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

ZL30250, ZL30251

4-Input, 3-Output Any-to-Any Clock Multiplier and Frequency Synthesizer ICs

Data Sheet

April 2016

Features

- **Four Input Clocks**
  - One crystal/CMOS input
  - Two differential/CMOS inputs
  - One single-ended/CMOS input
  - Any input frequency from 9.72MHz to 1250MHz (9.72MHz to 300MHz for CMOS)
  - Clock selection by pin or register control
- **Low-Jitter Fractional-N APLL and 3 Outputs**
  - Any output frequency from <1Hz to 1035MHz
  - High-resolution fractional frequency conversion with 0ppm error
  - Easy-to-configure, encapsulated design requires no external VCXO or loop filter components
  - Each output has independent dividers
  - Output jitter as low as 0.16ps RMS (12kHz-20MHz integration band)
  - Outputs are CML or 2xCMOS, can interface to LVDS, LVPECL, HSTL, SSTL and HCSL
  - In 2xCMOS mode, the P and N pins can be different frequencies (e.g. 125MHz and 25MHz)
  - Per-output supply pin 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 (ZL30250) or internal (ZL30251) EEPROM; up to four configs, pin-selectable
  - SPI or I^2C processor Interface
  - Numerically controlled oscillator mode
  - Spread-spectrum modulation mode
  - Tiny 5x5mm QFN package
  - Easy-to-use evaluation software

Ordering Information

ZL30250LDG1	32 Pin QFN	Trays
ZL30250LDF1	32 Pin QFN	Tape and Reel
ZL30251LDG1	32 Pin QFN	Trays
ZL30251LDF1	32 Pin QFN	Tape and Reel

Matte Tin

Package size: 5 x 5 mm

-40°C to +85°C

Applications

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

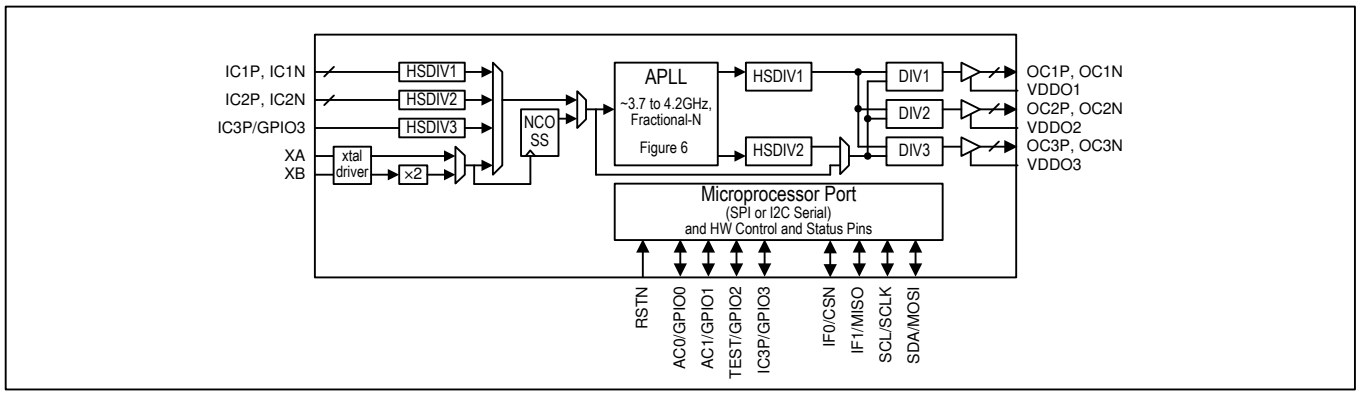

Figure 1 - Functional Block Diagram

Table of Contents

List of Figures

List of Tables

1. Application Examples

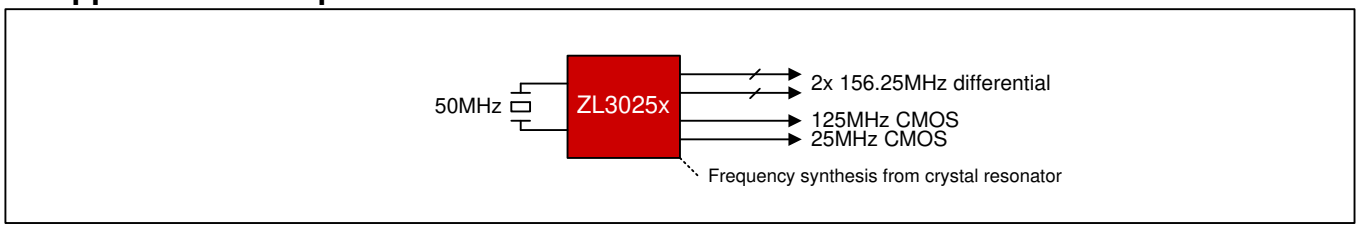

Figure 2 - Ethernet Frequency Synthesis Application

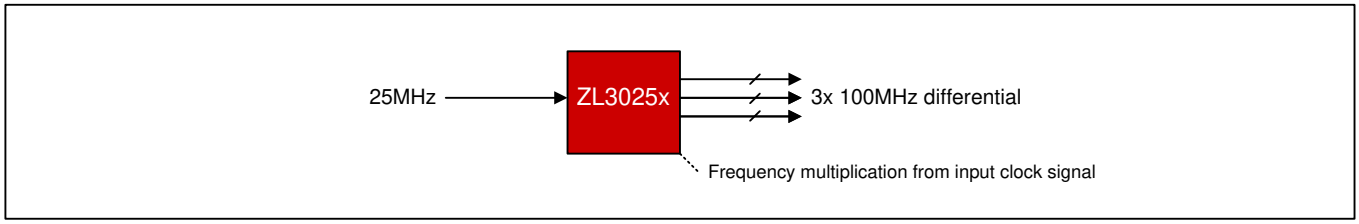

Figure 3 - PCI Express Frequency Multiplication Application

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 up to 1250MHz (differential) or 300MHz (CMOS)

2.2 APLL Features

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

2.3 Output Clock Features

- Three low-jitter output clocks
- Each output can be one differential output or two CMOS outputs
- Output clocks can be any frequency from 1Hz to 1035MHz (250MHz max for CMOS and HSTL outputs)
- Output jitter as low as 0.16ps RMS (12kHz to 20MHz integration band)
- In CMOS mode, an additional divider allows the OCxN pin to be an integer divisor of the OCxP pin (example: OC3P 125MHz, OC3N 25MHz)
- Outputs easily interface with CML, LVDS, LVPECL, HSTL, SSTL, 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 with high resolution and unlimited range
- Per-output enable/disable
- Per-output glitchless start/stop (stop high or low)

2.4 General Features

- SPI or I^2C serial microprocessor interface
- Automatic self-configuration at power-up from external (ZL30250) or internal (ZL30251) EEPROM memory; pin control to specify one of four stored configurations

- Numerically controlled oscillator (NCO) mode allows system software to steer frequency with resolution better than 0.01ppb
- Spread-spectrum modulation mode (meets PCI Express requirements)
- 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 ZL30250 or ZL30251 quick and easy
- Generates configuration scripts to be stored in external (ZL30250) or internal (ZL30251) EEPROM
- Generates full or partial configuration scripts to be run on a system processor
- Works with or without a ZL30250 or ZL30251 evaluation board

3. Pin Diagram

The device is packaged in a 5x5mm 32-pin QFN.

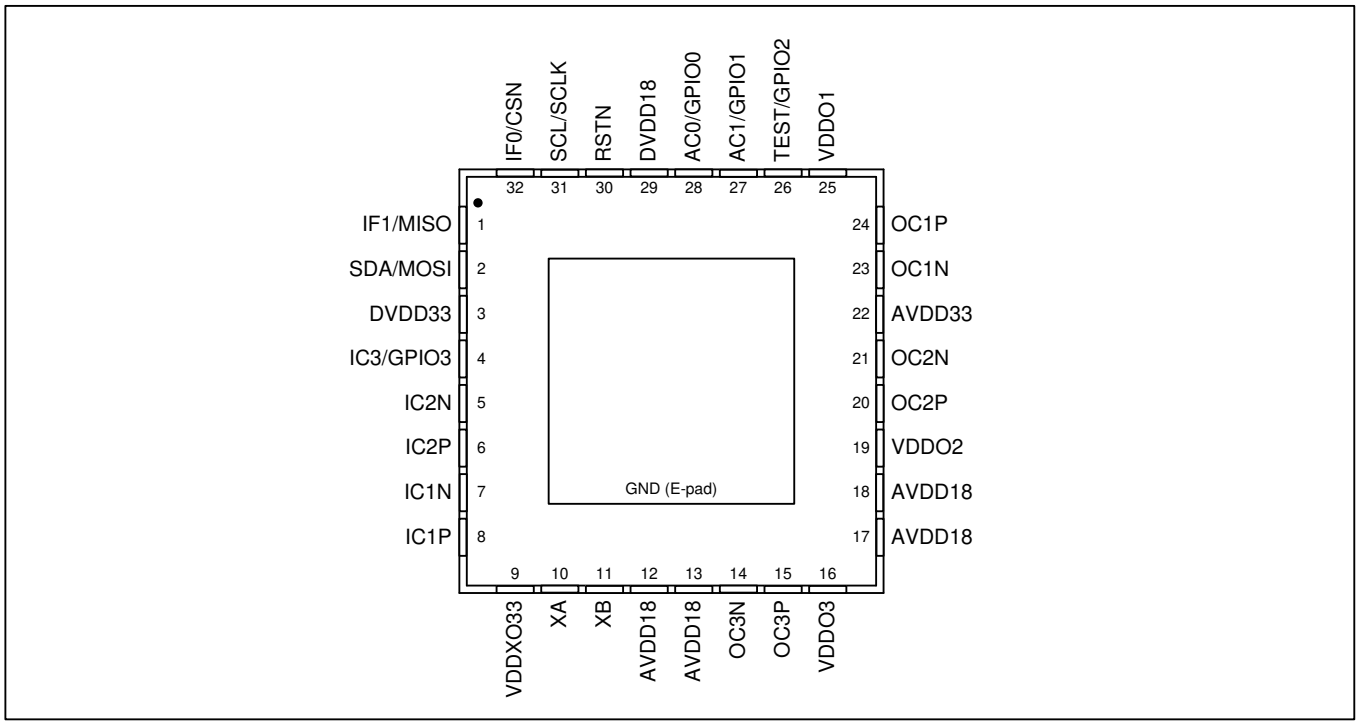

Figure 4 - Pin Diagram

4. Pin Descriptions

All device inputs and outputs are LVCMOS unless described otherwise. The Type column uses the following symbols: I – input, I_{PU} – input with 50kΩ internal pullup resistor, O – output, A – analog, P – power supply pin. All GPIO and SPI/I^2C interface pins have Schmitt-trigger inputs and have output drivers that can be disabled (high impedance).

Table 1 - Pin Descriptions

Pin #	Name	Type	Description
8 7 6 5 4	IC1P IC1N IC2P IC2N IC3P/GPIO3	I I I I I/O	**Input Clock Pins** Differential or Single-ended signal format. Programmable frequency. *Differential:* See Table 9 for electrical specifications, and see Figure 16 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 16, 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. If not needed as an input clock pin, IC3P can behave as general-purpose I/O pin GPIO3, which is configured by GPIOCR2. Its state is indicated in GPIOSR.
10 11	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.
24 23 20 21 15 14	OC1P OC1N OC2P OC2N OC3P OC3N	O	**Output Clock Pins** CML, HSTL or 1 or 2 CMOS. Programmable frequency and drive strength. See Table 10 and Figure 18 for electrical specifications and recommended external circuitry for interfacing to LVDS, LVPECL or CML input pins on neighboring devices. See Table 11 for electrical specifications for interfacing to CMOS and HSTL inputs on neighboring devices. See Figure 19 for recommended external circuitry for interfacing to HCSL inputs on neighboring devices.
30	RSTN	I_{PU}	**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. See section 5.10.

Table 1 - Pin Descriptions (continued)

Pin #	Name	Type	Description
28 27	AC0/GPIO0 AC1/GPIO1	I/O	**Auto-Configure [1:0] / General Purpose I/O 0 and 1** *Auto Configure:* On the rising edge of RSTN these pins behave as AC[1:0] and specify one of the configurations stored in EEPROM. See section 5.2. *General-Purpose I/O:* After reset these pins are GPIO0 and GPIO1. GPIOCR1 configures the pins. Their states are indicated in GPIOSR.
26	TEST/GPIO2	I/O	**Factory Test / General Purpose I/O 2** *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 GPIO2. GPIOCR2 configures the pin. It state is indicated in GPIOSR.
32	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 (ZL30250 only), the device asserts CSN to access an external SPI EEPROM during auto-configuration.
31	SCL/SCLK	I/O	**I^2C Clock / SPI Clock** *I^2C Clock:* When the device is configured as an I^2C slave, an external I^2C master must provide the I^2C 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(ZL30250 only), the device drives SCLK as an output to clock accesses to an external SPI EEPROM during auto-configuration.
1	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 (ZL30250 only), the device receives data on MISO from an external SPI EEPROM during auto-configuration. Note: On rev A parts, in I^2C interface mode this pin toggles between driving low and high-impedance during register accesses. Therefore in I^2C mode this pin must not be wired directly to VDD. To implement a static high value on IF1/MISO, wire it to VDD through a resistor (approximately 10kΩ recommended).
2	SDA/MOSI	I/O	**I^2C Data / SPI Master-Out-Slave-In** *I^2C Data:* When the device is configured as an I^2C slave, SDA is the bidirectional data line between the device and an external I^2C master. *SPI MOSI:* When the device is configured as a SPI slave, an external SPI

Pin #	Name	Type	Description
			master sends commands, addresses and data to the device on MOSI. When the device is configured as a SPI master (ZL30250 only), the device sends commands, addresses and data on MOSI to an external SPI EEPROM during auto-configuration.

Table 1 - Pin Descriptions (continued)

Pin #	Name	Type	Description
12 13 17 18	AVDD18	P	**Analog Power Supply.** 1.8V ±5%.
22	AVDD33	P	**Analog Power Supply.** 3.3V ±5%.
29	DVDD18	P	**Digital Power Supply.** 1.8V ±5%.
3	DVDD33	P	**Digital Power Supply.** 3.3V ±5%.
25	VDDO1	P	**Output OC1 Power Supply.** 1.5V to 3.3V ±5%.
19	VDDO2	P	**Output OC2 Power Supply.** 1.5V to 3.3V ±5%.
16	VDDO3	P	**Output OC3 Power Supply.** 1.5V to 3.3V ±5%.
9	VDDXO33	P	**Analog Power Supply for Crystal Driver Circuitry.** 3.3V ±5%.
E-pad	VSS	P	**Ground.** 0 Volts.

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 five device pins: TEST/GPIO2, AC1/GPIO1, AC0/GPIO0, IF1/MISO and IF0/CSN. For each of these pins, the first name (TEST, 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 5.10.

The hardware configuration pins are grouped into three sets:

1. TEST - Manufacturing test mode
2. IF[1:0] – Microprocessor interface mode and I^2C address
3. AC[1:0] – Auto-configuration from EEPROM

The TEST pin selects manufacturing test modes when TEST=1 (the AC[1:0] pins specify the test mode). For ZL30251, TEST=1 and AC[1:0]=00 configures the part so that production SPI EEPROM programmers can program the internal EEPROM (see section 5.12.2). For more information about auto-configuration from EEPROM see section 5.12.1.

5.2.1 ZL30250—External EEPROM or No EEPROM

For the ZL30250 the IF[1:0] pins specify the processor interface mode and the I^2C slave address. When IF[1:0]=11 (SPI) two options are available:

If AC[1:0]=00 the device sets up its processor interface as SPI slave through which it can be configured by software running on the SPI master. In this option the device cannot auto-configure from an external EEPROM.

If AC[1:0]=01, 10, or 11 the device first sets up its processor interface as a SPI master. It then auto-configures itself by reading the configuration number specified by AC[1:0] from an external SPI EEPROM connected to its SPI pins. After auto-configuration is complete, the device reconfigures its processor interface to be SPI slave.

These options are summarized in the following table:

IF1	IF0	AC1	AC0	Processor Interface	External EEPROM	Auto Configuration
0	0	0	0	I^2C, slave address 11011 00	No	n/a
0	1	0	0	I^2C, slave address 11011 01	No	n/a
1	0	0	0	I^2C, slave address 11011 10	No	n/a
1	1	0	0	SPI Slave	No	n/a
1	1	0	1	SPI Master to external EEPROM for auto-configuration then SPI Slave	Yes	Configuration 1
1	1	1	0		Yes	Configuration 2
1	1	1	1		Yes	Configuration 3

Notes about the device auto-configuring from external EEPROM:

1. The device's CSN pin must 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.

5.2.2 ZL30251—Internal EEPROM

For the ZL30251 the IF[1:0] pins specify the processor interface mode and the I^2C slave address. The AC[1:0] pins specify which of four device configurations in the EEPROM to execute after reset.

IF1	IF0	Processor Interface
0	0	I^2C, slave address 11011 00
0	1	I^2C, slave address 11011 01
1	0	I^2C, slave address 11011 10
1	1	SPI Slave

AC1	AC0	Auto Configuration
0	0	Configuration 0
0	1	Configuration 1
1	0	Configuration 2
1	1	Configuration 3

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.

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. When the device is configured for NCO or spread-spectrum operation, the DFREQZ parameter can be used to compensate for oscillator frequency error. When the device is configured for APLL-only mode, the APLL's fractional feedback divider value (AFBDIV) can be adjusted by ppb or ppm to compensate for oscillator frequency error.

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.

See Figure 5 for the crystal equivalent circuit and the recommended external capacitor connections. To achieve a crystal load (C_L) of 10pF, an external 16pF is placed in parallel with the 4pF internal capacitance of the XA pin, and an external 16pF is placed in parallel with the 4pF internal capacitance of the XB pin. The crystal then sees a load of 20pF in series with 20pF, which is 10pF total load. Note that the 16pF capacitance values in Figure 5 include all capacitance on those nodes. If, for example, PCB trace capacitance between crystal pin and IC pin is 2pF then 14pF capacitors should be used to make 16pF total.

The crystal, traces, and two external capacitors 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 sensitivies 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.

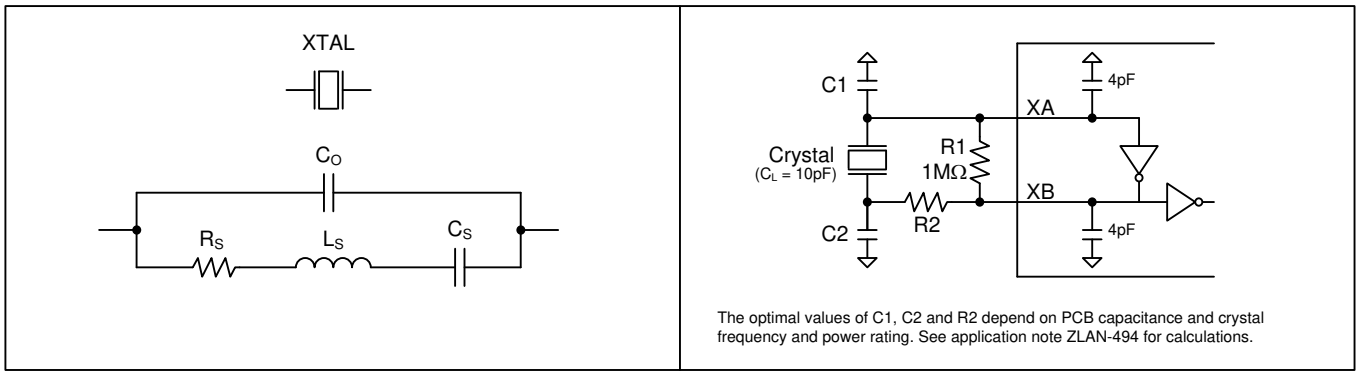

Figure 5 - Crystal Equivalent Circuit / Recommended Crystal Circuit

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		C_L		10		pF
Equivalent series resistance	f_{OSC} < 40MHz	R_S			60	Ω

$(ESR)^2$	f_{OSC} > 40MHz	R_S			50	Ω
Maximum crystal drive level			100			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.

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 Doubler

Figure 1 shows an optional clock doubler (“x2” block) following the crystal driver block. The doubler, which is enabled by setting MCR1.DBL=1, can be used to double the frequency of the internal crystal driver circuit or an clock signal on the XA pin. The following table shows scenarios when the clock doubler can be used.

Device Mode	With Crystal	With XO or Clock Signal
APLL, Integer Multiply	Maybe[1]	Maybe[1]
APLL, Fractional Multiply	Yes	Yes
NCO	Yes	Yes
Spread-Spectrum	Yes	Yes

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 large spur at the XA frequency (but this spur may be outside the band of interest for the application).

5.3.4 Ring Oscillator (for System Start-Up)

To ensure that registers can be written immediately after system start-up, in its power-on reset state the device operates its registers and processor interface from an internal ring oscillator.

When operating the device in NCO mode or spread spectrum mode, as soon as the external oscillator connected to the XA pin has stabilized and is ready to use, the MCR1.MCSEL bit must be set to source the NCO/SS master clock from XA. If the ring oscillator causes undesirable spurs it can be disabled (powered down) by setting MCR1.ROSCD=1.

When operating the part in APLL-only mode, a master clock signal on the XA pin is not required, and the ring oscillator is left enabled to provide a clock for the processor interface logic and registers.

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 16). 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. If not needed as an input clock pin, IC3P can behave as general-purpose I/O pin GPIO3.

5.5 Numerically Controlled Oscillator / Spread Spectrum Block (NCO/SS)

The NCO/SS block allows the device to behave as a numerically controlled oscillator and optionally perform spread-spectrum frequency modulation. This block is enabled by setting PLLEN.NCSSEN=1 and is the input reference for the APLL when APLLCR3.APLLMUX=11x. The NCO/SS block requires a clock signal from an external oscillator connected to the XA pin (see section 5.3.1). Table 8 shows the allowable frequency range for the XA signal. Note that device output jitter increases as XA frequency decreases.

5.5.1 Numerically Controlled Oscillator (NCO) Mode

The NCO/SS block operates in NCO mode when NCSSCR1.MODE=0001. In this mode system software controls the NCO/SS block by controlling the value of the 40-bit FREQZ field in the DFREQZ registers. The resolution of frequency control is better than 0.01ppb.

The nominal FREQZ value, hereafter referred to as FREQZ0, is computed by the evaluation software for the desired device configuration When the FREQZ field is set to the FREQZ0 value, the device's output clock frequencies have a fractional frequency offset of zero with respect to the NCO master clock signal applied to the XA pin.

(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 FREQZ0 value from the device. FREQZ0 is a 40-bit unsigned integer.

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

newFREQZ = round(FREQZ0 * (1 + FFO/1e6))

System software then writes the newFREQZ value directly to the FREQZ field in the DFREQZ registers.

Note that any subsequent frequency changes are calculated using the same equation from the original FREQZ0 value and are not a function of the previous newFREQZ value. The value of newFREQZ should be kept within ±1000ppm of FREQZ0 and within ±500ppm of the previous newFREQZ 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 FREQZ0 to avoid causing the APLL to lose lock.

5.5.2 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.

The NCO/SS block operates in spread spectrum mode when NCSSCR1.MODE=0010. In this mode the NCO/SS block performs frequency modulation starting from a base frequency offset specified in the 40-bit FREQZ field in the DFREQZ registers. 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 nominal value of FREQZ and the spread configuration register values are determined by the evaluation software.

Down-spread applications can be supported by converting them into center-spread. This is done by setting the NCO/SS block's center frequency to be the center of the modulation range rather than the high end of range. For example, 100MHz with -1% downspread can be converted into ±0.5% center spread with center frequency of 100MHz/1.005=99.502488MHz.

In PCI Express applications the device can be used as a "point of load" spread-spectrum generator. In such an application, the 100MHz PCI Express clock signal without SSM can be generated centrally and distributed to

various points in the system. A device positioned at one of those points can accept the 100MHz signal on its XA pin and generate multiple 100MHz signals on its outputs. System software can then choose to enable or disable SSM in the device as needed to suit the needs of the application.

5.6 APLL Configuration

5.6.1 APLL Input Selection and Frequency

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 APLL can also lock to the output of the NCO/SS block (see section 5.5).

The input to the APLL can be controlled by a GPIO pin or by the APLLCR3.APLLMUX register field. When APLLCR3.EXTSW=0, the APLLCR3.APLLMUX register field controls the APLL input mux.

When APLLCR3.EXTSW=1, a GPIO pin controls the APLL input mux. When the GPIO pin is low, the mux selects the input specified by APLLCR3.APLLMUX. When the GPIO pin is high, the mux selects the input specified by APLLCR3.ALTMUX. MCR2.EXTSS specifies which GPIO pin controls this behavior.

In APLL-only mode, the frequencies of all enabled input clocks (ICx and XA) 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. The XA pin does not have an internal divider, and, therefore, if XA is an enabled input clock then the XA frequency sets the APLL common PFD frequency. The polarity of an ICx input signal can be inverted by setting ICxCR1.POL.

5.6.2 APLL Output Frequency

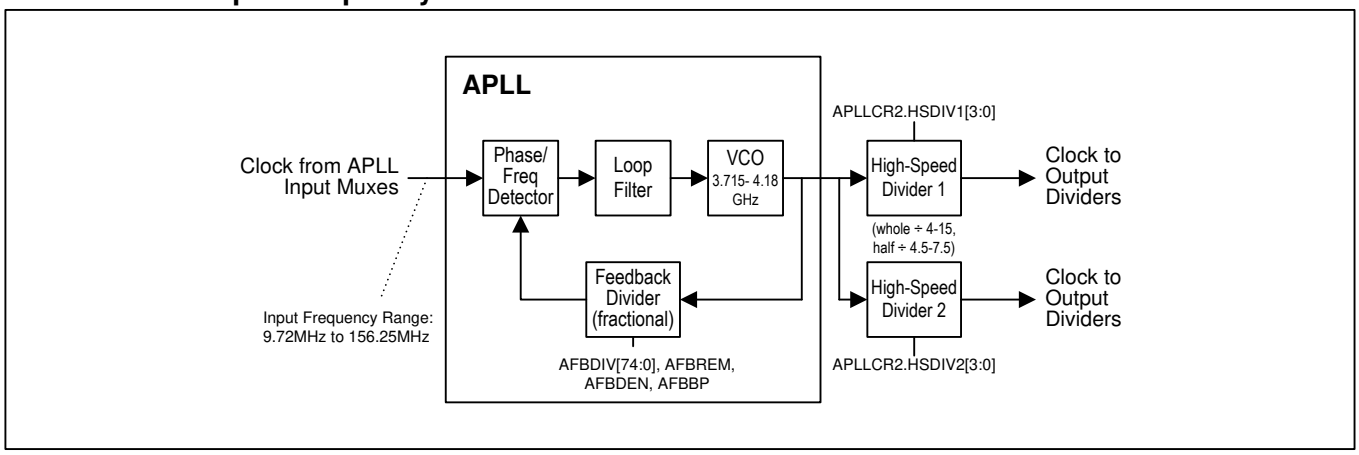

Figure 6 - APLL Block Diagram

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 6 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 APLLCR2.HSDIV1 and HSDIV2 fields specify how the VCO frequency is divided down by the high-speed dividers. Dividing by six is the typical setting to produce 622.08MHz for SDH/SONET or 625MHz for Ethernet applications.

Internally, the exact APLL feedback divider value is expressed in the form AFBDIV + AFBREM / AFBDEN * $2^{-(33-AFBBP)}$. 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 12). 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.

$$AFBDIV = trunc(M * N / D * 2^{24}) \quad (1)$$

$$lsb\_fraction = M * N / D * 2^{24} - AFBDIV \quad (2)$$

$$AFBDEN = D \quad (3)$$

$$AFBREM = round(lsb\_fraction * AFBDEN) \quad (4)$$

$$AFBBP = 33 - 24 = 9 \quad (5)$$

The trunc() function returns only the integer portion of the number. The round() function rounds the number to the nearest integer. In Equation (1), 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. Finally, in Equation (5) AFBBP is set to 33 – 24 = 9 to correspond with AFBDIV having 24 fractional bits.

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 ZL3025x 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 and ALK2.

5.6.3 APLL Phase Adjustment

The phase of the APLL's output clock can be incremented or decremented by 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 APLLCR4.PDSS field specifies the phase decrement control signal, which can be the APLLCR4.DECPH bit or any of the four GPIOs. The APLLCR4.PISS field specifies the phase increment control signal, which can be the APLLCR4.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 both high-speed dividers.

5.7 Output Clock Configuration

The device has three output clock signal pairs. 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 six 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 relative to an input signal, and the phases of output signals can be adjusted dynamically with high resolution and infinite range.

5.7.1 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 a CML output, an HSTL output, 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. In CML mode the normal 800mV V_{OD} differential voltage is available as well as a half-swing 400mV V_{OD}. 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.

Each output has its own power supply pin to allow CMOS or HSTL signal swing from 1.5V to 3.3V for glueless interfacing to neighboring components. 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 differential (CML) outputs must have a power supply of 3.3V.

The differential outputs can be easily interfaced to LVDS, LVPECL, CML, HCSL, HSTL and other differential inputs on neighboring ICs using a few external passive components. See Figure 18 for examples.

5.7.2 Output Frequency Configuration

The frequency of each output is determined by the configuration of the APLL, the high-speed dividers and the per-output dividers. Each output can be connected to either high-speed divider 1 (HSDIV1) or 2 (HSDIV2) using the OCxCR3.DIVSEL field.

Each output has two output dividers, a 7-bit medium-speed divider (OCxCR1.MSDIV) and a 25-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 HSDIV output frequency as a common multiple. For example, for Ethernet clocks, a 625MHz HSDIV 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 HSDIV 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 notest 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 high-speed divider output frequency divided by 128.

- An additional constraint is that the medium-speed divider must be configured to divide by 6 or more (i.e. must have OCxCR1.MSDIV≥5).

5.7.3 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.

5.7.4 Output Phase Adjustment and Phase Alignment

The device has flexible, high-resolution tools for managing the phases of the output clocks relative to one another. The key register fields for this are found in the PACR1 and PACR2 global configuration registers and the per-output OCxPH register.

Phase alignment and phase adjustment are done in the medium-speed dividers. Resoution is 0.5 periods (also known as unit intervals or UI) of the high-speed divider (HSDIV) output clock. For example, for an HSDIV output frequency of 800MHz, resolution is 625ps.

5.7.4.1 Phase Adjustment

A phase adjustment is a phase change for an output relative to that output's most recent phase. To cause the device to perform phase adjustment of an output clock, set PACR1.MODE=1, set OCxCR1.PHEN=1 to enable the output for phase adjustment, and write the phase adjustment amount to the output's OCxPH register. Then an arm/trigger methodology is used to cause the phase adjustment to happen.

The arm step tells the device that it is enabled to perform the phase adjustment when it sees the trigger stimulus. The source of the arm signal is specified by PACR2.ARMSRC. Options include the 0-to-1 transition of the PACR1.ARM bit, APLL transition from unlocked to locked, or a transition on one of the GPIO pins.

The source of the trigger signal is specified by PACR2.TRGSRC. Options include 0-to-1 transition of the PACR1.TRIG bit, APLL transition from unlocked to locked, or a transition on one of the GPIO pins. The trigger signal can be inverted by setting PACR1.TINV. With TINV=1, the same GPIO signal can arm on one edge and trigger on the opposite edge.

Any combination of outputs can be phase adjusted by the same trigger, and each output can be adjusted by a different amount. Only outputs with OCxCR1.PHEN=1 and OCxPH.PHADJ≠0 have their phases adjusted.

There are a few constraints on the range of possible phase adjustments. These have to do with the output's medium-speed divider value.

1) Phase adjustment is not available unless OCxCR1.MSDIV>0.
2) The largest negative phase adjustment magnitude in HSDIV periods is:
   If OCxCR1.MSDIV is odd: (OCxCR1.MSDIV – 1) / 2
   If OCxCR1.MSDIV is even: (OCxCR1.MSDIV – 2) / 2
3) The largest positive phase adjustment in HSDIV periods is:
   If OCxCR1.MSDIV is odd: (127 – OCxCR1.MSDIV) / 2
   If OCxCR1.MSDIV is even: (128 – OCxCR1.MSDIV) / 2

The implications of constraints 2) and 3) are shown in this table:

OCxCR1.MSDIV	**Largest Negative Phase Adjust,** HSDIV periods	**Largest Positive Phase Adjust,** HSDIV periods	**Notes**
1 or 2	0	63	no negative adjustment
3 or 4	1	62	
5 or 6	2	61	
…	…	…	
123 or 124	61	2	
125 or 126	62	1	
127	63	0	no positive adjustment

During a phase adjustment the MSDIV output period is changed for one period. The MSDIV output signal during that period will have longer high time (unless inverted) during a positive phase adjustment and shorter high time (unless inverted) during a negative phase adjustment. With negative phase adjustments care must be taken to not shorten the high time of the output clock signal to be too short for the components that receive the clock. There are several possible ways to avoid this issue including: (1) using small negative adjustments such as -0.5UI repeatedly instead of one larger negative adjustment, (2) using positive adjustments to “wrap around” to the desired negative adjustment, or (3) holding the components that receive the clock in reset during the phase adjustment.

An armed phase adjustment can be canceled before the trigger occurs by setting the PACR1.RST bit.

The PASR register has real-time status bits indicating whether a phase adjustment is armed and waiting for a trigger (ARMED bit) or in progress (BUSY bit). It also has a latched status bit (ADJL bit) to indicate the adjustment has completed.

Example: +1.0 HSDIV period phase adjustment for output OC1 using ARM and TRIG register bits:

```
OC1CR1.PHEN=1          (Enable phase adjust on OC1)
OC1PH.PHADJ=00000010   (Specify +1.0 HSDIV period phase adjustment)
PACR1.MODE=1           (Phase adjustment mode)
PACR2.ARMSRC=0001      (arm signal is PACR1.ARM bit)
PACR2.TRGSRC=0000      (trigger signal is PACR1.TRIG bit)
PACR1.RST=1            (reset phase adjust/align state machine after changing ARMSRC)
PACR1.ARM=1            (arm for phase adjust)
PACR1.TRIG=1           (do the phase adjust: add +1.0 UI to output phase)
repeat the next two writes as needed:
PACR1.ARM=1 .TRIG=0    (arm again; clearing the TRIG bit is required when MSDIV period < master
                        clock period because TRIG is not self-clearing in this situation)
PACR1.TRIG=1           (add +1.0 UI to output phase again)
```

5.7.4.2 Phase Alignment, Output-to-Output

A phase alignment is a special case of phase adjustment where the MSDIV and LSDIV dividers for all participating outputs are reset just before the phase adjustment occurs. For output-to-output alignment the trigger can be the PACR1.TRIG bit or the APLL lock signal.

To avoid glitches (i.e. “runt pulses”) on the output clock it is possible to manually stop the output(s), before triggering the phase alignment, and then restart the output(s) after the alignment (See section 5.7.5).

When aligning outputs, it is important to note that, by default, the phase of outputs configured as HSTL format or “two CMOS, OCxP inverted vs. OCxN” format is opposite that of CML outputs. For example, consider the case where OC1 is 100MHz CML format and OC2 is 100MHz HSTL format. When OC1 and OC2 are aligned then OC2N is high when OC1P is high. The polarity bit OCxCR2.POL can be used to change this as needed.

There are several rules when alignment is enabled for multiple outputs:

- All participating outputs must come from the same high-speed divider
- All outputs that use both medium-speed and low-speed divider must have the same MSDIV value, the same LSDIV value and PHADJ=0. Subsequent phase adjustment(s) can be used to move the output(s) to other phase(s).
- All outputs that only use medium-speed divider can have PHADJ values smaller than the period of the highest output frequency among them.
- When some outputs use only medium-speed divider and other outputs use both medium-speed and low-speed divider, all MSDIV values must be the same, and those output using low-speed divider must have PHADJ=0.

Contact Microsemi Timing Applications Support for help with alignment scenarios that don’t meet the rules listed above.

Example: OC1-to-OC2 alignment (+3.5 HSDIV UI offset) after the APLL locks:

```
OC1CR1.PHEN=1            (Enable phase adjust on OC1)
OC2CR1.PHEN=1            (Enable phase adjust on OC2)
OC1PH.PHADJ=00000000     (0.0UI)
OC2PH.PHADJ=00000111     (+3.5UI)
PACR1.MODE=0             (Phase alignment mode)
PACR2.ARMSRC=0001        (arm signal is PACR1.ARM bit)
PACR2.TRGSRC=0001        (trigger signal is APLL transition from unlocked to locked)
PACR1.RST=1              (reset phase adjust/align state machine after changing ARMSRC, TRGSRC)
PACR1.ARM=1              (arm for phase alignment)
 (Aligns/realigns outputs when the APLL locks or relocks)
```

5.7.4.3 Phase Alignment, Input-to-Output

The phase alignment tool described in section 5.7.4.2 can use a GPIO pin as the alignment trigger. However there is some uncertainty associated with sampling the GPIO signal. Therefore the phase alignment tool *by itself* is is not sufficient to achieve input-to-output phase alignment. The procedure is to first do a phase alignment as described in section 5.7.4.2 but with a GPIO input as the trigger. Then the phase measurement tool described in section 5.7.6 can be used to determine the phase difference between an output signal and the input signal. Then phase adjustment as described in section 5.7.4.1 can be used to change the phase of one or more output signals to align with input signal phase.

It is important to note that, by default, outputs that only use the medium-speed divider have their rising edge aligned with the rising edge of the trigger signal. Meanwhile,outputs that use both the medium-speed and low-speed dividers have their rising edge aligned with the falling edge of the trigger signal. Per-output polarity bits (OCxCR2.POL) can be used to invert the polarity of output signals as needed so that all are rising-edge aligned or falling-edge aligned or any combination as needed.

5.7.5 Output Clock Start and Stop

Output clocks can be stopped high or low. 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, stops low, or or does not stop. The OCxSTOP.SRC field specifies the source of the stop signal. Options include the OCxSTOP.STOP bit, assertion of one of the GPIO pins, and the arming of a phase adjustment (which is indicate by PASR.ARMED).

When the stop mode is Stop High (OCxSTOP.MODE=01) 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=10) and the stop signal is asserted, the output clock is stopped after the next falling edge of the output clock. 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.

OCxCR1.MSDIV must be > 0 for this function to operate since MSDIV=0 bypasses the start-stop circuits. Note that when OCxCR3.NEGLSD=1 the start-stop logic is bypassed for the OCxN pin, and OCxN may not start/stop without glitches.

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

When OCxCR2.STOPDIS=1 the output driver is disabled (high impedance) while the output clock is stopped.

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.

5.7.6 A-to-B Phase Offset Measurement

The phase or time offset between two signals (A and B) can be measured in units of a timebase clock. This capability can be used to for several purposes, including:

- Keeping output clocks and low-speed output phase/time signals—such as frame sync, multiframe sync, or 1 pulse per second (1PPS) signals—aligned with input phase/time signals. The A-to-B measurement circuitry can detect phase changes in the input signal. Then the output phase adjustment circuitry described in section 5.7.4 can be used to move phase(s) of output(s) to follow the input phase change.

- Keeping output clock signals and/or low-speed output phase/time signals aligned with one another. The A-to-B measurement circuitry can detect relative phase changes, and the phase adjustment circuitry described in section 5.7.4 can be used to move phase(s) of output(s) as needed.

The A and B signals can be any ICx input, any OCx output, or any GPIO, as specified by MABCR2.ASRC and MABCR3.BSRC. The timebase signal can be the external oscillator signal (or the output of the crystal driver circuit, optionally doubled by the clock doubler) or the output clock of any of the three medium-speed dividers (MSDIV1, MSDIV2, MSDIV3). The timebase signal is specified by MABCR1.TBSRC.

A new measurement is started by writing MABCR1.START=1. Any previously started measurement must be completed before a new measurement is started. If a measurement has not finished it can be aborted by writing MABCR1.RST=1 before starting a new measurement. The measurement is complete when MABSR1.RDYL is set.

Example: consider an SDH/SONET application where OC1 is a 19.44MHz output clock and OC2 is an 8kHz frame sync signal. The goal is to measure the phase offset of OC1 vs. OC2. If they are found to have a phase offset then the phase adjustment circuitry in section 5.7.4 can be used to slowly change the phase of OC1 to match the phase of OC2 (or vice versa).

MABCR1.TBSRC=001	(MSDIV1 output clock is 311.04MHz = 3.2 ns period)
MABCR2.ASRC=10001	(OC2 8kHz sync signal)
MABCR3.BSRC=10000	(OC1 19.44MHz clock)
MABCR1.START=1	(Start measurement)
Wait for MABSR1.RDYL=1	(Measurement ready)

Read MABSR1.OVFL	(to see if the measurement is valid)
MABSR1.RDYL=1, MABSR1.OVFL=1	(clear latched status bits)
Read MEAS bits from MABSR1 and MABSR2	

If, for example, MEAS = 111 1111 1001 (-8) then the rising edge of OC1 (the 'B' signal) precedes the rising edge of OC2 (the 'A' signal) by 8 MSDIV1 ouptut clock periods (25.7ns).

An A-to-B measurement is performed by sampling the A and B signals with the selected timebase clock and detecting the rising or falling edges to measure. The number of timebase clocks between the A and B edges is counted. If the counter doesn't overflow then the phase difference is reported in the MEAS field in MABSR1 and MABSR2. If the counter does overflow then MABSR1.OVFL is set and the value of MEAS is invalid.

While the measurement is in progress the MABSR1.BUSY bit is set to 1. When the measurement is complete MABSR1.BUSY is set to 0 and MABSR1.RDYL is set to 1. Since the A and B signals are sampled by the timebase signal, this measurement tool is only useful when the timebase signal is much higher frequency than the A and B signals (at least 8-10x). Also, when possible, the timebase signal frequency should be less than or equal to 1000 times faster than the the frequencies of the A and B frequencies to avoid measurement counter overflow.

Constraints on A-to-B measurement:

- $f_B = f_A \times N$ where f_A is the frequency of signal A, f_B is the frequency of signal B and N is a positive integer

When measuring from an ICx input or a GPIO (signal A) to an ICx or a GPIO (signal B) and when measuring from an OCx output to an OCx output, the measured value is MEAS * timebase_period. This measurement has a variability of 0 to +1 timebase clock period.

When measuring from an ICx input or a GPIO (signal A) to an OCx output (signal B), the measurement in time units is MEAS * timebase_period + 6 * HSCLK_period, where HSCLK_period is the period of the output of the high-speed divider from which OCx signal is derived. This measurement has a variability of 0 to +1 timebase clock period plus 0 to +1 HSCLK periods.

When measuring from an OCx output (signal A) to an ICx input or a GPIO (signal B), the the measurement in time units is MEAS * timebase_period – 6 * HSCLK_period, where HSCLK_period is the period of the output of the high-speed divider from which OCx signal is derived. This measurement has a variability of 0 to +1 timebase clock period plus 0 to +1 HSCLK periods.

Guidance for Use

When the A and B signals are aligned to within one timebase clock cycle, the measurement hardware does not report 0. Instead it reports a measurement value that is equivalent to +1 cycle of signal B.

If the timebase clock is ≤ 1023 times faster than signal B (so that the MEAS field cannot overflow, unless signal B is grossly too slow or not toggling at all) then system software should check the measured phase value. If the measured value is equal to the period of signal B then the A and B signals are aligned.

If the timebase clock is 1024 to 2047 times faster than signal B (and therefore the measurement counter can overflow) then the measurement hardware reports overflow when the A and B signals are aligned to within one timebase clock cycle. This report of overflow can be distinguished from other overflow cases by setting MABCR3.BINV=1 and then remeasuring from signal A to the opposite edge of signal B. If the new measured value is equal to half the period of signal B then the A and B signals are aligned.

If the timebase clock is > 2047 times faster than signal B then the measurement hardware reports overflow when the A and B signals are aligned to within one timebase clock cycle. This report of overflow is not distinguishable from other overflow cases. One way system software could work around this to determine that A and B are aligned

is to use phase adjustment to move one of the signals by 2 or more timebase clocks then remeasure. If the new measured value matches the phase adjustment then the signals were aligned before the phase adjustment. Software can then adjust the phase of the signal back to its original position. Not all applications can tolerate such phase adjustments; for those applications it is recommended that the timebase clock be ≤ 2047 times faster than signal B.

5.8 Microprocessor Interface

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

In SPI mode the ZL30250 can be configured at reset to be a SPI slave to a processor master or a SPI master to an external EEPROM slave. The ZL30251 can only be configured as a SPI slave to a processor master. Both devices are always slaves on the I^2C bus.

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

5.8.1 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 a ZL30251 the internal EEPROM memory is accessible when EESEL=1. On a ZL30250 EESEL 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 7.

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 9.

EEPROM Writes (ZL30251 Only). The internal EEPROM memory is accessible when EESEL=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 8 and Figure 9.

EEPROM Read Status (ZL30251 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 ZL30251, 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 17 and Figure 20 for AC timing specifications for the SPI interface.

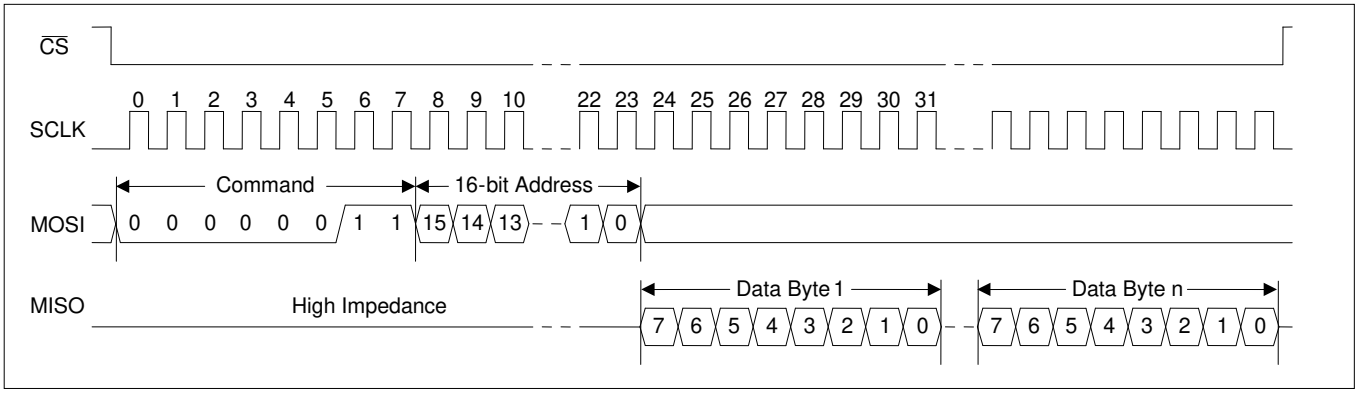

Figure 7 - SPI Read Transaction Functional Timing

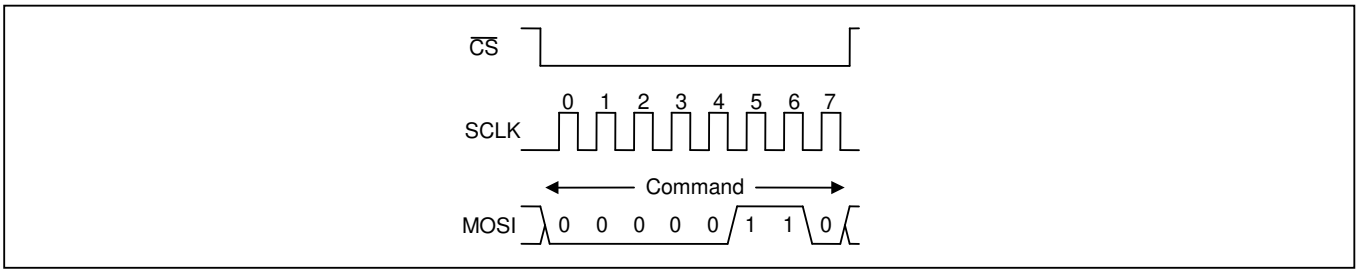

Figure 8 - SPI Write Enable Transaction Functional Timing (ZL30251 Only)

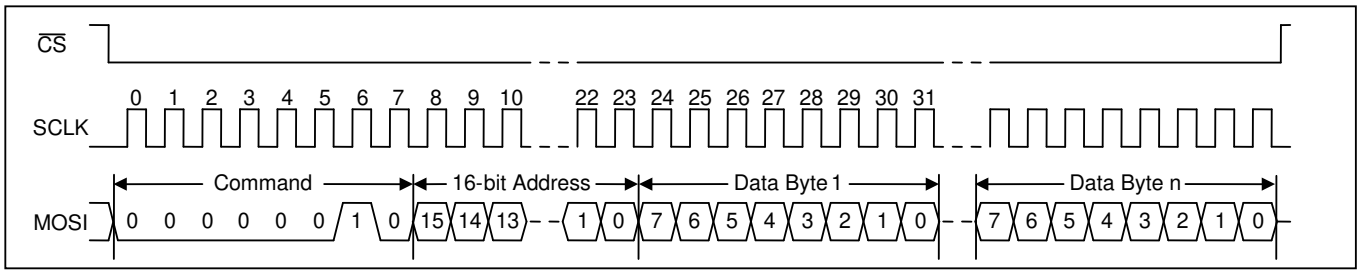

Figure 9 - SPI Write Transaction Functional Timing

5.8.2 SPI Master (ZL30250 Only)

After reset the ZL30250 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 7.

Writing the External EEPROM. Due to the small package size and low pin count of the device, there is no way to use the ZL30250 to write the external EEPROM. The auto-configuration data used by the ZL30250 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.