Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Register Map: Section 6.2

Microsemi

ZL30260-ZL30263

1-APLL, 6- or 10-Output Any-to-Any Clock Multiplier and Frequency Synthesizer

Data Sheet

December 2016

Features

- **Four Flexible Input Clocks**
    - One crystal/CMOS input
    - Two differential/CMOS inputs
    - One single-ended/CMOS input
    - Any input frequency from 9.72MHz to 1.25GHz (300MHz max for CMOS)
    - Activity monitors, automatic or manual switching
    - Glitchless clock switching by pin or register
- **6 or 10 Any-Frequency, Any-Format Outputs**
    - Any output frequency from 1Hz to 1045MHz
    - High-resolution frac-N APLL with 0ppm error
    - **The APLL has a fractional divider and an integer divider to make two independent frequency families**
    - Output jitter from integer multiply and dividers as low as 0.17ps RMS (12kHz-20MHz)
    - Output jitter from fractional dividers is typically < 1ps RMS, many frequencies <0.5ps RMS
    - Each output has an independent divider
    - Each output configurable as LVDS, LVPECL, HCSL, 2xCMOS or HSTL
    - In 2xCMOS mode, the P and N pins can be different frequencies (e.g. 125MHz and 25MHz)
    - Multiple output supply voltage banks with CMOS output voltages from 1.5V to 3.3V
    - Precise output alignment circuitry and per-output phase adjustment
    - Per-output enable/disable and glitchless start/stop (stop high or low)
- **General Features**
    - Automatic self-configuration at power-up from external (ZL30260 or 2) or internal (ZL30261 or 3) EEPROM; up to 8 configurations pin-selectable
    - External feedback for zero-delay applications
    - Numerically controlled oscillator mode
    - Spread-spectrum modulation mode
    - Generates PCIe 1, 2, 3, 4 low-jitter clocks
    - Easy-to-configure design requires no external VCXO or loop filter components
    - SPI or I^{2}C processor Interface
    - Core supply voltage options: 2.5V only, 3.3V only, 1.8V+2.5V or 1.8V+3.3V
    - Space-saving 8x8mm QFN56 (0.5mm pitch)

Ordering Information

ZL30260LDG1	ext. EEPROM	6 Outputs	Trays
ZL30260LDF1	ext. EEPROM	6 Outputs	Tape and Reel
ZL30261LDG1	int. EEPROM	6 Outputs	Trays
ZL30261LDF1	int. EEPROM	6 Outputs	Tape and Reel
ZL30262LDG1	ext. EEPROM	10 Outputs	Trays
ZL30262LDF1	ext. EEPROM	10 Outputs	Tape and Reel
ZL30263LDG1	int. EEPROM	10 Outputs	Trays
ZL30263LDF1	int. EEPROM	10 Outputs	Tape and Reel

Matte Tin
Package size: 8 x 8 mm, 56 Pin QFN
-40°C to +85°C

Applications

- Frequency conversion and frequency synthesis in a wide variety of equipment types

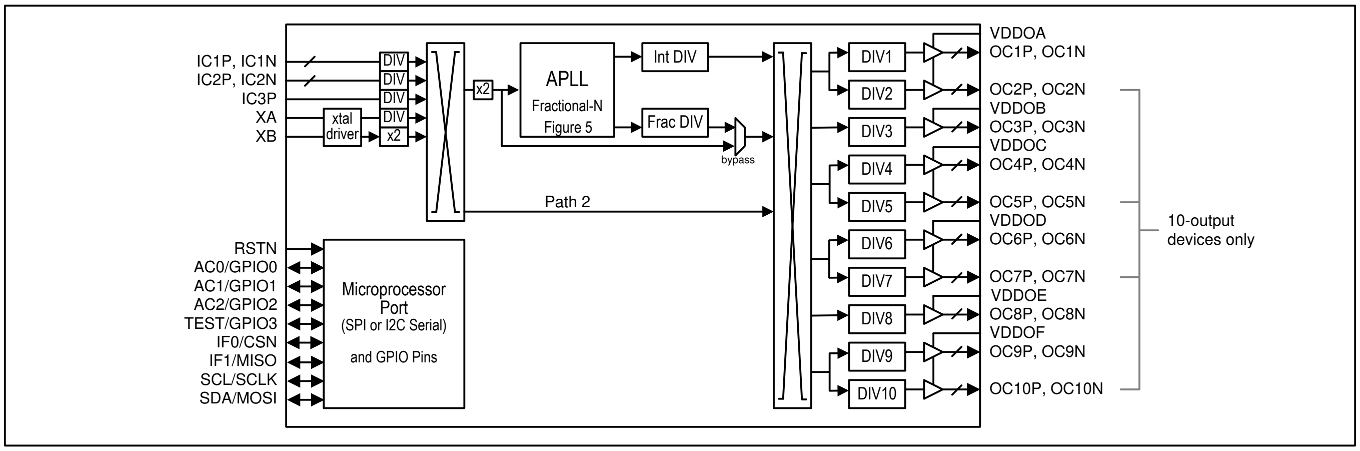

Figure 1 - Functional Block Diagram

Table of Contents

List of Figures

List of Tables

1. Application Example

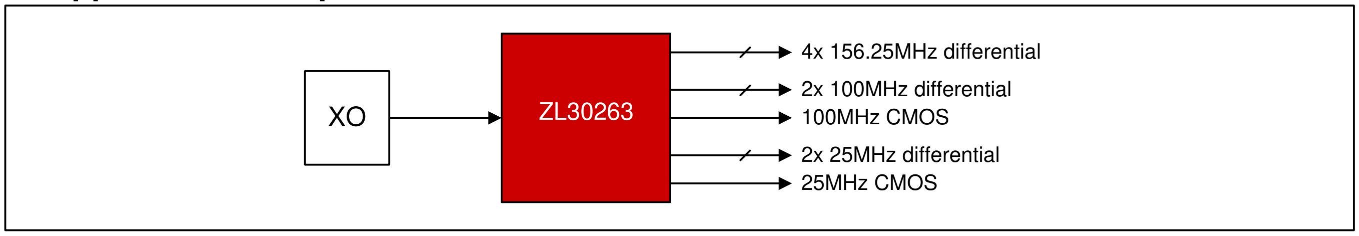

Figure 2 - Application Example: PCIe and Ethernet Clocks for Server Application

2. Detailed Features

2.1 Input Clock Features

- Four input clocks: one crystal/CMOS, two differential/CMOS, one single-ended/CMOS
- Input clocks can be any frequency from 9.72MHz to 1250MHz (differential) or 300MHz (single-ended)
- Supported telecom frequencies include PDH, SDH, Synchronous Ethernet, OTN, wireless
- Activity monitor and glitchless input switching

2.2 APLL Features

- Very high-resolution fractional (i.e. non-integer) frequency multiplication
- Any-to-any frequency conversion with 0ppm error
- Two APLL output dividers: one integer divider (4 to 15 plus half divides 4.5 to 7.5) and one fractional
- Easy-to-configure, completely encapsulated design requires no external VCXO or loop filter
- Bypass mode supports system testing

2.3 Output Clock Features

- Six (ZL30260 or ZL30261) or ten (ZL30262 or ZL30263) low-jitter output clocks
- Each output can be one differential output or two CMOS outputs
- Output clocks can be any frequency from 1Hz to 1045MHz (250MHz max for HCSL, CMOS and HSTL)
- Output jitter from integer multiply and integer dividers as low as 0.17ps RMS (12kHz to 20MHz)
- Output jitter from fractional dividers is typically <1ps RMS, many frequencies <0.5ps RMS (12kHz to 20MHz)
- In CMOS mode, the OCxP and OCxN pins can be different divisors (Example 1: OC3P 125MHz, OC3N 25MHz; Example 2: OC3P 25MHz, OC3N 1Hz/1PPS)
- Outputs directly interface (DC coupled) with LVDS, LVPECL, HSTL, HCSL and CMOS components
- Supported telecom frequencies include PDH, SDH, Synchronous Ethernet, OTN
- Can produce clock frequencies for microprocessors, ASICs, FPGAs and other components
- Can produce PCIe clocks (PCIe gen. 1, 2 and 3)
- Sophisticated output-to-output phase alignment
- Per-output phase adjustment
- Per-output enable/disable
- Per-output glitchless start/stop (stop high or low)

2.4 General Features

- SPI or I^{2}C serial microprocessor interface
- Automatic self-configuration at power-up; pin control to specify one of 8 stored configurations
  ZL30260 and ZL30262: preset configurations in ROM or user configurations in external EEPROM
  ZL30261 and ZL30263: user configurations in internal EEPROM
- Numerically controlled oscillator (NCO) mode allows system software to steer DPLL frequency with resolution better than 0.01ppb
- Spread-spectrum modulation mode (meets PCI Express requirements)
- Zero-delay buffer configuration using an external feedback path
- Four general-purpose I/O pins each with many possible status and control options
- Reference can be fundamental-mode crystal, low-cost XO or clock signal from elsewhere in the system

2.5 Evaluation Software

- Simple, intuitive Windows-based graphical user interface
- Supports all device features and register fields
- Makes lab evaluation of the ZL30260/5/6/7 quick and easy
- Generates configuration scripts to be stored in external (ZL30260,2) or internal (ZL30261,3) EEPROM
- Generates full or partial configuration scripts to be run on a system processor
- Works with or without an evaluation board

3. Pin Diagram

The device is packaged in a 8x8mm 56-pin QFN.

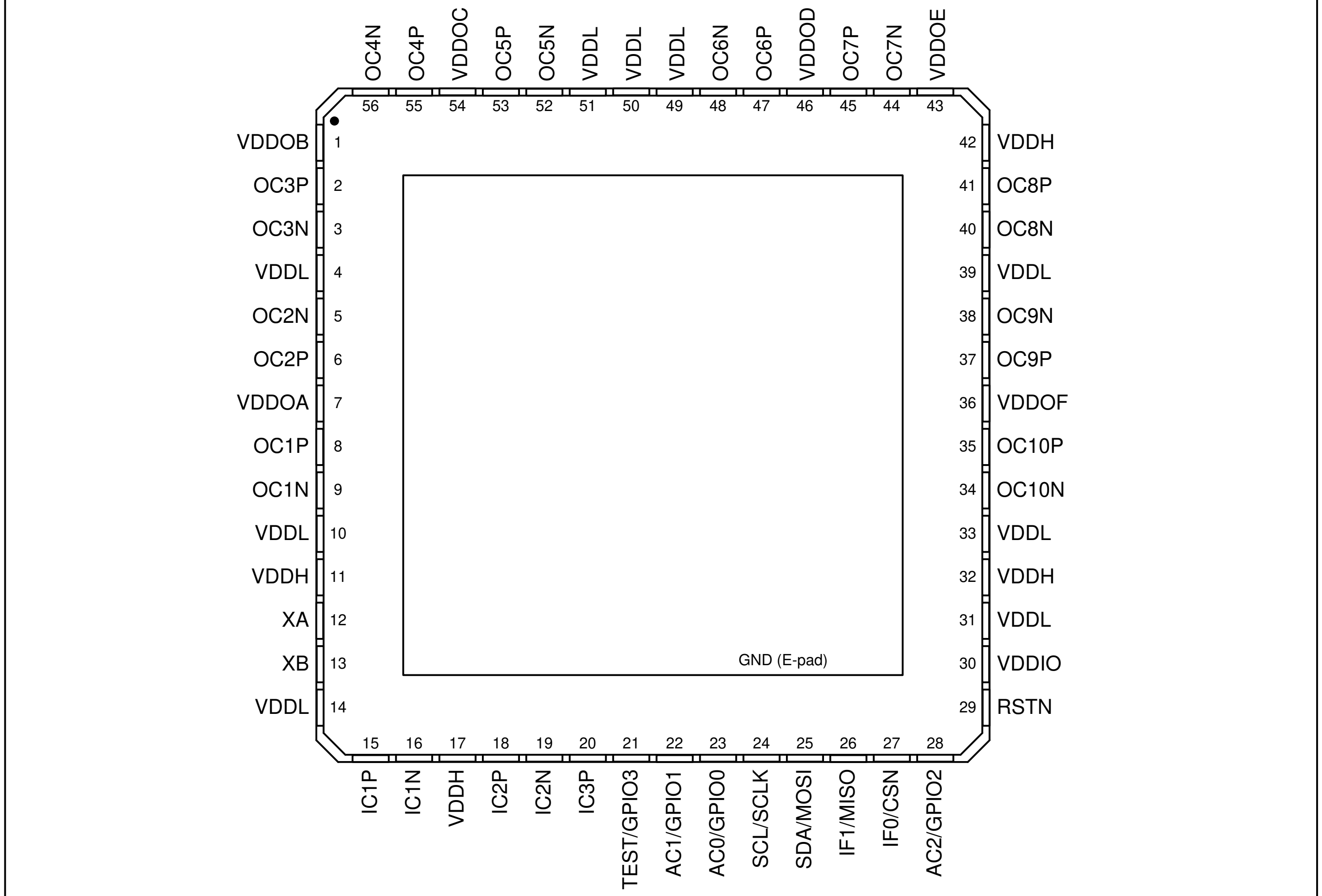

Figure 3 - Pin Diagram

4. Pin Descriptions

All device inputs and outputs are LVCMOS unless described otherwise. The Type column uses the following symbols: I – input, O – output, A – analog, P – power supply pin. All GPIO and SPI/I^{2}C interface pins have Schmitt-trigger inputs and have output drivers that can be disabled (high impedance).

Table 1 - Pin Descriptions

Pin #	Name	Type	Description
15, 16 18, 19 20	IC1P, IC1N IC2P, IC2N IC3P	I I I	**Input Clock Pins** Differential or Single-ended signal format. Programmable frequency. *Differential:* See Table 9 for electrical specifications, and see Figure 15 for recommended external circuitry for interfacing these differential inputs to LVDS, LVPECL, CML or HSCL output pins on neighboring devices. *Single-ended:* For input signal amplitude >2.5V, connect the signal directly to ICxP pin. For input signal amplitude ≤2.5V, AC-coupling the signal to ICxP is recommended. Connect the N pin to a capacitor (0.1µF or 0.01µF) to VSS. As shown in Figure 15, the ICxP and ICxN pins are internally biased to approximately 1.3V. Treat the ICxN pin as a sensitive node; minimize stubs; do not connect to anything else including other ICxN pins. *Unused:* Set ICEN.ICxEN=0. The ICxP and ICxN pins can be left floating. Note that the IC3N pin is not bonded out. A differential signal can be connected to IC3P by AC-coupling the POS trace to IC3P and terminating the signal on the driver side of the coupling cap.
12 13	XA XB	A / I	**Crystal or Input Clock Pins** *Crystal:* MCR1.XAB=01. An on-chip crystal driver circuit is designed to work with an external crystal connected to the XA and XB pins. See section 5.3.2 for crystal characteristics and recommended external components. *Input Clock:* MCR1.XAB=10. An external local oscillator or clock signal can be connected to the XA pin. The XB pin must be left unconnected. The signal on XA can be as large as 3.3V even when VDDH is only 2.5V.
8, 9 6, 5 2, 3 55, 56 53, 52 47, 48 45, 44 41, 40 37, 38 35, 34	OC1P, OC1N OC2P, OC2N OC3P, OC3N OC4P, OC4N OC5P, OC5N OC6P, OC6N OC7P, OC7N OC8P, OC8N OC9P, OC9N OC10P, OC10N	O	**Output Clock Pins** LVDS, programmable differential (which includes LVPECL), HCSL, HSTL or 1 or 2 CMOS. Programmable frequency. Programmable VCM and VOD in programmable differential mode. Programmable drive strength in CMOS and HSTL modes. See Figure 17 for example external interface circuitry. See Table 10, Table 11 and Table 12 for electrical specifications for LVDS, LVPECL and HCSL, respectively. See Table 13 for electrical specifications for interfacing to CMOS and HSTL inputs on neighboring devices. Outputs OC2, OC5, OC7 and OC10 are not present on 6-output products.
29	RSTN	I	**Reset (Active Low)** When this global asynchronous reset is pulled low, all internal circuitry is reset to default values. The device is held in reset as long as RSTN is low. Minimum low time is 1µs.
23 22 28	AC0/GPIO0 AC1/GPIO1 AC2/GPIO2	I/O	**Auto-Configure [2:0] / General Purpose I/O 0, 1 and 2** *Auto Configure:* On the rising edge of RSTN these pins behave as AC[2:0] and specify one of the configurations stored in ROM or EEPROM. See section 5.2.

Pin #	Name	Type	Description
			General-Purpose I/O: After reset these pins are GPIO0, GPIO1 and GPIO2. GPIOCR1 and GPIOCR2.GPIO2C configure these pins. Their states are indicated in GPIOSR which has both real-time and latched status bits.
21	TEST/GPIO3	I/O	**Factory Test / General Purpose I/O 3** *Factory Test:* On the rising edge of RSTN the pin behaves as TEST. Factory test mode is enabled when TEST is high. For normal operation TEST must be low on the rising edge of RSTN. *General-Purpose I/O:* After reset this pin is GPIO3. GPIOCR2.GPIO3C configures the pin. It state is indicated in GPIOSR which has both real-time and latched status bits.
27	IF0/CSN	I/O	**Interface Mode 0 / SPI Chip Select (Active Low)** *Interface Mode:* On the rising edge of RSTN the pin behaves as IF0 and, together with IF1, specifies the interface mode for the device. See section 5.2. *SPI Chip Select:* After reset this pin is CSN. When the device is configured as a SPI slave, an external SPI master must assert (low) CSN to access device registers. When the device is configured as a SPI master (ZL30260, ZL30262 only), the device asserts CSN to access an external SPI EEPROM during auto-configuration and then changes CSN to an input during normal operation. CSN should not be allowed to float.
26	IF1/MISO	I/O	**Interface Mode 1 / SPI Master-In-Slave-Out** *Interface Mode:* On the rising edge of RSTN the pin behaves as IF1 and, together with IF0, specifies the interface mode for the device. See section 5.2. *SPI MISO:* After reset this pin is MISO. When the device is configured as a SPI slave, the device outputs data to an external SPI master on MISO during SPI read transactions. When the device is configured as a SPI master (ZL30260, ZL30262 only), the device receives data on MISO from an external SPI EEPROM during auto-configuration.
24	SCL/SCLK	I/O	**I^{2}C Clock / SPI Clock** *I^{2}C Clock:* When the device is configured as an I^{2}C slave, an external I^{2}C master must provide the I^{2}C clock signal on the SCL pin. *SPI Clock:* When the device is configured as a SPI slave, an external SPI master must provide the SPI clock signal on SCLK. When the device is configured as a SPI master (ZL30260, ZL30262 only), the device drives SCLK as an output to clock accesses to an external SPI EEPROM during auto-configuration.
25	SDA/MOSI	I/O	**I^{2}C Data / SPI Master-Out-Slave-In** *I^{2}C Data:* When the device is configured as an I^{2}C slave, SDA is the bidirectional data line between the device and an external I^{2}C master. *SPI MOSI:* When the device is configured as a SPI slave, an external SPI master sends commands, addresses and data to the device on MOSI. When the device is configured as a SPI master (ZL30260, ZL30262 only), the device sends commands, addresses and data on MOSI to an external SPI EEPROM during auto-configuration.
11,17, 32,42	VDDH	P	**Higher Core Power Supply.** 2.5V or 3.3V ±5%. When VDDH=3.3V the device has additional internal power supply regulators enabled.

Pin #	Name	Type	Description
4,10, 14,31, 33,39, 49,50, 51	VDDL	P	**Lower Core Power Supply.** 1.8V ±5% or same voltage as VDDH.
30	VDDIO	P	**Digital Power Supply for Non-Clock I/O Pins.** 1.8V to VDDH.
7	VDDOA	P	**Power Supply for OC1P/N and OC2P/N.** 1.5V to VDDH.
1	VDDOB	P	**Power Supply for OC3P/N.** 1.5V to VDDH.
54	VDDOC	P	**Power Supply for OC4P/N and OC5P/N.** 1.5V to VDDH.
46	VDDOD	P	**Power Supply for OC6P/N and OC7P/N.** 1.5V to VDDH.
43	VDDOE	P	**Power Supply for OC8P/N.** 1.5V to VDDH.
36	VDDOF	P	**Power Supply for OC9P/N and OC10P/N.** 1.5V to VDDH.
E-pad	VSS	P	**Ground.** 0 Volts.

Important Note: The voltages on VDDL, VDDIO, and all VDDOx pins must not exceed VDDH. Not complying with this requirement may damage the device.

5. Functional Description

5.1 Device Identification

The 12-bit read-only ID field and the 4-bit revision field are found in the ID1 and ID2 registers. Contact the factory to interpret the revision value and determine the latest revision.

5.2 Pin-Controlled Automatic Configuration at Reset

The device configuration is determined at reset (i.e. on the rising edge of RSTN) by the signal levels on these device pins: TEST/GPIO3, AC2/GPIO2, AC1/GPIO1, AC0/GPIO0, IF1/MISO and IF0/CSN. For these pins, the first name (TEST, AC2, AC1, AC0, IF1, IF0) indicates their function when they are sampled by the rising edge of the RSTN pin. The second name refers to their function after reset. The values of these pins are latched into the CFGSR register when RSTN goes high. To ensure the device properly samples the reset values of these pins, the following guidelines should be followed:

1. Any pullup or pulldown resistors used to set the value of these pins at reset should be 1kΩ.
2. RSTN must be asserted at least as long as specified in section 5.9.

The hardware configuration pins are grouped into three sets:

1. TEST - Manufacturing test mode
2. IF[1:0] – Microprocessor interface mode and I^{2}C address
3. AC[2:0] – Auto-config configuration number (0 to 7)

The TEST pin selects manufacturing test modes when TEST=1 (the AC[2:0] pins specify the test mode). For ZL30261 and ZL30263 (devices with internal EEPROM), TEST=1, AC[2:0]=000, IF[1:0]=11 configures the part so that production SPI EEPROM programmers can program the internal EEPROM (see section 5.11.2). TEST=1 and AC[2:0]=011 causes the part to start normally except it does not auto-configure from EEPROM or ROM. For more information about auto-configuration from EEPROM or ROM see section 5.11.

For all of these pins Microsemi recommends that board designs include component sites for both pullup and pulldown resistors (only one or the other populated per pin).

5.2.1 ZL30260 and ZL30262—Internal ROM, External or No EEPROM

For these part numbers the IF[1:0] pins specify the processor interface mode, the I^{2}C slave address and whether the device should auto-configure from internal ROM or external EEPROM. The AC[2:0] pins specify which device configuration in the ROM or EEPROM to execute after reset. Descriptions of the standard-product ROM configurations are available from Microsemi.

IF1	IF0	Processor Interface	Configuration Memory to Use
0	0	I^2C, slave address 11101 00	Internal ROM
0	1	I^2C, slave address 11101 01	Internal ROM
1	0	SPI Slave	Internal ROM
1	1	SPI Master during auto-configuration then SPI Slave	External SPI EEPROM

To configure the device as specified in the first three rows above but *without* auto-configuring from internal ROM, wire devices pins as follows: TEST=1 and AC[2:0]=011, as described in section 5.2.

AC2	AC1	AC0	Auto Configuration
0	0	0	Configuration 0
0	0	1	Configuration 1
0	1	0	Configuration 2
0	1	1	Configuration 3
1	0	0	Configuration 4
1	0	1	Configuration 5
1	1	0	Configuration 6
1	1	1	Configuration 7

Notes about the device auto-configuring from external EEPROM:

1. The device's CSN pin should have a pull-up resistor to VDD to ensure its processor interface is inactive after auto-configuration is complete. The SCLK, MISO and MOSI pins should also have pull-up resistors to VDD to keep them from floating.
2. If a processor or similar device will access device registers after the device has auto-configured from external EEPROM, the SPI SCLK, MOSI and MISO wires can be connected directly to the processor, the device and the external EEPROM. The processor and device CSN pins can be wired together also. The EEPROM CSN signal must be controlled by the device's CSN pin during device auto-configuration and then held inactive when the processor accesses device registers.
3. The bits of the I^2C address are as shown above by default but can be changed in the I2CA register.

5.2.2 ZL30261 and ZL30263—Internal EEPROM

For these part numbers the IF[1:0] pins specify the processor interface mode and the I^2C slave address. The AC[2:0] pins specify which device configuration in the EEPROM to execute after reset.

IF1	IF0	Processor Interface
0	0	I^2C, slave address 11101 00
0	1	I^2C, slave address 11101 01
1	0	I^2C, slave address 11101 10
1	1	SPI Slave

AC2	AC1	AC0	Auto Configuration
0	0	0	Configuration 0
0	0	1	Configuration 1
0	1	0	Configuration 2
0	1	1	Configuration 3
1	0	0	Configuration 4
1	0	1	Configuration 5
1	1	0	Configuration 6
1	1	1	Configuration 7

Note: the bits of the I^2C address are as shown above by default but can be changed in the I2CA register. A device's I^2C slave address can be set to any value during auto-configuration at power-up by writing the I2CA register as part of the configuration script.

5.3 Local Oscillator or Crystal

Section 5.3.1 describes how to connect an external oscillator and the required characteristics of the oscillator. Section 5.3.2 describes how to connect an external crystal to the on-chip crystal driver circuit and the required characteristics of the crystal. The device does not require an external oscillator or crystal for operation.

5.3.1 External Oscillator

A signal from an external oscillator can be connected to the XA pin (XB must be left unconnected).

Table 8 specifies the range of possible frequencies for the XA input. To minimize jitter, the signal must be properly terminated and must have very short trace length. A poorly terminated single-ended signal can greatly increase output jitter, and long single-ended trace lengths are more susceptible to noise. When MCR1.XAB=10, XA is enabled as a single-ended input.

While the stability of the external oscillator can be important, its absolute frequency accuracy is less important because any known frequency inaccuracy of the oscillator can be compensated by adjusting the APLL's fractional feedback divider value (AFBDIV) by ppb or ppm.

The jitter on output clock signals depends on the phase noise and frequency of the external oscillator. For the device to operate with the lowest possible output jitter, the external oscillator should have the following characteristics:

- Phase Jitter: less than 0.1ps RMS over the 12kHz to 5MHz integration band
- Frequency: The higher the better, all else being equal

5.3.2 External Crystal and On-Chip Driver Circuit

The on-chip crystal driver circuit is designed to work with a fundamental mode, AT-cut crystal resonator. See Table 2 for recommended crystal specifications. To enable the crystal driver, set MCR1.XAB=01.

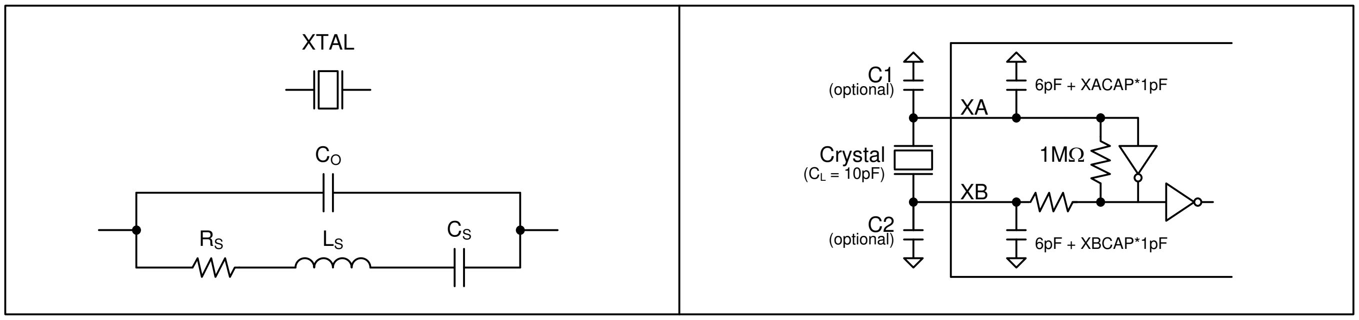

Figure 4 - Crystal Equivalent Circuit / Recommended Crystal Circuit

See Figure 4 for the crystal equivalent circuit and the recommended external component connections. The driver circuit design includes configurable internal load capacitors. For a 10pF crystal the total capacitance on each of XA and XB should be 2 x 10pF = 20pF. To achieve these loads without external capacitors, register field XACR3.XACAP should be set to 20pF minus actual XA external board trace capacitance minus XA's minimum internal capacitance of 6pF. For example, if external trace capacitance is 2pF then XACAP should be set to 20pF – 2pF – 6pF = 12pF. Register field XACR3.XBCAP should be set in a similar manner for XB load capacitance. Crystals with nominal load capacitance other than 10pF usually can be supported with only internal load capacitance. If the XACAP and XBCAP fields do not have sufficient range for the application, capacitance can be increased by using external caps C1 and C2.

Users should also note that on-chip capacitors are not nearly as accurate as discrete capacitors (which can have 1% accuracy). If tight frequency accuracy is required for the crystal driver circuit then set XACAP and XBCAP both to 0 and choose appropriate C1 and C2 capacitors with 1% tolerance.

The crystal, traces, and two external capacitors sites (if included) should be placed on the board as close as possible to the XA and XB pins to reduce crosstalk of active signals into the oscillator. Also no active signals should be routed under the crystal circuitry.

Note: Crystals have temperature sensitivities that can cause frequency changes in response to ambient temperature changes. In applications where significant temperature changes are expected near the crystal, it is recommended that the crystal be covered with a thermal cap, or an external XO or TCXO should be used instead.

Table 2 - Crystal Selection Parameters

Parameter		Symbol	Min.	Typ.	Max.	Units
Crystal Oscillation Frequency[1]		f_{OSC}	25		60	MHz
Shunt Capacitance		C_O		2	5	pF
Load Capacitance[3]		C_L	8	10	16	pF
Equivalent Series Resistance (ESR)[2]	f_{OSC} < 40MHz	R_S			60	Ω
	f_{OSC} > 40MHz	R_S			50	Ω
Maximum Crystal Drive Level			100	100, 200, 300		μW

Note 1: Higher frequencies give lower output jitter, all else being equal.

Note 2: These ESR limits are chosen to constrain crystal drive level to less than 100μW. If the crystal can tolerate a drive level greater than 100μW then proportionally higher ESR is acceptable.

Note 3: For crystals with 100μW max drive level: (a) f_{OSC}>55MHz and C_L≥12pF is not supported, and (b) f_{OSC}>45MHz and C_L≥16pF is not supported. Crystals with max drive level of 200μW or higher do not have these limitations.

Parameter	Symbol	Min.	Typ.	Max.	Units
Crystal Frequency Stability vs. Power Supply	f_{FVD}		0.2	0.5	ppm per 10% Δ in VDD

Any known frequency inaccuracy of the crystal can be compensated in the APLL by adjusting the APLL's fractional feedback divider value (AFBDIV) by ppb or ppm to compensate for crystal frequency error.

5.3.3 Clock Doublers

Figure 1 shows an optional clock doubler (“x2” block) following the crystal driver block. The doubler, which is enabled by setting MCR2.DBL=1, can be used to double the frequency of the internal crystal driver circuit or a 20MHz to 78.125MHz clock signal on the XA pin. For input clock frequencies from 25MHz to 78.125MHz the duty cycle of the signal can be anywhere in the 40% to 60% range. For input clock frequencies from 20MHz to 25MHz the duty cycle must be in the 45% to 55% range. Figure 1 also shows an optional doubler at the input of the APLL. This APLL input doubler, which is enabled by setting ACR1.INDBL=1, can be used to double the frequency of any of the inputs. The following table shows scenarios when the clock doubler can be used.

Scenario	With Crystal	With XO or Clock Signal
APLL, Integer Multiply	Yes[1]	Maybe[1]
APLL, Fractional Multiply	Yes	Yes
NCO	Yes	Yes
Spread-Spectrum	Yes	Yes
APLL bypass path or Path 2	No[2]	No[2]

Note 1: For APLL integer multiplication, use of the doubler is application-dependent. On the positive side, use of the doubler reduces random jitter. On the negative side, the doubler causes a spur at the XA frequency (but this spur may be outside the band of interest for the application).

Note 2: The signal generated by the doubler has a very narrow and variable pulse width and therefore it is not recommended to connect the doubler signal directly to the OCx outputs using the APLL bypass path or Path 2. The doubler signal is fine as an input to the APLL, which filters the duty cycle distortion and produces a 50% duty cycle output.

Note 3: Using both doublers in series to double the XA-doubled signal is not supported.

5.3.4 Ring Oscillator (for Auto-Configuration)

After reset the internal auto-configuration boot controller is clocked by an internal ring oscillator. After auto-configuration is complete (GLOBISR.BCDONE=1) the ring oscillator can be disabled by setting MCR1.ROSCD=1. The device’s processor interface is asynchronous and does not require the ring oscillator.

5.4 Input Signal Format Configuration

Input clocks IC1, IC2 and IC3 are enabled by setting the enable bits in the ICEN register. The power consumed by a differential receiver is shown in Table 6. The electrical specifications for these inputs are listed in Table 9. Each input clock can be configured to accept nearly any differential signal format by using the proper set of external components (see Figure 15). To configure these differential inputs to accept single-ended CMOS signals, connect the single-ended signal to the ICxP pin, and connect the ICxN pin to a capacitor (0.1μF or 0.01μF) to VSS. Each ICxP and ICxN pin is internally biased to approximately 1.3V. If an input is not used, both ICxP and ICxN pins can be left floating. Note that the IC3N pin is not present. A differential signal can be connected to IC3P by AC-coupling the POS trace to IC3P and terminating the signal on the driver side of the coupling cap.

5.5 APLL Configuration

5.5.1 APLL Input Frequency

The frequencies of all enabled input clocks (ICx and XA) associated with the APLL must divide to a common APLL phase-frequency detector (PFD) frequency from 9.72MHz to 156.25MHz. In this mode the input high-speed dividers (ICxCR1.HSDIV) can be used to divide the ICx frequencies by 1, 2, 4 or 8 and the XA divider (MCR2. XODIV2) can be used to divide the XA frequency by 1 or 2. The polarity of an ICx input signal can be inverted by setting ICxCR1.POL.

5.5.2 APLL Input Monitors

Each of the APLL's four inputs—IC1, IC2, IC3 and XA—have a simple activity monitor. If the monitor counts approximately four (eight if the input clock is doubled) APLL feedback clock cycles without seeing an input clock edge, the input is declared invalid and the corresponding ICxSR.ICV bit or XASR.ICV bit is set to 0. The input clock is declared valid, and the corresponding ICxSR.ICV bit or XASR.ICV bit is set to 1, when the input clock has no missing edges in an interval specified by the corresponding ICxCR1.VALTIME or XACR1.VALTIME field. The XASR and ICxSR registers provide real-time and latched status bits indicating the state of each input.

The feedback clock to use for each input monitor is specified by the MCR2.XAMCK and MCR2.ICxMCK bits.

5.5.3 APLL Input Selection

The APLL can lock to any of inputs IC1 through IC3, a clock signal on XA (optionally clock-doubled), or the crystal driver circuit (optionally clock-doubled) when a crystal is connected to XA and XB.

The input to the APLL can be controlled by a register field, a GPIO pin, or a simple input activity monitor. When ACR3.EXTSW=0 and ACR3.INMON=0, the ACR3.APLLMUX register field controls the APLL input mux.

When ACR3.EXTSW=1, a GPIO pin controls the APLL input mux. When the GPIO pin is low, the mux selects the input specified by ACR3.APLLMUX. When the GPIO pin is high, the mux selects the input specified by ACR3.ALTMUX. ACR1.EXTSS specifies which GPIO pin controls this behavior.

When ACR3.EXTSW=0 and ACR3.INMON=1, an input monitor (see section 5.5.2) controls the APLL input mux. When the monitor for the input specified by ACR3.APLLMUX says that input is valid (ICxSR.ICV=1 or XASR.ICV=1), the mux selects the input specified by ACR3.APLLMUX otherwise the mux selects the input specified by ACR3.ALTMUX.

The APLLSR.SELREF real-time status field indicates the APLL's selected reference.

5.5.4 APLL Output Frequency

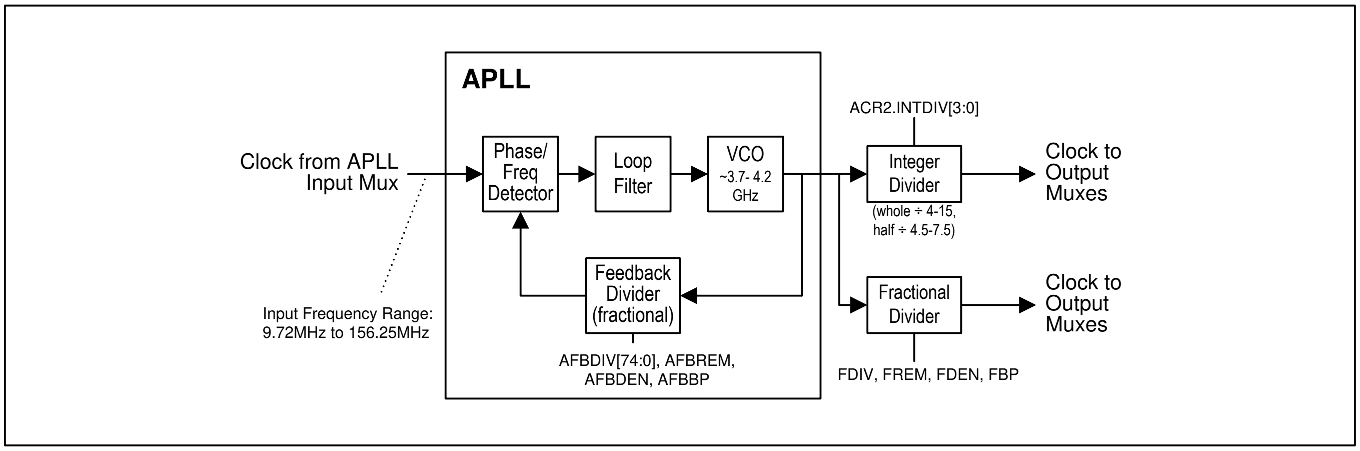

Figure 5 - APLL Block Diagram

The APLL is enabled when PLLEN.APLLEN=1. The APLL has a fractional-N architecture and therefore can produce output frequencies that are either integer or non-integer multiples of the input clock frequency. Figure 5 shows a block diagram of the APLL, which is built around an ultra-low-jitter multi-GHz VCO. Register fields AFBDIV, AFBREM, AFBDEN and AFBBP configure the frequency multiplication ratio of the APLL. The ACR2.INTDIV field specifies how the VCO frequency is divided down by the APLL's integer divider (which can also do some half divides). Dividing by 6 is the typical setting to produce 622.08MHz for SDH/SONET or 625MHz for Ethernet applications. The configuration registers for the APLL's fractional divider are described in section 5.5.5.

Internally, the exact APLL feedback divider value is expressed in the form AFBDIV + AFBREM / AFBDEN * $2^{-(33-\mathrm{AFBBP})}$. This feedback divider value must be chosen such that APLL_input_frequency * feedback_divider_value is in the operating range of the VCO (as specified in Table 14). The AFBDIV term is a fixed-point number with 9 integer bits and a configurable number of fractional bits (up to 33, as specified by AFBBP). Typically AFBBP is set to 9 to specify that AFBDIV has 33 – 9 = 24 fractional bits. Using more than 24 fractional bits does not yield a detectable benefit. Using less than 12 fractional bits is not recommended.

The following equations show how to calculate the feedback divider values for the situation where the APLL should multiply the APLL input frequency by integer M and also fractionally scale by the ratio of integers N / D. In other words, VCO_frequency = input_frequency * M * N / D. An example of this is multiplying 77.76MHz by M=48 and scaling by N / D = 255 / 237 for forward error correction applications.

$$\text{afbdiv} = \text{trunc}(M * N / D * 2^{24}) \qquad (1)$$

$$\text{lsb\_fraction} = M * N / D * 2^{24} - \text{afbdiv} \qquad (2)$$

$$\text{AFBDEN} = D \qquad (3)$$

$$\text{AFBREM} = \text{round}(\text{lsb\_fraction} * \text{AFBDEN}) \qquad (4)$$

$$\text{AFBBP} = 33 - 24 = 9 \qquad (5)$$

$$\text{AFBDIV}[41:0] = \text{afbdiv} * 2^{\text{AFBBP}} \qquad (6)$$

The trunc() function returns only the integer portion of the number. The round() function rounds the number to the nearest integer. In Equation (1), the temporary variable 'afbdiv' is set to the full-precision feedback divider value, M * N / D, truncated after the 24th fractional bit. In Equation (2) the temporary variable 'lsb_fraction' is the fraction that was truncated in Equation (1) and therefore is not represented in the afbdiv value. In Equation (3), AFBDEN is set to the denominator of the original M * N / D ratio. In Equation (4), AFBREM is calculated as the integer numerator of a fraction (with denominator AFBDEN) that equals the 'lsb_fraction' temporary variable. In Equation (5) AFBBP is set to 33 – 24 = 9 to correspond with AFBDIV having 24 fractional bits. Finally, in equation (6) the afbdiv bits are shifted into the proper position for the AFBDIV registers.

When a fractional scaling scenario involves multiplying an integer M times multiple scaling ratios N_1 / D_1 through N_n / D_n, the equations above can still be used if the numerators are multiplied together to get $N = N_1 \times N_2 \times \ldots \times N_n$ and the denominators are multiplied together to get $D = D_1 \times D_2 \times \ldots \times D_n$.

The easiest way to calculate the exact values to write to the APLL registers is to use the ZL3026x evaluation software, available on the Microsemi website. This software can be used even when no evaluation board is attached to the computer.

Note: After the APLL's feedback divider settings are configured in register fields AFBDIV, AFBREM, AFBDEN and AFBBP, the APLL enable bit PLLEN.APLLEN should be changed from 0 to 1 to cause the APLL to reacquire lock with the new settings. The real-time lock/unlock status of the APLL is indicated by APLLSR.ALK.

5.5.5 Fractional Output Divider

As shown in Figure 1 and Figure 5, the APLL has a fractional output divider. This allows the APLL to be the source of two unrelated frequency families, one from the integer divider, and one from the fractional divider.

Configuration of the fractional output divider is very similar to configuration of the APLL's feedback divider. The fractional divider is enabled by setting ACR1.ENFDIV. Internally, the exact divider value is expressed in the form FDIV + FREM / FDEN * $2^{-(36-FBP)}$. The input clock to the fractional divider is APLL VCO frequency divided by 2 ($f_{VCO}/2$). The FDIV term is a fixed-point number with 4 integer bits and a configurable number of fractional bits (up to 36, as specified by FBP). Typically FBP is set to 12 to specify that FDIV has 36 – 12 = 24 fractional bits.

The output clock from the fractional divider has good phase noise on rising edges but worse phase noise on falling edges and can have non-50% duty cycle. Applications that only use clock rising edges can use the fractional divider's output clock directly. For applications that care about 50% duty cycle and/or the phase noise of both rising edges and falling edges, the fractional divider should be followed by an even medium-speed divider value (2, 4, 6, 8…). The low-speed divider can be used to further divide the output clock if needed.

The maximum output frequency for the fractional divider is $f_{VCO}/10$. This means the minimum fractional divider value is 5.0. Including the need for a divide-by-2 in the medium-speed divider, the maximum frequency for a 50% duty-cycle output clock signal is $f_{VCO}/20$. The minimum output frequency for the fractional divider is $f_{VCO}/32$ since the internal FDIV value has 4 integer bits. The combination FDIV=0, FREM=0, FDEN=1 specifies to divide by 16.0. The medium-speed and low-speed dividers can be configured to follow the fractional output divider to create output frequencies down to <1Hz.

The following equations show how to calculate the register values for the situation where the fractional divider should divide by the integer M and the ratio of integers N / D. In other words,

$$\text{frac\_div\_output\_freq} = (\text{VCO\_freq} / 2) / (M * N / D)$$

An example of this is starting with VCO_freq = 3750MHz (to get low-jitter Ethernet frequencies through the APLL's integer divider) and using the APLL's fractional divider to get 155.52MHz for SONET/SDH applications. In this example, M=12, N=15625, D=15552 are appropriate values to get 155.52MHz.

$$\text{fdiv} = \text{trunc}(M * N / D * 2^{24}) \qquad (1)$$

$$\text{lsb\_fraction} = M * N / D * 2^{24} - \text{fdiv} \qquad (2)$$

$$\text{FDEN} = D \qquad (3)$$

$$\text{FREM} = \text{round}(\text{lsb\_fraction} * \text{FDEN}) \qquad (4)$$

$$\text{FBP} = 36 - 24 = 12 \qquad (5)$$

$$\text{FDIV}[39:0] = \text{fdiv} * 2^{\text{FBP}} \qquad (6)$$

The trunc() function returns only the integer portion of the number. The round() function rounds the number to the nearest integer. In Equation (1), the temporary variable 'fdiv' is the full-precision feedback divider value, M * N / D, truncated after the 24th fractional bit. In Equation (2) the temporary variable 'lsb_fraction' is the fraction that was truncated in Equation (1) and therefore is not represented in the fdiv value. In Equation (3), FDEN is set to the denominator of the original M * N / D ratio. In Equation (4), FREM is calculated as the integer numerator of a fraction (with denominator FDEN) that equals the 'lsb_fraction' temporary variable. In Equation (5) FBP is set to 36 – 24 =12 to correspond with FDIV having 24 fractional bits. Finally, in equation (6) the fdiv bits are shifted into the proper position for the FDIV registers.

When a fractional scaling scenario involves multiplying an integer M times multiple scaling ratios N_1 / D_1 through N_n / D_n, the equations above can still be used if the numerators are multiplied together to get N = N_1 x N_2 x … x N_n and the denominators are multiplied together to get D = D_1 x D_2 x … x D_n.

The easiest way to calculate the exact values to write to the APLL's fractional output divider registers is to use the ZL3026x evaluation software, available on the Microsemi website. This software can be used even when no evaluation board is attached to the computer.

5.5.6 Numerically Controlled Oscillator (NCO) Mode

5.5.6.1 Using the APLL's Feedback Divider

System software can steer output frequencies with high resolution by manipulating the APLL's AFBDIV value. The resolution can be better than 0.01ppb.

The nominal AFBDIV value, hereafter referred to as AFBDIV0, is the 0ppm nominal value for the desired device configuration.

(Fractional frequency offset (FFO) is defined as (actual_frequency – nominal_frequency) / nominal_frequency. FFO is a unitless number but is typically expressed in parts per billion (ppb), parts per million (ppm) or percent.)

To control the NCO, system software first reads the AFBDIV0 value from the device. Even though the AFBDIV register description describes AFBDIV as having 9 integer bits and 33 fractional bits, for the NCO calculations that follow, AFBDIV values should be treated as 42-bit unsigned integers.

To change the NCO frequency to a specific FFO (in ppm), system software calculates newAFBDIV (a 42-bit unsigned integer) as follows:

newAFBDIV = round(AFBDIV0 * (1 + FFO/1e6))

System software then writes the newAFBDIV value directly to the AFBDIV registers.

Note that any subsequent frequency changes are calculated using the same equation from the original AFBDIV0 value and are not a function of the previous newAFBDIV value. The value of newAFBDIV should be kept within ±1000ppm of AFBDIV0 and within ±500ppm of the previous newAFBDIV value to avoid causing the APLL to lose lock. If spread spectrum modulation is also in use, the total frequency change caused by spread spectrum modulation and NCO control should be kept within ±5000ppm of AFBDIV0 to avoid causing the APLL to lose lock.

During NCO operation using the APLL's feedback divider, AFBREM should be set to 0, AFBDEN should be set to 1 and AFBBP should be set to 0.

5.5.6.2 Using the Fractional Output Divider

System software can steer output frequencies derived from the fractional output divider with high resolution by manipulating the divider's FDIV value. The resolution can be better than 0.01ppb.

The nominal FDIV value, hereafter referred to as FDIV0, is the 0ppm nominal value for the desired device configuration.

(Fractional frequency offset (FFO) is defined as (actual_frequency – nominal_frequency) / nominal_frequency. FFO is a unitless number but is typically expressed in parts per billion (ppb), parts per million (ppm) or percent.)

To control the NCO, system software first reads the FDIV0 value from the device. Even though the FDIV register description describes FDIV as having 4 integer bits and 36 fractional bits, for the NCO calculations that follow, FDIV values should be treated as 40-bit unsigned integers.

To change the NCO frequency to a specific FFO (in ppm), system software calculates newFDIV (a 40-bit unsigned integer) as follows:

newFDIV = round(FDIV0 / (1 + FFO/1e6))

System software then writes the newFDIV value directly to the FDIV registers.

Note that any subsequent frequency changes are calculated using the same equation from the original FDIV0 value and are not a function of the previous newFDIV value.

During NCO operation using the fractional output divider, FREM should be set to 0, FDEN should be set to 1 and FBP should be set to 0.

5.5.7 Frequency Increment and Decrement

When ACR1.USEFDIV=0 the APLL's feedback divider value can be incremented or decremented by values ranging from approximately 1ppb to 500ppm. The value AID[23:0] x 2^7 is added to the APLL's feedback divider value each time the trigger specified by AIDCR.FISS changes state. AID[23:0] x 2^7 is subtracted from the APLL's feedback divider value each time the trigger specified by AIDCR.FDSS changes state. The value to be written to AID[23:0] s as follows:

AID = round(AFBDIV0 * FFO/1e6 / 2^7)

where FFO is the desired fractional frequency offset (FFO) per increment or decrement step in ppm

AFBDIV0 is the nominal AFBDIV value, obtained by reading AFBDIV when AFBDL.RDCUR=0

The current APLL feedback divider value (i.e. the value after increments and decrements) can be read from the AFBDIV registers when AFBDL.RDCUR=1. The original value of the AFBDIV registers before increments and decrements can be read from the AFBDIV registers when AFBDL.RDCUR=0. Incrementing and decrementing only occur when the APLL is locked (APLLSR.ALK=1). If the APLL loses lock, when it locks again the APLL sets its feedback divider value back to the AFBDIV0 value. The system must be designed to ensure the current feedback divider value stays within ±1000ppm of the AFBDIV0 value to avoid causing the APLL to lose lock. The maximum increment/decrement is 500ppm. Frequency increment/decrement behavior is mutually exclusive with spread-spectrum modulation (see section 5.5.8) because both behaviors use the AID registers. Frequency increment/decrement is enabled when ACR1.ENFID=1. When ENFID is set to 0 the APLL feedback divider instantly changes to the AFBDIV0 value. Therefore, to avoid a frequency jump on output clocks, system software should increment or decrement back to the AFBDIV0 value before changing ENFID to 0.

5.5.8 Spread-Spectrum Modulation Mode

For EMI-sensitive applications such as PCI Express, the device can perform spread spectrum modulation (SSM). In SSM the frequency of the output clock is continually varied over a narrow frequency range to spread the energy of the signal and thereby reduce EMI. This mode is a special case of NCO mode.

Spread spectrum mode is enabled by the ASCR.ENSS bit or by a GPIO pin as specified by ASCR.SPRDSS.

5.5.8.1 Using the APLL's Feedback Divider

When ACR1.USEFDIV=0 the device performs frequency modulation by modulating the APLL's feedback divider value starting from the nominal value specified in the AFBDIV registers.

For center-spread applications (ASCR.DWNEN=0), the frequency modulation is triangle-wave center-spread of up to ±0.5% deviation from the center frequency with modulation rate configurable from 25kHz to 55kHz. The spread deviation and modulation rate are controlled by specifying an increment/decrement step in the AID registers and the number of APLL input clock cycles to increment and decrement in the ASCNT registers. The values to be written to the device are calculated as follows:

$$ASCNT = round\left(\frac{F_{IN}}{4 * F_{MOD}}\right) - 2$$

$$AID = round\left(\frac{F_{VCO}}{F_{IN}} * \frac{S * 2^{32}}{ASCNT + 2}\right)$$

where F_{VCO} is the APLL"s VCO frequency in Hz,

F_{IN} is the input frequency at the APLL's PFD in Hz,

F_{MOD} is the spread-spectrum modulation frequency in Hz,

and S is the peak-to-peak spread percentage expressed as a decimal (example ±0.5% → S=0.01)

For down-spread applications (ASCR.DWNEN=1), such as PCI Express Refclk, the frequency modulation is triangle-wave down-spread of up to -1% deviation from the nominal frequency with modulation rate configurable from 25kHz to 55kHz. The spread deviation and modulation rate are controlled by specifying an increment/decrement step in the AID registers and the number of APLL input clock cycles to increment and decrement in the ASCNT registers. The values to be written to the device are calculated as follows:

$$ASCNT = round\left(\frac{F_{IN}}{2 * F_{MOD}}\right) - 2$$

$$AID = round\left(\frac{F_{VCO}}{F_{IN}} * \frac{S * 2^{33}}{ASCNT + 2}\right)$$

All of the input parameters are the same as for center spread above. Note the small differences between these down-spread equations and the center-spread equations above. ASCNT here has 2 in the denominator while it has 4 for center-spread. AID has 2^{33} in the numerator for down-spread while it has 2^{32} for center-spread.

During spread-spectrum operation using the APLL's feedback divider, AFBREM should be set to 0, AFBDEN should be set to 1 and AFBBP should be set to 0. Spread-spectrum modulation only occurs when the APLL is locked (APLLSR.ALK=1).

5.5.8.2 Using the Fractional Output Divider

When ACR1.USEFDIV=1 the device performs frequency modulation by modulating the fractional output divider value starting from the nominal value specified in the FDIV registers.

For center-spread applications (ASCR.DWNEN=0), the frequency modulation is triangle-wave center-spread of up to ±5% deviation from the center frequency with modulation rate configurable from 10kHz to 100kHz. (Values outside of these ranges are often achievable as well.) The spread deviation and modulation rate are controlled by specifying an increment/decrement step in the AID registers and the number of APLL input clock cycles to increment and decrement in the ASCNT registers. The ASCR.CNTEN register field must also be set propertly. The values to be written to the device are calculated as follows:

$$CNTEN = \begin{array}{l} 0 \text{ for } F_{IN} < 50MHz \\ 1 \text{ for } 50MHz \leq F_{IN} < 100MHz \\ 2 \text{ for } F_{IN} \geq 100MHz \end{array}$$

$$ASCNT = round\left(\frac{F_{IN}}{4 * F_{MOD} * 2^{CNTEN}}\right) - 2$$

$$AID = round\left(\frac{F_{VCO}}{F_{OUT}} * \frac{S * 2^{34}}{(ASCNT + 2) * (1 - 0.5S)}\right)$$

where F_{VCO} is the APLL"s VCO frequency in Hz,

F_{IN} is the input frequency at the APLL's PFD in Hz,

F_{OUT} is the fractional output divider block's output frequency in Hz,

F_{MOD} is the spread-spectrum modulation frequency in Hz,

and S is the peak-to-peak spread percentage expressed as a decimal (example ±0.5% → S=0.01)

For down-spread applications (ASCR.DWNEN=1), such as PCI Express Refclk, the frequency modulation is triangle-wave down-spread of up to -10% deviation from the nominal frequency with modulation rate configurable from 10kHz to 100kHz. (Values outside of these ranges are often achievable as well.) The spread deviation and modulation rate are controlled by specifying an increment/decrement step in the AID registers and the number of APLL input clock cycles to increment and decrement in the ASCNT registers. The ASCR.CNTEN register field must also be set propertly. The values to be written to the device are calculated as follows:

$$CNTEN = \begin{cases} 0 \text{ for } F_{IN} < 50MHz \\ 1 \text{ for } 50MHz \le F_{IN} < 100MHz \\ 2 \text{ for } F_{IN} \ge 100MHz \end{cases}$$

$$ASCNT = round\left(\frac{F_{IN}}{2 * F_{MOD} * 2^{CNTEN}}\right) - 2$$

$$AID = round\left(\frac{F_{VCO}}{F_{OUT}} * \frac{S * 2^{35}}{(ASCNT + 2) * (1 - 0.5S)}\right)$$

All of the input parameters are the same as for center spread above. Note the small differences between these down-spread equations and the center-spread equations above. ASCNT here has 2 in the denominator while it has 4 for center-spread. AID has 2^{35} in the numerator for down-spread while it has 2^{34} for center-spread.

During spread-spectrum operation using the fractional output divider, FREM should be set to 0, FDEN should be set to 1 and FBP should be set to 0. The F1CR1.MODE field must be set to 1 when doing spread-spectrum modulation with the fractional output divider value. Spread-spectrum modulation only occurs when the APLL is locked (APLLSR.ALK=1).

5.5.9 APLL Phase Adjustment

The phase of the APLL's output clock can be incremented or decremented. When the APLL's AFBDIV value is not an exact multiple of 0.5 then the phase adjustment step size is 1/8th of a VCO cycle. This phase step size is 30ps at maximum VCO frequency of 4180MHz and 33.7ps at minimum VCO frequency of 3715MHz. The ACR4.PDSS field specifies the phase decrement control signal, which can be the ACR4.DECPH bit or any of the four GPIOs. The ACR4.PISS field specifies the phase increment control signal, which can be the ACR4.INCPH bit or any of the four GPIOs. Phase is adjusted on every rising edge and every falling edge of the control signal. This phase adjustment affects the output of the APLL's output dividers (integer and fractional).

When the APLL's AFBDIV value is an exact multiple of 0.5 then the phase adjustment step size is one VCO cycle.

5.6 Output Clock Configuration

The ZL30260 and ZL30261 have six output clock signal pairs while the ZL30262 and ZL30263 have ten. Each output has individual divider, enable and signal format controls. In CMOS mode each signal pair can become two CMOS outputs, allowing the device to have up to 12 or 20 output clock signals. Also in CMOS mode, the OCxN pin can have an additional divider allowing the OCxN frequency to be an integer divisor of the OCxP frequency (example: OC3P 125MHz and OC3N 25MHz). The outputs can be aligned relative to each other, and the phases of output signals can be adjusted dynamically with high resolution.

5.6.1 Output Enable, Signal Format, Voltage and Interfacing

To use an output, the output driver must be enabled by setting OCxCR2.OCSF≠0, and the per-output dividers must be enabled by setting the appropriate bit in the OCEN register. The per-output dividers include the medium-speed divider, the low-speed divider and the associated phase adjustment/alignment circuitry and start/stop logic.

Using the OCxCR2.OCSF register field, each output pair can be disabled or configured as LVDS, LVPECL, HCSL, HSTL, or one or two CMOS outputs. When an output is disabled it is high impedance, and the output driver is in a low-power state. In CMOS mode, the OCxN pin can be disabled, in-phase or inverted vs. the OCxP pin. All of these options are specified by OCxCR2.OCSF. The clock to the output driver can inverted by setting OCxCR2.POL=1. The CMOS/HSTL output driver can be set to any of four drive strengths using OCxCR2.DRIVE.

When OCxCR2.OCSF=0001 the output driver is in LVDS mode. V_{OD} is forced to 400mV and OCxDIFF.VOD is ignored. V_{CM} can be configured in OCxDIFF.VCM, but the default value of 0000 is typically used to get V_{CM}=1.23V for LVDS.

When OCxCR2.OCSF=0010 the output driver is in programmable differential mode. In this mode the output swing (V_{OD}) can be set in OCxDIFF.VOD and the common-mode voltage can be set in OCxDIFF.VCM. Together these fields allow the output signal to be customized to meet the requirements of the clock receiver and minimize the need for external components. By default, when OCSF=0010 the output is configured for LVPECL signal swing with a 1.23V common mode voltage. This gives a signal that can be AC-coupled (after a 100Ω termination resistor) to receivers that are LVPECL or that require a larger signal swing than LVDS. The output driver can also be configured for LVPECL output with standard 2.0V common-mode voltage by seting OCxDIFF.VCM for 2.0V and setting OCxREG.VREG appropriately.

In both LVDS mode and programmable differential mode the output driver requires a DC path through a 100Ω resistor between OCxP and OCxN for proper operation. This resistor is usually placed as close as possible to the receiver inputs to terminate the differential signal. If the receiver requires a common-mode voltage that cannot be matched by the output driver then the POS and NEG signals can be AC-coupled to the receiver after the 100Ω resistor.

HCSL mode requires a DC path through a 50Ω resistor to ground on each of OCxP and OCxN. Note that each of the OCxDIFF.VCM, OCxDIFF.VOD and OCxREG.VREG register fields has a particular setting required for HCSL signal format. See the descriptions of these fields for details.

Outputs are grouped into six power supply banks, VDDOA through VDDOF to allow CMOS or HSTL signal swing from 1.5V to 3.3V for glueless interfacing to neighboring components. 10-output products have outputs grouped into banks in a 2-1-2-2-1-2 arrangement, as shown in Figure 1. 6-output products have one output per bank. If OCSF is set to HSTL mode then a 1.5V power supply voltage should be used to get a standards-compliant HSTL output. Note that LVDS, LVPECL and HCSL signal formats must have a power supply of 2.5V or 3.3V. Also note that VDDO voltage must not exceed VDDH voltage.

5.6.2 Output Frequency Configuration

The frequency of each output is determined by the configuration of the APLL, the APLL's output dividers, and the per-output dividers. Each bank of outputs can be connected to APLL's integer divider, the APLL's fractional divider, or Path 2 (see section 5.15) using the appropriate field in the OCMUX registers.

Each output has two output dividers, a 7-bit medium-speed divider (OCxCR1.MSDIV) and a 24-bit low-speed output divider (LSDIV field in the OCxDIV registers). These dividers are in series, medium-speed divider first then output divider. These dividers produce signals with 50% duty cycle for all divider values including odd numbers. The low-speed divider can only be used if the medium-speed divider is used (i.e. OCxCR1.MSDIV>0).

Since each output has its own independent dividers, the device can output families of related frequencies that have an APLL output frequency as a common multiple. For example, for Ethernet clocks, a 625MHz APLL output clock can be divided by four for one output to get 156.25MHz, divided by five for another output to get 125MHz, and divided by 25 for another output to get 25MHz. Similarly, for SDH/SONET clocks, a 622.08MHz APLL output clock can be divided by 4 to get 155.52MHz, by 8 to get 77.76MHz, by 16 to get 38.88MHz or by 32 to get 19.44MHz.

Two Different Frequencies in 2xCMOS Mode

When an output is in 2xCMOS mode it can be configured to have the frequency of the OCxN clock be an integer divisor of the frequency of the OCxP clock. Examples of where this can be useful:

- 125MHz on OCxP and 25MHz on OCxN for Ethernet applications
- 77.76MHz on OCxP and 19.44MHz on OCxN for SONET/SDH applications
- 25MHz on OCxP and 1Hz (i.e. 1PPS) on OCxN for telecom applications with Synchronous Ethernet and IEEE1588 timing

An output can be configured to operate like this by setting the LSDIV value in the OCxDIV registers to OCxP_freq / OCxN_freq - 1 and setting OCxCR3.LSSEL=0 and OCxCR3.NEGLSD=1. Here are some notes about this dual-frequency configuration option:

- In this mode only the medium speed divider is used to create the OCxP frequency. The low-speed divider is then used to divide the OCxP frequency down to the OCxN frequency. This means that the lowest OCxP frequency is the APLL divider output frequency divided by 128.
- An additional constraint is that the medium-speed divider must be configured to divide by 2 or more (i.e. must have OCxCR1.MSDIV≥1).

5.6.3 Output Duty Cycle Adjustment

For output frequencies less than or equal to 141.666MHz, the duty cycle of the output clock can be modified using the OCxDC.OCDC register field. This behavior is only available when MSDIV>0 and LSDIV > 1. When OCDC = 0 the output clock is 50%. Otherwise the clock signal is a pulse with a width of OCDC number of MSDIV output clock periods. The range of OCDC can create pulse widths of 1 to 255 MSDIV output clock periods. When OCxCR2.POL=0, the pulse is high and the signal is low the remainder of the cycle. When POL=1, the pulse is low and the signal is high the remainder of the cycle.

Note that duty cycle adjustment is done in the low-speed divider. Therefore when OCxCR3.LSSEL=0 the duty cycle of the output is not affected. Also, when a CMOS output is configured with OCxCR3.LSSEL=0 and OCxCR3.NEGLSD=1, the OCxN pin has duty cycle adjustment but the OCxP pin does not. This allows a higher-speed 50% duty cycle clock signal to be output on the OCxP pin and a lower-speed frame/phase/time pulse (e.g. 2kHz, 8kHz or 1PPS) to be output on the OCxN pin at the same time.

An output configured for CMOS or HSTL signal format should not be configured to have a duty cycle with high time shorter than 2ns or low time shorter than 2ns.

5.6.4 Output Phase Adjustment

The phase of an output signal can be shifted by 180° by setting OCxCR2.POL=1. In addition, the phase can be adjusted using the OCxPH.PHADJ register field. The adjustment is in units of bank source clock cycles. For example, if the bank source clock is 625MHz (from the APLL for example) then one bank source clock cycle is 1.6ns, the smallest phase adjustment is 0.8ns, and the adjustment range is ±5.6ns.

5.6.5 Output-to-Output Phase Alignment

A 0-to-1 transition of the ACR1.DALIGN bit causes a simultaneous reset of the medium-speed dividers and low-speed dividers for all output clocks following the APLL where OCxCR1.PHEN=1. After this reset, all PHEN=1 output clocks with frequencies that are exactly integer multiples of one another are rising-edge aligned, with the phase of each output clock signal adjusted as specified by its OCxPH.PHADJ register field. Similarly a 0-to-1 transition of the P2CR1.DALIGN bit aligns all output clocks following Path 2 where OCxCR1.PHEN=1. Alignment is not glitchess; i.e. it may cause a short high time or low time on participating output clock signals. A glitchless alignment can be accomplished by first stopping the clocks, then aligning them, then starting them. Output clock start and stop is described in section 5.6.7.

5.6.6 Output-to-Input Phase Alignment

The best approach for achieving output-to-input phase alignment is to use external feedback in which an OCx output is connected to an ICx input. To enable external feedback, set AFBDL.EXTFB=1, set AFBDL.FBSEL to specify the external feedback path, and provide the associated output-to-input wiring on the PCB. In this configuration the APLL, in a closed-loop manner, automatically phase-aligns all OCx outputs from the APLL to the APLL's selected reference. Any small error in this alignment due to wire delays can be compensated in the outputs' phase adjustment registers, OCxPH.PHADJ.

5.6.7 Output Clock Start and Stop

Output clocks can be stopped high or low or high-impedance. One use for this behavior is to ensure "glitchless" output clock operation while the output is reconfigured or phase aligned with some other signal.

Each output has an OCxSTOP register with fields to control this behavior. The OCxSTOP.MODE field specifies whether the output clock signal stops high, low, or high-impedance. The OCxSTOP.SRC field specifies the source of the stop signal. Options include control bits or one of the GPIO pins. When OCxSTOP.SRC=0001 the output clock is stopped when the corresponding bit is set in the STOPCR registers OR the MCR1.STOP bit is set.

When the stop mode is Stop High (OCxSTOP.MODE=x1) and the stop signal is asserted, the output clock is stopped after the next rising edge of the output clock. When the stop mode is Stop Low (OCxSTOP.MODE=x0) and the stop signal is asserted, the output clock is stopped after the next falling edge of the output clock. When the output is stopped, the output driver can optionally go high-impedance (OCxSTOP.MODE=1x). Internally the clock signal continues to toggle while the output is stopped. When the stop signal is deasserted, the output clock resumes on the opposite edge that it stopped on. Low-speed output clocks can take long intervals before being stopped after the stop signal goes active. For example, a 1 Hz output could take up to 1 second to stop.

When OCxCR2.POL=1 the output stops on the opposite polarity that is specified by the OCxSTOP.MODE field.

Generally OCxCR1.MSDIV must be > 0 for this function to operate correctly since MSDIV=0 bypasses the start-stop circuits.

When MSDIV=0, OCxSTOP.MODE=11 (stop high then go high-impedance) can be used to make outputs high-impedance, but the action won't necessarily be glitchless. To use this behavior to get "stop *low* then go-impedance" behavior, OCxCR2.POL can be set to 1.

Note that when OCxCR3.NEGLSD=1 the start-stop logic is bypassed for the OCxN pin, and OCxN may not start/stop without glitches.

Each output has a status register (OCxSR) with several stop/start status bits. The STOPD bit is a real-time status bit indicating stopped or not stopped. The STOPL bit is a latched status bit that is set when the output clock has stopped. The STARTL bit is a latched status bit that is set when the output clock has started.

5.7 Microprocessor Interface

The device can communicate over a SPI interface or an I^2C interface.

In SPI mode ZL3026x devices without internal EEPROM can be configured at reset to be a SPI slave to a processor master or a SPI master to an external EEPROM slave. (SPI master operation changes to SPI slave operation after auto-configuration from the external EEPROM is complete.) The ZL3026x devices with internal EEPROM can only be configured as a SPI slave to a processor master. All devices are always slaves on the I^2C bus.

Section 5.2 describes reset pin settings required to configure the device for these interfaces.

5.7.1 SPI Slave

The device can present a SPI slave port on the CSN, SCLK, MOSI, and MISO pins. SPI is a widely used master/slave bus protocol that allows a master and one or more slaves to communicate over a serial bus. SPI masters are typically microprocessors, ASICs or FPGAs. Data transfers are always initiated by the master, which also generates the SCLK signal. The device receives serial data on the MOSI (Master Out Slave In) pin and transmits serial data on the MISO (Master In Slave Out) pin. MISO is high impedance except when the device is transmitting data to the bus master.

Bit Order. The register address and all data bytes are transmitted most significant bit first on both MOSI and MISO.

Clock Polarity and Phase. The device latches data on MOSI on the rising edge of SCLK and updates data on MISO on the falling edge of SCLK. SCLK does not have to toggle between accesses, i.e., when CSN is high.

Device Selection. Each SPI device has its own chip-select line. To select the device, the bus master drives its CSN pin low.

Command and Address. After driving CSN low, the bus master transmits an 8-bit command followed by a 16-bit register address. The available commands are shown below.

Table 3 - SPI Commands

Command	**Hex**	**Bit Order, Left to Right**
Write Enable	0x06	0000 0110
Write	0x02	0000 0010
Read	0x03	0000 0011
Read Status	0x05	0000 0101

Read Transactions. The device registers are accessible when EESEL=0. On ZL3026x devices with internal EEPROM, the EEPROM memory is accessible when the EESEL bit is 1. On ZL3026x devices without internal EEROM, the EESEL bit must be set to 0. After driving CSN low, the bus master transmits the read command followed by the 16-bit address. The device then responds with the requested data byte on MISO, increments its address counter, and prefetches the next data byte. If the bus master continues to demand data, the device continues to provide the data on MISO, increment its address counter, and prefetch the following byte. The read transaction is completed when the bus master drives CSN high. See Figure 6.

Register Write Transactions. The device registers are accessible when EESEL=0. After driving CSN low, the bus master transmits the write command followed by the 16-bit register address followed by the first data byte to be written. The device receives the first data byte on MOSI, writes it to the specified register, increments its internal address register, and prepares to receive the next data byte. If the master continues to transmit, the device continues to write the data received and increment its address counter. The write transaction is completed when the bus master drives CSN high. See Figure 8.

EEPROM Writes (ZL30261, ZL30263 Only). The internal EEPROM memory is accessible when the EESEL bit is 1. After driving CSN low, the bus master transmits the write enable command and then drives CSN high to set the internal write enable latch. The bus master then drives CSN low again and transmits the write command followed by the 16-bit address followed by the first data byte to be written. The device first copies the page to be written from EEPROM to its page buffer. The device then receives the first data byte on MOSI, writes it to its page buffer, increments its internal address register, and prepares to receive the next data byte. If the master continues to transmit, the device continues to write the data received to its page buffer and continues to increment its address counter. The address counter rolls over at the 32-byte page boundary (i.e. when the five least-significant address bits are 11111). When the bus master drives CSN high, the device transfers the data in the page buffer to the appropriate page in the EEPROM memory. See Figure 7 and Figure 8.

EEPROM Read Status (ZL30261, ZL30263 Only). After the bus master drives CSN high to end an EEPROM write command, the EEPROM memory is not accessible for up to 5ms while the data is transferred from the page buffer. To determine when this transfer is complete, the bus master can use the Read Status command. After driving CSN low, the bus master transmits the Read Status command. The device then responds with the status byte on MISO. In this byte, the least significant bit is set to 1 if the transfer is still in progress and 0 if the transfer has completed.

Early Termination of Bus Transactions. The bus master can terminate SPI bus transactions at any time by pulling CSN high. In response to early terminations, the device resets its SPI interface logic and waits for the start of the next transaction. If a register write transaction is terminated prior to the SCLK edge that latches the least significant bit of a data byte, the data byte is not written. On devices with internal EEPROM, if an EEPROM write transaction is terminated prior to the SCLK edge that latches the least significant bit of a data byte, none of the bytes in that write transaction are written.

Design Option: Wiring MOSI and MISO Together. Because communication between the bus master and the device is half-duplex, the MOSI and MISO pins can be wired together externally to reduce wire count. To support this option, the bus master must not drive the MOSI/MISO line when the device is transmitting.

AC Timing. See Table 19 and Figure 18 for AC timing specifications for the SPI interface.

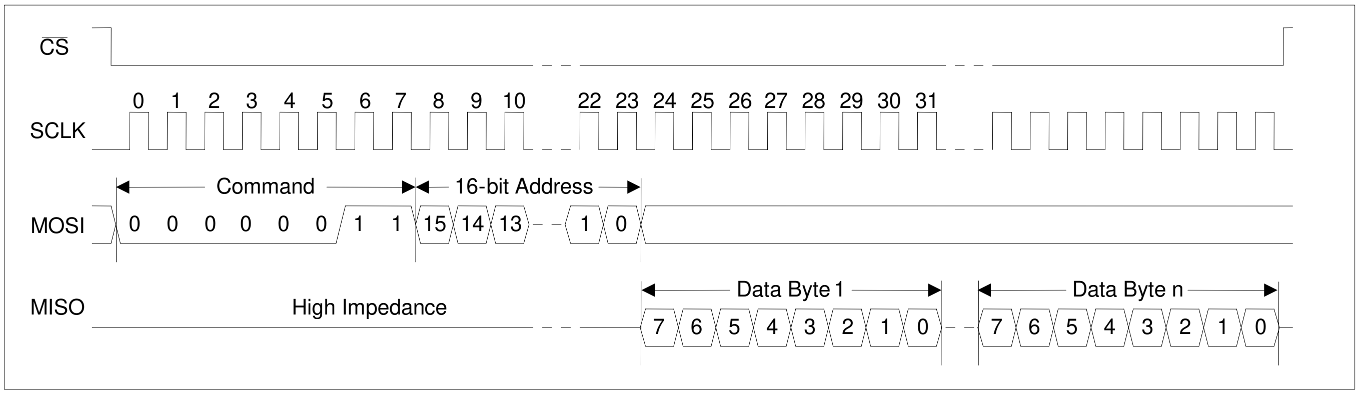

Figure 6 - SPI Read Transaction Functional Timing

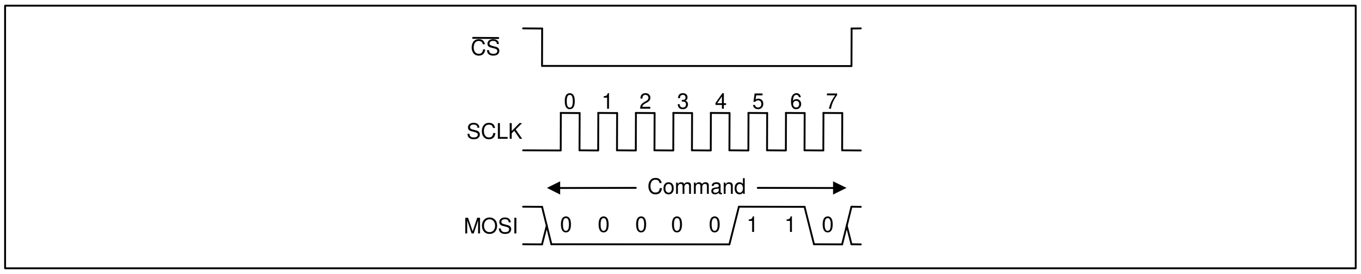

Figure 7 - SPI Write Enable Transaction Functional Timing (ZL30261 and ZL30263 Only)

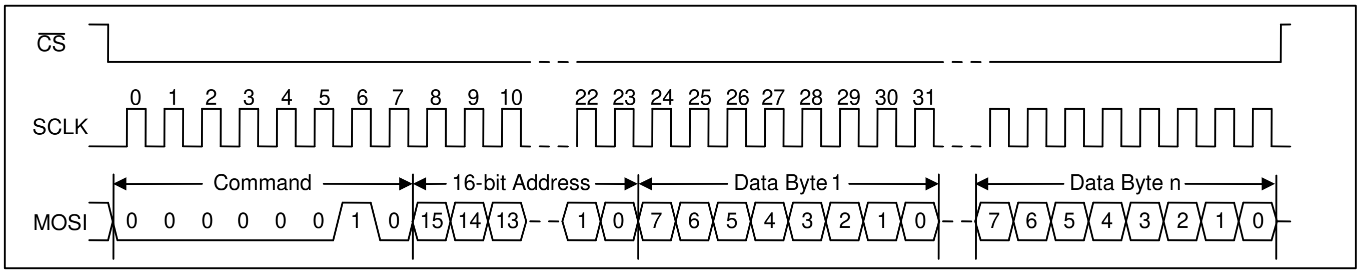

Figure 8 - SPI Write Transaction Functional Timing

5.7.2 SPI Master (ZL30260 and ZL30262 Only)

After reset these devices can present a SPI master port on the CSN, SCLK, MOSI, and MISO pins for auto-configuration using data read from an external SPI EEPROM. During auto-configuration the device is always the SPI master and generates the CSN and SCLK signals. The device transmits serial data on the the MOSI (Master Out Slave In) pin and receives serial data on the MISO (Master In Slave Out) pin.

Bit Order. The register address and all data bytes are transmitted most significant bit first on both MOSI and MISO.

Clock Polarity and Phase. The device latches data on MISO on the rising edge of SCLK and updates data on MOSI on the falling edge of SCLK.

Device Selection. Each SPI device has its own chip-select line. To select the external EEPROM, the device drives the CSN signal low.

Command and Address. After driving CSN low, the device transmits an 8-bit read command followed by a 16-bit register address. The read command is shown below.

Command	Hex	Bit Order, Left to Right
Read	0x03	0000 0011

Read Transactions. After driving CSN low, the device transmits the read command followed by the 16-bit register address. The external EEPROM then responds with the requested data byte on MISO, increments its address counter, and prefetches the next data byte. If the device continues to demand data, the EEPROM continues to provide the data on MISO, increment its address counter, and prefetch the following byte. The read transaction is completed when the device drives CSN high. See Figure 6.

Writing the External EEPROM. Due to the small package size and low pin count of the device, there is no way to use the ZL30260 or ZL30262 to write the external EEPROM. The auto-configuration data used by the ZL30260 or ZL30262 must be pre-programmed into the EEPROM by some other method, such as:

1. The EEPROM manufacturer can write the data to the EEPROM during production testing.
   This is a service they routinely provide.

2. A contract manufacturer or distributor can write the data to the EEPROM using a production EEPROM programmer before the EEPROM is mounted to the board.

5.7.3 I^{2}C Slave

The device can present a fast-mode (400kbit/s) I^{2}C slave port on the SCL and SDA pins. I^{2}C is a widely used master/slave bus protocol that allows one or more masters and one or more slaves to communicate over a two-wire serial bus. I^{2}C masters are typically microprocessors, ASICs or FPGAs. Data transfers are always initiated by the master, which also generates the SCL signal. The device is compliant with version 2.1 of the I^{2}C specification.

The I^{2}C interface on the device is a protocol translator from external I^{2}C transactions to internal SPI transactions. This explains the slightly increased protocol complexity described in the paragraphs that follow.

Read Transactions. The device registers are accessible when the EESEL bit is 0. On ZL30261 and ZL30263 the internal EEPROM memory is accessible when the EESEL bit is 1. On ZL30260 and ZL30262 the EESEL bit must be set to 0. The bus master first does an I^{2}C write to the device. In this transaction three bytes are written: the SPI Read command (see Table 3), the upper byte of the register address, and the lower byte of the register address. The bus master then does an I^{2}C read. During each acknowledge (A) bit the device fetches data from the read address and then increments the read address. The device then transmits the data to the bus master during the next 8 SCL cycles. The bus master terminates the read with a not-acknowledge (NA) followed by a STOP condition (P). See Figure 9. After the I^{2}C write there can be unlimited idle time on the bus before the I^{2}C read, but the device cannot tolerate other I^{2}C bus traffic between the I^{2}C write and the I^{2}C read. Care must be taken to ensure that the I^{2}C read is the first command on the bus after the I^{2}C write to ensure the two-part read transaction happens correctly.

Register Write Transactions. The device registers are accessible when the EESEL bit is 0. The bus master does an I^{2}C write to the device. The first three bytes of this transaction are the SPI Write command (see Table 3), the upper byte of the register address, and the lower byte of the register address. Subsequent bytes are data bytes to be written. After each data byte is received, the device writes the byte to the write address and then increments the write address. The bus master terminates the write with a STOP condition (P). See Figure 10.

EEPROM Writes (ZL30261 and ZL30263 Only). The EEPROM memory is accessible when the EESEL bit is 1. The bus master first does an I^{2}C write to transmit the SPI Write Enable command (see Table 3) to the device. The bus master then does an I^{2}C write to transmit data to the device as described in the Register Write Transactions paragraph above. See Figure 11.

EEPROM Read Status (ZL30261 and ZL30263 Only). The bus master first does an I^{2}C write to transmit the SPI Read Status command (see Table 3) to the device. The bus master then does an I^{2}C read to get the status byte. In this byte, the least significant bit is set to 1 if the transfer is still in progress and 0 if the transfer has completed. See Figure 12. Similar to read transactions described above, the I^{2}C write and the I^{2}C read cannot be separated by other I^{2}C bus traffic.

I^{2}C Features Not Supported by the Device. The I^{2}C specification has several optional features that are not supported by the device. These are: 3.4Mbit/s high-speed mode (Hs-mode), 10-bit device addressing, general call address, software reset, and device ID. The device does not hold SCL low to force the master to wait.

I^{2}C Slave Address. By default the upper 5 bits of the device's 7-bit slave address are fixed at 11101 and the lower 2 bits can be pin-configured for any of three values as shown in the table in section 5.2. For a device that can auto-configure from EEPROM at power-up, its I^{2}C slave address can be set to any value during auto-configuration at power-up by writing the the I2CA register as part of the configuration script.

Bit Order. The I^{2}C specification requires device address, register address and all data bytes to be transmitted most significant bit first on the SDA signal.

Note: as required by the I^{2}C specification, when power is removed from the device, the SDA and SCL pins are left floating so they don't obstruct the bus lines.