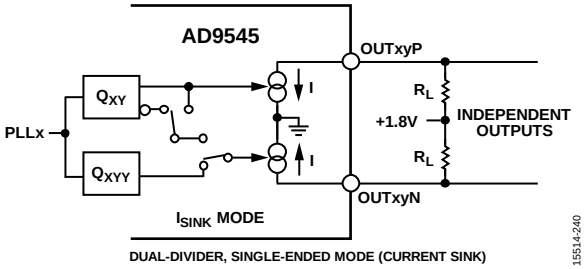


Figure 40. Dual-Divider, Single-Ended Mode Providing Independent Outputs (Current Sink Mode)

SYSTEM CLOCK PLL

SYSTEM CLOCK PLL OVERVIEW

The system clock PLL (see Figure 41) comprises an integer-N frequency synthesizer with a fully integrated loop filter and VCO. The VCO output is the AD9545 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.

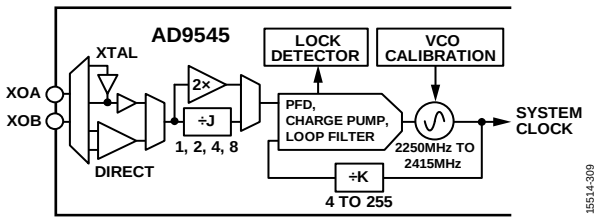


Figure 41. System Clock PLL Block Diagram

SYSTEM CLOCK INPUT FREQUENCY DECLARATION

Proper operation of the AD9545 requires the user to declare the input reference frequency to the system clock PLL. To do so, program the 40-bit unsigned integer SYSCLK reference frequency bit field in Register 0x0202 to Register 0x0206, which constitutes the nominal frequency of the system clock PLL input reference. The AD9545 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 23) or a direct path (see Figure 24). Path selection is via the enable maintaining amplifier bit via Bit D3 of Register 0x0201, 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 800 mV p-p amplitude.

Crystal Path

Select the crystal path by programming the enable maintaining amplifier bit = 1. Refer to Figure 23 for connecting a crystal resonator to the XOA and XOB pins. The C_{TUNE} capacitors shown in Figure 23 relate to C_{LOAD} and C_{STRAY}. C_{LOAD} is the specification of the crystal manufacturer and C_{STRAY} is any additional parasitic capacitance as follows:

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

where C_{LOAD} and C_{STRAY} are per the specification of the crystal manufacturer. 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 60 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 60 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 AD9545 with the aforementioned crystals, nor does Analog Devices endorse one crystal supplier over another. The AD9545 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 42 shows the proper way to connect the power limiting resistor, R_{LIMIT}, to achieve best performance.

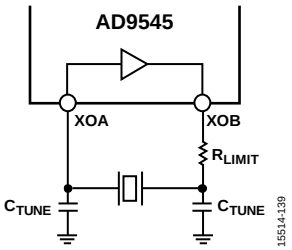


Figure 42. Power Limiting Resistor Connection

Direct Path

Select the direct path by programming the enable maintaining amplifier bit = 0. Refer to Figure 24 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 0.6 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). The external signal must exhibit a 50% duty cycle for best performance.

2× FREQUENCY MULTIPLIER

The system clock PLL provides the user with the option of doubling the reference frequency via the enable SYSCLK doubler bit in Bit D0 of Register 0x0201. Programming Logic 1 doubles the input reference frequency, which potentially reduces the PLL in band noise. Note the reference frequency must be less than 150 MHz when using the 2× frequency multiplier to satisfy the 300 MHz maximum PFD rate. The 2× frequency multiplier requires the reference input signal to have very near to 50% duty cycle; otherwise, the resulting spurious content may prevent the system clock PLL from locking.

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 AD9545. The user selects the divide ratio via the 2-bit SYSCLK input divider ratio bit field in Bits[D2:D1] of Register 0x0201. The corresponding divide value is 2^J, where J is the decimal value of the 2-bit number in the SYSCLK input divider ratio bit field.

For example, given a SYSCLK input divider ratio bit field value of 10 (binary) and J = 2 (decimal) yields a divide ratio of $2^J = 2^2 = 4$.

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 41). The feedback divide ratio has a range of 4 to 255, which the user programs via the 8-bit feedback divider ratio register (Register 0x0200). 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 as follows:

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

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 the SYSCLK locked bit in Bit D0 of Register 0x3001, where Logic 1 indicates locked and Logic 0 unlocked. The most common reason the system clock PLL fails to lock is due to the user employing the 2× frequency multiplier with a reference input clock that deviates from the 50% duty cycle.

SYSTEM CLOCK STABILITY TIMER

Because time processing blocks within the AD9545 depend on the system clock generating a stable frequency, the system clock PLL provides an indication of its 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 lock status, the user can introduce an additional hold off period via the unsigned 20-bit system clock stability period bit field in Register 0x0207 to Register 0x0209. The system clock stability period bit field constitutes units of milli-seconds (10^{-3} sec). Note there are two special values: 0 and 1,048,575. A value of 0 causes the system clock to indicate 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 causes the SYSCLK stable bit (Register 0x3001, Bit D1) to be 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. To do so, use the calibrate SYSCLK bit in Bit D2 of Register 0x2000. The calibrate SYSCLK bit is not an autoclearing bit; thus, the full programming sequence involves writing Logic 1 followed by an IO_UPDATE operation, and then writing Logic 0 followed by another IO_UPDATE operation. Furthermore, whenever the system clock PLL is in the process of calibrating, the device asserts the SYSCLK calibration busy bit in Bit D2 of Register 0x3001.

Although the AD9545 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 (below approximately 50 Hz) may cause the DPLL to unlock or experience intermittent unlock events. The classic 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 AD9545, 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 60 MHz) to the XOA and XOB pins per Figure 23, 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, and GPS) to an unused REFx or Mx input.

The stable source is the reference input to one of the DPLL channels, which enables the internal timing system in the AD9545 to benefit from both the stability of the highly stable source and the exceptional phase noise performance of the crystal resonator. This combination enables the AD9545 to apply system clock compensation to the second DPLL channel, thereby allowing that channel to use the low loop bandwidth required by the application.

REFERENCE CLOCK INPUT RECEIVERS

REFERENCE CLOCK RECEIVERS OVERVIEW

The AD9545 has four reference clock input pins supporting 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, allowing the AD9545 to handle up to four reference clock signals. 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 D0 of Register 0x0300 and Register 0x0304 (Logic 0 (default) selects single-ended mode and Logic 1 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.

SINGLE-ENDED MODE

When configured for single-ended operation, the input receivers accommodate either ac- or dc-coupled input signals

- AC-coupled interface
- DC-coupled 1.2 V CMOS
- DC-coupled 1.8 V CMOS
- Internal pull-up resistor

For programming the desired coupling, use the 2-bit REFx single-ended mode bit fields (where x is A, AA, B, or BB) in Bits[D7:D4] of Register 0x0300 and Register 0x0304. These bit fields are only effective when single-ended mode is in effect. Table 27 shows the settings for the single-ended mode bit fields.

Table 27. REFx Single-Ended Mode Bit Field Settings

REFx Single-Ended Mode (Decimal)	Description
0	AC-coupled interface
1	DC-coupled 1.2 V CMOS
2	DC-coupled 1.8 V CMOS
3	Internal pull-up resistor (disable pull-down resistor)

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 25 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 via the appropriate REFx single-ended mode bit field (Bits[D7:D4] of Register 0x0300 and Register 0x0304).

Single-Ended DC-Coupled Internal Pull-Up Resistor

Refer to Figure 27 for connecting an open-collector or open-drain driver to a reference input. Be sure to select the internal pull-up mode via the appropriate REFx single-ended mode bit field (Bits[D7:D4] of Register 0x0300 and Register 0x0304), which provides an internal pull-up resistor (~47 kΩ) to 1.2 V.

Single-Ended AC-Coupled Interface

Refer to Figure 26 to ac-couple a driver to a reference input. Be sure to select the ac-coupled interface mode via the appropriate REFx single-ended mode bit field (Bits[D7:D4] of Register 0x0300 and Register 0x0304), which provides the appropriate internal bias network.

DIFFERENTIAL MODE

When configured for differential operation, the input receivers accommodate either ac- or dc-coupled input signals

- AC-coupled
- DC-coupled
- DC-coupled LVDS

For programming the desired coupling, there are two bit fields: one for the REFA and REFAA differential inputs and one for the REFB and REFBB differential inputs. Use the REFx differential mode bit field (where x is A or B) in Bits[D3:D2] of Register 0x0300 and Register 0x0304. The REFx differential mode bit field is only effective when the corresponding differential mode is in effect.

Table 28 shows the settings for the single-ended mode bit fields.

Table 28. REFx Differential Mode Settings

REFx Differential Mode (Decimal)	Description
0	AC-coupled
1	DC-coupled
2	DC-coupled LVDS

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 below 10.24 MHz, use dc-coupled differential mode. For direct connection of a low frequency (below 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 28 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 via the appropriate REFx differential mode bit field (Bits[D3:D2] of Register 0x0300 and Register 0x0304), 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 29 to make an ac-coupled connection to an external driver. Be sure to select the ac-coupled mode via the appropriate REFx differential mode bit field (Bits[D3:D2] of Register 0x0300 and Register 0x0304).

Differential DC Coupling (>10.24 MHz)

Use Figure 30 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 via the appropriate REFx differential mode bit field (Bits[D3:D2] of Register 0x0300 and Register 0x0304).

REFERENCE DIVIDERS (R DIVIDERS)

Each reference input has a dedicated divider, R_x (where x is A, AA, B, or BB). 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 REFx R divide ratio bit fields (where x is A, AA, B, or BB) in Register 0x0400 to Register 0x0403, Register 0x0420 to Register 0x0423, Register 0x0440 to Register 0x0443, and Register 0x0460 to Register 0x0463.

The divide ratio of R_x depends on the value, R, the user programs in the REFx R divide ratio 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).

REFERENCE MONITOR

REFERENCE MONITOR OVERVIEW

The AD9545 has four independent reference monitors, one for each reference input. The A block in Figure 43 shows the REFA input along with its associated reference monitor. Figure 43 shows only the detail of the REFA input. The AA, B, and BB blocks shown in Figure 43 are associated with the REFAA, REFB, and REFBB input pins and are functionally equivalent to the A block of the diagram.

The user has the option of powering down unused references via the power-down REFx bits (where x is A, AA, B, or BB) in Bits[D3:D0] of Register 0x2001.

Each reference input has a dedicated input divider (for example, R_A in Figure 43) with a programmable divide ratio (see the Reference Dividers (R Dividers) section). Because the reference monitor is 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 time-to-digital converter (TDC) (see the Time to Digital Converter (TDC) section). The TDC delivers digital (numeric) timestamps to the reference monitor, which processes the timestamps to determine the status of the input clock signal (for example, one of the status indications is whether or not the clock signal meets predefined frequency limits). Each reference TDC can connect to either DPLL input via the digital cross point mux (see Figure 1). Even when a reference is not connected to one of the DPLLs via the cross point mux, its corresponding reference monitor connects and continues to monitor the reference signal to generate the appropriate status signals for that reference.

The AD9545 uses the status of the input reference 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 detail).

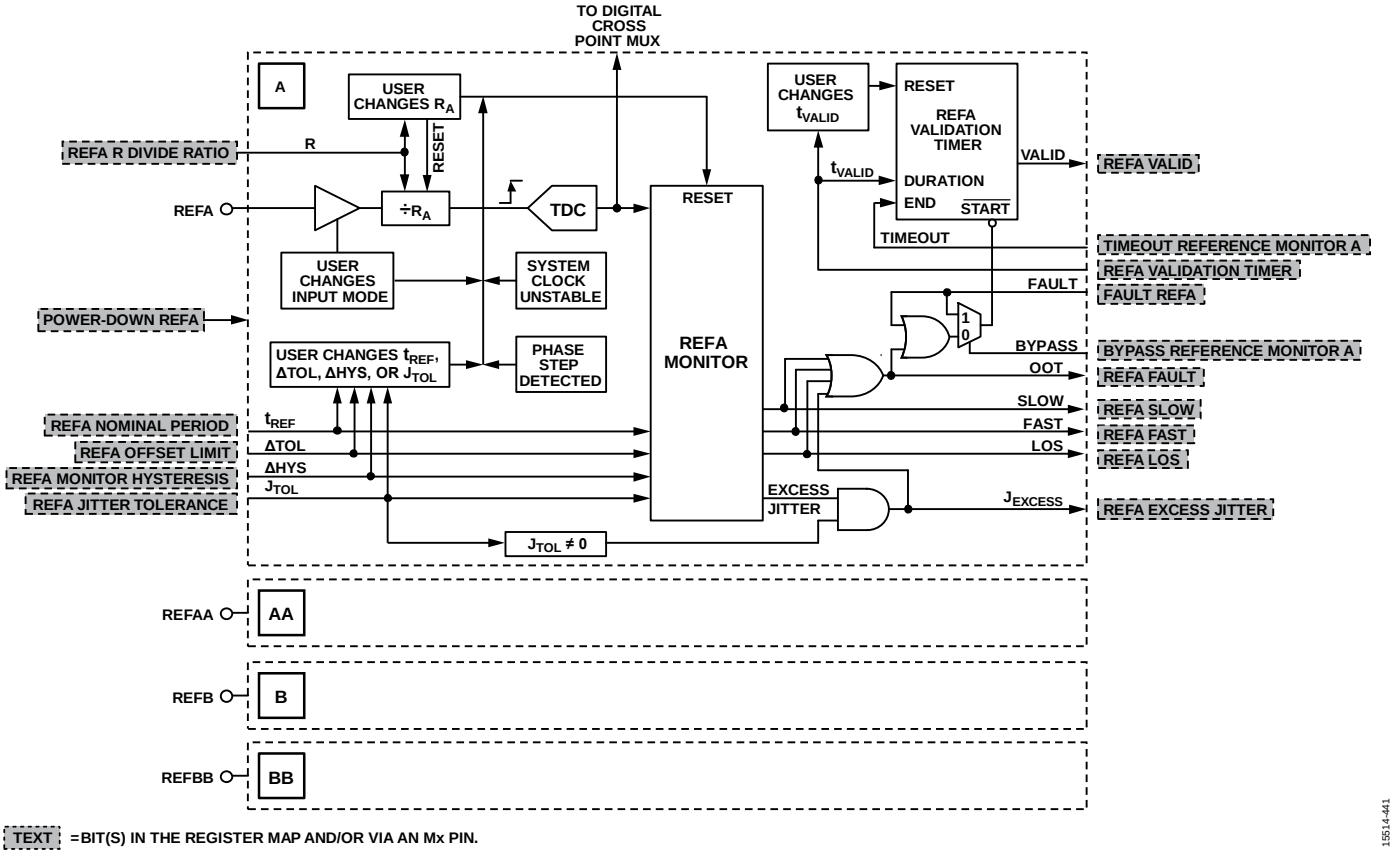


Figure 43. Reference Monitor

REFERENCE MONITOR STATE MACHINE

The REFA monitor, shown in Figure 43, is essentially a state machine. Therefore, each REFx input (where x is A, AA, B, or BB) has a dedicated state machine. The state machine uses certain input parameters (see the Reference Monitor Controls section) for performing calculations and comparisons.

The state machine operates directly on TDC time stamps, which occur with each rising edge from the output of the R-divider. As such, the state machine effectively operates on the input reference period scaled up by the division factor associated with the R-divider.

Because the ΔTOL input parameter (see Figure 43) is a proportional value rather than an absolute offset, the state machine can determine in or out of tolerance without direct knowledge of the value of the R-divider. Instead, the state machine continually observes the difference between successive time stamps and compares the statistics of those observations against ΔTOL to determine if the reference is fast, slow, absent, or exhibits excessive jitter.

REFERENCE MONITOR CONTROLS

The reference monitor requires certain advance information to function properly. The user provides this information by programming the appropriate parameters as follows.

REFx Nominal Period (t_{REF})

Each reference input has a dedicated bit field for specifying t_{REF}: the unsigned 60-bit REFx nominal period bit field (where x is A, AA, B, or BB) in Register 0x0404 to Register 0x040B, Register 0x0424 to Register 0x042B, Register 0x0444 to Register 0x044B, and Register 0x0464 to Register 0x046B. t_{REF} constitutes the nominal period of the associated input reference and represents time in units of atto-seconds (10^{-18} sec) with an upper limit of about 1 sec.

For example, 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 0x0424 to Register 0x042B with the 60-bit value corresponding to 488,281,250,000 attoseconds (0x 71).

REFx Offset Limit (ΔTOL)

Each reference input has a dedicated unsigned 24-bit REFx offset limit bit field (where x is A, AA, B, or BB) in Register 0x040C to Register 0x040E, Register 0x042C to Register 0x042E, Register 0x044C to Register 0x044E, and Register 0x046C to Register 0x046E to specify ΔTOL. ΔTOL constitutes a fractional portion of the corresponding t_{REF} value and is the maximum relative deviation the reference monitor uses for declaring a nonfaulted reference as out of tolerance. The 24-bit number represents fractional units of parts per billion (ppb) up to a maximum of approximately 17 million ppb (1.7%).

Because ΔTOL is 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 2×10^{-6} (2 ppm). That is, $\delta f = 2 \times 10^{-6}$.

From Equation 1, $\delta t = -1.999996 \times 10^{-6}$, or −1.999996 ppm, or −1999.996 ppb. Because ΔTOL is an unsigned number with integer units of ppb, ΔTOL = 2000. Therefore, program Register 0x042C to Register 0x042E with the 24-bit value corresponding to 2000 (0x0007D0 hexadecimal).

REFx Monitor Hysteresis (ΔHYS)

Each reference input has a dedicated 3-bit REFx monitor hysteresis bit field (where x is A, AA, B, or BB) in Bits[D2:D0] of Register 0x040F, Register 0x042F, Register 0x044F, and Register 0x046F. ΔHYS constitutes a scale factor per Table 29.

Table 29. Hysteresis Selection Table

REFx Monitor Hysteresis Bit Field Value	Scale Factor (SF)
0	0
1	0.03125
2	0.0625
3	0.125
4	0.25
5	0.5
6	0.75
7	0.875

The selected SF defines ΔHYS as a fractional portion of the value of ΔTOL. That is, ΔHYS is always less than or equal to ΔTOL.

$$\Delta HYS = \Delta TOL \times (1 - SF)$$

Given ΔTOL = 2,000 ppb and the selected SF value is 0.0625, determine the value of ΔHYS.

$$\Delta HYS = 2{,}000 \times (1 - 0.0625) \text{ ppb} = 1875 \text{ ppb}$$

ΔHYS applies to a faulted reference, whereas ΔTOL applies to a nonfaulted reference. The reference monitor uses t_{REF}, ΔTOL, and ΔHYS to make fault and unfault decisions with regard to a reference input, as shown in Figure 44.

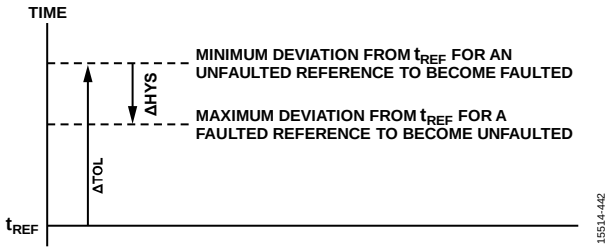


Figure 44. Reference Monitor Fault and Unfault Parameters

REFx Jitter Tolerance (J_{TOL})

Each reference input has a dedicated bit field to specify J_{TOL}: the unsigned 16-bit REFx jitter tolerance bit field (where x is A, AA, B, or BB) in Register 0x0413 to Register 0x0414, Register 0x0433 to Register 0x0434, Register 0x0453 to Register 0x0454, and Register 0x0473 to Register 0x0474. J_{TOL} represents time in units of nanoseconds (10^{-9} sec) up to a maximum of approximately

65.5 µs. Use J_{TOL} to define the maximum rms jitter the reference monitor accepts before indicating an excess jitter condition.

For example, if the expected jitter limit is 75 ns rms, then J_{TOL} = 75 (0x004B hexadecimal).

Programming J_{TOL} = 0 disables the J_{EXCESS} status bit (see Figure 43). Even though J_{TOL} = 0 disables the status bit, the reference monitor continues to assess jitter as part of its period estimation process.

REFx Validation Timer (t_{VALID})

Each reference input has a dedicated bit field to specify t_{VALID}: the unsigned 20-bit REFx validation timer bit field (where x is A, AA, B, or BB) in Register 0x0410 to Register 0x0412, Register 0x0430 to Register 0x0432, Register 0x0450 to Register 0x0452, and Register 0x0470 to Register 0x0472. t_{VALID} represents time in units of milliseconds up to a maximum of approximately 1048 sec (approximately 17.5 min). See the Reference Validation section for details on the validation timer.

For example, a validation time of 10 min requires:

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

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

Timeout Reference Monitor x (Timeout)

Each reference input has a dedicated timeout control bit to force the timing out of the reference validation process. Timeout control is via the Timeout Reference Monitor x bit (where x is A, AA, B, or BB) in Bits[D3:D0] of Register 0x2002. When the validation timer is in the process of validating a reference (see the Reference Validation section), programming timeout = 1 forces expiration of the validation timer, thereby indicating the valid control bit = 1 (see Figure 43). When timeout = 0 (default), the status of the valid control bit depends on the normal operation of the validation timer.

Bypass Reference Monitor x (Bypass)

Each reference input has a dedicated bypass control bit to allow bypassing of the valid/invalid decision in the reference monitor. Bypass control is via the Bypass Reference Monitor x bit (where x is A, AA, B, or BB) in Bits[D3:D0] of Register 0x2004. The bypass function does not actually bypass the reference monitor itself but only its ability to validate a reference.

Setting bypass = 0 (default) allows the out of tolerance (OOT) decision from the REFx monitor block to propagate to the validation timer (see Figure 43), which constitutes normal reference monitor operation. That is, reference validation is completely under the control of the reference monitor (except when the user programs fault = 1 (see Figure 43), which overrides an in tolerance decision by the REFx monitor block).

When bypass = 1, the fault bit becomes a substitute for the out of tolerance decision to the validation timer, placing reference validation under direct user control but still giving the user access to the OOT decisions of the reference monitor via the REFx fault, REFx fast, REFx slow, and REFx LOS bits in Register 0x3005 to Register 0x3008 (see Figure 43).

Fault REFx (Fault)

Each reference input has a dedicated fault control bit to force reference validation decisions. Fault control is via the fault REFx bit (where x is A, AA, B, or BB) in Bits[D3:D0] of Register 0x2003.

When bypass = 0, programming fault = 1 forces a reference to be invalid regardless of the in or out of tolerance decision of the reference monitor. However, the user still has access to the OOT decisions (see Figure 43). The user cannot force a reference to be valid when bypass = 0.

When bypass = 1, the fault bit allows the user to force a reference to be valid (fault = 0) or invalid (fault = 1). However, the user still has access to the OOT decisions of the reference monitor (see Figure 43).

Table 30 shows a summary of the interaction between the bypass and fault functions.

Table 30. Reference Monitor Bypass And Fault Summary

Bypass	Fault	Function
0	0	The reference monitor has complete control over REFx validation
0	1	Force REFx invalid
1	0	Force REFx valid
1	1	Force REFx invalid

Regardless of the programmed state of the bypass and fault bits, the user always has access to the OOT decisions of the reference monitor (see Figure 43).

In Table 30, the bypass = 0 and fault = 1 conditions constitute normal reference monitor operation, yet forces the associated reference to be invalid.

Choosing this condition can be useful when a reference signal must be traceable to a primary timing source and the user has external knowledge that the source is no longer traceable. In this case, the reference may still satisfy the in tolerance parameters so that the reference monitor sees the reference as valid. The user has the option of forcing the reference to be invalid by programming fault = 1 even though the reference monitor indicates an in tolerance condition.

In Table 30, the bypass = 1 and fault = 0 conditions and the bypass = 1 and fault = 1 conditions provide the user with total control over declaring a reference as valid or invalid. This control can be useful when a reference signal is of the gapped clock variety. In such a case, the reference monitor cannot reliably discern an in or out of tolerance condition, which can cause an otherwise valid reference to be invalid from the point of view of the reference monitor. The two aforementioned conditions provide a means to declare a reference as valid (bypass = 1 and fault = 0) or invalid (bypass = 1 and fault = 1) at the discretion of the user.

MONITOR TIME BASE

The reference monitor uses the internal system clock as the time base for measuring the input reference frequency. Because the reference monitor relies on the system clock as its time base, it must verify the system clock is locked and stable.

As such, the reference monitor remains in a reset state until the system clock status is locked and stable, at which time the reference monitor state machine activates.

The AD9545 has the ability to perform system clock compensation (see the System Clock Compensation section).

The user has the option to enable the reference monitor to use the compensated system clock for estimating the reference period (assumes programming of the appropriate system clock compensation controls).

REFERENCE PERIOD JITTER ESTIMATION

The main purpose of the reference monitor is to verify the reference period satisfies the requirements per the t_{REF}, ΔTOL, ΔHYS, and J_{TOL} parameters provided by user input via the register map. Ideally, if the reference signal is completely jitter free, the reference monitor can verify the reference period by simply measuring the time between successive rising edges. However, real-world signals are inherently noisy and possess timing jitter.

Jitter introduces uncertainty in the time measurement between successive rising edges of the reference signal. Not only is this true for the reference signal, but also for the internal time base the monitor uses to make its measurements. Both jitter sources compromise the ability of the monitor to determine when the reference is truly in or out of tolerance. That is, jitter dilutes the confidence of the monitor in making an accurate decision. The device has no inherent knowledge of the magnitude or spectral distribution of the jitter present on a particular reference. Furthermore, the jitter characteristics can vary over time.

Because the monitor uses numeric time stamps to measure the reference period, it can estimate the variance of the reference signal as it observes period samples. The variance estimate gives the monitor some idea of the actual jitter present on the reference signal. By comparing the variance estimate with the ΔTOL value provided by the user, the monitor determines how many period samples are needed to arrive at a period estimate with sufficient confidence to make an in or out of tolerance decision. This algorithm enables the monitor to perform a high confidence period estimate using a nearly optimal minimum observation time.

Another benefit of this statistical approach to monitoring the reference period is the monitor does not need to measure the actual period of the reference. It only needs to estimate the deviation of the period relative to the scaled T_{REF} value to a sufficiently high degree of confidence. The reference monitor algorithm has the necessary confidence requirement built into its design.

Because the reference monitor is able to assess the amount of reference jitter present via its variance estimate, it has the ability to determine if there is excessive jitter on the input signal and indicate such via the J_{EXCESS} status indicator. J_{EXCESS} status is via the REFx excess jitter bit (where x is A, AA, B, or BB) in Bit D2 of Register 0x3005 to Register 0x3008. Excessive jitter is jitter greater than the predefined threshold limit as defined by the input of the user determined J_{TOL} value (see the Reference Monitor Controls section).

REFERENCE MONITOR DECISION TIME

The reference period estimation algorithm takes into account the value of ΔTOL specified by the user (see the Reference Monitor Controls section) and the actual jitter present on the reference signal. In general, it is the ratio of the square root of the measured jitter variance to ΔTOL that governs the decision time of a marginally out of tolerance reference signal. The smaller the variance of the measured jitter, the less averaging that is necessary and the shorter the decision making time. The key point is that it is both the value of ΔTOL and the actual jitter present that governs the decision time for declaring 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, it has little effect on the decision time when the reference period is much greater than the scaled t_{REF} value (that is, t_{REF} divided by the R-divider factor) or when the reference signal disappears completely. The reference monitor relies on time stamps generated by reference signal edges to perform a reference period estimation. Thus, a catastrophic loss of the reference signal is problematic, as the reference signal edges cease to exist, thereby stalling the period estimation process of the reference monitor.

However, because the reference monitor knows the scaled t_{REF} value, it can rely on its local time base to watch for the next expected TDC input edge (note the reference monitor allows 15% excess margin for its expectation of the next reference input edge). If the reference monitor fails to observe the next reference input edge within a period of 115% of the scaled t_{REF} value, then the monitor declares a loss of signal (LOS), see Figure 43. LOS status is via the REFx LOS bit (where x is A, AA, B, or BB) in Bit D5 of Register 0x3005 to Register 0x3008.

After the first assertion of the LOS status bit, the LOS bit toggles on each subsequent occurrence of the expected period of the reference. Therefore, the LOS bit is not a reliable indicator of the loss of the reference signal. Instead, the user relies on the OOT signal as an indicator of the loss of the signal.

REFERENCE VALIDATION

When a reference is in a faulted state (OOT = 1, see Figure 43), the reference monitor continues to monitor the reference signal for an in tolerance condition. When the reference returns to in tolerance, OOT = 0, indicates a nonfaulted condition. OOT status is via the REFx fault bit (where x is A, AA, B, or BB) in Bit D3 of Register 0x3005 to Register 0x3008.

Some applications require a reference to be in a nonfaulted condition for a prescribed period before the reference can be considered valid. To accommodate these applications, the reference monitor includes a programmable validation timer. When a reference transitions from a faulted state to a nonfaulted state, it indicates a start event for the validation timer, which

begins counting down. Upon expiration of the timer, valid = 1 (see Figure 43) indicates the reference is available for use. The valid status is via the REFx valid bit (where x is A, AA, B, or BB) in Bit D4 of Register 0x3005 to Register 0x3008.

The user programs the validation timer period associated with the corresponding reference via t_{VALID} (see Figure 43). Note that changing t_{VALID} resets the validation timer, which invalidates a previously valid reference but does not fault the reference (because OOT status is unaffected by resetting the validation timer).

It is important to note the validation timer stops counting down whenever the reference status becomes faulted (OOT = 1, see Figure 43). A subsequent start event causes the timer to reinitiate a countdown from the programmed t_{VALID} value. Therefore, the reference does not attain valid status unless the reference remains in tolerance 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 Controlssection and Figure 43). 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.

REFERENCE MONITOR RESET

The reference monitor associated with a particular reference resets under certain conditions. A reset restarts the period estimation process for the reference monitor. The following list indicates the conditions that cause the reference monitor to reset (also shown in Figure 43):

- When the system clock status indicates unstable
- When the user changes: the mode of the input receiver, the R-divider value, the t_{REF} value, the Δt_{REF} value, the t_{TOL} value, or the t_{HYS} value
- Optionally, when the reference experiences a significant phase step as determined by the DPLL (assuming the user enables phase step detection via the enable step detect reference fault bit in Bit D7 of Register 0x2105 and Register 0x2205).

The last item requires that the user program a nonzero value in the Profile x phase step threshold bit field. This bit field occupies a block of consecutive register addresses with each block starting at Address 0x080A, Address 0x082A, Address 0x084A, Address 0x086A, Address 0x088A, Address 0x08AA, Address 0x08CA, and Address 0x08EA. See the Phase Step Limit section for details regarding the phase step limit feature.

REFERENCE DEMODULATOR

REFERENCE DEMODULATOR OVERVIEW

The AD9545 has four reference demodulators, one for each reference input, REFA, REFAA, REFB, and REFBB. When enabled, the demodulators extract embedded modulation events from their associated input carrier clock signal. These modulation events are essentially phase variations appearing on the falling edge of certain input clock cycles. The primary intent of the demodulators is to enable the detection of an embedded clock signal originating from the output distribution of another AD9545 (see the Distribution Embedded Output Clock Modulation section for more detail on embedded clock modulation).

Figure 45 shows a basic block diagram of the reference demodulator. Although Figure 45 specifically shows the demodulator associated with the REFA input, the diagram applies similarly to the REFAA, REFB, and REFBB inputs. A reference input configured for differential operation only has access to one demodulator: Demodulator A for the REFA and REFAA differential input and Demodulator B for the REFB and REFBB differential input (see Figure 1).

The input clock signal at a REFx input (where x is A, AA, B, or BB) passes through the reference receiver and drives the clock input of the reference divider, the normal signal path when the REFx input signal does not carry an embedded clock signal.

The output of the receiver also drives the demodulator input. When enabled, the demodulator extracts the embedded clock signal and uses it to synchronize the reference divider.

The demodulator expects a reference input signal containing modulation events as described in the Distribution Embedded Output Clock Modulation section. In general, the modulation constitutes a lower frequency clock signal embedded within the higher frequency carrier. The demodulator extracts the embedded clock and the polarity of the modulation events (optional).

The demodulated clock signal is available to the user via a suitably configured Mx pin.

Seven control functions are associated with each demodulator accessible via the register map, as follows:

- Enable
- Delay
- Sensitivity
- Polarity
- Auto polarity detection
- Persistence
- Bandwidth

DEMODULATOR ENABLE

To enable the demodulator use the enable REFx demodulator bit (where x is A, AA, B, or BB) in Bit D3 of Register 0x0302 to Register 0x0303 and Register 0x0306 to Register 0x0307. Logic 0 (default) disables the demodulator, while Logic 1 enables it. Note the demodulator requires at least an 8:1 ratio between the frequency of the reference input and the frequency of the embedded modulation events.

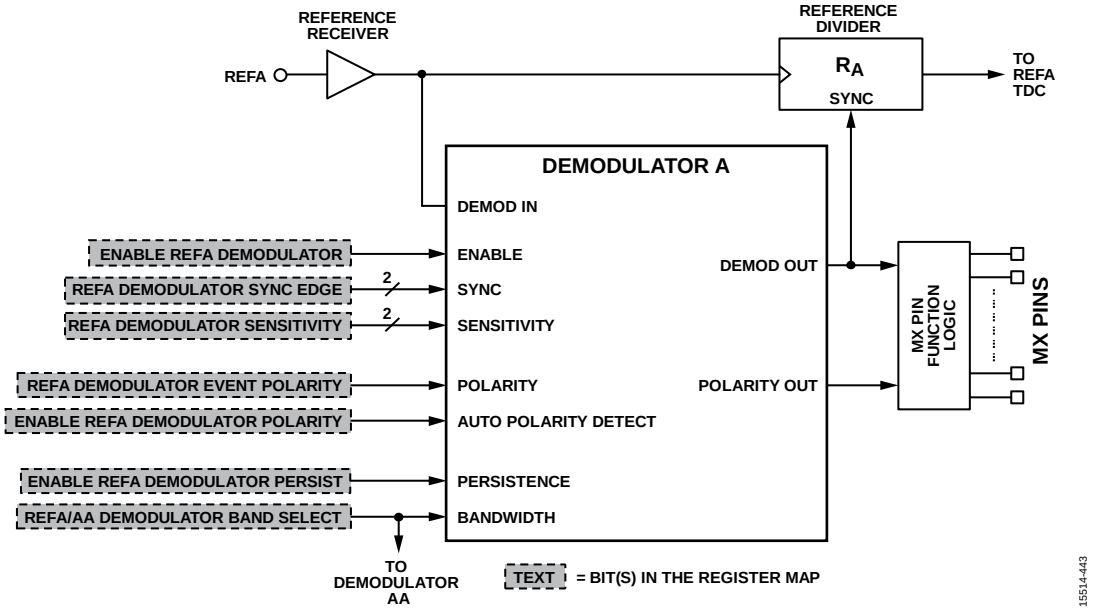


Figure 45. Reference Demodulator

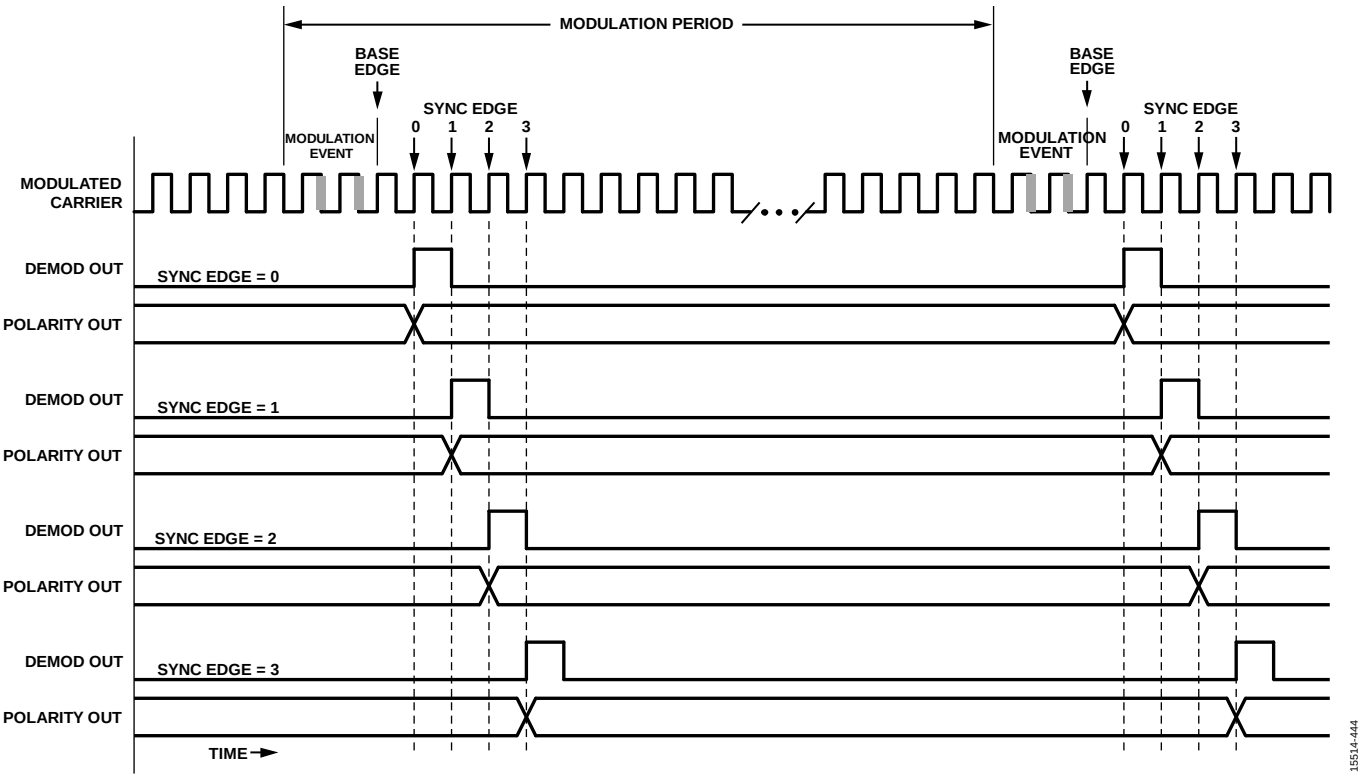


Figure 46. Demodulation Synchronization

DEMODULATOR DELAY

The demodulated signal is a single pulse with a duration equal to the period of the input carrier clock as shown by the demod out traces in Figure 46. When the demodulator identifies a modulation event within the carrier signal, it denotes the next rising edge of the carrier as the modulation base edge. The position of the pulse can be any one of four subsequent carrier clock periods after the base edge. The pulse position is programmable and allows some control over the latency through the demodulator. To specify the pulse position use the 2-bit unsigned REFx demodulator sync edge bit field (where x is A, AA, B, or BB) in Bits[D5:D4] of Register 0x0302 to Register 0x0303 and Register 0x0306 to Register 0x0307. The value of this bit field, 0 (default), 1, 2, or 3, corresponds to the desired sync edge in Figure 46 (REFx demodulator sync edge = 0 is not valid).

DEMODULATOR POLARITY

Modulation events span two clock periods of the carrier signal as detailed in the Distribution Embedded Output Clock Modulation section. Normal polarity constitutes the first pulse having greater than 50% duty cycle with an optional second pulse having less than 50% duty cycle, that is (50 + x)% followed by an optional (50 − x)%. Alternate polarity is the opposite: a less than 50% duty cycle pulse optionally followed by a more than 50% duty cycle pulse.

The demodulator is most sensitive when it is aware of the modulation polarity in advance. The user can take advantage of this sensitivity by informing the demodulator of the polarity of the embedded modulation events via the REFx demodulator event polarity bit (where x is A, AA, B, or BB) in Bit D2 of Register 0x0302 to Register 0x0303 and Register 0x0306 to Register 0x0307. Logic 0 (default) indicates the embedded modulation exhibits normal polarity, whereas Logic 1 indicates alternate polarity.

The REFx demodulator event polarity bit is only meaningful when automatic polarity detection is not in effect.

AUTOMATIC POLARITY DETECTION

The demodulators are capable of automatically detecting the polarity of modulation events. To enable this feature, use the enable REFx demodulator polarity bit (where x is A, AA, B, or BB) in Bit D7 of Register 0x0302 to Register 0x0303 and Register 0x0306 to Register 0x0307. Logic 0 (default) disables automatic polarity detection, whereas Logic 1 enables it.

When enable REFx demodulator polarity = 1, the demodulator detects the polarity (normal or alternate) of each modulation event and latches the result (normal polarity is Logic 0, alternate polarity is Logic 1) to polarity out (see Figure 45). When enable REFx demodulator polarity = 0, the demodulator does not produce a polarity out signal.

Automatic polarity detection slightly reduces the sensitivity of the demodulator. Therefore, it is possible for the demodulator to detect low magnitude modulation events reliably with automatic polarity detection disabled, but exhibit demodulation errors with auto polarity detection enabled.

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 the 2-bit unsigned REFx demodulator sensitivity bit field (where x is A, AA, B, or BB) in Bits[D1:D0] of Register 0x0302 to Register 0x0303 and Register 0x0306 to Register 0x0307.

The demodulator defaults to its most sensitive setting, REFx demodulator sensitivity = 0. Increasing the value to 1 decreases the sensitivity with 3 constituting the least sensitive setting. Decreased sensitivity implies the need for more robust modulation events (that is, larger changes in pulse width) for reliable demodulation.

DEMODULATOR PERSISTENCE

In some applications, the modulation events may not be continuous. The modulation events may occur in a burst, for example, as a sequence of 50 modulation events followed by no modulation. In this case, the demod out signal (see Figure 45) appears as a burst of 50 pulses.

The demodulator has a persistence feature; it is capable of identifying the modulation period and then generating the requisite pulses at the demod out output even after the input modulation events cease.

To enable the persistence feature, use the enable REFx demodulator persist bit (where x is A, AA, B, or BB) in Bit D6 of Register 0x0302 to Register 0x0303 and Register 0x0306 to Register 0x0307. Logic 0 (default) disables persistence, whereas Logic 1 enables it.

The persistence feature requires at least five consecutive modulation events to produce a persistent demod out signal. As such, the recommendation is to avoid using the persistence feature when the modulation period is greater than 100 ms (that is, modulation frequencies below 10 Hz).

DEMODULATOR BANDWIDTH

The demodulator is capable of handling input reference carrier frequencies from 1 MHz to 180 MHz. The demodulator splits this into two operating ranges: 1 MHz to 30 MHz carrier frequencies and 30 MHz to 180 MHz carrier frequencies.

Use the REFA/REFAA demodulator band select bit or the REFB/REFBB demodulator band select bit in Bit D0 of Register 0x0301 and Register 0x0305. Logic 0 (default) selects the lower frequency range, whereas Logic 1 selects the upper frequency range. Note that the bandwidth setting applies to demodulator pairs. That is, one bit affects the demodulator for REFA and REFAA, and the other bit affects the demodulator for REFB and REFBB.

DISTRIBUTION CLOCK OUTPUT DRIVERS

DISTRIBUTION CLOCK OUTPUT DRIVERS OVERVIEW

The AD9545 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 OUT0 group, while PLL1 can clock the OUT1 group.

The Output 0 (OUT0) 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 (OUT1) 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

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 a current source and a channel divider associated with it. The user has the option to set both subgroup outputs as either source current or sink current via the enable OUTxy HCSL bit (where x is 0 or 1 and y is A, B or C) in Bit D0 of Register 0x10D7 to Register 0x10D9 and Register 0x14D7 to Register 0x14D8. Logic 0 (default) selects current sink mode, while Logic 1 selects current source (or HCSL) mode.

To control the magnitude of the current, use the OUTxy driver current bit pairs (where x is 0 or 1 and y is A, B or C) in Bits[D2:D1] of Register 0x10D7 to Register 0x10D9 and Register 0x14D7 to Register 0x14D8. Table 31 relates the bit field value to the magnitude of the driver current.

Table 31. Output Driver Current

OUTxy Driver Current Bit Field Value	Output Current (mA)
0	7.5 (default)
1	12.5
2	15
3	Not applicable

OUTPUT MODE CONTROL

The user has control over how each subgroup pin pair relates to the channel dividers of the subgroup via the OUTxy driver mode bit field (where x is 0 or 1 and y is A, B, or C) in Bits[D4:D3] of Register 0x10D7 to Register 0x10D9 and Register 0x14D7 to Register 0x14D8 per Table 32.

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. Figure 32 shows an example of single-divider differential mode.

In single-divider single-ended mode, the clock signals at the OUTxyP and OUTxyN pins are in phase relative to one another. Figure 37 shows an example of single-divider single-ended mode.

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. Figure 39 shows an example of dual-divider single-ended mode.

Table 32. Driver Output Mode

OUTxy Driver Mode Bit Field Value	Output Driver Mode
0	Single-divider differential (default)
1	Single-divider single-ended
2	Dual-divider single-ended
3	Not applicable

OUTPUT DRIVER CONFIGURATIONS

Table 33 shows a summary of the pertinent driver control bit field names and corresponding registers (where x is 0 or 1 and y is A, B, or C). The various driver control settings have corresponding driver termination requirements.

Table 33. Output Driver Control Summary

Bit Field Name	Registers
Enable OUTxy HCSL	Register 0x10D7, Bit 0 to Regsiter 0x10D9, Bit 0 Register 0x14D7, Bit 0 to Register 0x14D8, Bit 0
OUTxy Driver Current	Register 0x10D7, Bits[2:1] to Register 0x10D9. Bits[2:1] Register 0x14D7, Bits[2:1] to Register 0x14D8, Bits[2:1]
OUTxy Driver Mode	Register 0x10D7, Bits[4:3] to Register 0x10D9, Bits[4:3] Register 0x14D7, Bits[4:3] to Register 0x14D8, Bits[4:3]

Each pin of an output pin pair (OUTxyP/OUTxyN) has an independent driver capable of either sourcing or sinking current (via the enable OUTxy HCSL bit). Note, however, that the source/sink selection applies to both current sources of an output pin pair (that is, either both pin drivers source current or both pin drivers sink current per the enable OUTxy HCSL bit). The current source (or sink) of the individual driver is either on or off, depending on the logic level it receives from the Qxy or Qxyy divider (0 = off, 1 = on). the Qxy divider has normal and inverted logic outputs, which means when the Qxy divider

connects to both drivers, one current source is on and the other is off, allowing a differential output signal.

Differential HCSL Output

To drive a standard HCSL receiver, refer to Figure 32. This configuration requires programming the driver settings as follows: enable OUTxy HCSL = 1 (source current), OUTxy driver current = 3 (15 mA), and OUTxy driver mode = 0 (single-divider differential). 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 31. This configuration requires programming the driver settings as follows: enable OUTxy HCSL = 1 (source current), OUTxy driver current = 0 (7.5 mA) and OUTxy driver mode = 0 (single-divider differential). 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 34. This configuration requires programming the driver settings as follows: enable OUTxy HCSL = 0 (sink current), OUTxy driver current = 0 (7.5 mA), and OUTxy driver mode = 0 (single-divider differential). 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 31). The additional termination also requires increasing the drive current to 15 mA (OUTxy driver current = 2). 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 35, which exhibits the equivalent of a 50 Ω pull-up to 1.42 V on each output pin. This configuration requires programming the driver settings as follows: enable OUTxy HCSL = 0 (sink current), OUTxy driver current = 0 (7.5 mA), and OUTxy driver mode = 0 (single divider differential). 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 31), as well as increasing the drive current to 15 mA (OUTxy driver current = 2). 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 like receiver that can handle boosted signal swing, use a Thevenin equivalent termination per Figure 36, 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: enable OUTxy HCSL = 0 (sink current), OUTxy driver current = 2 (15 mA), and OUTxy driver mode = 0 (single-divider differential). 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 33. When using a 50 Ω pull-up resistor to 1.2 V, this configuration requires programming the driver settings as follows: enable OUTxy HCSL = 0 (sink current), OUTxy driver current = 0 (7.5 mA), and OUTxy driver mode = 0 (single-divider differential). 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, increase the drive current to 15 mA (OUTxy driver current = 2). 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 37. This configuration requires programming the driver settings as follows: enable OUTxy HCSL = 1 (source current) and OUTxy driver mode = 1 (single-divider, single-ended). 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) 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 39. This configuration requires programming the driver settings as follows: enable OUTxy HCSL = 1 (source current) and OUTxy driver mode = 2 (dual-divider, single-ended). 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 D2 of Register 0x2101 allows the user to reset all the driver pairs of Channel 0. Bit D2 of Register 0x2201 allows the user to reset all the driver pairs of Channel 1. The user can reset individual driver pairs via Bit D5 of Register 0x2102 to Register 0x2104 and Register 0x2202 to Register 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 PLL Channel 0 or Channel 1 use the mute all Channel x drivers bit (Logic 0 by default) in Bit D1 of Register 0x2101 and Register 0x2201. To mute individual drivers use the dedicated mute OUTxy bits (where x is 0 or 1 and y is A, AA, B, BB, C, or CC) in Bits[D3:D2] of Register 0x2102 to Register 0x2104 and Register 0x2202 to Register 0x2203. The mute OUTxy bits are Logic 1 (mute) by default. The mute bits allow the user to turn off the clock signal at the output pins without affecting the input clock signal to the driver.

Automatic Output Unmuting

Automatic unmuting works in conjunction with the autosync function (see the Autosync Trigger section). Part of the synchronization sequence involves synchronously unmuting the output drivers. The AD9545 provides the user options for the timing of the unmuting of the output drivers.

To select the automatic unmute conditions, use the 2-bit DPLLx autounmute mode bit field (where x is 0 or 1) in Bits[D1:D0] of Register 0x10DC and Register 0x14DC. Table 34 shows the autounmute conditions invoked by these bits. Note that autounmute control applies to PLL Channel 0 and PLL Channel 1, independently.

Table 34. Autounmute Conditions

DPLLx Autounmute Mode Bit Field Value (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 DPLLx autounmute mode = 0 (default), the drivers unmute immediately upon release of the sync request. When DPLLx autounmute mode = 1, the drivers do not unmute until release of the sync request and subsequent activation of a hitless/zero delay profile. When DPLLx autounmute mode = 2, the drivers do not unmute until release of the sync request and subsequent phase lock of the appropriate PLL channel (assuming a hitless/zero delay profile). When DPLLx autounmute mode = 3, the drivers do not unmute until release of the sync 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 autounmuting feature on a per output basis. The mask OUTxy autounmute bits (Logic 0 by default) accomplish this function (where x is 0 or 1 and y is A, AA, B, BB, C, or CC) in Bits[D7:D2] of Register 0x10DC and Bits[D5:D2] of Register 0x14DC.

After an automute event occurs for a particular PLL channel, the user can clear the automute state by setting the associated Channel x automute clear bit (Logic 0 by default) in Bit D4 of Register 0x2107 and Register 0x2207. However, this action is generally unnecessary, because the device automatically clears the automute state whenever the user changes the DPLLx autounmute mode bit field and issues an IO_UPDATE operation.

DISTRIBUTION DIVIDERS (Q DIVIDERS)

DISTRIBUTION DIVIDERS OVERVIEW

There are five pairs of Q dividers corresponding 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, which implement the various features of the Q dividers. Figure 47 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 the location of the 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 sync request (see the Distribution Output Clock Synchronization section following a power-up or reset to initialize the Q dividers.

Q DIVIDER CLOCK SOURCE SELECTION

Each pair of Q dividers shares a common clock input (see Figure 47) that originate from one of two sources:

- From the output of the associated APLL channel of the Q divider.
- From the system clock divided by three (~800 MHz).

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 divided by three, which is useful for clocking certain peripheral devices (a microprocessor, for example). In this scenario, the user can employ an optional external 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).

The enable SYSCLK Qxy bits (where x is 0 or 1 and y is A, B, or C) in Bits[D3:D1] of Register 0x10DA and Register 0x14DA select the Q divider clock source for each Q divider pair. 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.

When the AD9545 is configured with the system clock as the clock source for a Q divider pair, use the enable SYSCLK sync mask bit in Bit D0 of Register 0x10DA and Register 0x14DA to prevent those particular dividers from being resynchronized every time a sync trigger occurs (see the Distribution Output Clock Synchronization section). When this bit is set, any of the Q dividers associated with a particular PLL channel and having the system clock as their clock source are 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. To do so, initiate a synchronization sequence (see the Distribution Output Clock Synchronization section), then assert the enable SYSCLK sync mask bit. To reconfigure the affected Q dividers by changing the divide ratio, for example, the user must first clear the enable SYSCLK sync mask bit.

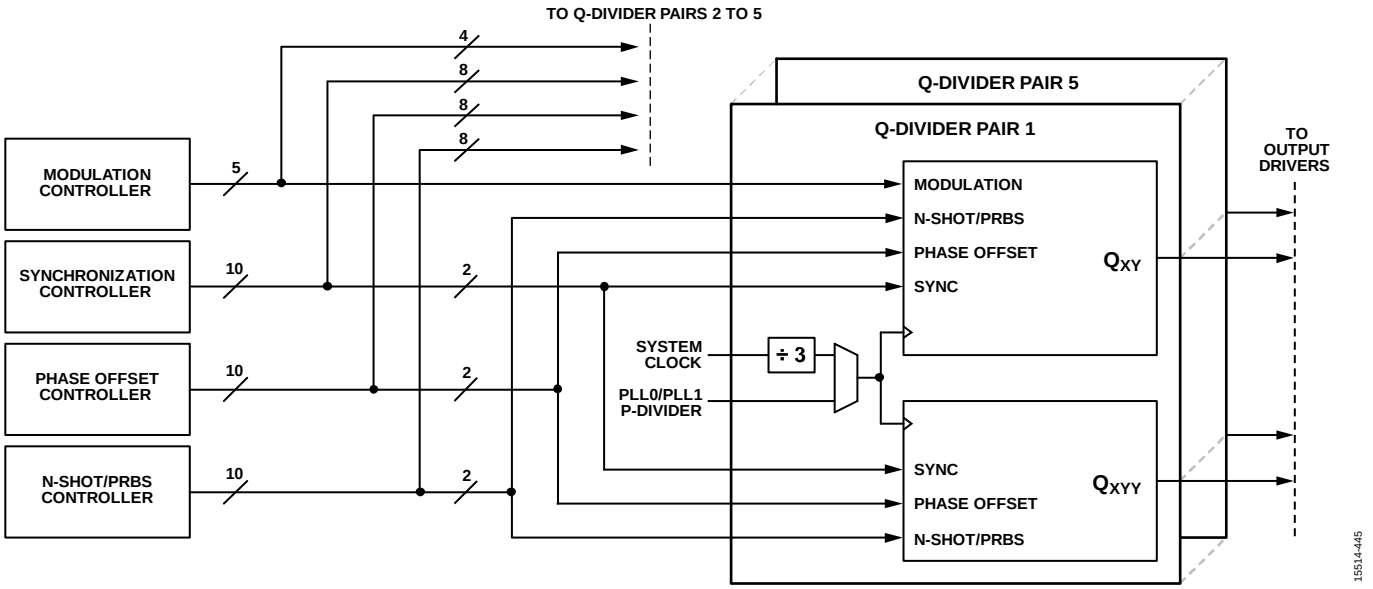


Figure 47. Functional Block Diagram of Q Dividers and Controllers

INTEGER DIVISION

The Q divider divide ratio (Q) depends on the 32-bit unsigned Qxy divider ratio bit fields (where x is the channel (0 or 1) and y is the Q divider (A, AA, B, BB, C, or CC)) in registers starting with Address 0x1100, Address 0x1109, Address 0x1112, Address 0x111B, Address 0x1124, Address 0x112D, Address 0x1500, Address 0x1509, Address 0x1512, and Address 0x151B. There are 10 occurrences of this bit field, one for each Q divider.

To program the divide ratio, write the value of Q into the corresponding Qxy divider ratio bit field and then set the IO_UPDATE bit. The range of the Q divider is 1 to 4,294,967,295.

For example, for a desired divide ratio of Q = 1,000,000. The required bit field value is 1,000,000 (0x000F4240 hexadecimal).

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 the enable Qxy half divide bit (where x is the 0 or 1 and y is the A, AA, B, BB, C, or CC).

The enable Qxy half divide bit resides in Bit D5 of Register 0x1108, Register 0x1111, Register 0x111A, Register 0x1123, Register 0x112C, Register 0x1135, Register 0x1508, Register 0x1511, Register 0x151A, and Register 0x1523.

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

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

Q DIVIDER RESET

Each Q divider has a dedicated reset bit, reset Qxy (where x is 0 or 1 and y is A, AA, B, BB, C, or CC in D1 of Register 0x2102 to Register 0x2104 and Register 0x2202 to Register 0x2203. The reset Qxy 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).

Q DIVIDER CONSTRAINTS

Changing the divide ratio of any one of the Q dividers invokes a synchronization process (see the Autoreconfiguration Sync 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 of 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/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 (see the Internal Zero Delay (Hitless) Mode section in the Frequency Translation Loops section). This feedback mechanism is via the hitless/zero delay feedback and sync block of the N-shot/PRBS controller in Figure 48. The 5-bit internal/external zero delay feedback path bit field resides in Bits[D4:D0] of Register 0x1202, Register 0x1222, Register 0x1242, Register 0x1262, Register 0x1282, Register 0x12A2, Register 0x1602, Register 0x1622, Register 0x1642, Register 0x1662, Register 0x1682, and Register 0x16A2, which selects the desired Q divider for feedback to the DPLL N-divider.

DISTRIBUTION PHASE OFFSET CONTROL

OUTPUT PHASE OFFSET OVERVIEW

The phase offset controller (see Figure 47) governs the application of phase offsets to the individual Q dividers. The controller implements two categories of phase offset:

- Initial phase offset
- Subsequent phase offset

An initial phase offset applies after the device is powered up or reset and the user issues a sync request that is completed (see the Distribution Output Clock Synchronization section). The completed sync request results in the establishment of the initial phase offset. The value of the initial phase offset is per the 33-bit unsigned Qxy phase bit field associated with each Q divider (where x is 0 or 1 and y is A, AA, B, BB, C, or CC) in registers starting with Address 0x1104, Address 0x110D, Address 0x1116, Address 0x111F, Address 0x1128, Address 0x1131, 0x1504, Address 0x150D, Address 0x1516, and Address 0x151F. There are 10 occurrences of this bit field, one for each Q divider.

Subsequent phase offsets apply for all subsequent sync requests (that is, any sync requests occurring after the initial power-up or reset sync request). See the Distribution Output Clock Synchronization section regarding sync requests.

Iinitial and subsequent phase offsets require the enable Qxy pulse width control bit of the Q divider to be Logic 0.

INITIAL PHASE OFFSET

The initial phase offset of a particular Q divider is dependent on the value of its corresponding Qxy divider ratio bit field and enable Qxy half 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 Qxy divider ratio and Enable Qxy half divide bit fields see the Distribution Dividers (Q Dividers) section.

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

$$E = (2 \times Qxy\ Divider\ Ratio) + Enable\ Qxy\ Half\ Divide$$
$$= 2 \times Q_N$$

where:
E is the total number of input edges per output period of the Q divider.
Q_N is the complete divide factor (for example, 101.5) of a specific Q divider.

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

Use the Qxy phase bit field to program the initial phase offset. The value of the Qxy phase bit field represents phase offset (θ, in degrees) as a fraction of E.

$$\theta = 360° \times (Qxy\ Phase/E)$$

This equation implies that the value of the Qxy phase bit field must be less than E (that is, the range of Qxy phase is 0 to E − 1). Note that programming an invalid Qxy phase value results in the phase offset controller taking no action other than setting one of the DPLLx phase control error bits associated with the corresponding Q divider (where x is 0 or 1) in Bits[D5:D0] of Register 0x310E and Bits[D3:D0] of Register 0x320E. 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 DPLLx phase control error bits on a per channel basis. That is, the logical OR of the six DPLL0 phase control error bits associated with Channel 0 or the logical OR of the four DPLL1 phase control error bits associated with Channel 1.

SUBSEQUENT PHASE OFFSETS

Subsequent phase offsets involve writing a new phase offset value to the appropriate Qxy phase bit field 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 Qxy phase bit field 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 phase replaces the relatively large instantaneous phase step of a phase adjustment with a series of small phase steps, which reduces 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, it sets the corresponding DPLLx phase slew active bit (where x is 0 or 1) in Bits[D5:D0] of Register 0x310D and 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 DPLLx phase slew active bits on a per channel basis. That is, the logical OR of the six DPLL0 phase slew active bits associated with Channel 0 or the logical OR of the four DPLL1 phase slew active bits associated with Channel 1.

Maximum Phase Slew Step Size

During the execution of a subsequent phase offset, the user controls the maximum step size of the phase steps via the 3-bit unsigned maximum phase slew step bit field in Bits[D2:D0] of Register 0x1108, Register 0x1111, Register 0x111A, Register 0x1123, Register 0x112C, Register 0x1135, Register 0x1508, Register 0x1511, Register 0x151A, and Register 0x1523. Each Q divider has a dedicated maximum phase slew step bit field that establishes the maximum phase step size per Table 35 (in which E is 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 35 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 value defined by the Qxy phase bit field.

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

When maximum phase slew step = 0 or 1, the slewing operation is limited to half-cycles or full cycles, respectively, of the input clock. Because this bit setting represents a small maximum phase step for most Q divider divide ratios, most choices of Qxy phase incurs phase slewing as though maximum phase slew step equals 0 or 1.

Table 35. Maximum Phase Slew Step Size

Maximum Phase Slew Step Bit Field Value	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°

Maximum phase slew step values greater than 1 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 and the device flags an error (via the aforementioned Channel x phase control error bit). 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 DPLLx phase control error bit.

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 the Qxy phase slew mode bits (where x is 0 or 1 and y is A, AA, B, BB, C, or CC) in Bit D3 of Register 0x1108, Register 0x1111, Register 0x111A, Register 0x1123, Register 0x112C, Register 0x1135, Register 0x1508, Register 0x1511, Register 0x151A, and Register 0x1523.

When Qxy phase slew mode is Logic 0 (default), the phase controller slews phase in the direction that reduces the output frequency during the stepwise phase adjustment sequence. Alternatively, when Qxy phase slew mode is Logic 1, the phase controller slews phase in the direction requiring the fewest number of steps. An exception is when enable Qxy half divide = 1 and the desired phase shift is within one Q divider input half-cycle of the midpoint of an output cycle. In this case, the phase controller may choose the slightly longer direction. When Qxy phase slew mode = 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 the enable Qxy pulse width control bit (where x is 0 or 1 and y is A, AA, B, BB, C, or CC) in Bit D4 of Register 0x1108, 0x1111, Register 0x111A, Register 0x1123, Register 0x112C, Register 0x1135, Register 0x1508, Register 0x1511, Register 0x151A, and Register 0x1523.

To set the pulse width, program the desired pulse width via the Qxy phase bit field and set the corresponding enable Qxy pulse width control bit to Logic 1. Then, assert the IO_UPDATE bit. When enable Qxy pulse width control = 1, the phase controller interprets the Qxy phase bit field as a pulse width value rather than a phase offset value.

The Qxy phase bit field relates to the pulse width as follows:

$$Pulse\ width\ (\%) = 100 \times Qxy\ phase/E \quad (2)$$

where:
Pulse width defines the portion of an output clock cycle from the rising edge to the falling edge.
E is as defined in the Initial Phase Offset section.

For example, determine the value of the Qxy phase bit field necessary to yield a 14% pulse width given E = 7301. Solving Equation 1 for Qxy phase yields the following:

$$Qxy\ Phase = Pulse\ Width \times E/100$$

$$= 14 \times 7301/100$$

$$= 1022 \text{ (nearest integer)}$$

$$= 0x0000003FE \text{ (hexadecimal)}$$

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 sync (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 47). The N-shot/PRBS controller provides the AD9545 with the capability of generating system reference (SYSREF) and device clock (DEVCLK) clocks per the JESD204B standard as well as gapped and pseudorandom clock signals.

A block diagram of the N-shot/PRBS clock controller appears in Figure 48, which includes a more detailed representation of the relationship between the Q dividers, the modulation controller and the N-shot/PRBS controller.

In the text that follows, Qxy denotes a particular Q divider, whereas Q_{xy} denotes its output (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.

Although Figure 48 is specific to PLL Channel 0, it is also representative of PLL Channel 1, because there are dedicated controllers for each PLL channel.

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 ones 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.

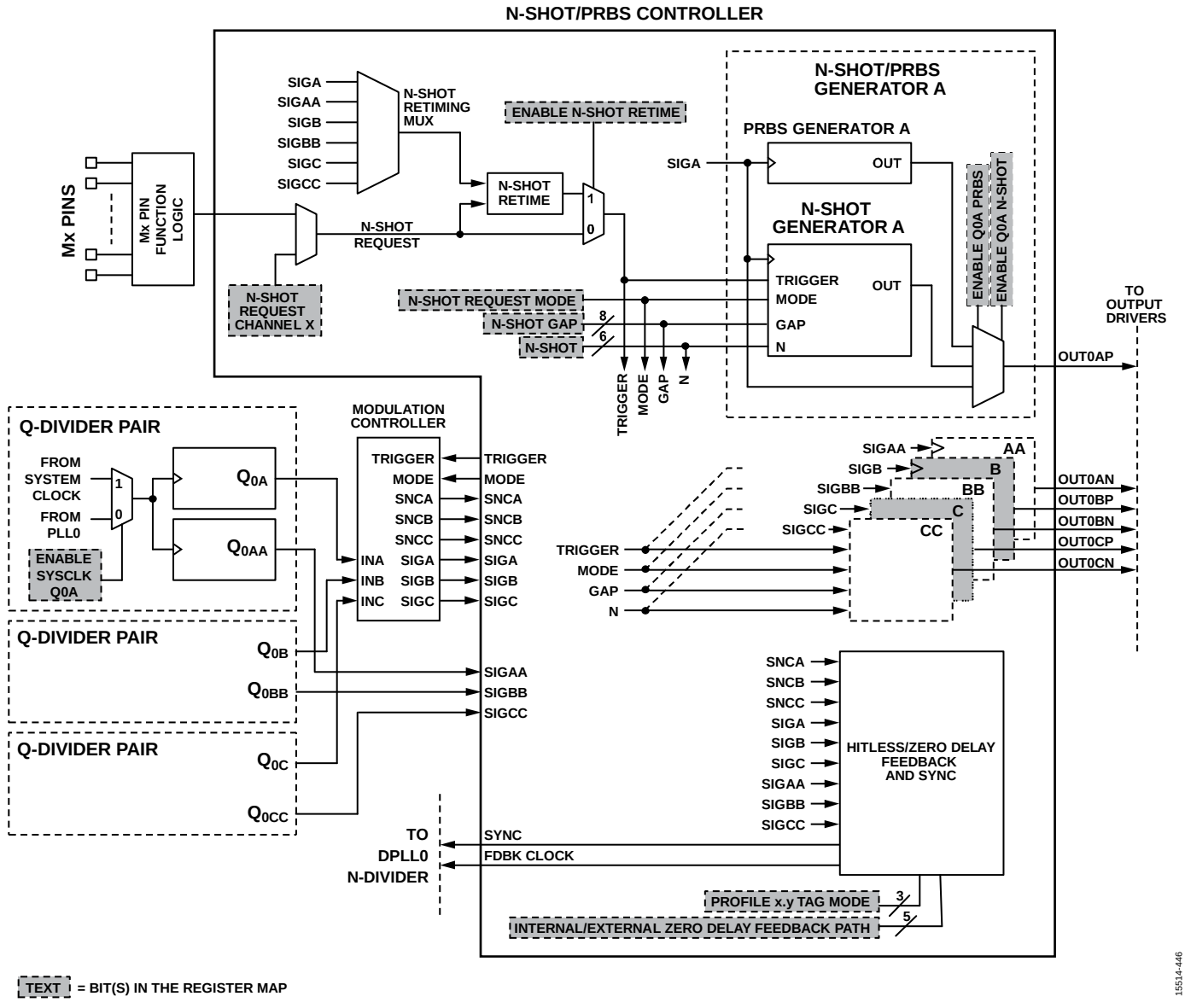


Figure 48. Block Diagram of JESD204B/Gapped Clock Controller

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 AP in Figure 48. The enable Qxy PRBS and enable Qxy N-shot bits select the N-shot/PRBS output clock function (where x is 0 or 1 and y is A, AA, B, BB, C, or CC).

The enable Qxy PRBS bits reside in Register 0x10D4 (Bit 1, Bit 3, Bit 5, and Bit 7), Register 0x10D5 (Bit 1 and Bit 3), and Register 0x14D4 (Bit 1, Bit 3, Bit 5, Bit 7).

The enable Qxy N-shot bits reside in Register 0x10D4 (Bit 0, Bit 2, Bit 4, and Bit 6), Register 0x10D5 (Bit 0 and Bit 2), and Register 0x14D4 (Bit 0, Bit 2, Bit 4, and Bit 6).

Table 36 shows a list of the N-shot/PRBS output clock function selections.

Table 36. N-Shot/PRBS Output Clock Function Selections

Enable Qxy PRBS	Enable Qxy N-Shot	Output Clock Function
0	0	Normal clock (Q divider)
0	1	Burst/gapped clock
1	0	Randomized clock
1	1	Not applicable

Although it is physically possible to set both enable Qxy PRBS and enable Qxy N-shot bits to Logic 1, the user must avoid doing so. This setting 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. A representation of the output clock signal appears in Figure 49.

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 in exchange for suppressing the harmonic spectral lines.

Because the user can program more than one output of the AD9545 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 have 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 AD9545 to generate JESD204B clock signals and gapped clock signals derives from it 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 single burst pattern. In either case, triggering a pattern is via the N-shot request Channel x bit (where x is 0 or 1) in Register 0x2101, Bit 0, and Register 0x2201, Bit 0. Alternatively, an external signal can serve as the N-shot trigger via the Mx pins.

In Figure 48, the N-shot parameters apply to all N-shot generators in a PLL channel, except for the enable Qxy N-shot bits, which apply to each individual N-shot generator. As such, the N-shot parameters are common to all outputs in a channel that have burst/gapped clock functionality enabled (per Table 36).

Using the N-shot generators imposes a minimum divide ratio of 8 on the associated Q dividers.

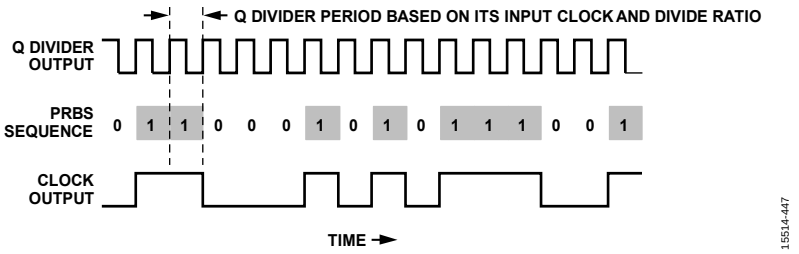


Figure 49. PRBS Clock Signal

N-Shot Pattern Generation

An N-shot pattern consists of a specified clock segment and gap segment per the 6-bit unsigned N-shot bit field and 8-bit unsigned N-shot gap bit field, respectively. The N-shot bit fields reside in Register 0x10D3, Bits[5:0] and Register 0x14D3, Bits[5:0]. The N-shot gap bit fields reside in Register 0x10D2 and Register 0x14D2.

The clock and gap segment are in units of the Q divider output period, as shown in Figure 50. The N-shot generator treats a value of 0 for the N-shot bit field as clock segment = 0 and a value of 0 for the N-shot gap bit field as gap segment = 1, resulting in no output.

The N-shot generator is capable of burst or periodic operation. To select burst or periodic operation, use the N-shot request mode bit in Register 0x10D3, Bit 6, and Register 0x14D3, Bit 6. When N-shot request mode = 0, the N-shot generators operate in burst mode (see Figure 51). When N-shot request mode = 1, the N-shot generators operate in periodic gapped mode (see Figure 52).

In burst operation, the rising edge of the trigger signal constitutes the trigger event. Prior to the burst, the generator is in its default state (Logic 0), which it 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 50). 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 N-shot bit field (where the clock segment = 6 in Figure 50). The gap period (where the gap segment = 10 in Figure 50) is effectively an extension of the default output condition, which is the low state.

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 52. 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, it stalls at the end of the current pulse as shown in Figure 53. 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 53).

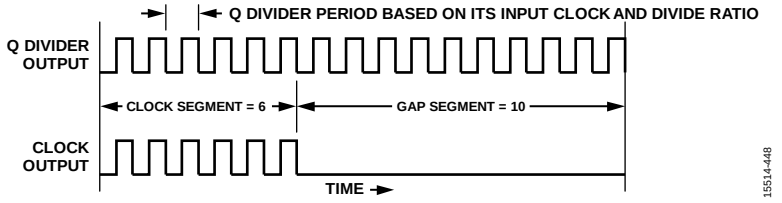


Figure 50. N-Shot Plus Gap Clock Signal

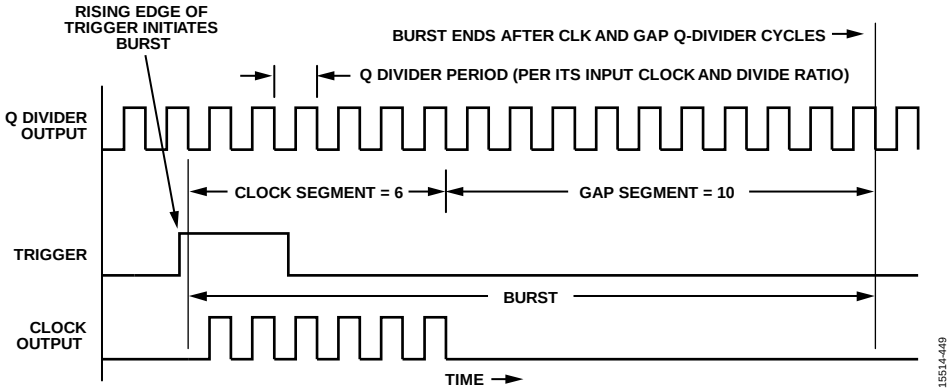


Figure 51. Burst Clock Signal

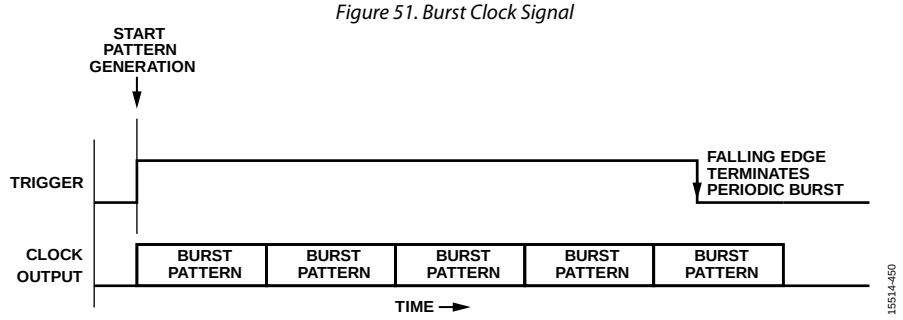


Figure 52. Periodic Gapped Clock Signal

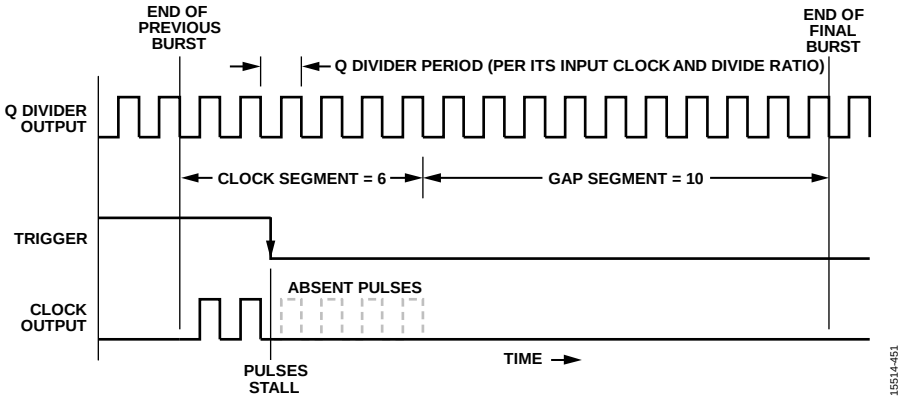


Figure 53. Stalling a Periodic Gapped Clock Signal

N-Shot Triggering

The N-shot trigger signal originates from one of two sources:

- N-shot request Channel x bit
- External signal applied via the Mx pins

Whether using the N-shot request Channel x bit or an external Mx pin signal, a Logic 1 constitutes a trigger event. In general, the user must apply Logic 1 to trigger the N-shot generators, 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 the N-shot request mode bit. As described in the N-Shot Pattern Generation section, this bit makes the trigger input of the N-shot generators edge or level sensitive for generating burst or periodic gapped clock signals, respectively (per Figure 51 and Figure 52).

The trigger mechanism for delivering a trigger signal to the N-shot generators appears in the upper left section of Figure 48. The N-shot generators support two triggering methods: direct and retimed.

To select the desired triggering method, use the enable N-shot retime bit in Register 0x10D6, Bit 0, and Register 0x14D6, Bit 0. Logic 0 (default) selects direct, whereas Logic 1 selects retimed.

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 Qxy phase 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 it waits for the rising edge of the slowest of all the Q dividers enabled for N-shot operation (user specified). The trigger signal, qualified by the slowest Q divider rising edge, 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 61.

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 is required between the retiming output (slowest) and the subsequent N-shot enabled output (fastest), which is 48 Q divider input half-cycles.

The retimed trigger is on the rising edge of the slowest N-shot enabled Q divider to accommodate multiple N-shot generators producing 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 being 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 device automatically selects the appropriate N-shot enabled Q divider for trigger retiming, such that the output with the largest Qxy phase value is always the retiming clock.

DISTRIBUTION EMBEDDED OUTPUT CLOCK MODULATION

MODULATION CONTROLLER OVERVIEW

The AD9545 has the capability to embed a low frequency clock within a high frequency carrier. Referring to Figure 48, 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 dynamically varying the pulse width of the designated Q divider output clock, where each variation of the pulse width constitutes a modulation event. A modulation event always spans two Q divider clock cycles (see the Balanced and Unbalanced Modulation section).

An expanded diagram of the modulation controller appears in Figure 54. Although Figure 54 is specific to PLL Channel 0, it is also representative of PLL Channel 1, because there is a dedicated modulation controller for each PLL channel.

Any one or more single-letter Q divider outputs (for example, Q_{0A} but not Q_{0AA}) can operate as an embedded clock modulator. To enable embedded modulation, program the appropriate enable Qxy modulator bits (where x is 0 or 1 and y is A, B or C) in Bit D0 of Register 0x10CF to Register 0x10D1 and Register 0x14CF to Register 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 54).

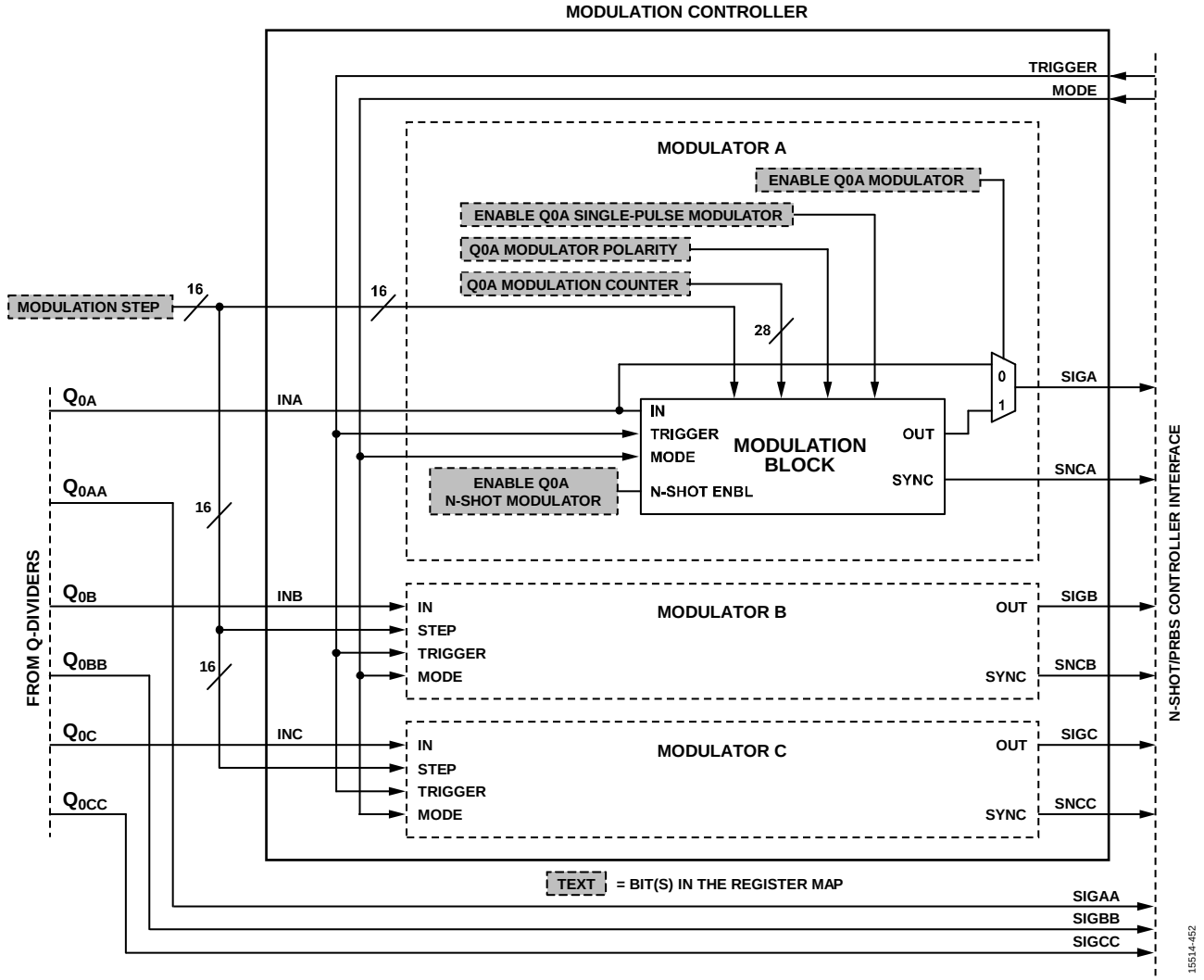


Figure 54. Block Diagram of Embedded Clock Modulation Controller

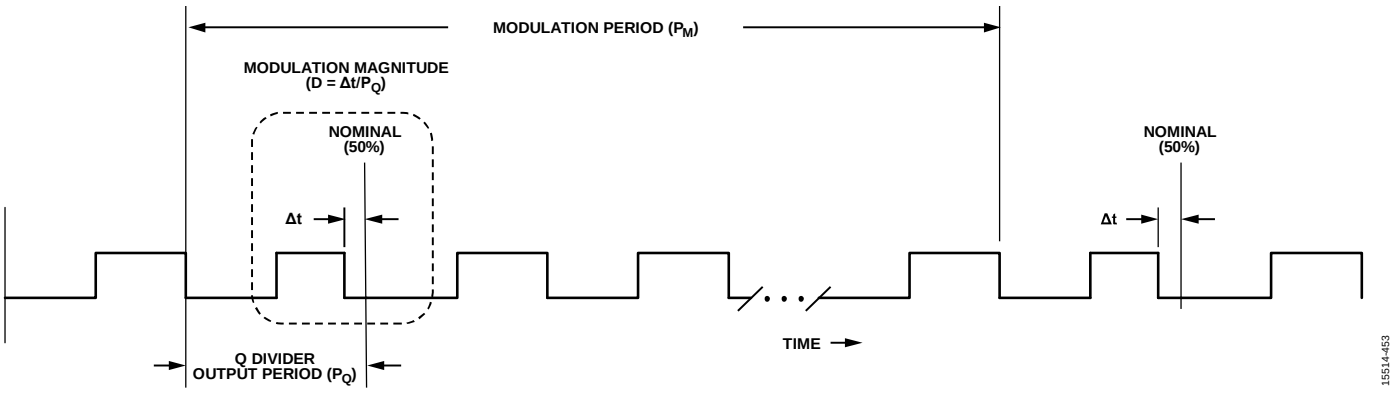


Figure 55. Modulation Magnitude and Period

Modulation control consists of two parameters: Δt and P_M. Parameter Δt defines the desired magnitude of the modulation edge variation, and P_M defines the modulation period (see Figure 55). The modulated signal consists of a time step of magnitude Δt occurring at regular intervals of period, P_M. The magnitude parameter, Δt, is common to all the modulators within a PLL channel, whereas the period parameter, P_M, is unique to each modulator.

MODULATION MAGNITUDE

To set the modulation magnitude use the 16-bit unsigned Modulation step bit field in Register 0x10C0 to Register 0x10C1 and Register 0x14C0 to Register 0x14C1. The units associated with the modulation step bit field are 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 55 shows *D* as $\Delta t/P_Q$.
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 bit field 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 Q_{0A} 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 prececing example (modulation step = 100), find the modulation magnitude for Modulator B assuming Q_{0B} has a divide ratio of 8025.5.

$$D = modulation\ step/(2 \times Qxy)$$
$$= 100/(2 \times 8025.5)$$
$$= 0.00623 \text{ (0.623\%)}$$

MODULATION PERIOD

The modulation period is the interval between modulation events (see Figure 55). To control the modulation period parameter, use the 28-bit unsigned Qxy modulation counter bit field (where x is 0 or 1 and y is A, B or C) in Register 0x10C2 to Register 0x10CD and Register 0x14C2 to Register 0x14C9. The units associated with the Qxy modulation counter bit field is the period of the output clock of the associated Q divider.

The Qxy modulation counter bit field has a constraint on its minimum value:

$$Qxy\ modulation\ counter \geq 6$$

The modulation period (P_M) depends on the input clock frequency (f_{IN}), the divide ratio (Q_{xy_DIV}) of the associated Q divider and the value of the Qxy modulation counter bit field.

$$P_M = Qxy\ modulation\ counter \times (Qxy/f_{IN}) \quad (4)$$

Suppose the input clock to Q Divider Q_{0A} 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 Qxy modulation counter bit field required to yield a modulation period (P_M) of 1 ms (0.001 sec) for a Q divide ratio (Q_{xy_DIV}) of 107.5.

Substituting the appropriate values into Equation 4 yields

$$10^{-3} = Qxy\ modulation\ counter \times (107.5/(1.19 \times 10^9)),$$

Therefore,

$$Qxy\ modulation\ counter = 11{,}070 \text{ (nearest integer)}$$
$$= 0x00002B3E \text{ (hexadecimal)}$$

The desired modulation period (P_M) is 1 ms. Because the Qxy 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 P_M value is 1.0000210084 ms.

BALANCED AND UNBALANCED MODULATION

Modulation consists of periodic modulation events occurring at regular intervals, P_M, 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 Qxy modulation counter bit fields) 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.

To select the modulation type, use the enable Qxy single-pulse modulator bit (where x is 0 or 1 and y is A, B or C) in Bit D2 of Register 0x10CF to Register 0x10D1 and Register 0x14CF to Register 0x14D0. Logic 0 (default) selects balanced modulation, whereas Logic 1 selects 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 56). 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 is unaltered (see Figure 57) and resulting in a waveform with a dc level slightly less than its nominal 50% amplitude point.

By default, the modulator applies negative polarity to Δt for the first pulse of a modulation event (and positive Δt to the second pulse if balance modulation is in effect). However, the user can force alternate polarity via the Qxy modulator polarity bit (where x is 0 or 1 and y is A, B, or C) in Bit D1 of Register 0x10CF to Register 0x10D1 and Register 0x14CF to Register 0x14D0. 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 58). Note the positive shift in dc offset for unbalanced modulation in this case.

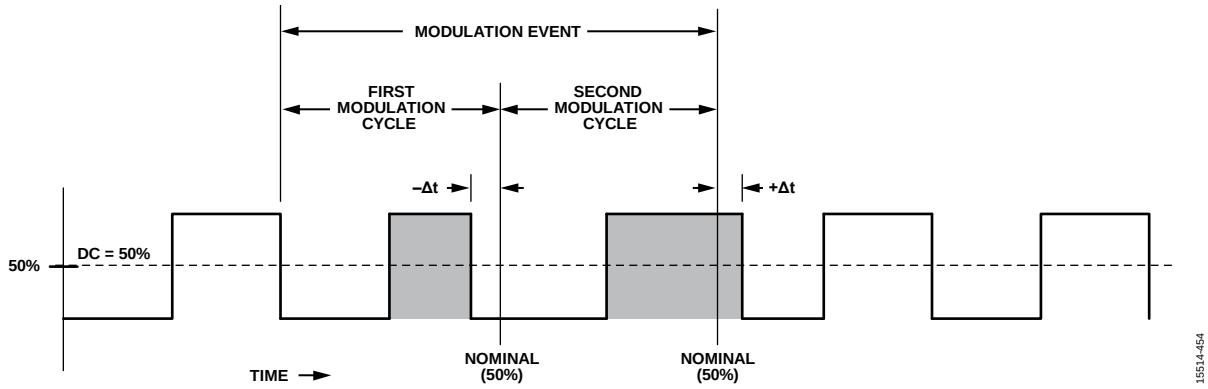


Figure 56. Balanced Modulation Waveform

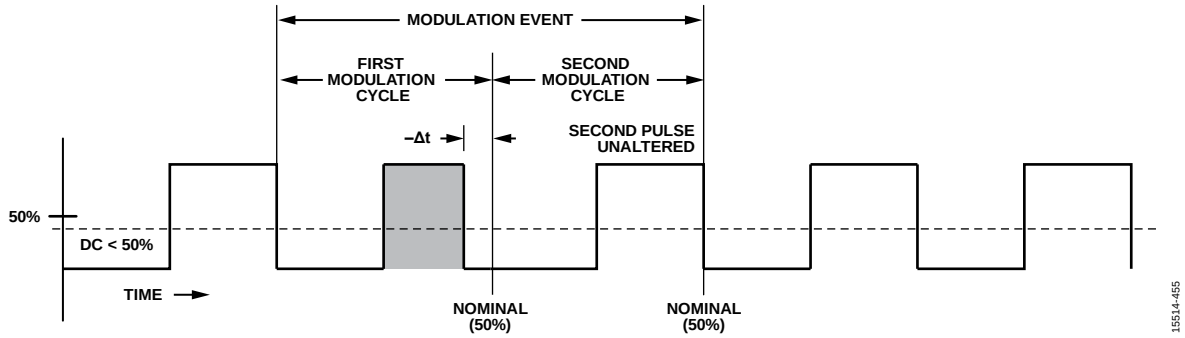


Figure 57. Unbalanced Modulation Waveform

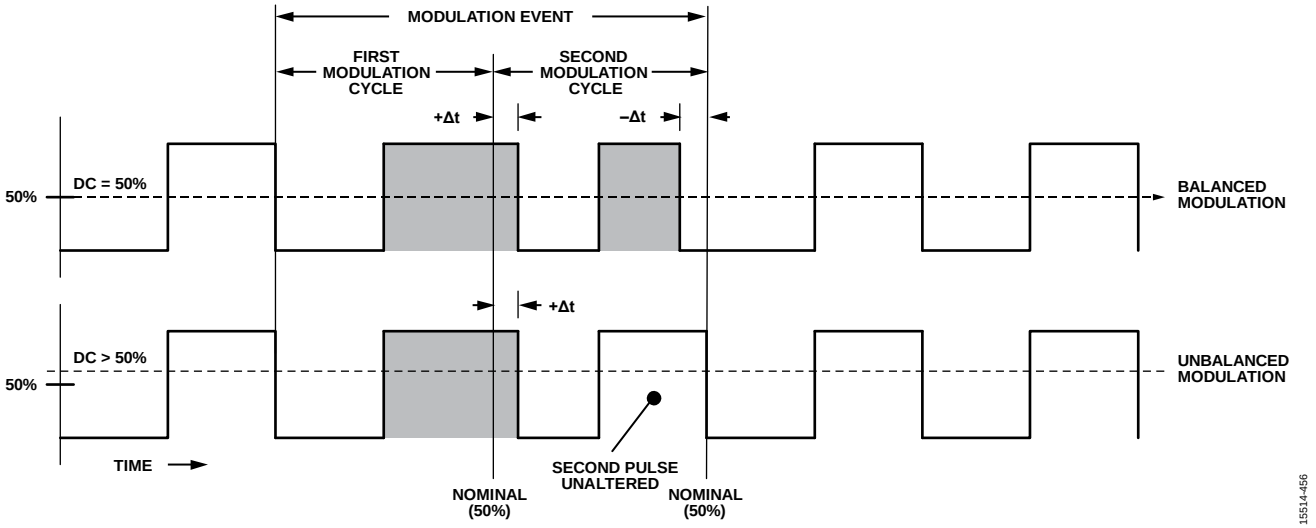


Figure 58. Alternate Modulation Polarity

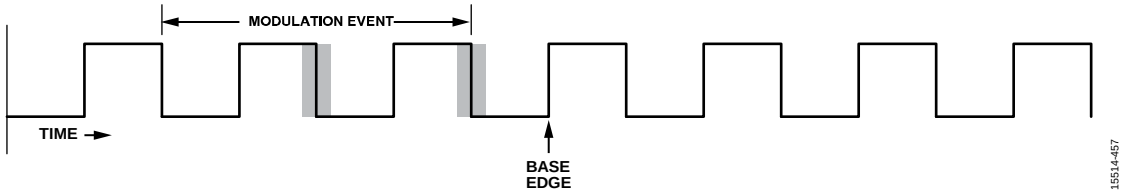


Figure 59. Modulation Synchronization Edges

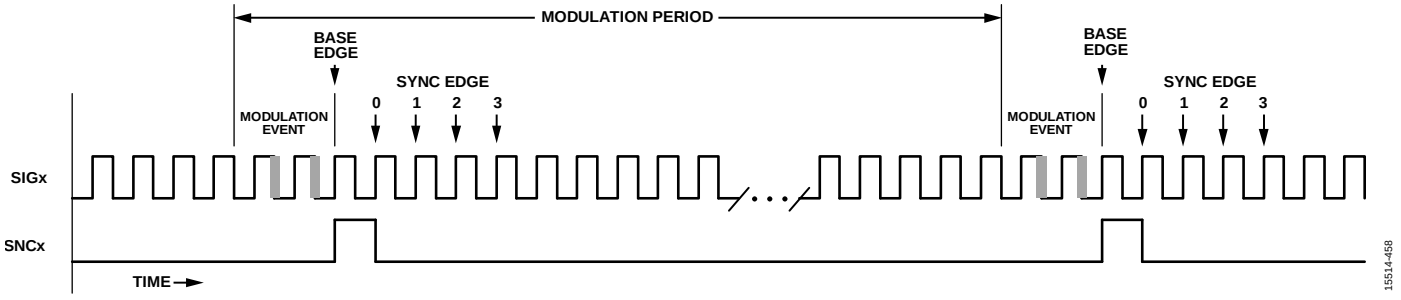


Figure 60. Modulation Synchronization

MODULATION SYNC

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 56 and Figure 57). 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 modulation cycle marks the base edge for synchronization purposes (see Figure 59).

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 60.

As shown in Figure 48, the SNCx signals route to the DPLL via the hitless/zero delay feedback and sync 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 is based on the 3-bit Profile x.y tag mode bit field (where x is 0 or 1 and y is 0 to 5) in Bits[D4:D2] of Register 0x1203, Register 0x1223, Register 0x1243, Register 0x1263, Register 0x1283, Register 0x12A3, Register 0x1603, Register 0x1623, Register 0x1643, Register 0x1663, Register 0x1683, and Register 0x16A3.

Because the SNCx signal identifies the modulation base edge, the user can synchronize the DPLL N-divider with a specific sync edge per Figure 60. To specify a sync edge for the N-divider use the 2-bit unsigned feedback divider sync edge bit field in Bits[D1:D0] of Register 0x10CE and Register 0x14CE. The value of the feedback divider sync edge bit field (0, 1, 2, or 3) corresponds to the desired sync edge in Figure 60.

There is an additional constraint on the value of Qxy modulation counter (see the Modulation Period section) when using the Q divider in the DPLL feedback path. The constraint relates to the value of Feedback divider sync edge as follows:

Qxy Modulation Counter ≥ Feedback Divider Sync Edge + 7

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. Enable this feature for a particular modulator via the enable Qxy N-shot modulator bit (where x is 0 or 1 and y is A, B, or C) in Bit D3 of Register 0x10CF to Register 0x10D1 and Register 0x14CF to Register 0x14D0. 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 48). The mode behavior differs slightly from that of the N-shot/PRBS controller. Specifically, when N-shot request mode = 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 48) is Logic 1. This behavior allows synchronous enable/disable of modulation events without losing synchronization during modulation.

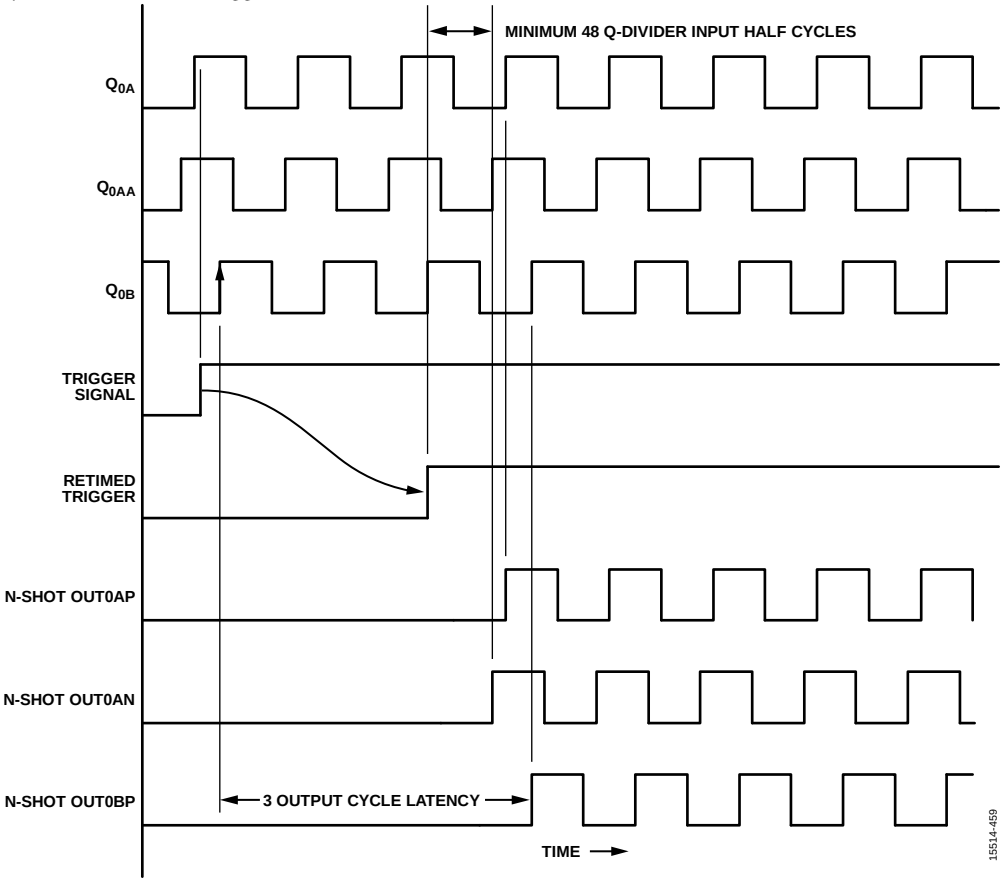


Figure 61. Trigger Retiming (with Q_{0AA} as the Retiming Q Divider)

DISTRIBUTION OUTPUT CLOCK SYNCHRONIZATION

SYNCHRONIZATION OVERVIEW

The AD9545 has a builtin 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.

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 sync trigger before activating the output clocks. Until a sync trigger occurs, the synchronization controller holds all the distribution Q dividers in a reset state, which prevents any clock signals from appearing at the OUTX pins. Therefore, the user must initiate the synchronization sequence to have any clock signals appear at the output.

There are three ways to trigger the synchronization sequence:

- Manual sync via an Mx pin, via the sync all bit in Bit D3 of Register 0x2000, via the sync all Channel x dividers bit (where x is 0 or 1) in Bit D3 of Register 0x2101 and Register 0x2201, or via an op code in an EEPROM download
- Autoreconfiguration (changes to Q divider values)
- Autosync

Autosync is the preferred option for output clock synchronization.

A sync trigger event is channel specific. That is, it applies to either PLL Channel 0 or PLL Channel 1. The exception is a manual sync, which can apply to both channels via the sync all bit. Assertion of the sync all bit does not imply synchronization of both channels relative to each other but causes a simultaneous triggering of the synchronization process for both channels.

A sync trigger initiates the synchronization sequence. Following a trigger event, the synchronization controller checks whether the enable DPLLx reference sync bit of either PLL channel is set set (Bit D2 of Register 0x10DB and Register 0x14DB). 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, 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 the enable DPLLx reference sync bit of either PLL channel is set so that it can align the release of the output clocks with a rising edge of the active reference. However, if the enable DPLLx reference sync bit is clear, the synchronization controller checks for release of manual sync (that is, a clearing of the sync all bit, or a clearing of the sync all Channel x dividers bit (where x is 0 or 1). 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 sync trigger.

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

MANUAL SYNC TRIGGER

A manual sync trigger occurs when the user sets one of three manual sync bits. The sync all bit initiates the synchronization sequence for all of the AD9545 output clocks. Alternatively, the user can synchronize only those outputs associated with one of the PLL channels via the sync all Channel x dividers bit. 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 manual sync 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 sync 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 sync bit.

AUTORECONFIGURATION SYNC TRIGGER

An autoreconfiguration sync trigger occurs when the user changes any of the Q divider ratios and subsequently sets the IO_UPDATE bit. However, the autoreconfiguration sync 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 sync 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 sync 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 Q_{0X} or Q_{0XX} divider results in disturbing all of the OUT0 clock outputs because it initiates a synchronization sequence for those outputs. Likewise, changing the value of any Q_{1X} or Q_{1XX} divider results in disturbing all of the OUT1 clock outputs. These output clock disturbances are unavoidable, because they result from the synchronization process. The disturbances are, however, glitch free, because the synchronization sequence mutes the affected output drivers during synchronization.

The aforementioned output clock disturbances are avoidable, however, when the enable SYSCLK to Qxy bits are set to Logic 1 and the enable SYSCLK sync mask bit are set to Logic 1 (see the

Q Divider Clock Source Selection section) for a subset of the device outputs.

AUTOSYNC TRIGGER

Although the AD9545 provides the user with the ability to synchronize the distribution outputs via a manual sync trigger, it is not a recommended action. Instead, the AD945 provides an autosync trigger feature that applies only after meeting certain conditions, which overcomes potential usability issues associated with manual trigger scenarios. In fact, the immediate autosync mode (see Table 37) is an effective replacement for manual sync triggering.

The user selects the desired autosync trigger conditions via the 2-bit autosync mode bit field in Bits[D1:D0] of Register 0x10DB and Register 0x14DB. Autosync control applies to PLL Channel 0 and PLL Channel 1, independently. Table 37 shows the selectable autosync conditions.

Table 37. Autosync Trigger Selections

Autosync Mode Bit Field Value	Autosync Trigger Condition
0	Disabled (default)
1	Immediate
2	On DPLL phase lock
3	On DPLL frequency lock

Autosync enables the AD9545 to execute a distribution synchronization sequence without any user intervention. The nonzero autosync selections specify conditions for generating an autosync trigger.

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

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

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

Note that after the autosync trigger occurs for a particular PLL channel, the autosync trigger generation sequence terminates. Subsequent autosync sequences on the same PLL channel are not possible unless the user sets the associated Clear Channel x autosync one-shot bit (Logic 0 by default) in Bit D0 of Register 0x2107 or Register 0x2207, or by updating the Autosync mode bits in Bits[D1:D0] 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 establishs 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 the enable DPLLx reference sync bit (where x is 0 or 1) in Bit D2 of Register 0x10DB and Register 0x14DB. Setting the enable DPLLx reference sync bit (Logic 0 by default) enables reference synchronization of those outputs associated with the corresponding PLL channel.

When using reference synchronization in conjunction with autosync mode (that is, autosync mode = 1, 2, or 3), be sure to program enable DPLLx reference sync = 1 and assert the I/O_UPDATE bit prior to programming autosync mode = 1, 2, or 3.

FREQUENCY TRANSLATION LOOPS

FREQUENCY TRANSLATION LOOPS OVERVIEW

The frequency translation capability of the AD9545 comprises two independent dual PLL channels (Channel 0 and Channel 1). Each PLL channel comprises a digital PLL (DPLL) followed by an analog PLL (APLL). Both channels have programmable reference dividers at the input and programmable channel dividers at the output. A block diagram of a single channel appears in Figure 62.

The DPLL channels (DPLL0 and DPLL1) are capable of very low loop bandwidths (μHz) well suited for jitter cleanup applications involving the 1 pulse per second (pps) input signals that originate from GPS and 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 AD9545 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 4.04 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
- Reference and feedback TDC tagging options
- DPLL fast acquisition (FACQ) options

When a particular profile is in effect, the DPLL uses the parameters as specified in that profile. That is, the DPLL has no memory with regard to switching between profiles.

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 controls)
- Q divider value (via the distribution controls).

Each divider type has a dedicated section in this data sheet with detailed information on the use and programming of the divider values.

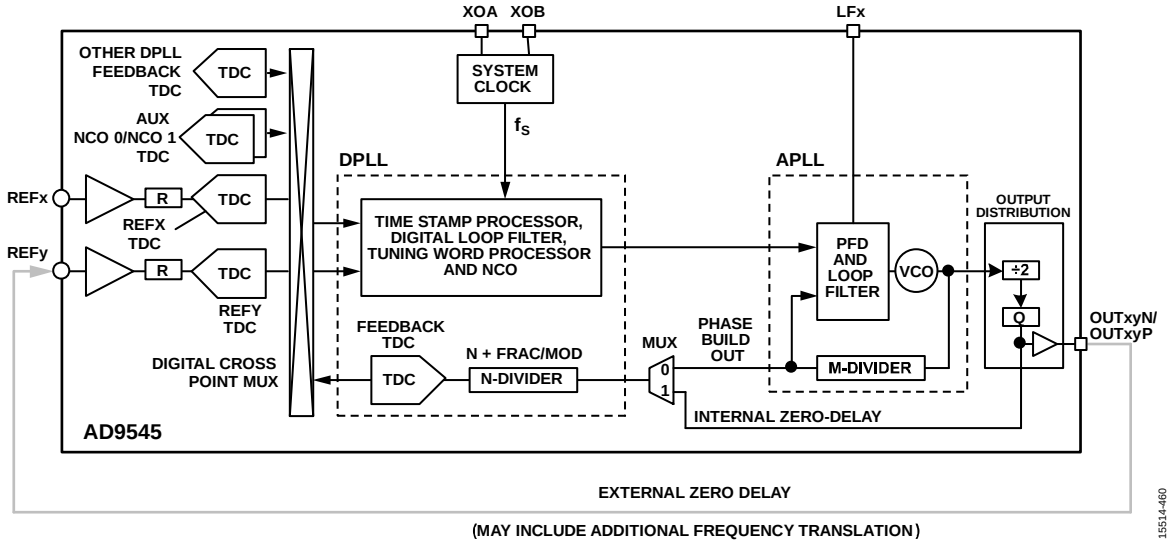


Figure 62. Frequency Translation Loop Block Diagram

PROFILE ENABLE

For a DPLL channel to have access to a translation profile, the user must enable the profile via its corresponding enable Profile x.y bit (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)). The enable Profile x.y bit resides in Bit D0 of Register 0x1200, Register 0x1220, Register 0x1240, Register 0x1260, Register 0x1280, Register 0x12A0, Register 0x1600, 0x1620, Register 0x1640, Register 0x1660, Register 0x1680, and Register 0x16A0. To enable a translation profile, program the corresponding enable Profile x.y bit to Logic 1 (default is Logic 0). If the translation profile is enabled, the DPLL selects the profile and the profile becomes active (see the Reference Switching section for what constitutes an active translation profile). Certain functional blocks within the AD9545 use the active/inactive state of a translation profile.

PROFILE PRIORITY

Profile priority works in conjunction with the reference switching capability of the AD9545 (see the Reference Switching section). The user programs a priority value via the 5-bit unsigned Profile x.y selection priority bit field (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)) in the register associated with the corresponding translation profile. The Profile x.y selection priority bit fields reside in Bits[D5:D1] of Register 0x1200, Register 0x1220, Register 0x1240, Register 0x1260, Register 0x1280, Register 0x12A0, Register 0x1600, Register 0x1620, Register 0x1640, Register 0x1660, Register 0x1680, and Register 0x16A0.

When the device must switch from one input reference to another, it can do so according to user defined priority per the value stored in the Profile x.y selection priority bit field, where 0 constitutes highest priority and 31 lowest priority.

INPUT REFERENCE SOURCE SELECTION

The AD9545 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, either of the two auxiliary NCOs, and a special configuration that uses the feedback signal from the other DPLL channel. Each translation profile can specify a different reference source.

Program the source selection via the 5-bit Profile x.y reference source selection bit field (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)) in the register associated with the corresponding translation profile. The Profile x.y reference source selection bit fields reside in Bits[D4:D0] of Register 0x1201, Register 0x1221, Register 0x1241, Register 0x1261, Register 0x1281, Register 0x12A1, Register 0x1601, Register 0x1621, Register 0x1641, Register 0x1661, Register 0x1681, and Register 0x16A1. The bit field value assigns the source, as shown in Table 38.

Table 38. Reference Source Selection

Bit Field Value (Decimal)	Source
0	REFA
1	REFAA
2	REFB
3	REFBB
4	Feedback from DPLL0 (applies to Channel 1 only)
5	Feedback from DPLL1 (applies to Channel 0 only)
6	Not applicable
7	Not applicable
8	Auxiliary NCO 0
9	Auxiliary NCO 1
10 to 31	Not applicable

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 AD9545 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 matching of 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 AD9545 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 (δθ/δt) 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 δθ/δt) by using the frequency clamp feature (see the Tuning Word Offset Clamp section).

Even during a hitless acquisition, the AD9545 builds out the initial phase offset between the reference and feedback signals (as though it were 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 Application Note AN-1420.

Translation Mode Selection

PLL0 and PLL1 are capable of operating 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 particular profile.

Each profile has two bits that allow the user to set the operating mode for that profile (see Table 39). The first is the enable Profile x.y hitless bit (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)), which resides in Bit D0 of Register 0x1203, Register 0x1223, Register 0x1243, Register 0x1263, Register 0x1283, Register 0x12A3, Register 0x1603, Register 0x1623, Register 0x1643, Register 0x1663, Register 0x1683, and Register 0x16A3.

The second is the enable Profile x.y external zero delay bit (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)), which resides in Bit D1 of Register 0x1203, Register 0x1223, Register 0x1243, Register 0x1263, Register 0x1283, Register 0x12A3, Register 0x1603, Register 0x1623, Register 0x1643, Register 0x1663, Register 0x1683, and Register 0x16A3.

Table 39. Frequency Translation Modes

Enable Profile x.y Hitless Bit	Enable Profile x.y External Zero Delay Bit	Translation Mode
0	0	Phase buildout
0	1	Not applicable
1	0	Internal zero delay
1	1	External zero delay

PHASE BUILDOUT MODE

Figure 63 shows the phase buildout configuration. When the AD9545 switches from one input reference to another in phase buildout mode, it does so with virtually no phase disturbance at the output, which is possible because the AD9545 is able to determine the phase offset between the old and new references (see the Initial Phase Skew Refinement Steps section for more detail). As such, the device can compensate for the time offset of the new reference by adding the appropriate value to (or subtracting from) timestamps generated by the TDC associated with the new reference.

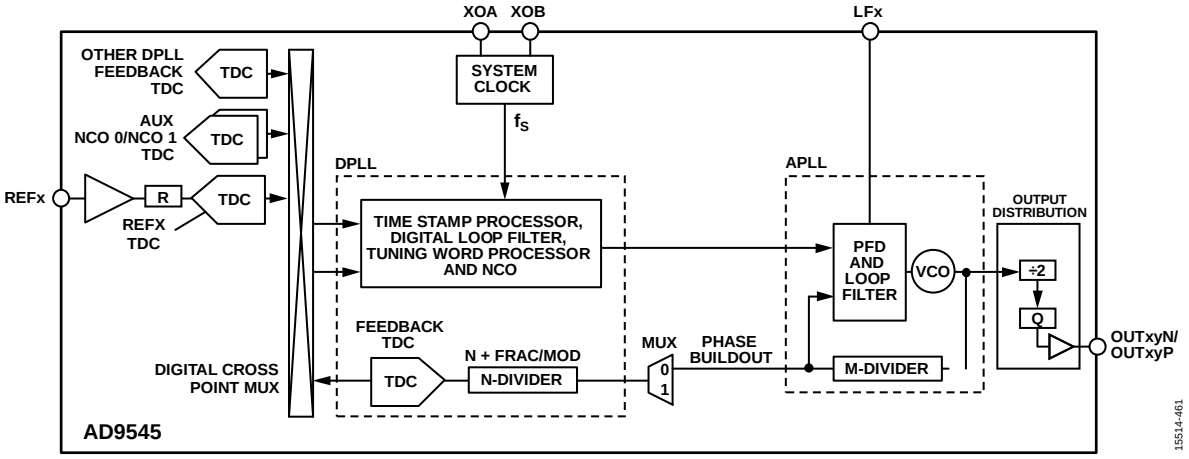


Figure 63. Phase Buildout PLL Configuration

The frequency translation factor for phase buildout mode is:

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

However, 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 bounds.

$$f_{TDC} = f_{REFx}/R$$
$$= (2 \times Q/(M \times (N + FRAC/MOD))) \times f_{OUTx} \quad (5)$$
$$f_{NCO} = f_{PFD} = (2 \times Q/M) \times f_{OUTx}$$
$$= ((N + FRAC/MOD)/R) \times f_{REFx} \quad (6)$$
$$f_{VCO} = 2 \times Q \times f_{OUTx}$$
$$= ((N + FRAC/MOD) \times M/R) \times f_{REFx} \quad (7)$$

When the reference source is auxiliary NCO 0, auxiliary NCO 1, or the feedback from the other DPLL, then in Euquation 5 through Equation 6, use R = 1.

INTERNAL ZERO DELAY (HITLESS) MODE

Figure 64 shows the internal zero delay configuration. Internal zero delay mode is a hitless mode (rather than phase buildout operating mode), in which the AD9545 gradually changes the output phase as it 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 AD9545 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 a user selected Q divider in the output distribution section. The feedback Q divider selection is via the 5-bit unsigned internal/external zero delay feedback path bit field (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)). The internal/external zero delay feedback path bit fields reside in Bits[D4:D0] of Register 0x1202, Register 0x1222, Register 0x1242, Register 0x1262, Register 0x1282, Register 0x12A2, Register 0x1602, Register 0x1622, Register 0x1642, Register 0x1662, Register 0x1682, and Register 0x16A2.

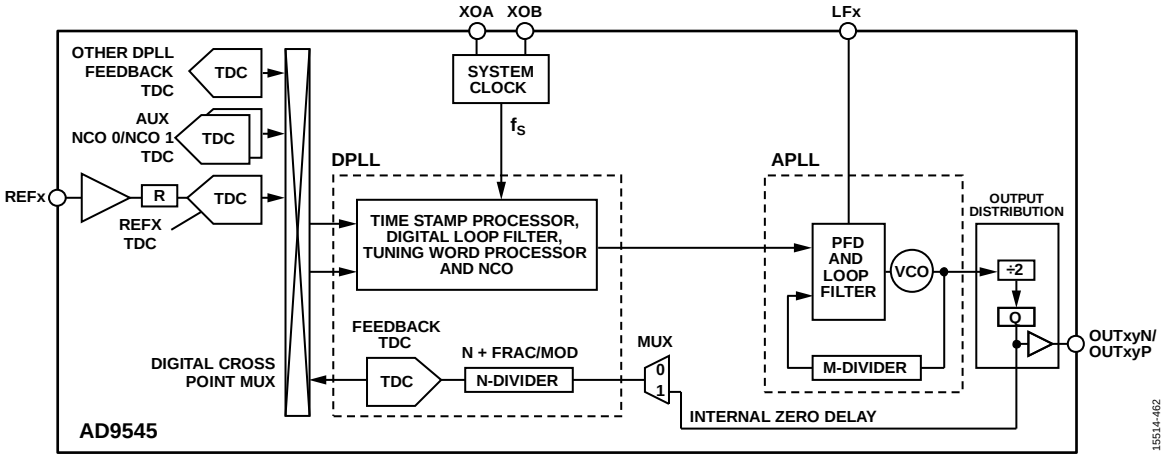


Figure 64. Internal Zero Delay PLL Configuration

The internal/external zero delay feedback path bit field value relates to the Q divider output per the internal zero delay column in Table 40. 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.

Table 40. Zero Delay Feedback Path Selection

Internal/ External Zero Delay Feedback Path Value	DPLL	Internal Zero Delay	External Zero Delay
0	0	OUT0AP	REFA
0	1	OUT0AP	REFA
1	0	OUT0AN	REFAA
1	1	OUT0AN	REFAA
2	0	OUT0BP	REFB
2	1	OUT0BP	REFB
3	0	OUT0BN	REFBB
3	1	OUT0BN	REFBB
4	0	OUT0CP	Not applicable
4	1	Not applicable	Not applicable
5	0	OUT0CC	Not applicable
5	1	Not applicable	Not applicable
6 to 31	n/a	Not applicable	Not applicable

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 bounds.

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

$$f_{NCO} = f_{PFD} = (2 \times Q/M) \times f_{OUTx} = ((2 \times Q \times N)/(R \times M)) \times f_{REFx} \quad (9)$$

$$f_{VCO} = 2 \times Q \times f_{OUTx} = (2 \times Q \times N/R) \times f_{REFx} \quad (10)$$

When the reference source is auxiliary NCO 0, auxiliary NCO 1, or the feedback from the other DPLL, then in Equation 8 through Equation 10, use R = 1.

The DPLL provides the user with two status bits to indicate when the DPLL transitions to or from hitless operation. The DPLLx hitless entered bit (where x is 0 or 1) latches to Logic 1 when the DPLL enters hitless operating mode. The DPLLx hitless exited bit (where x is 0 or 1) latches to Logic 1 when the DPLL exits hitless operating mode. These bits reside in Bits[D4:D3] of Register 0x3011 and Register 0x3016. Because these are latched bits, the user must clear them via the IRQ map DPLLx clear registers (Bits[D4:D3] of Register 0x200C and Register 0x2011) to obtain visibility of subsequent state transitions into and out of hitless operation.

Caveat to Internal Zero Delay Operation

Although the digital phase detector associated with the DPLL can typically handle input frequencies as low as 1 Hz, the internal zero delay mode imposes a lower boundary of 2 kHz on the feedback input to the digital phase detector. The 2 kHz lower bound applies only to the feedback input of the digital phase detector, not to its reference input.

For internal zero delay operation in which the reference input to the digital phase detector is less than 2 kHz, the user must program the DPLL to use tagged feedback operation (see the Time Stamp Tagging Options section in the Digital PLL (DPLL) section). That is, for internal zero delay operation, the device supports tagged feedback rates as low as 1 Hz whenthe untagged feedback rate is greater than 2 kHz and $f_{OUTx} \le f_{NCO}$.

EXTERNAL ZERO DELAY (HITLESS) MODE

The external zero delay configuration appears in Figure 65. 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 enter/exit hitless status bits apply.

In external zero delay mode, the feedback path of the PLL is via an external connection from an appropriate OUTx output to a REFx input. The REFx input selection is via the 5-bit unsigned internal/external zero delay feedback path bit field (per the external zero delay column in Table 40). 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 65, they appear as REFx and REFy, respectively.

The frequency translation factor from REFx to OUTxyP/OUTxyN for external zero delay mode is:

$$\frac{f_{OUTx}}{f_{REFx}} = \frac{Ry}{Rx \times Z}$$

where 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 bounds.

$$f_{TDC} = f_{REFx}/Rx$$
$$= (Z/Ry) \times f_{OUTx} \quad (11)$$
$$f_{NCO} = f_{PFD} = ((2 \times Q \times Ry)/(Rx \times M)) \times f_{REFx}$$
$$= (2 \times Q/M) \times f_{OUTx} \quad (12)$$
$$f_{VCO} = ((2 \times Q \times Ry)/(Rx \times Z)) \times f_{REFx}$$
$$= 2 \times Q \times f_{OUTx} \quad (13)$$

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 65, it requires special treatment for external zero delay operation. See the DPLL Feedback Divider (N-Divider section for details.

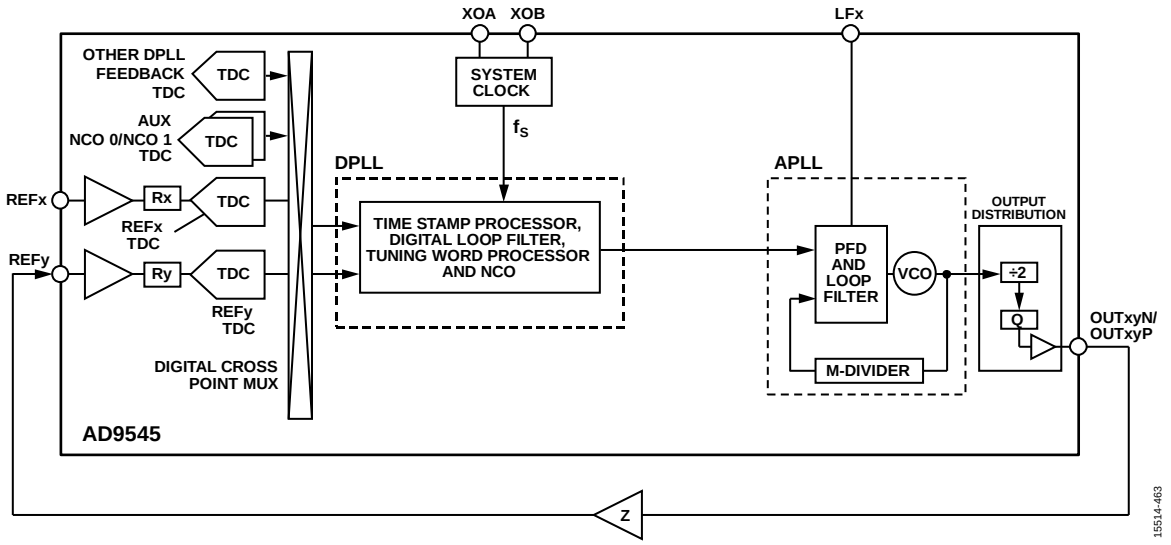


Figure 65. External Zero Delay PLL Configuration

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 register map provides access to the eight source profiles via Register 0x0800 to Register 0x08F1. 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

The source profiles relate to the Profile x.y reference source selection bit field associated with the DPLL translation profiles found in Register 0x1200 to Register 0x12B7 and Register 0x1600 to Register 0x16B7. That is, a translation profile attaches to a particular source profile based on the value of the associated Profile x.y reference source selection bit field of the translation profile (see the Translation Profiles section).

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 AD9545 has the ability to switch between multiple reference inputs, some applications use only one input and handle reference switching externally (see Figure 66). This arrangement forfeits the ability of the AD9545 to mitigate the output disturbance associated with a reference switchover, because the reference switchover is not under the control of the AD9545. However, the AD9545 offers a phase transient threshold detection feature to help identify when an external reference switchover occurs and act accordingly.

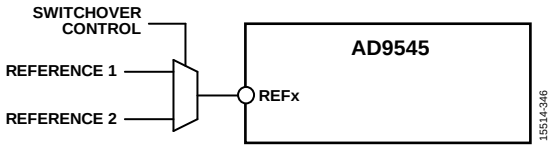


Figure 66. External Reference Switching

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 activate the phase transient threshold detection, program the 32-bit unsigned Profile x phase step threshold bit field (where x is a value from 0 through 7 corresponding to a particular source profile) in registers starting with Address 0x080A, Address 0x082A, Address 0x084A, Address 0x086A, Address 0x088A, Address 0x08AA, Address 0x08CA, and Address 0x08EA.

The default value is zero, which disables the phase transient threshold detector. A nonzero value denotes the desired phase step threshold (Φ_{THRESH}) in units of picoseconds per the following equation.

$$\Phi_{THRESH} = Profile\ x\ Phase\ Step\ Threshold \times 10^{-12} \quad (14)$$

The phase transient threshold detector is not active unless the DPLL indicates frequency locked status.

Determine the value of the Profile x phase step threshold bit field necessary for a 12 ns limit (that is, $\Phi_{THRESH} = 12 \times 10^{-9}$). Solving Equation 14 for Profile x phase step threshold yields

$$Profile\ x\ Phase\ step\ Threshold = \Phi_{THRESH}/10^{-12}$$
$$= (12 \times 10^{-9})/10^{-12}$$
$$= 12{,}000$$
$$= 0x00002EE0\ (hexadecimal)$$

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 with for Profile x phase step threshold = 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 Profile x phase step threshold = 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, scaling 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 the enable step detect reference fault bit in Bit D7 of Register 0x2105 and Register 0x2205:

- The DPLL initiates a new acquisition sequence
- The reference monitor is reset

When the enable step detect reference fault bit 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 (FACQ) Options section), assuming it is active, which is especially 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 the enable step detect reference fault bit 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 statics (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. Phase step detection status is via the DPLLx phase step detected bit (where x is 0 or 1) in Bit D0 of Registers 0x3011 and 0x3016. Because these are latched IRQ bits, the user must clear them via Bit D0 of Register 0x200C and Register 0x2011 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 via the DPLLx phase offset bit field (see the DPLL Phase Offset Control section) or the Profile x phase skew bit field (see 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 picoseconds).
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, for example. Because MPSR in Equation 16 constitutes an upper bound, it is best to use a lower value of MPSR to provide margin (a 25% reduction is reasonable).

In fact, 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, then rather than using the phase transient step detector a more robust solution is to invalidate the associated reference manually. To do so, force a reference fault condition via the appropriate fault REFx bit field (where x is A, AA, B, or BB) in Bits[D3:D0] of Register 0x2003. Using this method 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 force 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/GPS reference sources, for example. That is, a user may 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, program the 24-bit signed Profile x phase skew bit field (where x is 0 to 7) in registers starting with Address 0x080E, Address 0x082E, Address 0x084E, Address 0x086E, Address 0x088E, Address 0x08AE, Address 0x08CE, and Address 0x08EE. The default value is zero, which disables the skew adjustment feature. A nonzero value enables the skew adjustment feature and denotes the desired time skew in units of picoseconds per the following equation:

$$Time\ skew = Profile\ x\ Phase\ Skew \times 10^{-12} \quad (17)$$

For example, determine the value of the Profile x phase skew bit field necessary for a time skew of –35 ns. Solving Equation 17 for Profile x phase skew yields the following:

$$\begin{aligned} Profile\ x\ Phase\ Skew &= Time\ Skew/10^{-12} \\ &= (-35 \times 10^{-9})/10^{-12} \\ &= -35{,}000 \\ &= 0xFF7748\ (hexadecimal) \end{aligned}$$

INITIAL PHASE SKEW REFINEMENT STEPS

Whenever the AD9545 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 67). 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).

By definition, this initial phase offset insertion constitutes a phase buildout and is a requirement for reference switchover involving a phase buildout translation profile. However, because the AD9545 supports both hitless and phase buildout translation profiles, it can leverage its phase buildout capability to assess the reference phase offset and build out the initial phase offset of a hitless switchover as well. By applying phase buildout at the beginning of a hitless switchover, the AD9545 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 AD9545 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 persistance provides some time for the DPLL to compare the new reference signal to the old one.

As shown in Figure 67, 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 AD9545 provides a phase skew refinement feature. To enable the phase skew refinement feature, program a non-zero value into the 8-bit unsigned Profile x phase skew refinement steps bit field (where x is 0 to 7) in Register 0x0811, Register 0x0831, Register 0x0851, Register 0x0871, Register 0x0891, Register 0x08B1, Register 0x08D1, and Register 0x08F1.

The default value is zero, which disables the skew refinement function. With the skew refinement function disabled, the phase buildout value is a 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.

Programming a nonzero value, K, sets the number of phase samples the AD9545 analyzes as part of the skew refinement process. That is, instead of taking the first phase sample (jitter included) as the phase buildout value, the skew refinement feature processes the first K phase samples to assess the reference jitter and to determine the phase buildout value. As such, the skew refinement feature extends the time required to determine a phase buildout value following a reference switchover, but results in the application of a more robust phase buildout value.

The skew refinement process operates under the assumption that the feedback and reference clocks are of the same frequency, or at least very close. If not, the frequency offset appears as a linear phase slew, which quickly becomes the dominant phase contributor and masks out 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 skew refinement feature.

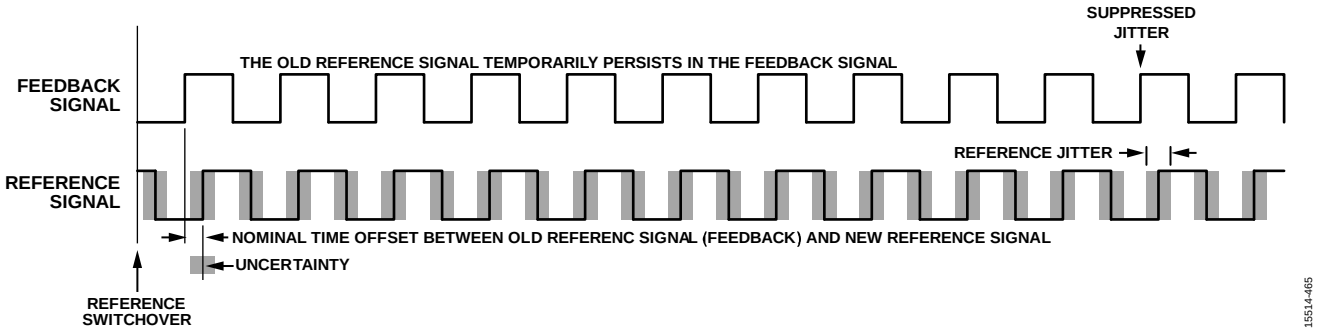


Figure 67. DPLL Reference and Feedback Signals

DIGITAL PLL (DPLL)

DPLL OVERVIEW

Note that throughout this section, unless otherwise specified, any referenced bits, registers, or bit fields reside in the DPLL Channel 0 and DPLL Channel 1 registers (Address 0x1000 to Address 0x102A, and Address 0x1400 to Address 0x142A) of the register map.

The DPLL is an all digital implementation of a phase-locked loop (PLL). Figure 68 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 a numerically controlled oscillator (NCO), which relies on a digital frequency tuning word (FTW) to produce the output frequency. A VCO inherently produces a timing signal because it is, by definition, an oscillator, whereas the AD9545 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).

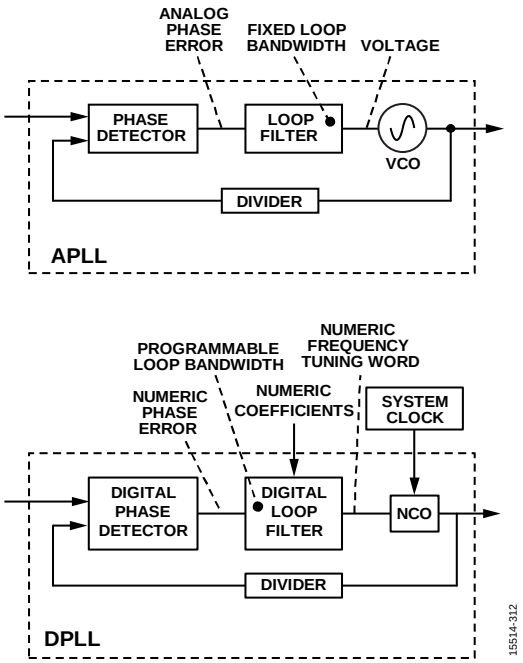


Figure 68. APLL vs. DPLL

The DPLLs in the AD9545 have a digital TDC-based phase detector 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 (including 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). The AD9545 handles a reference switchover by briefly entering holdover mode, loading the new DPLL parameters, and then immediately recovering.

Holdover or Freerun

Typically, the holdover or freerun state is in effect when all of the input references are invalid. However, the user can force the holdover mode even when one or more references are valid by setting the DPLLx force holdover bit (where x is 0 or 1) to Logic 1 (Bit D1 of Register 0x2105 and Register 0x2205). To force freerun mode, see the Freerun Tuning Word section. In holdover mode, the output frequency remains fixed (to the extent of the stability of the system clock). The accuracy of the AD9545 in holdover mode is dependent on the device programming and availability of the tuning word history.

When the DPLL is in holdover or freerun mode, it implies 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 the DPLLx inactive profile index bit field (where x is 0 or 1) in Bits[D2:D0] of Register 0x102A or Register 0x142A. The DPLLx inactive profile index bit field defines which translation profile attaches to the DPLL when in holdover or freerun mode.

Because the DPLL attaches to the profile specified by the DPLLx inactive profile index bit field, the recommendation is for the user to program the DPLLx inactive profile index bit field with one of the translation profiles the user has purposefully configured. For example, if the user configures Translation Profile 0, Translation Profile 2, and Translation Profile 5 for use by DPLL0, program the DPLL0 inactive profile index bit field with 0, 2, or 5.

Recovery from Holdover

When in holdover and an enabled translation profile becomes available, the device exits holdover operation. The loop controller restores the DPLL to closed-loop operation, locks to the selected reference, and sequences the recovery of all the loop parameters based on the profile settings for the active reference.

If DPLLx force holdover = 1, 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 the DPLLx force holdover bit.

DPLL FEEDBACK DIVIDER (N-DIVIDER)

The DPLL feedback N-divider consists of an integer and fractional part. The fractional part, however, is only available in buildout mode (see the Frequency Translation Loops section). There are four unsigned 32-bit bit fields associated with the N-divider as follows:

- Profile x.y buildout N-divider
- Profile x.y hitless N-divider
- Profile x.y buildout fraction
- Profile x.y buildout modulus

Although these four bit fields comprise 32 bits each, only the lower 30 bits are useable as a valid divide ratio (that is, the two most significant bits must always be zero). There are 12 occurrences of these bit fields: six occurrences associated with the translation profiles for PLL0 and six associated with the translation profiles for PLL1. These bit fields reside in the Register 0x1200 to Register 0x12B7 and Register 0x1600 to Register 0x16B7).

The programming requirements for the N-divider depends on the selected operating mode (see the Frequency Translation Loops section).

N-Divider for Phase Buildout Mode

In phase buildout mode, the N-divider is an integer plus fractional divider (N + FRAC/MOD). Note that changing the value of N or MOD when the corresponding translation profile is active 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.

To program the value of N, use Profile x.y buildout N-divider bit field (where x is 0 or 1 and y is a value from 0 to 5). The Profile x.y buildout N-divider bit fields reside in registers starting with Address 0x120C, Address 0x122C, Address 0x124C, Address 0x126C, Address 0x128C, Address 0x12AC, Address 0x160C, Address 0x162C, Address 0x164C, Address 0x166C, 0x168C, and Address 0x16AC.

Program the Profile x.y buildout N-divider bit field with a value one less than the desired divide ratio. As a result, the N-divider provides integer divide ratios from 1 to 1,073,741,824.

For example, for a divide ratio of N = 1,000,000, the Profile x.y buildout N-divider bit field value is 999,999 (0x000F423F hexadecimal).

Programming the fractional part of the N-divider requires two integers. One integer constitutes the numerator of the fraction (FRAC). The other integer constitutes the denominator of the fraction (MOD).

Program FRAC via the 24-bit unsigned Profile x.y buildout fraction bit field (where x is 0 or 1 and y is a value from 0 to 5). The Profile x.y buildout fraction bit fields reside in registers starting with Address 0x1210, Address 0x1230, Address 0x1250, Address 0x1270, Address 0x1290, Address 0x12B0, Address 0x1610, Address 0x1630, Address 0x1650, Address 0x1670, Address 0x1690, and Address 0x16B0.

Program MOD via the 24-bit unsigned Profile x.y buildout modulus bit field (where x is 0 or 1 and y is a value from 0 to 5). MOD must satisfy the constraint: FRAC < MOD. The Profile x.y buildout modulus bit fields reside in registers starting with Address 0x1213, Address 0x1233, Address 0x1253, Address 0x1273, Address 0x1293, Address 0x12B3, Address 0x1613, Address 0x1633, Address 0x1653, Address 0x1673, Address 0x1693, and Address 0x16B3.

For phase buildout applications requiring the N-divider to be integer only, program MOD = 0, which disables the fractional divide portion of the N-divider.

For example, suppose the desired fractional part of the N-divider is 0.387429. Then the corresponding fraction is FRAC/MOD = 387,429/1,000,000. Generally, one reduces the FRAC/MOD fraction to its lowest terms, but in this example, the fraction is irreducible. Therefore, the Profile x.y buildout fraction bit field value is 387,429 (0x0005E965 hexadecimal) and the Profile x.y buildout modulus bit field value is 1,000,000 (0x000F4240 hexadecimal).

N-Divider for Zero Delay Mode

In the zero delay modes, the AD9545 automatically disables the fractional divide portion of the N-divider. That is, zero delay (hitless) operation necessitates that the DPLL operate as an integer-N PLL.

N-Divider in Internal Zero Delay Mode

Program N via the 32-bit unsigned Profile x.y hitless N-divider bit field (where x is 0 or 1 and y is a value from 0 to 5). The Profile x.y hitless N-divider bit fields reside in registers starting with Address 0x1208, Address 0x1228, Address 0x1248, Address 0x1268, Address 0x1288, Address 0x12A8, Address 0x1608, Address 0x1628, Address 0x1648, Address 0x1668, Address 0x1688, and Address 0x16A8. The format of this bit field is identical to the Profile x.y buildout N-divider bit field. However, even though the device is operating in zero delay (hitless) mode, the user must still program the Profile x.y buildout N-divider bit field to ensure proper operation of the DPLL loop filter. The divide ratio is the DPLL NCO frequency divided by the frequency of the TDC associated with the DPLL reference input and rounded to the nearest integer. In the case of internal zero delay operation, the Profile x.y buildout N-divider bit field value relates to the Profile x.y hitless N-divider bit field value as follows (rounded to nearest integer):

$$N_{BUILDOUT} = N_{HITLESS} \times 2 \times Q/M$$

where Q is the Q divider and M is the M divider (see Figure 62).

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 Profile x.y hitless N-divider bit field. However, the user must

still program the Profile x.y buildout N-divider bit field to ensure proper operation of the DPLL loop filter. Program the Profile x.y buildout N-divider bit field as follows (refer to Figure 65):

$$N_{BUILDOUT} = \text{round}((2 \times Q \times R_1 \times f_{OUTx})/(M \times f_{REFx1}))$$

where:
round() is a function to round the quantity in () to the nearest integer, assuming the quantity is within the range of the Profile x.y buildout N-divider bit field.
Q is the Q divider.
M is the M divider (see Figure 62).

DPLL LOOP FILTER

DPLL0 and DPLL1 make use of a digital IIR filter loop filter. The digital filter is analogous to the third-order analog filter shown in Figure 69, where input is on the left (from the phase detector) and output is on the right (to the VCO).

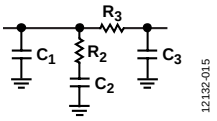


Figure 69. Third-Order Analog Loop Filter

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 is a function of numeric coefficients and is not subject to thermal drift and aging. Figure 70 is a block diagram showing the digital loop filters and their control elements.

For completeness, Figure 70 shows the fast acquisition control block of the AD9545 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 (FACQ) 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, namely, 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, entry into holdover or freerun mode).

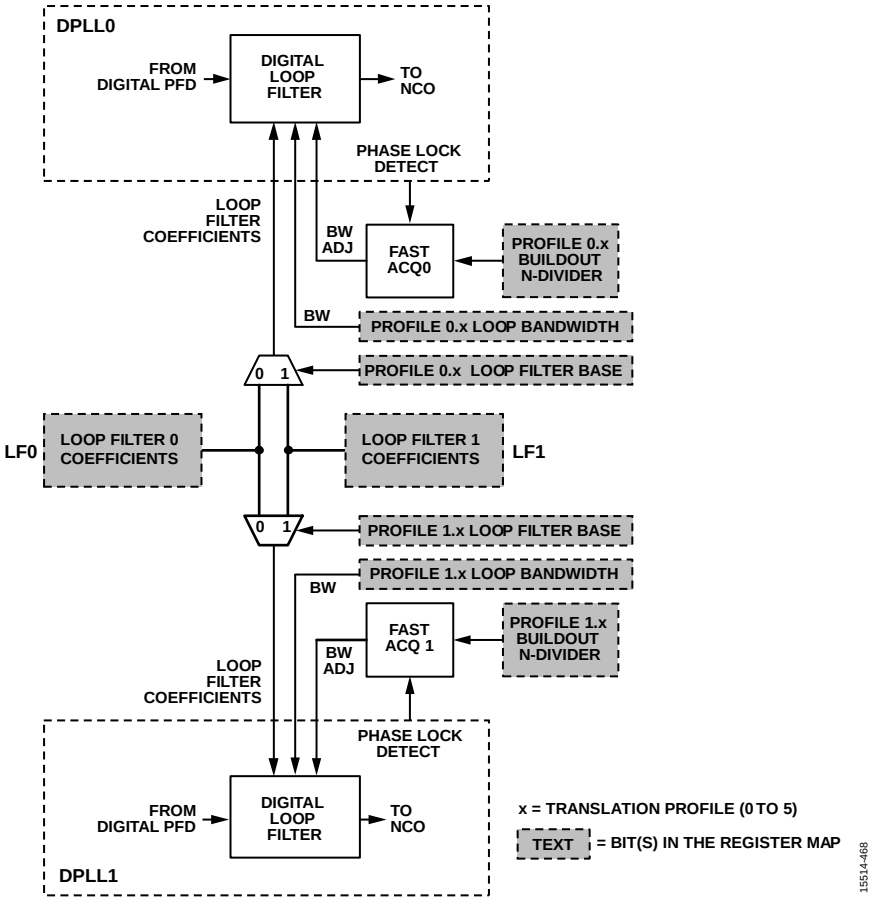


Figure 70. Digital Loop Filter Block Diagram

DPLL Loop Filter Base Coefficients

The response of characteristic of a digital loop filter is a function of digital coefficients. In the case of the AD9545, 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 AD9545 offers two independent sets of base coefficients, LFx (where x is 0 or 1), residing in Register 0x0C00 to Register 0x0C17. The LF0 coefficients are Alpha 0, Beta 0, Gamma 0, and Delta 0. The LF1 coefficients are Alpha 1, Beta 1, Gamma 1, and Delta 1. Each coefficient has two components—a significand and an exponent (for example, the Alpha Significand 0 and Alpha Exponent 0 bit fields in Register 0x0C00 to Register 0x0C02, respectively).

Each DPLL has access to LF0 and LF1 per Figure 70. To specify which response the loop filter of the DPLL uses, program the Profile x.y loop filter base bit accordingly (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)). Logic 0 (default) selects LF0, whereas Logic 1 selects LF1. The Profile x.y loop filter base bits reside in Bit D7 of Register 0x1203, Register 0x1223, Register 0x1243, Register 0x1263, Register 0x1283, 0x12A3, Register 0x1603, Register 0x1623, Register 0x1643, Register 0x1663, Register 0x1683, and Register 0x16A3.

The AD9545 provides default 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 88.5° phase margin (nominal). Furthermore, the LF1 default coefficients yield a nearly flat closed-loop response with <0.1 dB peaking.

Although it is generally not necessary to alter the response of LF0 or LF1, the user can contact Analog Devices for assistance with calculating new base coefficients to tailor the DPLL loop filters to specific requirements.

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 resides in the translation profiles as an unsigned 32-bit Profile x.y loop bandwidth bit field, as shown in Figure 70 (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)). The Profile x.y loop bandwidth bit fields reside in registers starting with Address 0x1204, Address 0x1224, Address 0x1244, Address 0x1264, Address 0x1284, Address 0x12A4, Address 0x1604, Address 0x1624, Address 0x1644, Address 0x1664, Address 0x1684, and Address 0x16A4.

The frequency units associated with the Profile x.y loop bandwidth bit field depend on the normalization of the LF0 and LF1 base coefficients (see the DPLL Loop Filter Base Coefficients section). However, for the default base coefficients, the normalized frequency units are μHz (10^{-6} Hz) for both LF0 and LF1.

For example, to set the 0 dB open-loop bandwidth to 50 Hz, divide 50 Hz by the normalized frequency established by the selected set of base coefficients (LF0 or LF1), which is 10^{-6} Hz for the default LF0 and LF1 values.

$$\textit{Profile x.y Loop Bandwidth} = 50\ \text{Hz}/10^{-6}\ \text{Hz}$$
$$= 50{,}000{,}000$$
$$= 0\text{x}02\text{FAF}080\ \text{(hexadecimal)}$$

Note that the 32-bit Profile x.y loop bandwidth bit field implies a maximum DPLL loop bandwidth of nearly 4.3 kHz. 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 41 relates the maximum DPLL loop filter bandwidth as a function of the value of the DPLLx NCO gain filter bandwidth bit field (where x is 0 or 1). The active translation profile of the DPLL determines whether the LF0 or LF1 column applies for a given DPLL channel.

Table 41. Maximum DPLL Loop Filter Bandwidth

DPLLx NCO Gain Filter Bandwidth Bit Field Value	LF0 Maximum Loop BW (HZ)	LF1 Maximum Loop BW (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 AD9545, the external clock source drives the XOA and XOB input pins, from which an integrated PLL synthesizer generates an approximately 2.4 GHz clock signal.

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 frequency tuning word (FTW) source for the NCO. However, under certain conditions, the DPLL operates in an open-loop configuration. Figure 71 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.

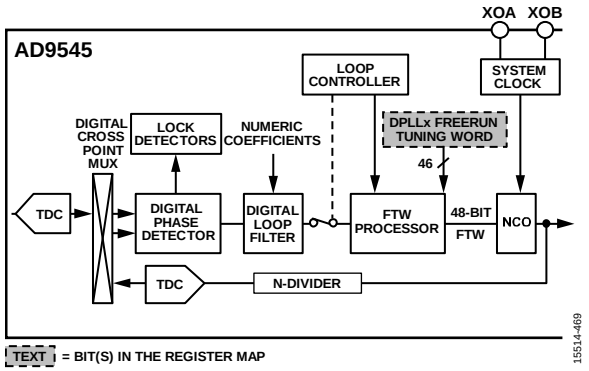


Figure 71. DPLL Block Diagram

Figure 71 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 71 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 71 shows the N-divider connected directly to the NCO output, this diagram is a simplification of the actual feedback path (see Figure 62 in the Frequency Translation Loops section). However, with regard to 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 a numerically controlled oscillator (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 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() is a function to round the value in () to the nearest integer.

The NCO automatically converts the 48-bit FTW into two components: an integer part (INT) and a fractional part (FRAC). The following constraints apply to INT and FRAC:

- $7 \le INT \le 13$
- $0.05 \le FRAC \le 0.95$

The constraints on INT and FRAC necessarily impose limitations on the choice of FTW. To determine if FTW (as prescribed by Equation 18) is valid, calculate INT and FRAC for a given FTW, per Equation 19, where the operation on the left side of the equation yields a number with an integer part and fractional part (INT.FRAC).

$$2^{48}/FTW = INT.FRAC \quad (19)$$

For example, let f_S = 2.30 GHz and f_{NCO} = 245.76 MHz, which yields the following:

$$FTW = \text{round}(2^{48} \times 245.76 \text{ MHz}/2.30 \text{ GHz})$$

$$= 30{,}076{,}213{,}163{,}657$$

Use FTW to determine INT.FRAC as follows:

$$2^{48}/FTW = INT.FRAC$$

$$2^{48}/30{,}076{,}213{,}163{,}657 = 9.3587239583$$

That is, INT = 9 and FRAC = 0.3587239583, which satisfies the $7 \le INT \le 13$ and $0.05 \le FRAC \le 0.95$ constraint.

Although the preceding example validates FTW for the given f_S and f_{NCO}, it does not necessarily validate FTW for a give 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 variation on FTW for a given application. That is, there is an upper and lower FTW value to consider, which leads to upper and lower INT.FRAC values as well.

For example, assume in the preceding example that the particular application causes 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.FRAC values:

Lower INT.FRAC = 9.4057527219

Upper INT.FRAC = 9.3121631426

In this case, both INT values and both FRAC values satisfy the constraints on INT and FRAC.

It is imperative that the upper and lower INT values be the same. Otherwise, it 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 it 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 71, 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 72), 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).

To control the filter bandwidth use the 4-bit unsigned DPLLx NCO gain filter bandwidth bit field (where x is 0 or 1) in Bits[D3:D0] of Register 0x1009 and Register 0x1409 per Table 42.

Table 42. NCO Gain Tuning Word Filter Bandwidth Selections

DPLLx NCO Gain Filter Bandwidth Bit Field Value	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

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 42 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 (FACQ) Options section).

The NCO gain tuning word filter has implications when using the NCO as a traditional, open-loop digital frequency synthesizer. For example, when programming the AD9545 to freerun mode (see the Freerun Tuning Word section), the DPLL operates like a traditional NCO. That is, the user can program different frequency 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 72). 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 72 (see Table 42 for the transition time to 99% of a step input).

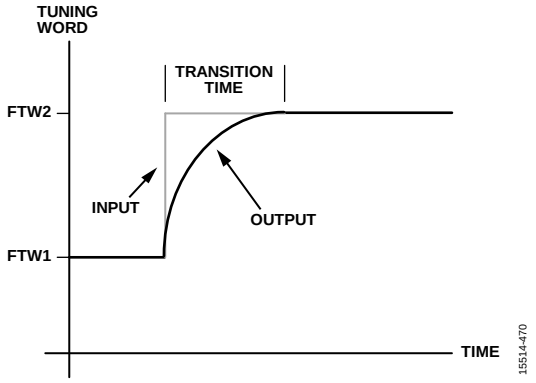


Figure 72. NCO Gain Tuning Word Filter Response

DPLL LOCK DETECTORS

DPLL Phase Lock Detector

Each DPLL channel (DPLL0 and DPLL1) contains an all digital phase lock detector. The user controls the threshold sensitivity and hysteresis of the phase detector via the source profiles (see the Source Profiles section).

The phase lock detector provides the user with a dynamic status bit, DPLLx phase lock (where x is 0 or 1), located in Bit D1 of Register 0x3100 and Register 0x3200. However, because this bit is dynamic in nature, the recommendation is to use the IRQ mechanism for phase lock indication.

The DPLLx phase locked and DPLLx phase unlocked bit pairs (where x is 0 or 1) indicate when the DPLL phase lock detector changes state via Bits[D1:D0] of Register 0x3010 and Register 0x3015. The DPLLx phase locked bit latches to Logic 1 when the DPLL changes state from not phase locked to phase locked. The DPLLx phase unlocked bit latches to Logic 1 when the DPLL changes state from phase locked to not phase locked. Because these are latched bits, they may represent a condition that is no longer true. Therefore, the user must clear these bits (via Bits[D1:D0] of Register 0x200B and Register 0x2010) to obtain visibility of subsequent state transitions of 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 73). 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. To specify the size of the fill and drain buckets, use the unsigned 8-bit Profile x phase lock fill rate and Profile x phase lock drain rate bit field (where x is a

value from 0 through 7, corresponding to a particular source profile).

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 last condition. Figure 73 shows this concept with an overlay of an example of the instantaneous water level (vertical) vs. time (horizontal) and the resulting lock/unlock states.

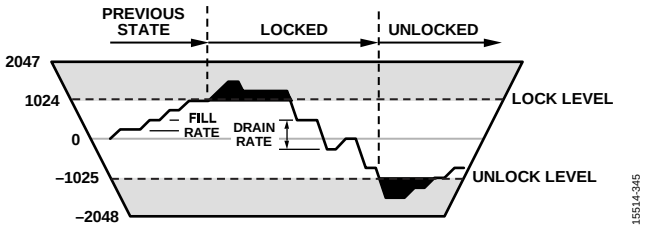


Figure 73. Lock Detector Diagram

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 threshold level specified by the user in the 24-bit unsigned Profile x phase lock threshold bit field. The bit field value 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, it removes one drain bucket from the tub. Note that it is the magnitude, relative to the phase threshold value, that determines whether to fill or drain the bucket, and not the polarity of the phase error sample.

An exception to the fill/drain process occurs when the phase slew limiter is active. When the phase slew limiter is actively in the limiting process, the lock detector blocks 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 AD9545 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.

DPLL Frequency Lock Detector

The operation of the frequency lock detector is identical to that of the phase lock detector, with 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.

The frequency lock detector provides the user with a dynamic status bit, DPLLx frequency lock (where x is 0 or 1), located in Bit D2 of Register 0x3100 and Register 0x3200. However, because this bit is dynamic in nature, the recommendation is to use the IRQ mechanism for frequency lock indication (see below).

The DPLLx frequency locked and DPLLx frequency unlocked bit pairs (where x is 0 or 1) indicate when the DPLL frequency lock detector changes states via Register 0x3010, Bits[3:2] and Register 0x3015, Bits[3:2]. The DPLLx frequency locked bit latches to Logic 1 when the DPLL changes state from not frequency locked to frequency locked. The DPLLx frequency unlocked bit latches to Logic 1 when the DPLL changes state from frequency locked to not frequency locked. Because these are latched bits, they may represent a condition that is no longer true. Therefore, the user must clear these bits (via the Register 0x200B, Bits[3:2] and Register 0x2010, Bits[3:2]) to obtain visibility of subsequent state transitions of the frequency lock detector (see the Interrupt Request (IRQ) section).

The frequency lock detector uses the 24-bit unsigned Profile x frequency lock threshold bit field specified in units of picoseconds (where x is a value from 0 through 7 corresponding to a particular source profile) in registers starting with Address 0x0805, Address 0x0825, Address 0x0845, Address 0x0865, Address 0x0885, Address 0x08A5, Address 0x08C5, and Address 0x08E5. Thus, the frequency threshold value extends from 0 ps to 16.7 µs. It represents the absolute value of the difference in period between the reference and feedback signals at the input to the DPLL.

$$Profile\ x\ frequency\ lock\ threshold = |1/f_{REF} - 1/f_{FB}|/10^{-12}$$

where:
f_{REF} is the frequency of the signal at the DPLL PFD reference input.
f_{FB} is the frequency of the signal at the DPLL PFD feedback input.

For example, suppose the users want to set the Profile x frequency lock threshold bit field to meet the frequency threshold when the signal from the reference TDC is 80 kHz and the signal from the feedback TDC is 79.32 kHz (or vice versa).

$$\text{Profile x Frequency Lock Threshold} = |1/f_{REF} - 1/f_{FB}|/10^{-12}$$
$$= |1/80{,}000 - 1/79{,}320|/10^{-12}$$
$$= 170{,}161 \text{ (nearest integer)}$$
$$= 0\text{x}0298\text{B}1 \text{ (hexadecimal)}$$

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 71 indicates 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 open and a static FTW processor provides the 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, DPLLx force freerun = 1 in Bit D0 of Register 0x2105 or Register 0x2205), the loop controller opens the switch and the FTW processor routes the contents of the DPLLx freerun tuning word bit field to the NCO.

In this case, the 46-bit unsigned DPLLx freerun tuning word bit field (where x is 0 or 1) establishes the NCO output frequency. The DPLLx freerun tuning word bit fields reside in Register 0x1000 to Register 0x1005 and Register 0x1400 to Register 0x1405.

$$f_{NCO} = f_S \times DPLLx\ Freerun\ TW/2^{48}$$

where:
f_{NCO} is the NCO output frequency.
f_S is the system clock frequency.
DPLLx Freerun TW represents the value of the 46-bit DPLLx freerun tuning word bit field.

For example, assume a system clock frequency of 2.3 GHz and a desired NCO output frequency of 245.76 MHz. Solving the f_{NCO} equation for DPLLx freerun TW yields the required value for the DPLLx freerun tuning word bit field:

$$DPLLx\ Freerun\ TW = 2^{48} \times (f_{NCO}/f_S) \quad (21)$$
$$= 2^{48} \times (245.76 \times 10^6/(2.3 \times 10^9))$$
$$= 30{,}076{,}213{,}163{,}657 \text{ (nearest integer)}$$
$$= 0\text{x}1\text{B}5\text{AAA}007689 \text{ (hexadecimal)}$$

Although the tuning resolution of the NCO is 48 bits, the DPLLx freerun tuning word bit field supports only 46 bits. The reason is that, under all normal operating conditions, the two most significant bits of the DPLLx freerun TW calculation (see Equation 21) are always Logic 0. Therefore, the tuning word is 48 bits, but the DPLLx freerun tuning word bit field only uses the rightmost 46 bits of the tuning word.

The value of DPLLx freerun tuning word must satisfy the constraints imposed by the NCO SDM (see the DPLL NCO section).

DPLL FAST ACQUISITION (FACQ) OPTIONS

FACQ Overview

Certain applications necessitate very low loop bandwidths. 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 very long frequency acquisition and phase lock times in the range of minutes.

To overcome the long acquisition time imposed by a sub 1 Hz loop bandwidth, the AD9545 provides a built-in fast acquisition (FACQ) feature that greatly reduces the lock time of the DPLL. The FACQ feature works by automatically changing the DPLL loop filter bandwidth in a controlled manner. When the FACQ feature is active (see the FACQ Bandwidth Control section) the FACQ controller uses the following bit fields to control the fast acquisition process (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)):

- Profile x.y loop bandwidth
- Profile x.y fast acquisition excess bandwidth
- Profile x.y fast acquisition lock settle time
- Profile x.y fast acquisition timeout

These bit fields reside in Register 0x1200 to Register 0x12B7 and Register 0x1600 to Register 0x16B7.

The controller begins the fast acquisition process by applying a user defined excess bandwidth factor to the bandwidth specified by the Profile x.y loop bandwidth bit field. The controller then successively reduces the excess bandwidth until it reaches the bandwidth specified by the Profile x.y loop bandwidth bit field. 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 (see Figure 70).

FACQ Status Indicators

While the FACQ controller is in the process of performing a fast acquisition, it sets the DPLLx FACQ active bit (where x is 0 or 1). When the FACQ controller finishes the fast acquisition procedure it sets the DPLLx FACQ done bit (where x is 0 or 1). The user can query these bits to determine the state of the fast acquisition controller. These bits reside in Bits[D5:D4] of Register 0x3102 and Register 0x3202.

These bits relate to the IRQ function (see the Interrupt Request (IRQ) section) per the DPLLx fast acquisition started and DPLLx fast acquisition complete bits (where x is 0 or 1) in Bits[D3:D2] of Register 0x3012 and Register 0x3017. Because these IRQ bits are latched bits, the user must clear them via Bits[D3:D2] of Register 0x200D and Register 0x2012 to obtain visibility of subsequent state transitions of the FACQ controller.

FACQ Bandwidth Control

The FACQ feature is active when the user programs the 4-bit unsigned Profile x.y fast acquisition excess bandwidth bit field with a nonzero value. The Profile x.y fast acquisition excess bandwidth bit fields reside in Bits[3:0] of Register 0x1216, Register 0x1236, Register 0x1256, Register 0x1276, Register 0x1296, Register 0x12B6, Register 0x1616, Register 0x1636, Register 0x1656, Register 0x1676, Register 0x1696, and Register 0x16B6.

When this value is nonzero, the FACQ controller uses the value to define the maximum starting loop bandwidth (BW_0). The value in the Profile x.y fast acquisition excess bandwidth bit field defines BW_0 by scaling the value in the Profile x.y loop bandwidth bit field by a power of two.

Assume the Profile x.y loop bandwidth bit field is set to yield a 0.0001 Hz (100 µHz) loop bandwidth. To set BW_0 to 0.0512 Hz, program Profile x.y fast acquisition excess bandwidth = 9.

$$BW_0 = 0.0001\ \text{Hz} \times 2^{\textit{Profile x.y Fast Acquisition Excess Bandwidth}}$$
$$= 0.0001\ \text{Hz} \times 2^9$$
$$= 0.0001\ \text{Hz} \times 512$$
$$= 0.0512\ \text{Hz}$$

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 FACQ 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 value specified in the Profile x.y loop bandwidth bit field. By using wider loop bandwidths during the signal acquisition process, the fast acquisition feature achieves lock much quicker than the DPLL. The phase lock characteristic of the loop is subject to the DPLL lock detector settings (see the DPLL Lock Detectors section).

FACQ Settling Time Control

Because each step in the FACQ process relies on phase lock indication to continue on to the next step, it is important that the phase lock indication 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, it may cause the FACQ controller to advance prematurely.

To help minimize the effect of a chattering phase lock indication, the FACQ controller allows the user to specify a lock indication settling time via the 3-bit unsigned Profile x.y fast acquisition lock settle time bit field. The Profile x.y fast acquisition lock settle time bit fields reside in Bits[D2:D0] of Register 0x1217, Register 0x1237, Register 0x1257, Register 0x1277, Register 0x1297, Register 0x12B7, Register 0x1617, Register 0x1637, Register 0x1657, Register 0x1677, Register 0x1697, and Register 0x16B7.

The value of the Profile x.y fast acquisition lock settle time bit field tells the FACQ controller how long to observe a nonchanging phase lock indication before moving on to the next step. Table 43 show the available settling time durations.

Table 43. FACQ Lock Detect Settling Time

Profile x.y Fast Acquisition Lock Settle Time Bit Field Value	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

In operation, if the DPLL indicates a phase unlock within the settling period, the FACQ controller does not advance to the next step. Instead, it 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.

FACQ Timeout Control

To prevent the FACQ 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 FACQ 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 FACQ controller to wait for phase lock via the 3-bit unsigned Profile x.y fast acquisition timeout bit field. The Profile x.y fast acquisition timeout bit fields reside in Bits[D6:D4] of Register 0x1217, 0x1237, Register 0x1257, Register 0x1277, Register 0x1297, Register 0x12B7, Register 0x1617, Register 0x1637, Register 0x1657, Register 0x1677, Register 0x1697, and Register 0x16B7.

The available timeout values appear in Table 44. If the FACQ controller does not observe phase lock and settling within the prescribed timeout interval, the FACQ controller automatically advances to the next step in the fast acquisition process. When the FACQ controller reaches the final bandwidth step and the DPLL indicates phase lock within the specified settle time, the FACQ controller sets the DPLLx FACQ done bit in Bit D5 of Register 0x3102 or Register 0x3202. However, if the FACQ controller reaches the last step due to a timeout, it does not set the DPLLx FACQ done bit.

Table 44. FACQ Timeout

Profile x.y Fast Acquisition Timeout Bit Field Value	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

FACQ Trigger

The preceding text describes the timing of the fast acquisition process but not what triggers it. The user controls what triggers the fast acquisition process per the following four bits that reside in Bits[D3:D0] of Register 0x2106 and Register 0x2206:

- Enable DPLLx fast acquisition from freerun
- Enable DPLLx fast acquisition from holdover
- Enable DPLLx fast acquisition first
- Enable DPLLx fast acquisition no output

The FACQ controller treats the specific restrictions implied by these bits in logical OR fashion. That is, the controller obeys all the restrictions that apply per the set bits. Note that when all four of these bits are Logic 0, no restrictions apply. That is, the FACQ controller is completely unrestricted; the DPLL always uses fast acquisition. However, making any one of these bits Logic 1 puts the FACQ controller into a restricted operating mode where it must obey the restrictions explicitly as follows. Fast acquisition triggers under four sets of conditions.

- When the enable DPLLx fast acquisition from freerun bit is Logic 1 and the DPLL enters closed-loop operation from freerun mode, fast acquisition triggers.
- When the enable DPLLx fast acquisition from holdover bit is Logic 1 and the DPLL enters closed-loop operation from holdover mode, fast acquisition triggers
- When the enable DPLLx fast acquisition first is Logic 1 and this is the first execution of the fast acquisition process, fast acquisition triggers. That is, the FACQ controller has not completed a fast acquisition sequence previously (more specifically, when the FACQ done bit is Logic 0). When the FACQ done bit is Logic 1, the user can clear it by writing Logic 1 to the autoclearing clear DPLLx fast acquisition done bit in Bit D3 of Register 0x2107 or Register 0x2207. This provides a mechanism to have subsequent first acquisition events.
- When the enable DPLLx fast acquisition no output is Logic 1, and all the DPLLx clock distribution outputs are not toggling, fast acquisition triggers.

As an example of FACQ trigger control, assume the enable DPLLx fast acquisition from holdover and enable DPLLx fast acquisition first bits 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 in 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 74). 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 AD9545 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 the 40-bit signed DPLLx phase offset bit field (where x is 0 or 1), which has units of picoseconds. The DPLLx phase offset bit fields reside in Register 0x1015 to Register 0x1019 and Register 0x1415 to Register 0x1419).

Two other sources for applying a phase offset also feed the summing node (not explicitly shown in Figure 74). 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 sect ion).

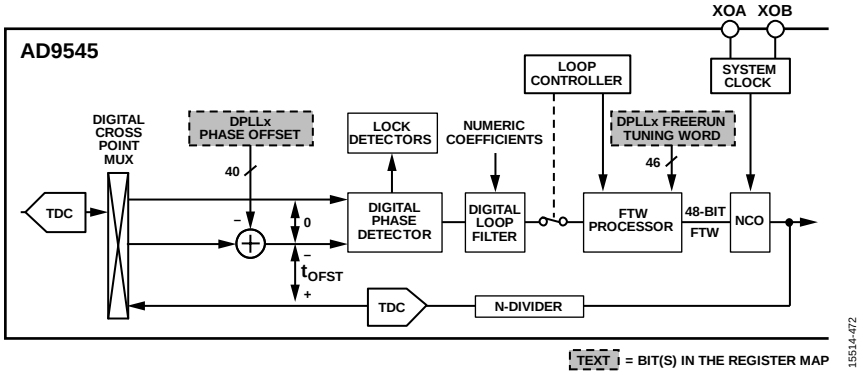


Figure 74. DPLL Phase Offset Feature

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 74. The signed nature of the DPLLx phase offset bit field means the user can 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} (seconds) and the value of the DPLLx phase offset bit field is

$$DPLLx\ Phase\ Offset = t_{OFST}/10^{-12}$$

For example, find the value of the DPLLx phase offset bit field necessary to advance the output signal by 75 ns (that is, t_{OFST} = −75 ns).

$$DPLLx\ Phase\ Offset = (-75 \times 10^{-9})/10^{-12}$$
$$= -75{,}000$$
$$= 0xFFFFFEDB08\ (hexadecimal)$$

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 is beneficial for applications in which downstream devices cannot tolerate frequencies beyond prescribed limits.

The clamp feature, as shown in Figure 75, uses the 24-bit unsigned DPLLx freerun tuning word offset clamp bit field (where x is 0 or 1) in Register 0x1006 to Register 0x1008 and Register 0x1406 to Register 0x1408. The frequency clamp feature is always active. However, the default value of the DPLLx freerun tuning word offset clamp bit field is a maximum value, which establishes a default frequency clamp limit of approximately ±586 kHz (for a system clock frequency of 2.4 GHz). An NCO output frequency of 320 MHz equates to an offset of approximately 1800 ppm (or 0.018%).

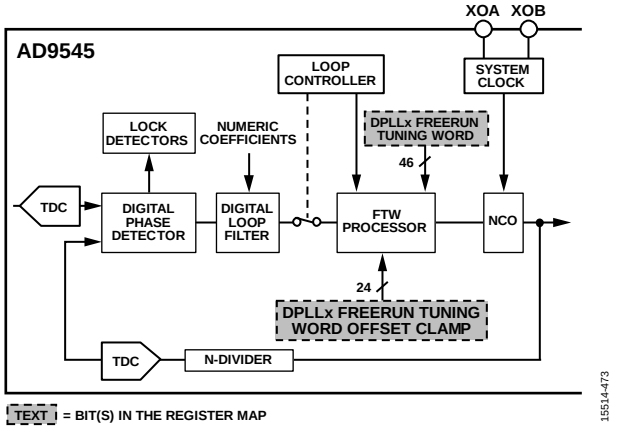


Figure 75. Tuning Word Offset Clamp Feature

The frequency offset, f_{CLAMP}, defines the separation between a center frequency, f_0, and the upper and lower frequency bounds as shown in Figure 76. The Example A traces show f_0 as a fixed value over time, whereas the Example B traces show the center frequency can assume different values over time.

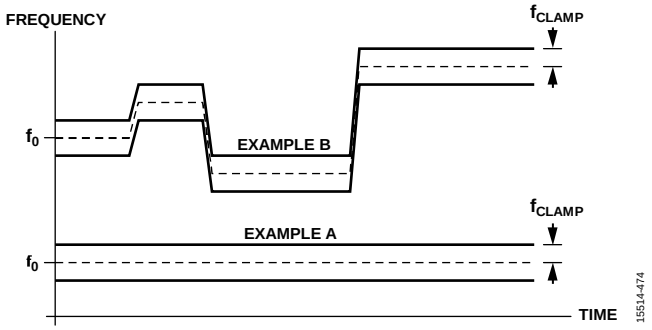


Figure 76. Example Frequency Clamp Plot

The center frequency (f_0) for the frequency clamp feature has three possible sources. One source is the DPLLx freerun tuning word bit field (where x is 0 or 1). This source is in effect when DPLLx force freerun = 1 (where x is 0 or 1) in Bit D0 of Register 0x2105 and Register 0x2205.

The second source of f_0 is the DPLL loop filter. This source is in effect when the tuning word history feature is either inactive or is active but has not had sufficient time to produce a valid result.

The final source of f_0 is the output of the tuning word history block (see the Tuning Word History section). This source is in effect when the tuning word history feature is active and the tuning word history block has had sufficient time to produce a valid result.

Switching between DPLL operating modes (freerun to active to holdover, for example) results in different sources of f_0; therefore, the frequency clamp function tends to behave like Example B in Figure 76.

The relationship between f_{CLAMP}, the system clock frequency (f_S), and the value of the DPLLx freerun tuning word offset clamp bit field is as follows:

$$f_{CLAMP} = DPLLx\ Freerun\ Tuning\ word\ Offset\ Clamp \times (f_S/2^{36})$$

For example, assume a system clock frequency of 2.4 GHz and a desired frequency offset clamp limit of ±10 kHz (that is $f_{CLAMP} = 10^4$). Solving the f_{CLAMP} equation for the DPLLx freerun tuning word offset clamp bit field, DPLLx FTWOC, yields

$$DPLLx\ FTWOC = 2^{36} \times f_{CLAMP}/f_S \quad (20)$$
$$= 2^{36} \times 10^4/(2.4 \times 10^9)$$
$$= 286{,}331\ (nearest\ integer)$$
$$= 0x045E7B\ (hexadecimal)$$

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).

Assume 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}). Solve for f_{CLAMP} as

$$f_{CLAMP} = 25 \times 10^{-6} \times 250\ MHz$$
$$= 6250\ Hz$$

Next, substitute f_{CLAMP} into Equation 20:

$$DPLLx\ FTWOC = 2^{36} \times f_{CLAMP}/f_S$$
$$= 2^{36} \times 6250/(2.4 \times 10^9)$$
$$= 178{,}957 \text{ (nearest integer)}$$
$$= 0x02BB0D \text{ (hexadecimal)}$$

The frequency clamping circuitry provides a status bit indicating when it is actively clamping the frequency, namely, the DPLLx frequency clamped bit (where x is 0 or 1) in Bit D1 of Register 0x3102 or Register 0x3202.

The DPLLx frequency clamped bit relates to the IRQ function of the AD9545 via the DPLLx frequency clamp activated and DPLLx frequency clamp deactivated bits (where x is 0 or 1). The DPLLx frequency clamp activated bit (Bit D6 of Register 0x3010 or Register 0x3015) latches to Logic 1 when the frequency clamp changes state from not clamped to clamped. The DPLLx frequency clamp deactivated bit (Bit D7 of Register 0x3010 or Register 0x3015) latches to Logic 1 when the frequency clamp changes state from clamped to not clamped. Because these are latched bits, the user must clear them via their corresponding IRQ clear bits (Bits[D7:D6] of Register 0x200B and Register 0x2010) to obtain visibility of subsequent state transitions of the frequency clamp.

PHASE SLEW RATE LIMIT

The digital phase detector uses the reference and feedback TDCs to determine the associated phase error and pass it 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 DPLLx phase offset bit field (Register 0x1015 to Register 0x1019 and Register 0x1415 to Register 0x1419)
- Changes to the Profile x phase skew bit field (Register 0x080E to Register 0x0810, Register 0x082E to Register 0x0830, Register 0x084E to Register 0x0850, Register 0x086E to Register 0x0870, Register 0x088E to Register 0x0890, Register 0x08AE to Register 0x08B0, Register 0x08CE to 0x08D0, and Register 0x08EE to Register 0x08F0)
- From the delay compensation block
- As the result of a phase buildout operation

For certain phase sources, the DPLL provides a phase slew rate limiting function (see Figure 77) 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 it to a phase ramp with the maximum slope set by the user via the DPLLx phase slew limit rate bit field (where x is 0 or 1) in Register 0x1011 to Register 0x1014 and Register 0x1411 to Register 0x1414.

The phase slew rate limiter provides the user with two status bits: DPLLx phase slew limiter activated and DPLLx phase slew limiter deactivated (where x is 0 or 1). The DPLLx phase slew limiter activated bit (Bit D4 of Register 0x3010 or Register 0x3015) latches to Logic 1 when the phase slew rate limiter changes state from not limiting to limiting. The DPLLx phase slew limiter deactivated bit (Bit D5 of Register 0x3010 or Register 0x3015) latches to Logic 1 when the phase slew rate limiter changes state from limiting to not limiting. Because these are latched bits, the user must clear them via Bits[D5:D4] of Register 0x200B and Register 0x2010 to obtain visibility of subsequent state transitions of the phase slew rate limiter.

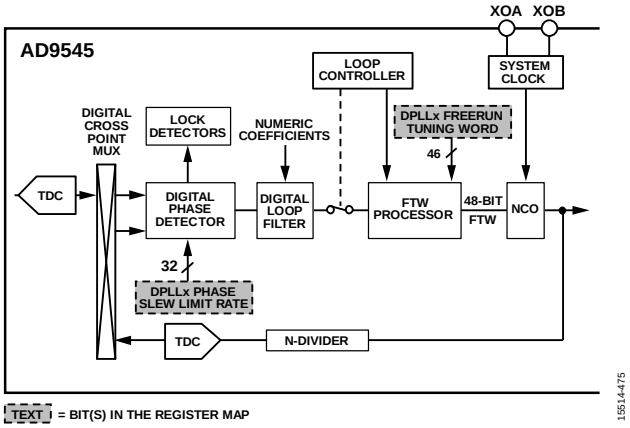


Figure 77. Phase Slew Limit Feature

The phase slew rate limiter operates only on phase transients introduced by the following phase sources:

- DPLLx phase offset bit field
- Profile x phase skew bit field
- initial phase correction due to activating a hitless mode translation profile

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 DPLLx phase offset bit field and the Profile x phase skew bit field.

The phase slew rate limit feature uses the 32-bit unsigned DPLLx phase slew limit rate bit field (units are 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 DPLLx phase slew limit rate values except zero, which disables the phase slew rate limiter (this is not recommended). However, there is little reason to disable the rate limiter. Instead, programming the rate limiter to its 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 default value of the DPLLx phase slew limit rate bit field establishes a phase slew rate limit of 100,663,296 ps/s (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 DPLLx phase slew limit rate bit field is as follows:

$$DPLLx\ Phase\ Slew\ Limit\ Rate = (\Delta t/t) \times 10^{12}$$

Calculate the value of the DPLLx phase slew limit rate bit field necessary to limit the rate of change of phase (Δt/t) to 0.25 ppm (2.5×10^{-7}).

$$DPLLx\ Phase\ Slew\ Limit\ Rate = (\Delta t/t) \times 10^{12}$$
$$= (2.5 \times 10^{-7}) \times 10^{12}$$
$$= 250{,}000$$
$$= 0x0003D090\ (hexadecimal)$$

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 appearing in Figure 71, including the loop controller, FTW processor, and the switch. The NCO can receive tuning words from three possible sources.

- DPLLx freerun tuning word bit field
- Digital loop filter
- Tuning word averaging processor

This section focuses on the tuning word averaging processor, which provides three digital outputs residing in the register map:

- 46-bit DPLLx tuning word history bit field
- DPLLx history available status bit
- DPLLx history updated bit

The DPLLx tuning word history bit field (where x is 0 or 1) resides in Register 0x3103 to Register 0x3108 and Register 0x3203 to Register 0x3208. The DPLLx history available status bit (where x is 0 or 1) resides in Bit D0 of Register 0x3102 and Register 0x3202. The DPLLx history updated bit resides in Bit D2 of Register 0x3011 and Register 0x3016. The user also has access to the DPLLx history available and DPLLx history update bits 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 a sufficient number of samples to allow a valid tuning word average computation, it sets the DPLLx history available bit to Logic 1. This setting indicates 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 DPLLx freerun tuning word bit field, depending on the configuration of the averaging processor.

The averaging processor comprises 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 DPLLx history available 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:

- Any event dependent delay
- 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 delay history control bits:

- DPLLx delay history phase lock
- DPLLx delay history frequency lock
- DPLLx delay history until not slew limiting

These bits reside in Bits[D5:D3] of Register 0x100E and Register 0x140E.

The DPLLx delay history phase lock bit (where x is 0 or 1) causes the averaging process to delay until the DPLL phase locks. The DPLLx delay history frequency lock bit (where x is 0 or 1) causes the averaging process to delay until the DPLL frequency locks. The DPLLx delay while not slew limiting bit (where x is 0 or 1) 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 hold off period to expire (assuming DPLLx history hold off time ≠ 0) and starts the averaging process (even if any of the status conditions become false after the averaging processor starts averaging).

To specify a hold off period, program the 8-bit unsigned DPLLx history hold off time bit field (where x is 0 or 1) in Register 0x1010 and Register 0x1410 with a non-zero value. The hold off period relates to the bit field value as follows:

$$Hold\ off\ period = DPLLx\ history\ hold\ off\ time \times (t_{HAT}/8)$$

where t_{HAT} is the time specified by the DPLLx history accumulation timer bit field (where x is 0 or 1). See the Averaging Processor Windowed Average section for details on the DPLLx history accumulation timer bit field.

The timed 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

The DPLLX History Accumulation Timer Bit Field

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 the 28-bit unsigned DPLLx history accumulation timer bit field (where x is 0 or 1). The DPLLx history accumulation timer bit field resides in Register 0x100A to Register 0x100D and Register 0x140A to Register 0x140D.

The units associated with the DPLLx history accumulation timer bit field are milliseconds. The 28-bit range of the DPLLx history accumulation timer bit field allows averaging time spans from 0.001 sec to 268,435.455 sec (~74.5 hours). The default value (0x0000000A) of the DPLLx history accumulation timer provides an averaging window time span of 10 ms.

The averaging window time span relates to the value of the DPLLx history accumulation timer bit field as follows:

$$Time\ span = DPLLx\ history\ accumulation\ timer \times 10^{-3}$$

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.

Determine the value of the DPLLx history accumulation timer bit field necessary to provide a 15-minute windowed average. Solving the time span equation for the bit field value yields:

$$DPLLx\ History\ Accumulation = Time\ span \times 1000$$
$$= (15 \times 60) \times 1000$$
$$= 900{,}000$$
$$= 0x00DBBA0\ (hexadecimal)$$

Programming DPLLx history accumulation timer = 0 is a special case; it bypasses the averaging processor by routing tuning word samples from the digital loop filter to both the DPLL NCO and the 46-bit tuning word history bit field. Thus, programming DPLLx 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 DPLLx history available status bit (Bit D0 of Register 0x3102 and Register 0x3202) is Logic 0 when DPLLx history accumulation timer = 0.

Changing the value of the DPLLx history accumulation timer bit field exhibits different behavior depending on the current state of the history circuit. The value matters only during hold off 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 circuit does not change states when the value of the DPLLx history accumulation timer bit field is changed.

Updating the timer value while 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 hold-off state (see the Averaging Processor Delay section), behavior is slightly different. The number of 1/8th intervals already completed remains unchanged regardless of the actual time elapsed. Future 1/8th intervals uses the new period. However, the current 1/8th interval continues to use the old value.

Note that changing the DPLLx history hold off time bit field value (see the Averaging Processor Delay section) has no effect until the beginning of the next hold-off period (in its entirety, not a sub-interval). If the behavior associated with the hold-off period, due to updating either the history period or the hold-off value, needs to reflect the new value as soon as possible, the user can restore the history circuit to its initialization state via the DPLLx clear history bit (see the DPLLx Clear History Bit section).

Tuning Word Averaging Process

In operation, the averaging processor performs averaging by partitioning the time span defined by the DPLLx history accumulation timer bit field into eight subintervals. During each sub-interval, the averaging processor computes the average of the tuning word samples for that subinterval. A full tuning word average, as defined by the DPLLx history accumulation timer bit field, constitutes the average of eight consecutive sub-interval averages.

At the termination of each successive sub-interval (given the averaging processor has set the DPLLx history available bit) the averaging processor triggers the following events:

- Update the rolling average computation
- Save the result to the DPLLx tuning word history bit field
- Update the IRQ circuitry via the DPLLx history update bit

As such, at the end of each subinterval, the averaging processor stores the result of the updated rolling average computation in the DPLLx tuning word history bit field. At the same time, the averaging processor sets the DPLLx history updated bit to Logic 1. The DPLLx history updated bit provides the user with a mechanism for knowing when the DPLLx tuning word history

bit field has fresh data. However, the DPLLx history updated bit is a latched bit; therefore, it indicates that the DPLLx tuning word history bit field has data, but not necessarily the most recent data.

DPLLx Clear History Bit

The user can clear the averaging history at any time via the DPLLx clear history bit (where x is 0 or 1) in Bit D1 of Register 0x2107 and Register 0x2207. Programming this bit to Logic 1 forces the averaging processor to clear the averaging circuitry, thereby erasing any previous tuning word history information and forces the DPLLx history available bit to Logic 0 (until a newly computed initial average is available).

Programming the DPLLx history available bit to Logic 1 does not clear the contents of the DPLLx tuning word history bit field. That is, the last update of this register remains intact until the processor calculates a new average and sets the DPLLx history updated bit, thereby notifying the user a new average is available.

DPLLx 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 DPLLx history available bit (indicating no history is available), then the DPLL, by default, uses the DPLLx freerun tuning word bit field as the tuning word for the NCO. Alternatively, the user can have the DPLL use the most recent tuning word from the loop filter instead. To use this feature, write a Logic 1 to the DPLLx single sample history bit (where x is 0 or 1) in Bit D1 of Register 0x100E and Register 0x140E.

DPLLx 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 DPLLx history accumulation timer bit field to compute a full tuning word history average and to set the DPLLx history available bit (thereby indicating the averaging processor has processed enough tuning words to compute an initial average). However, the user has the option of allowing the averaging processor to indicate the history is available earlier, by writing a Logic 1 to the DPLLx quick start history bit (where x is 0 or 1) in Bit D2 of Register 0x100E and Register 0x140E. Under this condition, the averaging processor sets the DPLLx history available bit after the first two subintervals expire and updates the DPLLx tuning word history bit field. Updates of the DPLLx tuning word history bit field 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 DPLLx history accumulation timer bit field 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 DPLLx history accumulation timer bit field.

The DPLLx single sample history bit and the DPLLx quick start history bit are not mutually exclusive. Both can be set, providing 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 DPLLx single sample history bit and the DPLLx quick start history bit.

DPLLx Persistent History Bit

The DPLLx persistent history bit (where x is 0 or 1) works in conjunction with the control bits described in the Averaging Processor Continue or Reset section. The DPLLx persistent history bit resides in Bit D0 of Register 0x100E and Register 0x140E.

When DPLLx persistent history is Logic 0, the tuning word processor resets the averaging processor based on the control bits described in the Averaging Processor Continue or Reset section below. When DPLLx persistent history is Logic 1, it 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 its 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 (where x is 0 or 1 as appropriate) provide optional control over resetting or pausing the averaging processor based on the DPLL state:

- DPLLx pause history phase unlock
- DPLLx pause history frequency unlock
- DPLLx pause history while phase slew limiting

These bits pause or reset the averaging process depending on the DPLLx persistent history bit. When DPLLx persistent history = 0, these three bits cause the averaging processor to reset. When DPLLx persistent history = 1, these three bits cause the averaging processor to pause (and then resume when the conditions dictate).

The DPLLx pause history phase unlocked bit in Bit D0 of Register 0x100F and Register 0x140F pauses or resets the averaging processor when the DPLL is not phase locked. The DPLLx pause history frequency unlock bit in Bit D1 of Register 0x100F and Register 0x140F pauses or resets the

averaging processor when the DPLL is not frequency locked. The DPLLx pause history while phase slew limiting bit in Bit D2 of Register 0x100F and Register 0x140F pauses or resets the averaging processor when the phase slew limiter is slew limiting (see the Phase Slew Rate Limit section).

The user may 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, when any of the selected status conditions is satisfied, it resets or pauses the averaging processor.

DELAY COMPENSATION

Overview

Delay compensation provides an automatic phase offset to the DPLL to counteract temperature dependent delay variation. Figure 78 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 of the delay compensation 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 78. The independent variable of the polynomial is temperature (T). The five polynomial coefficients (C_1 to C_5), which the user enters via the register map, scale corresponding powers of T. Note that there is no need for a constant term (C_0) in the delay compensation block, because the DPLLx phase offset bit field (see the DPLL Phase Offset Control section) serves that function. The Δt computation implied by Figure 78 is

$$\Delta t = \sum_{i=1}^{5} CiT^i$$

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 the external temperature bit field (Register 0x2900 to Register 0x2901). When using the internal temperature sensor, the device inherently uses the appropriate scaling units. However, when using the external temperature option, the user must provide the proper scaling (see the External Temperature Source section for details).

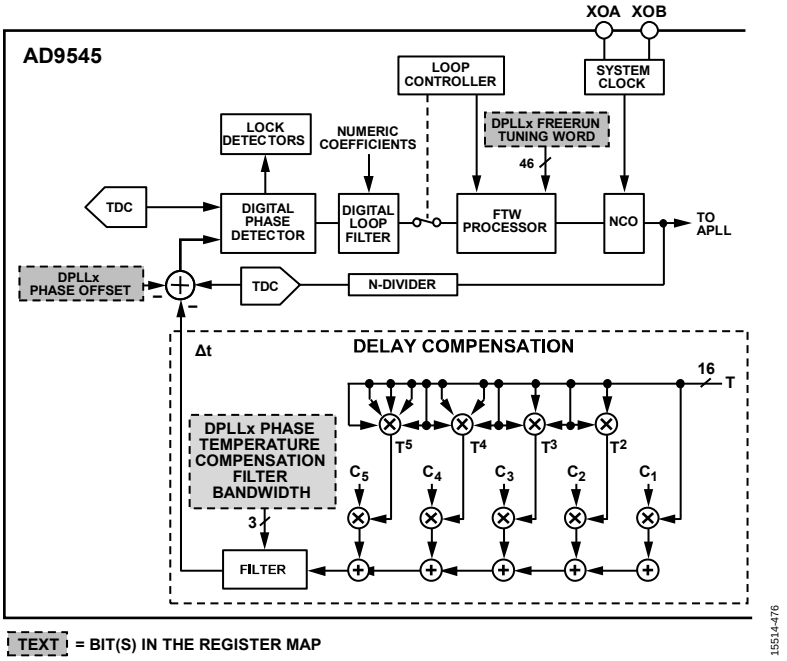


Figure 78. Delay Compensation

Programming Delay Compensation Coefficients (C_x)

Coefficient C_1 through C_5 apply scale factors to the appropriate powers of T. The C_x coefficients carry units of seconds/(°C^x). The user programs the C_1 through C_5 coefficients via the signed (twos complement) 16-bit DPLLx phase temperature compensation C_k significand and signed (twos complement) 8-bit DPLLx phase temperature compensation C_x exponent bit fields, where x (where x is 0 or 1). The C_x coefficients reside in Register 0x101A to Register 0x1028 and Register 0x141A to Register 0x1428.

The C_x coefficient bit fields (DPLLx phase temperature compensation C_x significand and DPLLx phase temperature compensation C_x exponent) program in exactly the same manner as the T^x significand and T^x exponent bit fields in the System Clock Compensation section of the register map. An example follows (see the Compensation Method 1 section for details on converting coefficient values to bit field values).

Suppose a user makes measurements that indicate the output clock signals of the AD9545 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, obtain the following bit field values for C_1:

$$C1\_ExpVal = \left\lfloor \frac{\log|C_1|}{\log 2} \right\rfloor + 1$$

$$= \left\lfloor \frac{\log|10^{-9}|}{\log 2} \right\rfloor + 1$$

$$= -29$$

This value is greater than the quantization limit of −127. Therefore, for the DPLLx phase temperature compensation (PTC) C_1 exponent,

$$DPLLx\ PTC\ C_1\ Exponent = C1\_ExpVal$$

$$= -29$$

$$= 0xE3\ (hexadecimal)$$

For the DPLLx PTC C_1 significand,

$$DPLLx\ PTC\ C_1\ significand = C_1 \times 2^{15\ -\ C1\_ExpVal}$$

$$= 10^{-9} \times 2^{15\ -\ (-29)}$$

$$= 17{,}592\ (nearest\ integer)$$

$$= 0x44B8\ (hexadecimal)$$

Delay Compensation Filter

The Δt values produced by the delay compensation block depend on temperature measurements (internal or external) 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 the 3-bit unsigned DPLLx phase temperature compensation filter bandwidth bit field (where x is 0 or 1) in Bits[D2:D0] of Register 0x1029 and Register 0x1429. The filter comprises a single-pole response yielding a 3 dB bandwidth per Table 45 (which also shows the transition time to 99% of a step input, associated with a step change at the input to the filter).

Table 45. Delay Compensation Filter Bandwidth

Binary Value	3 dB Bandwidth (Hz)	Transition Time (ms)
000	243	3.27
001	121	6.58
010	61	13.0
011	30	26.5
100	15	53.1
101	8	99.5
110	4	199
111	2	398

TIME STAMP TAGGING OPTIONS

The input to the DPLL constitutes a reference TDC source and a feedback TDC source. 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 originating from the reference and/or feedback TDCs.

The use of tagged timestamps 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 the Profile x.y tag mode bit field in Bits[D4:D2] of Register 0x1203, Register 0x1223, Register 0x1243, Register 0x1263, Register 0x1283, Register 0x12A3, Register 0x1603, Register 0x1623, Register 0x1643, Register 0x1663, Register 0x1683, and

Register 0x16A3. There are five tagged operating modes, as follows:

- No tagging (default)
- Reference tagging only
- Feedback tagging only
- Combined reference and feedback tagging with equal carrier rates
- Combined reference and feedback tagging with unequal carrier rates

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 or not the time stamp is a tagged time stamp.

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 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. Thus, 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 is used to lock both DPLLs to the same reference source, yet have both DPLLs remain phase locked, even if all reference sources are lost. To understand cascaded configuraions, first consider device operation without cascaded capability, as shown in Figure 79. As a prerequisite, the reader must be familiar with the Translation Profiles section and the Reference Switching section. Note that Figure 79 and Figure 80 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 79 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. Therefore, both DPLLs lock to a common reference as desired.

The middle portion of Figure 79 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 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 79 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 exactly 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.

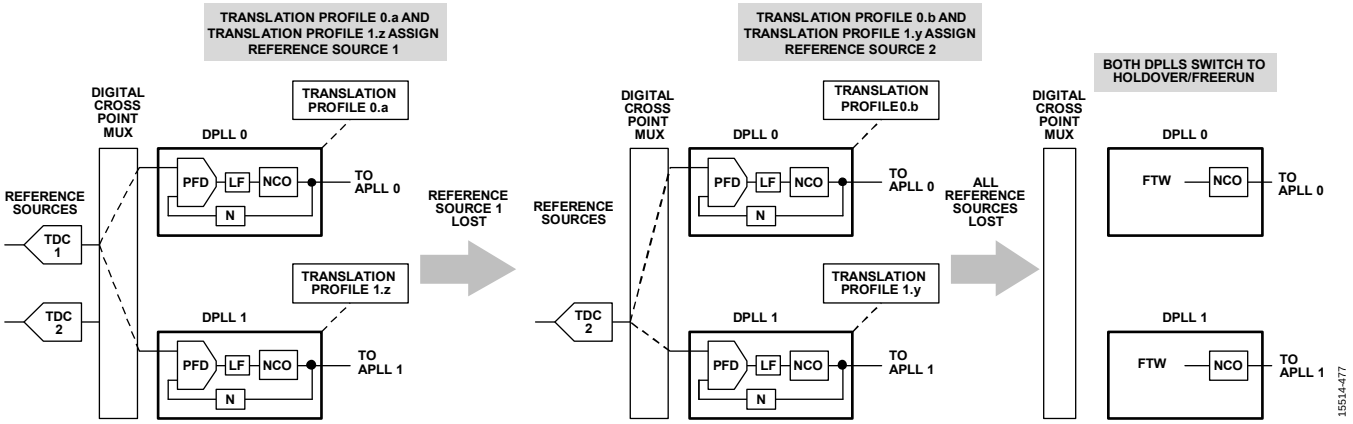


Figure 79. Non-Cascaded DPLL Operation

Figure 80 shows cascaded DPLL configuration and its operation. The main difference between Figure 79 and Figure 80 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 80).

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 the DPLLx inactive profile index bit field (where x is 0 or 1) in Bits[D2:D0] of Register 0x102A or Register 0x142A. By default, DPLLx inactive profile index = 0, which assigns Translation Profile 0.0 or Translation Profile 1.0 as the inactive profile assignment, but the user can specify any one of the six translation profiles as the inactive profile by programming DPLLx inactive profile index = n (where n is a value from 0 to 5). Figure 80 indicates Translation Profile 0.b as the inactive profile to demonstrate a nondefault example.

Consider the left side of Figure 80, which is the same as in Figure 79 except for the fact that DPLL0 has Translation Profile B assigned in its DPLL0 inactive profile index bit field. Like Figure 79, both DPLLs lock to Reference Source 1. The middle portion of Figure 80 applies when Reference Source 1 is lost. Like Figure 79, both DPLLs lock to Reference Source 2. When Reference Source 2 is lost (as shown in the right portion of Figure 80), 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 the 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 it 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.

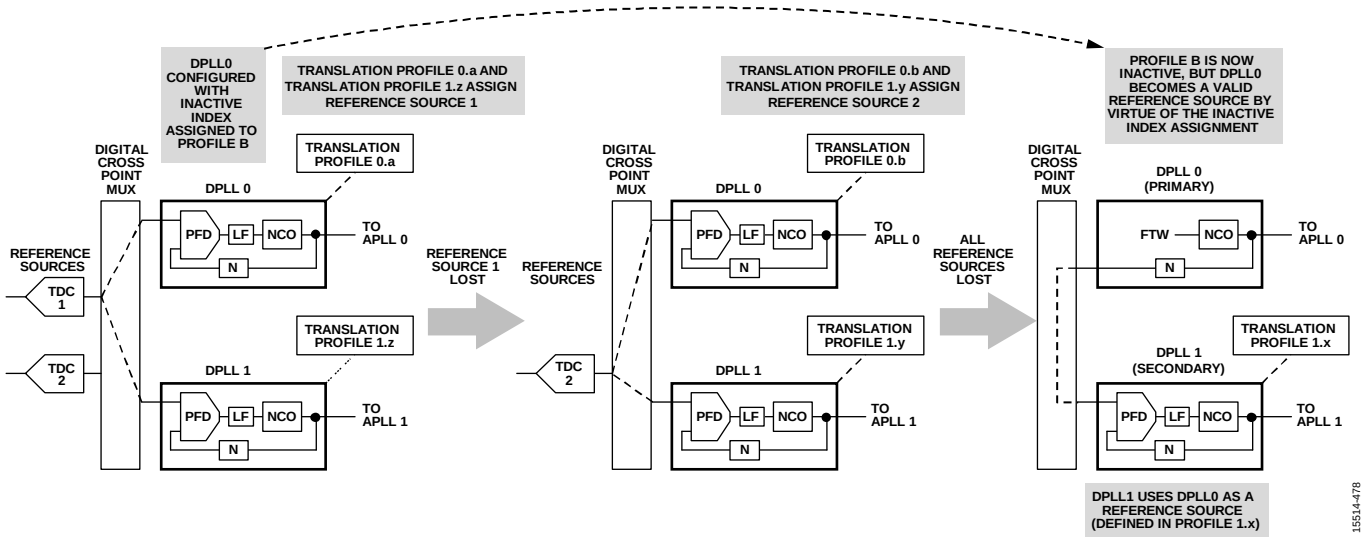


Figure 80. Cascaded DPLL Configuration

CAVEATS OF CASCADED DPLL OPERATION

To activate cascaded DPLL operation, the transition to holdover or freerun must occur by conventional means, as a result of a reference invalidation, for example, rather than by means of forced holdover or freerun via the register map.

The recommendation is to avoid using reference synchronization when employing cascaded DPLL operation (see the Reference Synchronization section).

When the profile targeted by the DPLLx inactive profile index bit field 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 sync 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 Profile x.y buildout N-divider bit field of 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 Profile x.y buildout N-divider bit field.
- The user cannot specify the external zero delay configuration for cascaded DPLL operation. That is, the enable profile x.y external zero delay bit must be Logic 0 (where x is the PLL channel (0 or 1) and y is the profile number (0 to 5)).
- The device uses the information related to timestamp tagging according to the primary DPLL. Specifically, the tagging mode defined for the feedback path of the primary DPLL determines whether timestamp 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 in the Profile x.y tag mode bit field (where x is 0 or 1 and y is 0 to 5). In addition, the translation profile for secondary DPLL must use a value of 1, 3, or 4 in the Profile x.y tag mode bit field (where x is 0 or 1 and y is 0 to 5).

ANALOG PLL (APLL)

APLL OVERVIEW

Figure 81 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 phase/frequency detector (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 clean, 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 46. The output frequency of the APPL is ½ its VCO frequency due to the P-divider (see Figure 81).

Table 46. 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 covers its approximately 800 MHz frequency range by employing 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 AD9545 incorporates an automated VCO calibration feature.

VCO Calibration

To initiate VCO calibration of APLL0 or APLL1 independently, write a Logic 1 to the appropriate calibrate APLLx bit (where x is 0 or 1) in Bit D1 of Register 0x2100 and Register 0x2200. The calibrate APLLx bit is not autoclearing; therefore, the user must immediately restore these bits to Logic 0, because maintaining a static Logic 1 state may cause unexpected device behavior.

To calibrate both APLL channels, the user can write a Logic 1 to the calibrate all bit in Bit D1 of Register 0x2000 rather than programming both calibrate APLLx bits. The calibrate all bit calibrates the system clock PLL as well.

There are two options for checking the status of the APLL VCO calibration process. One option is via Bits[5:4] of Register 0x3100 and Register 0x3200, which contain the APLLx calibration busy and APLLx calibration done bits (where x is 0 or 1). Logic 1 constitutes the status indicated by the name of the bit. The other option is via Bits[D1:D0] of Register 0x3014 and Register 0x3019, which contain the APLLx calibration started and APLLx calibration completed bits (where x is 0 or 1). Because these bits are part of the IRQ mechanism (see the Interrupt Request (IRQ) section), they constitute latched versions of the APLL calibration busy and APLL calibration done bits. The user must clear the APLLx calibration started and APLLx calibration completed bits via Bits[D1:D0] of Register 0x200F and Register 0x2014 to obtain visibility of subsequent state changes of the VCO calibration process.

The APLL calibration busy status bit is also available as an output signal via an appropriately configured Mx pin.

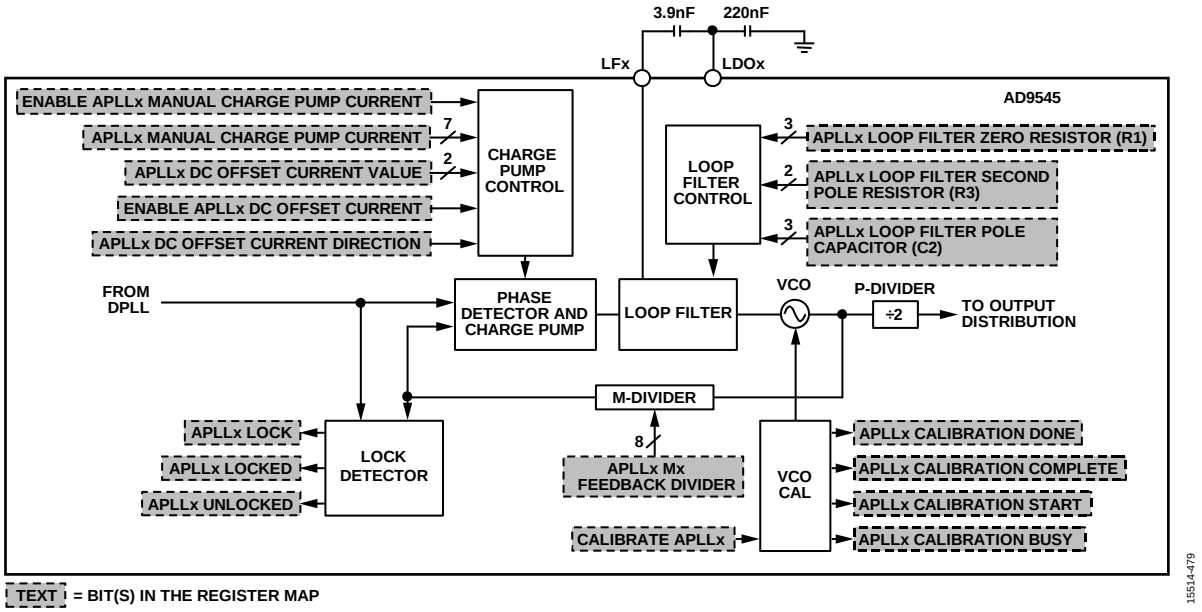


Figure 81. APLL Block Diagram

APLL FEEDBACK DIVIDER (M-DIVIDER)

The user programs the APLL feedback M-divider via the 8-bit unsigned APLLx Mx feedback divider bit fields (where x is 0 or 1) in Register 0x1081 and Register 0x1481. The divide ratio of the M-divider is the value, M, the user programs in the APLLx Mx feedback divider bit fields. The M-divider provides divide ratios from 1 to 255. The M-divider interprets a value of 0 as 1.

For a desired divide ratio of 27, the required bit field value is 27 (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 the APLLx lock bit in Bit D3 of Register 0x3100 and Register 0x3200, where Logic 1 indicates locked status. These bits are also available as an output signal via an appropriately configured Mx pin.

The IRQ section of the register map provides the APLLx locked and APLLx unlocked bits (where x is 0 or 1) in Bits[D3:D2] of Register 0x3014 and Register 0x3019. Logic 1 constitutes the status indicated by the name of the bit. Because these bits are part of the IRQ mechanism (see the Interrupt Request (IRQ) section), they constitute latched indicators of the status of the APLL lock detector. The user must clear the APLLx locked and APLL unlocked bits via Bits[D3:D2] of Register 0x200F and Register 0x2014 to obtain visibility of subsequent changes of the APLL lock or unlock state.

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 the enable APLLx manual charge pump current bit (where x is 0 or 1) in Bit D7 of Register 0x1080 and Register 0x1480. Logic 1 (default) selects manual mode, whereas Logic 0 selects automatic mode.

In manual mode, the user programs the charge pump current via the 7-bit unsigned APLLx manual charge pump current bit field (where x is 0 or 1) in Bits[D6:D0] of Register 0x1080 and Register 0x1480. The charge pump current (I_{CP}) relates to the bit field value as follows:

$$I_{CP} = APLLx\ Manual\ Charge\ Pump\ Current \times 8\ \mu A \quad (22)$$

I_{CP} allows a current range of 0 µA to 1016 µA. The default bit field value yields 128 µA.

For example, given I_{CP} = 743 µA, determine the value of the APLLx manual charge pump current bit field. Solving Equation 22 for the bit field value yields

$$APLLx\ Manual\ Charge\ Pump\ Current = I_{CP}/8\ \mu A$$
$$= 743\ \mu A/8\ \mu A$$
$$= 93\ (\text{nearest integer})$$
$$= 0x5D\ (\text{hexadecimal})$$

In automatic mode, the APLLx manual charge pump current bit field is ineffective. Instead, the APLL automatically adjusts the charge pump current (I_{CP}) based on the value of the M-divider according to Table 47. 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. In light of this constraint, it is generally best practice to avoid automatic mode.

Table 47. APLL Charge Pump Current in Automatic Mode

M-Divider Value	I_{CP}
1 to 63	M × 16 µA
64 to 255	1016 µA

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/disable the dc offset current feature use the enable APLLx dc offset current bit (where x is 0 or 1) in Bit D0 of Register 0x1083 and Register 0x1483. Logic 1 (default) enables this feature, whereas Logic 0 disables it. To control the polarity (positive or negative) of the offset current, use the APLLx dc offset current direction bit (where x is 0 or 1) in Bit D3 of Register 0x1083 and Register 0x1483. Logic 0 (default) is positive, whereas Logic 1 is negative. The magnitude of the offset current depends on the value of the 2-bit unsigned APLLx dc offset current value bit field (where x is 0 or 1) in Bits[D2:D1] of Register 0x1083 and Register 0x1483. The value of this bit field sets the dc offset current as a fraction of I_{CP} per Table 48, 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 −4 µA).

Table 48. APLL Charge Pump DC Offset Current

APLLx DC Offset Current Value Bit Field Value	DC Offset Current (% of I_{CP})
0	50
1	25 (default)
2	12.5
3	6.25

In general, the default values of the APLLx dc offset current value and APLLx manual charge pump current bit fields yield the best overall performance. For this reason, the recommendation is to use only the manual charge pump mode (that is, enable APLLx manual charge pump current = 1).

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 82 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.

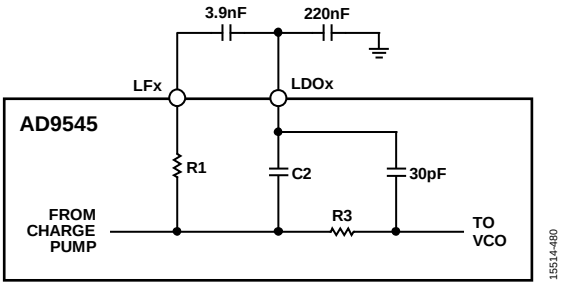


Figure 82. APLL Loop Filter

To program R1, use the 3-bit unsigned APLLx loop filter zero resistor (R1) bit field (where x is 0 or 1) in Bits[D7:D5] of Register 0x1082 and Register 0x1482. To program R3, use the 2-bit unsigned APLLx loop filter second pole resistor (R3) bit field (where x is 0 or 1) in Bits[D1:D0] of Register 0x1082 and Register 0x1482. To program C2, use the 3-bit unsigned APLLx loop filter pole capacitor (C2) bit field (where x is 0 or 1) in Bits[D4:D2] of Register 0x1082 and Register 0x1482. Table 49 shows the relationship between the bit field values and the values of the loop filter components.

Table 49. APLL Loop Filter Component Values

Bit Field Value	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 the best performance. While it is generally not necessary to alter the response of the APLL loop filter, 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 AD9545 has a sophisticated reference switching mechanism administered by an internal controller. In general, the DPLL operates in closed-loop mode, wherein it locks to the frequency and phase of a reference input signal. However, if a reference becomes invalid, the AD9545 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 AD9545 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 83. The holdover, Translation Profile x.y valid, and priority selection bubbles have associated flowcharts per Figure 84, Figure 85 and Figure 86, respectively.

The DPLL has three status indicators to notify the user when the DPLL begins the switch to a new reference input or when the DPLL enters freerun or holdover mode. These indicators (where x is 0 or 1) reside in Bits[D7:D5] of Register 0x3011 and Register 0x3016, as follows:

- DPLLx reference switching
- DPLLx freerun entered
- DPLLx holdover entered bit

The DPLLx reference switching bit latches to Logic 1 when the DPLL is in the process of switching to an active translation profile. The DPLLx freerun entered bit latches to Logic 1 when the DPLL enters freerun mode. The DPLLx holdover entered bit latches to Logic 1 when the DPLL enters holdover mode. Because these are latched bits, the user must clear them via Bits[D7:D5] 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).

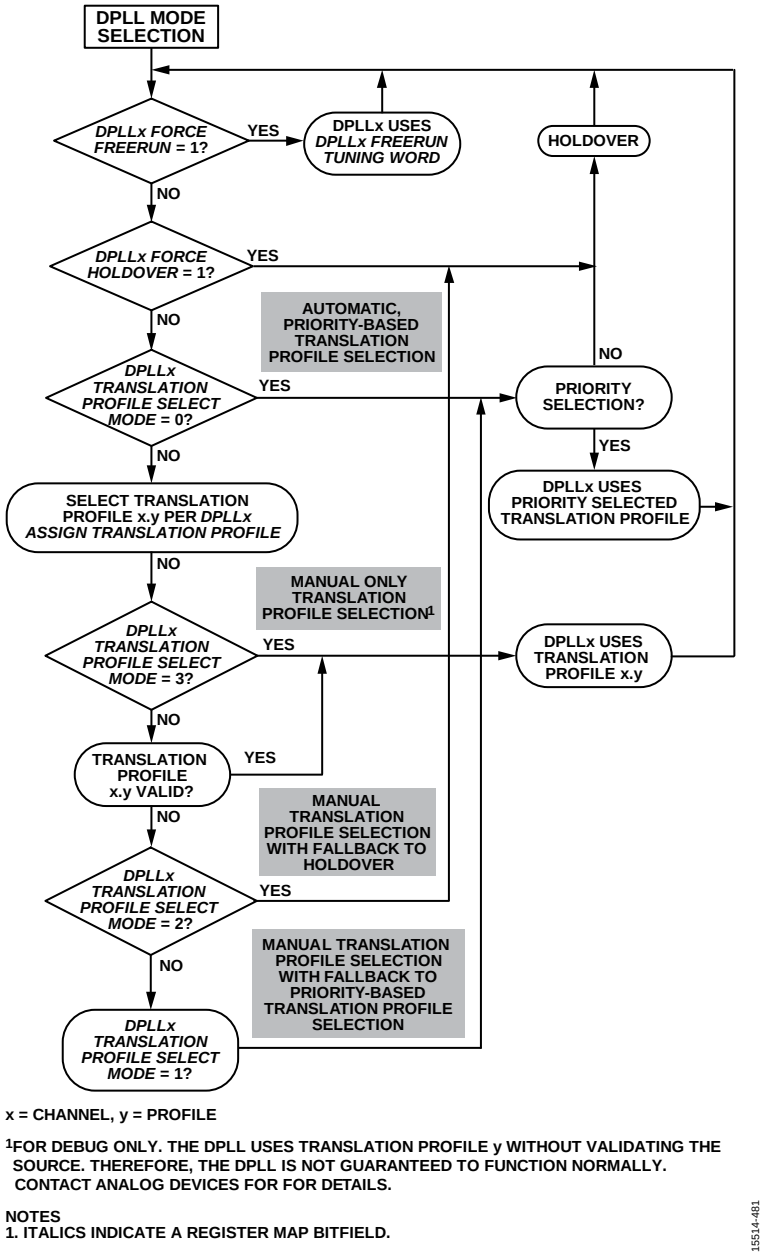


Figure 83. DPLL Mode Selection Flowchart

FORCED FREERUN MODE

Although the automatic reference switching capability of the AD9545 can cause the DPLL to switch to freerun mode, the user can force freerun mode. In freerun mode, the DPLL operates open loop and produces a static output frequency as dictated by the DPLLx freerun tuning word bit field (where x is 0 or 1). Refer to the Freerun Tuning Word section for details.

FORCED HOLDOVER MODE

Although the automatic reference switching capability of the AD9545 can cause the DPLL to switch to holdover mode, the user can force holdover mode by writing Logic 1 to the DPLLx force holdover bit (where x is 0 or 1) in Bit D1 of Register 0x2105 and Register 0x2205. 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). A summary of the holdover tuning word selection appears in Figure 84.

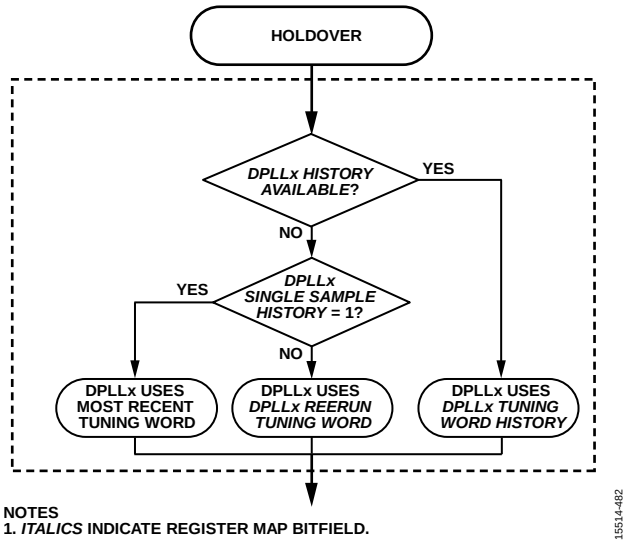


Figure 84. Holdover Flow Chart

Note that freerun mode overrides holdover mode (see Figure 83). Furthermore, freerun mode and holdover mode both override the manual/automatic source selection modes available via the DPLLx translation profile select mode bit field (see the Manual/Automatic Translation Profile Selection section).

MANUAL/AUTOMATIC TRANSLATION PROFILE SELECTION

Assuming that both the DPLLx force holdover and DPLLx force freerun bits are Logic 0, the DPLL selects a translation profile based on the 2-bit DPLLx translation profile select mode bit field (where x is 0 or 1) in Bits[D3:D2] of Register 0x2105, Bits[3:2] or 0x2205, Bits[3:2]. The DPLLx translation profile select mode bit field supports four translation profile selection modes:

- Automatic, priority-based translation profile selection (default)
- Manual translation profile selection with fallback to priority-based translation profile selection
- Manual translation profile selection with fallback to holdover
- Manual only translation profile selection

To program the DPLLx translation profile select mode bit field, select the corresponding mode (see Table 50).

Table 50. DPLLx Translation Profile Select Mode Settings

DPLLx Translation Profile Select Mode (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 (DPLLx translation profile select mode = 3) is not a recommended operating mode as it carries with it several operational caveats. Contact ADI 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 (DPLLx translation profile select mode = 2), the user specifies a particular translation profile for manual operation using the 3-bit DPLLx assign translation profile bit (where x is 0 or 1) in Bits[D6:D4] of Register 0x2105 or 0x2205. The value (0 to 5) programmed into the DPLLx assign translation profile bit field identifies the desired translation profile, x.y, where y corresponds to the value in the DPLLx assign translation profile bit field. 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 (DPLLx translation profile select mode = 1), 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, it switches to holdover or freerun operation per Figure 84.

Automatic, Priority-Based Translation Profile Selection

The priority associated with a particular translation profile depends on the programmed value of the 5-bit Profile x.y selection priority bit field (where x is the DPLL channel (0 or 1) and y is the profile (0 to 5)) in Bits[D5:D1] of register addresses: 0x1200, 0x1220, 0x1240, 0x1260, 0x1280, 0x12A0, 0x1600, 0x1620, 0x1640, 0x1660, 0x1680, and 0x16A0. The lower the value stored in the Profile x.y selection priority bit field, the higher the priority of the associated input reference source. That is, 31 is the lowest (least favored) selection priority and 0 the highest (most favored) selection priority.

In the automatic, priority-based translation profile selection mode (DPLLx translation profile select mode = 0), the DPLL relies solely on the automatic priority selection process shown in Figure 86 for choosing a translation profile. However, if the DPLL is unable to find a translation profile per the priority

selection process, it switches to holdover or freerun operation per Figure 84.

As shown in Figure 86, priority-based selection involves an automatic search through the Profile x.y selection priority bit fields in Bits[D5:D1] of register addresses: 0x1200, 0x1220, 0x1240, 0x1260, 0x1280, 0x12A0, 0x1600, 0x1620, 0x1640, 0x1660, 0x1680, and 0x16A0. The goal is to identify the translation profile with the most favored (lowest) Profile x.y selection priority bit field value for a given channel (DPLL0 or DPLL1). Thus, the DPLL selects the most favored translation profile when the current profile becomes invalid.

Translation Profile Validation

As shown in Figure 83, the manual/automatic translation profile selection modes require validation of the selected translation profile. Figure 85 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 one of the three different source types listed below (per Table 38), determining what constitutes a valid reference depends on the source type.

- REFA, REFAA, REFB, or REFBB reference input
- Auxiliary NCO 0 or NCO 1
- DPLL0/DPLL1 feedback path

The source dependent validation decisions appear in the reference source validation portion of the flowchart shown in Figure 85.

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 become 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 value of the Profile x.y selection priority bit field of the current translation profile is at least 8 more than the Profile x.y selection priority bit field of the switched from translation profile, it means that 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 Profile x.y selection priority bit field of the current translation profile is 0 to 7 more than the Profile x.y selection priority bit field 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 value of the Profile x.y selection priority bit field of the switched from translation profile to be at least 8 lower than the Profile x.y selection priority bit field of the current translation profile. Otherwise, non-revertive reference switching applies. That is, a priority separation of 8 marks the boundary between revertive and nonrevertive reference switching.

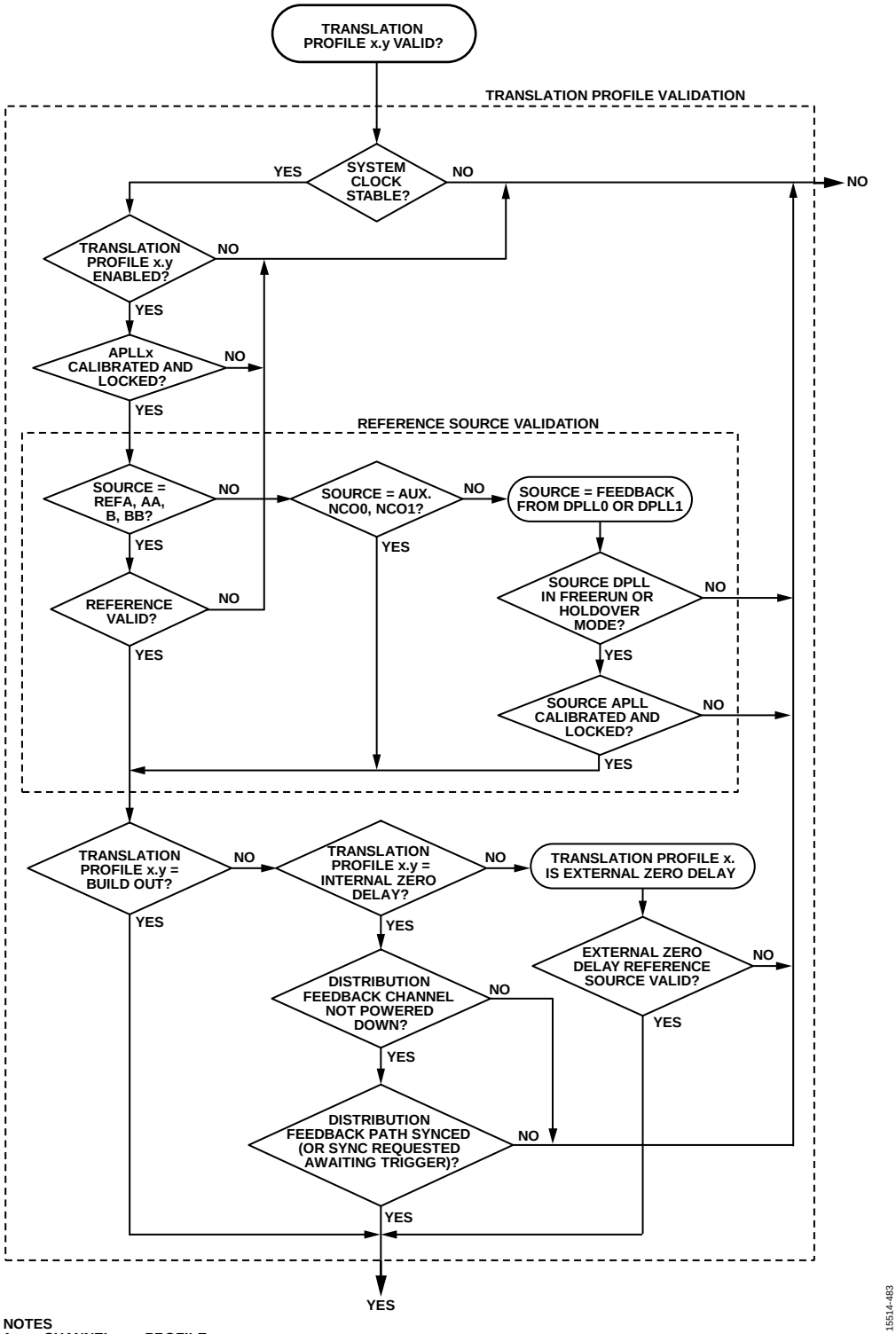
Active Profile

When the DPLL selects a particular translation profile, whether manually or automatically, the selected profile becomes active. When a translation profile is active, the corresponding DPLLx Profile y active bit is Logic 1, with Logic 0 constituting inactive. The DPLLx Profile y active bits (where x is 0 or 1 and y is the 0 to 5) reside in Bits[D5:D0] of Register 0x3009 and Register 0x300A.

The IRQ status section of the register map indicates Logic 0 to Logic 1 transitions of the DPLLx Profile y active bits via their corresponding DPLLx Profile y activated bits in Bits[D5:D0] of Registers 0x3013 and 0x3018. Because these are latched bits, the user must clear them via Bits[D5:D0] of Register 0x200E and Register 0x2013 to obtain visibility of subsequent Logic 0 to Logic 1 transitions of the DPLLx Profile y active bits (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 the 3-bit DPLLx active profile bit field (where x is 0 or 1) in Bits[D6:D4] of Register 0x3101 and Register 0x3201. The value of the 3-bit number corresponds to the most recently active profile. If a translation profile is currently active (as indicated by one of the DPLLx Profile y active bits), the DPLLx active profile bit field reflects the value of the currently active profile.



NOTES
1. x = CHANNEL, y = PROFILE.

Figure 85. Flowchart for Translation Profile Validation

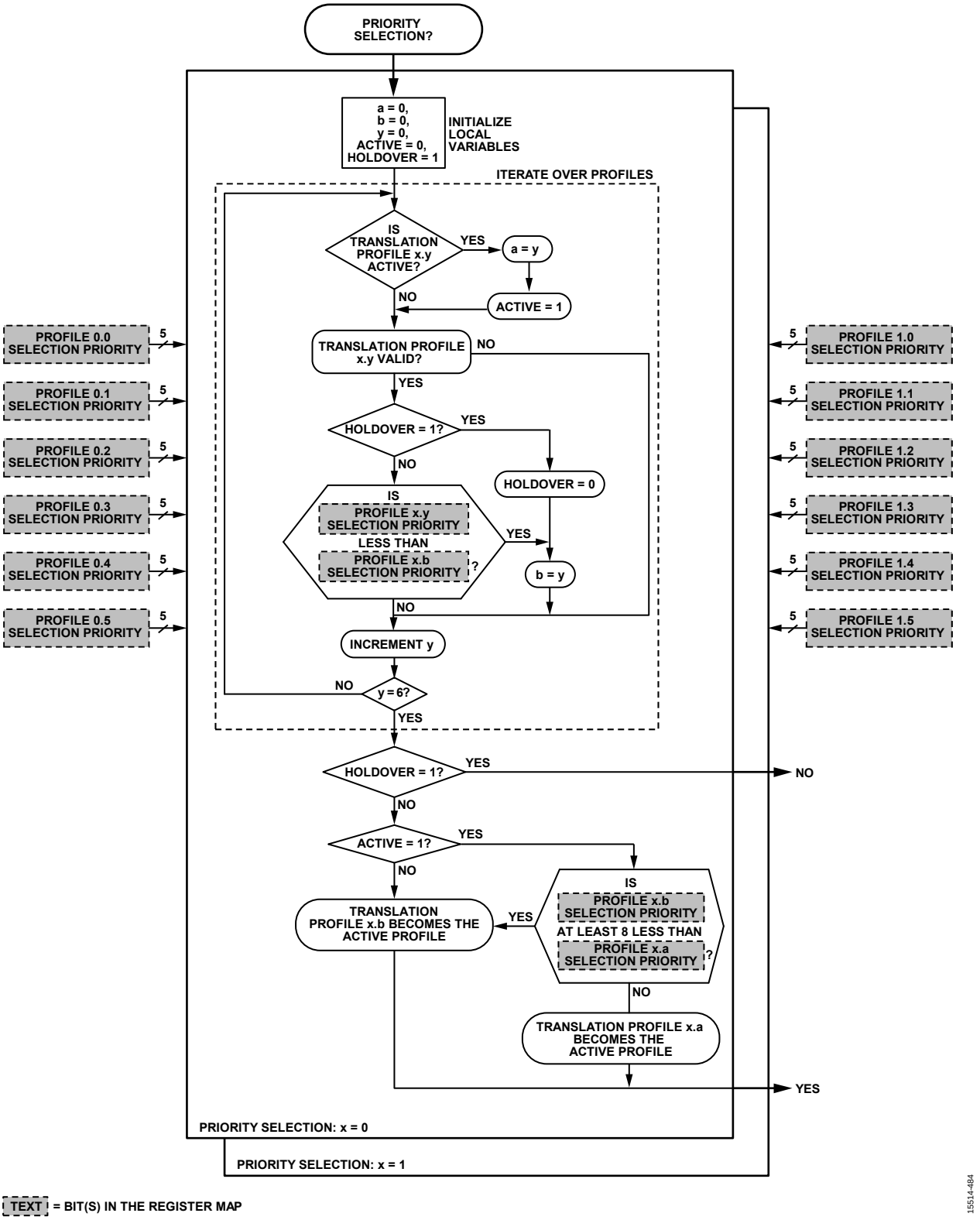


Figure 86. Priority-Based Translation Profile Selection Flowchart

TIME TO DIGITAL CONVERTER (TDC)

Time to digital converters (TDCs) are at the core of the DPLL technology in the AD9545. 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 AD9545 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 AD9545 implements the entire PLL function digitally by employing a digital PFD, digital loop filter and a NCO.

Figure 87 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.

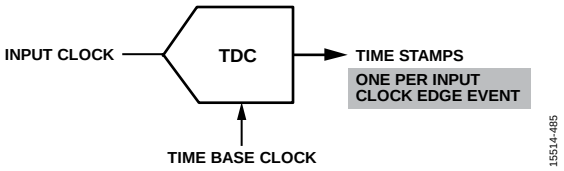


Figure 87. TDC Diagram

The AD9545 contains eight TDCs:

- Four associated with the four reference inputs at the output of the reference dividers
- Two associated with the two DPLLs at the output of the feedback divider
- Two auxiliary TDCs available to the user for general use

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. It is up to the user, however, to 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 AD9545 to provide system clock compensation, assuming the user activates system clock compensation via Register 0x0280 to Register 0x0282 (see the System Clock Compensation section).

TIMESTAMPS

TIMESTAMPS OVERVIEW

The AD9545 has 11 time stamp sources:

- REFA TDC
- REFAA TDC
- REFB TDC
- REFBB TDC
- DPLL0 feedback TDC
- DPLL1 feedback TDC
- Auxiliary TDC 0
- Auxiliary TDC 1
- Ping pong TDC
- Auxiliary NCO 0
- Auxiliary NCO 1

Most of the time stamp sources in the AD9545 are TDCs. The ping pong TDC, however, is an indirect time stamp source (see the Ping Pong TDC section of the Auxiliary TDCs section). The Auxiliary NCOs also generate timestamps (see the Auxiliary NCOs section), but without using a TDC.

In general, the timestamp 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. The exception to this rule is the ping pong TDC (see the Ping Pong TDC section).

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 Access to Time Stamps 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 AD9545. However, not every time stamp source can connect to every time stamp destination. Table 51 summarizes the interconnections of the various time stamp sources to potential destinations. X means valid connection.

Table 51. Time Stamp Source/Destination Matrix

	Time Stamp Destination				
Time Stamp Source	**DPLL0 Reference input**	**DPLL1 Reference input**	**SYSCLK Compensation PLLReference input**	**Time Skew Measurement input**	**User Time Stamp Processor Input**
REFA TDC	X	X	X	X	X
REFAA TDC	X	X	X	X	X
REFB TDC	X	X	X	X	X
REFBB TDC	X	X	X	X	X
DPLL0 Feedback TDC		X		X	X
DPLL1 Feedback TDC	X			X	X
Auxiliary TDC0			X	X	X
Auxiliary TDC1			X	X	X
Ping Pong TDC				X	X
Auxiliary NCO 0	X	X		X	X
Auxiliary NCO 1	X	X		X	X

TAGGED TIME STAMPS

A tagged time stamp is a time stamp with a special marker, or tag, attached to it. Tagged time stamps allow the AD9545 to embed a lower frequency clock 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 it as a result of the tagged time stamps.

The various time stamp sources within the AD9545 are capable of generating tagged time stamps (the auxiliary NCOs, for example). Likewise, certain time stamp destinations within the AD9545 are capable of processing only tagged time stamps while ignoring untagged time stamps (the user time stamp processors, for example). 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, a 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 the 16-bit auxiliary NCO x tag ratio bit fields (where x is 0 or 1) in Register 0x280B to Register 0x280C and Register 0x284B to Register 0x284C.

The value represented by the auxiliary NCO x tag ratio bit field, N, results in every N^{th} time stamp being tagged. That is, auxiliary NCO x tag ratio = 1 means every other time stamp is tagged, auxiliary NCO x tag ratio = 2 means every third time stamp is tagged, 3 means every fourth time stamp is tagged, 4 means every fifth time stamp is tagged, and this pattern continues up to 65,535. Note that auxiliary NCO x tag ratio = 0 (default) disables time stamp tagging (no tags).

The user can also shift the time position of the tagged flags via the 16-bit auxiliary NCO x tag delta bit fields (where x is 0 or 1) in Register 0x280D to Register 0x280E and Register 0x284D to Register 0x284E. The auxiliary NCO x tag delta bit fields are autoclearing bit fields. The value represented by the auxiliary NCO x tag delta bit field, K, 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 auxiliary NCO x tag ratio value. 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 Distribution Embedded Output Clock Modulation section). Sync 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 sync events associated with the modulation controller (see the Modulation Sync ssection). The user can program the DPLL to use the tagged time stamps from a feedback TDC.

USER ACCESS TO TIME STAMPS

USER ACCESS TO TIME STAMPS OVERVIEW

The user can observe time stamps from the various timestamp sources (see the Timestamps section) via the user timestamp processor. The AD9545 has two user timestamp processors with their output results available via the register map as shown in Figure 88.

The user selects the desired timestamp source via the 5-bit Timestamp Source x (where x is 0 or 1) bit fields in Bits[D4:D0] of Register 0x2A12 and Register 0x2A13. Each Timestamp Source x bit field gives the user access to the 11 timestamp sources, but with the interconnection constraints per Table 51.

The user timestamp processors work in conjunction with the auxiliary NCOs in that the user assigns an auxiliary NCO to a particular user timestamp processor by means of the Timebase Source x bit (where x is 0 or 1) in Bit D7 of Register 0x2A12 and Register 0x2A13. Note that either one of the user timestamp processors can be associated with either one of the auxiliary NCOs. The user timestamp processor uses elapsed time information from its assigned auxiliary NCO as the reference time for reporting user timestamps.

READING USER TIME STAMPS

The user time stamp processors make their time stamp results available via the Event x time bit fields (where x is 0 or 1) in Register 0x3A14 to Register 0x3A1D and Register 0x3A20 to Register 0x3A29. The user must issue an IO_UPDATE operation prior to reading an Event x time bit field. The processors perform time stamp conversions as the time stamps arrive from their prescribed time stamp sources. As such, the user time stamp processors are event driven. Because of this event driven process, the user cannot be inherently certain that a time stamp read from one of the Event x time bit fields is relevant. There are two methods for mitigating this problem.

The first method is to read the Event x time bit fields periodically. It is best to read the bit fields at a rate greater than the expected rate of the time stamp source assigned to the input of the user time stamp processor. 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 AD9545 (see the Interrupt Request (IRQ) section). Whenever a user time stamp processor completes a time stamp conversion, it sets the corresponding Timestamp x event status bit (where x is 0 or 1) in Bits[D3:D2] of Register 0x300F (assuming the status bit is unmasked). When the appropriate event occurs, issue an IO_UPDATE operation and read the appropriate Event x time bit field (see Figure 88).

The event driven nature of the user time stamp processor means that the user may miss one or more user time stamps if multiple time stamps occurred before the user reads one of the Event x time bit fields. The User Time Stamp x missed count bit fields (where x is 0 or 1) in Register 0x3A1E and Register 0x3A2A provide a way for the user to check for any missed input time stamps. User Time Stamp x missed count = 0 means the current user time stamp is the latest one in a contiguous series. A nonzero value in the User Time Stamp x missed count register means the user missed the indicated number of time stamps (missed time stamps are lost); therefore, the current time stamp constitutes the first one in a new series of time stamps.

In operation, the user must issue an IO_UPDATE operation, which refreshes both the Event x time and User Time Stamp x missed count bit fields with the most recent time stamp. After assertion of the IO_UPDATE bit, the user must read both bit fields simultaneously, because they occupy adjacent address locations in the register map. Doing so provides for assessing the status of the User Time Stamp x missed count value, which is an integral part of capturing the time stamp value. The User Time Stamp x missed count bit field clears 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 the User Time Stamp x low resolution bit (where x is 0 or 1) in Bit D0 of Register 0x3A1F and Register 0x3A2B. In general, the user must discard a time stamp when its associated User Time Stamp x low resolution bit is Logic 1.

There are two categories of user time stamps. The first category is generic time stamps, which the user reads via the Event x time bit fields. The second category is time stamps strictly associated with the internal time scales of the auxiliary NCOs (see the Auxiliary NCOs section).

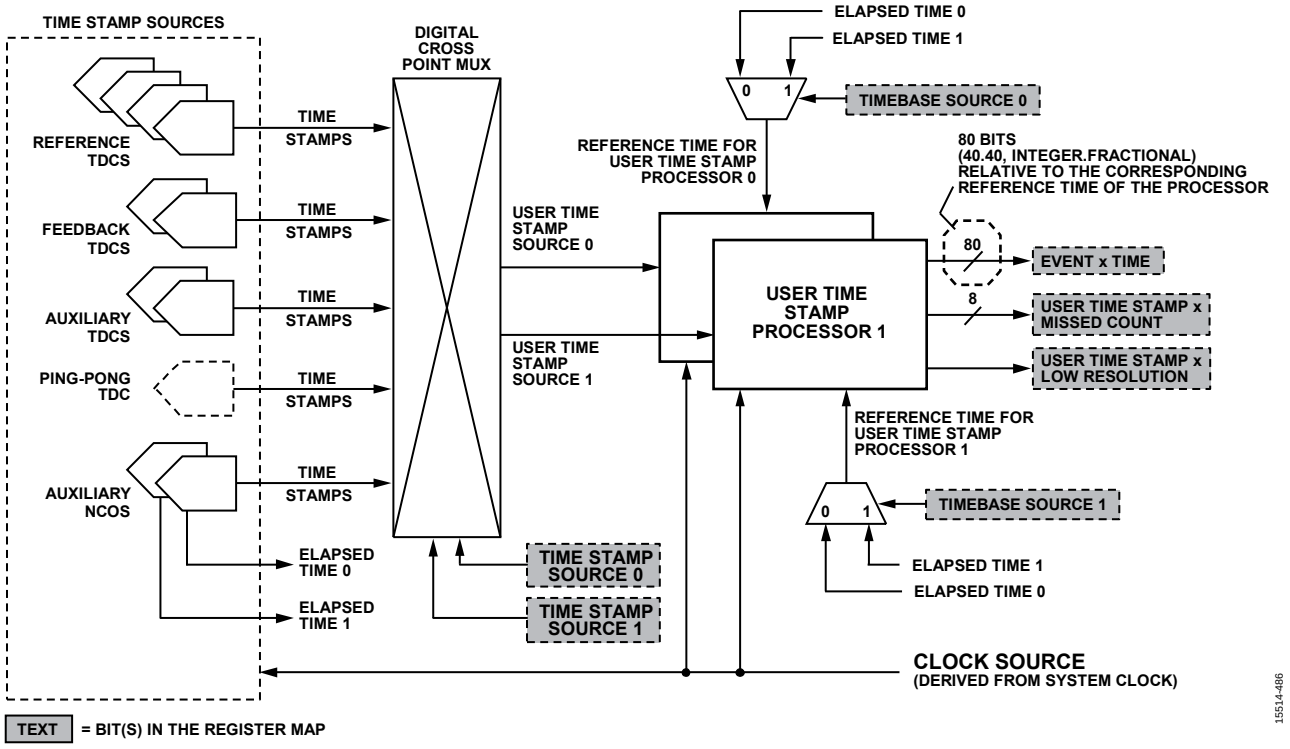


Figure 88. User Time Stamp Block Diagram

INTERPRETING USER TIME STAMPS

The user time stamps appearing in the Event x time bit fields comprise 80-bit words with the least significant bit weighted as 2^{-40}. The units associated with a time stamp result depend on the auxiliary NCO assigned to the user time stamp processor associated with the particular Event x time bit field. 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 89. Each count of the integer part represents one period of the assigned auxiliary NCO. The 40 fractional bits represent the fractional part of one period with a resolution of 2^{-40}.

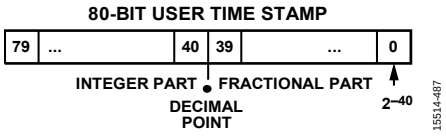


Figure 89. User Time Stamp

Assume the user time stamp value is the 80-bit hexadecimal number, 0x 0000 0000 B10E 04F0 07C1. Its 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

$$\textit{Time stamp value} = 194{,}673{,}770{,}497{,}985 \times 2^{-40}$$
$$= 177.054762842076343076769262552 26$$
$$= 177.0547628421$$

However, the value of the user time stamp is relative to the programmed period of the auxiliary NCO associated with the user time stamp processor. That is, the integer part of the user time stamp constitutes the programmed period of the auxiliary NCO associated with the user time stamp processor.

Suppose the user programs the frequency of the auxiliary NCO serving as the reference clock for the user time stamp processor with 10052.630025 Hz (the corresponding period is 99.476455167760936272992897696939 µs). Multiply the user time stamp from the previous example by the period of the auxiliary NCO to obtain a time stamp value of 0.0176127802 sec (rounded to 10 decimal places).

The user time stamp value in this example is approximately 17.6 ms. However, be aware that user time stamps have a resolution of approximately 1 ps; thus, round user time stamp results to 1 ps resolution because digits representing subpicosecond values are not meaningful.

Because the value of a user time stamp reflects time relative to the most recently completed auxiliary NCO period, it is not useful as a time clock. Auxiliary NCO rollover events have little correlation with the timing of external events. The usefulness of time stamps becomes apparent, however, when they occur in a series. In a series, the difference between successive time stamps constitutes the period underlying the source of the time stamps. For example, if one of the time stamp processors has the REFA TDC as its time stamp source, the difference between successive time stamps indicates the period of the signal at the input of the REFA TDC.

TAGGED USER TIME STAMPS

The user time stamp processor can operate on tagged time stamps from the various time stamp sources shown in Figure 88. When the selected time stamp source is generating tagged time stamps, the user can program the user time stamp processor to only pay attention to the tagged time stamps, while ignoring untagged time stamps. To enable tagged user time stamp processing program the appropriate Time Stamp Only Tags x bit (where x is 0 or 1) in Bit D5 of Register 0x2A12 and Register 0x2A13 to Logic 1.

USER TIME STAMP SYSTEM CLOCK COMPENSATION

The user time stamp processors, like the TDCs, can benefit from the system clock compensation capability. Because the user time stamp processors rely on the auxiliary NCOs as a time base reference, the source of the user time stamp processor and its assigned auxiliary NCO TDCs must use the same system clock compensation via the following bit fields (where x is 0 or 1),: compensate auxiliary NCO x via auxiliary DPLL, compensate auxiliary NCO x via DPLLx, and compensate auxiliary NCO x via coefficients (shown in Figure 104 of the System Clock Compensation section as auxiliary NCOx compensation destination bits). Otherwise, the results from the user time stamp processor are in error, because the time stamp source and auxiliary NCO are using dissimilar time bases.

TIMING SKEW MEASUREMENTS USING TWO TDCS

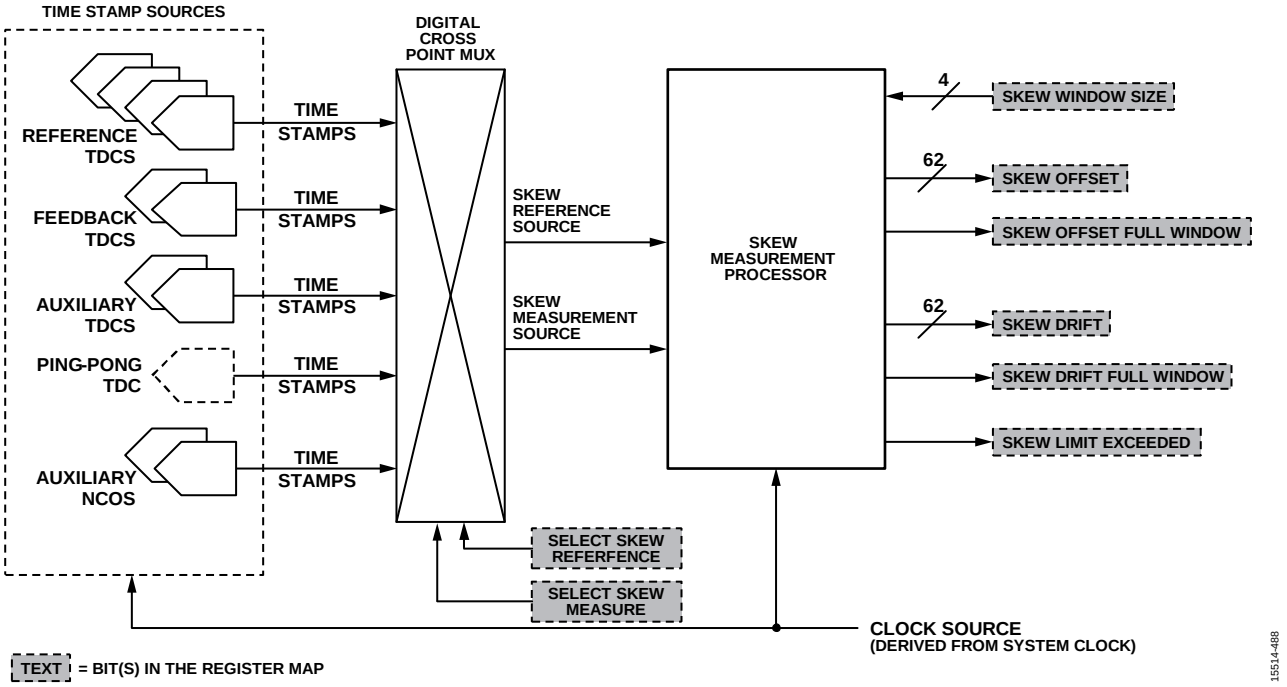


Figure 90. Skew Measurement Block Diagram

The user time stamp processor (see the User Access to Time Stamps section) allows the user to read time stamps originating from various time stamp sources. The time stamps indicate when the rising edge of a signal source occurs. Sometimes, however, it is desirable to know the relative phase offset between the rising edges of two different signal sources rather than the corresponding event times of those edges. To this end, the AD9545 provides the ability to measure and report the time difference (timing skew) between two TDCs, which 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 90 shows a block diagram of the skew measurement processor.

Time skew measurements involve a reference TDC and a measurement TDC. The user assigns the reference TDC time stamp source via the 5-bit select skew reference bit field in Bits[D4:D0] of Register 0x2A15 and the measurement TDC time stamp source via the 5-bit select skew measure bit field in Bits[D4:D0] of Register 0x2A16.

Unlike the Event x time readings (see the User Access to Time Stamps section), which are 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 the 4-bit skew window size bit field in Bits[D3:D0] of Register 0x2A14 according to Table 52.

The selected skew window size bit field value sets the number of averaging samples for both skew offset and skew drift measurements. However, a particular selection can yield a different number of averaging samples for offset measurements vs. drift measurements, as shown in Table 52.

Table 52. Skew Measurement Window Settings

Skew Measurement Window Value	Skew Offset of N	Skew Drift of N
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
15	65536	16

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, it takes a minimum of N samples for the processor to compute the desired average, after which it 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 the skew offset full window bit in Bit D7 of Register 0x3A33 or the skew drift full window bit in Bit D7 of Register 0x3A3B. These bits are 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 skew offset or skew drift bit field, the measurement is fully averaged if the corresponding skew offset full window or skew drift full window bit is set. When these bits are cleared, offset or drift readings captured are not fully averaged but are still useable. The results, however, exhibit greater variance than a fully averaged measurement.

Unlike the results from the user time stamp processor, 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.

The Skew offset bit field comprises a 62-bit signed integer (twos complement) in Register 0x3A2C to Register 0x3A33. The skew offset bit field represents time in units of 2^{-16} picoseconds (a range of approximately ±35 sec). Skew offset measurements assume the reference and measurement 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 user 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 measurement source lead time stamps from the reference source by as much as ½ UI, whereas a positive value means time stamps from the measurement 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.

Consider a skew offset value of 0x FFFFFF8A67324B5, which is −31,567,174,475 decimal. Scale this by 2^{-16} ps to yield −481,676.8566131591796875 ps. This result implies the measurement source leads the reference source by approximately 482 ns.

The skew drift bit field comprises a 62-bit signed integer (twos complement) in Register 0x3A34 to Register 0x3A3B. The skew drift bit field has units of 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.

Consider two signals with a nominal frequency of 100 Hz. A skew drift value of 0x3FFFFFF8A67324B5 is −31,567,174,475 decimal. Scale this value by 2^{-16} ps/UI to yield −481,676.8566131591796875 ps/UI. Because the reference period is 10 ms, this result indicates the measurement 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 exactly the same frequency have no drift and must yield a drift measurement of zero. However, if the frequency difference is too great, it exceeds the limits of the skew measurement processor, which is $1/16^{th}$ UI. Upon reaching this limit, the skew measurement processor sets the skew limit exceeded bit in Bit D5 of Register 0x300C.

TAGGED SKEW MEASUREMENT TIME STAMPS

The skew measurement processor can operate on tagged time stamps from the various time stamp sources shown in Figure 90. 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 the skew reference tags only bit in Bit D5 of Register 0x2A15 to Logic 1. To enable tagged skew measurement time stamp processing program the skew measure tags only bit in Bit D5 of Register 0x2A16 to Logic 1.

AUXILIARY TDCS

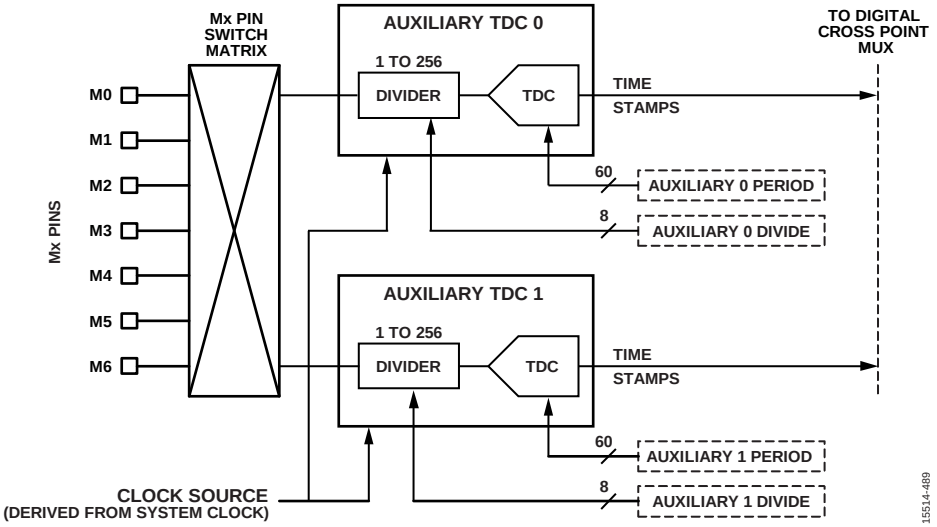


Figure 91. Auxiliary TDC Block Diagram

Typically, the TDCs in the AD9545 associate with a particular functional block. For instance, the REFA TDC generates time stamps associated with the REFA input pin. The two auxiliary TDCs, conversely, are undedicated and available for general use by the user. The input signal to the auxiliary TDCs is a clock source connected to any one of the seven Mx pins (see the Status and Control Pins section). Figure 91 shows a block diagram of the auxiliary TDCs.

To assign an auxiliary TDC to an Mx pin requires programming the appropriate Mx pin controls in Register 0x0100 to Register 0x0109 (see the Status and Control Pins section). The auxiliary TDCs generate time stamps associated with the rising edges of the signal applied to the corresponding Mx pin. The time stamps generated by the auxiliary TDCs constitute a time stamp source, which the user can connect to various time stamp destinations per Table 51. The time stamp destinations include

- Auxiliary DPLL reference input (via the Profile x reference source selection bit fields (where x is 0 or 1) associated with the translation profiles)
- User time stamp processor (via Bits[D4:D0] of Register 0x2A12 and Register 0x2A13)
- Skew measurement processor (via Register 0x2A15, Bits[4:0] and Register 0x2A16, Bits[4:0])

Each auxiliary TDC has a dedicated input divider. The user programs the divider via the 8-bit unsigned Auxiliary x divide bit field (where x is 0 or 1) in Register 0x2A00 and Register 0x2A09. The dividers have divide values spanning 1 to 256. The divide value is one more than the register value (for example, a bit field value of zero equates to a divide value of one).

To function properly, the auxiliary TDCs require the user to provide the approximate period of the signal appearing at the assigned Mx pin. Enter the expected input period in the 60-bit unsigned Auxiliary x period bit field (where x is 0 or 1) in Register 0x2A01 to Register 0x2A08 and Register 0x2A0A to Register 0x2A11. The integer value of the Auxiliary x period bit field carries units of attoseconds (10^{-18} sec).Tthe user must ensure the programmed period does not deviate from the actual input period by more than 100 ppm. However, the more accurate the programmed value is to the actual value, the better.

Assume a 2.048 MHz input clock signal. First, ensure the input frequency to the TDC is below its 200 kHz limit by programming the corresponding divider of the auxillary TDC with a value of 10 or more (that is, divide by 11 or more). To program the input period, use the period of the signal at the Mx pin (that is, at the input to the divider).

In this example, the period is 488,281,250,000 attoseconds (1/(2.048 MHz)), yielding the corresponding 60-bit hexadecimal value of 0x0000071AFD498D0.

PING PONG TDC

The ping pong TDC is a special configuration comprising both auxiliary TDCs. A simplified diagram of the ping pong TDC appears in Figure 92. The ping pong TDC effectively consists of both auxiliary TDC 0 and auxiliary TDC 1 connected to the same Mx input pin, which serves as the signal source to be time stamped.

The ping pong TDC automatically becomes active when the user assigns an Mx pin as an input and the Mx pin destination as the ping pong TDC (see the Status and Control Pins section). Because the time stamps generated by the ping pong TDC route to the digital cross point mux, the user accesses the ping pong TDC results via the user time stamp processor (see the User Access to Time Stamps section).

The TDCs in the AD9545 generally require advance knowledge of the period of the input signal. The ping pong TDC, however,

relaxes this requirement. The ping pong TDC processor monitors the two auxiliary TDCs and performs the requisite control functions necessary to observe two consecutive rising edge events without requiring advance knowledge of the input period. Because the ping pong TDC processor requires time to execute the control functions, the user must ensure at least 10 µs between a pair of successive rising edges. Otherwise, the ping pong TDC may miss or incorrectly interpret an edge event.

The ping pong TDC is especially useful for monitoring the occurrence of an input signal with arbitrarily spaced pulses. For example, suppose an input signal is a pulse that identifies some random asynchronous event with a long wait time. Because the ping pong TDC routes to the user time stamp processor, the user can set up a pulse monitor with an extended span enabling the capture of a random event.

This extended span is possible because user time stamps have a 40-bit integer part for which an increment of one corresponds to the period of its time base. Furthermore, the time base period is established by one of the auxiliary NCOs. As such, the user may, for example, program the auxiliary NCO for 1 Hz, yielding a user time stamp span of 2^{40} sec (~35,000 years) with 1 ps granularity. The ability to use a 1 Hz time base enables the ping pong TDC to monitor input pulses having an extended span between pulses.

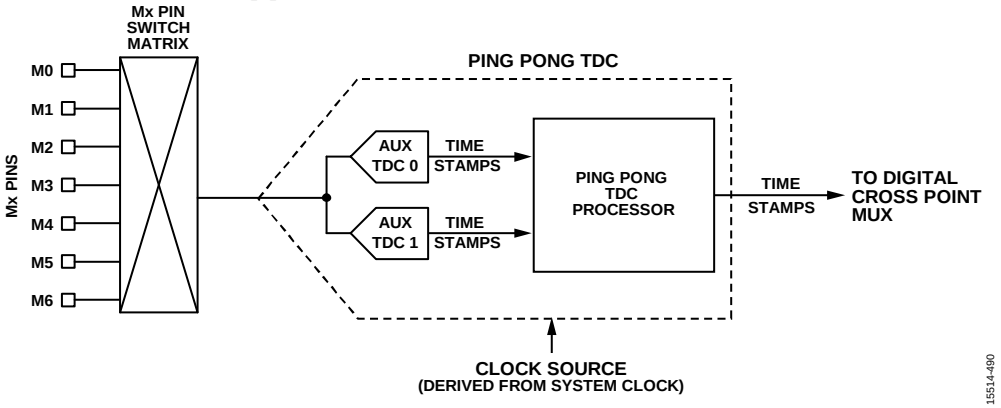


Figure 92. Ping Pong TDC Block Diagram

AUXILIARY NCOS

AUXILIARY NCO OVERVIEW

The AD9545 has two auxiliary NCOs. The NCOs generate time stamps and/or physical output signals based on a very precise programmable frequency specified by the user. The NCOs have frequency tuning resolution of 2^{-40} Hz, or approximately 1 pHz. Figure 93 shows a block diagram of the NCOs. The auxiliary NCOs also provide elapsed time information to the user time stamp processors (see the User Access to Time Stamps section).

AUXILIARY NCO FREQUENCY

The user programs the output frequency of the auxiliary NCOs by assigning a center frequency and an offset frequency.

The center frequency assignment is via the auxiliary NCO x center frequency bit fields (where x is 0 or 1) in Register 0x2800 to Register 0x2806 and Register 0x2840 to Register 0x2846. The center frequency is a 56-bit unsigned integer carrying units of 2^{-40} Hz. The offset frequency assignment is via the auxiliary NCO x offset frequency bit fields (where x is 0 or 1) in Register 0x2807 to Register 0x280A and Register 0x2847 to Register 0x284A. The offset frequency is a 32-bit unsigned integer carrying units of 2^{-24} Hz.

Figure 94 details the relationship between the center and offset frequencies. The center frequency has an upper limit of approximate 65 kHz, and the offset frequency has an upper limit of approximately 256 Hz.

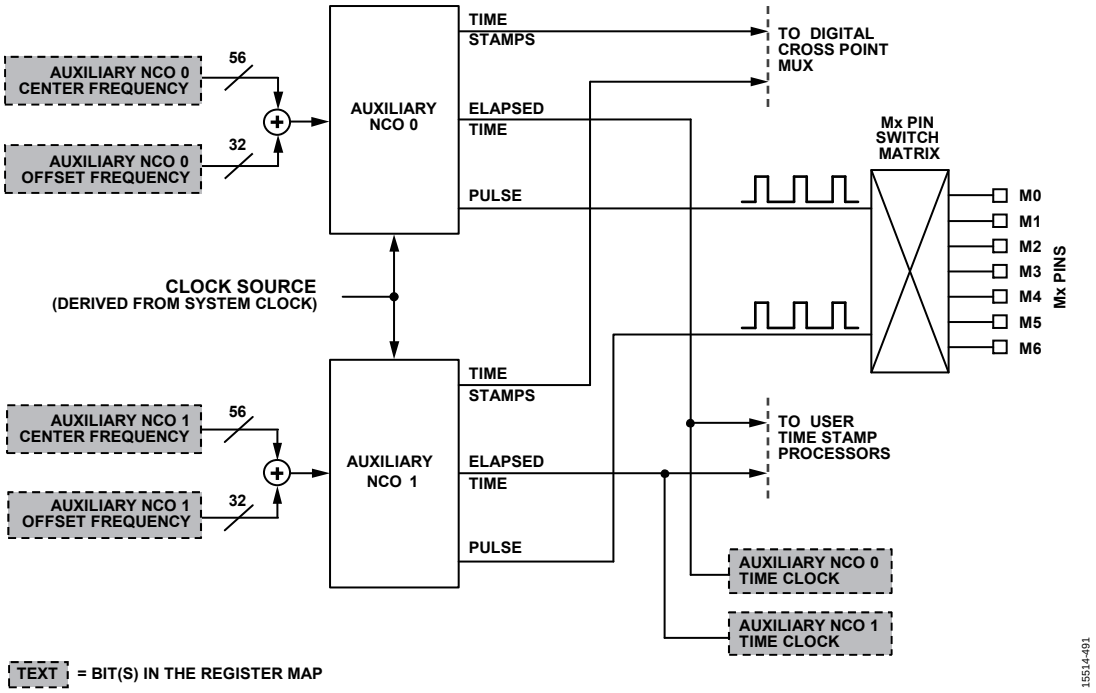


Figure 93. Auxiliary NCO Block Diagram

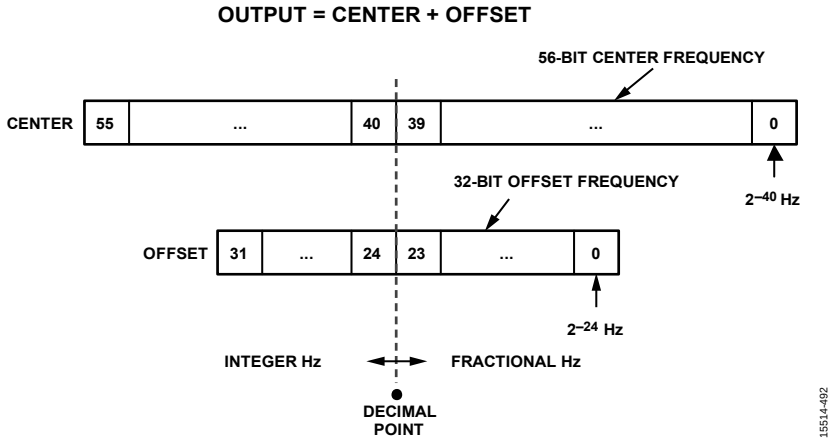


Figure 94. Auxiliary NCO Frequency Tuning

By default, the offset frequency value is zero; therefore, the default output frequency is the center frequency. The main purpose for the offset frequency is to give an external p rocessor 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 it to the register map.

The 32-bit auxiliary NCO x offset frequency bit field alleviates this problem because the user can initially program the 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, while obtaining better tuning resolution. For example, it may be possible to 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 auxiliary NCO x center frequency bit field. Using the auxiliary NCO x center frequency bit field provides 2^{-40} Hz tuning resolution (as opposed to 2^{-24} Hz tuning resolution for the auxiliary NCO x offset frequency bit field).

AUXILIARY NCO PHASE OFFSET

In addition to specifying the frequency of the auxiliary NCO, the user can specify a relative phase offset, via the signed (twos complement) 40-bit auxiliary NCO x delta bit field (where x is 0 or 1) in Register 0x2814 to Register 0x2818 and Register 0x2854 to Register 0x2858. The units associated with auxiliary NCO x delta bit field depend on the value of the auxiliary NCO x delta type bit (where x is 0 or 1) in Bit D0 of Register 0x281F and Register 0x284F.

When auxiliary NCO x delta type = 0, auxiliary NCO x delta constitutes a signed absolute time offset with units of picoseconds (a range of approximately ±0.55 sec, but the user must limit the value to ±1 period of the auxiliary NCO). When auxiliary NCO x delta type = 1, auxiliary NCO x delta constitutes a signed relative offset specified as a fraction of the period of the auxiliary NCO (a range of −½ UI to almost +½ UI). In either case, if the user specifies an auxiliary NCO x delta value exceeding the upper bound (that is, greater than 1 UI or |½ UI| per auxiliary NCO x delta type), the device sets the auxiliary NCO x delta overflow to Logic 1 (where x is 0 or 1) in Bit D7 of Register 0x3002 or Register 0x3002.

For the following two cases, to introduce a phase offset of −15.5°assume an auxiliary NCO frequency of 10 kHz per the auxiliary NCO x center frequency and auxiliary NCO x offset frequency bit fields.

Case 1: Auxiliary NCO x Delta Type = 0

A phase offset of −15.5° corresponds to a time offset of

t = (−15.5/360)/(10 kHz)

= −4.3055555556 µs (to 10 decimal places)

Because auxiliary NCO x delta requires units of picoseconds,

Auxiliary NCO x Delta = (−4.3055555556 µs)/ps

= −4,305,556 (nearest integer)

= 0xFFFFBE4D6C (40-bit hexadecimal)

Case 2: Auxiliary NCO x Delta Type = 1

Because auxiliary NCO x delta type = 1, the phase offset must be within ±180° to satisfy the ±½ UI range. A phase offset of −15.5° satisfies this requirement; therefore,

UI = −15.5°/360°

= −0.0430555556 (to 10 decimal places)

Because the auxiliary NCO x delta bit field is a signed 40-bit integer, scale the result by 2^{40}:

Auxiliary NCO x Delta = $-0.0430555556 \times 2^{40}$

= −47,340,083,974 (nearest integer)

= 0xF4FA4FA4FA (40-bit hexadecimal)

When the user programs an auxiliary NCO x delta value and asserts the IO_UPDATE bit, the AD9545 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.

AUXILIARY NCO PHASE SLEW LIMIT

When the user requests the application of a phase offset on one of the auxiliary NCOs (via the auxiliary NCO x delta bit field), the AD9545 does not necessarily apply the full phase offset all at once. Instead, the contents of the auxiliary NCO x delta rate limit bit field (where x is 0 or 1) in Register 0x2810 to Register 0x2813 and Register 0x2850 to Register 0x 2853 controls how the device implements the phase offset.

When auxiliary NCO x delta rate limit = 0, normal phase offset control ensues. That is, the device applies whatever phase offset value is in the auxiliary NCO x delta bit field all at once.

When auxiliary NCO x delta rate limit ≠ 0, the device gradually introduces phase changes by regulating the rate at which the phase changes occur to prevent exceeding the limit programmed in the auxiliary NCO x delta rate limit bit field. The device keeps track of the total phase change and terminates the process when the total phase change satisfies the value established by the auxiliary NCO x delta bit field.

The auxiliary NCO x delta rate limit bit field constitutes a 32-bit unsigned number. The units are 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. The user must program the

auxiliary NCO x delta rate limit bit field 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 per second (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 the auxiliary NCO x delta rate limit units are 2^{-36} UI/UI, the required value for the 32-bit auxiliary NCO x delta rate limit bit field is $10^{-6}/2^{-36}$ = 68,719 (rounded to the nearest integer) or 0x00010C6F hexadecimal. With the auxiliary NCO x delta rate limit bit field programmed per this example, when the user invokes a phase offset specified by the auxiliary NCO x delta bit field, the AD9545 automatically ensures that the rate of change of phase does not exceed 1 µs per second, as specified.

When auxiliary NCO x delta rate limit ≠ 0 and while the auxiliary NCO is in the process of limiting the phase slew rate, the device sets the auxiliary NCO x delta slewing bit to Logic 1 (where x is 0 or 1) in Bit D6 of Register 0x3002 or Register 0x3002.

MANUAL CYCLE ADJUSTMENT

As shown in Figure 93, the user can read the elapsed time generated by the auxiliary NCOs via the 80-bit auxiliary NCO x time clock bit fields (where x is 0 or 1) in Register 0x3A00 to Register 0x3A13. The format of the auxiliary NCO x time clock bit fields is a 40-bit integer part (most significant bits) and a 40-bit fractional part (least significant bits). The integer part tracks rollovers of the associated auxiliary phase accumulator. The fractional part effectively represents the phase of the accumulator within the current cycle. Note the contents of the auxiliary NCO x time clock bit fields indicate the elapsed time up to the moment of the most recent IO UPDATE.

The user has the option of manually overwriting the 40-bit integer portion of the auxiliary NCO x time clock bit field based on the contents of the auxiliary NCO x cycle absolute bit field and the auxiliary NCO x cycle type bit (where x is 0 or 1). The auxiliary NCO x cycle type bit (Bit D1 of Register 0x280F and Register 0x284F) controls the meaning of the auxiliary NCO x cycle absolute bit field (Register 0x2819 to Register 0x281D and Register 0x2859 to Register 0x285D). One use for this manual overwrite functionality, for example, is to force one of the auxiliary NCO x time clock bit fields to indicate time of day (assuming an auxiliary NCO frequency of 1 Hz).

When auxiliary NCO x cycle type = 0, the auxiliary NCO x cycle absolute bit field value constitutes an unsigned 40-bit integer that directly replaces the corresponding 40-bit integer portion of the auxiliary NCO x time clock bit field.

When auxiliary NCO x cycle type = 1, the auxiliary NCO x cycle absolute bit field value constitutes a signed (twos complement) 40-bit integer that increments or decrements the corresponding 40-bit integer portion of the Auxiliary NCO x time clock bit field relative to its current value.

AUXILIARY NCO TIME STAMPS

The auxiliary NCOs generate time stamps marking the rollover of the NCOs phase accumulator. The user has access to these time stamps via the 80-bit Auxiliary NCO x time clock bit fields (where x is 0 or 1) in Register 0x3A00 to Register 0x3A13. To convert the 80-bit number to auxiliary NCO time, use the same format as shown in Figure 89 (see the User Access to Time Stamps section). Because these time stamps are relative to the rollover rate of the NCOs phase accumulator, they are not useful for time stamping external events.

Auxiliary NCO time stamps are also accessible via the user time stamp processor (see the User Access to Time Stamps section). Because user time stamps are relative to the selected time base for the user time stamp processor, auxiliary NCO time stamps accessed via the user time stamp processor are relative to the most recent rollover of the selected time base.

For example, the user can program auxiliary NCO 0 with a particular frequency (1 kHz, for example) and auxiliary NCO 1 with a much lower frequency (1 Hz, for example). By programming the user time stamp processor to use auxiliary NCO 1 as its time base, accessing time stamps from auxiliary NCO 0 via the user time stamp processor indicates time relative to the most recent rollover of auxiliary NCO 1. That is, reading two consecutive time stamps from the user time stamp processor in this example yields the following: a first time stamp with an arbitrary value somewhere between 0 sec and 1 sec (because of the 1 Hz frequency programmed into auxiliary NCO 1) and the second time stamp with a value 1 ms later than the first (because of the 1 kHz frequency programmed into auxiliary NCO 0).

AUXILIARY NCO PULSE OUTPUT

In addition to generating time stamps, the auxiliary NCOs also produce a physical output signal (see Figure 93). The output signal is effectively a single pulse that occurs close to the zero crossing of the auxiliary phase accumulator. The device quantizes the pulse timing such that the peak-to-peak pulse jitter is approximately 1.4 ns.

The user programs the duration of the pulse via the auxiliary NCO x pulse width exponent in Bits[D7:D4] of Register 0x281E and Register 0x285E and the auxiliary NCO x pulse width significand bits in Bits[D3:D0] of Register 0x281E and Register 0x285E (where x is 0 or 1). The pulse width duration is as follows:

$$Pulse\ Width = (96/f_S) \times (1 + S \times 2^{(E+5)})$$

where:
f_S is the system clock frequency.
S is the decimal value of the auxiliary NCO x pulse width significand bit field.
E is the decimal value of the auxiliary NCO x pulse width exponent bit field.

For example, given f_S = 2.4 GHz, S = 6, and E = 2, then

$$\begin{aligned} \textit{Pulse Width} &= (96/f_S) \times (1 + S \times 2^{(E+5)}) \\ &= (96/(2.4 \times 10^9)) \times (1 + 6 \times 2^{(2+5)}) \\ &= (96/(2.4 \times 10^9)) \times (1 + 6 \times 2^7) \\ &= 30.76\ \mu s \end{aligned}$$

The user must ensure the programmed pulse width is less than the fundamental period of the auxiliary NCO. For this example, 30.76 µs is the maximum allowable fundamental period. Therefore, the corresponding fundamental frequency (f_{NCO_MAX}) must satisfy

$$\begin{aligned} f_{NCO\_MAX} &< 1/(\textit{Pulse Width}) \\ &< 1/(30.76\ \mu s) \\ &< 32.51\ kHz \end{aligned}$$

The key point is that the output frequency of the auxiliary NCO and the pulse width of its physical output signal are independently programmable parameters.

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). Although the user has the option of generating pulses on tagged time stamps (see the Tagged Auxiliary NCO Time Stamps section), the maximum pulse duration is the fundamental period of the auxiliary NCO.

TEMPERATURE SENSOR

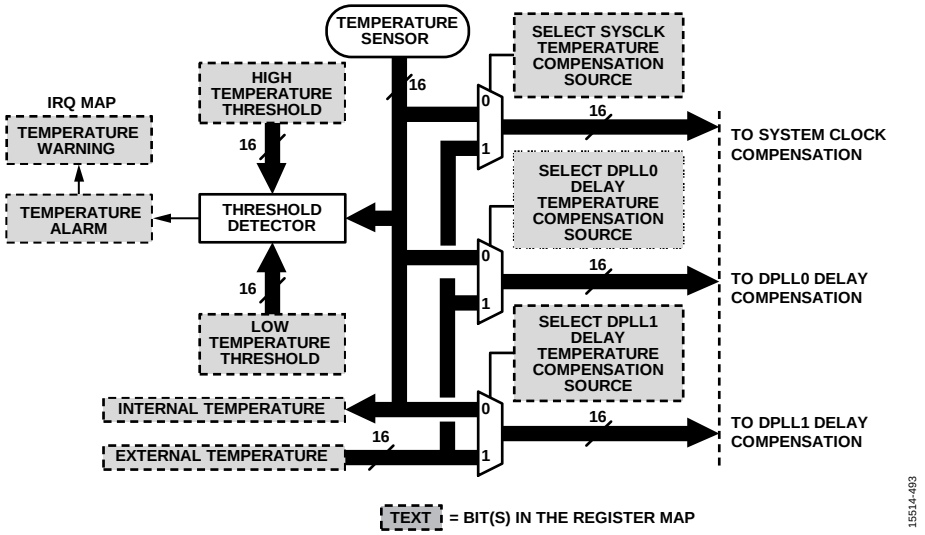


Figure 95. Temperature Sensor Block Diagram

TEMPERATURE SENSOR OVERVIEW

Figure 95 shows a block diagram of the temperature sensor and supporting circuitry. Three AD9545 functional blocks are capable 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)
- The 16-bit external temperature bit field (which follows the temperature format shown in Figure 96)

TEMPERATURE SOURCE SELECTION

The user selects the desired temperature source for each of the temperature compensation functional blocks via Bits[D2:D0] of Register 0x2902. The select SYSCLK temperature compensation source bit controls the routing of the temperature source to the system clock compensation block (see the Compensation Method 1 section). The select DPLLx temperature compensation source bits (where x is 0 or 1) control the routing of the temperature source to the DPLL0 and DPLL1 delay compensation block (see the Delay Compensation section). For each of the temperature compensation source bits, Bits[D2:D0], in Register 0x2092, 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 AD9545 has an internal temperature sensor with a digital output (see Figure 95). The sensor reflects the temperature of the silicon die, which the user can read via the internal temperature bit field in 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 96. This format provides a range of ±256°C with 0.0078°C resolution.

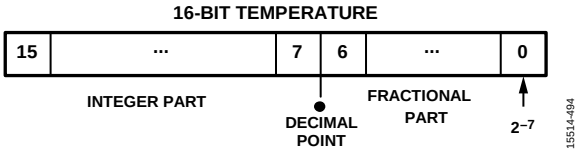


Figure 96. Temperature Format

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),

$$T = -1997 \times 2^{-7}\text{°C}$$

$$= -15.6015625\text{°C}$$

The temperature sensor also routes to a threshold detector. The threshold detector detects when temperature readings are outside the range specified by the 16-bit high temperature threshold (Register 0x2905 to Register 0x2906) and low temperature threshold (Register 0x2903 to Register 0x2904) bit

fields. When programming these bit fields, use the temperature format shown in Figure 96.

If the output value of the temperature sensor falls outside the specified limits, the temperature alarm bit in Bit D0 of Register 0x3002 assumes a Logic 1 state (otherwise, it is Logic 0). The temperature alarm bit allows the user to check the status of the threshold detector in real time.

The temperature alarm bit is the source of the temperature warning bit in Bit D4 of Register 0x300C. The temperature warning bit 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 the external temperature bit field in Register 0x2900 to Register 0x2901. Note that 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 96). This format yields a temperature range of ±256°C with 0.0078°C resolution.

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 AD9545 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 AD9545.

The stability of the system clock, in turn, relates directly to the stability of 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

$$FFE = (f - f_O)/f_O$$

or

$$FFE = f/f_O - 1$$

In the case of TDCs, the difference between successive time stamps is the period of the underlying frequency. The difference between successive time stampes 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 system clock source. The AD9545 has the option of applying system clock compensation via two methods.

- Open-loop method
- Closed-loop method

Open-Loop Method

Figure 97 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 AD9545. Although Figure 97 shows an NCO, this section also applies to a TDC.

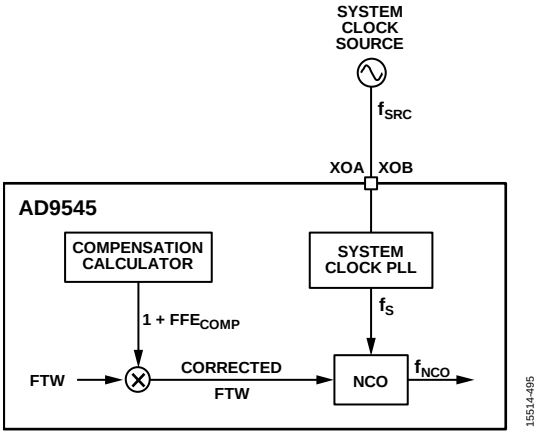


Figure 97. Open-Loop Method

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 $FFE_{COMP} = 0$ in Figure 97, the NCO produces an output frequency, f_{NCO}, proportional to the applied numeric frequency tuning word, 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 $FFE_{COMP} = -FFE$,

$$f_{NCO} = f_0 \times (1 + FFE) \times (1 - FFE) \times K_{PLL} \times FTW \times K_{NCO} \quad (23)$$

Note that 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 below 1 ppm, the value of FFE^2 is exceedingly small. Under such conditions, FFE^2 is practically zero, and Equation 23 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, the FFE associated with the system clock source is nullified by applying the fractional correction term, FFE_{COMP}.

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 AD9545, x = 5; therefore, the latter assumptions is expressible as

$$FFE_{SRC} = c_5T^5 + c_4T^4 + c_3T^3 + c_2T^2 + c_1T^1 + c_0 \quad (24)$$

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 24 equal to zero. The open-loop method assumes advance knowledge of the c_x values, which the user programs into the appropriate AD9545 register map locations.

The overall concept of the open-loop method is to program the AD9545 with the appropriate coefficients, c_x, such that the polynomial closely models the temperature dependent behavior of the system clock source. The AD9545 has a builtin compensation calculator to implement the FFE polynomial (see the Compensation Method 1 section for more details). The AD9545 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, it 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 98).

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 the stable frequency source is constant and the frequency of the system clock source is constant. However, the frequency of the stable frequency source is constant, by definition, 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, should f_{SRC} change (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 (it 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_0).

In the closed-loop method, FTW variations lead to a correction factor, 1 + FFE_{COMP}. The FTW analyzer shown in Figure 98 is an integral component of the AD9545. Applying the correction factor to other designated NCOs and TDCs in the AD9545 is the basis of the closed-loop method.

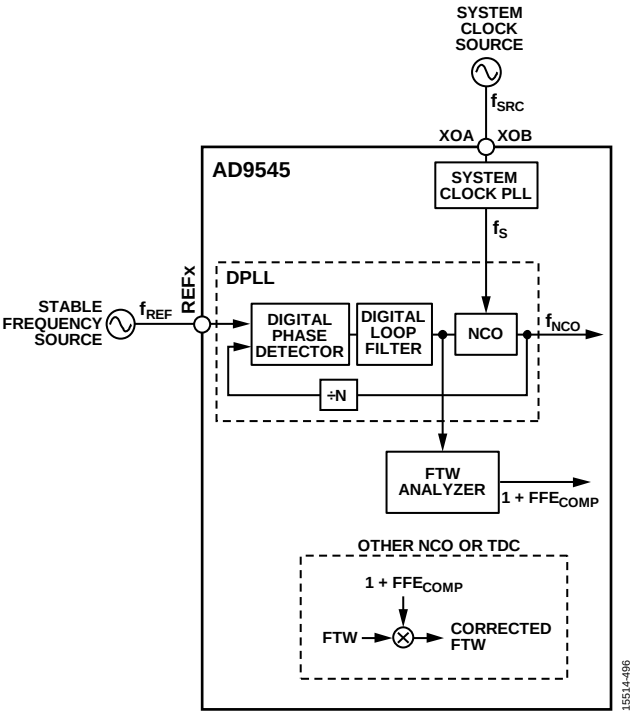


Figure 98. Closed-Loop Method

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 AD9545 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 98. The second form uses an independent special purpose DPLL (the auxiliary DPLL) as the DPLL shown in Figure 98. 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 99, 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).

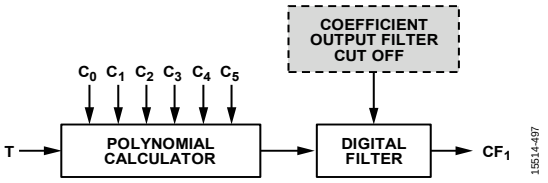


Figure 99. Compensation Method Number 1

Program the coefficients via Register 0x028A to Register 0x029C. 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 97.

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 53 shows a hypothetical example of a 25 MHz source in the two leftmost columns.

Table 53. 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

Because the AD9545 implements compensation in terms of fractional error, it is necessary to convert the frequencies in Table 53 to FFE values (refer to the System Clock Compensation Overview section). Table 53 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 fractional period error (FPE) data (refer to the System Clock Compensation Overview section for the conversion formula). Table 53 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 53 yields the following coefficients:

$$C_0 = 1.0046936 \times 10^{-3}$$

$$C_1 = -2.8927765 \times 10^{-6}$$

$$C_2 = -1.9110192 \times 10^{-7}$$

$$C_3 = 2.2493907 \times 10^{-9}$$

$$C_4 = 2.7943570 \times 10^{-11}$$

$$C_5 = -2.4817310 \times 10^{-13}$$

Figure 100 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 53.

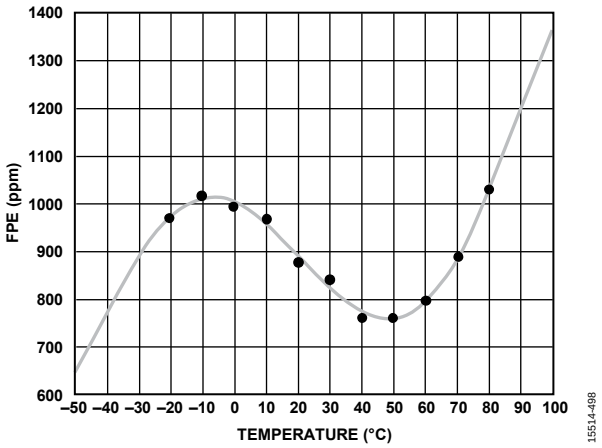


Figure 100. FPE vs. Temperature

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 (Ck) for Compensation Method 1 section for details).

Programming the Coefficients (C_k) for Compensation Method 1

Compensation Method 1 makes use of six coefficients, C_x, where the Index x ranges from 0 to 5. The user programs the constant coefficient, C_0, which is independent of the temperature parameter, via the signed 40-bit constant compensation value bit field in Register 0x0289 to Register 0x028D. The relationship between C_0 and the value stored in the constant compensation value (CV) bit field is:

$$Constant\ CV = C_0/2^{-45}$$

For example, convert $C_0 = 1.0046936 \times 10^{-3}$ to its corresponding bit field value.

$$Constant\ CV = C_0/2^{-45}$$
$$= 1.0046936 \times 10^{-3}/2^{-45}$$
$$= 35{,}349{,}513{,}305 \text{ (nearest integer)}$$
$$= 0x083AFEC459 \text{ (hexadecimal)}$$

C_1 through C_5 apply scale factors to corresponding powers of T. Program the C_1 through C_5 coefficients via the signed (twos complement) 16-bit T^x significand (Register 0x028E to Register 0x28F, Register 0x0291 to Register 0x0292, Register 0x0294 to Register 0x0295, Register 0x0297 to Register 0x0298, and Register 0x029A to Register 0x029B) and signed (twos complement) 8-bit T^x exponent bit fields (Register 0x0290, Register 0x0293, Register 0x0296, Register 0x0299, and Register 0x029C), where x corresponds to the coefficient subscript.

The C_1 through C_5 coefficients have the following format:

$$C_x = S_x \times 2^{Ex}$$

where:
S_x is the signed integer associated with the T^x significand bit fields.
Ex is the signed integer associated with the T^x exponent bit fields. In practice, Coefficient C_1 to Coefficient C_5 are Base 10 numbers. The following procedure explains how to convert Base 10 numbers to the necessary bit field format for C_1 to C_5. 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),

$$T^x\ Significand = 0 \text{ (0x0000 hexadecimal)}$$
$$T^x\ Exponent = 0 \text{ (0x00 hexadecimal)}$$

To test if C_x is less than the quantization limit, check whether the following inequality is true:

$$\left\lfloor \frac{\log|C_x|}{\log 2} \right\rfloor + < -127 \quad (25)$$

Log(x) is the logarithm of x (ln(x), $\log_2(x)$, $\log_{10}(x)$, for example), |x| is the absolute value of x, and ⌊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 25 is false, then

$$ExpVal = \left\lfloor \frac{\log|C_x|}{\log 2} \right\rfloor + 1$$
$$T^x\ Exponent = ExpVal$$
$$T^x\ Significand = \text{round}(iC_x \times 2^{15 - ExpVal})$$

where round*()* means round to the nearest integer.

Using C_1 as an example, convert Coefficient $C_1 = -2.8927765 \times 10^{-6}$ to its corresponding bit field values. First, check that C_1 is greater than the quantization limit.

$$C1\_ExpVal = \left\lfloor \frac{\log|C_x|}{\log 2} \right\rfloor + 1$$
$$= \left\lfloor \frac{\log|-2.8927765 \times 10^{-6}|}{\log 2} \right\rfloor + 1$$
$$= -18$$

This value is greater than the quantization limit of −127. Therefore,

$$T^1\ exponent = C1\_ExpVal = -18$$
$$= 0xEE \text{ (hexadecimal)}$$
$$T^1\ significand = \text{round}(C_1 \times 2^{15 - C1\_ExpVal})$$
$$= \text{round}(-2.8927765 \times 10^{-6} \times 2^{15 - (-18)})$$
$$= -24{,}849$$
$$= 0x9EEF \text{ (hexadecimal)}$$

Using C_2 as an example, convert Coefficient $C_2 = -1.9110192 \times 10^{-7}$ to its corresponding bit field values. First, check that C_2 is greater thna the quantization limit.

$$C2\_ExpVal = \left\lfloor \frac{\log|C_2|}{\log 2} \right\rfloor + 1$$
$$= \left\lfloor \frac{\log|-1.9110192 \times 10^{-7}|}{\log 2} \right\rfloor + 1$$
$$= -22$$

This vaue is greater than the quantization limit of −127. Therefore,

$$T^2\ Exponent = C2\_ExpVal = -22$$
$$= 0xEA \text{ (hexadecimal)}$$
$$T^2\ Significand = \text{round}(C_2 \times 2^{15 - C2\_ExpVal})$$
$$= \text{round}(-1.9110192 \times 10^{-7} \times 2^{15 - (-22)})$$
$$= -26{,}265$$
$$= 0x9967 \text{ (hexadecimal)}$$

Using C_3 as an example, convert Coefficient $C_3 = 2.2493907 \times 10^{-9}$ to its corresponding bit field values. First, check that C_3 is greater than the quantization limit.

$$C3\_ExpVal = \left\lfloor \frac{\log|C_3|}{\log 2} \right\rfloor + 1$$
$$= \left\lfloor \frac{\log|2.2493907 \times 10^{-9}|}{\log 2} \right\rfloor + 1$$
$$= -28$$

This value is greater than the quantization limit of −127.

Therefore,

$$T^3\ Exponent = C3\_ExpVal = -28$$
$$= 0xE4\ (hexadecimal)$$
$$T^3\ Significand = \text{round}(C_3 \times 2^{15 - C3\_ExpVal})$$
$$= \text{round}(2.2493907 \times 10^{-9} \times 2^{15 - (-28)})$$
$$= 19{,}786$$
$$= 0x4D4A\ (hexadecimal)$$

Using C_4 as an example, convert Coefficient $C_4 = 2.7943570 \times 10^{-11}$ to its corresponding bit field values. First, check that C_4 is greater than the quantization limit.

$$C4\_ExpVal = \left\lfloor \frac{\log|C_4}{\log 2} \right\rfloor + 1$$
$$= \left\lfloor \frac{\log|2.7943570 \times 10^{-11}|}{\log 2} \right\rfloor + 1$$
$$= -35$$

This is above the quantization limit of −127, therefore

$$T^4\ Exponent = C4\_ExpVal = -35$$
$$= 0xDD\ (hexadecimal)$$
$$T^4\ Significand = \text{round}(C_4 \times 2^{15 - C4\_ExpVal})$$
$$= \text{round}(2.7943570 \times 10^{-11} \times 2^{15 - (-35)})$$
$$= 31{,}462$$
$$= 0x7AE6\ (hexadecimal)$$

Using C_5 as an example, convert Coefficient $C_5 = -2.4817310 \times 10^{-13}$ to its corresponding bit field values. First, check that C_5 is above the quantization limit.

$$C5\_ExpVal = \left\lfloor \frac{\log|C_5}{\log 2} \right\rfloor + 1$$
$$= \left\lfloor \frac{\log|-2.4817310 \times 10^{-13}}{\log 2} \right\rfloor + 1$$
$$= -41$$

This value is greater than the quantization limit of −127. Therefore,

$$T^5\ Exponent = C5\_ExpVal = -41$$
$$= 0xD7\ (hexadecimal)$$
$$T^5\ Significand = \text{round}(C_5 \times 2^{15 - C5\_ExpVal})$$
$$= \text{round}(-2.4817310 \times 10^{-13} \times 2^{15 - (-41)})$$
$$= -17883$$
$$= 0xBA25\ (hexadecimal)$$

Use the following formulas to verify the value of the bit field computations closely matches the desired coefficient value;

$$C_D = Constant\ compensation\ value \times 2^{-45} \quad (26)$$

$$C_{x|x \neq D} = T^x\ significand \times 2^{Tx\ exponent - 15} \quad (27)$$

Note that there may be a slight difference between the values produced by Equation 26 and Equation 27 and the original coefficient values due to quantization effects.

Example (coefficient verification)

$$C_0 = 35{,}349{,}513{,}305 \times 2^{-45}$$
$$= 1.004694 \times 10^{-3}\ (verified)$$
$$C_1 = -24{,}849 \times 2^{-18-15}$$
$$= -2.892804 \times 10^{-6}\ (verified)$$
$$C_2 = -26{,}265 \times 2^{-22-15}$$
$$= -1.911030 \times 10^{-14}\ (verified)$$
$$C_3 = 19{,}786 \times 2^{-28-15}$$
$$= 2.249408 \times 10^{-9}\ (verified)$$
$$C_4 = 31{,}462 \times 2^{-35-15}$$
$$= 2.794387 \times 10^{-11}\ (verified)$$
$$C_5 = -17{,}883 \times 2^{-41-15}$$
$$= -2.481765 \times 10^{-13}\ (verified)$$

Digital Filter

The CF_1 values produced by the polynomial calculator block in Figure 99 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 the 3-bit unsigned coefficient output filter cutoff bit field in Bits[D2:D0] of Register 0x0288 according to Table 54. The filter comprises two colocated poles that yield a 3 dB bandwidth per Table 54.

Table 54. Digital Filter Bandwidth

Binary Value	3 db Bandwidth (Hz)
000	156
001	78
010	39
011	20
100	10
101	5
110	2
111	1

The device 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 occur 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 101, Compensation Method 2 makes use of one of the DPLLx channels (where x is 0 or 1). The user must specify which DPLL via the DPLL channel error source bit in Bit D0 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 98.

Compensation Method 2 generates the correction factor, CF_2, which the user can apply to other NCOs and TDCs within the AD9545. However, a compensation factor, $COMP_x$, can be applied to the NCO of DPLLx in Figure 101. 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 $COMP_x$.

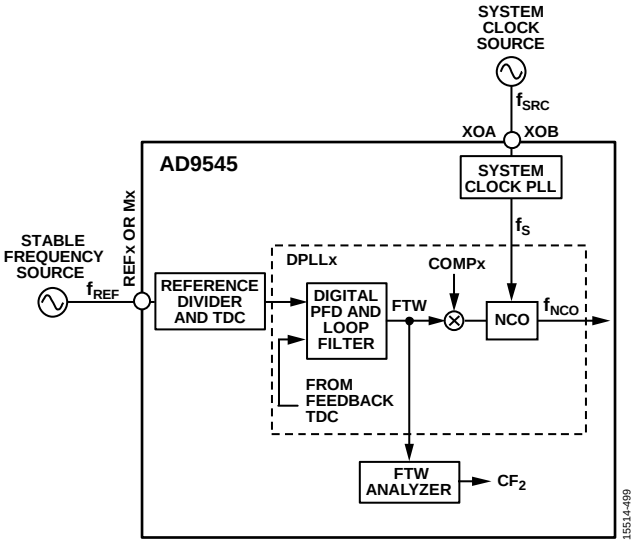


Figure 101. Compensation Method Number 2

COMPENSATION METHOD 3

Compensation Method 3 employs the closed-loop method of system clock compensation (see the Closed-Loop Method section). As shown in Figure 102, Compensation Method 3 produces compensation factor CF_3, where CF_3 embodies the factor, 1 + FFE_{COMP}, in Figure 98. Compensation Method 3 is similar to Compensation Method 2, but 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 Mx pins, the user must assign the TDC associated with the external clock source to the auxiliary DPLL via the 5-bit auxiliary DPLL source bit field in Bits[D4:D0] of Register 0x0284. Table 55 shows the assignment codes for the various TDCs.

The auxiliary DPLL can itself receive compensation from other sources via $COMP_x$ in Figure 102 (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.

Table 55. Auxiliary DPLL TDC Assignment Codes

Auxiliary DPLL Source Bit Field 5-Bit Code	TDC Description
00000	REFA TDC (default)
00001	REFAA TDC
00010	REFB TDC
00011	REFB TDC
00100	Unused
00101	Unused
00110	Auxiliary TDC 0
00111	Auxiliary TDC 1
01000 to 11111	Unused

Assuming the auxiliary DPLL is locked and has fully settled, the period of the stable reference must exactly match the period indicated by the contents of the appropriate REFx nominal period (see Register 0x0404 to Register 0x040B, Register 0x0424 to Register 0x042B, Register 0x0444 to Register 0x044B, Register 0x0464 to Register 0x046B) or Auxiliary x period bit fields (see Register 0x2A01 to Register 0x2A08 and Register 0x2A0A to Register 0x2A11). 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). Consider what happens 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 desired nominal frequency and the actual frequency. The desired nominal frequency derives from the entry of the user in the REFx nominal period or Auxiliary x period bit fields. The actual frequency is f_{REF}, the true frequency of the stable reference (see Figure 102). The accuracy error is the difference between the user entry and the actual nominal period of the stable reference.

Because CF_3 is a composite of both the stability and accuracy error sources, the reference period entry of the user via the REFx nominal period or Auxiliary x period bit fields directly effects 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).

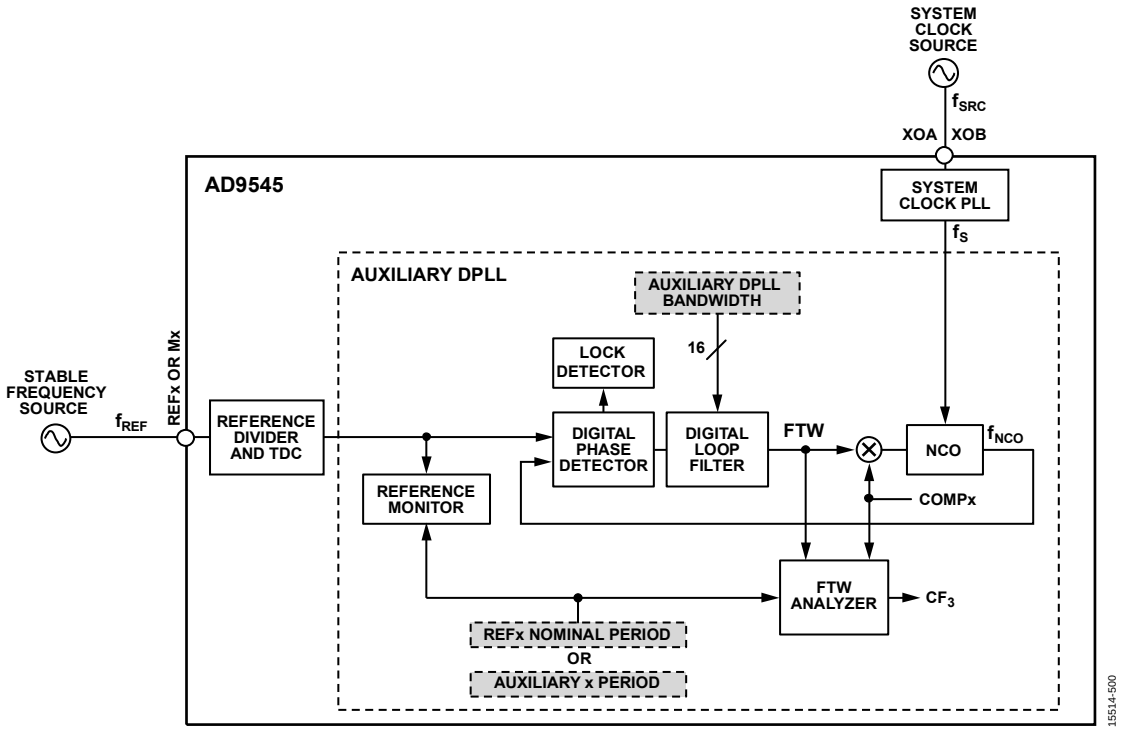


Figure 102. Compensation Method Number 3

Auxiliary DPLL Loop Bandwidth

The user controls the loop bandwidth of the auxiliary DPLL servo loop via the unsigned 16-bit auxiliary DPLL bandwidth bit field in Register 0x0285 to Register 0x0286. The loop bandwidth of the auxiliary DPLL is 0.1 Hz scaled by the value of the auxiliary DPLL bandwidth bit field.

Given a desired loop bandwidth of 247.63 Hz, determine the appropriate value of the auxiliary DPLL bandwidth bit field by the following:

Auxiliary DPLL Bandwidth = (247.63 Hz)/(0.1 Hz)

= 2476 (nearest integer)

= 0x09AC (hexadecimal)

The programmed bandwidth affects how long it takes the auxiliary DPLL to lock to the stable reference: the lower the bandwidth, the longer it takes the auxiliary DPLL to lock. The programmed bandwidth also affects the ability of the auxiliary DPLL to track variations of the system clock frequency. 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 Status

The auxiliary DPLL has a dedicated reference monitor to provide indication that the reference period is within limits. The reference monitor relies on user input of the expected reference period via the appropriate REFx nominal period or Auxiliary x period bit fields. The reference monitor status is available via the auxiliary DPLL unfaulted and auxiliary DPLL faulted bits in Bits[D3:D2] of Register 0x300C or via the auxiliary DPLL reference fault bit in Bit D2 of Register 0x3002.

The auxiliary DPLL has a builtin lock detector for indicating lock status. The lock/unlock thresholds for the lock detector are fixed and not user programmable. The lock detector status is available via the auxiliary DPLL unlocked and auxiliary DPLL locked bits in Bits[D1:D0] of Register 0x300C or via the auxiliary DPLL lock detect bit in Bit D1 of Register 0x3002.

Auxiliary DPLL Reference Monitor

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 period error exceeds ±1%

When the auxiliary DPLL is in the holdover state, it 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 AD9545. The compensation subsystem allows the user to employ any combination of the three compensation methods to any of the NCOs and TDCs within the AD9545 (barring a few exceptions per the Compensation Assignment Guidelines section). Figure 103 shows a simplified block diagram of the compensation subsystem.

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).

Each compensation method produces its own compensation factors (CF_1, CF_2, and CF_3). The combiner translates the three CFx factors to eight possible combined compensation factors, $COMP_x$, according to Table 56. The index, x, of $COMP_x$ is the 3-bit value programmed in the associated compensation destination bit field per Figure 104.

Table 56. 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 AD9545 implements the distributor through six 3-bit bit fields, where each bit field corresponds to a specific compensation destination per Figure 104. The user assigns a $COMP_X$ factor to a particular 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.

Note that 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, it 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 with compensation sources from exceeding a preset limit.

The user controls rate limiting via the 3-bit slew rate limit bit field in Bits[D2:D0] of Register 0x0283 according to Table 57. 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 based on the slew rate limit bit field value. 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.

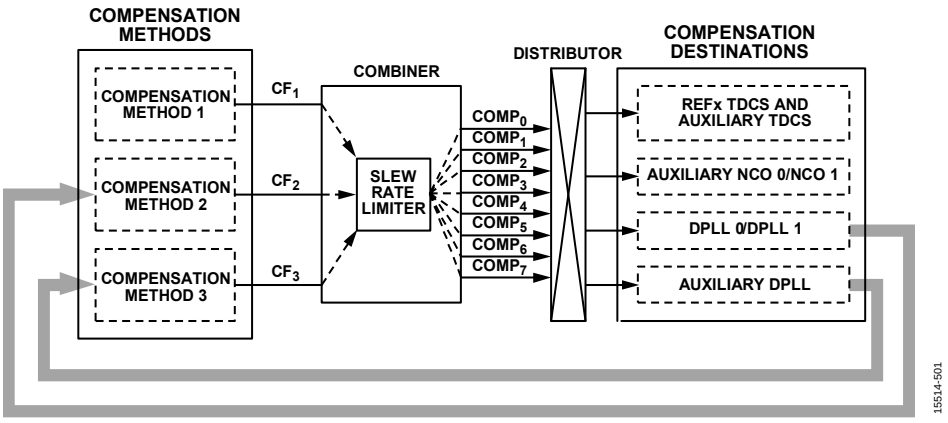


Figure 103. Integrated Compensation Subsystem

Table 57. Compensation Rate Limiting

Binary Value of the Slew Rate Limit Bit Field	Rate Limit
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 a user time stamp processor (see the User Access to Time Stamps section) is in use, the auxiliary NCO associated with that user time stamp processor as well as the time stamp source to that user time stamp processor must both use the same $COMP_x$ factor. That is, the user must program identical values in the bit field of the REFx/auxiliary TDC compensation destination associated with the user time stamp processor source and in the bit field of the auxiliary NCO 0 (or auxiliary NCO 1) compensation destination serving as the reference time source for the user time stamp processor (see Figure 104).

To avoid a positive feedback loop in the integrated compensation subsystem, do not assign $COMP_x$ sources involving CF_2 in Table 56 to the following compensation destinations:

- DPLL0 NCO compensation destination when the DPLL channel error source bit is Logic 0
- DPLL1 NCO compensation destination when the DPLL channel error source bit is Logic 1

Likewise, do not assign $COMP_x$ sources involving CF_3 in Table 56 to the auxiliary DPLL.

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 103. The three registers function as a connection matrix with six 3-bit fields distributed over the three registers (see Figure 104).

Each 3-bit field associates with one of the six compensation destinations:

- Auxiliary DPLL
- REFX TDCs and auxiliary TDCs
- Auxiliary NCO 0
- Auxiliary NCO 1
- DPLL 0
- DPLL 1

The value of the 3-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 104, 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 103). Certain compensation destinations have constraints on which compensation methods apply (see the Compensation Assignment Guidelines section).

To provide system clock compensation to Auxiliary NCO 0, use a combination of Compensation Method 1 and Compensation Method 3.

First, determine the appropriate compensation destination bit field per Figure 104. In this case, Bits[D2:D0] of Register 0x0281 applies because the compensation destination is auxiliary NCO 0. Next, determine the 3-bit code to write to Bits[D2:D0] of Register 0x0281 per the leftmost column of Table 56 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 56. In conclusion, by programming Bits[D2:D0] = 5 in Register 0x0281, auxiliary NCO 0 receives both Compensation Method 1 and Compensation Method 3.

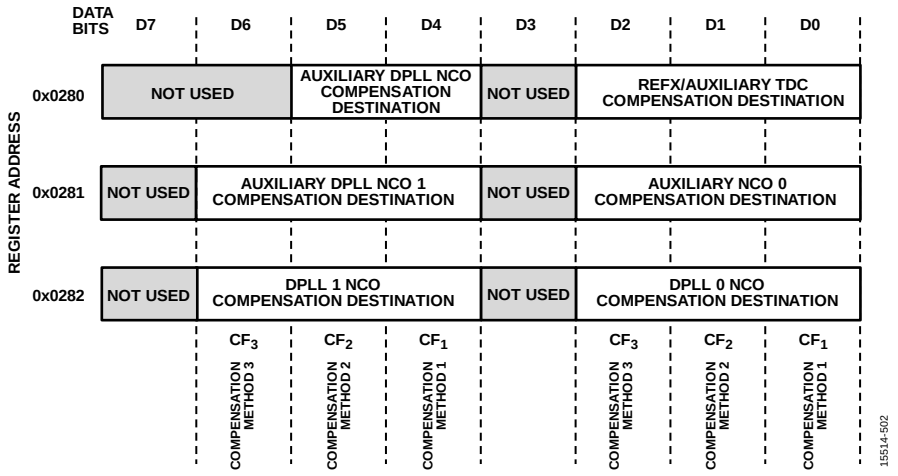


Figure 104. System Clock Compensation Registers

STATUS AND CONTROL PINS

STATUS AND CONTROL PINS OVERVIEW

The AD9545 features seven independently configurable digital CMOS status/control pins (M0 to M6). 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. Register 0x0102 to Register 0x0108 control both the nature of the pin (either status or control via Bit D7), as well as the selection of the status source or control destination associated with the pin via Bits[D6:D0]. During power-up or reset, the Mx pins temporarily become inputs and only allow the device to autoconfigure. Figure 105 is a block diagram of the Mx pin functionality.

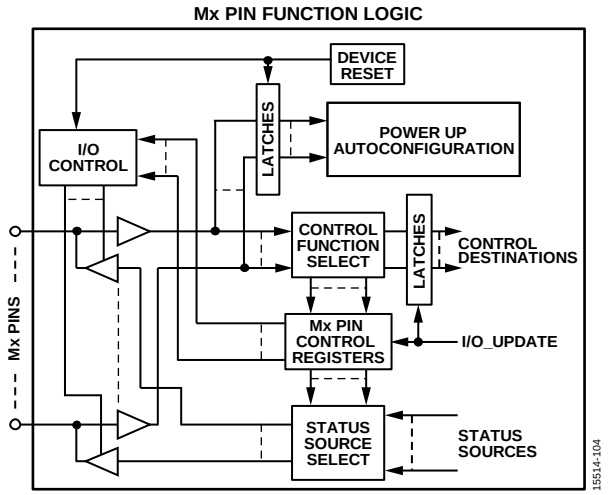


Figure 105. Mx Pin Logic

The Mx pin control logic uses special register write detection logic to prevent these 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 input/output update causes the corresponding pins to switch to the new function according to the newly programmed register contents. Note that 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 particular Mx pin function register causes all the Mx pin control functions to latch their values. Assertion of an input/output update switches to the newly programmed pin function, at which time normal behavior resumes. Note that, 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 the input/output 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 input/output update.

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 (NOOP) and assert the input/output update. This configuration avoids input/output contention on the Mx pin or other undesired behavior because, prior to the assertion of the input/output update, the device continues to drive the Mx pin. Following the assertion of the input/output update, the device releases the Mx pin but ignores the logic level on the pin due to the programmed NOOP function. The recommendation is to avoid using a high impedance source on an Mx pin configured as an input because this 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 NOOP to the desired function.

MULTIFUNCTION PINS AT RESET/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/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 58). Although the M0, M1, M2, M5, and 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 58 shows the Mx pin start-up conditions. M0, M1, and M2 do not appear in Table 58 because these pins have no explicit function during a power-up or reset operation.

Table 58. Mx Pin Function at Startup or Reset

Mx Pin	Startup/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)
M5	I^2C address offset	See Table 59	See Table 59
M6	I^2C address offset	See Table 59	See Table 59

When the start-up conditions select the serial port to be I²C mode (that is, M4 is Logic 1 at startup), the M5 and M6 pins determine the I²C port device address offset, per Table 59. The logic levels in Table 59 only apply during a power-up or reset operation.

Table 59. I²C Device Address Offset

M6	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.

STATUS FUNCTIONALITY

Configuring an Mx pin as a status pin gives the user access to specific internal device status/IRQ functions in the form of a hardware pin that produces a logic signal. Each Mx pin has a corresponding Mx function register. To assign an Mx pin as a status pin, write a Logic 1 to the Mx output enable bit in the corresponding Mx pin function register (Register 0x0102 to Register 0x0108).

To assign a specific status/IRQ function to an Mx pin configured as a status pin, program the appropriate 7-bit code to Bits[D6:D0] of the corresponding Mx function register. See the Interrupt Request (IRQ) section for details regarding IRQ functionality.

When configured as a status pin, the output mode of an Mx pin depends on a 2-bit mode code per Table 60. The 2-bit codes reside in Register 0x100 through Register 0x101, where the 2-bit codes constitute the Mx receiver/driver bit fields. The Mx receiver/driver bit fields perform a different function when the Mx pin is a control pin (see the Control Functionality section).

Table 60. Mx Receiver/Driver Bit Field Codes for Mx Status Pins

Code	Mode	Description
00	CMOS, active high	Output is Logic 0 when deasserted and Logic 1 when asserted (default operating mode).
01	CMOS, active low	Output is Logic 1 when deasserted and Logic 0 when asserted.
10	PMOS, open drain	Output is high impedance when deasserted and active high when asserted.
11	NMOS, open drain	Output is high impedance when deasserted and active low when asserted.

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 AD9545 devices or other compatible devices—for example, to implement an IRQ bus).

The drive strength of an Mx status pin is programmable via the corresponding Mx configuration bits (Bits[D6:D0] of the pin drive strength register). Logic 0 (default) selects normal drive strength (~6 mA) and Logic 1 selects weak drive strength (~3 mA).

CONTROL FUNCTIONALITY

Configuring an Mx pin as a control pin gives the user control of the specific internal device functions via an external hardware logic signal. Each Mx pin has a corresponding Mx function register. To assign an Mx pin as a control pin, write a Logic 0 to the Mx output enable bit in the corresponding Mx function register (Register 0x0102 to Register 0x0108).

To assign an Mx control pin to a specific function, program the appropriate 7-bit code to Bits[D6:D0] of the corresponding Mx function register. See the Interrupt Request (IRQ) section for details regarding IRQ functionality.

When configured as an Mx control pin, the logical level applied to the Mx pin translates to the selected device function. It is also possible to assign multiple Mx control pins to the same control function with the multiple pins implementing a Boolean expression. The Boolean operation associated with an Mx control pin depends on a 2-bit code per Table 61. The 2-bit codes reside in Register 0x100 through Register 0x101, where the 2-bit codes constitute the Mx receiver/driver bit fields. The Mx receiver/driver bit fields perform a different function when the Mx pin is a status pin (see the Status Functionality section).

Table 61. Mx Receiver/Driver Bit Field Codes for Mx Control Pins

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 M0, M2, M3, and M6 are configured as control pins and are assigned to the input/output update control function; that is, Bits[D6:D0] = 0x01 in Register 0x0102, Register 0x0104, Register 0x0105, and Register 0x0108.

In addition, M0 is assigned for AND operation, M2 for NOT OR operation, M3 for NOT AND operation, and M6 for OR operation (that is, the 2-bit codes in Register 0x100 and Register 0x101 according to Table 61).

With these settings, the input/output (I/O) update function behaves according to the following Boolean equation:

I/O update = ((NOT *M2*) OR *M6*) AND *M0* AND (NOT *M3*)

In this case, an I/O 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 61 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 AD9545 monitors certain internal device events, potentially allowing them to trigger an IRQ event. Three groups of registers (see Figure 106) control the IRQ functionality within the AD9545:

- IRQ status registers (Register 0x300B through Register 0x3019)
- IRQ mask registers (Register 0x010C through Register 0x011A)
- IRQ clear registers (Register 0x2006 through Register 0x2014)

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 monitor bits. In addition, the IRQ logic offers IRQ event status results for particular groups of specific IRQ events, namely, the PLL0 IRQs, PLL1 IRQs, and common IRQs (see Figure 106).

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 associated with the system clock, the watchdog timer, and the EEPROM.

IRQ MONITOR

The IRQ monitor registers (in the general status section of the register map) 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 monitor. The output of the IRQ monitor provides the mechanism for generating IRQ event status results (see the PLL0 IRQ, PLL1 IRQ, common IRQ, or any IRQ signal shown in Figure 106).

IRQ MASK

The IRQ mask registers (in the Mx pin status and control section of the register map) comprise a bit for bit correspondence with the specific IRQ event bits within the IRQ monitor. Writing a Logic 1 to a mask bit enables (unmasks) the corresponding specific device event to the IRQ monitor. A Logic 0 (default) disables (masks) the corresponding specific device event to the IRQ monitor. Therefore, a specific IRQ event is the result of a logical AND of a specific device event and its associated IRQ, mask bit.

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 event status result (PLL0 IRQ, PLL1 IRQ, common IRQ, or any IRQ signal in Figure 106).

The default state of the IRQ mask register bits is Logic 0; therefore, the device is not capable of generating an IRQ event status result until the user populates the IRQ mask with a Logic 1 to unmask the desired specific IRQ events. Writing a Logic 1 to an IRQ mask bit may result in immediate indication of an IRQ status event result if the corresponding specific device event is already asserted (that is, the device previously registered the corresponding device event).

IRQ CLEAR

The IRQ clear registers (in the operational controls section of the register map) comprise a bit for bit correspondence with the IRQ monitor. Writing a Logic 1 to an IRQ clear bit forces the corresponding IRQ monitor bit to Logic 0, thereby clearing that specific IRQ event. Note that the IRQ clear registers are autoclearing; therefore, after writing a Logic 1 to any IRQ clear bit in Register 0x2006 to Register 0x2014, the device automatically restores the IRQ clear bit to Logic 0. The IRQ event status results remain asserted until the user clears all of the bits in the IRQ monitor responsible for the IRQ status result (that is, the entire group of status bits associated with PLL0 IRQ, PLL1 IRQ, common IRQ, or any IRQ signal shown in Figure 106).

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 D0 clears all IRQ monitor bits. Bit D1 clears the common IRQ bits. Bit D2 clears the PLL0 IRQ bits. Bit D3 clears 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 106).

The recommendation for clearing IRQ status events is to first service the specific IRQ event (as needed) and then clear the specific IRQ for that particular 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 monitor bits. Clearing asserted IRQ monitor bits eliminates the record of the associated device events, subsequently erasing any history of those events having occurred.

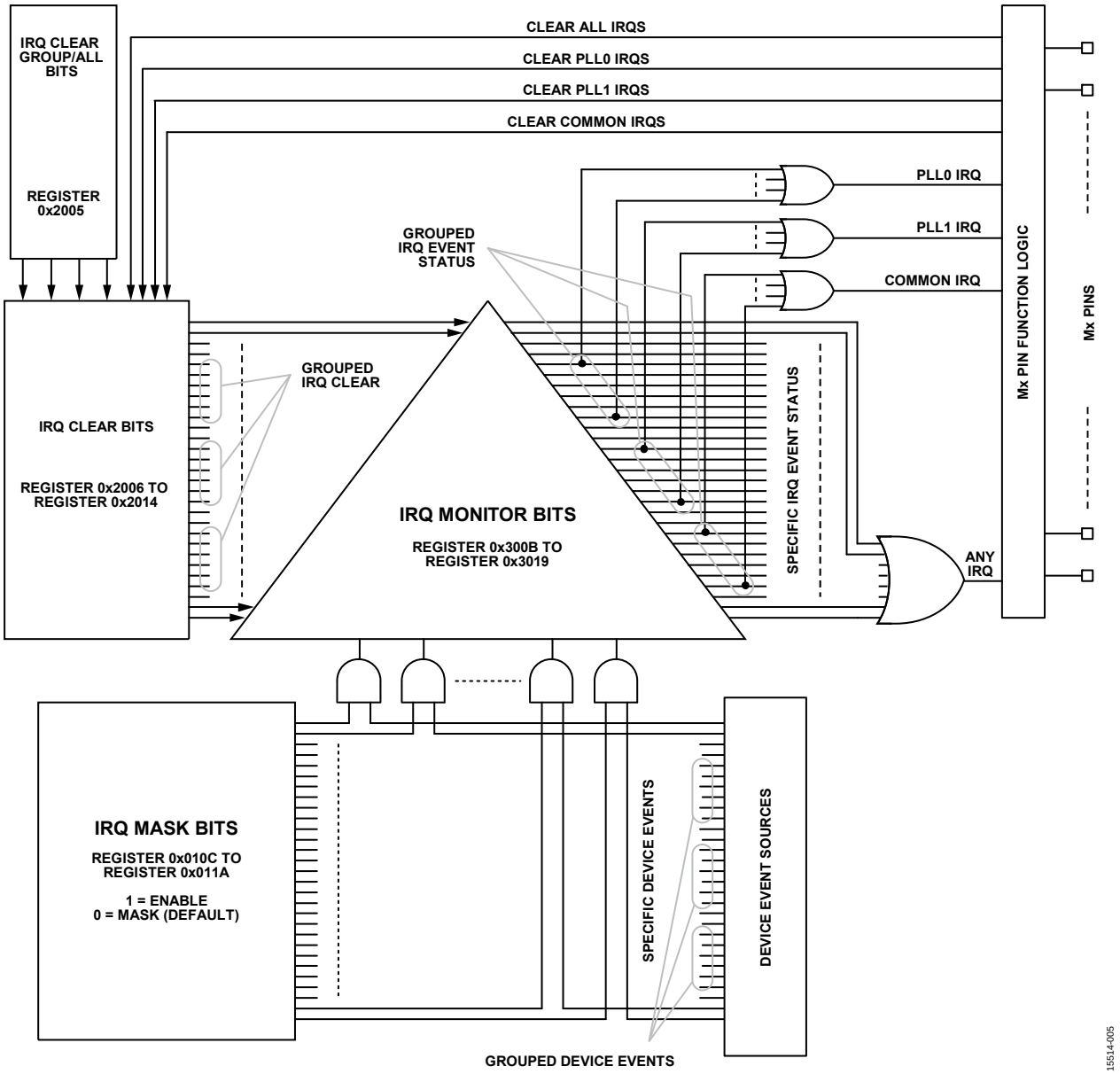


Figure 106. IRQ System Diagram Register 0x2005

WATCHDOG TIMER

The watchdog timer is a general-purpose programmable timer capable of triggering a specific IRQ event (see Figure 107). 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 by programming the watchdog timer (ms) bit field with a 16-bit timeout value. A nonzero value sets the timeout period in units of milliseconds, providing a range of 1 ms to 65.535 sec, whereas a zero value (0x0000, the default value) disables the timer. The relative accuracy of the timer is approximately 0.1% with an uncertainty of 0.5 ms. Note that whenever the user writes to the watchdog timer (ms) bit field, it automatically clears the timer, ensuring a correct timeout period (per the new value) starting from the moment of the bit field update.

The watchdog timer (ms) bit field field relates to the timeout period as follows:

$$Watchdog\ Timer\ (ms) = Timeout\ Period \times 10^3$$

To determine the value of the watchdog timer (ms) bit field bit field necessary for a timeout period of 10 sec,

$$Watchdog\ Timer\ (ms) = Timeout\ Period \times 10^3$$
$$= 10 \times 10^3$$
$$= 10{,}000$$
$$= 0x2710\ (hexadecimal)$$

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 D2 of Register 0x300B.

Because this bit is a latched IRQ bit, the user must clear it via Bit D2 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 the clear watchdog timer bit (an autoclearing bit) in Bit D7 of 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 it 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. In this way, as long as the external device keeps resetting the watchdog timer before it 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 its timeout period expires, the watchdog timer eventually times out, resetting the external device via the appropriately configured Mx status pin.

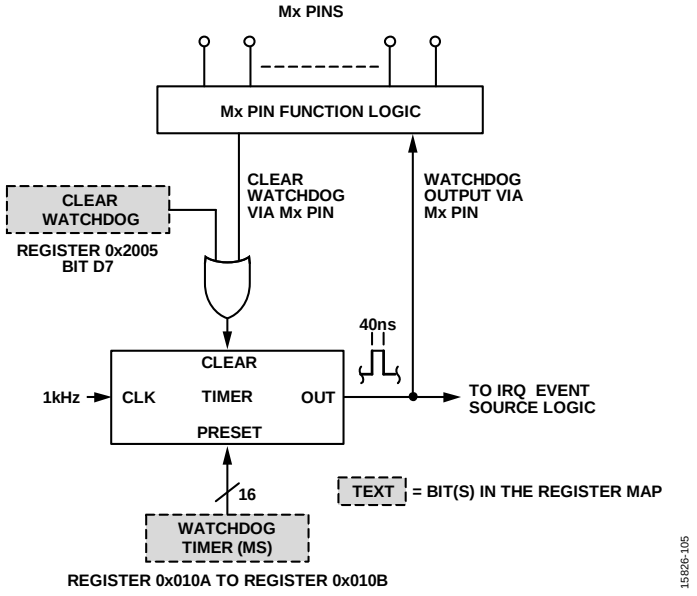


Figure 107. Watchdog Timer

EEPROM USAGE

EEPROM OVERVIEW

The AD9545 supports an external, I²C-compatible, EEPROM with dedicated access. With some restrictions, the AD9545 also supports multidevice access to a single external EEPROM on a shared I²C common bus. The AD9545 has an on-chip I²C master to interface to the EEPROM through the Mx pins.

Because the default register settings of the AD9545 do not define a particular 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. When the oscillator starts, it 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 AD9545 programming registers and invokes specific actions per the instructions stored in the EEPROM (see Table 62). Automatic downloading is the most common method for initiating an EEPROM download sequence, which initiates at power-up of the AD9545, provided Pin M3 is Logic 1 at power-up (see the Multifunction Pins at Reset/Power-Up section). Alternatively, instead of cycling power to the AD9545 to initiate an EEPROM download, the user can force the RESETB pin to Logic 0, force Pin M3 to Logic 1, and 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 AD9545) by writing a Logic 1 to the EEPROM load bit in the EEPROM section of the register map.

The load from EEPROM bit does not require an input/output update. Writing a Logic 1 to this bit immediately triggers a download sequence.

The EEPROM controller sets the EEPROM load in progress bit (in the general status section of the register map) 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 AD9545 register contents in the EEPROM, the user must write a Logic 1 to the EEPROM save bit in the EEPROM section of the register map. The EEPROM save bit does not require an input/output update. Writing a Logic 1 to this bit immediately triggers an upload sequence.

The AD9545 has the equivalent of a write protect feature in that the user must write a Logic 1 to the EEPROM write enable bit (in the EEPROM section of the register map) prior to requesting an upload to the EEPROM. Attempting to upload to the EEPROM without first setting the EEPROM write enable bit results in a fault indication (that is, the AD9545 asserts the EEPROM fault bit in the general status section of the register map).

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. 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 AD9545 register data).

The EEPROM controller sets the EEPROM save in progress bit (in the status readback section of the register map) 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, it typically cannot hold enough instructions to upload a complete set of AD9545 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 in the EEPROM section of the register map, with the 15th instruction being a pause instruction (see Table 62).
2. Initiate an EEPROM upload by writing Logic 1 to the EEPROM save bit. When the EEPROM controller reaches the pause instruction, it suspends the upload process and waits for another assertion of the EEPROM save bit.
3. While the controller pauses, write the remaining six bytes of the upload sequence into the EEPROM sequence registers in the EEPROM section of the register map, followed by an end of data instruction (see Table 62).
4. Initiate an EEPROM upload by writing Logic 1 to the EEPROM save bit. When the EEPROM controller reaches the end of data instruction, it terminates the upload process.

The previous 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 62) 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 the EEPROM CRC error bit in the status readback section of the register map.

To minimize the possibility of downloading a corrupted EEPROM data set, the user can execute a checksum test by asserting the verify EEPROM CRC bit in the EEPROM section of the register map, which causes the EEPROM controller to execute a download sequence, but without actually transferring data to the AD9545 registers. The controller still computes an on the fly checksum, performs the checksum comparison, and asserts the EEPROM CRC error bit if the checksums do not match. Therefore, after the device deasserts the EEPROM load in progress bit, the user can check the EEPROM CRC fault bit to determine if the test passed (that is, EEPROM CRC error = 0). However, even if the test fails, device operation is unaffected because there was no transfer of data to the AD9545 registers.

EEPROM Header

The EEPROM controller adds a header to stored data that carries information related to the AD9545, 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 AD9545. If the controller detects a mismatch, it asserts the EEPROM fault bit in the status readback section of the register map 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. A summary of the EEPROM controller instructions appears in Table 62.

When the controller downloads the EEPROM contents to the AD9545 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, as well as computed data that the controller calculates on the fly.

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 AD9545 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.

Note that 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/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 to and from the target register (that is, one more than the value of the instruction).

Input/Output Update Instruction (0x80)

When the EEPROM controller encounters an input/output update instruction during an upload sequence, it stores the instruction in EEPROM. When encountered during a download sequence, however, the EEPROM controller initiates an input/output update event (equivalent to the user asserting the IO_UPDATE bit in the serial port section of the register map).

Device Action Instructions (0x90 to 0xAF)

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 62.

Table 62. EEPROM Controller Instruction Set Summary

Instruction Code (Hexadecimal)	Response	Comments
0x00 to 0x7F	Register transfer	Requires a 2-byte register address suffix
0x80	Input/output update	Assert input/output update during download
0x81 to 0x8F	Not applicable	Undefined
0x90	Calibrate APLLs	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	Calibrate only APLL0 during download
0x93	Calibrate APLL1	Calibrate only APLL1 during download
0x94 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

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 the EEPROM controller encounters a conditional instruction during an upload sequence, it stores the instruction in the EEPROM. When the EEPROM controller detects a conditional instruction during a download sequence, it affects the condition map as well as the outcome of 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, while 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 the 4-bit EEPROM load condition bit field in the EEPROM section of the register map. 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. 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.

Note that 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. An example of the use of conditional processing is shown in Table 63.

Table 63. 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	Input/ouput 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 AD9545 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 in the EEPROM section of the register map (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 AD9545 devices to share the contents of a single EEPROM. There are two levels of multidevice support. Level 1 supports a configuration where multiple AD9545 devices share a single EEPROM through a dedicated I²C bus. Level 2 supports a configuration where multiple AD9545 devices share a single EEPROM connected to a common I²C bus that includes other I²C master devices. Figure 108 and Figure 109 show the Level 1 and Level 2 configurations, respectively.

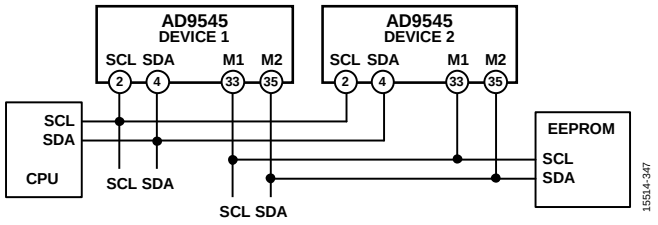


Figure 108. Level 1 Multidevice Configuration

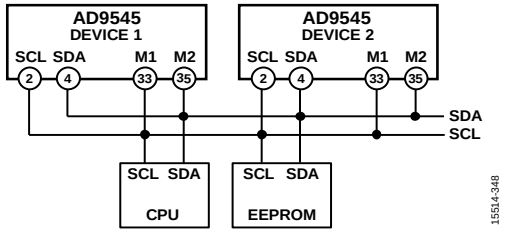


Figure 109. Level 2 Multidevice Configuration

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, it 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, it 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, it starts the arbitration timer. 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 AD9545 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 64.

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 the EEPROM load condition bit field of Device 1 with a condition ID of 1 and Device 2 with a condition ID of 2.

Table 64. 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	Input/output update
0xFF	End of sequence

APPLICATIONS INFORMATION

OPTICAL NETWORKING LINE CARD

One example of an optical networking application appears in Figure 110, which shows a line card and optical module using direct timing. The AD9545 accepts redundant input clocks from two remote timing cards. The AD9545 has all of the necessary functionality to provide automatic switching between the two input clocks in the event either one of them fails. By employing PLL0 and PLL1 with both using the same input clock, the PLLs generate the necessary clock frequencies for the transmit (Tx) and receive (Rx) paths.

Another optical networking example (see Figure 111) operates in loop timing mode typical of a WAN application. Similar to the direct timing mode, PLL0 uses one of the redundant input timing signals to generate clocks for the Tx path with the appropriate frequency translation ratio. With loop timing, however, the other PLL (PLL1) uses the recovered clock from the CDR as its input, suppresses the jitter commonly present in the Rx path, regenerates output clocks for the Rx path, and provides a reference clock to the central timing card via the backplane.

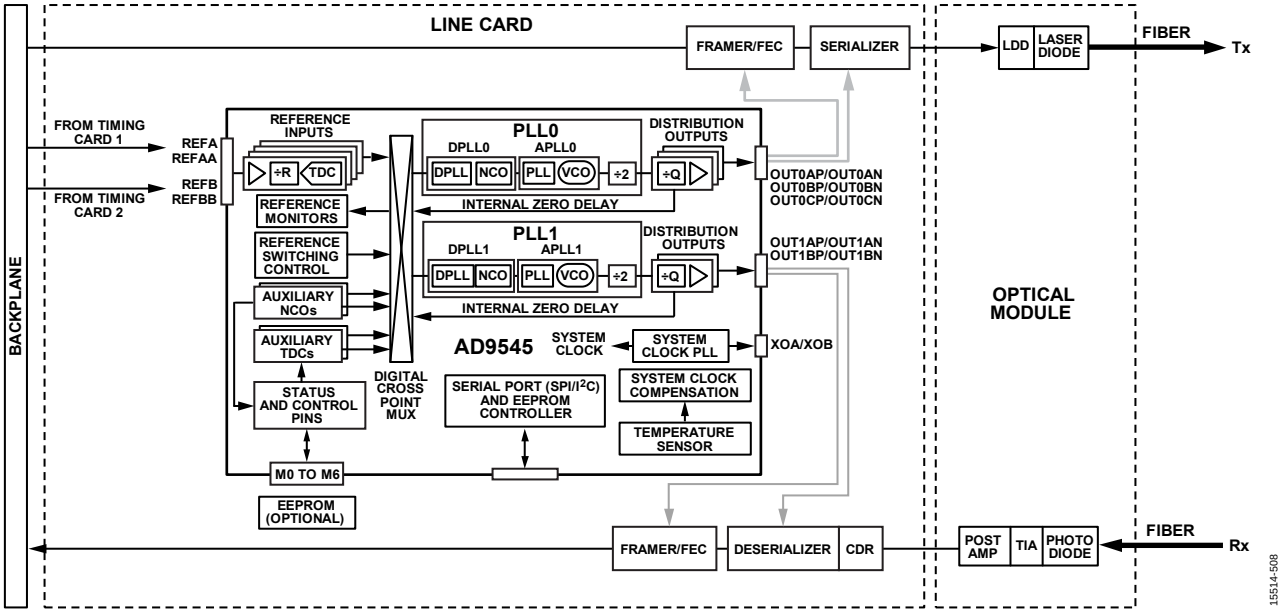


Figure 110. Optical Line Card Direct Timing Example

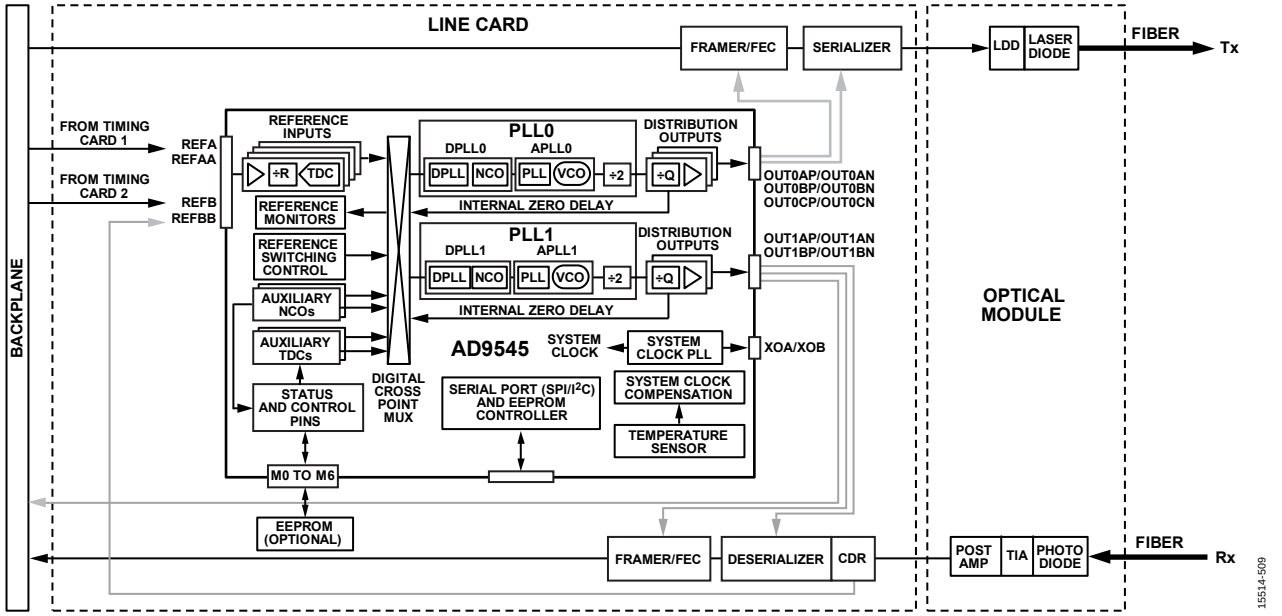


Figure 111. Optical Line Card Loop Timing Example

The AD9545 can also perform specialized frequency translation tasks to accommodate FEC and/or jitter suppression associated with a gapped clock scenario, as shown in Figure 110 and Figure 111.

The flexibility and available feature set of the AD9545 enables it to meet the continually evolving nature of optical line card protocols and functions. However, the examples given Figure 110 and Figure 111 are by no means exhaustive.

SMALL CELL BASE STATION

The diagram shown in Figure 112 is an example of the AD9545 providing all of the synchronization necessary for the baseband unit of a small cell base station. The built-in JESD204B support enables a particularly compact and efficient design.

The AD9545 can lock to any of the following input sources:

- GPS
- SyncE
- IEEE1588 (requires a separate IEEE1588 servo and software stack—see Figure 113 for an example of using the AD9545 as part of an IEEE1588 servo loop)
- Loop timed input (in the case of SONET/SDH backhaul)

In this small cell application, PLL0 provides up to four device system reference (SYSREF) clocks for clocking wireless transceivers (such as the AD9371), a clock signal for distribution through a 1:4 buffer for clocking up to two radio cards and an optional CPU interface clock for the baseband processor. Independently, PLL1 provides a clock for the backhaul interface via the network connection. PLL0 and PLL1 derive their input timing from the appropriate input sources (GPS, IEEE1588, loop timing from the network, or SyncE).

Because the AD9545 supports an optional EEPROM, the user can store the application configuration for automatic download at power-up.

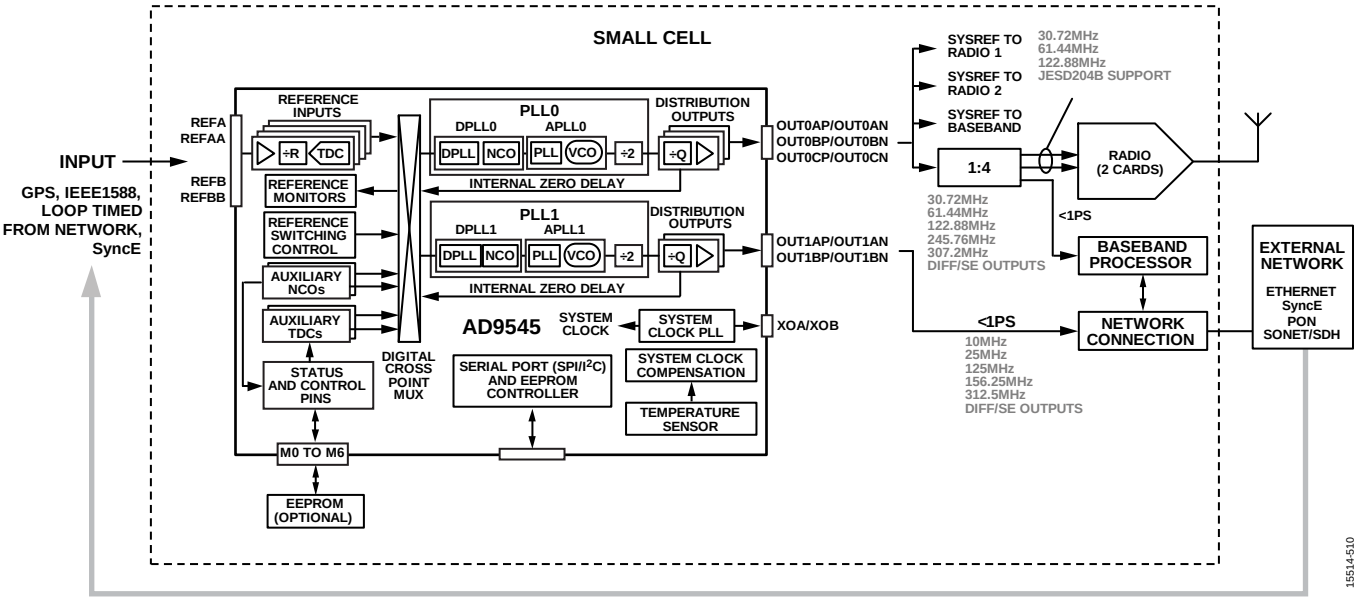


Figure 112. AD9545 Small Cell Application

IEEE 1588 SERVO

The AD9545 provides all of the jitter cleaning, phase shifting, and synchronization for an IEEE 1588 servo and software stack in a Telecom-BC (T-BC) with hybrid support application (see Figure 113). The diagram in Figure 113 supports IEEE 1588 master and slave clock operation.

PLL0 provides two filtered, 125 MHz SyncE clocks. PLL1 provides for two output clocks, each operating from 1 pps up to 500 MHz with high resolution output phase adjustment and slew rate limiting capability, which allows filtering a GPS signal, for example, and provides PTP asymmetry correction (with appropriate PTP software support).

Figure 113 also shows the ability of the AD9545 to compensate for system clock errors by taking advantage of a frequency stable source applied at the reference input (REF IN) input and using the auxiliary NCOs as references to PLL0 and PLL1. This capability improves the overall timing precision of the system.

The performance and feature set of the AD9545 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.

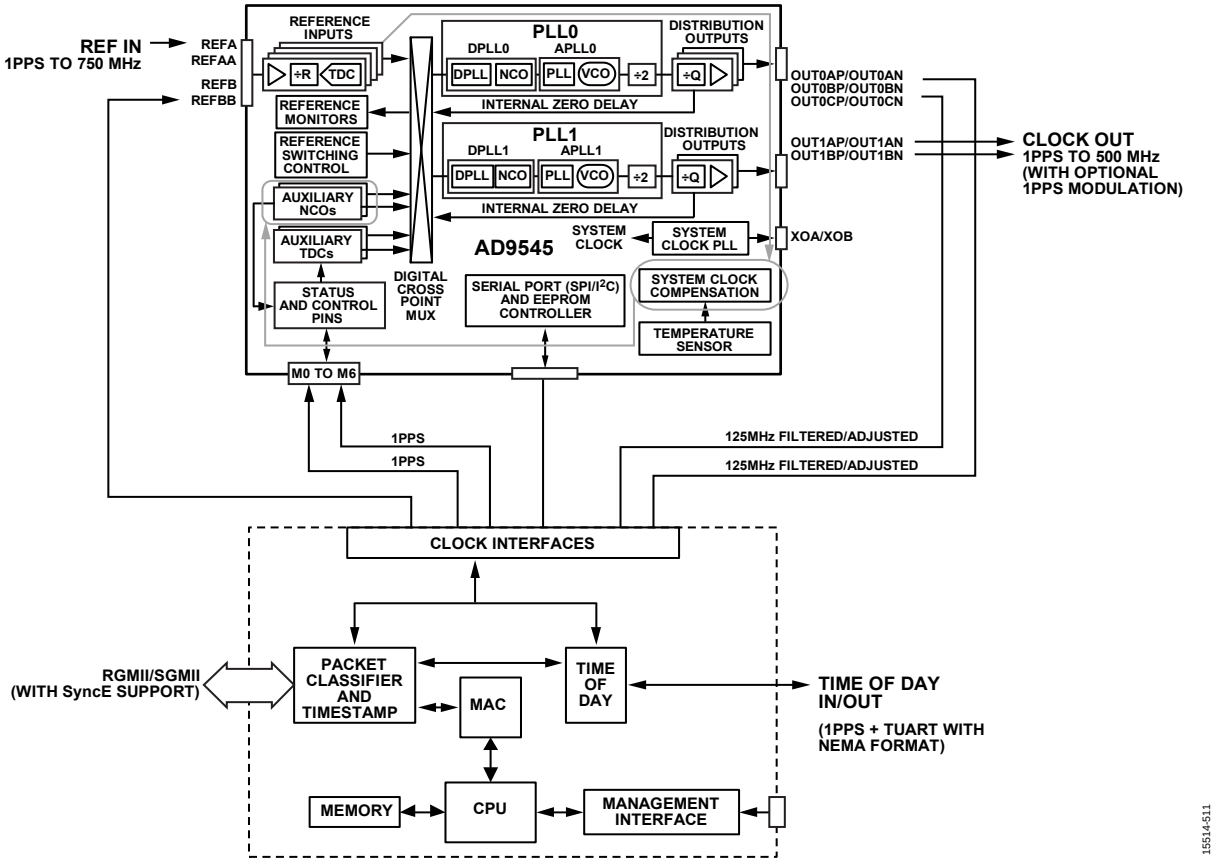


Figure 113. Using an AD9545 in an IEEE1588 Version 2 Application

INITIALIZATION SEQUENCE

Figure 114 through Figure 116 describe the sequence for powering on and programming the AD9545.

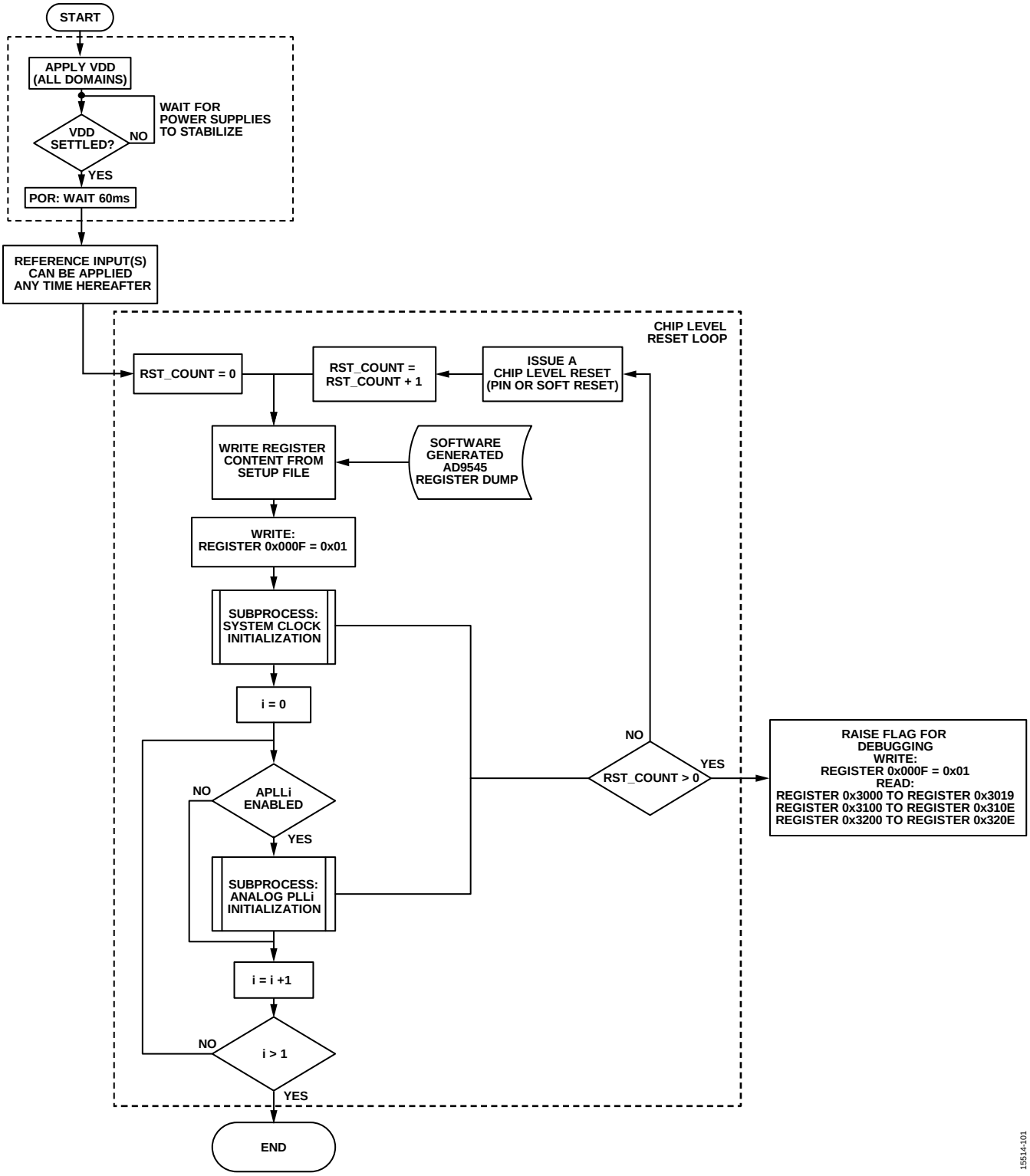


Figure 114. Programming Sequence Loop

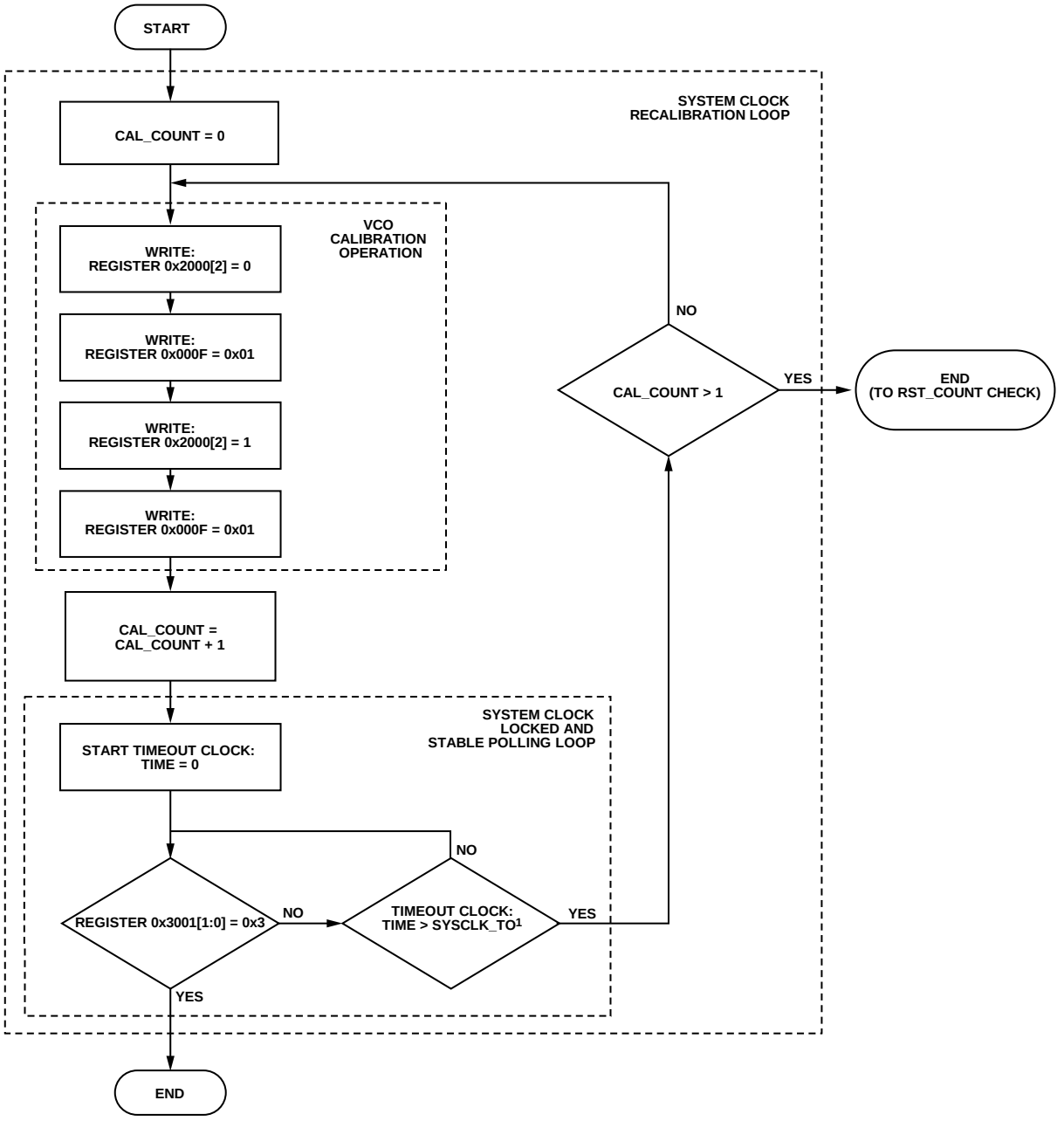
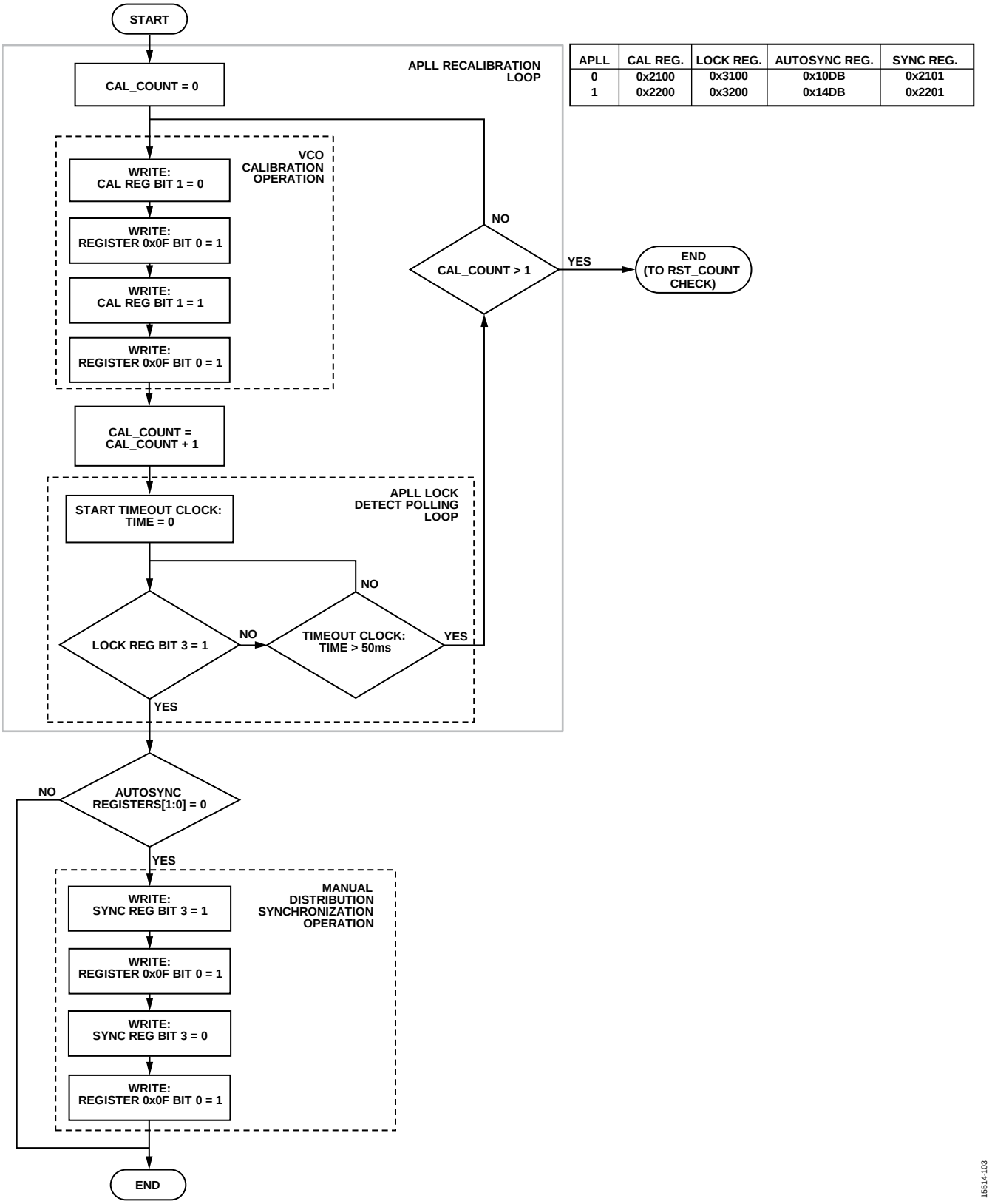


Figure 115. AD9545 System Clock Initialization Subprocess



APLL	CAL REG.	LOCK REG.	AUTOSYNC REG.	SYNC REG.
0	0x2100	0x3100	0x10DB	0x2101
1	0x2200	0x3200	0x14DB	0x2201

Figure 116. AD9545 Analog PLL Initialization Subprocess

SERIAL CONTROL PORT

SERIAL CONTROL PORT OVERVIEW

The AD9545 serial control port is a flexible, synchronous serial communications port that provides a convenient interface to many industry-standard microcontrollers and microprocessors. The AD9545 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 AD9545 register map.

The AD9545 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 0x00.

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 pin goes high. The W1 and W0 bits in the traditional SPI become the A12 and A13 bits of the register address. This format is similar to streaming mode in the traditional SPI.
- The address ascension bit (Register 0x00) controls whether register addresses are automatically incremented or decremented regardless of the LSB/MSB first setting. In 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 AD9545 supports both the SPI and I²C serial port protocols, only one is active following power-up (as determined by the M4 multifunction 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 59 for the I²C address assignments.

SPI SERIAL PORT OPERATION

Pin Descriptions

The serial clock (SCLK) pin 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 pin supports a maximum clock rate of 50 MHz.

The SPI port supports both 3-wire (bidirectional) and 4-wire (unidirectional) hardware configurations and both MSB first and LSB first data formats. Both the hardware configuration and data format features are programmable. The 3-wire mode uses the serial data input/output (SDIO) pin for transferring data in both directions. The 4-wire mode uses the SDIO pin for transferring data to the AD9545, and the SDO pin for transferring data from the AD9545.

The chip select (CSB) pin is an active low control that gates read and write operations. Assertion (active low) of the CSB pin initiates a write or read operation to the AD9545 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 0x00). The user must deassert the CSB pin following the last byte transferred, thereby ending the stream mode. This pin is internally connected to a 10 kΩ pull-up resistor. When CSB is high, the SDIO and SDO pins 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: 0x012
- 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 AD9545 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.

Write

When the instruction word indicates a write operation, the payload is written into the serial control port buffer of the AD9545. 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. Note that 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.

Most of the serial port registers are buffered; therefore, data written into buffered registers does not take effect immediately.

To transfer buffered serial control port contents to the registers that actually control the device requires an additional operation, an IO_UPDATE operation, implemented in one of two ways. One is to write a Logic 1 to Bit D0 of Register 0x000F (this bit is an autoclearing bit). The other is to use an external signal via an appropriately programmed multifunction pin. The user can change as many register bits as desired before executing an input/output update. The input/output update operation transfers the buffer register contents to their active register counterparts.

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. Note that 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 0x01, Bit D5.

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 66). The SPI controller ignores A14, treating it as Logic 0, because the AD9545 has no register addresses requiring more than a 14-bit address word.

SPI MSB-/LSB-First Transfers

The AD9545 instruction word and payload can be transferred MSB first or LSB first. The default for the AD9545 is MSB first. To invoke LSB first mode, write a Logic 1 to Register 0x00, Bit D6. Immediately after invoking LSB first mode, subsequent serial control port operations are LSB first.

Address Ascension

If the address ascension bit (Register 0x0000, Bit D5) 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 D5) 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. Note that 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 65. Streaming Mode (No Addresses Skipped)

Address Ascension	Stop Sequence
Increment	0x0000 … 0x3A3B
Decrement	0x3A3B … 0x0000

Table 66. 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

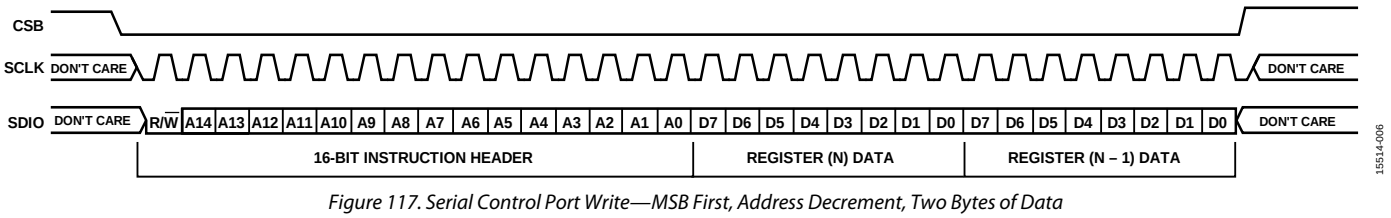


Figure 117. Serial Control Port Write—MSB First, Address Decrement, Two Bytes of Data

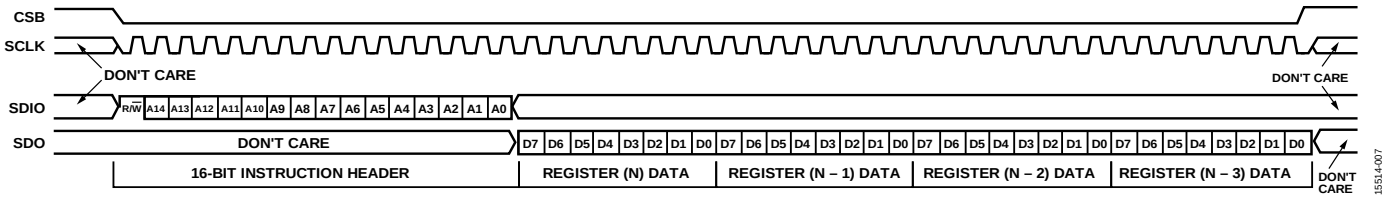


Figure 118. Serial Control Port Read—MSB First, Address Decrement, Four Bytes of Data

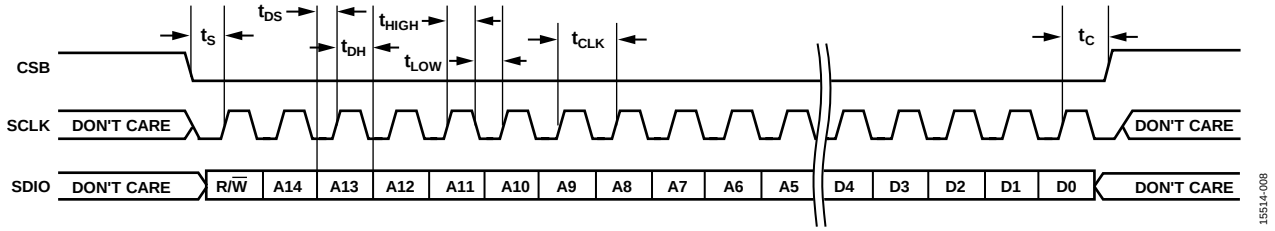


Figure 119. Timing Diagram for Serial Control Port Write—MSB First

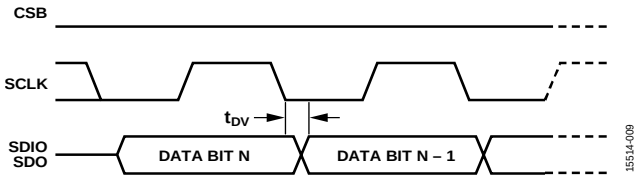


Figure 120. Timing Diagram for Serial Control Port Register Read—MSB First

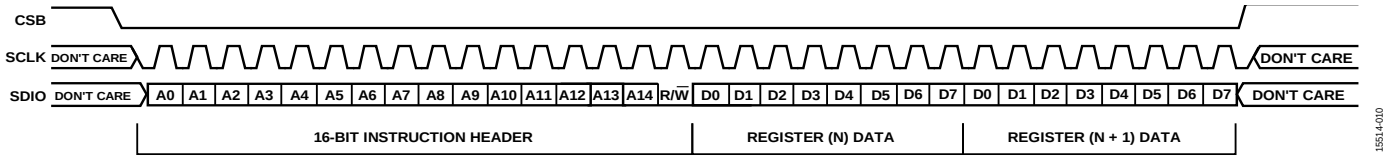


Figure 121. Serial Control Port Write—LSB First, Address Increment, Two Bytes of Data

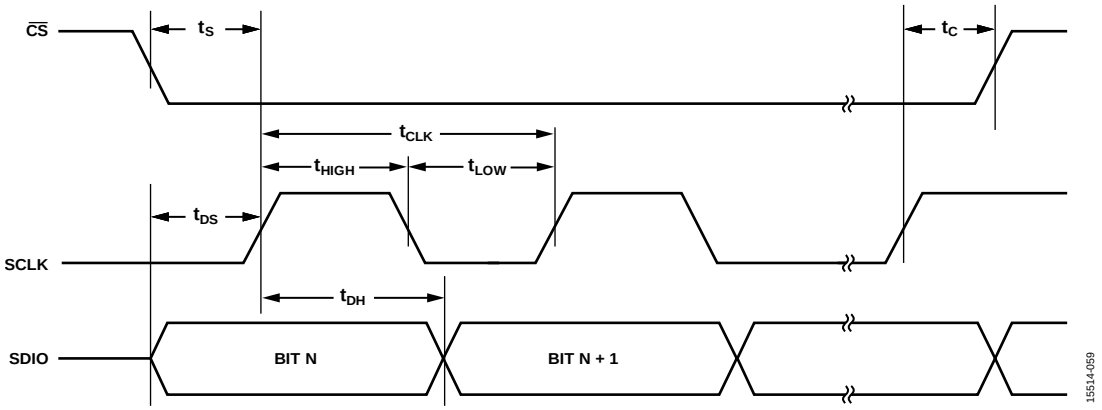


Figure 122. Serial Control Port Timing—Write

Table 67. 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 120)

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 AD9545 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 AD9545 does not strictly adhere to every requirement in the original I²C specification. In particular, it does not support specifications such as slew rate limiting and glitch filtering. Therefore, the AD9545 is I²C compatible, but not necessarily fully I²C compliant.

The AD9545 I²C port consists of a serial data line (SDA) and a serial clock line (SCL). In an I²C bus system, the AD9545 connects to the serial bus (data bus SDA and clock bus SCL) as a slave device; that is, the AD9545 does not generate an I²C clock. The AD9545 uses direct 16-bit memory addressing rather than 8-bit memory addressing, which is more common.

The AD9545 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 59 lists the supported device slave addresses.

I²C Bus Characteristics

Table 68 shows a summary of the various I²C abbreviations.

Table 68. I²C Bus Abbreviation Definitions

Abbreviation	Definition
S	Start
Sr	Repeated start
P	Stop
A	Acknowledge
$\overline{A}$	Nonacknowledge
$\overline{W}$	Write
R	Read

Figure 123 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.

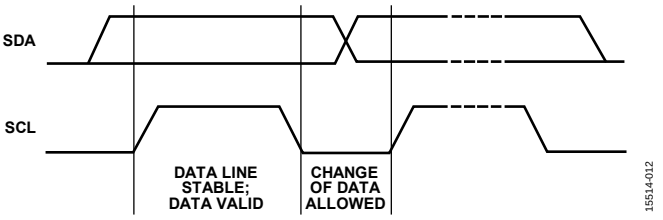


Figure 123. Valid Bit Transfer

Figure 124 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 AD9545 I²C state machine goes into an idle state.

Data Transfer Process

The master initiates 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 a 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 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.

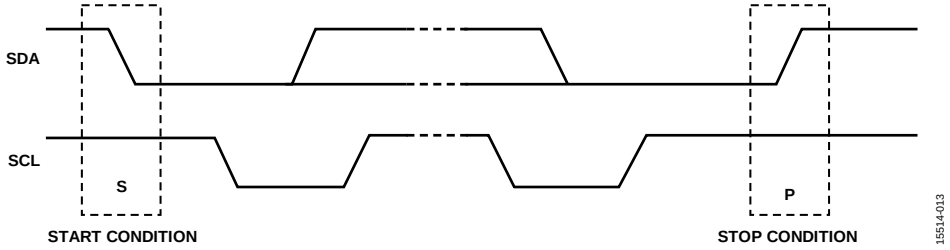


Figure 124. Start and Stop Conditions

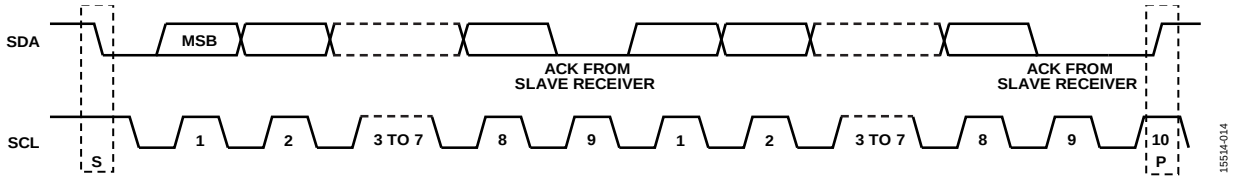


Figure 125. Acknowledge Bit

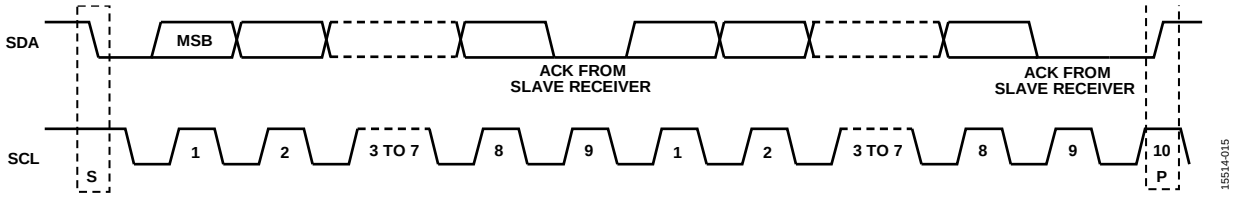


Figure 126. Data Transfer Process (Master Write Mode, 2-Byte Transfer)

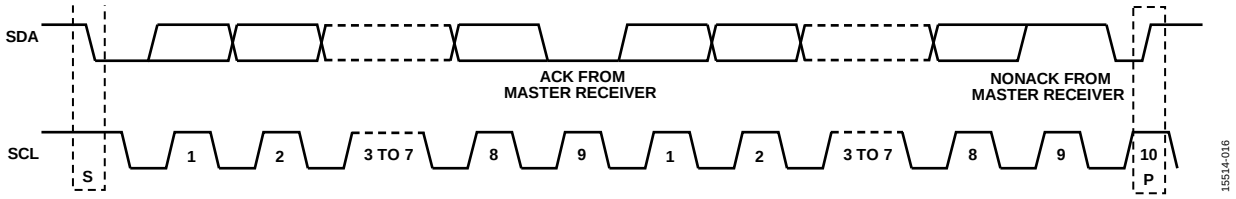


Figure 127. Data Transfer Process (Master Read Mode, 2-Byte Transfer), First Acknowledge From Slave

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 69). The send byte format is used to set up the register address for subsequent reads (see Table 70). The receive byte format is used to read the data byte(s) from RAM starting from the current address (see Table 71). The read byte format is the combined format of the send byte and the receive byte (see Table 72).

Table 69. 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 70. Send Byte Format

S	Slave address	$\overline{W}$	A	RAM address high byte	A	RAM address low byte	A	P

Table 71. Receive Byte Format

S	Slave address	R	A	RAM Data 0	A	RAM Data 1	A	RAM Data 2	$\overline{A}$	P

Table 72. 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

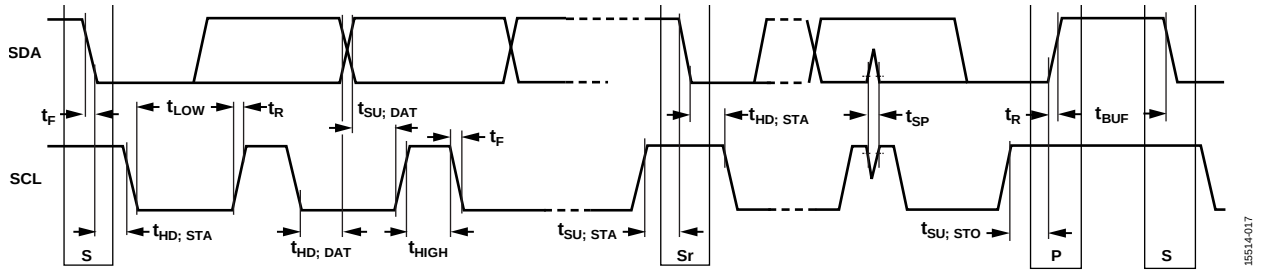
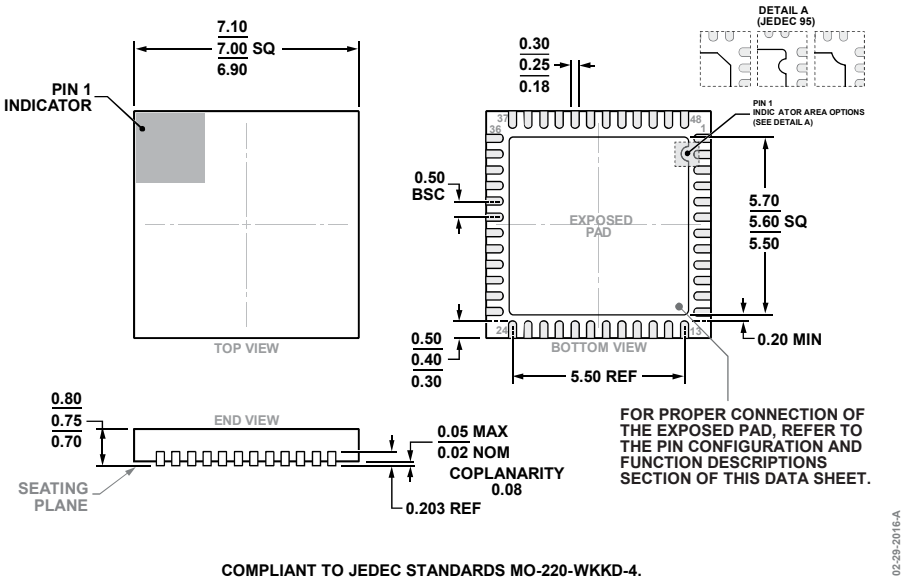


Figure 128. I²C Serial Port Timing

Table 73. I²C Timing Definitions

Parameter	Description
f_{SCL}	Serial clock
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



COMPLIANT TO JEDEC STANDARDS MO-220-WKKD-4.

Figure 129. 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
AD9545BCPZ	−40°C to +85°C	48-Lead Lead Frame Chip Scale Package [LFCSP)	CP-48-13
AD9545BCPZ-REEL7	−40°C to +85°C	48-Lead Lead Frame Chip Scale Package [LFCSP]	CP-48-13
AD9545/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).