# AU5508: Quad PLL Frequency Translator / Jitter Attenuator / Network and Port Synchronizer

## General Description

AU5508 offers most integrated 5G clock solution. It Combines BBU, Wireline, data converter clocks and Serdes requirements in one part definition.

- Unique integration of Sync Features
  - Enables Systems with Low Constant Time Interval Error
  - IEEE1588/SyncE/1 PPS full support
- Lower jitter for higher data rate links
- Lower close in noise and JESD204B/C full support for Data Converter clocks
- Integrated feature rich single part offers unparalleled flexibility to the system designer

## Features

- Ultra-Performance PLLs
- Fully Integrated design with no external components
- 120 fs Typical RMS integrated jitter (12k-20M)
- 122.88M Output with excellent close in noise performance
- Fully Flexible Output and Input Mux: High level of flexibility in output allocation for PLLs
- JESD204B/C Support for data converter clocks
- 1 pps Input / Output Support with sub 20s lock time
- External EEPROM Support
- TDC Mode available on all input clocks to measure input delays with < 10 ps accuracy: 10 TDC Channels available (independent of the PLLs)
- Frequency Control DCO: DCO Control on all outputs (0.001 ppt)
- Phase Control DCO: Fine phase adjustment knob for phase of all outputs from a PLL (adjustment accuracy < 1ps) in both closed loop and open loop modes
- Internal modes to combine wander of OCXO with jitter of XO for holdover– Provides 24-hour holdover with programmable HO accuracy settings
- Best in class hitless switching performance: PBO with sub 25 ps hit, Phase Propagation & Frequency Ramp with programmable frequency/phase slopes
- Fully integrated Jitter and wander attenuation options down to 0.09 mHz
- Repeatable input to output delays with output relative delay adjust
- Internal ZDB Mode with < 0.5 ns Input to Output delay variation, independently available for each PLL
- Outputs can be phase aligned an independent sync pulse
- 72 QFN 10mm X 10mm Package

| Product Family | Inputs /Outputs | Input Freq | Output Frequency | RMS Jitter | Packages |
|---|---|---|---|---|---|
| 55xx | 5 Diff / 10 SE Inputs<br>Up to 12 Diff / 24 SE Outputs | 0.5 Hz - 2.1 GHz | 0.5 Hz - 2.94912 GHz | ~ 120 fs typ | 72 QFN |

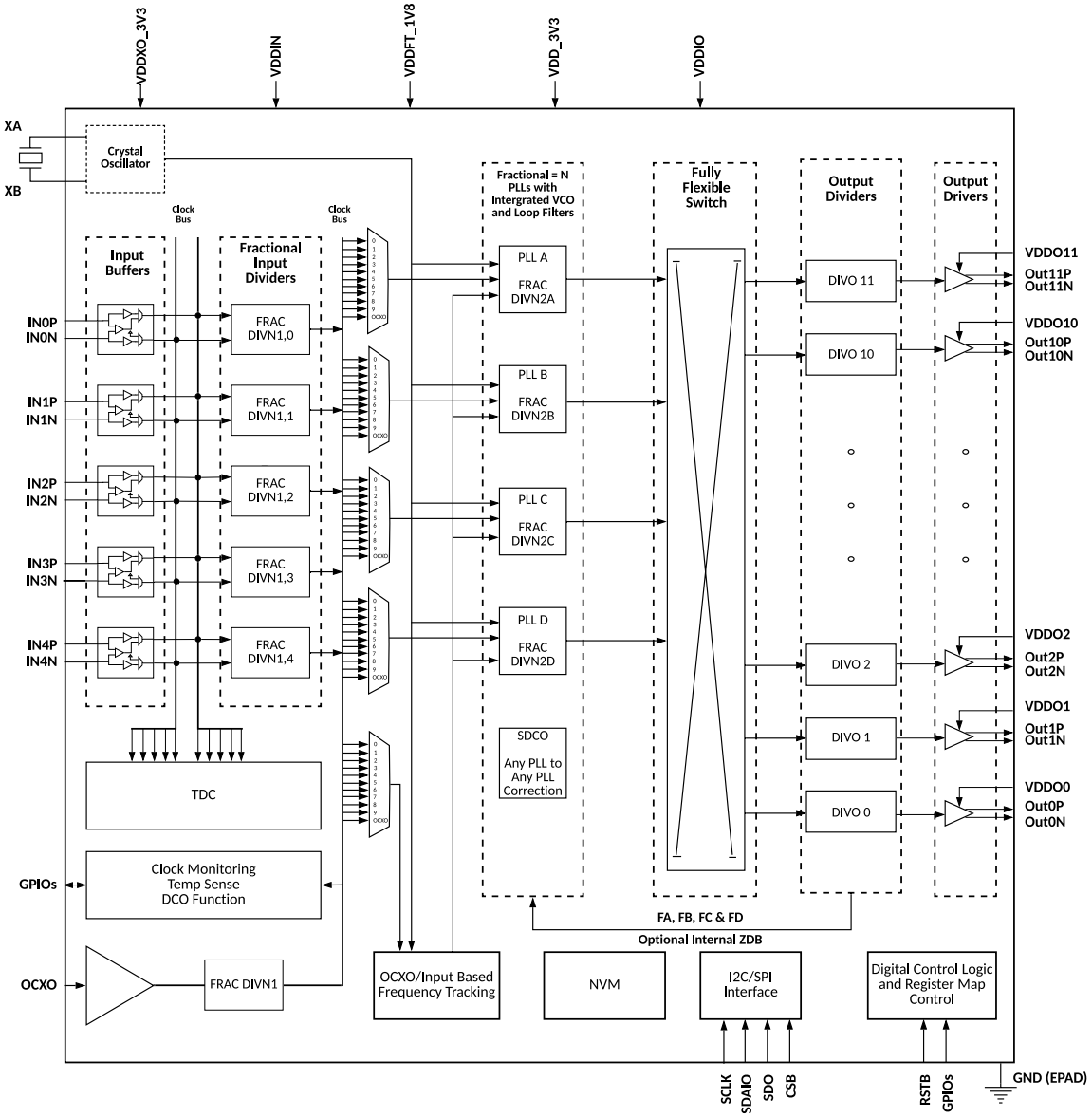



**Figure 1 Functional Overview**

# Table of Contents

## List of Tables

## List of Figures

# 1 Pin Description

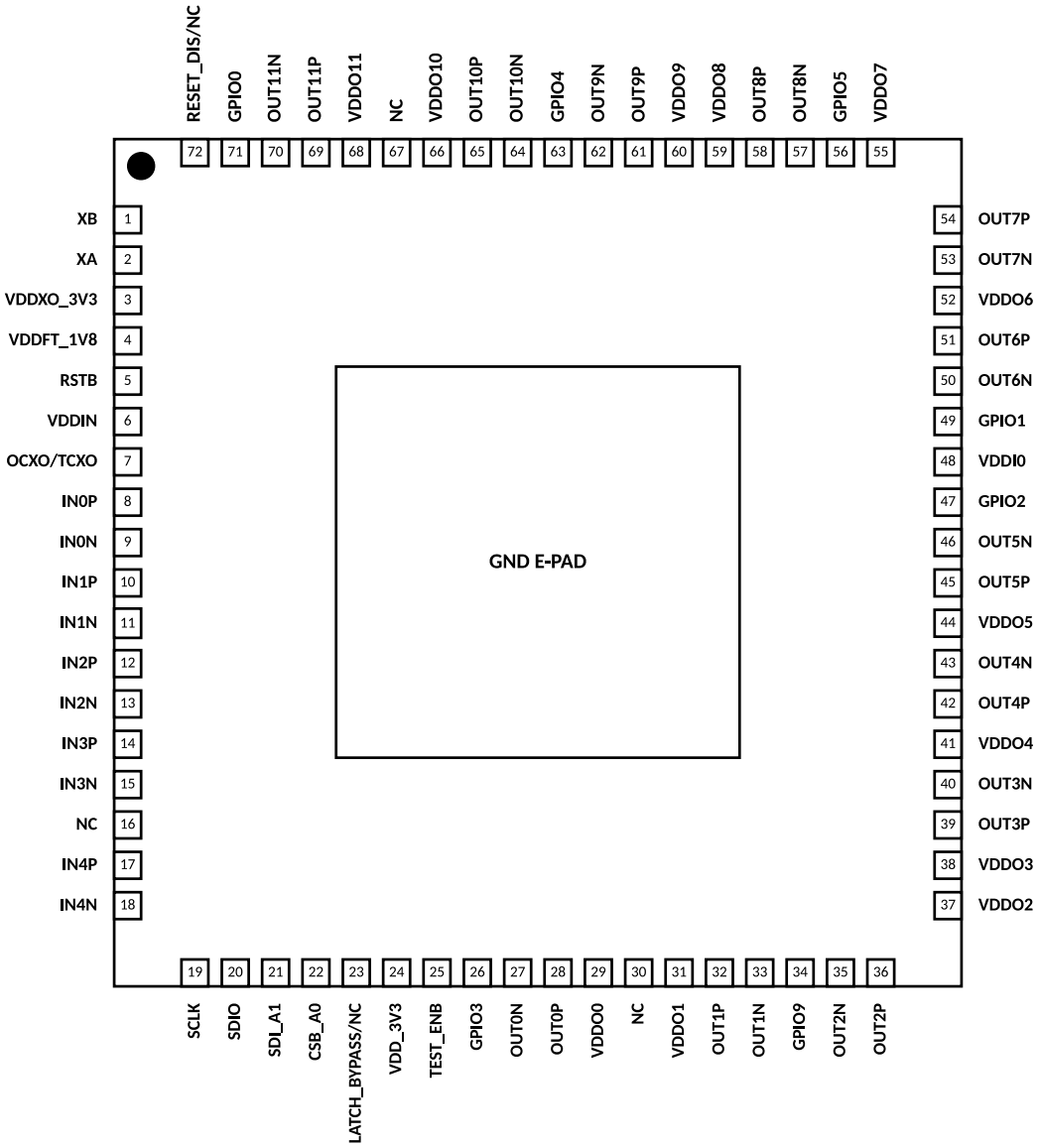



**Figure 2 AU5508 Pin Configuration**

**Table 1 Pin Description**

| Name | I/O Type | Pin. No | Voltage Level | Default Pull Up/ Down | Description |
|---|---|---|---|---|---|
| XB | Output | 1 | | | Crystal Output. Crystal connects to this pin. Leave unconnected if External oscillator is connected to XA. |
| XA | Input | 2 | < 1.8 V | | Crystal Input. External reference clock input. |
| VDDXO_3V3 | Power | 3 | 2.375 V – 3.6 V | | Analog Power supply for Crystal Oscillator |
| VDDFT_1V8 | Power | 4 | 1.71 V – 2 V | | Analog Power Supply for Frequency tracking PLL, Input TDC. |
| RSTB | Reset | 5 | VDDIO (Pin 48) | Pull up | Reset Input. RSTB should be held low and released from low to high ONLY when all supplies to the chip have crossed 90% of their final value. |
| VDDIN | Power | 6 | 1.71 V/2.5 V/ 3.6 V | | Analog Power Supply for Input buffers and dividers as well as Clock Monitors |

| Name | I/O Type | Pin. No | Voltage Level | Default Pull Up/ Down | Description |
|---|---|---|---|---|---|
| OCXO/TCXO | Input | 7 | -0.5V – 3.6 V | | Clock reference input. A low wander input such as an OCXO or TCXO clock can be applied on this input. Alternatively, it can be used as an extra single ended CMOS input. |
| IN0P | Input | 8 | -0.5V – 3.6 V | | Clock + for differential clock input. Single ended Input clock. |
| IN0N | Input | 9 | -0.5V – 3.6 V | | Clock - for differential clock input. Single ended Input clock. |
| IN1P | Input | 10 | -0.5V – 3.6 V | | Clock + for differential clock input. Single ended Input clock. |
| IN1N | Input | 11 | -0.5V – 3.6 V | | Clock - for differential clock input. Single ended Input clock. |
| IN2P | Input | 12 | -0.5V – 3.6 V | | Clock + for differential clock input. Single ended Input clock. |
| IN2N | Input | 13 | -0.5V – 3.6 V | | Clock - for differential clock input. Single ended Input clock. |
| IN3P | Input | 14 | -0.5V – 3.6 V | | Clock + for differential clock input. Single ended Input clock. |
| IN3N | Input | 15 | -0.5V – 3.6 V | | Clock - for differential clock input. Single ended Input clock. |
| NC | No Connect | 16 | | | Pin not bonded. |
| IN4P | Input | 17 | -0.5V – 3.6 V | | Clock + for differential clock input. Single ended Input clock. |
| IN4N | Input | 18 | -0.5V – 3.6 V | | Clock - for differential clock input. Single ended Input clock. |
| SCLK | I/O | 19 | VDDIO (Pin 48) | Pull Up | Serial Clock for I2C/SPI Mode |
| SDIO | I/O | 20 | VDDIO (Pin 48) | Pull Up | I/O data port in I2C/3W SPI. Output port in 4W SPI mode. |
| SDI_A1 | I/O | 21 | VDDIO (Pin 48) | Pull Up | Input data port in 4W SPI. Optional I2C Address bit. |
| CSB_A0 | Input | 22 | VDDIO (Pin 48) | Pull Up | Input port as chip select in SPI mode. Optional I2C Address bit. |
| LATCH_BYPASS/ NC | Input | 23 | VDDIO (Pin 48) | Pull Down | Enable bypass Wake Up GPIO latching/ No Connect for parts where GPIO latching function is needed. |
| VDD_3V3 | Power | 24 | 2.375 V – 3.6 V | | Analog PLL Power Supply. |
| TEST_ENB | Input | 25 | VDDIO (Pin 48) | Pull Up | Enable DFT Test Mode. The TEST_ENB pin needs pull up less than 5k to VDDIO. |
| GPIO3 | I/O | 26 | VDDIO (Pin 48) | Pull Down | Configurable as General Purpose IO. |
| OUT0N | Output | 27 | VDDO0 | | Clock – for differential Output. Single ended CMOS Output. |
| OUT0P | Output | 28 | VDDO0 | | Clock + for differential Output. Single ended CMOS Output. |
| VDDO0 | Power | 29 | 1.71 V – 3.6 V | | Power supply for OUT0 drivers. |
| NC | No Connect | 30 | | | Pin not bonded. |
| VDDO1 | Power | 31 | 1.71 V– 3.6 V | | Power supply for OUT1 drivers. |
| OUT1P | Output | 32 | VDDO1 | | Clock + for differential Output. Single ended CMOS Output. |
| OUT1N | Output | 33 | VDDO1 | | Clock - for differential Output. Single ended CMOS Output. |
| GPIO9 | I/O | 34 | VDDIO (Pin 48) | Pull Up | Used I2C/SPI select during power up. Configurable as General Purpose IO. |

| Name | I/O Type | Pin. No | Voltage Level | Default Pull Up/ Down | Description |
|---|---|---|---|---|---|
| OUT2N | Output | 35 | VDDO2 | | Clock – for differential Output. Single ended CMOS Output. |
| OUT2P | Output | 36 | VDDO2 | | Clock + for differential Output. Single ended CMOS Output. |
| VDDO2 | Power | 37 | 1.71 V– 3.6 V | | Power supply for OUT2 drivers. |
| VDDO3 | Power | 38 | 1.71 V – 3.6 V | | Power supply for OUT3 drivers. |
| OUT3P | Output | 39 | VDDO3 | | Clock + for differential Output. Single ended CMOS Output. |
| OUT3N | Output | 40 | VDDO3 | | Clock - for differential Output. Single ended CMOS Output. |
| VDDO4 | Power | 41 | 1.71 V – 3.6V | | Power supply for OUT4 drivers. |
| OUT4P | Output | 42 | VDDO4 | | Clock + for differential Output. Single ended CMOS Output. |
| OUT4N | Output | 43 | VDDO4 | | Clock - for differential Output. Single ended CMOS Output. |
| VDDO5 | Power | 44 | 1.71 V– 3.6 V | | Power supply for OUT5 drivers. |
| OUT5P | Output | 45 | VDDO5 | | Clock + for differential Output. Single ended CMOS Output. |
| OUT5N | Output | 46 | VDDO5 | | Clock - for differential Output. Single ended CMOS Output. |
| GPIO2 | I/O | 47 | VDDIO (Pin 48) | Pull Up | Configurable as General Purpose IO. |
| VDDIO | Power | 48 | 1.71 V– 3.6 V | | GPIO/Serial Interface port Power Supply. |
| GPIO1 | I/O | 49 | VDDIO (Pin 48) | Pull Up | Configurable as General Purpose IO. |
| OUT6N | Output | 50 | VDDO6 | | Clock – for differential Output. Single ended CMOS Output. |
| OUT6P | Output | 51 | VDDO6 | | Clock + for differential Output. Single ended CMOS Output. |
| VDDO6 | Power | 52 | 1.71 V– 3.6 V | | Power supply for OUT6 drivers. |
| OUT7N | Output | 53 | VDDO7 | | Clock – for differential Output. Single ended CMOS Output. |
| OUT7P | Output | 54 | VDDO7 | | Clock + for differential Output. Single ended CMOS Output. |
| VDDO7 | Power | 55 | 1.71 V– 3.6 V | | Power supply for OUT7 drivers. |
| GPIO5 | I/O | 56 | VDDIO (Pin 48) | Pull Up | Configurable as General Purpose IO. |
| OUT8N | Output | 57 | VDDO8 | | Clock – for differential Output. Single ended CMOS Output. |
| OUT8P | Output | 58 | VDDO8 | | Clock + for differential Output. Single ended CMOS Output. |
| VDDO8 | Power | 59 | 1.71 V– 3.6 V | | Power supply for OUT8 drivers. |
| VDDO9 | Power | 60 | 1.71 V– 3.6 V | | Power supply for OUT9 drivers. |
| OUT9P | Output | 61 | VDDO9 | | Clock + for differential Output. Single ended CMOS Output. |
| OUT9N | Output | 62 | VDDO9 | | Clock - for differential Output. Single ended CMOS Output. |
| GPIO4 | I/O | 63 | VDDIO (Pin 48) | Pull Up | Configurable as General Purpose IO. |
| OUT10N | Output | 64 | VDDO10 | | Clock – for differential Output. Single ended CMOS Output. |
| OUT10P | Output | 65 | VDDO10 | | Clock + for differential Output. Single ended CMOS Output. |
| VDDO10 | Power | 66 | 1.71 V– 3.6 V | | Power supply for OUT10 drivers. |
| NC | No Connect | 67 | | | Pin not bonded. |

| Name | I/O Type | Pin. No | Voltage Level | Default Pull Up/ Down | Description |
|---|---|---|---|---|---|
| VDDO11 | Power | 68 | 1.71 V– 3.6 V | | Power supply for OUT11 drivers. |
| OUT11P | Output | 69 | VDDO11 | | Clock + for differential Output. Single ended CMOS Output. |
| OUT11N | Output | 70 | VDDO11 | | Clock - for differential Output. Single ended CMOS Output. |
| GPIO0 | I/O | 71 | VDDIO (Pin 48) | Pull Down | Configurable as General Purpose IO. |
| RESET_DIS/NC | I/O | 72 | VDDIO (Pin 48) | Pull Down | Hard Reset Disable function. Disables Hard reset (RSTB pin5) when pulled high. |
| Ground | | ePad | 0 V | | Ground Pad |

Note: The internal pull up and pull-down resistors are 25k by default.

# 2 Electrical Characteristics

**Table 2 Absolute Maximum Ratings**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Supply Voltages | Reference XO Supply<br>FTPLL Supply<br>Input Supply<br>PLL Supply<br>GPIO Supply | $V_{DDXO\_3V3}$<br>$V_{DDFT\_1V8}$<br>$V_{DDIN}$<br>$V_{DD\_3V3}$<br>$V_{DDIO}$ | -0.5 | | +3.63 | V |
| Output bank supply voltage | Output Driver Supply | $V_{DDO}$ | -0.5 | | +3.63 | V |
| Input voltage, All Inputs | Relative to GND<br>Clock Inputs<br>OCXO Input<br>GPIO inputs | $V_{IN}$ | -0.5 | | +3.63 | V |
| XO Inputs | Relative to GND | $V_{XO}$ | -0.5 | | +1.8 | V |
| I2C Bus input voltage | SCLK<br>SDIO | $V_{INI2C}$ | -0.5 | | +3.63 | V |
| SPI Bus input voltage | SDI_A1<br>CSB_A0 | $V_{INSPI}$ | -0.5 | | +3.63 | V |
| Storage temperature | Non-functional, Non-Condensing | $T_S$ | -55 | | +150 | °C |
| Programming Temperature | | $T_{PROG}$ | +25 | | +85 | °C |
| Maximum Junction Temperature in Operation | | $T_{JCT}$ | | | +125 | °C |
| Programming Voltage (for Programming the OTP (Fuse Memory). | | $V_{PROG}$ | 2.375 | 2.5 | 2.625 | V |
| ESD (human body model) | JESD22A-114 | $ESD_{HBM}$ | | | 2000 | V |
| ESD (charge device model) | | $ESD_{CDM}$ | | | 500 | V |
| Latch Up | JEDEC JESD78D | LU | | | 100 | mA |

Notes:

- Exceeding maximum ratings may shorten the useful life of the device.
- Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or at any other conditions beyond those indicated under the DC Electrical Characteristics is not implied.
- Exposure to Absolute-Maximum-Rated conditions for extended periods may affect device reliability or cause permanent device damage.
- Any pin category not covered here has a default minimum rating of -0.5 V and a default maximum rating of 3.63 V.

**Table 3 Operating Temperature and Thermal Characteristics**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Ambient temperature | | TA | –40 | – | +85 | °C |
| Junction temperature | | TJ | | | +125 | °C |
| **AU5508: 72-QFN package** | | | | | | |
| Thermal Resistance Junction to Ambient | Still Air | $\theta_{JA}$ | | 18.1 | | °C/W |
| | Air Flow :1m/s | | | 15.3 | | |
| | Air Flow: 2m/s | | | 14.3 | | |
| Thermal Resistance Junction to Case | | $\theta_{JC}$ | | 6.7 | | °C/W |
| Thermal Resistance Junction to Board | | $\theta_{JB}$ | | 6.46 | | °C/W |

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| **AU5508** | | | | | | |
| PLL Supply | 2.5 V range: ± 5% | $V_{DD\_3V3}$ | 2.375 | 2.5 | 2.625 | V |
| | 3.3 V range: ± 10% | | 2.97 | 3.3 | 3.63 | V |
| XO Supply | 2.5 V range: ± 5% | $V_{DDXO\_3V3}$ | 2.375 | 2.5 | 2.625 | V |
| | 3.3 V range: ±10% | | 2.97 | 3.3 | 3.63 | V |
| FTPLL/TDC Supply | 1.8 V range: ±5% | $V_{DDFT\_1V8}$ | 1.71 | 1.8 | 1.89 | V |
| Input Buffer Supply | 1.8 V range: ±5% | $V_{DDIN}$ | 1.71 | 1.80 | 1.89 | V |
| | 2.5 V range: ±5% | | 2.375 | 2.50 | 2.625 | V |
| | 3.3 V range: ±10% | | 2.97 | 3.3 | 3.63 | V |
| GPIO Supply | 1.8 V range: ±5% | $V_{DDIO}$ | 1.71 | 1.80 | 1.89 | V |
| | 2.5 V range: ±5% | | 2.375 | 2.50 | 2.625 | V |
| | 3.3 V range: ±10% | | 2.97 | 3.3 | 3.63 | V |
| Output Driver Supply | 1.8 V range: ±5% | $V_{DDO}$ | 1.71 | 1.80 | 1.89 | V |
| | 2.5 V range: ±5% | | 2.375 | 2.50 | 2.625 | V |
| | 3.3 V range: ±10% | | 2.97 | 3.3 | 3.63 | V |
| **Power Dissipation (VDD_3V3 = VDDXO_3V3 = VDDO = 3.3 V, VDDFT_1V8 = 1.8)** | | | | | | |
| Total Power Dissipation (3.3V LVDS Outputs @ 156.25M) | 4 PLLs, 12 Outputs<br>5 DE Inputs, OCXO PLL | Pd | | 2.2 | | W |
| | 1 PLL, 2 Outputs<br>1 DE | | | 500 | | mW |
| **Supply Current** | | | | | | |
| VDDIN | All Five DE Inputs assumed to be enabled | $I_{DDIN}$ | | 55 | | mA |
| VDD_3V3 | All Four PLLs and All 12 Outputs enabled | $I_{DD\_3V3}$ | | 350 | | mA |
| VDDFT_1V8 | FTPLL and All 10 Input TDC enabled | $I_{DDFT\_3V3}$ | | 100 | | mA |
| VDDXO_3V3 | Crystal Oscillator | $I_{DDXO\_3V3}$ | | 10 | | mA |
| VDDO[4,7] | LVPECL, output pair terminated 50 Ω to $V_{TT}$ (VDDO – 2 V). | $I_{DDO}$ [1,2,3,5] | | 44 | | mA |
| | CML, output pair terminated 50 Ω to VDDO | $I_{DDO}$ [1,5] | | 20 | | mA |
| | HCSL, output pair with HCSL termination | $I_{DDO}$ [1,5] | | 34 | | mA |
| | LVDS output pair terminated with an AC or DC Coupled diff 100 Ω | $I_{DDO}$ [1,5] | | 28 | | mA |
| | LVCMOS, 250 MHz, 2.5V output, 5-pF load | $I_{DDO}$ [5,6] | | 15 | | mA |

Notes:

1. LVPECL standard is supported for VDDO = {2.5 V, 3.3 V}. HCSL, CML and LVDS standards are supported for VDDO = {1.8 V, 2.5 V, 3.3 V}.
2. LVPECL mode provides 6 mA of common mode current on each output.
3. A 50 Ω Termination resistor with a DC bias of VDDO – 2 V for LVPECL standards is supported for VDDO = {2.5 V, 3.3 V}.

4. IDDOx Output driver supply current specified for one output driver in the table. This includes current in each of the output module that includes output dividers, drivers and clock distributions.
5. Refer to the Output Termination Information in Section 9 in the data sheet for the description of the various terminations that are supported.
6. Both P and N terminal are active in LVCMOS mode.
7. Current consumption doesn't account for load current. LVCMOS current include load current also.

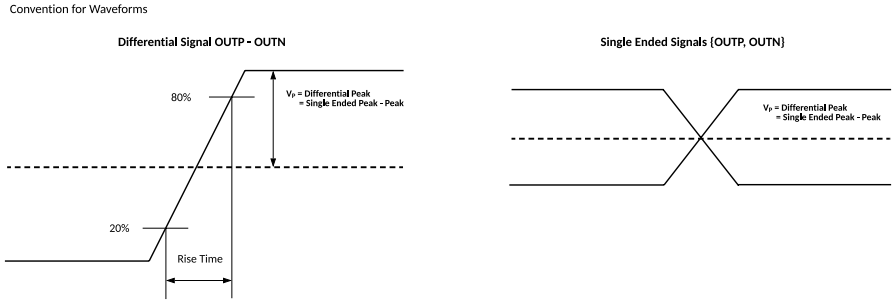
**Table 4 Input Clock Characteristics**

| Parameter | Conditions | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Input Frequency Range | Differential[1] | $f_{IN}$ | 0.5 | — | 2100M | Hz |
| | P Side Single Ended[1] | | 0.5 | — | 250M | Hz |
| | N Side Single Ended[2] | | 0.5 | — | 187.5M | Hz |
| PLL Input Frequency Range [3] | PLL Input Frequency | $f_{IN\_DPLL}$ | 0.5 | — | 12.5M | Hz |
| **Input Buffer with Differential DC or AC Coupled (See Input Slave Description for Termination information)** | | | | | | |
| Input Common Mode Voltage Range (for DC Coupled Differential Inputs)[4] | Differential DC coupled inputs; Defined as cross point | $V_{CMR}$ | 0.25 | - | VDDIN – 0.85 | V |
| Voltage Swing (Differential Amplitude Peak or Single Ended Peak to Peak for the differential signal) | $f_{IN}$ < 400 MHz | $VP_{IN}$[5] | 100 | — | - | mV |
| | 400 MHz < $f_{IN}$ < 750 MHz | | 225 | — | - | mV |
| | 750 MHz < $f_{IN}$ < 2100 MHz | | 350 | — | - | mV |
| Slew Rate | | SR | 400 | - | - | V/µs |
| Input Capacitance | | $C_{IN}$ | - | 1 | - | pF |
| Input Resistance | Differential DC (on each input) | $R_{IN}$ | — | 26 | — | kΩ |
| | Differential AC | | - | 10 | - | |
| **Single Ended AC Coupled input (IN0P/N, IN1P/N, IN2P/N, IN3P/N, IN4P/N, OCXO Input)** | | | | | | |
| Voltage Swing (Vpp) | AC-Coupled $f_{IN}$ < 250 MHz | $f_{IN,SE\_AC}$ | 500 | _ | 3600 | mV |
| Slew Rate[6] | | SR | 400 | — | — | V/µs |
| Duty Cycle | | DC | 40 | — | 60 | % |
| Input Capacitance | | $C_{IN}$ | — | 0.3 | — | pF |
| Input Resistance | AC Coupled SE | $R_{IN}$ | — | 24 | — | kΩ |
| **Single Ended DC-coupled (IN0P/N, IN1P/N, IN2P/N, IN3P/N, IN4P/N, OCXO Input)** | | | | | | |
| Input Voltage | | $V_{IL}$ | –0.2 | — | 0.4 | V |
| | | $V_{IH}$ | 0.8 | — | VDDIN | V |
| Slew Rate[6] | | SR | 400 | — | — | V/µs |
| Duty Cycle[7] | | DC | 40 | — | 60 | % |
| Input Resistance | | $R_{IN}$ | — | 26 | — | kΩ |
| **Reference Clock (Applied to XA), Can be external XO** | | | | | | |
| Reference Clock Frequency | Range for best jitter | $F_{IN\_REF}$ | 48 | - | 160 | MHz |
| | Overall supported range | | 25 | - | 160 | MHz |
| Input Voltage Swing[8] | Single Ended peak to peak | $V_{IN\_SE}$ | 365 | - | 2000 | mVpp_se |

| Parameter | Conditions | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Slew rate | | SR | 400 | - | - | V/us |
| Duty Cycle | | DC | 40 | - | 60 | % |

Notes:

1. Differential and SE P Side has fractional divider with integer divide range of 2 to 65535
2. SE N Side has integer divider with divide range of 2-15
3. For proper device operation, the input frequency is internally divided down to 12.5 MHz or less (PLL Phase Detector maximum frequency = 12.5 MHz). This is achieved using internal dividers in the chip.
4. For VDDIN = 1.8 V, max $V_{CMR}$ = VDDIN - 0.5 V.
5. $VP_{IN}$ is the single-ended peak-peak of the input signal which is equal to the differential peak. This is the swing requirement for both AC and DC coupled differential inputs where the swing is considered at the pin inputs.




6. Minimum slew rate specification is for best noise performance.
7. The following Table shows the minimum Pulse Width supported for various Input Clock frequencies with 54 MHz Reference clock

| XTAL/XO (with Doubler enabled) | Input Frequency (Hz) > | Input Frequency (Hz) <= | Minimum Input Clock Pulse |
|---|---|---|---|
| 108 MHz | 0.5 | 15 | 10 ms |
| | 15 | 210 | 1 ms |
| | 210 | 3.5 K | 50 us |
| | 3.5 K | 55 K | 5 us |
| | 55 K | 850 K | 200 ns |
| | 850 K | 30 M | 15 ns |
| | 30 M | 1 G | DC: 40%-60% |

8. Max Voltage at XA pin should be < 1.8 V.

**Table 5 Serial Data and GPIOs**

| Parameter | Condition | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Input Voltage | | $V_{IL}$ | — | — | 0.3 x $V_{DDIO}$[1] | V |
| | | $V_{IH}$ | 0.7 x $V_{DDIO}$[1] | — | VDDIO | V |
| Input Capacitance | | $C_{IN}$ | — | 1 | — | pF |
| Input Resistance | | $R_{IN}$ | — | 25 | — | kΩ |
| Minimum Pulse Width | FINC, FDEC [2,3] | PW | 100 | — | — | ns |
| Minimum RESET duration | | $T_{RES}$ | 100 | | | µs |
| Update Rate | FINC, FDEC [2,3] | $F_{UR}$ | 1 | — | — | µs |
| Minimum RESET Pulse Width | | | 100 | — | — | us |
| Minimum time between RSTb release and SPI/I2C ready | | | 1 | — | — | ms |

Notes:

1. VDDIO is the voltage used for all the status GPIOs and the serial interface. This is the voltage applied on Pin 48
2. FINC/FDEC are the increment and decrement for the DCO operation from the pins
3. Minimum Pulse width/update rate are specified for free run PLL DCO

**Table 6 Output Serial and Status Pin**

| Parameter | Test Condition | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| **All VDDIO based GPIOs** | | | | | | |
| Output Voltage | $I_{OH}$ = -2 mA | $V_{OH}$ | $V_{DDIO}$ x 0.75 | — | — | V |
| | $I_{OL}$ = 2 mA | $V_{OL}$ | — | — | $V_{DDIO}$ x 0.25 | V |

Notes:

- VDDIO is the voltage used for all the status GPIOs and the serial interface. This is the voltage applied on Pin 48

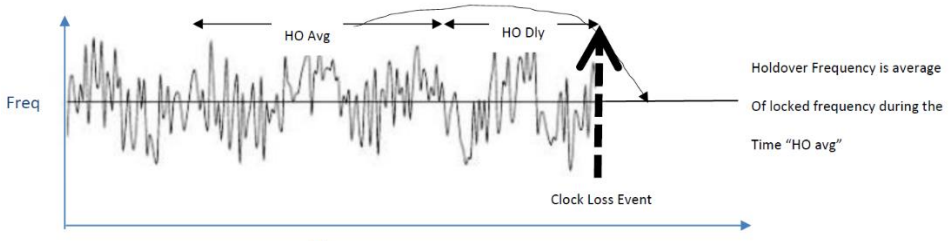
**Table 7 Output Clock Characteristics**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Differential output frequency continuous support[1],[2] | Differential output standards | $F_{OUT, DIFF}$ [1,3] | 0.5 | | 800M | Hz |
| Differential output frequency Selective Higher Frequency Support (For >800MHz outputs) [3] | Differential Output Standards | | 61.44M* {15, 20, 24, 30, 32, 40, 48} = 921.6M, 1228.8M, 1474.56M, 1843.2M, 1966.08M, 2457.6M, 2949.12M | | | Hz |
| Single ended output frequency | LVCMOS outputs | $F_{OUT, SE}$ | 0.5 | | 250 M | Hz |
| PLL loop bandwidth | Programmable | $F_{BW}$ | 0.00009 | | 4000 | Hz |
| Jitter peaking [4] | Meets SONET Jitter Peaking requirements in closed loop (see footnote) | $J_{PEAK}$ | | | 0.1 | dB |
| Time delay before the historical average for output frequency is considered. | Programmable in register map | $H_{DELAY}$ [5] | 0.002 | | 35 | s |
| Length of time for which the average of the frequency is considered | Programmable in register map | $H_{AVG}$ [5] | 0.004 | | 70 | s |
| Power Supply to I2C or SPI interface ready | No I2C or SPI transaction valid till 10ms after all power supplies are ramped to 90% of final value. | $T_{START}$ | | | 10 | ms |
| Hold Time for GPIO Latching[6] | Hold time for GPIO latched inputs available on the GPIOs after the RSTB pin is driven from low to high | $T_{HOLD}$ [6] | 1 | | | ms |
| DCO Mode Frequency Step Resolution[7] | Frequency Increment or Decrement resolution. This is controlled through the register map. | $F_{RES,DCO}$ [7] | 0.001 | | | ppt |
| Output Phase Shift [8] | Resolution for output delay between clocks from same PLL. Resolution Programmable per output clock with this resolution for a total delay range of ±T/2 where T is the time period of the output clock | $T_{RESO}$ [8] | | 35 | | ps |
| Output Skew[9] | Skew between outputs from the same PLL set up with same phase shift code | $T_{SKEW}$ [9] | -50 | | 50 | ps |

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Input Phase Shift Resolution[10] | Programmable delay resolution for all outputs that are locked to a particular input. Input phase shift is programmable per input clock with this resolution for a total delay of ±T/2 where T is the time period of the input clock | $T_{RESI}$ [10] | 1 | | | ps |
| PLL Lock Time[11] | Standard Mode | $T_{LOCK}$ | | 300m | | sec |
| | 1 pps Mode, Input to Output Phase Error < +/- 30ns | $T_{LOCK\_PHASE\_1PPS}$ | | | 30 | sec |
| Maximum Phase Hit[12] | Default PBO Hitless Switching Mode (no phase propagation) | $T_{MAX}$ [12] | -25 | | 25 | ps |
| Input to Output Delay variation in external ZDB mode (external ZDB with feedback on the PCB)[13][14] | Any output to any input external feedback is possible. Supported for all PLLs. Multiple PLLs in external ZDB are supported concurrently | $T_{ZDELAY}$ | -100 | | 100 | ps |
| Internal ZDB Mode Input to Output Delay Variation [14] | Supported for all PLLs. Multiple PLLs in internal ZDB are supported concurrently. | $T_{ZDELAY,INT}$ | -500 | | 500 | ps |
| Uncertainty in Input to Output Delay | Maximum variation in the static delay from input to output clock between repeated power ups of the chip | $\Delta T_{DELAY}$ | -225 | | 225 | ps |
| PLL Pull Range | | $\omega_P$ | | 500 | | ppm |

Notes:

1. The continuous frequency support implies that all output frequencies till 800 MHz are available with no gaps.
2. The VCOs support two ranges: A Low Band Range of 4843.75 MHz to 5898.24 MHz and a High Band Range of 6875 MHz to 8000 MHz. Along with the fully flexibly output multiplexer for the output clocks, this provides for customer use cases to be easily supported with several concurrent frequencies for application scenarios possible from a single PLL.
3. Specific multiples of 61.44 MHz are provided for Wireless applications. Please contact Aurasemi for more frequency options and details.
4. Jitter peaking limit of < 0.1 dB can be enabled as an option for cases (such as SONET) where there is such a critical requirement. For other cases, the jitter peaking can be made slightly higher to enable faster transients and lock characteristics.
5. Hitless Switching enables PLL to switch between input clocks when the current clock is lost,
    a. Clock Loss can be defined as a specified number of consecutive missing pulses.
    b. Priority list for the input clocks can be set in the register map independently for each PLL.
    c. Output is truly hitless (no phase transient and 0 ppb relative error in frequency) for exactly same frequency input clocks that are switched in Phase Build Out Mode.
    d. Hitless switching support is both revertive and non-revertive
    i. Revertive / Non-revertive Support: Assume Clock Input 0 is lost and switch is made to Clock Input 1. Then, PLL reverts to Clock Input 0 when it becomes valid again in Revertive mode. It does not switch back to Clock Input 0 even when it becomes valid again in the non-Revertive mode.
    c. Entering hold over mode is supported with the frequency frozen at a historical average determined from the $H_{DELAY}$ and $H_{AVG}$ settings.




6. The AU5508 chip provides a GPIO latching function that allows for certain GPIOs to function as latched GPIOs that are latched along with the release of chip reset (using the RSTB pin) and can the same pin is released for other functions in steady state. GPIO latching is disabled by default. GPIO latching is offered as a factory programmed part with non 00 code.

a. This requires the RSTB pin to be held low and released from low to high ONLY when all supplies to the chip have crossed 90% of their final value.
b. This further requires that the GPIOs whose inputs are used for GPIO latching should “Hold” the expected value for latching for at least 1ms of time after the RSTB pin has reached 90% of the VDDIO supply.

7. The 0.001 ppt specification is for the smallest frequency step resolution available. The frequency resolution for the DCO mode frequency step is independently programmable for each DCO step.
c. DCO steps are applicable on both XO referenced PLLs (free run) and input referenced PLL (sync mode)
d. DCO step size is programmed in the programmable interface (PIF) using the serial I2C or SPI interface
e. DCO is enacted in response to a trigger. This trigger can be provided either with a register based write through the serial interface or a pin-based trigger where a GPIO is programmed for the purpose of the increment and decrement DCO trigger.
8. The delay referred to here is delay between outputs from the same PLL. Such a delay can delay a clock by as much as 180 degrees which is half of a time period of the output clock with the resolution of 35 ps.
f. All output clocks from one specific PLL are phase aligned by default. Relative delay adjustment is then possible on each clock individually as defined by the $T_{RES}$ parameter for a total delay range of ±T/2 where T is the time period of the output clock.
9. This is the skew between outputs from the same PLL such that they are set up with the same relative output phase shift code.
10. The delay referred to here is from input to output hence it appears in the system as an input phase shift. All outputs from a PLL that is locked to a particular input can move by as much as 180 degrees of the input clock for the PLL (which is half the time period for the PLL input clock) and with a resolution of +/- 10ps.
11. For low PLL Loop Bandwidths, wake up time can be very large unless the speed up features are used. The speed up feature provides the user options to use a completely independent time constants for the wake-up transitioning to the regular bandwidth after frequency and phase are locked
a. Fast Lock Bandwidth needs to be less than 100 times smaller than the input clock frequency (divided input at PLL phase detector) for stable and bounded (in time) lock trajectory of the PLL
b. 1 PPS locking has extra features inbuilt especially with respect to facilitating the sub +/- 30ns phase lock in addition to the frequency lock within the 30 seconds specification
12. This test is for 2 inputs at 8M that are switched to get a 156.25M output.
13. Both input and feedback at 8MHz with the delays exactly matched and same slew for both for the chip
14. Internal ZDB mode is supported concurrently on any number of PLLs. The delay specification here is for delay between the input port and output port of the chip. With PLL A and B configured in Internal ZDB mode, only internal feedback frequency can be brought out on OUT0 and with PLL C and D configured in Internal ZDB mode, only internal feedback frequency can be brought out on OUT11.

**Table 8 Fault Monitoring Indicators**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Clock Loss Indicator Thresholds | Clock Loss Indicators can be set on any of the 10 inputs or OCXO or ClkInSync. Loss of 2-16 consecutive pulses can be used to indicate a clock loss. Programmable in the register map. | $CL_X$ [1,4,5] | 2 | | 16 | Pulses |
| Fine Frequency Drift Indicator Thresholds: Range | Fine Drift Indicators programmable from ±2 ppm upto ±500 ppm in steps of ±2 ppm. | $FD_X$ [2,3,4,5] | ±2 | | ±510 | ppm |
| Coarse Frequency Drift Indicator Thresholds | Coarse Drift Indicators programmable from ±100 ppm upto ±1600 ppm in steps of ± 100 ppm. | $FD_X$ [2,3,4,5] | ±100 | | ±1600 | ppm |
| Phase Lock Loss Indicator | Provides phase locking indication for lock of the input and feedback clock phases with respect to each other for each PLL. This is particularly useful for 1 PPS lock | LLPH | 1n | | 100u | sec |

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Lock Loss Indicator Threshold[6] | Lock Loss Indicator threshold is programmable in the range specified from the following choices for setting and clearing LL:<br>{±0.05, ±0.1} ppb,<br>{±0.5, ±1} ppb,<br>{±0.2, ±0.4} ppm,<br>{±2, ±4} ppm,<br>{±20, ±40} ppm,<br>{±200, ±400} ppm,<br>{±2000, ±4000} ppm | LL | ±0.05 ppb | | ±4000 ppm | |

Notes:

1. Clock Loss Indicators are used for:
    a. Hitless Switching Triggers
    b. Update in Status Registers in the register map
2. Frequency Drift Indicators can use any one of the 10 inputs, OCXO, Internal ClockInSync or the Crystal / Reference input clock as the golden reference with respect to which FDx for all other clocks can be recorded in the Status Registers. FDx thresholds for each clock input for each clock can be set independently.
3. Coarse and Fine Frequency Drift indicators can be concurrently enabled. This enables the user to detect fast drifting frequencies since detecting fine drifts will take longer measurements.
4. Clock loss and Lock loss indicators are available as alerts on GPIO pins as described in Section 7.1.
5. Clock Loss can be combined with either of the frequency drift monitors (coarse and fine) to trigger the hitless switching event in the PLLs. The trigger for a hitless switching event in the PLL can therefore be either the Clock Loss event or either of Clock Loss or Frequency Drift.
6. Following Table shows the resolution of lock detection threshold

| LL Threshold | 0.05 | 0.1 | 0.5 | 1 | 0.2 | 0.4 | 2 | 4 | 20 | 40 | 200 | 400 | 2000 | 4000 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Detection Resolution | 0.02 | | 0.2 | | 0.02 | | 0.2 | | 2 | | 20 | | 200 | |
| Units | ppb | | ppb | | ppm | | ppm | | ppm | | ppm | | ppm | |

**Table 9 Crystal Requirements**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| **High Fundamental Frequency Crystal Reference (H000FF)** | | | | | | |
| Crystal Frequency | Can be supported with a fundamental crystal of 100-160 MHz range. | $XTAL_{IN}$ | 100 | | 160 | MHz |
| C0 cap for crystal | Small range around CL only | $XTAL_{C0}$ | | | 2 | pF |
| CL cap for crystal | | $XTAL_{CL}$ | | 5 | | pF |
| ESR for crystal | ESR defined at frequency of oscillation | $XTAL_{ESR}$ | | | 40 | Ω |
| Rm1 for crystal | | $XTAL_{Rm1}$ | | | 20 | Ω |
| Power delivered to crystal | Drive Level to the crystal | $XTAL_{PWR}$ | | 100 | | µW |
| **Low Frequency Fundamental Crystal (LFF)** | | | | | | |
| Crystal Frequency | Can be supported with a fundamental crystal > 25 MHz range. For Best Performance use an LFF crystal > 48 MHz | $XTAL_{IN}$ | 25 | | 54 | MHz |
| C0 cap for crystal | Small range around CL only | $XTAL_{C0}$ | | | 2 | pF |
| CL cap for crystal [2] | | $XTAL_{CL}$ | | 8 | | pF |
| ESR for crystal | ESR defined at frequency of oscillation | $XTAL_{ESR}$ [1] | | | 40 | Ω |
| Rm1 for crystal | | $XTAL_{Rm1}$ | | | 40 | Ω |
| Power delivered to crystal | Drive Level to the crystal | $XTAL_{PWR}$ | | 100 | | µW |

Notes:

1. ESR relates to the motional resistance Rm with the relationship $ESR = Rm\ (1 + C0/CL)^2$
2. The table specifies the Ci, Ce and Cs for the 54MHz XTAL with different CL. Ci is the internal differential capacitance offered by the chip whereas Ce (Ce=Ce1=Ce2) and Cs(Cs=Cs1=Cs2) are the Single Ended external and the stray cap on the PCB.

| XTAL | Ci | Ce | Cs | Unit |
|---|---|---|---|---|
| 54MHz (CL = 8pF) | 7.5 | 0 | 1 | pF |
| 54MHz (CL = 12pF) | 8.1 | 7 | 1 | pF |

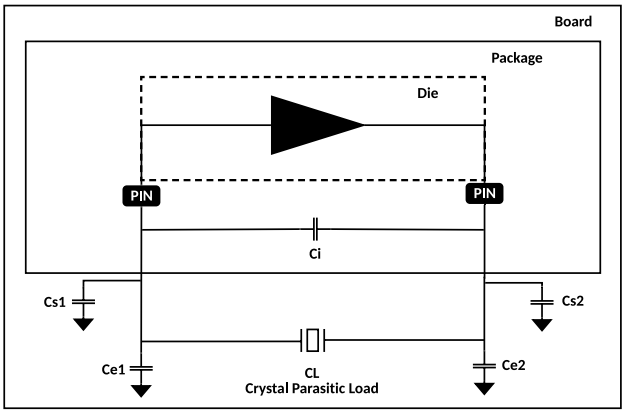



**Table 10 Output RMS Jitter in Frequency Translation Modes**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| RMS Jitter for 12 kHz – 20 MHz Integration Bandwidth $F_{IN}$ = 38.88 MHz, PLL BW = 100 Hz, Single PLL Profile | $F_{OUT}$ = 156.25 MHz 54 MHz Crystal [3] | $RMS_{JIT}$ [1,2] | | 120 | | fs rms |
| | $F_{OUT}$ = 156.25 MHz 54 MHz External XO [4] | | | 110 | | fs rms |

Notes:

1. For best noise performance in jitter attenuation mode, use lowest usable loop bandwidth for the PLL.
2. Does not include noise from the input clocks to the PLL for the 54M Crystal Mode.
3. Crystal is assumed to meet the specifications mentioned in the AU5508 Datasheet.
4. External XO with R&S SMA100 equipment for the reference (XO) and input Fin is used.

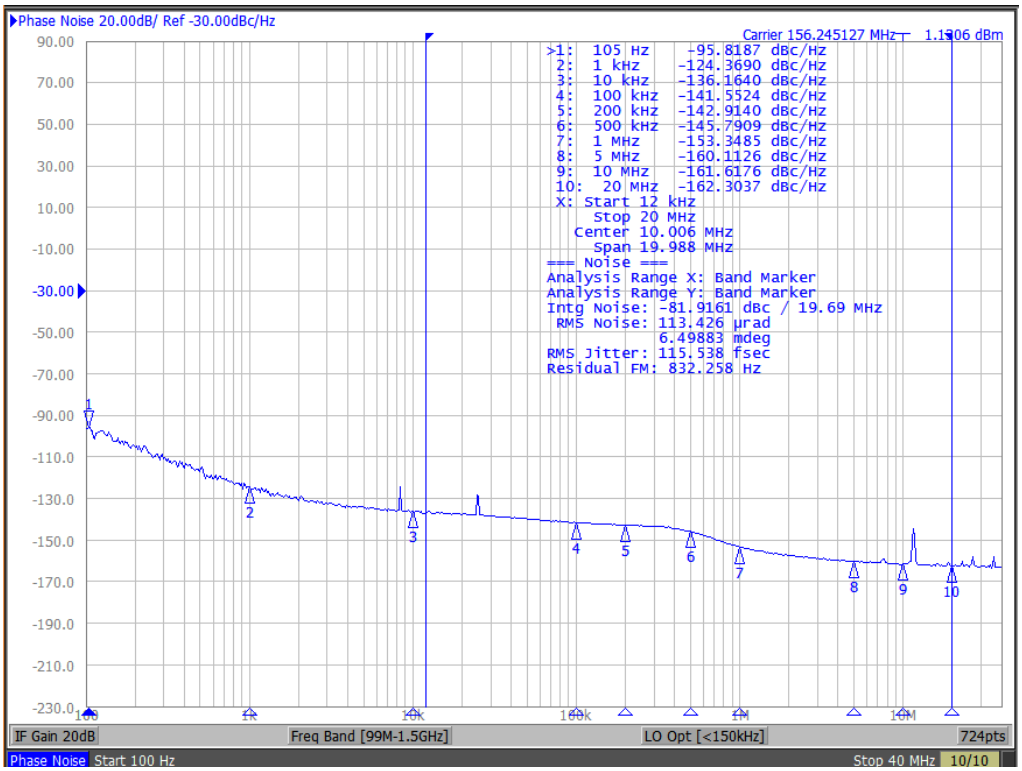



**Figure 3 Representative Phase Noise Measurement [54M XTAL- Free Run Mode]**

Notes:

FOUT = 156.25 MHz, Fref = 54 MHz Crystal

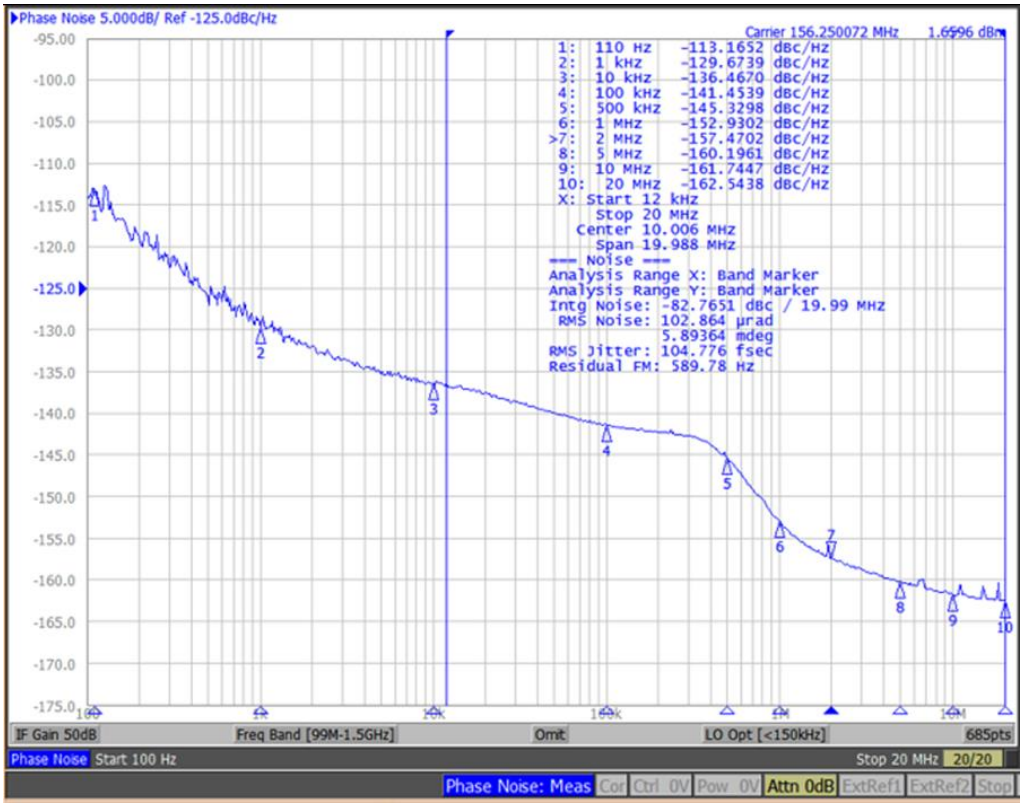



**Figure 4 Representative Phase Noise Measurement [54M XO External- Sync Mode]**

Notes:

FOUT = 156.25 MHz, Fref = 54 MHz XO from SMA100A, Fin= 61.44MHz, PLL BW = 100Hz

**Table 11 Close-In Offset Phase Noise**

| Description | Offset Frequency | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Phase Noise Skirt<br>$F_{IN}$ = 38.88 MHz<br>$F_{OUT}$ = 122.88 MHz,<br>PLL BW = 100 Hz<br>54 MHz Crystal [2] | 100 Hz | PN[1] | | -115 | | dBc/Hz |
| | 1 kHz | | | -130 | | |
| | 10 kHz | | | -136 | | |
| | 100 kHz | | | -141 | | |
| | 1 MHz | | | -153 | | |
| | 10 MHz | | | -161 | | |

Notes:

1. This is the noise contribution of the chip only without including the input and reference self-contributions.
2. External XO and Clock Inputs are fed from SMA100A source and HSM3001B respectively

**Table 12 Power Supply Rejection**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $F_{OUT}$ = 156.25 MHz,<br>$F_{SPUR}$ = 100 kHz,<br>BW = 100 Hz<br>PSRR on PLL Supply | VDD_3V3 = 3.3 V | $PSRR_{VDD}$ | | -90 | | dBc |
| $F_{OUT}$ = 156.25 MHz,<br>$F_{SPUR}$ = 100 kHz,<br>BW = 100 Hz<br>PSRR on Input Supply | VDDIN = 3.3 V | $PSRR_{VDDIN}$ | | -100 | | dBc |
| $F_{OUT}$ = 156.25 MHz,<br>$F_{SPUR}$ = 100 kHz,<br>BW = 100 Hz<br>PSRR on Output Driver Supply | VDDO = 3.3 V | $PSRR_{VDDO}$ | | -90 | | dBc |

Notes:

- The PSRR is measured with a 50 mVpp sinusoid in series with the supply and checking the spurious level relative to the carrier on the output in terms of phase disturbance impact.
- Output PSRR measured with LVDS standard which is the recommended standard for AC Coupled terminations

**Table 13 Adjacent Output Cross Talk**

| Description | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| 156.25 M and 155.52 M on adjacent outputs | LVDS Output | $X_{TALK}$ | | -75 | | dBc |

Notes:

- Measured across adjacent outputs- All adjacent outputs are covered and the typical value for the worst-case output to output coupling is reported.
- This cross talk between outputs is mainly package dependent therefore terminated outputs are used for reporting these numbers ensuring that there is signal current in the bond wires.

**Table 14 Output Clock Specifications**

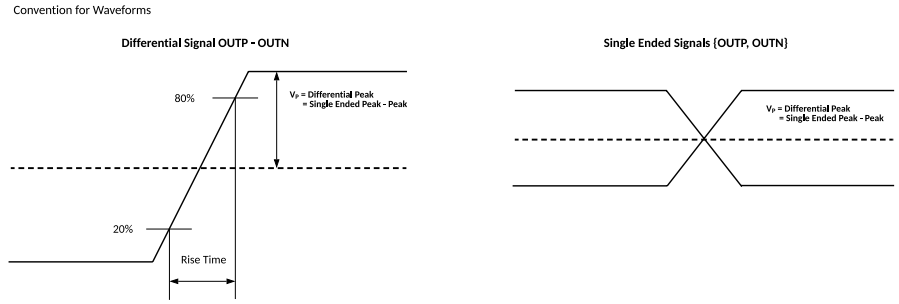
| Descriptions | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| **DC Electrical Specifications - LVCMOS output (Complementary Out of Phase Outputs or One CMOS Output per Output Driver)** | | | | | | |
| Output High Voltage | 4 mA load, VDD = 3.3 V | $V_{OH}$ | VDDO-0.3 | | - | V |
| Output High Voltage | 4 mA load,<br>VDD = 1.8V and 2.5 V | $V_{OH}$ | VDDO-0.4 | | - | V |
| Output Low Voltage | 4 mA load | $V_{OL}$ | | | 0.3 | V |
| **DC Electrical Specifications - LVCMOS output (In Phase Outputs)** | | | | | | |
| Output High Voltage | 4 mA load, VDD = 3.3 V | $V_{OH}$ | VDDO-0.35 | | - | V |
| Output High Voltage | 4 mA load, VDD = 2.5 V | $V_{OH}$ | VDDO-0.45 | | - | V |
| Output High Voltage | 4 mA load, VDD = 1.8 V | $V_{OH}$ | VDDO-0.5 | | - | V |
| **DC Electrical Specifications – LVDS Outputs (VDDO = 1.8 V, 2.5 V or 3.3 V range)** | | | | | | |

| Descriptions | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Output Common-Mode Voltage | VDDO = 2.5 V or 3.3 V range | $V_{OCM}$ | 1.125 | 1.2 | 1.375 | V |
| Output Leakage Current | Output Off, VOUT = 0.75 V to 1.75 V | $I_{OZ}$ | -20 | | 20 | µA |
| **AC Electrical Specifications LVCMOS Output Load: 10 pF < 125 MHz, 7.5 pF < 150 MHz, 5 pF > 150 MHz** | | | | | | |
| Output Frequency | | $f_{OUT}$ | 0.5 | | 250M | Hz |
| Output Duty cycle | Measured at 1/2 VDDO, loaded, $f_{OUT}$ < 125 MHz | $t_{DC}$ | 45 | | 55 | % |
| Output Duty cycle | Measured at 1/2 VDDO, loaded, $f_{OUT}$ > 125 MHz | $t_{DC}$ | 40 | | 60 | % |
| Rise/Fall time | VDDO = 1.8 V, 20-80%, Highest Drive setting | $t_{RFCMOS}$ | | | 2 | ns |
| Rise/Fall time | VDDO = 2.5 V, 20-80%, Highest Drive setting | $t_{RFCMOS}$ | | | 1.5 | ns |
| Rise/Fall time | VDDO = 3.3 V, 20-80%, Highest Drive setting | $t_{RFCMOS}$ | | | 1.2 | ns |
| **AC Electrical Specifications (LVPECL, LVDS, CML)** | | | | | | |
| Clock Output Frequency | | $f_{OUT}$ | 0.5 | | 2949.12 M | Hz |
| LVPECL Output Rise/Fall Time | 20% to 80% of AC levels. Measured at 156.25 MHz for PECL outputs. | $t_{RF}$ | | | 350 | ps |
| CML Output Rise/Fall Time | 20% to 80% of AC levels. Measured at 156.25 MHz for CML outputs | $t_{RF}$ | | | 350 | ps |
| LVDS Output Rise/Fall Time | 20% to 80% of AC levels. Measured at 156.25 MHz for LVDS outputs. | $t_{RF}$ | | | 350 | ps |
| HCSL Output Rise/Fall Time | 20% to 80% of AC levels. Measured at 156.25 MHz for HCSL outputs. | $t_{RF}$ | | | 350 | ps |
| Output Duty Cycle | Measured at differential 50% level, 156.25 MHz | $t_{ODC}$ | 45 | 50 | 55 | % |
| Programmable Output differential peak LVDS DC Coupled Output[1],[2] | Measured at 156.25M Output (Programmable Typical Levels mentioned) | VP | | 100 | | mV |
| | | | | 200 | | |
| | | | | 300 | | |
| | | | 300 | 400 | 500 | |
| | | | | 500 | | |
| | | | | 600 | | |
| | | | | 700 | | |
| | | | 500 | 800 | 1000 | |
| Programmable Output differential peak HCSL Coupled Output[1],[2] | Measured at 156.25M Output (Programmable Typical Levels mentioned) DC Coupled True HCSL termination assumed. See foot note | VP | | 500 | | mV |
| | | | | 550 | | |
| | | | | 600 | | |
| | | | | 650 | | |
| | | | | 700 | | |
| | | | | 750 | | |
| | | | 400 | 800 | 1000 | |
| Programmable Output differential peak CML Coupled Output[1],[2] | Measured at 156.25M Output (Programmable Typical Levels mentioned) DC Coupled True CML termination assumed. See foot note | VP | | 50 | | mV |
| | | | | 100 | | |
| | | | | 150 | | |
| | | | | 200 | | |
| | | | | 250 | | |
| | | | | 300 | | |
| | | | | 350 | | |

| Descriptions | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| | | | 250 | 400 | 600 | |
| Programmable Output differential peak LVPECL Coupled Output[1],[2] | Measured at 156.25M Output (Programmable Typical Levels mentioned) DC Coupled True LVPECL termination assumed. See foot note | VP | | 500 | | mV |
| | | | | 540 | | |
| | | | | 580 | | |
| | | | | 620 | | |
| | | | | 660 | | |
| | | | | 680 | | |
| | | | | 700 | | |
| | | | 450 | 720 | 900 | |

Notes:

1.




2. Please see Section 9 related to the output slave to determine the output termination supported as per standard.
   a. LVDS standard is recommended for most AC Coupled termination cases OR DC coupled differential 100 Ohm loading
   b. LVPECL Standard mentioned here supports the true traditional standard with DC coupled 50 Ohm terminations to VDDO- 2V or its corresponding Thevenin equivalent network.
   c. CML Standard mentioned here supports the true traditional standard with DC coupled 50 Ohm terminations to VDDO supply.
   d. HCSL Standard mentioned here supports the true traditional standard with DC coupled 50 Ohm terminations to Ground which are then AC or DC coupled to a differential receiver

# 3 Functional Description

The AU5508 is a jitter attenuating frequency translation device that offers four independent PLLs. The fully integrated part offers IEEE1588/ SyncE / 1 pps full support along with low integrated jitter for higher data rate links along with low close-in noise and full JESD204B/C support for Data Converter Clocks. These features provide a unique feature rich definition in a highly integrated part.

The five unique differential (or 10 single ended) clocks can be routed to any of the four independent PLLs as well as a high precision independent input TDC (time to digital converter) that can be used to measure any relative phase delays between inputs. A fully flexible output RF multiplexer allows any PLL's output to be routed to any output. This offers a very flexible frequency translation arrangement with independent control of each PLL in terms of jitter attenuation, bandwidth control and input clock selection with redundancy. The hierarchy of the clocks, nomenclature of the various frequency dividers as well as the clock translation pathways available on the chip are shown in Figure 5, Figure 6, and Figure 7.

The digital architecture of the chip is partitioned into a master digital controller and nine slave controllers. The master controller and each of the nine controllers has an associated volatile programmable interface (PIF). The overall PIF structure is a register map that is divided into several pages according to function. Each controller (master and slaves) has an associated unique Page number. Each Page has an independent 8 bit addressable PIF memory. In all the pages, the last address, FF, holds the current page number and is reserved for changing the page. The current page to be communicated with, can be set by writing the page number in hexadecimal form {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E} corresponding to pages {0, 1, 2, 3, 4, 5, 6, 7, A, B, C, D, E} in the address FF on any page.

- Page 0,1: Master System
- Page 2: Input System
- Page 3,4: Output System
- Page A: High Performance PLL A
- Page B: High Performance PLL B
- Page C: High Performance PLL C
- Page D: High Performance PLLD
- Page E: Frequency track PLL (FTPLL) for OCXO based tracking and auxiliary functionality
- Page 5: Input TDC System
- Page 6, 7: Clock Monitor System

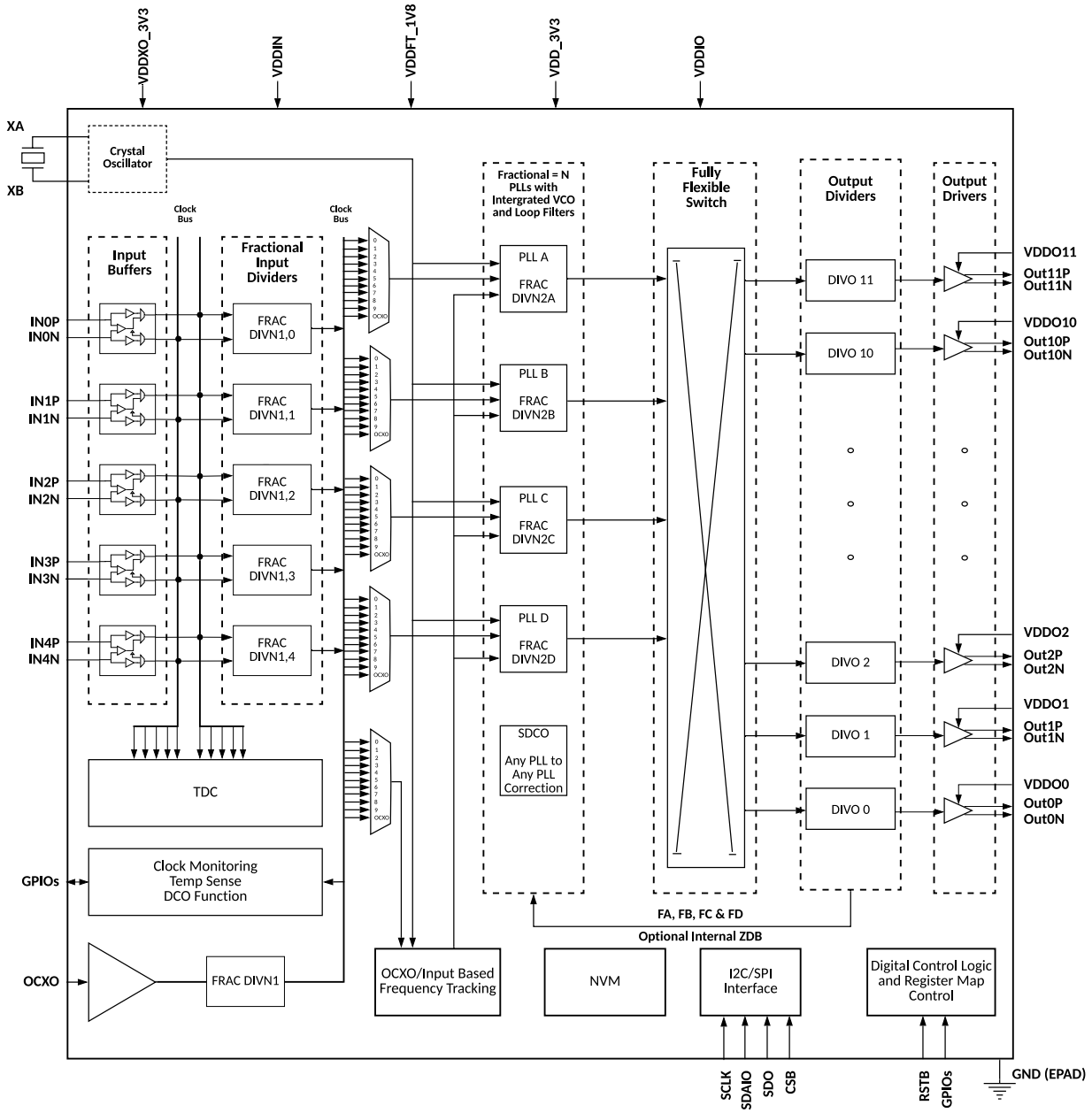



**Figure 5 AU5508 Functional Block Diagram**

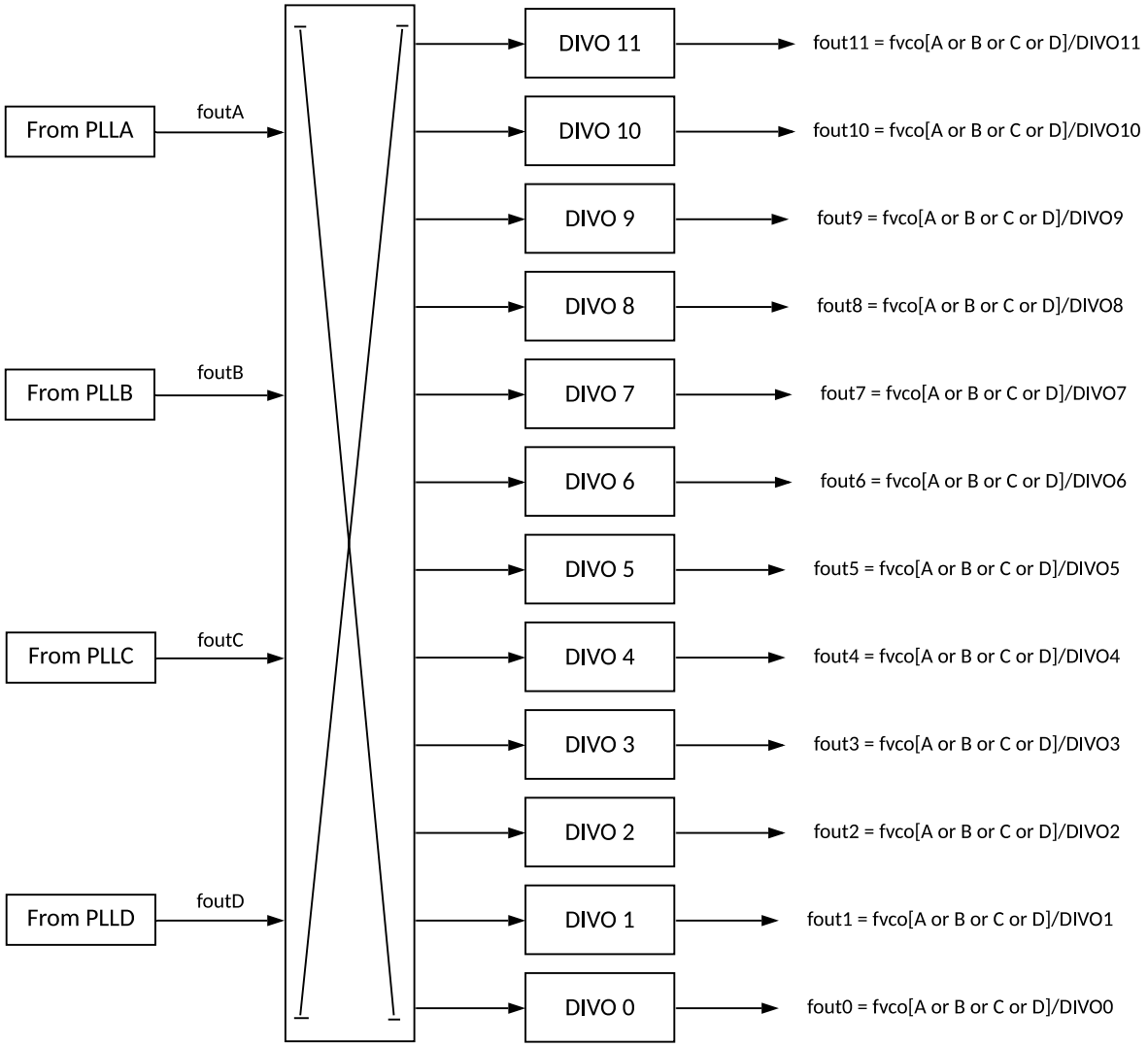



**Figure 6 Output Clock Distribution**

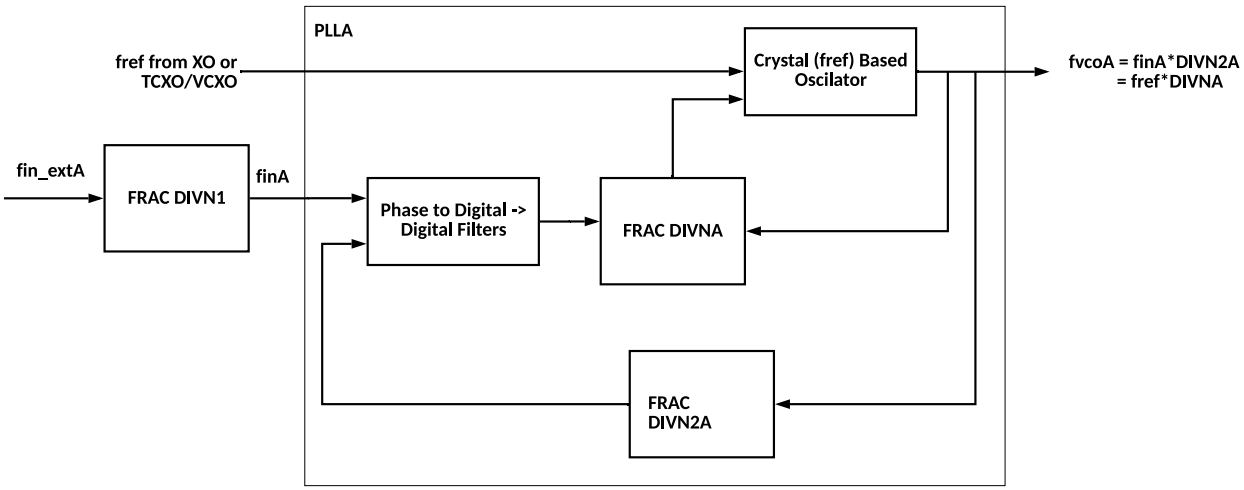



**Figure 7 PLL Internals**

Note: Similar for all PLLs, shown here for PLL A

**Table 15 PIF Description**

| Page | Contents | Summary of contents |
|---|---|---|
| 0 | Master | All Generic Information related to the chip<br>Chip Configuration details<br>Control for the master sequencer FSM<br>Crystal Reference Related Information |
| 1 | | |
| 2 | Input Slave | Input 4P,4N,3P,3N,2P,2N,1P,1N,0P,0N, OCXO related information<br>(Input type, DIVN1 divider configuration) |
| 3 | Output Slave | ODR 10,9,8,7,4,3<br>ODR Standards, DIVO, Programmable delay configurations for each |
| 4 | | ODR 11,6,5,2,1,0.<br>(ODR Standards, DIVO, Programmable delay configurations for each) |
| 5 | Input TDC Slave | Phase delay detection for 10 Pairs up to 10 different clocks |
| 6 | ClkMon Slave | Clock Loss related Function. Frequency Drift related function |
| 7 | | |
| A | PLL A Slave | All PLL related functionality |
| B | PLL B Slave | All PLL related functionality |
| C | PLL C Slave | All PLL related functionality |
| D | PLL D Slave | All PLL related functionality |
| E | FTPLL Slave | All FTPLL related functionality |

# 4 Master and Slaves: Architecture Description and Programming Procedures

## 4.1 Overview of the programming procedure

The AU5508 part can be programmed using different methods. The chip has a serial SPI/I2C port to access the Programmable Interface (PIF) that comprises of the register map. Additionally, key parts of that register map can be populated from the on chip one time programmable memories (NVM OTP eFuse memories) as well as from an on board I2C addressable EEPROM. The on board EEPROM is provided as an optional additional feature adding more flexibility to the programmable autonomous functions for the part at wake up. The chip can operate with all features being available without the on board EEPROM also.

The part can be used in one of the following three configurations:

**Configuration 1**: Completely autonomous wake up and reaching the final operational state with output clocks available with no SPI/I2C serial port access needed. In such cases there are three options available.

a. Option 1: This can be with a configuration that is loaded entirely in the on-chip OTP eFuse memory.
b. Option 2: Alternatively, this can be with a configuration that is loaded entirely in the on board I2C addressable EEPROM.
c. Option 3: Alternatively, this can be with a configuration that is loaded partly from the OTP and partly from the on board I2C addressable EEPROM. (All of the above three autonomous wake up options involve the use of the lock pattern that indicates an autonomous wake up. The use of the lock pattern for the master and slave memories is described in more detail in Section 4.2

**Configuration 2**: Partial autonomous wake up such that one PLL and a few outputs are available at wake up without any SPI/I2C serial port access while the remaining part of the chip is programmed using the SPI/I2C serial interface.

a. In this case, the on-chip OTP memory normally contains the partial wake up configuration that can enable selected PLLs and outputs. However, if the user desires such a partial wake up configuration can also be stored in and obtained from the on board EEPROM.
b. After the initial configuration available from fuse or EEPROM autonomously, the serial SPI/I2C ports can be used to program the remaining PLLs and outputs as well as enable more outputs from the PLL that was pre-programmed to provide the autonomous outputs at the wake up.

**Configuration 3**: In this case the chip has no output clocks available without the programming by the customer using the SPI/I2C serial port. Hence a custom profile is programmed by the user in the field using the serial SPI/I2C port. This is the most commonly used configuration for setting up the chip.

a. The chip can get partial settings from both the on-chip OTP and the optional on board EEPROM after which it is programmed with the desired configuration from the serial SPI/I2C interface. This feature of using partial configuration from the OTP memory and/or the optional EEPROM can still be done even if there is no output clock autonomously available.
b. Alternatively, the entire chip programming sequence is handled using the SPI/I2C profile. This will be the most commonly used setting within Configuration 3.

The on chip non-volatile memory (NVM) in every case contains basic configuration information that is recorded during the automatic test procedure in production. At every wake up, the part downloads the basic information from the on chip NVM OTP memory. After that, the chip then reaches its operational steady state condition in any of the three configurations listed above.

In case of I2C mode selection and no EEPROM is connected to AU5508, device will become I2C master and start searching for EEPROM after reset release. I2C bus will be engaged for ~ 3ms after reset/POR is released. CPU/FPGA should not initiate I2C transaction during that time period.

## 4.2 Master and Slaves: Architecture Description and Programming Procedures

The chip comprises of a Page based mechanism that is split between a Master page and several Slave Pages. As a first step to understand the wake up arrangement, the memory arrangement of the Master and Slaves is described.

The Master controller is the first system to autonomously wake up on the application of power to the chip due to on-chip power on reset circuitry. All generic system information resides in the Master controller memory and it proceeds to wake up the Slaves as required based on this information. The relative wake up sequences of the Master and the various Slaves are described in more detail after the description of the memory structures.

The Master memory structure is shown in Figure 8. It contains a one-time programmable non-volatile memory (NVM) that stores the settings for the chip associated with the master controller. The master controller also contains a volatile PIF bank (NVMCopy) that has an exact copy of the NVM at every chip power up. This volatile PIF (NVMCopy) is the memory that is addressable using the serial interface (I2C / SPI) on Page 0 and can be overwritten from the I2C / SPI interface. This volatile PIF can also be loaded with settings which are stored on external I2C EEPROM. Procedure to configure the device from external I2C EEPROM is described after description of wake up sequence of the Master and the various slaves. The “Chip Settings” is the memory space that is not addressable from the I2C / SPI / EEPROM control and is the actual control for the chip.

For all configuration wake ups where the chip is configured using the I2C/SPI interface, the autonomous wake up is not needed. For the cases where an autonomous wake up is desired, the lock pattern is used. The NVM contains a two bit “Lock Pattern” that can be set to “10” or “01” to ‘lock’ the chip configuration once the final configuration is determined and wake up of the entire chip is desired in this configuration using a completely autonomous wake up. Additionally, there is a bit in the NVM that is an active low indicator of a manual wake up. This bit set to “1” along with the ‘lock’ for the configuration leads to an autonomous wake up of the chip using the ‘locked’ configuration. Any number of different configurations can alternatively be tried at all times using only the volatile NVMCopy PIF section which is the default mode for the cases where an autonomous wake up is not desired. This is useful for evaluations as well as allowing real time programming of the chip in various configurations with complete flexibility in the field.

The Master Controller Finite State Machine (FSM) described later in this section controls the device behaviour in accordance with the configuration in this memory structure and as per the wake up mode.

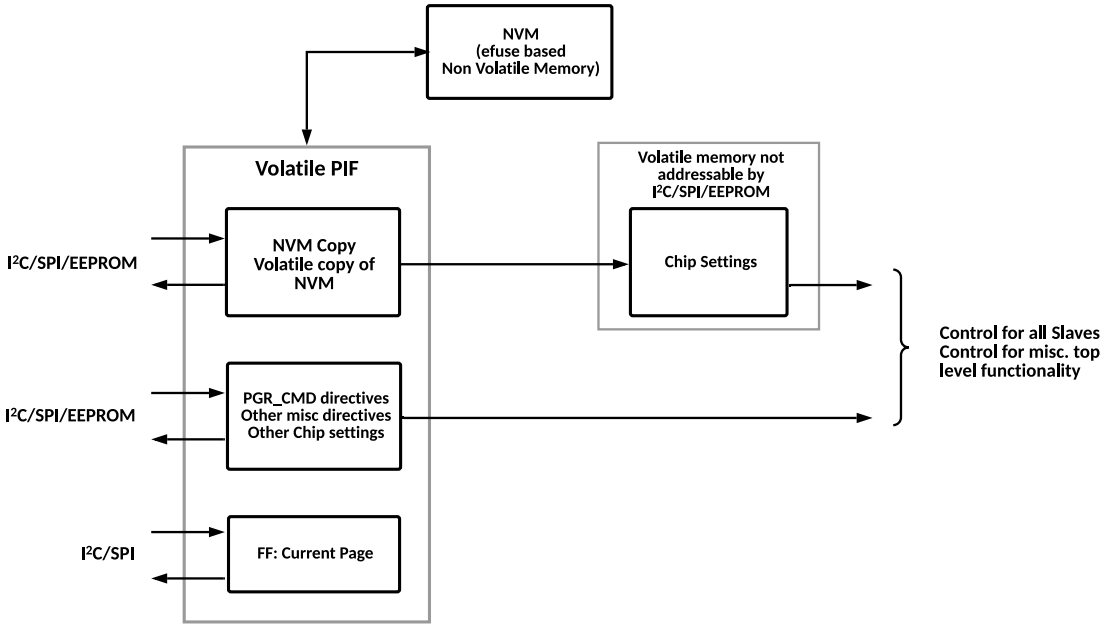



**Figure 8 Master Memory Structure**

The memory structure for each slave is shown in Figure 9 and is similar in construction to the master controller memory structure with some minor differences. The NVMCopy volatile PIF for the slave is addressable by the serial interface with the unique Page number associated with the slave. The “Slave Settings” is the memory space that is not addressable from the I2C / SPI / EEPROM control and is the actual control for the slave.

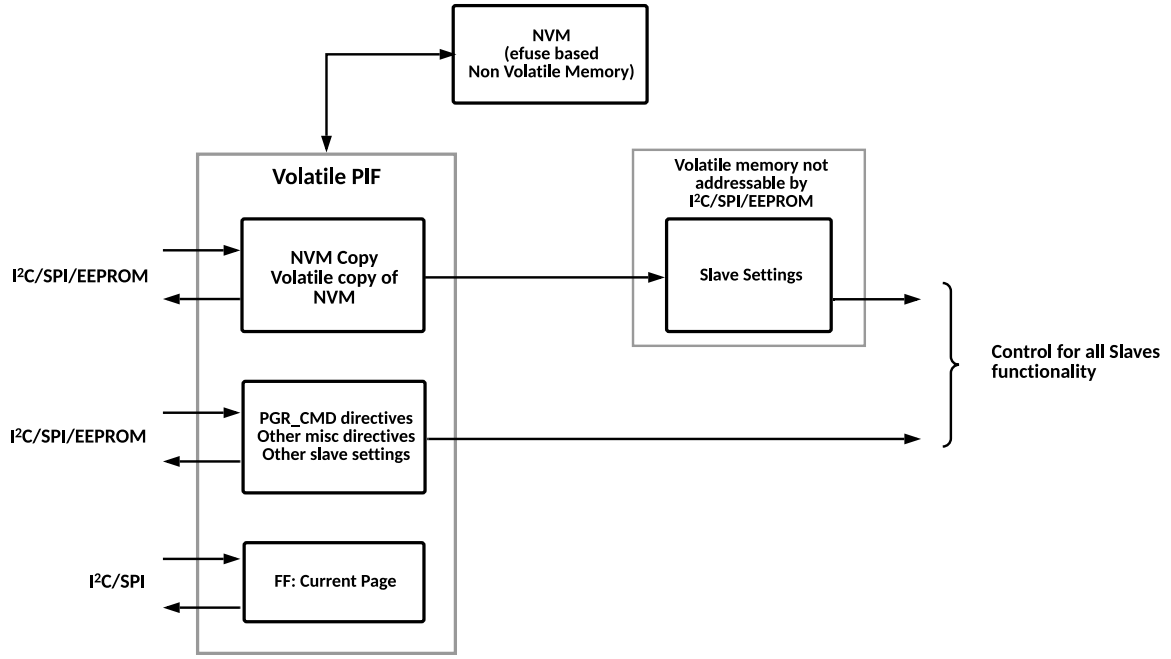



**Figure 9 Slave Memory Structure**

The Master Wake-Up Finite State Machine (FSM) is shown in more detail in Figure 8. At every power up of the device (or release from hard reset), the power-on-reset circuitry resets all systems and then autonomously releases only the master controller from reset.

Latching is disabled by default in AU5508. GPIO latching and its corresponding latching functions are factory programmed by Aurasemi with non 00 code marking. Please contact Aurasemi for more details.

GPIO can be configured as multiple user selectable functions. Input and output functions are defined in Section 4.2.3 GPIO Modes during Regular Operation.

When reset is applied available GPIOs will be configured as shown in Table 16. After releasing reset, the status of the GPIO pins will be latched. After hold time of 1ms GPIOs can release their reset levels and assume their normal operation modes. GPIO latching mechanism is described in Section 4.2.1 GPIO Latching.

The NVM contents are copied to the NVMCopy volatile space on Page 0 which is in turn copied to the “Chip Settings”. The master controller now re-configures the GPIOs according to the Chip Settings. The latching of the GPIOs at release of reset allows the chip to re-use the same GPIOs for other functions in the steady state hence making a more efficient use of the same resource. The master controller decodes the latched GPIO input using Chip Settings and configures the serial port. It also looks for I2C EEPROM access, if I2C EEPROM access is available and I2C mode is selected through latched value of GPIO9, then it becomes the I2C master and starts searching the configuration block in available EEPROMs.

One of the GPIOs can be factory progrmmaed as “EEPROM read done” output. No GPIO is configured for this purpose on AU5508. Please contact Aurasemi for more details. It is offered as a custom profile with a new part number.

If respective configuration block with Header and CRC is found in EEPROM then it re-configures the volatile PIF and Chip Settings with EEPROM data. After successful EEPROM read, the master notifies on a GPIO that the EEPROM read is done and becomes either SPI or I2C slave depending upon the configuration word read from the EEPROM.

If EEPROM read fails due to any error then it will notify respective error along with EEPROM read done indication and becomes SPI or I2C slave depending upon the Chip Settings. After EEPROM read done notification, status of EEPROM read can be checked by reading respective EEPROM error notify register or one of the available GPIO can be configured to show the EEPROM error notify output. If there is no EEPROM error notify along with EEPROM read done, it means EEPROM read is successful. If EEPROM read fails, the data inside volatile PIF may get corrupted so master does not update the Chip Settings; Master becomes either SPI or I2C slave depending upon the EFUSE Chip settings and proceeds to Program Command Wait State. Configuring Chip Settings from EEPROM is described later down the same section. If EEPROM access is not available or EEPROM read is successful, the master controller now decides if the chip configuration is locked and it is an autonomous wake up of the entire chip or if a manual wake up is desired through the PIF based on the contents in the “Chip Settings”.

With dedicated EEPROM for AU5508, typical successful readback happens within ~150ms. If there are CRC errors then it could go up to ~600ms.

In case a “Lock” is detected and an autonomous wake up is desired, the Master controller proceeds to enable the Crystal oscillator and associated fref pathways followed by the Slave systems in a pre-determined sequence. This finally leads the chip to the “Active State” with all desired outputs available as a result of all slave systems released from reset by the master controller. This is according to the requested settings that are programmed in the Master and the Slave NVM banks.

For the case where the final chip settings are not frozen hence the “Lock” pattern is not exercised, the master controller FSM reaches the Program Command Wait State (PRG_CMD). The desired chip settings can be written in the NVMCopy on Page 0 using the serial interface and desired slave sub-systems can be enabled. Several PRG_CMD state directives are available that are exercisable only in this state (refer Figure for the master Finite State Machine). Using these directives, the desired settings written in NVMCopy can now be copied to “Chip Settings” followed by issuing the directive for the FSM to proceed to the “Active” state where each slave can now be manually written with the desired settings and in turn asked to proceed to its “Active” state.

A similar “Lock” pattern is available in the NVM bank of each slave. The currently used NVM bank for a slave can be locked for the autonomous wake up of each slave. The slave wake-up FSM is shown in Figure 11 and it similarly has a PRG_CMD state with associated directives. On Proceed to Active state directive on the slave, the slave controller wakes up the various blocks in its sub-system with the correct pre-determined sequence.

The NVM bank for the master and each slave can be programmed with a PRG_CMD directive in that state to lock a configuration / setting specific to the respective sub-system.

Each FSM (Master and Slaves) allows an escape sequence to go back to PRG_CMD state from its Active State. This can be used to selectively change the settings for that particular sub-system.

Note that the NVM for the master controller and for all slaves should be locked after writing desired settings for a completely autonomous wake up of the entire chip. For evaluations of the chip as well as cases where flexible on-the-fly programmable settings are desired, the chip can be used without engaging the NVM banks at all by using the NVMCopy space for the master and each slave in conjunction with the PRG_CMD directives. It is also possible to lock some of the slaves (to not re-write their settings for each wake up) while use programmable settings for other slaves.

This provides complete flexibility in terms of programming and using the chip in all scenarios.

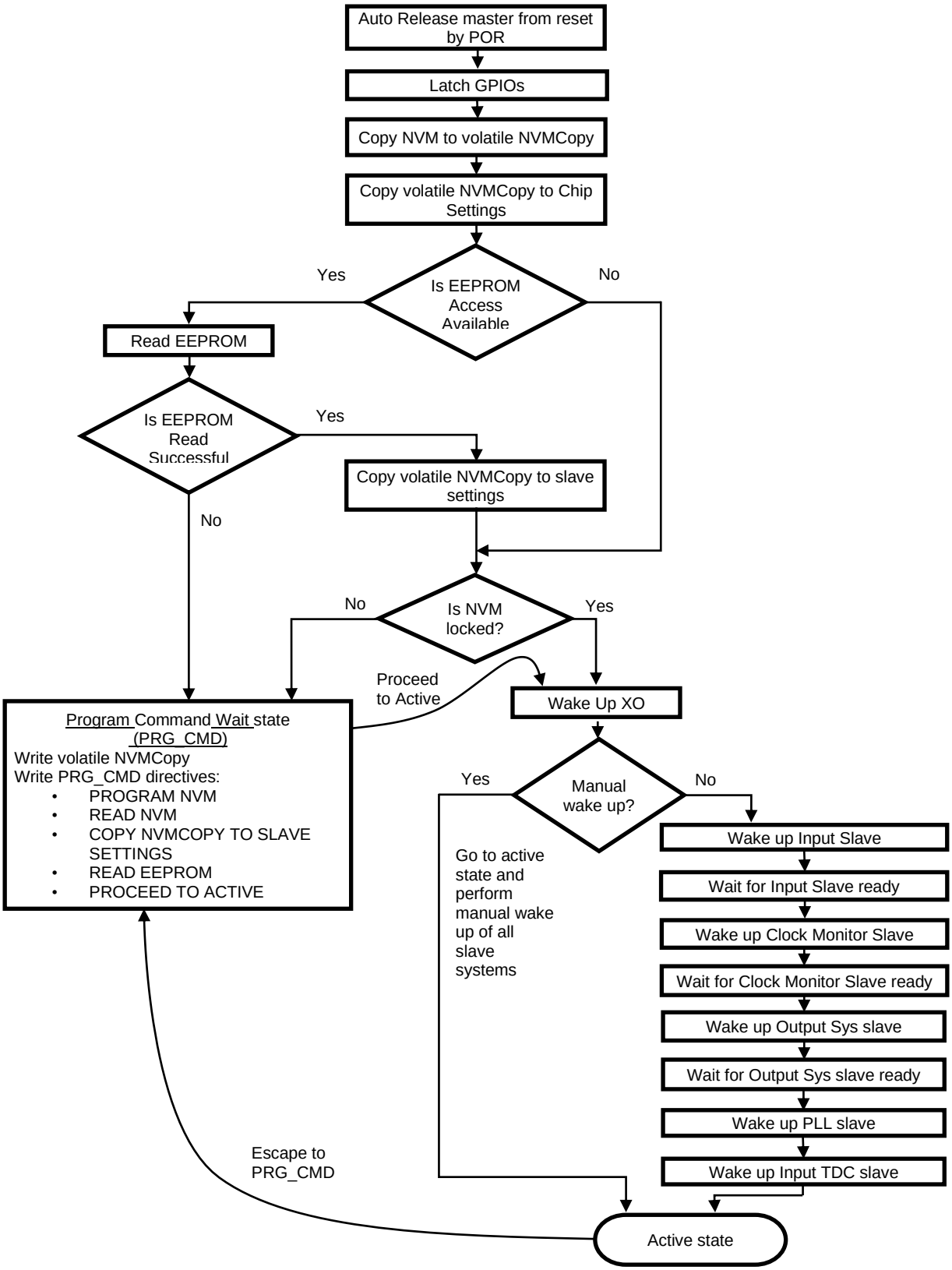



**Figure 10 Master Wake-Up Finite State Machine**

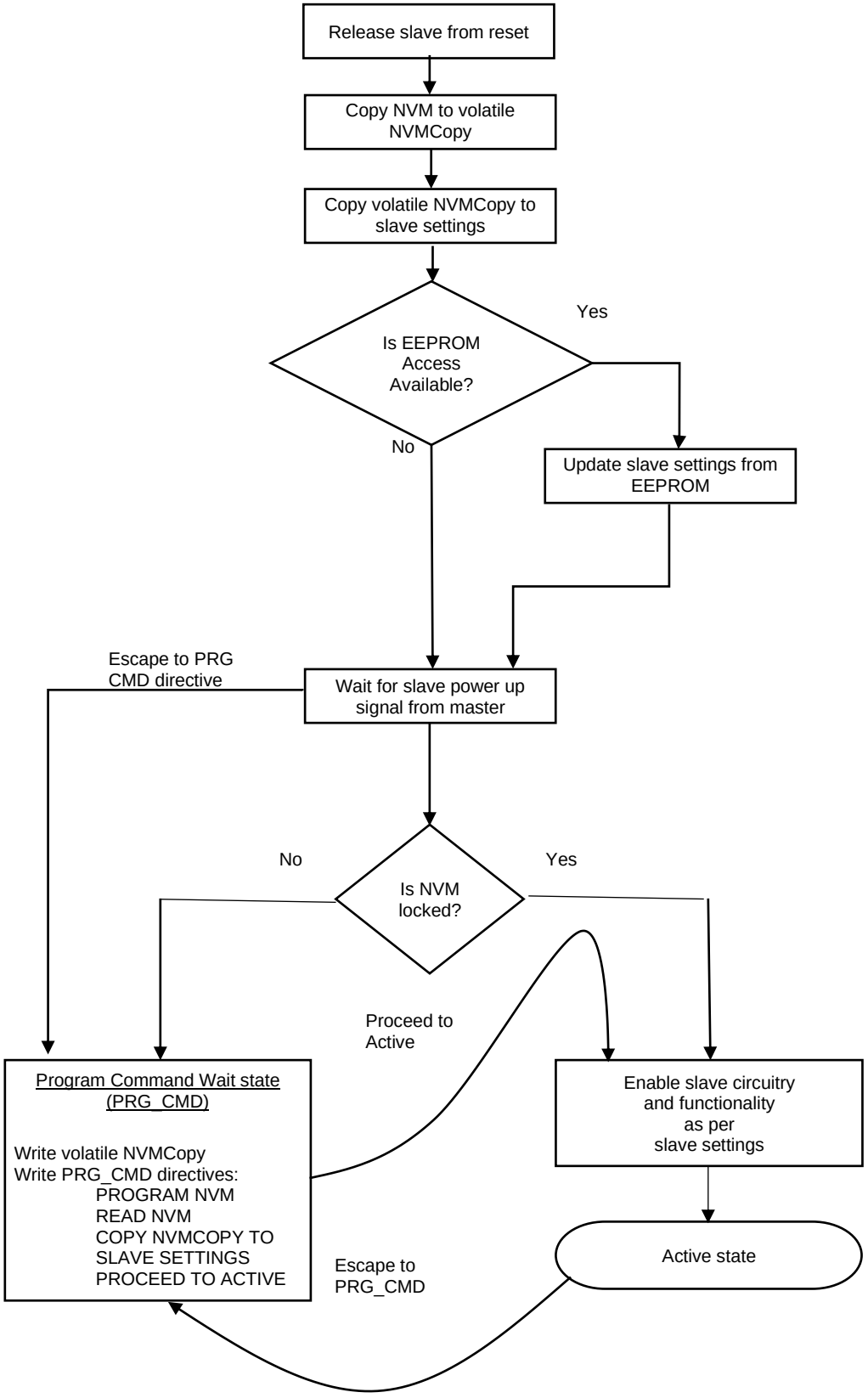



**Figure 11 Slave Wake-Up Finite State Machine**

### 4.2.1 GPIO Latching

When reset is applied available GPIOs will be configured as shown in Table 16. After releasing reset, the status of the GPIO pins will be latched. After hold time of 1ms GPIOs can release their reset levels and assumes their normal operation modes. Latched value of GPIOs will be decoded later when NVM contents are copied to Chip Settings. Using GPIO latch configuration from Chip Settings, latched value of GPIOs will be decoded as shown in Table 17. GPIO9 latch has static configuration for I2C / SPI slave selection. The use of GPIO latching ensures

optimal use of resources by making the same GPIOs that are latched at wake up available for other alternate functions during regular operation.

**Table 16 GPIO Latching Configuration**

| GPIO | Pin | Default Direction | Default Pull Up / Pull Down | Function | Description |
|---|---|---|---|---|---|
| GPIO0 | 71 | Input | Weak Pull Down | User Defined | User can define the functionality of these GPIOs using GPIO latch configuration as shown in Table 17. |
| GPIO3 | 26 | Input | Weak Pull Down | User Defined | |
| GPIO9 | 34 | Input | Weak Pull Up | I2C / SPI Mode Select | 0 : SPI Mode<br>1 : I2C Mode |
| GPIO1 | 49 | Input | Weak Pull Up | User Defined | User can define the functionality of these GPIOs using GPIO latch configuration as shown in Table 17. |
| GPIO2 | 47 | Input | Weak Pull Up | User Defined | |
| GPIO4 | 63 | Input | Weak Pull Up | User Defined | |
| GPIO5 | 56 | Input | Weak Pull Up | User Defined | |

**Table 17 GPIO Latch Decoding**

| gpio_latch_cfg[2:0] | Function | Description |
|---|---|---|
| 3'd0 | Unused default | If any GPIO is configured with this settings then latched value will be ignored |
| 3'd1 | EEPROM access disable [1] | Disable EEPROM access during Wakeup sequence<br>0 : If I2C mode selected using GPIO9 device will try to read the configuration block from external EEPROM.<br>1 : Device will not attempt to read the configuration block from external EEPROM.<br>By default, no GPIO is used for this purpose, so the device will try to read the configuration block from external EEPROM. |
| 3'd2 | I2C slave address A2 [1] | If I2C mode is selected using GPIO9, will be used as I2C slave address A2. In SPI mode latched value will be ignored.<br>By default, no GPIO is used for this purpose, so default I2C slave address will have a 0 for bit A2. |
| 3'd3 | SPI mode 3wire 4wire [1] | If SPI mode is selected using GPIO9, will be used as control for 3-wire/4-wire mode of SPI. In I2C mode latched value will be ignored.<br>0 : 4wire mode<br>1 : 3wire mode<br>By default, no GPIO is used for this purpose, so device will be in 4-wire SPI mode. |
| 3'd4 | Spare | Reserved |
| 3'd5 | Spare | Reserved |
| 3'd6 | Spare | If any GPIO is configured with this settings then latched value will be ignored |
| 3'd7 | Spare | If any GPIO is configured with this settings then latched value will be ignored |

By default GPIO latching function is not enabled on AU5508. GPIO latching and its corresponding latching functions are factory programmed by Aurasemi with non 00 code marking.Please contact Aurasemi for more details.

Using GPIO latch configuration any of the GPIO from GPIO0 to GPIO5 can be used as following:

- EEPROM Access Disable control: A high input value on a GPIO programmed with this function prevents a device from attempting to read configuration data from an external I2C EEPROM. If no GPIOs are configured in this mode then the device will attempt read configuration data from an external I2C EEPROM. If multiple GPIOs are configured to perform this function then any one of them being active will disable EEPROM accesses, so it is recommended that no more than one GPIO be programmed for this function.
- I2C Address bit A2: If a GPIO is configured to this function This is for the serial port when selected as I2C during the start-up sequence using GPIO9. If no GPIOs are configured in this mode, bit A2 of the

slave serial port base address will be zero. If more than one GPIO is programmed with this functionality, only the one with the highest index will be used (e.g., if both GPIO5 and GPIO1 are programmed to do this, only GPIO5 would be used).

- SPI Mode 3-Wire/ 4-Wire: If device is configured as SPI slave using GPIO9, a high input value on a GPIO programmed with this function set the device in 3-wire SPI mode. If no GPIOs are configured in this mode, then the device will be set to 4-wire SPI mode. If multiple GPIOs are configured to perform this function then any one of them being active will set the device in 3-wire SPI mode, so it is recommended that no more than one GPIO be programmed for this function

### 4.2.2 Status versus Notify available on the chip

AU5508 provides various Status and notify bits that can be accessed from the register map. Below are the details of the procedure to be followed to access the same.

The alarm registers are a set of three types of registers distributed between the various pages as described in Table 32.

- The Status registers are the current dynamic status of a defect. The live status defects are active high with a '1' indicating the defect is present.
- The Notify registers are the sticky bits for a defect. Sometimes, we may get a very short pulse for the status and it may not be possible for the external user to capture the same. Hence a notify register is provided. The notify register is set to 1 whenever there is a rising edge of the corresponding status register. This is a sticky bit and stays at 1 till the user writes a 1 to that specific bit to clear it.
- Each notify has a masking bit to enable or disable its operation. The notify sticky bit operates only if the corresponding masking bit is set to 1. If the masking register bit is set to 0, notify will not be asserted even when status toggles. The default value for the mask register is 0xff so all the notify signals are enabled. Once the user writes a 1 to clear the notify, the notify bit can again go high on the next rising edge of the status.

These registers operate on the internal 4 MHz RC clock. When there is a defect (i.e. status) of any bit in register it gets asserted and de-asserted in a live mode. User can read the corresponding register location to see the current status at any time.

On the other hand, the default value of the notify and masking register is 0xff – user has to write 0xff to both these registers to clear them at the beginning and use all notifies.

### 4.2.3 GPIO Modes during Regular Operation

Using Chip Settings, all the available GPIOs (GPIO0 to GPIO5 and GPIO9) can be configured to one of the following modes. Also each GPIO can be pulled up, pulled down or set to high impedance state individually using respective Chip Settings. GPIOs will be configured to respective mode in master wake up sequence when NVM content is copied to Chip Settings. This is the regular steady state operational mode for the GPIO which can be configured differently compared to the functionality that is latched at the release of reset. GPIOs can also be reconfigured with settings from EEPROM or from the volatile register writes during chip configuration. All the GPIOs are fully configurable so that any GPIO can perform any function of the following:

- Input Modes:
  - General Purpose Input - In this mode of operation, the GPIO pin will act as an input that is latched to specified internal register. That register can be read over the serial port.
  - DCO Select – This is used such that the PLL that is used for DCO can be selected. The same function is available directly from the register map as well alternatively.
  - DCO Increment – A low to high transition on the DCO increment pin causes a DCO increment on the selected PLL. The same function is available directly from the register map as well alternatively.
  - DCO Decrement - A low to high transition on the DCO decrement pin causes a DCO decrement on the selected PLL. The same function is available directly from the register map as well alternatively.
  - Output Disable Control – This is an OEb pin such that it can be used as an active high to disable all the outputs.
  - External Clock Input Switch – An external clock mux that is used on the board to select and switch the active clock for a PLL is often used. For such cases the selection for the mux can be sent to the PLL to ensure a clean clock switching transient even for cases where the clock mux is external and located on the PCB
  - SYS ref – SYSREF trigger related to the JESD204B function
  - Divider_restart – Divider restart can be used to dynamically change the relative delays between outputs from the same PLL where the delays are pre-loaded in the register map using the serial port and this pin is used as a GPIO trigger for the same

- PLL Syncb Select –Selects which PLL to be used for the SYNCB trigger
- PLL Syncb trigger – The actual SyncB trigger that is used for the selected PLL
- Phase Adjustment Trigger – This is used as a trigger to sync the output clock to an input clock within 2 cycles of input clock. It is useful for 1 pps clocks.
- Manual Clock Select: GPIO can be used as manual clock selection for PLLs
- Clock Disqualification: GPIO can be assigned as trigger to disqualify selected clock as active clock to selected PLL.

- Output Modes : GPIO0 to GPIO5 and GPIO9 can be in individually configured for bellow outputs. GPIO output polarity is programmable.
  - General Purpose Output - In this mode of operation, the GPIO pin will act as an output that is driven to the logic level specified in an internal register. That register can be written over the serial port.
  - TDC Data Ready Status – PLL PFD TDC data ready status can come out of GPIO. It is useful in external software filter mode.
  - Alarm Signal Outputs - Each GPIO can be independently configured to act as a Single-Purpose alarm or Aggregated alarm output. If multiple GPIOs are configured the same configuration settings, they will all have the same output values.
  - Clock Loss - In this mode the GPIO will act as Clock Loss output. There are three types of Clock Loss indication as
    - Clock loss - status as well as sticky notify bit can be give out on GPIO.
    - Frequency Drift - status as well as sticky notify bit can be given out on GPIO.
    - Clock loss with frequency drift - Only status bit can be given out on GPIO.
  - Loss of Lock Status - In this mode the GPIO will act as loss of lock indicator of respective PLL. There are two types of loss of lock indicators as
    - Inner Loop Loss of lock - Status as well as sticky notify bit can be give out on GPIO.
    - Outer Loop Loss of lock - Status as well as sticky notify bit can be give out on GPIO.
    - Phase loss of lock - Status as well as sticky notify bit can be give out on GPIO.
  - Holdover Freeze - In this mode the GPIO will act as Holdover indicator, which indicates that respective PLL is entered in Holdover mode. Status as well as sticky notify bit can be give out on GPIO.
  - EEPROM Status - In this mode the GPIO will act as EEPROM Status output to send out EEPROM defects. Status as well as sticky notify bit can be given out on GPIO.
  - EEPROM Read Done Indicator – In this mode GPIO will act as EEPROM read done Indicator to indicate that EEPROM read is done and AU5508 became the slave. External micro controller can check the EEPROM status whether EEPROM read was successful or not. If there is no EEPROM defect then EEPROM read is successful.
  - Aggregated Alarm output - In this mode the GPIO will act as the logical OR of all alarm indicators that are enabled to drive this output. Only sticky bits are available to drive the GPIO in this mode. This output will be asserted if any of the “sticky” bits are asserted and enabled to cause the Alert (aggregated alarm). To clear this output, all contributing “sticky” bits must be individually cleared. This output will be active-high to indicate one or more alarms are asserted.

### 4.2.4 Configuring the Device from External EEPROM

AU5508 can be configured from external EEPROM. During Wakeup sequence master checks for EEPROM access. AU5508 initiates reading the configuration block from EEPROM if serial interface is configured to I2C mode.

If EEPROM access is available and $I^2C$ mode is selected, one of the GPIO can be configured to send out the EEPROM read done signal and the Master will start searching the configuration block in available EEPROMs with 400 KHz $I^2C$ frequency and with EEPROM device address 7’b1010000. If there is no acknowledge from EEPROM then the Master increments EEPROM device address to 7’b1010001 and repeats the search. This will be repeated up to EEPROM device address is 7’b1010111. If there is still no acknowledge, it will repeat the search with 100 KHz $I^2C$ frequency. If EEPROM controller receives acknowledge from the EEPROM then it starts reading the data from address 0x00000 of that EEPROM. While reading the data, it proceeds with checking the EEPROM size, device id, page id, page size and crc id. If mismatch occurs it sets the respective defect flag and starts looking for next available block or EEPROM until it gets valid configuration block. While reading the configuration data from EEPROM Master starts calculating the CRC, if calculated CRC matches with the received CRC, then all defect flags will be de-asserted and EEPROM read done indication will be sent out on the configured GPIO. If configuration block is not found in all possible data blocks or there is no acknowledge from EEPROM then sticky notifies for respective defect will be set and EEPROM read done signal will be sent out on configured GPIO. Once outer controller receives EEPROM read done signal it can look for the defect notifies, if none of the notifies are set then EEPROM read is successful and master proceeds to next step.

EEPROM data needs to be organized as shown in Table 18

AU5508 configuration data needs to be organized in blocks of 64Kb. One 64Kb block will have configuration data for one AU5508 device. EEPROM can contain multiple configuration data blocks for multiple devices depending upon the size of EEPROM.

**Table 18 EEPROM Data Organization**

| word_address | EEPROM data | size |
|---|---|---|
| 0x00000 | EEPROM Size | 64 Kb block |
| 0x00001 | Device ID | |
| 0x00002 | Config Word | |
| 0x00003 | Page ID | |
| 0x00004 | Page Size | |
| 0x00005 | Page Register Address | |
| 0x00006 | Page Register Data | |
| 0x00007 | Page Register Address | |
| . | Page Register Data | |
| . | . | |
| . | . | |
| . | Page ID | |
| . | Page Size | |
| . | Page Register Address | |
| . | Page Register Data | |
| . | Page Register Address | |
| . | Page Register Data | |
| . | . | |
| . | . | |
| . | CRC ID | |
| . | crc[3] | |
| . | crc[2] | |
| . | crc[1] | |
| . | crc[0] | |
| . | . | |
| 0x01FFF | . | |
| 0x02000 | Device ID | 64 Kb block |
| 0x02001 | Config Word | |
| 0x02002 | Page ID | |
| 0x02003 | Page Size | |
| 0x02004 | Page Register Address | |
| 0x02005 | Page Register Data | |
| 0x02006 | Page Register Address | |
| 0x02007 | Page Register Data | |
| . | . | |
| . | . | |
| . | Page ID | |
| . | Page Size | |
| . | Page register Address | |
| . | Page register Data | |
| . | Page register Address | |
| . | Page register Data | |
| . | . | |
| . | . | |
| . | CRC ID | |
| . | CRC[3] | |

| word_address | EEPROM data | size |
|---|---|---|
| . | CRC[2] | |
| . | CRC[1] | |
| . | CRC[0] | |
| . | . | |
| 0x03FFF | . | |

EEPROM size :

0x00 -> 64 Kb EEPROM, 1 block of 64Kb.

0x01 -> 128 Kb EEPROM, 2 blocks of 64Kb.

0x02 -> 256 Kb EEPROM, 4 blocks of 64Kb.

0x03 -> 512 Kb EEPROM, 8 blocks of 64Kb.

0x04 -> 1024 Kb EEPROM, 16 blocks of 64Kb.

If block's starting address is 0x00000 then it should contain EEPROM size at location 0x00000, else if block's starting address is other than 0x00000 then it should not contain EEPROM size. Table 19 shows available data blocks with their starting address in EEPROM.

**Table 19 EPPROM Data Blocks**

| Blocks | starting address | EEPROM Size | | | | |
|---|---|---|---|---|---|---|
| | | 64Kb | 128Kb | 256Kb | 512Kb | 1024Kb |
| 1 | 00000 | yes | Yes | yes | yes | yes |
| 2 | 02000 | no | Yes | yes | yes | yes |
| 3 | 04000 | no | No | yes | yes | yes |
| 4 | 06000 | no | No | yes | yes | yes |
| 5 | 08000 | no | No | no | yes | yes |
| 6 | 0A000 | no | No | no | yes | yes |
| 7 | 0C000 | no | No | no | yes | yes |
| 8 | 0E000 | no | No | no | yes | yes |
| 9 | 10000 | no | No | no | no | yes |
| 10 | 12000 | no | No | no | no | yes |
| 11 | 14000 | no | No | no | no | yes |
| 12 | 16000 | no | No | no | no | yes |
| 13 | 18000 | no | No | no | no | yes |
| 14 | 1A000 | no | No | no | no | yes |
| 15 | 1C000 | no | No | No | no | yes |
| 16 | 1E000 | no | No | No | no | yes |

- **Device ID :**
  Device ID field indicates to which AU5508 device this configuration block belongs. This device id will be read by Master and compared with $I^2C$ device address of AU5508 device, if device id matches configuration data from this block will be read otherwise EEPROM controller will start looking for next blocks.
- **Config Word:**
  Config Word field contains configuration data. After successful EEPROM Read, device will be configured with respective settings. Config_Word[0] is used to reconfigure the serial interface after successful EEPROM read.
  If Config_Word[0] = 0 then AU5508 will be configured as SPI slave.
  If Config_Word[0] = 1 then AU5508 will be configured as I2C slave.
- **Page ID :**
  Page ID field indicates to which page the configuration data belongs. This page id will be read by Master and respective page will be enabled to write the configuration data.
- **Page Size :**

Page Size field contains a number which indicates how many registers of respective page needs to be configured from EEPROM.

- **Page Register Address & Page Register Data :**
  Page Register Address contains register address of respective page in which configuration data i.e. Page Register Data has to be written.
- **CRC ID :**
  CRC ID indicates that following four bytes (CRC[3],CRC[2],CRC[1] and CRC[0]) are the 32 bit CRC for respective configuration block.

## 5 Crystal Pathway Connectivity Options

The CMOS XO/TCXO output and the termination components should be placed as close as possible to the XA/XB pins

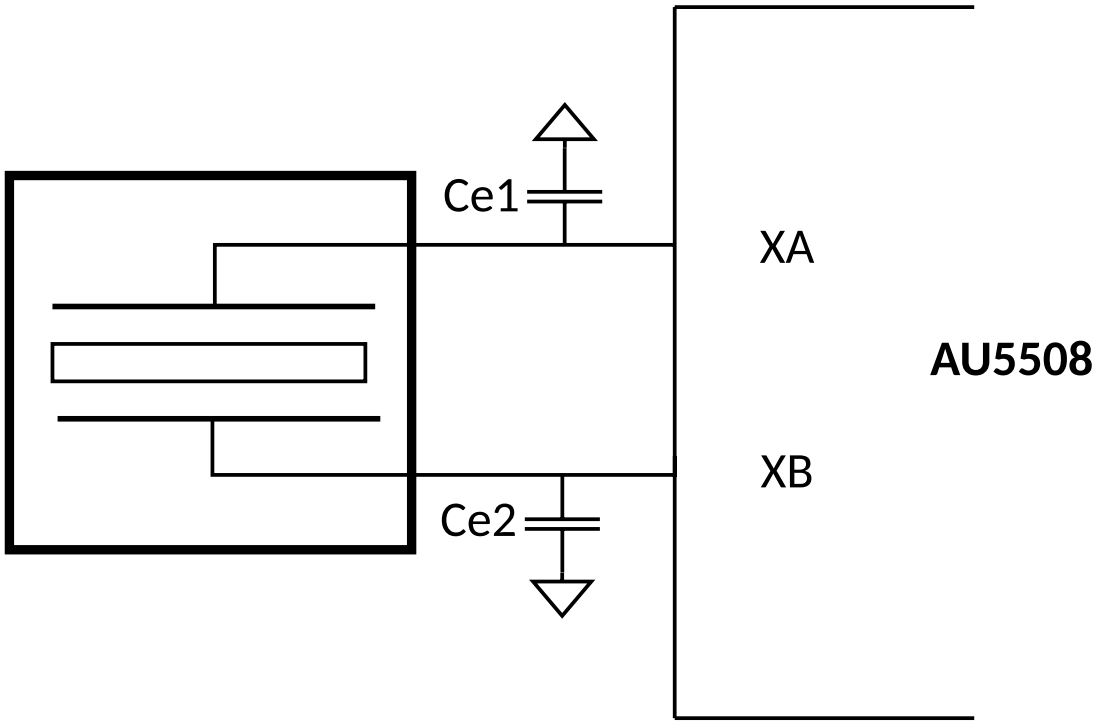



**Figure 12 Crystal Connections**

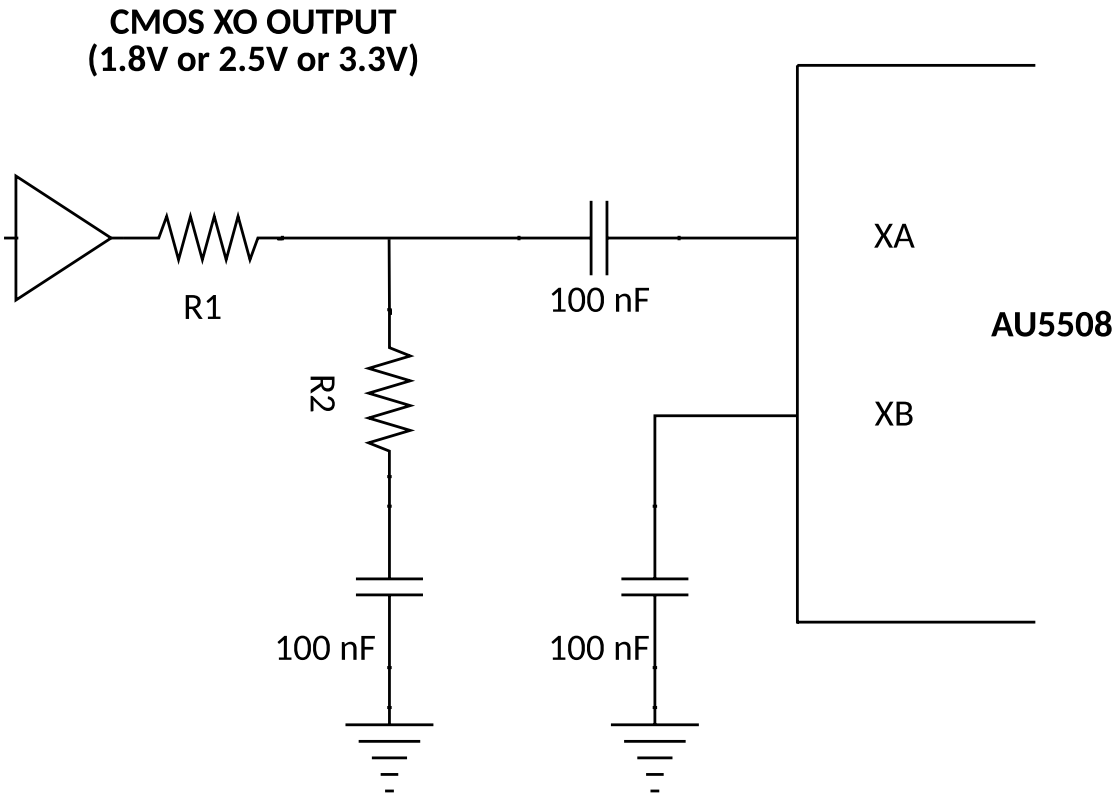


**Figure 13 CMOS XO Connections**

| CMOS XO Driver Supply | R1 (Ohms) (Includes 50ohm driver impedance) | R2 (Ohms) |
|---|---|---|
| 1.8 V | 0 | DNP |
| 2.5 V | 140 | 360 |
| 3.3 V | 230 | 275 |

Note :
5pF can be placed across R1 to improve the slew rate if PCB routing length is significant from the attenuator network to the XA pin.

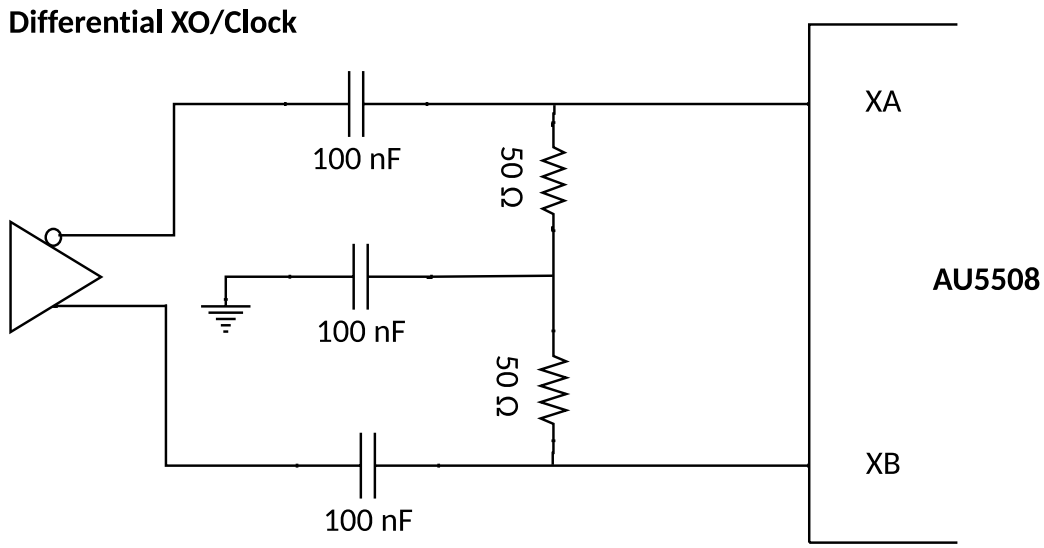



**Figure 14 Differential XO Connections**

# 6 Input Slave Description

5 independent differential clock inputs (or 10 independent single ended clock inputs) are available on the chip that can be routed to any PLL with complete flexibility. In addition to the 5 differential or 10 SE inputs, one OCXO SE input is also available which can be configured as DC or AC coupled input. Input divider can be configured in OCXO input path.

Both DC and AC coupled clock inputs are possible. The input clock receiver settings (to receive a single ended or differential clock) as well as the input clock divider settings are configurable on Page 2 that is assigned to the Input Slave. It is possible to bypass the input clock divider and use the input clock directly as an input to the PLL.

The various input termination interface options available for AU5508 are shown in this section.

Apart from the above given interface examples, AU5508 differential input buffer can accept 3.3 V/2.5 V domain universal input clock standards like LVPECL, LVDS, CML and other differential signals that meet input common mode voltage, slew rate and swing requirements specified in Table 4.

Differential buffers supports a wide common mode and input signal swing.

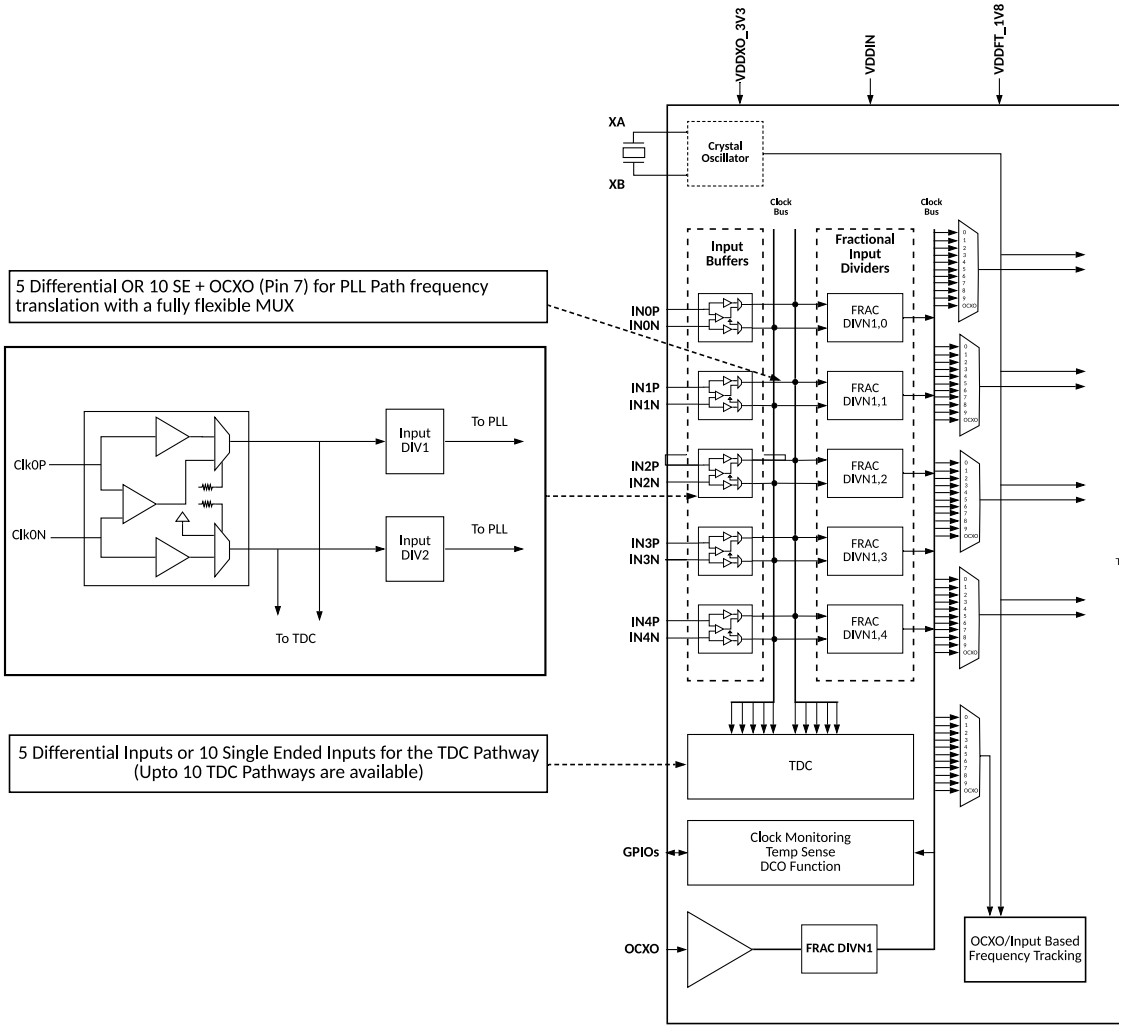



**Figure 15 AU5508 Input Slave Architecture**

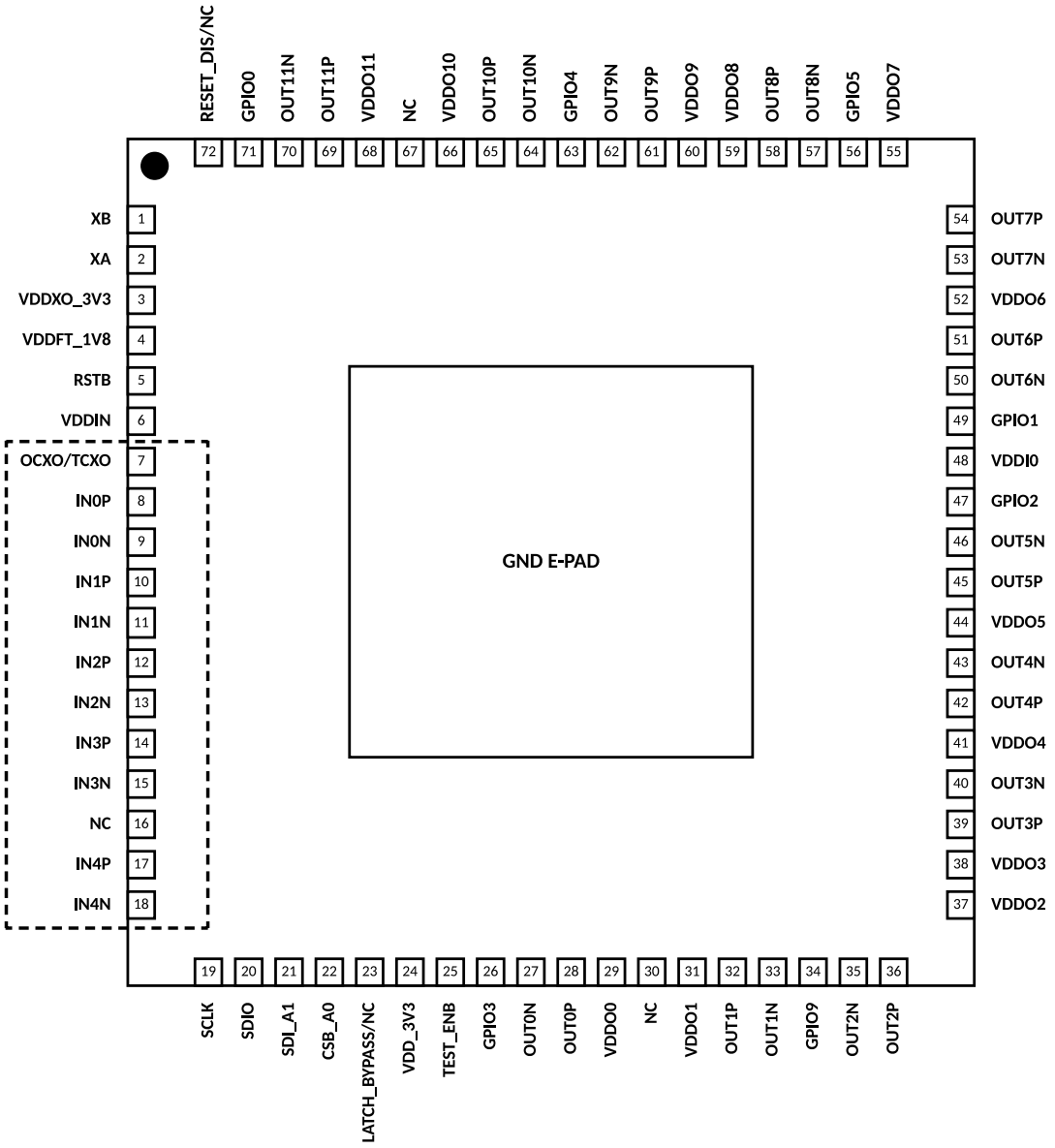



**Figure 16 5 Differential / 10 SE Inputs + OCXO SE Input**

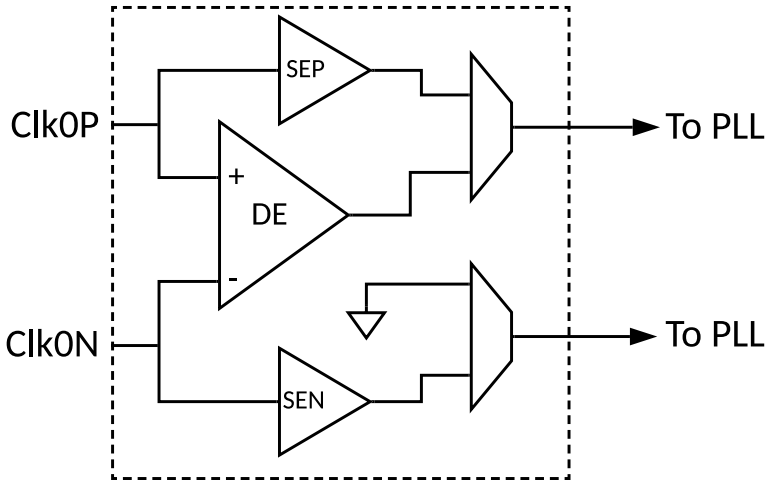
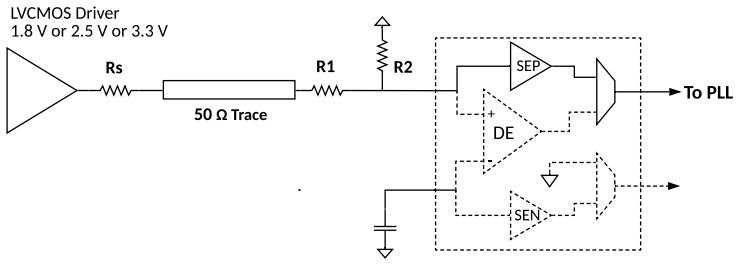



**Figure 17 Input Buffer Structure**

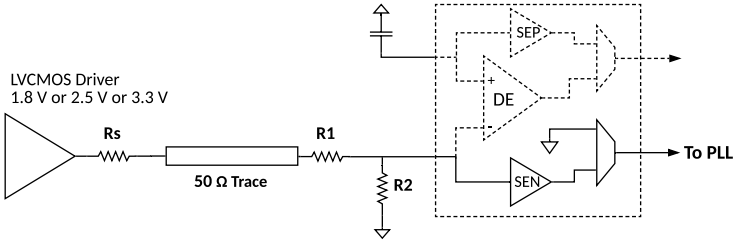
Note; DE – Differential Buffer; SEP – Single Ended Buffer at CLKP; SEN – Single Ended Buffer at CLKN

**Interface Type 1: SE-DC (Single Ended--DC Coupled)**

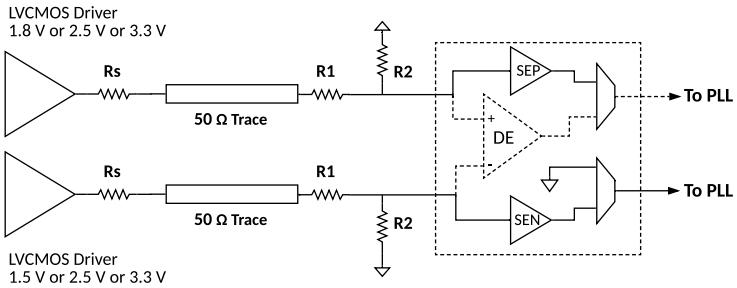
**1A: Only at CLKP**



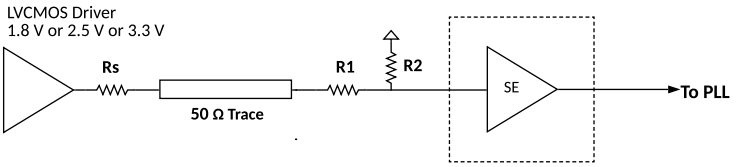

**1B: Only at CLKN**




**1C: At both Inputs CLKP and CLKN**




**1D: OCXO Input**




| Driver Supply | R1 (ohm) | R2 (ohm) |
|---|---|---|
| 1.8 V | 140 | 665 |
| 2.5 V | 325 | 475 |
| 3.3 V | 445 | 365 |
| Rs is resistance to match Ro_driver + Rs = 50 ohm | | |

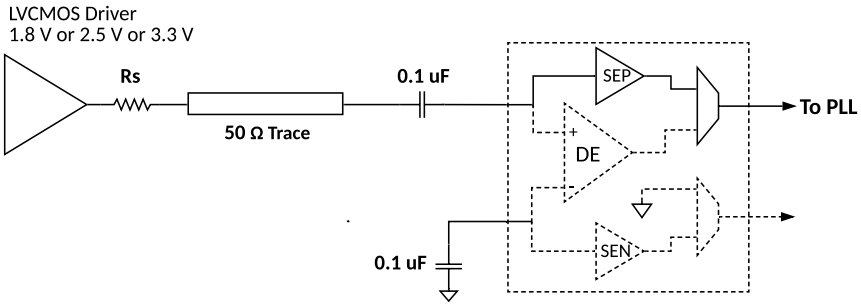
**Figure 18 Interface Type-I: Single Ended DC Coupled**

Note:

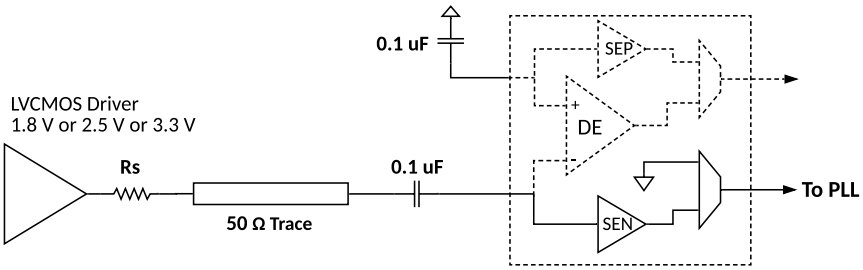
- R1/R2 Termination is recommended for optimum performance. Input interface works without divider for VDDIN equal or more than LVCMOS Driver supply.
- Interface connections are recommended for both OCXO and CLKIN.

**Interface Type 2: SE-AC-Direct (Single Ended—Direct AC Coupled)**

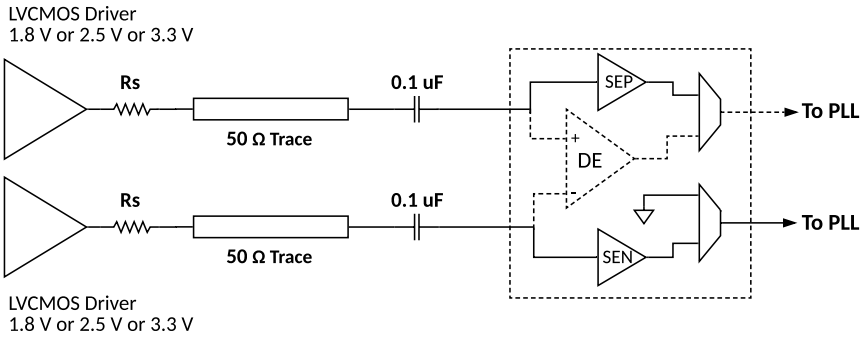
**2A: Only at CLKP**



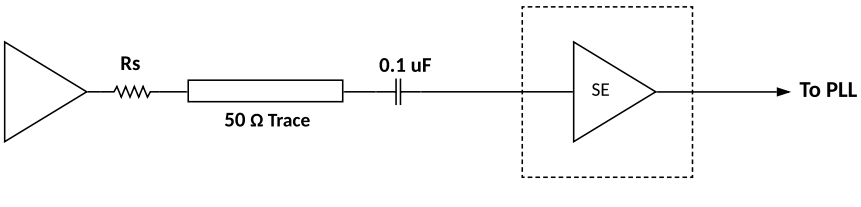

**2B: Only at CLKN**




**2C: At both Inputs CLKP and CLKN**




**2D: OCXO Input**



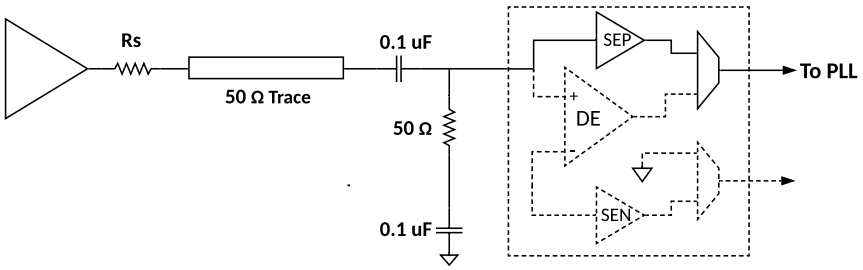

Rs is resistance to Match; R0_Driver + Rs = 50 ohms

**Figure 19 Interface Type II Single Ended Direct AC Coupled**

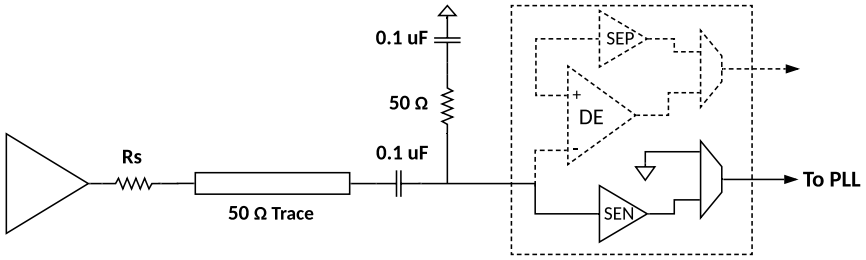
Note: Interface connections are recommended for both OCXO and CLKIN.

**Interface Type 3: SE-AC-Direct with 50 Ω Resistor Termination**

**3A: Only at CLKP**




**3B: Only at CLKN**




**3C: OCXO Input**

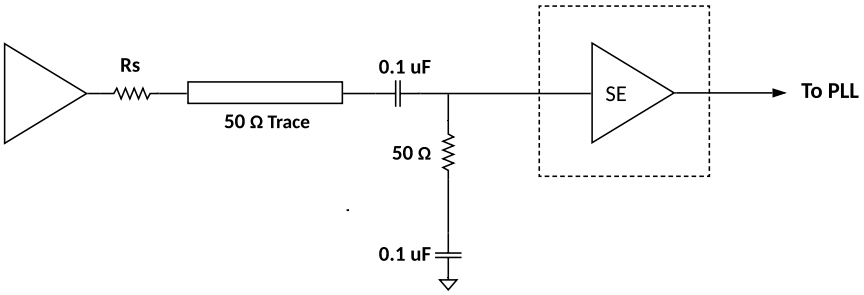



**Figure 20 Interface Type III SE AC Coupled with 50 ohm termination**

**4A**

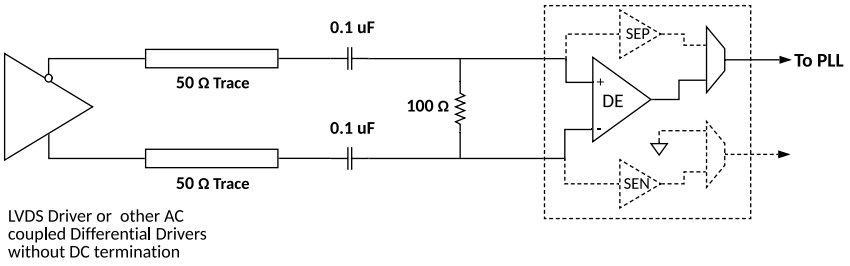



**Figure 21 Interface Type IV Differential AC Coupled 4A**

4B

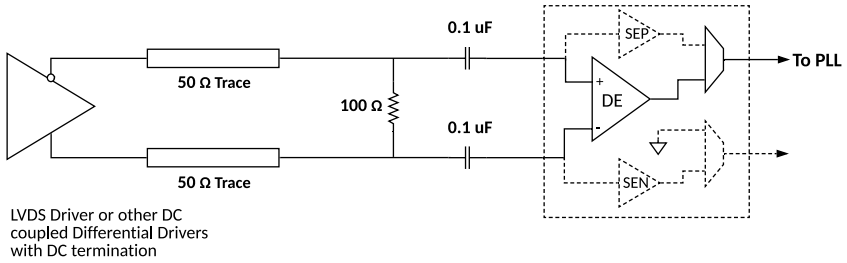



**Figure 22 Interface Type IV Differential AC Coupled 4B**

4C

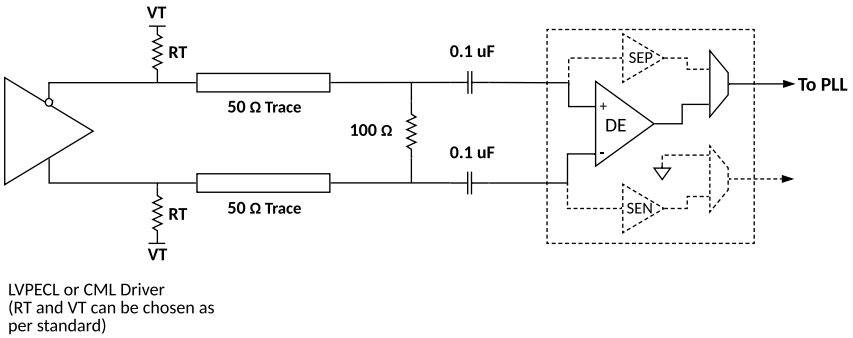



**Figure 23 Interface Type IV Differential AC Coupled 4C**

5A

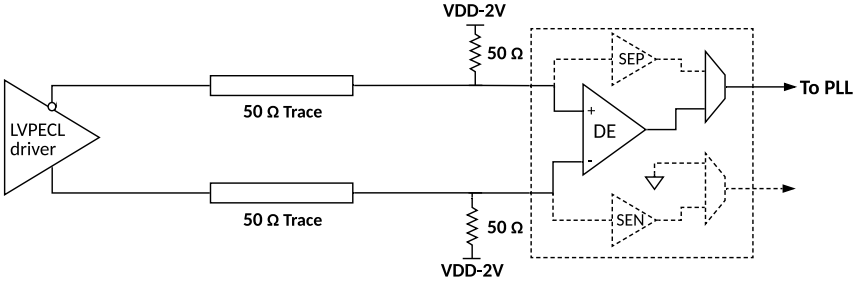



**Figure 24 Interface Type V Differential DC Coupled 5A**

5B

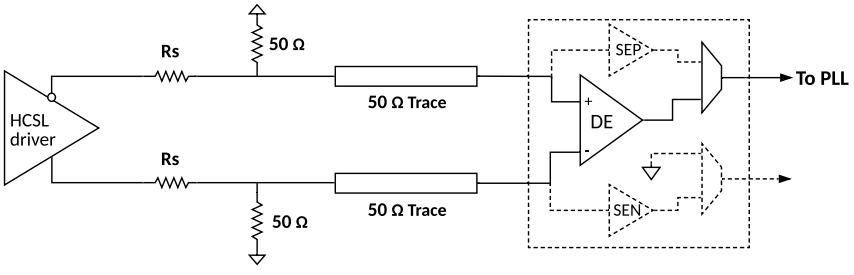



**Figure 25 Interface Type V Differential DC Coupled 5B**

# 7 Clock Monitor Slave Description

Various fault monitoring indicators are available on the chip. The Clock Loss and the Frequency Drift indicators are configurable with the Clock Monitor Slave. The specifications of these fault monitors are indicated in Table 8.

Defect monitoring on any of the clock monitors can be accessed using multiple techniques. The current status of the defect is available as an Active High defect that can be read from the PIF. The “status” is a current indicator of the defect that is high only during the defect (for example during the time that a Clock Loss event is on-going). Additionally, a sticky indicator of the defect called “Notify” can be enabled in the PIF. In this case, the concerned “notify” bit is high the first time the respective defect occurs and stays high till cleared.

There are multiple GPIOs (Flexible IOs) available in the system that can be programmed to monitor individual “notify” signals or a combination of them (as an OR logic). The choice of which fault defect is monitored as an output on the GPIO pin is flexible and can be programmed. Additionally there are selected GPIOs that are hard coded for the information for the clock defects.

## 7.1 Fault Monitoring System

The AU5508 parts provide an elaborate arrangement of fault monitoring indicators. There are 4 categories of clock monitoring that are necessary for the chip namely: Clock Loss Monitor (CL), Frequency Drift Monitor (FD), Lock Loss Monitor (LL) and XO Clock Loss Monitor (CL_XO).

Clock Loss (CL) monitors loss of input clocks defined as a pre-determined number of consecutive edges missing.

Frequency Drift (FD) monitors frequency drift of a particular clock against a pre-determined Golden Reference.

Lock Loss (LL) monitors the loss of lock in any PLL by monitoring the difference in frequency between the feedback and input clocks.

XO Clock Loss (CL_XO) monitors the loss of the XO reference that is generated from either an external oscillator (XO / TCXO / OCXO) or using the on chip XO amplifier that can work with a crystal blank on the PCB.

Each of these categories monitors the health of a particular clock for a certain failure type as illustrated in the name of the clock monitoring category.

For each clock failure observed by the clock monitor block there are two types of indicators provided to the user using the register map:

Live Failure Bit: There is a bit to indicate the live status of a particular failure. [Status]

Sticky Failure Bit: For each live failure bit there is a corresponding sticky bit that is set the first time that corresponding failure is encountered and stays set even if the failure has gone away. Only when the user clears the bit does it clear. [Notify]

The status of these can be either read from the register map or from the pins as a dynamic alarm monitoring arrangement. Additionally, sticky notify registers are available which have sticky status read back from the register map for the various defects. These can be selectively chosen to create an INTRB de-assertion on the INTRB pin as well.

An important point to note is that all of the fault monitoring indicators mentioned above that work with respect to the input clock work on the divided input clock post the DIVN1,k dividers. This implies that the fault monitoring indicators use the frequency fink that is input to the PLL post the DIVN1,k divider translation rather than the external frequencies fin_extk.

### 7.1.1 Clock Loss Monitors

Each of the inputs are monitored for Clock Loss in terms of missing edges to indicate a loss of input signal. The number of edges used to indicate a clock loss (or recovery from a clock loss) is programmable in the AU5508 GUI interface allowing for flexibility in choosing these thresholds. In addition there is a programmable “Wait Time” all of which are to be interpreted as follows:

Assertion of Clock Loss-

We declare a CL if “Trigger Edge” number of consecutive edges are missing. The “Trigger Edge” parameter is programmable in the chip GUI.

De-Assertion of Clock Loss-

We declare a ~CL if the clock is back and has less than “Clear Edge” consecutive edges missing. The “Clear Edge” parameter is programmable in the chip GUI.

Wait Time: After the clock is established to have returned, it is ensured that no CL error as defined by the de-assertion threshold occurs for “Val Time” seconds. This valid wait time is programmable using the chip GUI using the “Val Time” parameter which is programmable from the following options: {2m, 128m, 256m, 1, 4 ,32, 64, 128} sec. The use of the this valid wait time ensures that sporadic edges in the input clock (such as ones caused by noise on floating nodes or intermittent unstable clock edges) does not de-assert clock loss and it is established over a user determined period of time that the input clock is available and stable.

### 7.1.2 Frequency Drift Monitors

Any one of the input clocks or the XO clock can be used as the Golden Clock for calculating the frequency drifts of the other clocks. The Golden Clock can be chosen in the GUI and is used as the “0 ppm” Reference Clock for all monitoring.

Fine Frequency Drift has a step size of ±2 ppm.

Fine Frequency Drift has a range of ±2 to ±510 ppm and an independent threshold is programmable for “Set” (for setting the FD monitor) and for “Clear” (for clearing the FD monitor).

Fine Frequency Drift has an implicit hysteresis with resolution of ±2 ppm since the same range is available for the FD assertion and de-assertion. Use of hysteresis prevents unwanted oscillation of the FD monitor output at the decision threshold and is recommended for robust operation. The value of the FD threshold hysteresis is implicit in the choice of the set and clear thresholds.

Assertion of Fine Frequency Drift:

We declare as a Fine FD if Drift in input clock is greater than programmable “set ppm” in the chip GUI.

De-Assertion of Fine Frequency Drift:

We declare as a ~Fine FD if Drift in input clock over a measurement time is less than programmable “clr ppm” in the chip GUI. Additionally from the point of view of Fine Frequency Drift de-assertion, there is a delay from the point in time that lower than the specified ppm value is achieved to the point where the actual LL is de-asserted to the user such that LL never asserts during this delay period. The choice of this delay is with a timer that ensures that the delay is in line with the BW of the PLL loopWait timer is added During this time

Wait Time: After the clock is established to have returned, it is ensured that no Fine Frequency Drift error as defined by the de-assertion threshold occurs for “Val Time” seconds. This valid wait time is programmable using the chip GUI using the “Val Time” parameter which is programmable from the following options: {2m, 128m, 256m, 1, 4 ,32, 64, 128} sec. The use of the this valid wait timer ensures that input clock is stable and error is less the clr threshold over a user determined period of time.

The Fine Frequency Drift monitors provide precise information for input clock frequency drift. However, since the resolution of the measurement determines time for the measurement- an alternate faster measurement mechanism for drift is needed. This is Coarse Frequency Drift which has coarser measurement but is fast. It is available for cases where the drift is very fast in the input frequency and is programmable from options as shown below.

Coarse Frequency Drift has a step size of ±100 ppm.

Coarse Frequency Drift has a range of ±100 to ±1600 ppm and an independent threshold is programmable for “Set” (for setting the FD monitor) and for “Clear” (for clearing the FD monitor).

Coarse Frequency Drift has an implicit hysteresis with resolution of ±100 ppm since the same range is available for the FD assertion and de-assertion. Use of hysteresis prevents unwanted oscillation of the FD monitor output at the decision threshold and is recommended for robust operation. The value of the FD threshold hysteresis is implicit in the choice of the set and clear thresholds.

Assertion of Coarse Frequency Drift:

We declare as a Coarse FD if Drift in input clock is greater than programmable “set ppm” in the chip GUI.

De-Assertion of Coarse Frequency Drift:

We declare as a ~Coarse FD if Drift in input clock over a measurement time is less than programmable “clr ppm” in the chip GUI. Wait timer is added During this time

Wait Time: After the clock is established to have returned, it is ensured that no Coarse Frequency Drift error as defined by the de-assertion threshold occurs for “Val Time” seconds. This valid wait time is programmable using the chip GUI using the “Val Time” parameter which is programmable from the following options: {2m, 128m, 256m, 1, 4 ,32, 64, 128} sec. The use of the this valid wait timer ensures that input clock is stable and error is less the clr threshold over a user determined period of time.

Important Note regarding the above monitors with respect to clock switch in the PLL:

Normally the CL monitor is used for ascertaining a clock is lost for the PLL to switch to a secondary reference or proceed to Holdover. However, the Fine and/or Coarse FD monitors can also be used in addition to the CL monitor to cause a PLL switch. This implements an “OR” logic for the FD Monitors to be used in addition to the CL monitors for triggering a PLL input clock switch or entry to Holdover. This is programmable as an option in the GUI.

#### 7.1.2.1 Intsync clock as a golden clock

Frequency Drift monitor(FD) monitors the frequency drift of a particular clock against a predetermined Golden Reference(one of the input clocks or xo clock) for calculating the frequency drifts of the other clocks. The Golden Clock can be chosen in the GUI and is used as the “0 ppm” Reference Clock for all monitoring.

There is a special case can be achieved by choosing Intsync as golden clock(Generated clock from PLLB)

In the Normal mode of operation, PLLB is locked with precise input clock, PLLB output follows to input clock

In Holdover mode

In case of Input clock is lost, PLLB output follows the XO clock and therefore XO clock becomes golden clock automatically.

### 7.1.3 Lock Loss Monitors

Lock loss is programmable for each PLL with lock loss triggered if the frequency of the input reference to the PLL phase detection arrangement and the feedback clock to same PLL are different as per the programmed assertion and de-assertion thresholds.

The Set/Clear threshold for asserting the LL monitor is programmable from {±0.05 ppb, ±0.1 ppb, ±0.5 ppb, ±1 ppb, ±0.2 ppm, ±0.4 ppm, ±2 ppm, ±4 ppm, ±20 ppm, ±40 ppm, ±200 ppm, ±400 ppm, ±2000 ppm, ±4000 ppm}. A pre-determined level of hysteresis is implicit by choosing appropriately the set and clear thresholds for the LL monitor. Clear threshold should be programmed less than the Set threshold for proper functioning of LL monitor.

Additionally from the point of view of LL de-assertion, there is a delay from the point in time that lower than the specified ppm value is achieved to the point where the actual LL is de-asserted to the user such that LL never asserts during this delay period. The choice of this delay is with a timer that ensures that the delay is in line with the BW of the PLL loop. It is fully programmable from the GUI and is useful to ensure complete settling of the PLL without un-necessary toggling before LL de-assertion.

### 7.1.4 XO Clock Loss Monitors

The XO Clock Loss Monitor asserts the XO Clock Loss Alarm when the external reference input to the XA pin (XO or TCXO or OCXO) or the internal XO clock generated with the crystal blank is not available.

# 8 PLL Slave Description

The Digital Low Pass Filter (DLPF) receives the phase information from the Phase-to-Digital block, does gain correction and filtering and applies a correction to the Digitally Controlled Oscillator (DCO). The block also does holdover of past correction during clock switching and phase correction after clock switching. There are hooks available to perform external filtering, phase and frequency correction and incremental changes to the fraction applied to the fractional divider. The basic block diagram of the -DLPF is shown in Figure 26. The DCO here refers to the crystal referenced loop shown as the “Crystal (fref) Based Oscillator” in the diagram describing the PLL internals earlier in this document.

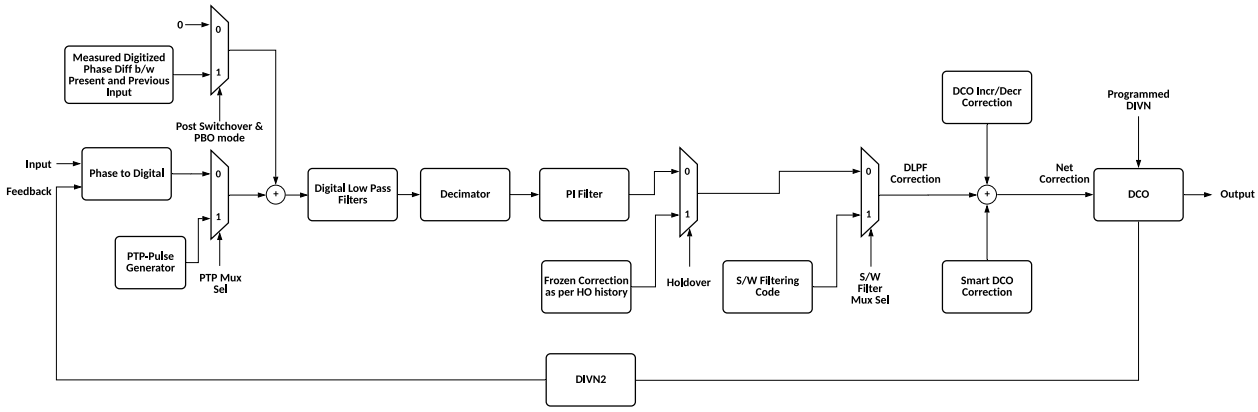



**Figure 26 Basic DLPF Block Diagram**

## 8.1 Mode of Operations

### 8.1.1 Basic Mode of Operation

In this mode, the digitized phase information is conditioned by the low-pass filters, decimated and passed through the PI-path filters. Gain correction of the digital word is also done. The conditioned signal is applied as a correction to the programmed division control word (DIVN). This main path is the highlighted in the Figure 27.

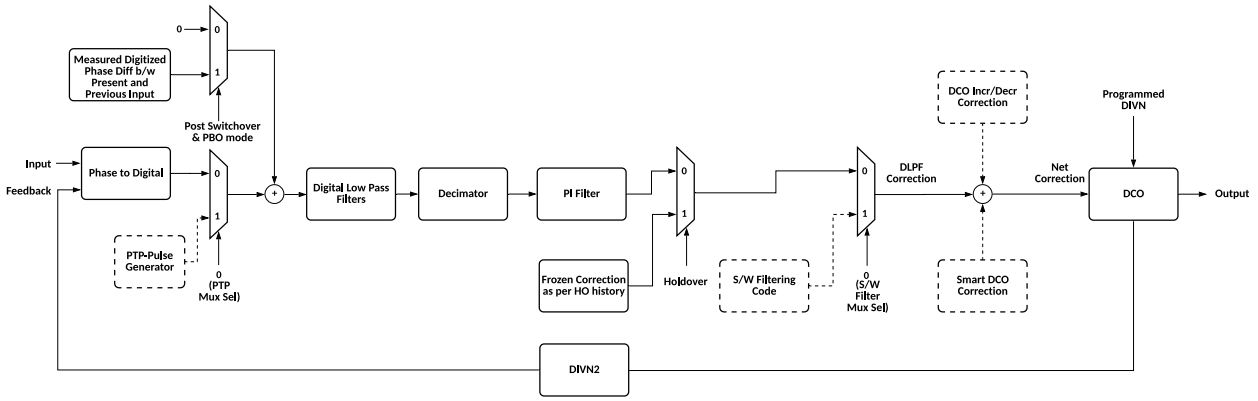



**Figure 27 Basic Mode of Operation**

### 8.1.2 Holdover and Phase Build-Out (PBO) modes

During normal mode of operation, a holdover control mode maintains a history of the recent correction applied by the filter. When a clock loss happens, DLPF enters the holdover state. In this mode, the last correction from holdover history is applied at the DLPF output, until another clock is available for switchover. When a new clock is available, DLPF exits the holdover state. If PBO mode is enabled, then DLPF calculates the phase difference between the previous and present inputs and applies this as an offset before exiting the holdover state.

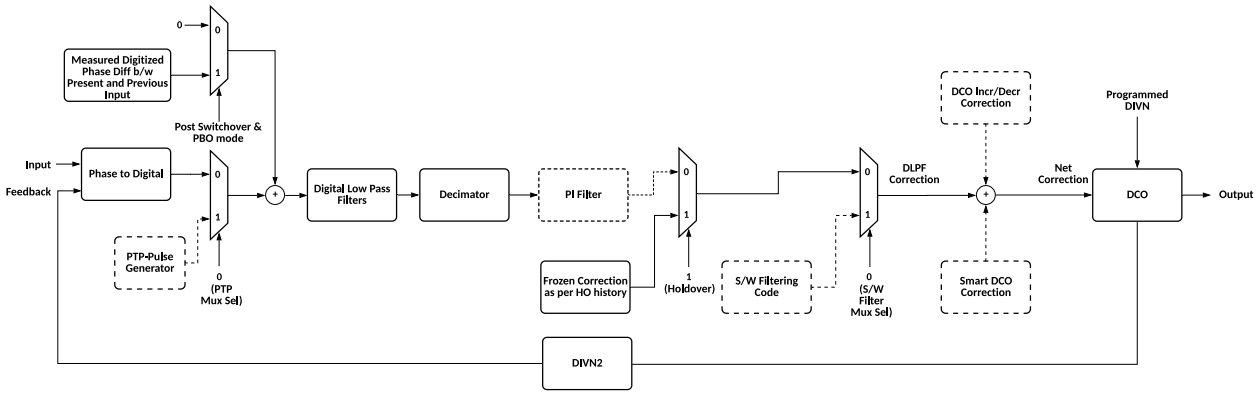



**Figure 28 Holdover and Phase Build-Out Modes**

### 8.1.3 Holdover and Phase Propagation (PPG) modes

Holdover and Phase Propagation mode is used for the case of two input clocks at the same frequency at the PLL input with any phase relationship between them such that the input phase difference is propagated to the output with a programmable slope or PLL bandwidth. This mode is selected from the GUI with the slope setting given options in GUI within the range of us/s to ns/s settings.

### 8.1.4 Frequency Ramp Mode

For the input clocks to the PLL that are not exactly the same frequency(plesiochronous clock) but only nominally the same frequency, the Frequency Ramp feature can be enabled from the GUI, In this case the output frequency will ramp such that is started tracking the new switched input reference with controlled slope ramp rate(ppm/sec) which can also be programmed in GUI. For the case where the input clocks are only nominally the same but plesiochronous in nature, the frequency ramp feature can be used to control the rate of change of the output frequency

### 8.1.5 Clock Select Modes

In sync mode the clock on which a PLL is locked to is said to be an active clock. This can be chosen in auto select (can be programmed via GUI) or manual select mode. In auto select, clock switching happens automatically depending on the clock's availability, At initialization PLL is programmed to use one as the Active clock and the others as Spare clocks. In manual select it happens on the request from user, controllable via Pin/Registers if GPIO is selected else PLL is always locked to Active clock and stay in holdover in case active clock lost . There are two mode in Manual active select case when GPIO is selected Direct Clock Selection: In this user controls the clock for the PLL's set in Manual Select Mode simultaneously via GPIO's or register bits. Direct clock information is given as an input. Indirect Clock Selection: In this user programs the two clocks for each PLL in registers and then switch between those two via GPIO or registers per PLL.

## 8.2 Software (S/W) Filtering Modes

S/W Filtering Modes can be used to bypass the internal PI-Filters of the DLPF and use external filters implemented in S/W. This mode can be used in open-loop as well as closed-loop operation.

### 8.2.1 Closed-Loop S/W Filtering Mode

To implement the filters externally in S/W, users will need the digitized phase samples at the input of the DLPF. These samples can be read via the serial interface of the chip. To do the processing at a lower rate, output of the decimator can be read. Once the samples are read, the external S/W filter must do conditioning as well as gain correction and the result of the processing can be applied at the output of the PI-Path. The serial interface of the chip is used to apply the samples as well. The block diagram highlighting the S/W Filtering path is shown below.

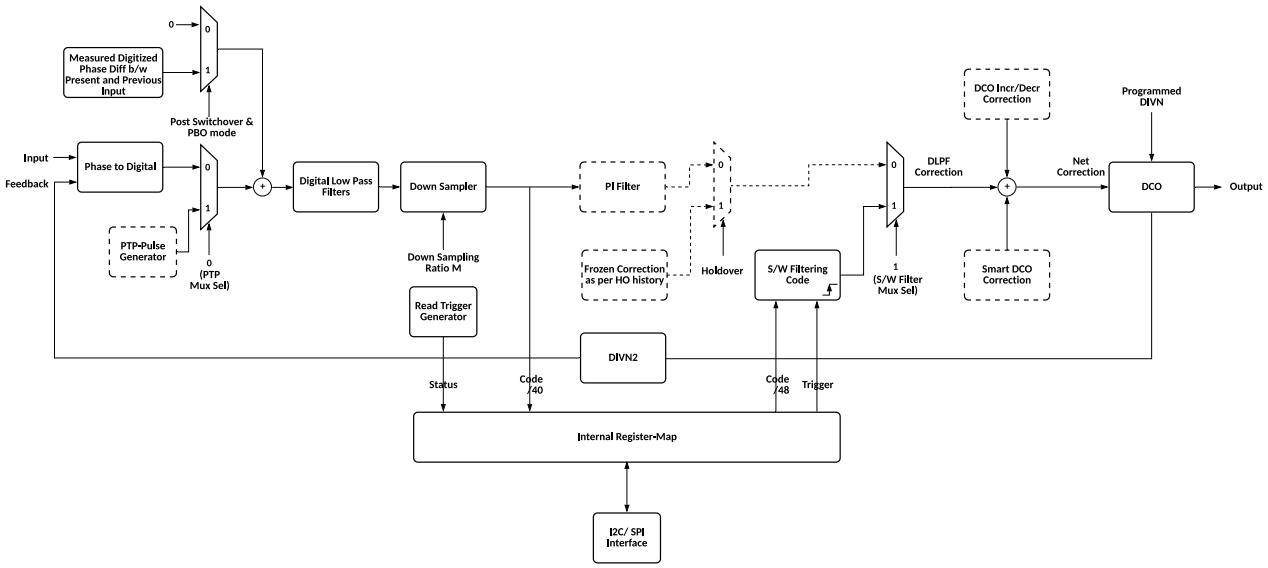



**Figure 29 Closed-Loop S/W Filtering Mode**

### 8.2.1.1 Down-sampling Rate, M

Users can choose to read the digitized phase data at the input rate or at a down-sampled rate, M. M can be programmed in powers of 2, from 1 to $2^{16}$.

### 8.2.1.2 Read Interface

Whenever a new sample is available at the output of the decimator, for processing, DLPF generates a read-trigger signal to notify users. The read trigger signal is a square wave of same frequency as the data rate at the output of the decimator. A 0→1 transition of this read-signal indicates that a new sample is available for processing. Users can utilize the read signal in three ways.

1. The read-trigger signal is mapped to a status bit in the register map. The status bit can be polled to detect a 0→1 transition and then, the new sample can be read.
2. The 0→1 transition of the read-trigger also sets the corresponding notify bit in the register map. If the associated interrupt is unmasked by users, then the notify will cause the external interrupt pin to be pulled low, indicating that a new sample is available for processing. In this mode, users will have to scan all notify bits to figure out the cause of the interrupt. Once the event that trigged the interrupt is known to be S/W filtering read-trigger generator, the new sample can be read and the interrupt can be cleared by writing a one to the notify bit.
3. The status bit can be brought out via one of the GPIOs of the chip and can be polled by users.

The samples have a resolution of 40 bits.

### 8.2.1.3 Write Interface

After a sample is read and processed, the output of the external S/W filter can be applied to the output of the DLPF using the write interface. The output samples have a resolution of 48 bits. Once the sample is written to a set of registers in the register map, via the serial interface, a write trigger bit has to be written to 1 to apply the sample. A 0→1 transition of the write-trigger bit will apply the sample to the output of the DLPF.

### 8.2.1.4 Closed-Loop S/W filtering sequence

The Figure 30 illustrates the closed-loop S/W filtering sequence.

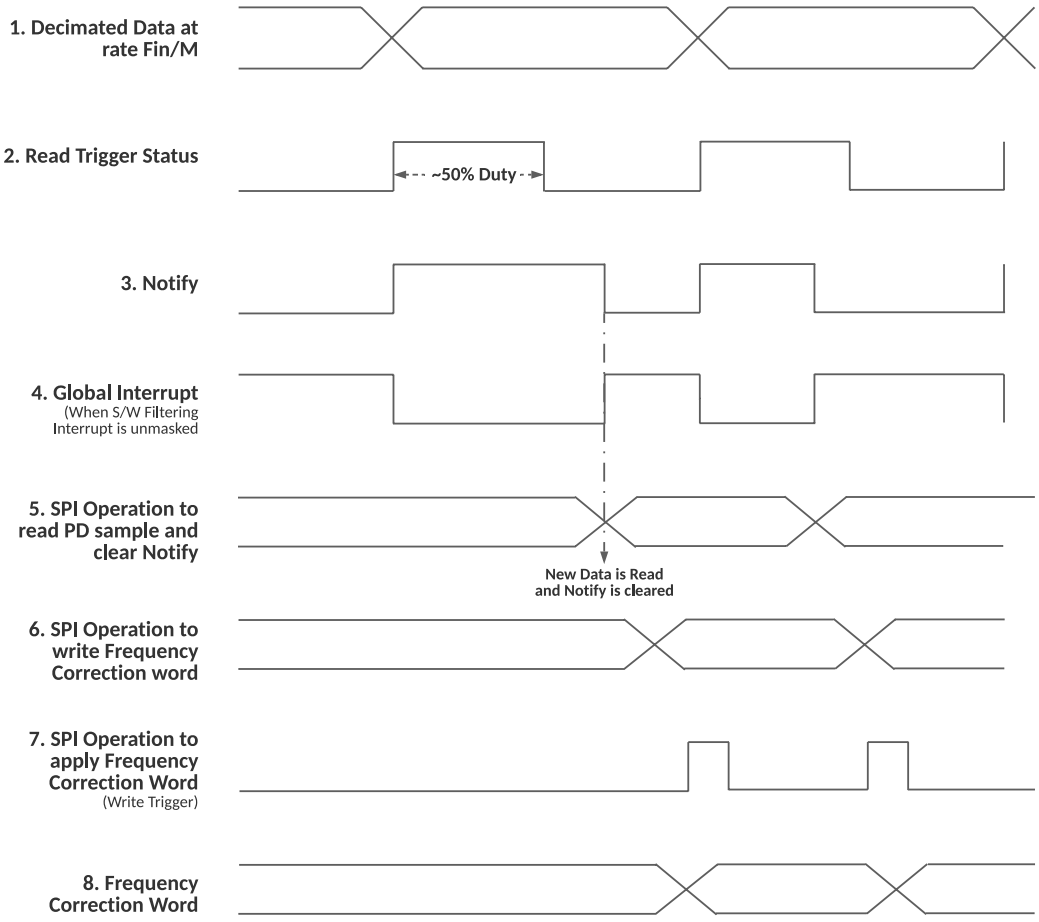



**Figure 30 Timing Diagram illustrating the closed-loop S/W filtering sequence**

- Timing waveform (1) is the data at the output of the down-sampler. The down-sampling ratio has been programmed to be M.
- Timing waveform (2) is the read-trigger signal, which has been mapped to a status bit in the register map. Waveform (3) is the associated notify bit.
- Waveform (4) in the global interrupt pin of the chip. It is assumed that the S/W filtering interrupt has been unmasked in this case.
- Waveform (5) depicts the SPI read operations to scan the notify bits to find out the cause of the interrupt, then to read the new 40-bit sample at the output of the decimator and also an SPI write operation to clear the notify bit. The global interrupt gets de-asserted, when the S/W filtering notify is cleared.
- Waveform (6) is the SPI write operations to write the 48-bit sample frequency correction word computed by the external S/W filter
- Waveform (7) indicates the SPI write operations to apply the word at the output of the DLPF (write-trigger)
- Waveform (8) indicates the state of the output of the DLPF

### 8.2.2 Open-Loop S/W Filtering Mode

This mode is similar to the closed-loop S/W filtering mode, except for that the digitized phase samples at the input of the DLPF are not read. Internal PI-Filter is bypassed and a frequency-correction word is applied via the serial interface. Once applied, the frequency-correction word remains in effect until a new word is applied.

## 8.3 PTP Modes

PTP modes are used to apply a phase-correction word at the input of the DLPF. DLPF can work either in a PTP only mode, without an input clock or the PTP pathway can be used in regular mode of operation to apply a phase-correction word. These modes are explained below

.

### 8.3.1 PTP Only Mode

In PTP only mode, the PTP pulse generator pathway is always enabled. Users can program the height of the pulse as a 40-bit pulse-correction word and the width of the pulse is specified in steps of PTP clock, as a 16-bit word. The PTP clock is generated by dividing XO and the division factor can be programmed by users to achieve any desired clock frequency. Once the parameters of the pulse are written, the pulse is applied by writing a trigger bit in the register map. A 0 -> 1 transition of this bit applies the PTP pulse. The DLPF block diagram highlighting the PTP pathway is shown in the figure below.

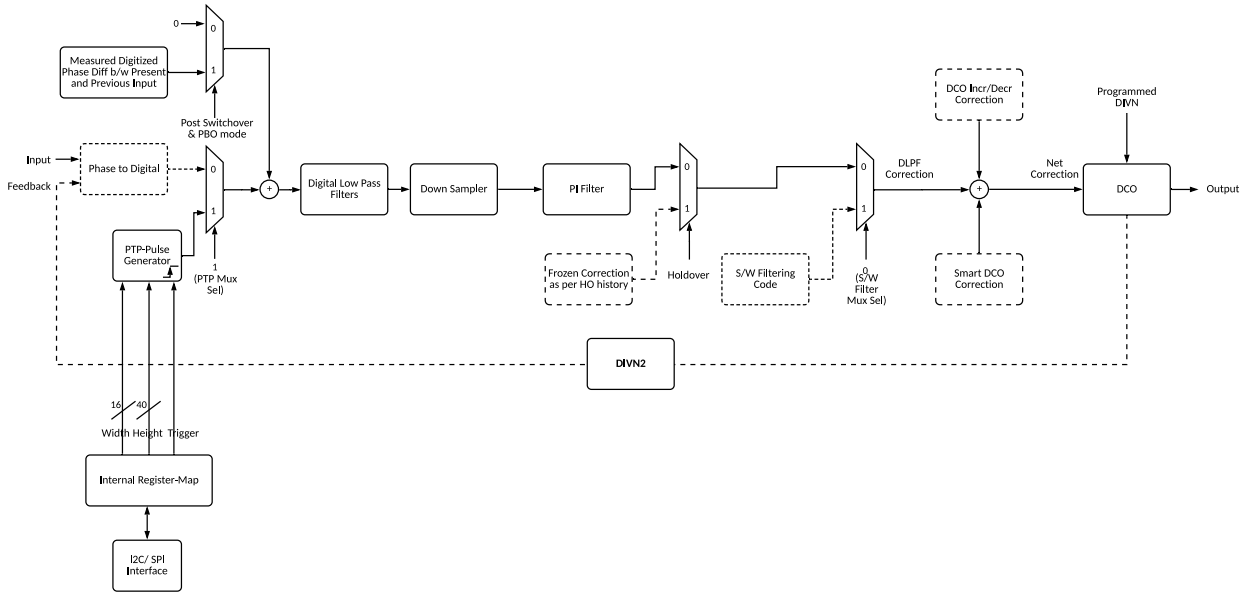



**Figure 31 PTP Mode**

### 8.3.2 Phase-Correction during normal mode

The PTP pathway can be used in normal mode as well, to apply a phase correction word. To select the PTP pathway, the PTP Mux Sel bit must be written to 1, the width and height of the pulse have to be specified just as in the PTP Only Mode and the trigger must be written to apply the pulse. In this case, the pulse width is specified in steps of the clock period at which the DLPF operates in the normal mode.

## 8.4 DCO Increment/Decrement Mode

This mode is useful to steer the clock frequency up or down, in small increments or decrements. The DCO increment/decrement path for the inner loop is highlighted in the figure below. A similar path also exists for the outer-loop and will be explained later.

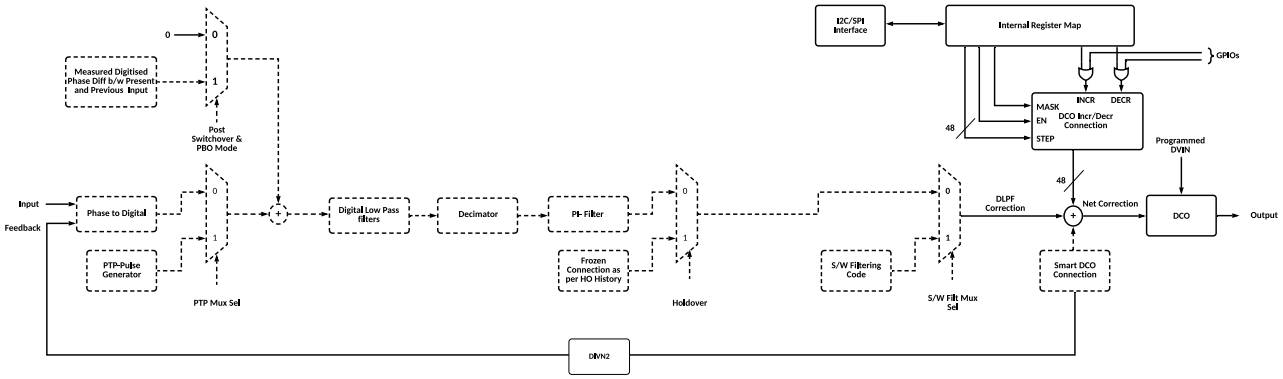



**Figure 32 DCO Increment/Decrement path for inner-loop**

### 8.4.1 Inner-Loop DCO increment/decrement path architecture

To provide more insight into the increment/decrement path of the inner-loop, a detailed architecture is presented in the Figure 33.

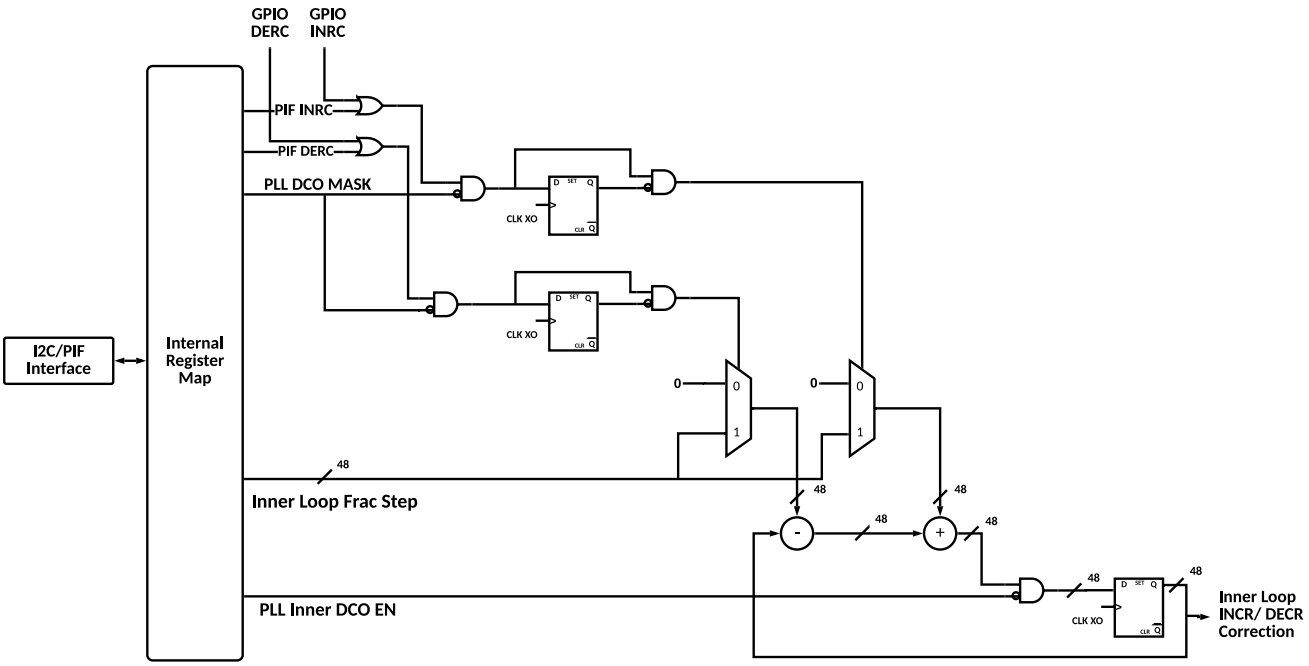



**Figure 33 Architecture of the inner-loop increment/decrement path**

To change the frequency in steps, a step size has to be programmed. The step size is specified as a 48-bit fraction in the range [-0.5, 0.5). Increments and decrements can be done by applying a pulse either via GPIOs or via the register map. Since the same set of IOs control all PLLs, the DCO mask bit of the PLL of interest whose frequency need to be changed must be written to 0. And the DCO enable bit must be written to 1. The DCO increment/decrement through register bits from the serial interface is unique to each PLL. With all these settings programmed, increment/decrement pulse can applied. A 0 -> 1 transition of the increment (decrement) causes the fraction step (negative of fraction step) to be accumulated and applied as a correction. The process can be repeated any number of times to obtain the desired net correction. Once this is done, the PLL can be masked and the same process can be repeated for another PLL, if required. The accumulator will continue to remember its state until the DCO enable is cleared.

### 8.4.2 Outer-Loop DCO increment/decrement path architecture

The outer-loop increment/decrement path has a similar architecture to that of the inner-loop path. While the PLL DCO mask bits are shared between the inner- and outer-loop paths, the paths have dedicated DCO enable bits. In addition to a 48-bit fraction step, outer-loop DCO also has a 24-bit integer step, in order to perform corrections of larger magnitude. Since the correction steps of the inner- and outer-loop DCOs are mapped to different sets of registers in the register map, both the inner- and outer-loop increment/decrement paths can be operated simultaneously.

## 8.5 Smart DCO Mode

Smart DCO is used for sharing clock accuracy information among multiple DPLLs. Thus one PLL can receive correction from and thus track other PLLs in the chip through dedicated signal paths on the chip. It is useful in creating a stable DCO for PTP applications using either OCXO or Sync-E available in the system. Using SDCO, the inherent low noise of XO and stability of OCXO or Sync-E can be provided in a seamless manner satisfying the ITU-T G.8273.2 standard as shown in Figure 34.

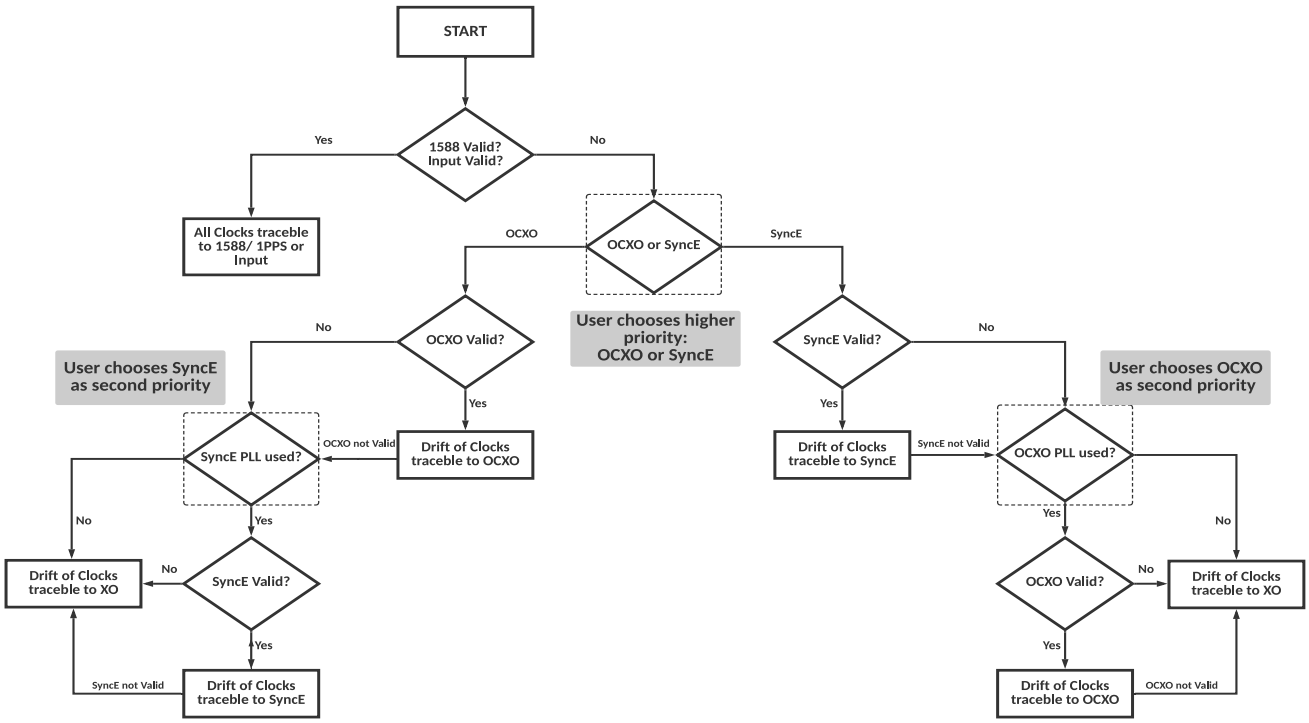



**Figure 34 Smart DCO Mode**

Note: User Programmable options are marked in Grey. The rest of the flowchart is autonomous algorithm available on the chip

Each PLL has SmartDCO and each SmartDCO has two transmitters Tx1,Tx2 and two receivers Rx1,Rx2 as shown in Figure 35. Rx1/Rx2 of a chosen PLL can receive from Tx1/Tx2 of any other PLL.

PLL which is acting as Rx can receive corrections from upto two other PLLs. PLL which is acting as Tx can send its correction to other PLLs. If a PLL is both an Rx as well as a Tx, then the received corrections can be optionally combined with the PLL's own correction before transmission. Thus the Smart DCOs in the PLLs can be daisy chained, i.e., the corrections received in a PLL can be combined with the correction of the PLL and then transmitted to other PLLs.

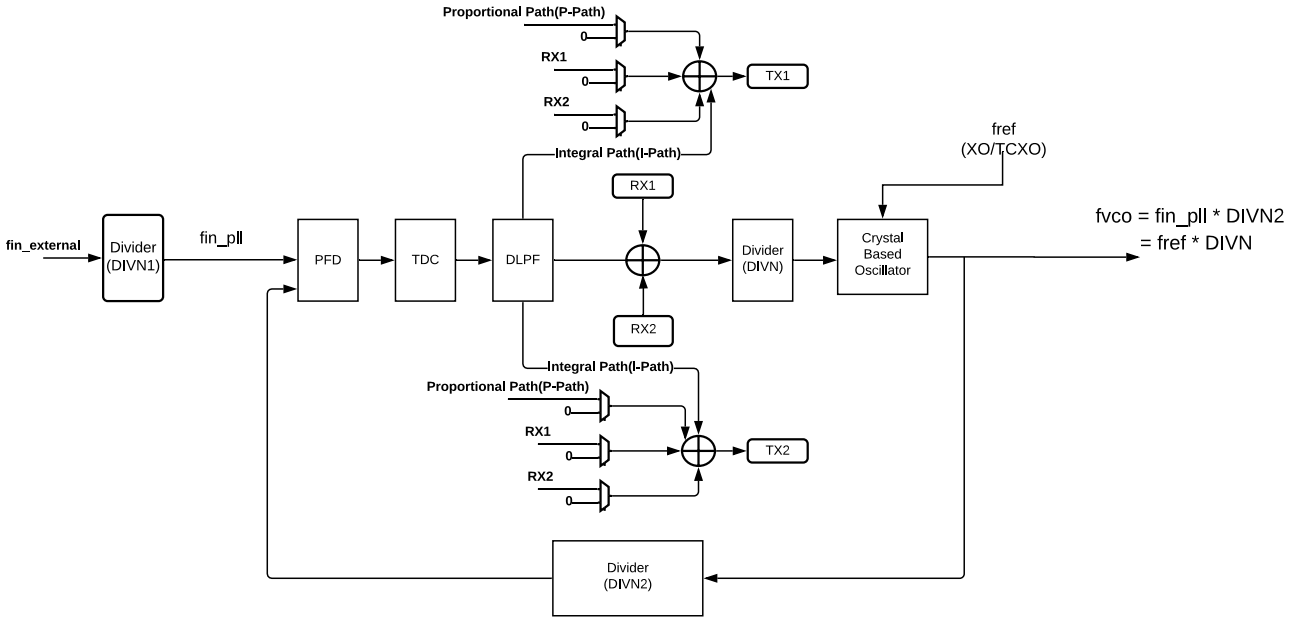



**Figure 35 Smart DCO Transmitters (Tx1, Tx2) and Receivers (Rx1, Rx2) in a PLL**

# 9 Output Slave Description

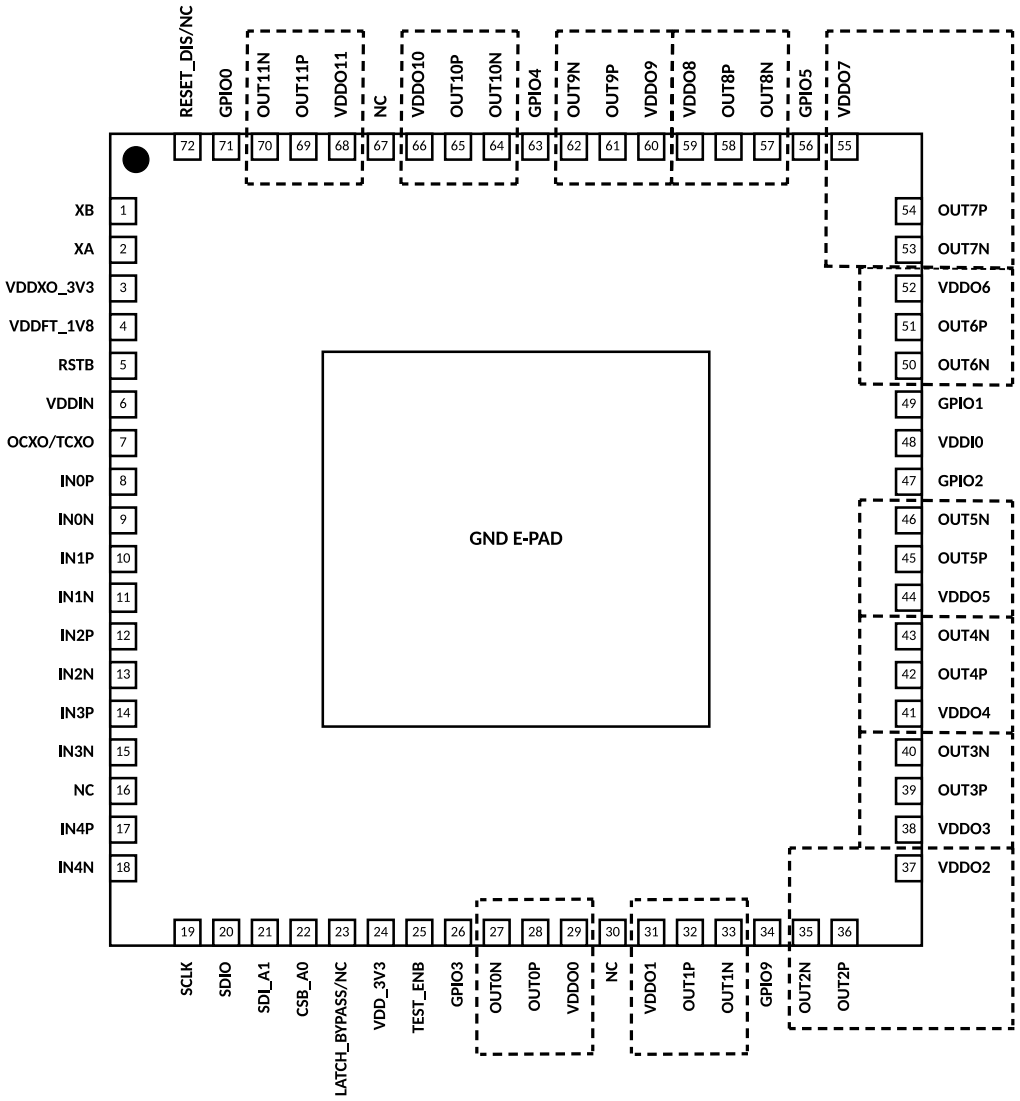



**Figure 36 AU5508 Output Driver**

- Several Terminations are available for the AU5508 part Output Driver which serve several industry standards.
- AU5508 parts offer programmable output swing for the various termination arrangements.
- Additionally internal termination modes are also available
  - If the receiver is correctly terminated these modes are not recommended
  - The internal termination modes are recommended for cases where transmission line integrity or far end termination is not robust.
- This section summarizes the various termination standards that are supported and the associated programmable swings. The outputs are in Hi-Z state after the chip reset and before the chip configuration. Figure 37 describes the interpretation used for swings.

Convention for Waveforms

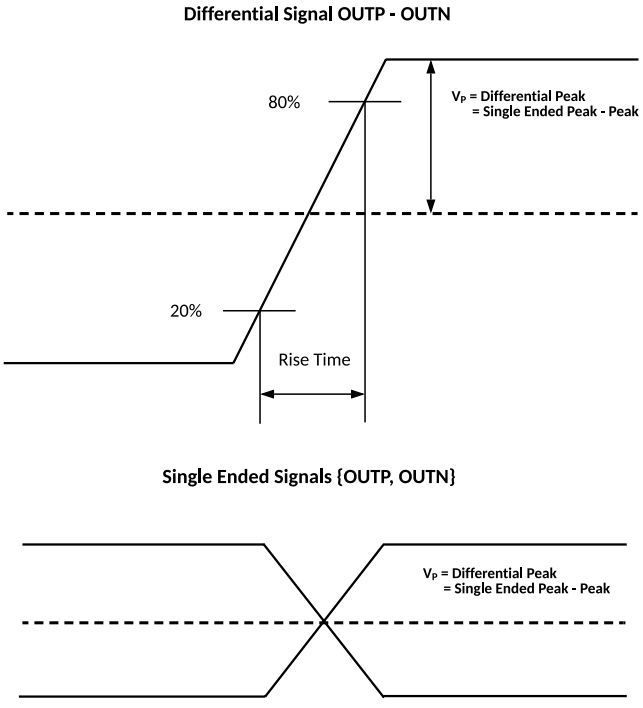



**Figure 37 Waveform Convention**

## 9.1 Regular High Swing modes

Differential Output Drivers: Regular High Swing Modes

VDDO = 1.8 V, 2.5 V, 3.3 V

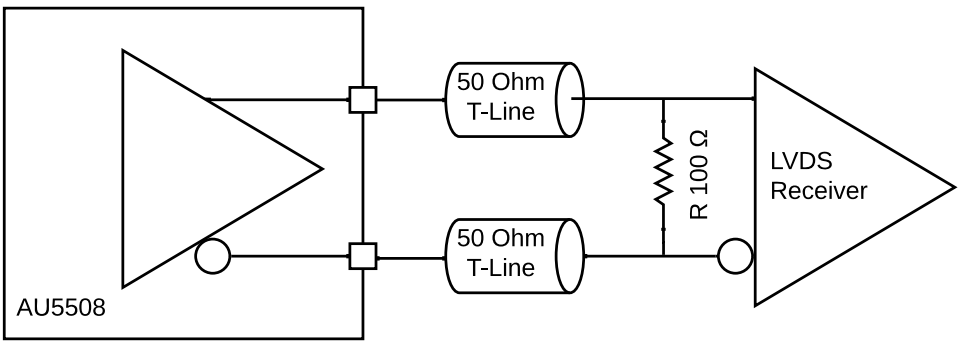



**Figure 38 Output Driver used with traditional DC Coupled LVDS receiver**

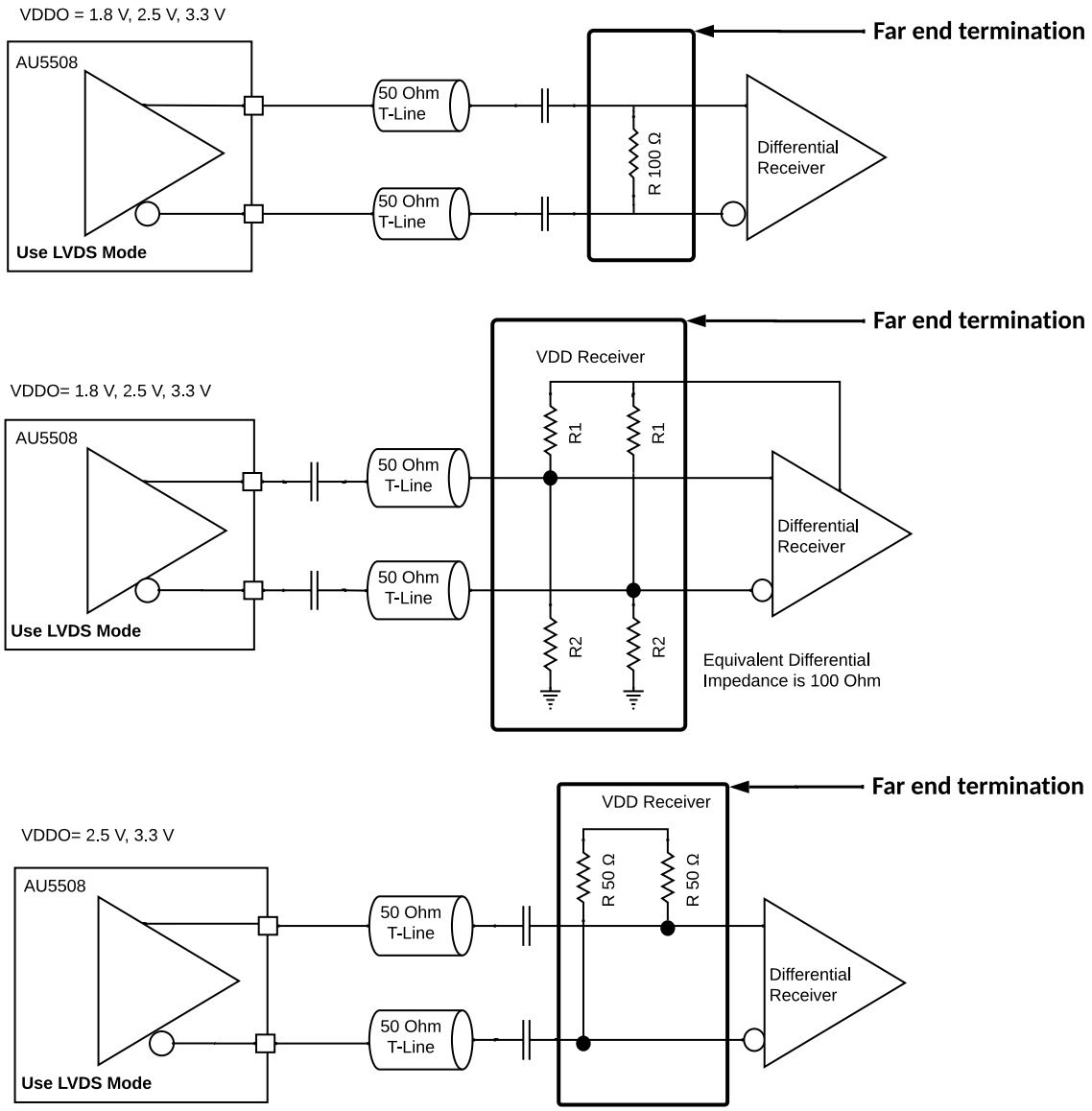



**Figure 39 AC Coupled Receiver side resistive Termination options**

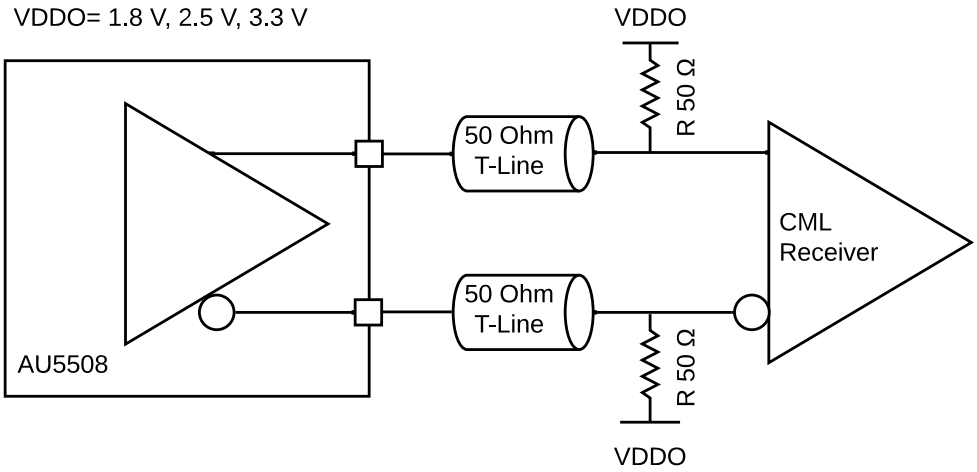



**Figure 40 Output Driver used with traditional DC Coupled CML receiver**

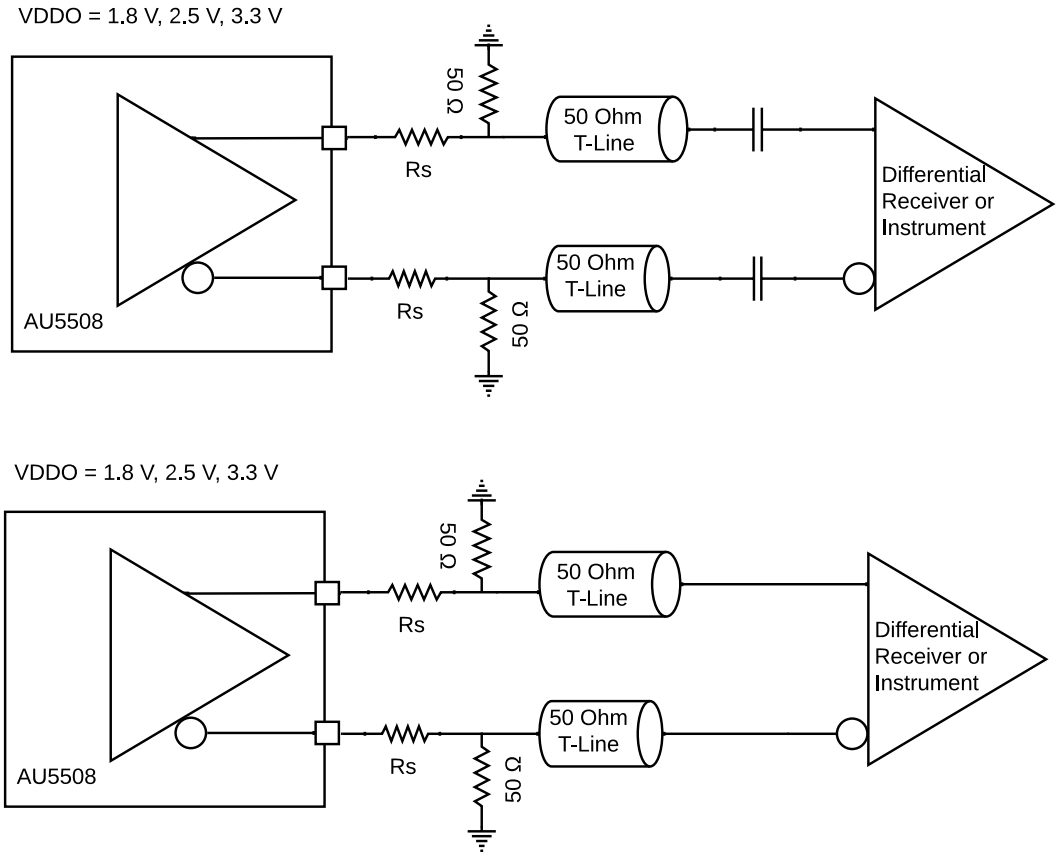



**Figure 41 Output Driver used with traditional HCSL receiver**

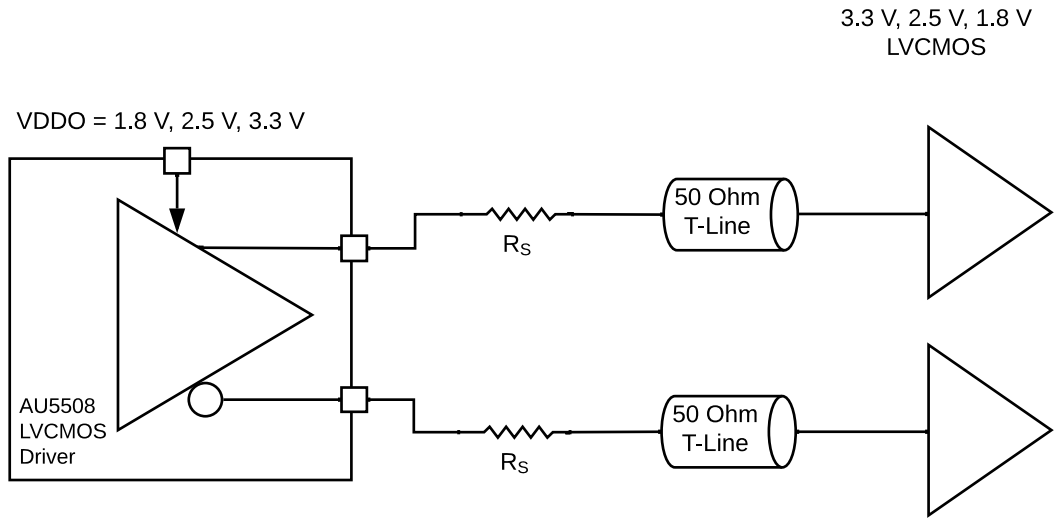



**Figure 42 DC Coupled LVCMOS**

| VDDO | RS |
|---|---|
| 3.3 V | 10 Ω |
| 2.5 V | 5 Ω |
| 1.8 V | 0 Ω |

## 9.2 Internal Termination modes

Differential Output Drivers: Internal Termination Modes

- If the receiver end of the transmission line (far end termination) is correctly terminated these modes are not recommended
- The internal termination is recommended for cases where transmission line integrity or far end termination is not robust.

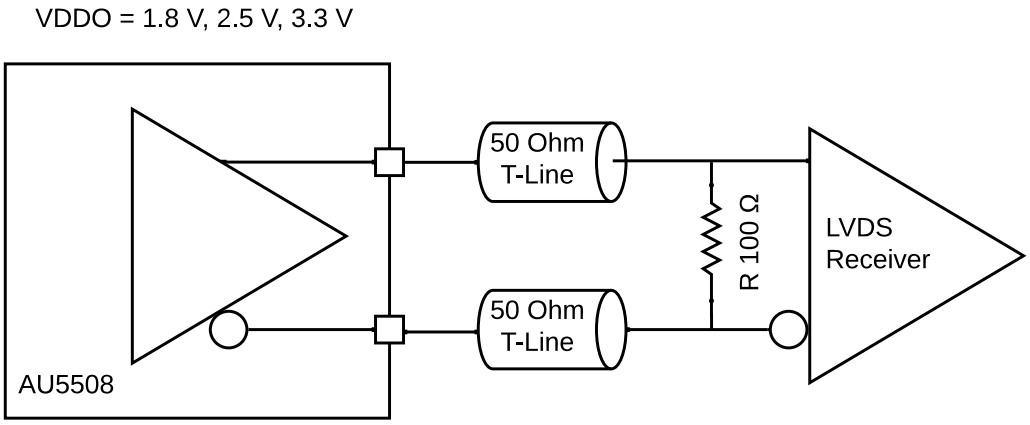



**Figure 43 Output Driver used with traditional DC Coupled LVDS receiver**

# 10 Output Clock Modes

AU5508 can provide 12 differential clocks or 12 pair of single ended clocks. These output clocks can be programmed differently to serve different applications at system level. This document explains output clock behavior in different functional modes of operations.

## 10.1 Functional Modes

1. SYSREF mode (JESD204B)
2. SYSREF mode with Pulser
3. SYNCB mode
4. Phase hopping fast lock mode

## 10.2 SYSREF Mode

For JESD204B application, AU5508 can provide up to 6 pairs of DEVICE clock and SYSREF clock. We can choose which output to be programmed as SYSREF clock independently. SYSREF clock can be triggered through 2 ways which are below.

1. Sysref trigger using PIN
2. Sysref trigger using REG WRITE (SPI OR I2C operation)

In sysref mode, there is an internal clock which is pre-aligned with known phase offset with respect to device clock. Using sysref_trigger signal, this clock is gated inside in a glitch free manner. Sysref clock (Figure 44) only appears after 2 negative edges after sysref_trigger is asserted high. Similarly, sysref_clock_out is disabled after 2 negative edges of internal clock when sysref_trigger is de-asserted to low.

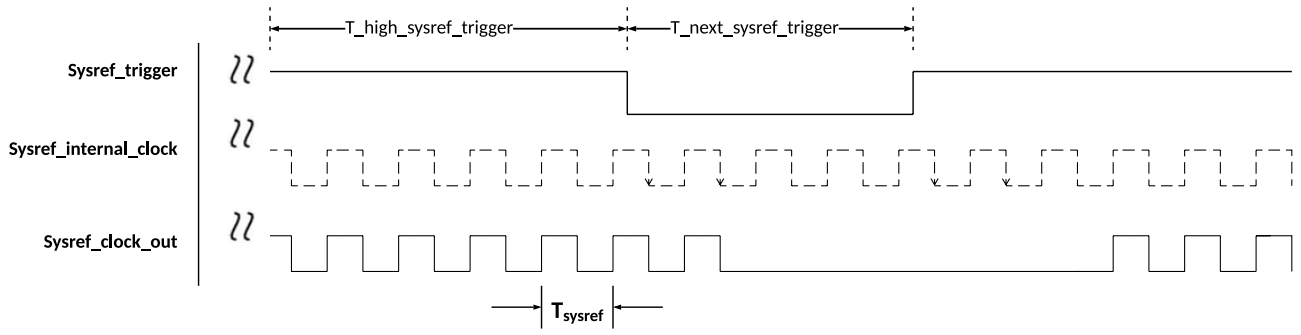



**Figure 44 sysref_clock_out behaviour with respect to sysref_trigger**

For SYSREF to work as shown above, following conditions must be true

- $T\_high\_sysref\_trigger \geq 2 * T_{sysref}$
- $T\_next\_sysref\_trigger \geq 2 * T_{sysref}$

## 10.3 SYSREF Mode with Pulser

In few JESD204B applications, sysref clocks are preferred as defined number of pulses rather than a continuous clock. This is done to reduce cross-talk between SYSREF and DEVICE clock. In AU5508, sysref clocks can be programmed in pulser mode of operation too. There are 8-bits independent to each sysref clock to program number of output pulses from 1 to 255. This mode can also be triggered through below 2 methods.

- Sysref trigger using PIN
- Sysref trigger using REG WRITE (SPI OR I2C operation)

In Figure 45, an example of sysref clock in pulser mode of operation is shown. In this example, sysref clocks are programmed for 3 pulses.

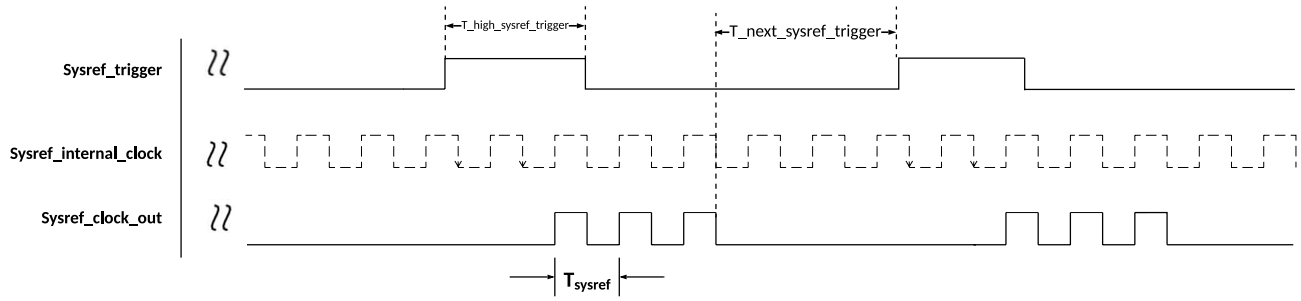



**Figure 45 sysref_clock_out behaviour with respect to sysref_trigger in pulser mode**

For SYSREF to work in ***pulser mode***, following conditions must be true

- $T\_high\_sysref\_trigger \geq 2 * T_{sysref}$
- $T\_next\_sysref\_trigger \geq T_{sysref}$

## 10.4 SYNCB Mode

In SYNCB mode of operation, any output clock can be independently programmed to react to syncb_trigger signal. In this mode, syncb selected output clock(s) will appear after certain delay which can be programmed with a resolution of 1 clock period of VCO. This mode can also be triggered through below 2 methods.

1. Syncb trigger using PIN
2. Syncb trigger using REG WRITE (SPI OR I2C operation)

In Figure 46 below, an example of syncb trigger is shown. syncb_clock_out appears after T_syncb_delay of time delay when syncb_trigger is asserted high and it is stopped in glitch-free manner after 2 negative edges when syncb_trigger is de-asserted low.

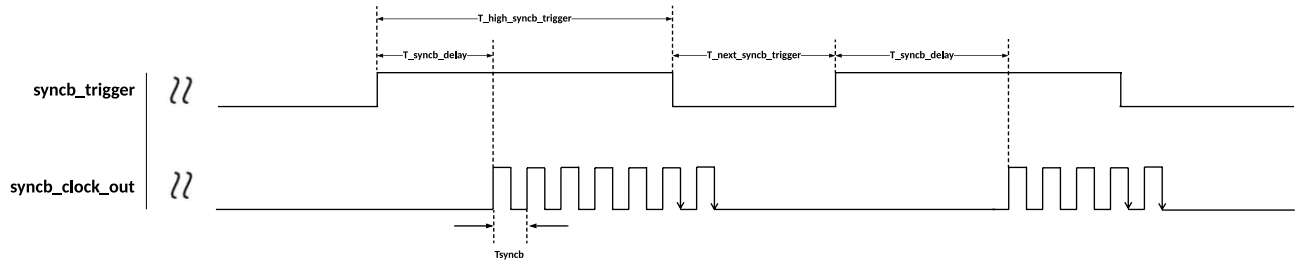



**Figure 46 syncb_clock_out behaviour with respect to syncb_trigger**

Definition:

$\boldsymbol{T_{syncb}}$ is one time period of syncb_clock_out

$\boldsymbol{T_{\_syncb\_delay}}$ is user programmed delay. First edge of syncb_clock_out will appear after this delay when ***syncb_trigger*** is asserted. It can be programmed with a resolution of 1 TVCO.

$\boldsymbol{T_{\_high\_syncb\_trigger}}$ is minimum duration of ***syncb_trigger*** staying high for reliable operation as per above definition.

$\boldsymbol{T_{\_next\_syncb\_trigger}}$ is minimum duration of ***syncb_trigger*** staying low for reliable operation as per above definition and it is directly represented as minimum time duration after which next ***syncb_trigger*** is asserted)

For SYNCB to work, following conditions must be true

- $T\_high\_syncb\_trigger \geq T\_syncb\_delay + (2 * T_{syncb})$
- $T\_next\_syncb\_trigger \geq 2 * T_{syncb}$

## 10.5 PHASE HOPPING FAST LOCK

In PHASE HOPPING FAST LOCK mode of operation, one of the output clock runs at 1PPS and it is phase locked with input 1PPS clock. At any point of time if input clock changes its phase more than programmed threshold value 〖±ϕ〗_th, output clock locks to new phase within 2 seconds (two periods of input clock). This feature is called “phase snapping”.

This feature can be triggered through any of the following method.

1. Input phase change auto detect ( ϕ_t≥〖|ϕ〗_th |)
2. Phase hopping fast lock trigger using PIN

3. Phase hopping fast lock trigger using REG WRITE (SPI OR I2C operation)

In order to relock to new input phase, output clock behavior depends directly on new phase of input clock with respect to its original phase. Figure 47 to Figure 52 show how output clock will look like with input clock phase change more than programmed threshold.

### 10.5.1 $-\pi < \phi_t \leq -0.5\pi$

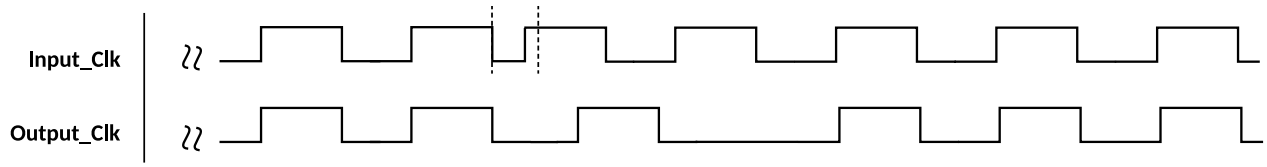



**Figure 47 Case for Phase difference between –π and -0.5*π**

### 10.5.2 $-2\pi < \phi_t \leq -\pi$

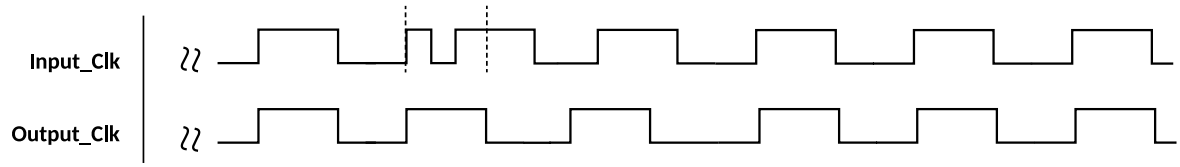



**Figure 48 Case for Phase difference between –2*π and -π**

### 10.5.3 $0.5\pi < \phi_t \leq \pi$

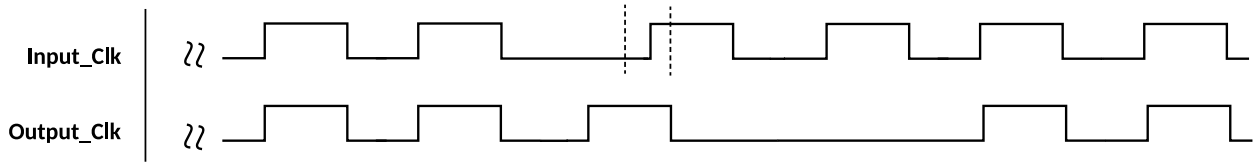



**Figure 49 Case for Phase difference between 0.5*π and π**

### 10.5.4 $\pi < \phi_t \leq 2\pi$

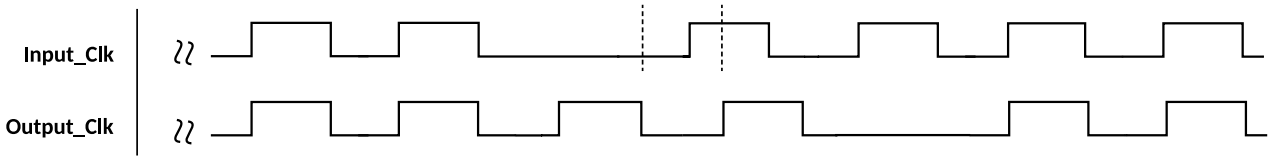



**Figure 50 Case for Phase difference between π and 2*π**

### 10.5.5 $0 < \phi_t \leq 0.5\pi$

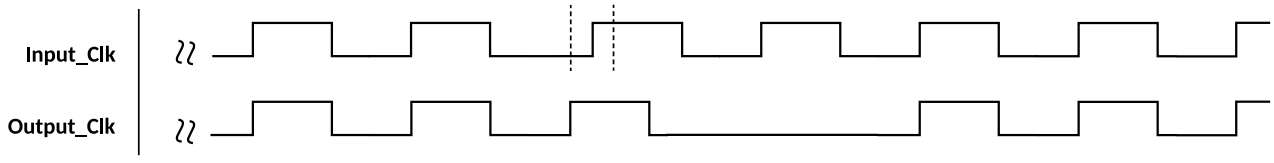


**Figure 51 Case for Phase difference between 0 and 0.5*π**

### 10.5.6 $-0.5\pi < \phi_t \leq 0$

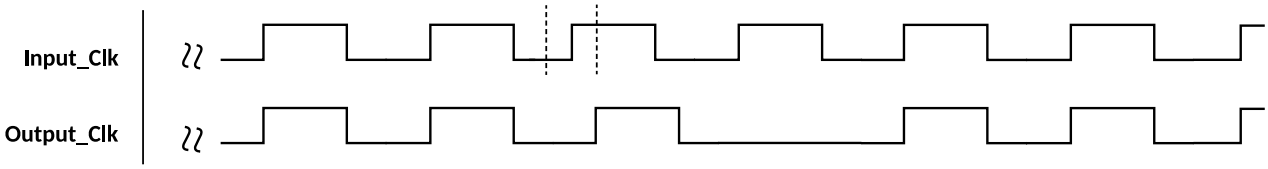



**Figure 52 Case for Phase difference between -0.5*π and 0**

# 11 Input TDC Slave Description

In telecom systems, consisting of Line Cards and Timing Cards, it is needed that the system contribute very little constant Time Error (cTE) to the network clock. AU5508 provides TDC functionality to measure the skew between clocks.

This document describes the use of TDC for precise clock phase measurement.

## 11.1 Input TDC Block Diagram

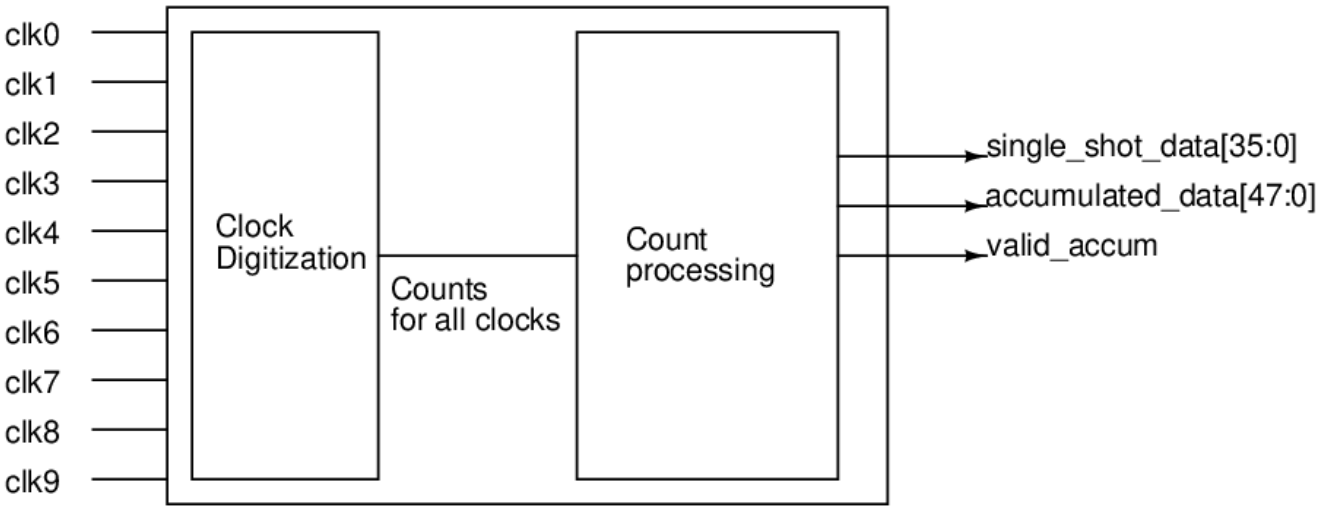



**Figure 53 TDC Block Diagram**

Figure 53 shows the input TDC system. The measurement clock is generated from an internal PLL. The time base for the measurement clock can be either OCXO or XO. Using an OCXO will provide better accuracy in the measurement compared to XO.

The TDC can measure phase difference between two inputs of same frequency or integer related frequencies as shown in Figure 54.

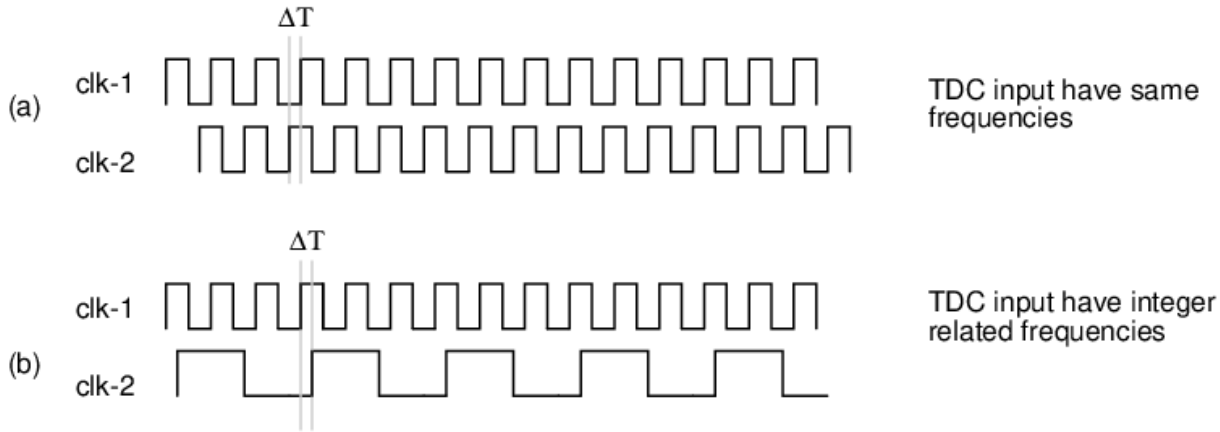



**Figure 54 TDC with same and integer related inputs**

For integer-N cases, the minimum phase difference between clocks is reported as described in Figure 55.

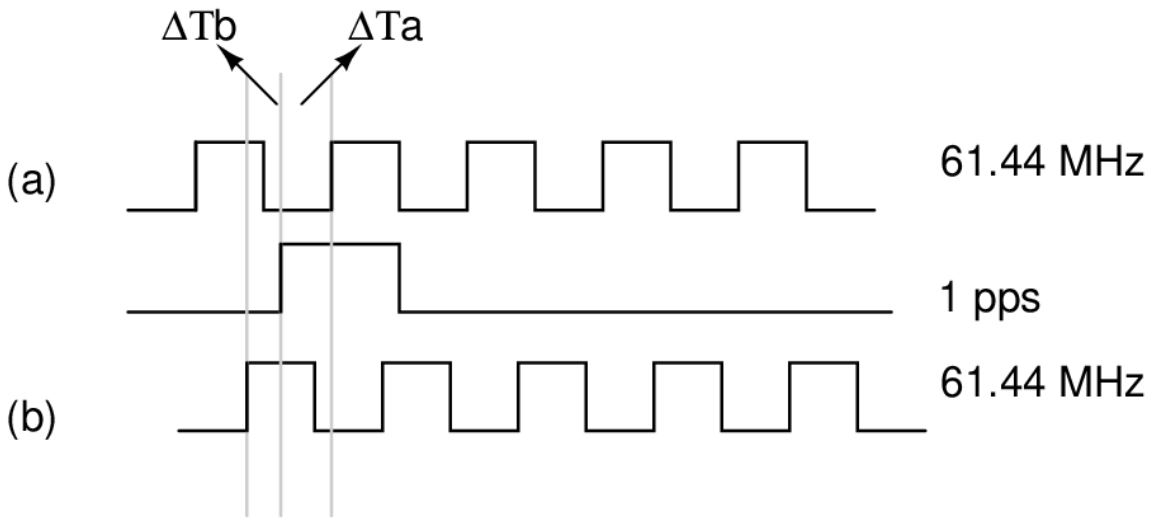



**Figure 55 Minimum phase difference is reported for integer related cases**

For Figure 55(a) ΔTa will be reported and for Figure 55(b) ΔTb will be reported. Moreover, ΔTa will be positive indicating 61.44 MHz edge in (a) has come later than 1 pps and ΔTb will be negative as the 1 pps edge is later compared to 61.44 MHz in (b).

**Table 20 TDC Specifications**

| Specification | Value | Comments |
|---|---|---|
| Input frequency range | 0.5 Hz – 250 MHz | |
| Edge polarity | Rising-Rising, Falling-Falling | |
| Averaging mode resolution | 10 ps | |
| Measurement time for averaging mode | $2^{Nsample+1}$ | Nsample can be programmed up to 11 |
| One time measurement resolution | 62.5 ps | |
| Phase difference range | $2\pi$ radians | |
| Single shot phase difference[35:0] | Digital read back | MSB bit represents the sign information. |
| Averaging mode phase difference[47:0] | Digital read back | MSB bit represents the sign information. Resolution depends on the samples used for averaging. |

If the TDC is configured to process C1 – C2 as in Figure 56, where C1 corresponds to clk-1 and C2 corresponds to clk-2. A positive phase difference means, clk-1 is lagging clk-2, a negative sign means clk-1 is leading clk-2.

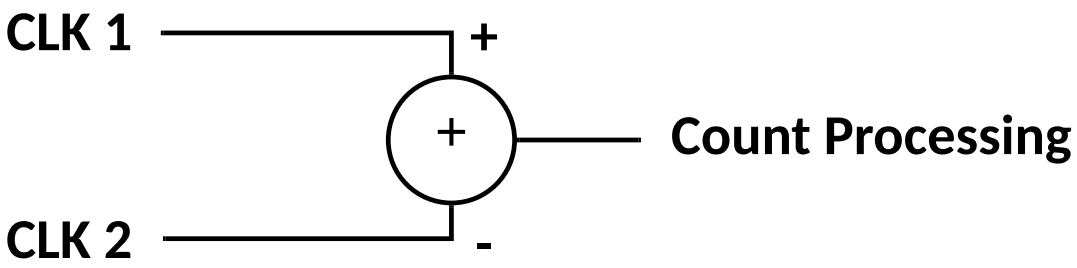



**Figure 56 Count Processing**

## 11.2 Analog and Digital Channel Selection

There are 10 analog channels and 10 digital pairs. User can choose to power up or power down any channel based on the requirement. Programmability is explained in the Table 21

**Table 21 Analog and Digital Channel Selection**

| Reg Field name | Field type | Default Value | Description |
|---|---|---|---|
| ANA_INPUT_CHNL_PDN_X | R/W | 0 | Out of 10 Analog input channels, each channel can be selectively powered up or powered down. By Default all channels are powered up. X-> 0 to 9 channels<br>0 : power up<br>1 : power down |
| DIG_PAIR_PUP_X | R/W | 0 | Out of 10 digital output channels, each channel can be selectively powered up or powered down. By default all channels are powered down. X -> 0 to 9 channels<br>0: Power down<br>1: Power up |

## 11.3 Input Clock Pair Selection

There are 10 digital pairs. Each Pair can select any 2 clocks based on the settings described in the table below. There are 10 such Clock pairs. Figure 57 shows the abstract level view of the clock pair selection

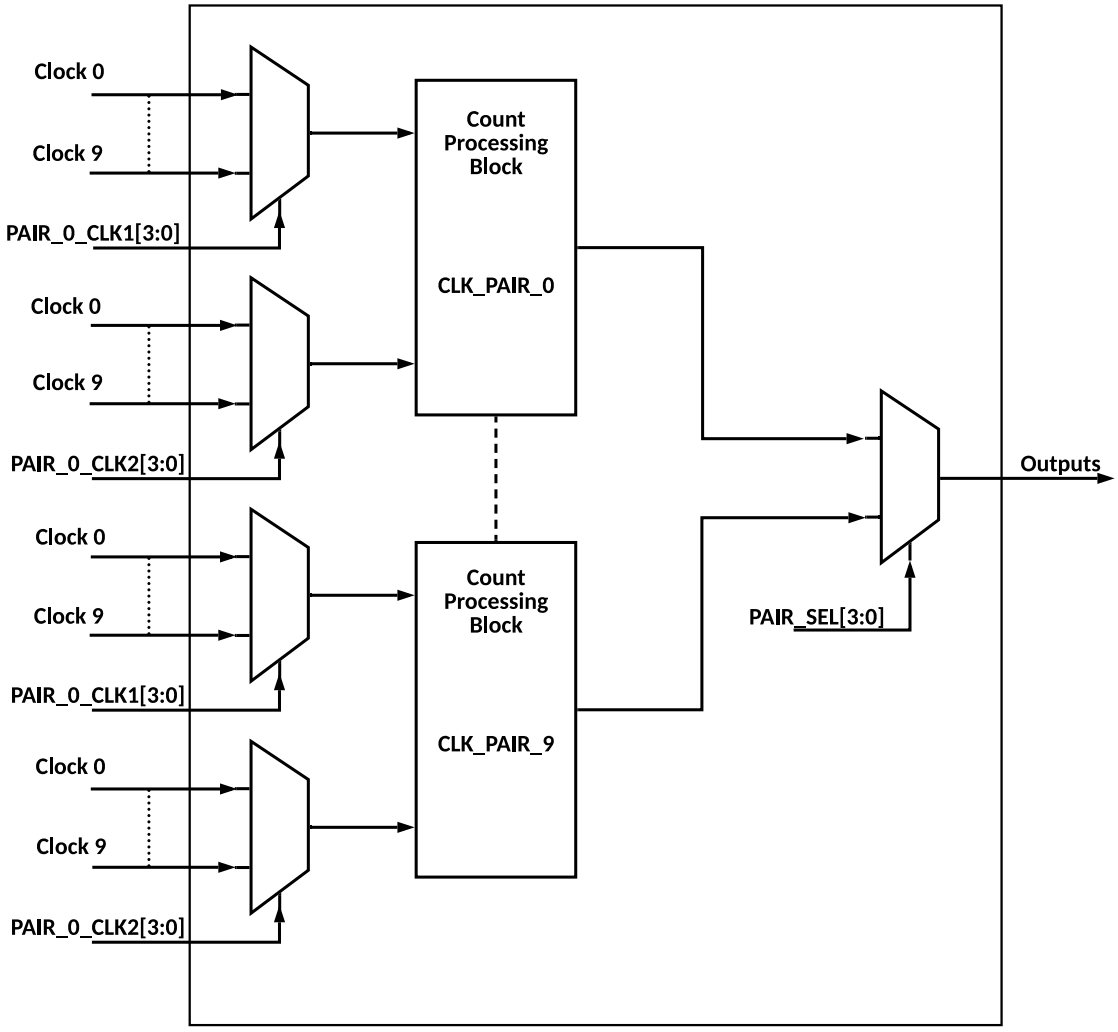



**Figure 57 Clock Pair Selection**

**Table 22 Clock Pair Selection for Comparison**

| Reg Field Name | Field Type | Default Value | Description |
|---|---|---|---|
| PAIR_X_CLK1[3:0] | R/W | 0 | Each Digital channel can take two clocks. First clock selection is done using this bit<br>X -> 0 to 9 Digital pair<br>0: Clock0<br>1: Clock1<br>2: Clock2<br>3: Clock3<br>4: Clock4<br>5: Clock5<br>6: Clock6<br>7: Clock7<br>8: Clock8<br>9: Clock9 |

| Reg Field Name | Field Type | Default Value | Description |
|---|---|---|---|
| PAIR_X_CLK2[3:0] | R/W | 0 | Each Digital channel can take two clocks. Second clock selection is done using this bit<br>X-> 0 to 9 Digital Pair<br>0: Clock0<br>1: Clock1<br>2: Clock2<br>3: Clock3<br>4: Clock4<br>5: Clock5<br>6: Clock6<br>7: Clock7<br>8: Clock8<br>9: Clock9 |

## 11.4 Output Clock Pair Selection

Out of 10 clock pairs, user can choose only one pair of interest to read out. The selection can be done using following settings. This is the final mux that decides which data should be read out as shown in Figure 57.

**Table 23 Output Clock Pair Selection**

| Reg Field Name | Field type | Default Value | Description |
|---|---|---|---|
| PAIR_SEL[3:0] | R/W | 0 | Out of 10 available pairs, User can read data for selected digital pair.<br>0:Clock pair 0<br>1:Clock pair 1<br>2:Clock pair 2<br>3:Clock pair 3<br>4:Clock pair 4<br>5:Clock pair 5<br>6:Clock pair 6<br>7:Clock pair 7<br>8:Clock pair 8<br>9:Clock pair 9 |

## 11.5 Read Interface

User can get data by reading following registers. When user want to read data READ_UPDATE should be issued so that the 48 bit accumulated data and 36bit single shot data will be captured internally at a single instance after which user can read the stable data.

**Table 24 Read Interface**

| Reg Field Name | Filed type | Default Value | Description |
|---|---|---|---|
| SINGLE_SHOT_DATA[35:0] | R | 0 | This Field provides phase difference between selected clocks at every slower clk in frequency rate. Accuracy is 62.5ps |
| ACCUMULATED_DATA[47:0] | R | 0 | This field provides phase difference between selected clocks after accumulating 2**(NSAMPLE+1) samples at slowest fin rate |
| VALID_ACCUM | R | 0 | 1:indicates first 2**(NSAMPLE+1) data is accumulated and data is valid<br>0: not valid |
| READ_UPDATE | R/W | 0 | This bit should be toggled from 0 to 1 when user wants to read SINGLE_SHOT_DATA[35:0] and ACCUMULATED_DATA[47:0] . This bit will be auto cleared internally. This bit should be polled until it becomes 0 then only SINGLE_SHOT_DATA, ACCUMULATED_DATA should be read |

## 11.6 Programmability

Along with clock selection, there are few programmability that are described in Table 25.

**Table 25 Input TDC Programming Options**

| Reg Field Name | Field Type | Default Value | Description |
|---|---|---|---|
| NSMAPLE_X[3:0] | R/W | 0 | 2**(NSAMPLE+1) samples will be accumulated at slower fin rate<br>X-> $0^{th}$ pair to $9^{th}$ pair |
| INPUT_CLK_POL_X | R/W | 0 | This bit is used to change the polarity of input clock for the selected analog channel<br>X -> 0 to 9 clocks<br>0: No inversion in polarity<br>1: Change the polarity of the corresponding analog channel |
| UPDATE_SETTINGS | R/W | 0 | This bit should be toggled from 0 to 1 when any settings are changed by user. This bit is cleared internally. |

## 11.7 TDC output post-processing

The chip provides the final count for the programmed Nsample value. To obtain the equivalent phase difference in seconds, it needs to post processed using below equation

Phase difference = Accumulated_data / (2^(Nsample+1) * Ndiv* fxo * 8)

Nsample, fxo are programmed by the user. The chip will provide the readback for Ndiv.

# 12 Serial Programming Interface Description

The AU5508 device has a serial programming interface. The serial programming interface supports I2C(Master) protocol to configure the device from the external EEPROM, and I2C(slave) and SPI(slave) serial interface protocols, for reconfiguring the device settings using register read/write. Status of the GPIO1, GPIO2, GPIO4, GPIO5 and GPIO9 is latched during power up when reset is released. Default settings during power up and configuration registers for GPIOs are as shown in Table 26. Latched value of GPIOs will be decoded later when NVM contents are copied to Chip Settings. Using GPIO latch configuration from Chip Settings latched value of GPIO1, GPIO2, GPIO4 and GPIO5 will be decoded as shown in Table 27 to configure the Serial Interface. GPIO9 latch has static configuration for I2C / SPI slave selection. A 1/HIGH on the GPIO9 latch sets the device in I2C mode and a 0/LOW in SPI mode.

**Table 26: GPIO Latching Configuration**

| GPIO | Default Direction | Configuration Register (gpio_latch_cfg[2:0]) | Default Pull Up/Pull Down | Function | Description |
|---|---|---|---|---|---|
| GPIO0 | Input | Page0 reg36[2:0] | Weak Pull Down | User Defined | User can define the functionality of these GPIOs using GPIO latch configuration as shown in Table 27. |
| GPIO3 | Input | Page0 reg37[6:4] | Weak Pull Down | User Defined | |
| GPIO9 | Input | | Weak Pull Up | I2C / SPI Mode Select | 0 : SPI Mode<br>1 : I2C Mode |
| GPIO1 | Input | Page0 reg36[6:4] | Weak Pull Up | User Defined | User can define the functionality of these GPIOs using GPIO latch configuration as shown in Table 27. |
| GPIO2 | Input | Page0 reg37[2:0] | Weak Pull Up | User Defined | |
| GPIO4 | Input | Page0 reg38[2:0] | Weak Pull Up | User Defined | |
| GPIO5 | Input | Page0 reg38[6:4] | Weak Pull Up | User Defined | |

**Table 27: GPIO latch Decoding**

| gpio_latch_cfg[2:0] value | Decoded Function | Description |
|---|---|---|
| 3'd0 | Unused default | If any GPIO is configured with this settings then latched value will be ignored |
| 3'd1 | EEPROM access disable[1] | Disable EEPROM access during Wakeup sequence<br>0 : If I2C mode is selected using GPIO9, the device will try to read the configuration block from the external EEPROM.<br>1 : Device will not attempt to read the configuration block from external EEPROM.<br>By default, no GPIO is used for this purpose, so the device will try to read the configuration block from external EEPROM. |
| 3'd2 | I2C slave address A2[1] | If I2C mode is selected using GPIO9, the latched value from appropriate GPIO will be used as I2C slave address bit A2. In SPI mode latched value will be ignored.<br>By default, no GPIO is used for this purpose, so default I2C slave address will have a 0 for bit A2. |
| 3'd3 | SPI mode 3wire 4wire[1] | If SPI mode is selected using GPIO9, the latched value from appropriate GPIO will be used as control for 3-wire/4-wire mode of SPI. In I2C mode, latched value will be ignored.<br>0 : 4wire mode<br>1 : 3wire mode<br>By default, no GPIO is used for this purpose, so device will be in 4-wire SPI mode. |
| 3'd4 | Spare | Reserved |
| 3'd5 | Spare | Reserved |
| 3'd6 | Spare | If any GPIO is configured with this settings then latched value will be ignored |
| 3'd7 | Spare | If any GPIO is configured with this settings then latched value will be ignored |

By default GPIO latching function is not enabled on AU5508. GPIO latching and its corresponding latching functions are factory programmed by Aurasemi with non 00 code marking. Please contact Aurasemi for more details.

## 12.1 Serial Interface Pins

Following pins of the device are used as a Serial Interface. These pins are configured for different functionality in different modes.

- SCLK

- SDIO
- SDI
- CSB

## 12.2 Serial Programming Interface Description

### 12.2.1 I2C protocol

#### 12.2.1.1 I2C Master

AU5508 can be configured from external I2C EEPROM. In wakeup sequence master checks for the EEPROM access, if serial interface is configured to I2C mode, AU5508 tries to read the configuration block from EEPROM during wake up.

To read the configuration data from the external I2C EEPROM, this device becomes the I2C Master and initiates the I2C transaction as shown in Figure 58. Serial Interface Pin configuration in I2C master mode is as follows.

- SCLK (SCL)
- SDIO (SDA)
- SDI ($A_1$)
- CSB ($A_0$)
- GPIOx ($A_2$)  Note: x = 1, 2, 4, 5.

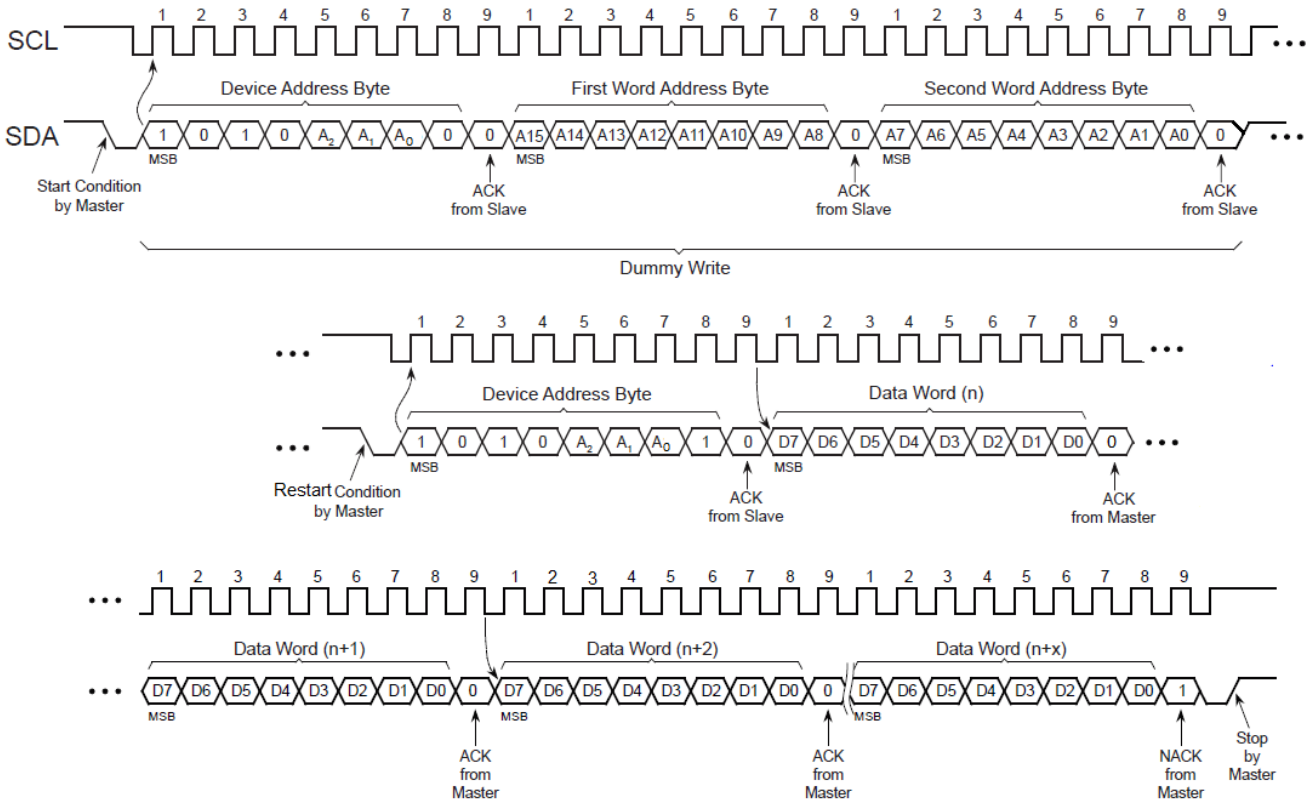



**Figure 58 EEPROM read**

In I2C master mode device supports following I2C features.

- I2C master is compliant with 100 KHz and 400 KHz I2C interface.
- Supports 7 bit I2C bus address. .
- Clock Synchronization
- Arbitration
- Start Byte

## Clock Synchronization

Device supports multi master communication. In the multi master mode the SCL line synchronization is required. In clock synchronization process the SCL's high time is decided by the clock with the shortest high period and the low time is decided by the clock with the longest low time. The SCL line is a WAND, so if any one of the clock is counting its low time, the other clocks are also forced to stay low.

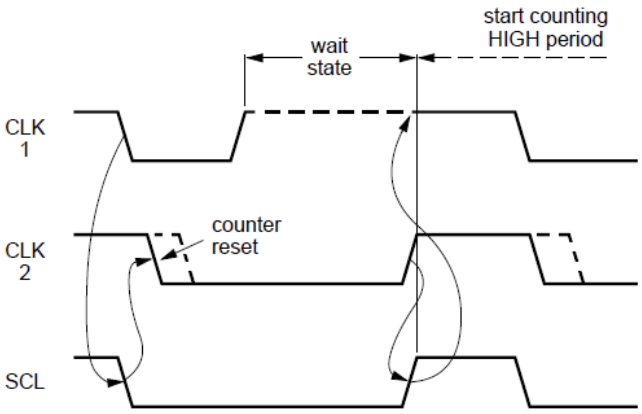



**Figure 59 SCL Synchronization**

## Arbitration

Arbitration refers to a portion of the protocol required only if more than two masters are used in the system. Two masters may generate a START condition together within the hold time of the START condition which results in a valid START condition on the bus. Arbitration is then required to decide which master will complete its transaction. Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks to see if the SDA level matches what it has sent. This process may take many bits. Two masters can actually complete an entire transaction without error, as long as the transmission is identical. The first time a master tries to send a HIGH, but detects the SDA level is low, the master knows that it has lost arbitration and turns off the transaction.

The arbitration is checked only in a few states. Arbitration is checked only when the master is sending data on the SDA line.

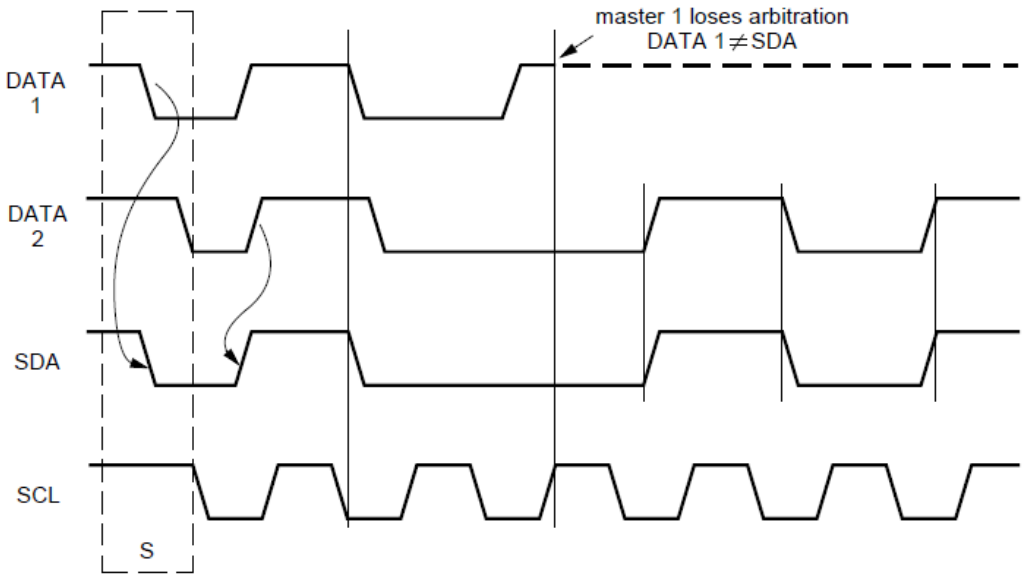



**Figure 60 Arbitration**

## Start Byte

Microcontrollers can be connected to the I2C-bus in two ways. A microcontroller with an on-chip hardware I2C-bus interface can be programmed to be only interrupted by requests from the bus. When the device does not have such an interface, it must constantly monitor the bus via software. There is therefore a speed difference between fast hardware devices and a relatively slow microcontroller which relies on software polling. In this case, data transfer can be preceded by a start procedure which is much longer than normal (please see Figure 61). The start procedure consists of:

- A START condition (S)
- A START byte (0000 0001)

- An acknowledge clock pulse (ACK)
- A repeated START condition (Sr).

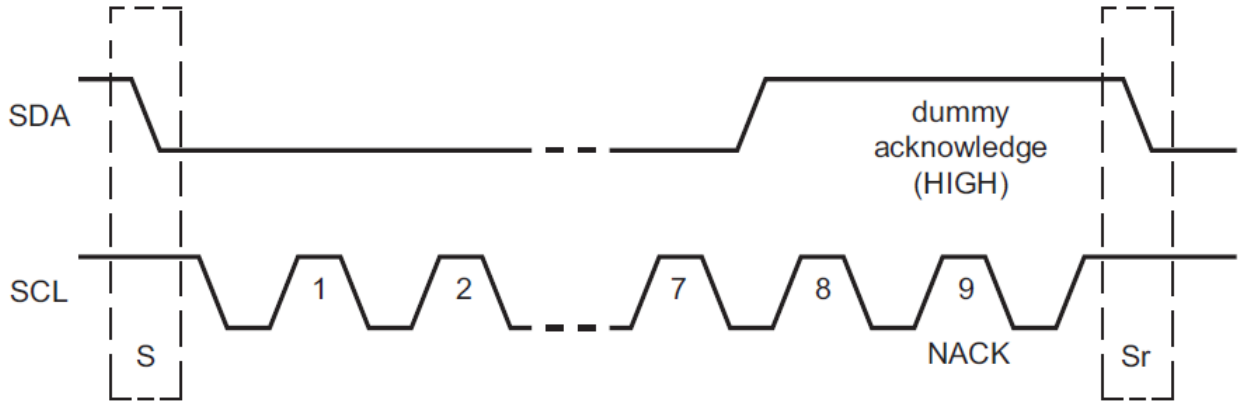



**Figure 61 Start Byte Procedure**

After the START condition S has been transmitted by a master which requires bus access, the START byte (0000 0001) is transmitted. Another microcontroller can therefore sample the SDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this LOW level on the SDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition which is then used for synchronization. A hardware receiver resets upon receipt of the repeated START condition and therefore ignores the START byte. An acknowledge-related clock pulse is generated after the START byte. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

#### 12.2.1.2 I2C Slave

Pin configuration of serial interface in I2C slave mode is as follows.

- SCLK (SCL)
- SDIO (SDA)
- SDI ($A_1$)
- CSB ($A_0$)
- GPIOx ($A_2$) Note: x = 1, 2, 4, 5.

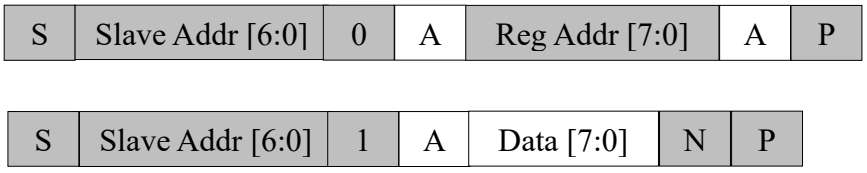
The device uses the SDIO and SCLK pins for a 2-wire serial interface that operates up to 400 Kb/s in Read and Write modes. It complies with the I2C bus standard. The I2C access protocol in device is byte access (random access) only for Write mode and both random and sequential access for Read mode.

The I2C serial interface can operate at either Standard rate (100 Kbps) or Fast rate (400 Kbps).

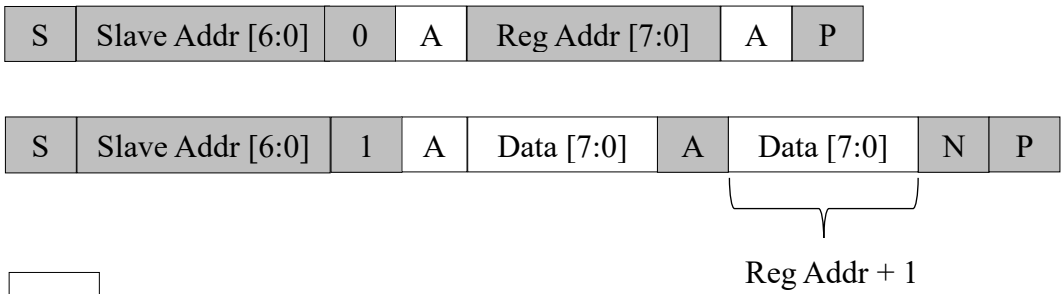
For AU5508 devices, the default Slave address is 1011, {GPIOx}, {SDI}, {CSB} where GPIOx, SDI and CSB values are controlled by pins on the device in the I2C mode. The default address is 0x5B.

The device also supports variable Slave addresses which can be provided via the EFUSE. The LSBs A2, A1 and A0 are controlled via the pins on the device. This allows eight choices of Slave addresses for any system where in the first 4 bits of the slave address can be the same. Device supports disable A2 address from the pin by configuring the GPIO or from the I2C slave address A2 override registers (override enable => Page0 reg39[1], override value => Page0 reg3a[1]) of NVM. If no GPIO is configured for the I2C slave address A2 functionality, and if it is not overridden from the I2C slave address A2 override register, then A2 will be set to 0. In this case default I2C slave address will be 10110{SDI},{CSB}. In this scenario four devices can be controlled.

**Read Operation - Single Byte**

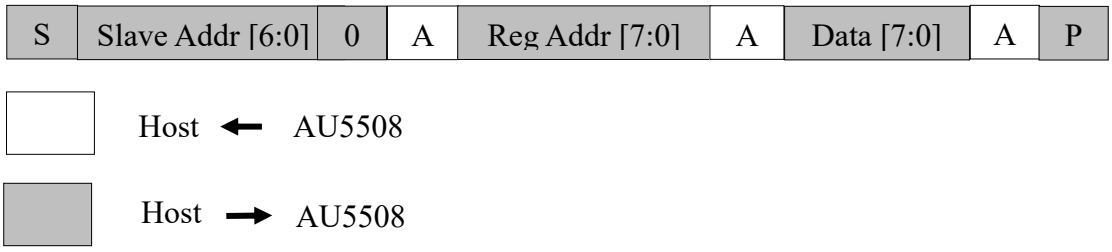



**Read Operation - Burst (Auto Address Increment)**




**1- Read, 0 - Write, A - Acknowledge, N - Not Acknowledge, S - Start Condition, P - Stop Condition**

**Write Operation - Single Byte**




**1- Read, 0 - Write, A – Acknowledge (SDA LOW), N - Not Acknowledge (SDA HIGH), S - Start Condition, P - Stop Condition**

**Single Byte Write**

- The master initiates the transaction by issuing a start condition, writes 7-bit slave address and then the read/write bit is written as 0 (write)
- The slave acknowledges by driving zero on the bus
- The master then writes the 8-bit register map address
- The slave acknowledges by driving zero on the bus
- The master then writes the 8-bit data to be written to the register map address specified
- The slave acknowledges by driving zero on the bus
- The master ends the transaction by issuing a stop condition

**Single Byte Read**

- The master initiates the transaction by issuing a start condition, writes 7-bit slave address and then the read/write bit is written as 0 (write)
- The slave acknowledges by driving zero on the bus
- The master then writes the 8-bit register map address
- The slave acknowledges by driving zero on the bus
- The master ends the transaction by issuing a stop condition
- The master re-initiates the transaction by issuing a start condition, writes 7-bit slave address and then the read/write bit is written as 1 (read)
- The slave then writes the 8-bit data to be written to the register map address specified
- The master does not acknowledge this transaction as the slave may assume a multi-byte read operation and there is a risk of slave holding the bus low
- The master ends the transaction by issuing a stop condition

**Multi Byte Read**

The multi-byte read mode is used to read a continuous segment of the register map. The multi-byte read is faster than performing multiple single byte reads as the device address and register map address need not be specified for every byte read from the register map

- The master initiates the transaction by issuing a start condition, writes 7-bit slave address and then the read/write bit is written as 0 (write)
- The slave acknowledges by driving zero on the bus
- The master then writes the 8 bit register map address
- The slave acknowledges by driving zero on the bus
- The master ends the transaction by issuing a stop condition
- The master re-initiates the transaction by issuing a start condition, writes 7 bit slave address and then the read/write bit is written as 1 (read)
- The slave then writes the 8 bit data to be written to the register map address specified
- The master acknowledges by driving zero on the bus
- The slave automatically increments the register map address and writes the data in at that address to the bus and the master acknowledges
- When all bytes of data are read, master ends the operation by not acknowledging the last read
- The master then ends the transaction by issuing a stop condition

#### 12.2.1.3 I2C Bus Timing Specifications

**Table 28: I2C bus Timing Specifications**

| Description | Symbol | Standard Mode | | Fast Mode | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| SCLK clock frequency | $f_{SCL}$ | – | 100 | – | 400 | kHz |
| Hold time START condition | $t_{HD:STA}$ | 0.4 | – | 0.6 | – | µs |
| Low period of the SCK clock | $t_{LOW}$ | 4.7 | – | 1.3 | – | µs |
| High period of the SCK clock | $t_{HIGH}$ | 4.0 | – | 0.6 | – | µs |
| Setup time for a repeated START condition | $t_{SU:STA}$ | 4.7 | – | 0.6 | – | µs |
| Data hold time | $t_{HD:DAT}$ | 300 | – | 300 | – | ns |
| Data setup time | $t_{SU:DAT}$ | 100 | – | – | – | ns |
| Rise time | $t_R$ | – | 1000 | – | 300 | ns |
| Fall time | $t_F$ | – | 300 | – | 300 | ns |
| Setup time for STOP condition | $t_{SU:STO}$ | 4.0 | – | 0.6 | – | µs |
| Bus-free time between STOP and START conditions | $t_{BUF}$ | 4.7 | – | 1.3 | – | µs |

| Description | Symbol | Standard Mode | | Fast Mode | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| Data valid time | $t_{VD;DAT}$ | – | 3.45 | – | 0.9 | µs |
| Data valid acknowledge time | $t_{VD;ACK}$ | – | 0.9 | – | 0.9 | µs |

Note: In I2C mode, the serial data and clock have an on-chip 25 kΩ pull up resistor to VDDIO. Please refer Figure 62 for the nomenclature of these parameters.

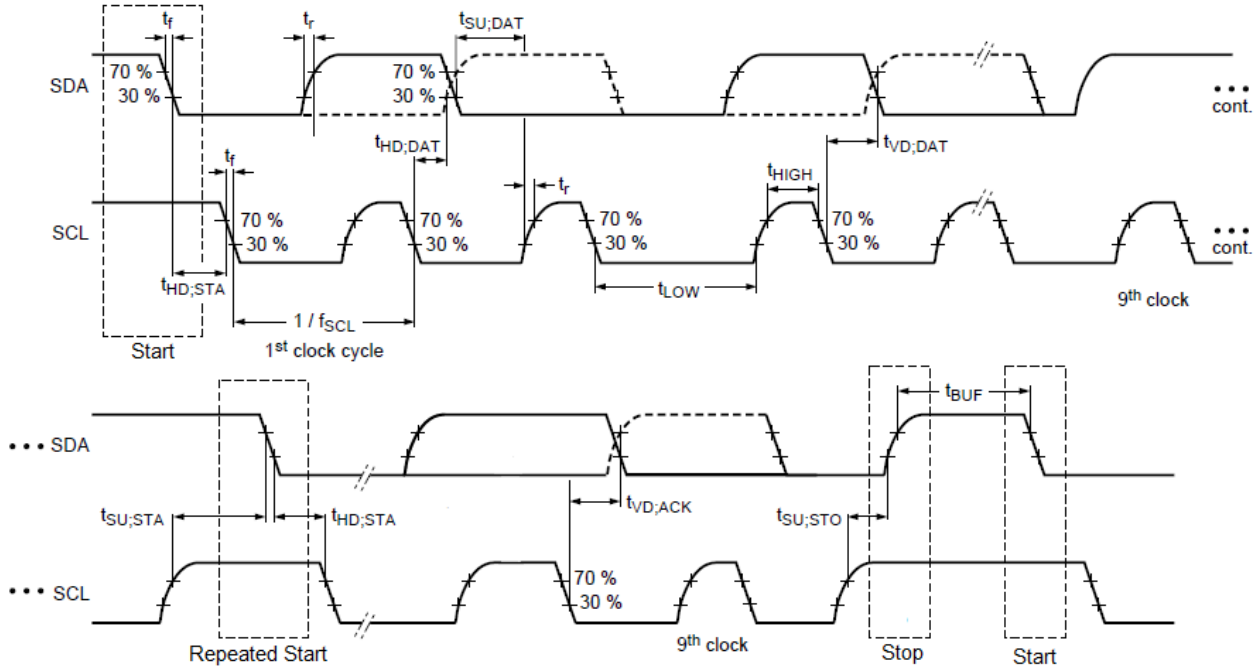



**Figure 62 I2C Timing Waveform**

## 12.2.2 SPI Protocol

Pin configuration of serial interface in SPI mode is as shown in Table 29

.

**Table 29 SPI Pin Configuration**

| PIN | 4 wire mode | 3wire mode |
|---|---|---|
| SCLK | Serial Clock | Serial Clock |
| SDIO | Serial Output | Serial Input/Output (Bi directional) |
| SDI | Serial Input | Not used |
| CSB | Chip Select | Chip Select |

The SPI in a four-pin interface with Chip Select (CSB), Serial Input (SDI), Serial Output (SDIO), and Serial Clock (SCLK) pins. The SPI in a three-pin interface with Chip Select (CSB), Serial Input (SDIO), Serial Output (SDIO), and Serial Clock (SCLK) pins. SDIO pin acts as an input when receiving data from external SPI master, and acts as an output when transmiting data to external SPI master. The SPI bus on the device can run at speed up to 20 MHz. The SPI is a synchronous serial interface, which uses clock and data pins for serial access. When I2C1_SPI0 pin is Low, a Low on the CSB pin activates the SPI access.

1. The SPI can operate up to 20 MHz for write/read operations.
2. The SPI receives serial data from the external master and provides Wr/rdn, address and data to the register map during the write operation.

3. The SPI receives serial data from the external master and provides Wr/rdn and address to the register map and uses the read data obtained from the register map, serializes the same and transmit to the master.
4. In AU5508, the total packet size for each SPI single write or single read transaction is 24 bits where the first 8 bits are instruction byte, the next 8 bits are address and the last 8 bits are data. For incremental write or incremental read operation, the total packet size is 16 bits where the first 8 bits are for instruction byte and the next 8 bits are for data. For burst write or burst read opration, the total packet size is n+2 bytes where first byte is for instruction, second byte is for address and n bytes are for data

**Table 30 SPI Modes of operation**

| Function | Value of Instruction Byte |
|---|---|
| Single Write | 100xxxxx |
| Incremental Write | 101xxxxx |
| Burst Write | 110xxxxx |
| Single Read | 000xxxxx |
| Incremental Read | 001xxxxx |
| Burst Read | 010xxxxx |

5. In AU5508 for single write operation, the master assembles the instruction byte, address and data for write operation on the falling edge of the spi clock and the slave in the AU5508 captures the same on the rising edge of the SPI clock. For incremental write operation the master assembles the instruction byte and data for write operation on the falling edge of the spi clock and the slave in the AU5508 captures the same on the rising edge of the SPI clock. For the burst write operation, the master assembles the instuction bytes, address byte and the data bytes for write operation on the falling edge of the spi clock and the slave in the AU5508 captures the same on the rising edge of the SPI clock.
6. In AU5508 for read operation, the master assembles the instruction byte, address for read operation on the falling edge of the spi clock and the slave in the AU5508 captures the same on the rising edge of the SPI clock. The falling edge after the 16$^{th}$ rising SPI clock (i.e. the last address bit), is used by the slave to assemble the first read data bit which is captured by the master on the 17$^{th}$ edge of the SPI clock. Subsequent 7 more clocks are used for the 7 remaining data bits. For incremental read operation, the master assembles the instruction byte on the falling edge of the SPI clock and the slave in the AU5508 captures the same on the rising edge of the SPI clock. The falling edge after the 8$^{th}$ rising edge of SPI clock (i.e. the last instruction bit), is used by the slave to assemble the first read data bit which is captured by the master on the 9$^{th}$ edge of the SPI clock. Subsequent 7 more clocks are used for the 7 remaining data bits. For burst read operation, the masater assembles the instruction byte, address for read operation on the falling edge of the spi clock and the slave in the AU5508 captures the same on the rising edge of the SPI clock. The falling edge after the 16$^{th}$ rising SPI clock (i.e. the last address bit), is used by the slave to assemble the first read data bit which is captured by the master on the 17$^{th}$ edge of the SPI clock. Subsequent clocks are used for the remaining data bits.
7. In AU5508 the transmitter always sends data on the falling edge of the SPI clock to be captured in the receiver by the rising edge of the SPI clock. The transmitter can be the master for the whole operation of the write and for the control and address portions of the read. The slave is the transmitter during the data portion of the read cycle.
8. In a 4-wire mode of operation, the output data bit slave is controlled only when transmitting the read data bits and is in HiZ for the rest of the times. In 3-wire mode of operation, the data bit is always considered as an input except when the read data is being transimitted by chaning the direction to output.

#### 12.2.2.1 SPI Timing Specifications

**Table 31 SPI Timing Specifications**

| Description | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| SCLK clock frequency | $f_{SCLK}$ | - | – | 20 | MHz |
| Clock pulse width HIGH | $t_{CH}$ | 20 | | | ns |
| Clock pulse width LOW | $t_{CL}$ | 20 | | | ns |
| CSB HIGH time | $t_{CS}$ | 50 | | | ns |
| CSB setup time | $t_{CSS}$ | 25 | | | ns |
| CSB hold time | $t_{CSH}$ | 25 | | | ns |
| Data in setup time | $t_{SD}$ | 10 | | | ns |
| Data in hold time | $t_{HD}$ | 10 | | | ns |
| Output valid | $t_{CO}$ | | | 10 | ns |
| Output setup time | $t_{SO}$ | | | 10 | ns |
| Output hold time | $t_{HO}$ | | | 10 | ns |

Note: In SPI mode, the SDIO pin is driven to HiZ when CSB is high for the chip in SPI Mode. Please refer Figure 63 for the nomenclature of these parameters.

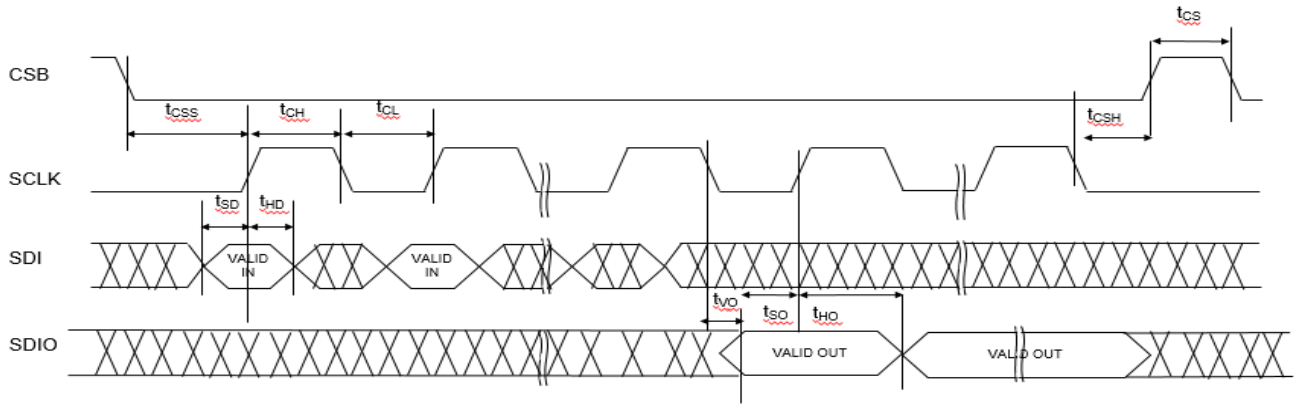



**Figure 63 SPI Timing Diagram**

### 12.2.2.2 SPI Single byte write

- The master initiates the transaction by issuing a start condition by pulling csb_i to active low
- The master assembles the serial data on the falling edge of the SPI clock so the SPI slave can capture the same on the rising edge of the SPI clock
- The first 8 bits are instruction bits
- The next 8 bits (second byte) are used for the register map address
- The next 8 bits (third byte) are used for the register map data
- The 24th rising edge of the SPI clock is used to capture the last data bit. The SPI slave then assembles the address, data, enable and wr_rdn to the PIF slave block. The inverted version of the next falling edge of the SPI clock is used by the SPI slave to capture the address, data, enable and wr_rdn to write to the respective registers.
- The CSB is then de-activated (by going high) by the master
- For the next write operation, CSB is held high for at least a duration of two spi clocks following which the entire operation can start again.

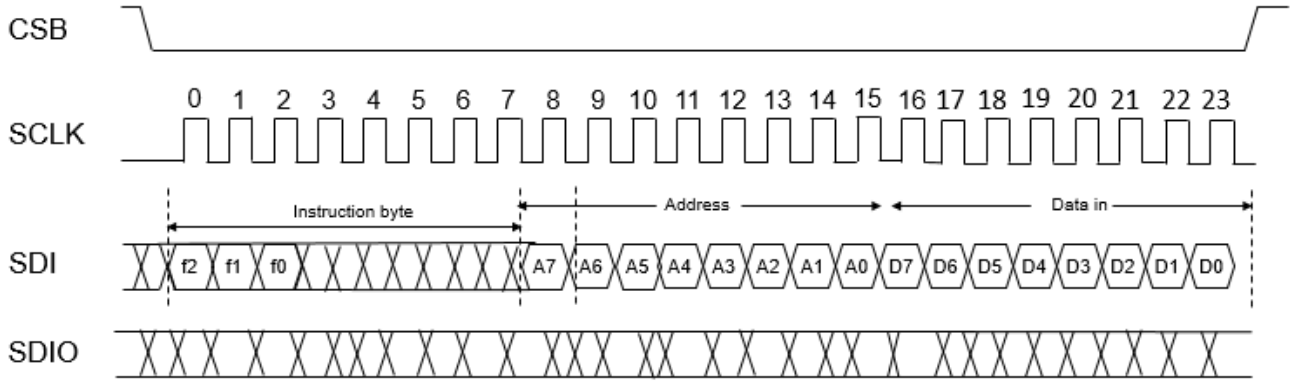



**Figure 64 SPI Single Write – AU5508**

### 12.2.2.3 SPI Incremental byte write

- The master initiates the transaction by issuing a start condition by pulling csb_i to active low
- The master assembles the serial data on the falling edge of the SPI clock so the SPI slave can capture the same on the rising edge of the SPI clock
- The first 8 bits are instruction bits
- The next 8 bits (second byte) are used for the register map data

- The address is an automatic increment of previous address used. Hence the master should have done a single write transaction before starting the incremental write operation
- The 16th rising edge of the SPI clock is used to capture the last data bit. The SPI slave then assembles the address, data, enable and wr_rdn to the PIF slave block. The inverted version of the next falling edge of the SPI clock is used by the SPI slave to capture the address, data, enable and wr_rdn to write to the respective registers.
- The CSB is then de-activated (by going high) by the master
- For the next write operation, CSB is held high for at least a duration of two spi clocks following which the entire operation can start again.

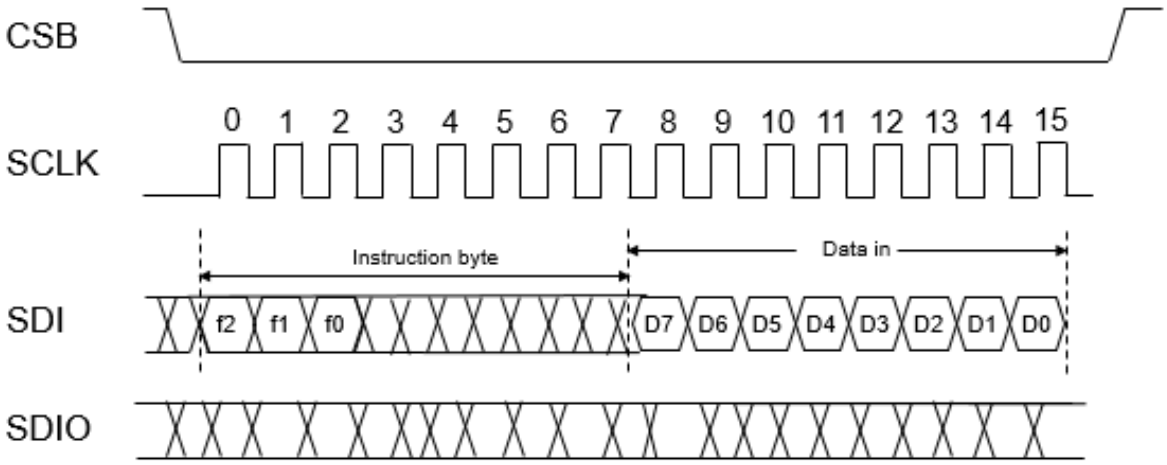



**Figure 65 SPI Incrémental Write – AU5508**

#### 12.2.2.4 SPI Burst write

- The master initiates the transaction by issuing a start condition by pulling csb_i to active low
- The master assembles the serial data on the falling edge of the SPI clock so the SPI slave can capture the same on the rising edge of the SPI clock
- The first 8 bits are instruction bits
- The next 8 bits (second byte) are used for the register map address
- The next sets of 8 bits (n bytes) are used for the register map data for n write operations.
- The 24th rising edge of the SPI clock is used to capture the last data bit of the first data byte. The SPI slave then assembles the address, data, enable and wr_rdn to the PIF slave block. The inverted version of the next falling edge of the SPI clock is used by the SPI slave to capture the address, data, enable and wr_rdn to write to the respective registers. Following this after every 8 clocks, a write operation is performed for all the n bytes.
- The CSB is then de-activated (by going high) by the master
- For the next write operation, CSB is held high for at least a duration of two spi clocks following which the entire operation can start again.

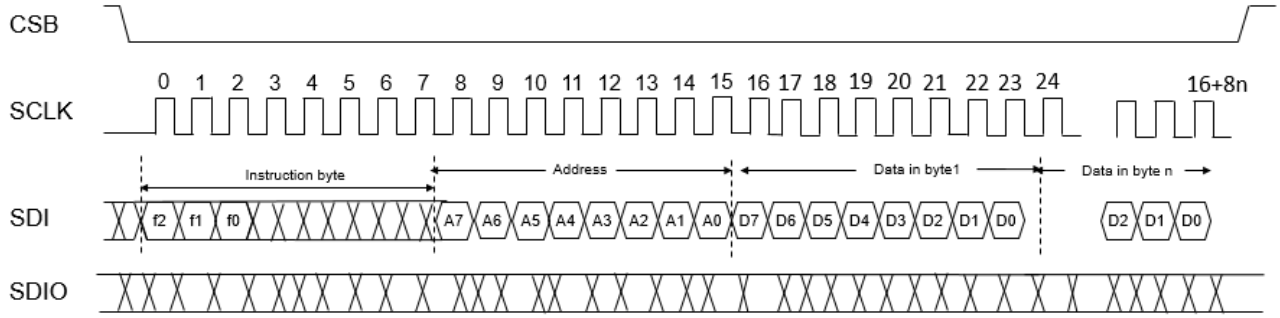



**Figure 66 SPI Burst Write – AU5508**

### 12.2.2.5 SPI Single byte Read

- The master initiates the transaction by issuing a start condition by pulling csb_i to active low
- The master assembles the serial data on the falling edge of the SPI clock so the SPI slave can capture the same on the rising edge of the SPI clock
- The first 8 bits are instruction bits
- The next 8 bits (second byte) are used for the register map address
- The next 8 bits (third byte) are used for the register map data
- The 16th rising edge of the SPI clock is used to capture the last address bit. The SPI slave then assembles the address, enable and wr_rdn to the PIF slave block. The PIF slave block then returns the 8 bit read data where the first bit is transmitted on the falling edge after the 16th clock to be captured by the master on the 17th clock. After 7 more clocks all the bits of the read data are transmitted.
- The CSB is then de-activated (by going high) by the master
- For the next read operation, CSB is held high for at least a duration of two spi clocks following which the entire operation can start again.

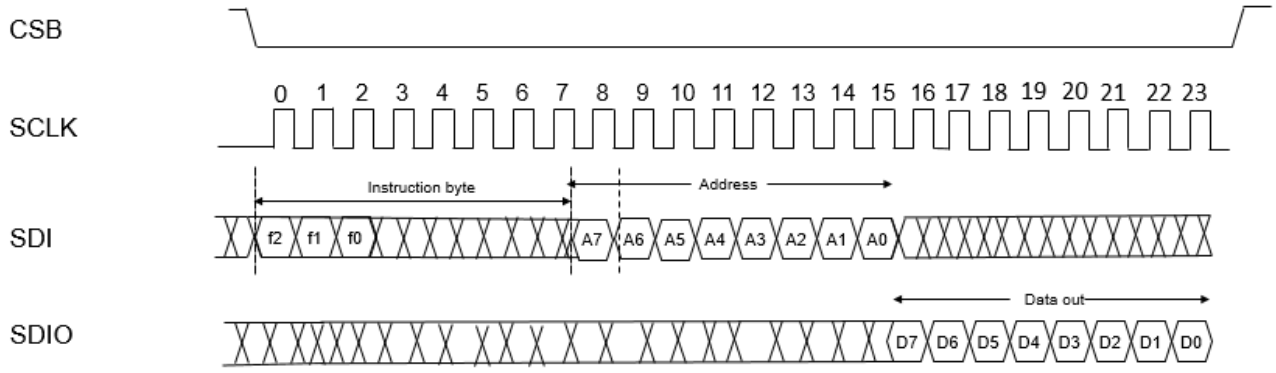



**Figure 67 SPI Single Read – AU5508**

### 12.2.2.6 SPI Incremental byte Read

- The master initiates the transaction by issuing a start condition by pulling csb_i to active low
- The master assembles the serial data on the falling edge of the SPI clock so the SPI slave can capture the same on the rising edge of the SPI clock
- The first 8 bits are instruction bits
- The devices use an auto-increment of the previous used address for current read operation. The master should have hence done at least one single write/read operation for this increment read operation to work.
- The 8th rising edge of the SPI clock is used to capture the last instruction bit. The SPI slave then assembles the address, enable and wr_rdn to the PIF slave block. The PIF slave block then returns the 8 bit read data

where the first bit is transmitted on the falling edge after the 8th clock to be captured by the master on the 9th clock. After 7 more clocks all the bits of the read data are transmitted.

- The CSB is then de-activated (by going high) by the master
- For the next read operation, CSB is held high for at least a duration of two spi clocks following which the entire operation can start again.

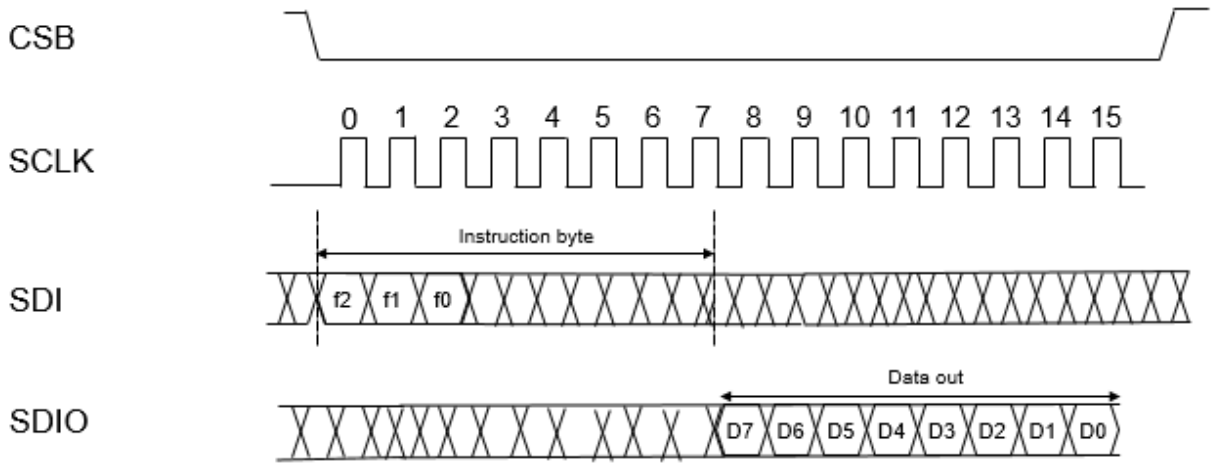



**Figure 68 SPI Incremental Read – AU5508**

### 12.2.2.7 SPI Burst byte Read

- The master initiates the transaction by issuing a start condition by pulling csb_i to active low
- The master assembles the serial data on the falling edge of the SPI clock so the SPI slave can capture the same on the rising edge of the SPI clock
- The first 8 bits are instruction bits
- The next 8 bits (second byte) are used for the register map address
- The 16th rising edge of the SPI clock is used to capture the last address bit. The SPI slave then assembles the address, enable and wr_rdn to the PIF slave block. The PIF slave block then returns the 8 bit read data where the first bit is transmitted on the falling edge after the 16th clock to be captured by the master on the 17th clock. After 7 more clocks all the bits of the first read data are transmitted. For subsequent bytes to be read, the address is auto-incremented and 8 clocks are used to transmit for each of the n bytes
- The CSB is then de-activated (by going high) by the master
  For the next read operation, CSB is held high for at least a duration of two spi clocks following which the entire operation can start again.

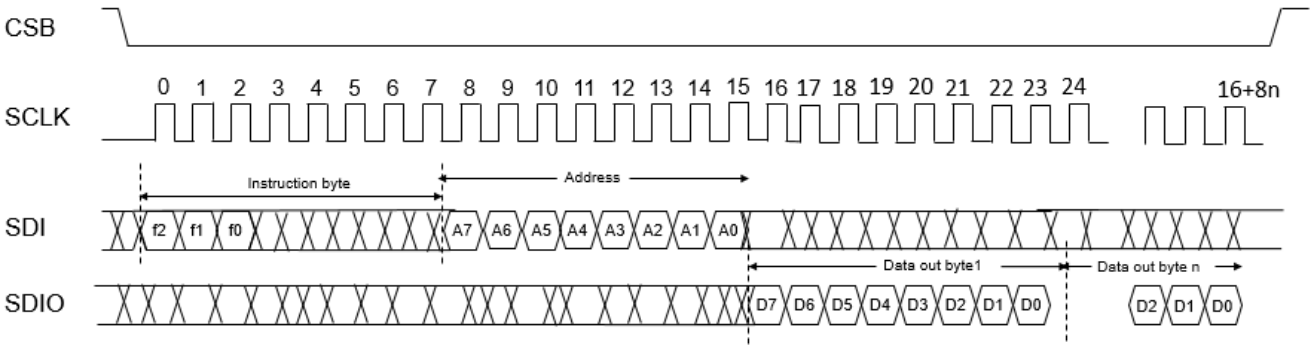



**Figure 69 SPI Burst Read – AU5508**

# 13 Monitoring through the register map read back: Status and Notify

AU5508 provides various Status and notify bits that can be accessed from the register map. Below are the details of the procedure to be followed to access the same.

The alarm registers are a set of three types of registers distributed between the various pages as described in Table 32 and illustrated with some examples.

- The Status registers are the current dynamic status of a defect. The live status defects are active high with a '1' indicating the defect is present.
- The Notify registers are the sticky bits for a defect. Sometimes, we may get a very short pulse for the status and it may not be possible for the external user to capture the same. Hence a notify register is provided. The notify register is set to 1 whenever there is a rising edge of the corresponding status register. This is a sticky bit and stays at 1 till the user writes a 1 to that specific bit to clear it.
- Each notify has a masking bit to enable or disable its operation. The notify sticky bit operates only if the corresponding masking bit is set to 1. If the masking register bit is set to 0, notify will not be asserted even when status toggles. The default value for the mask register is 0xff so all the notify signals are enabled. Once the user writes a 1 to clear the notify, the notify bit can again go high on the next rising edge of the status.

These registers operate on the internal 4 MHz RC clock. When there is a defect (i.e. status) of any bit in register, it gets asserted and de-asserted in a live mode. User can read the corresponding register location to see the current status at any time.

On the other hand, the default value of the notify and masking register is 0xff – user has to write 0xff to both these registers to clear them at the beginning and use all notifies.It is recommended to clear all notifies after programming a profile.

The INTRB pin is used as a NOR operation of the selected notifies. The choice of the Notify listing that is used for the INTRB pin is selectable in the GUI when creating the profile. The sticky notifies that are selected and used for INTRB need to be cleared for restoring the INTRB to 1 for further sticky defect monitoring using this pin.

## 13.1 Tabular Listing

The Table 32 details the name of the alarm, the page it is located in, the address of status, notify and mask registers in that page and the bit number in the address. In order to access a page of the register map, the particular page number has to be written to address 0xff. For instance, to either write to or read from page 1, first the user needs to write to 0xff a value of 0x01. Following this, any number of write or read operations can be done with page 1.

**Table 32 Tabular Listing of Alarm Registers**

| Sr No | Name of Signal | Description | Pg no | Status Register | | Notify Register | | Mask Register | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Register Address | Bit No | Register Address | Bit No | Register Address | Bit No |
| 1 | xoclk_loss | XO clock Loss | 0 | 0x02 | 4 | 0x03 | 4 | 0x04 | 4 |
| 2 | cmon_pll_inner_lol | clock mon pll inner loss of lock | | | 5 | | 5 | | 5 |
| 3 | plla_outer_lol | plla outer loss of lock | 0 | 0x06 | 0 | 0x07 | 0 | 0x08 | 0 |
| 4 | pllb_outer_lol | pllb outer loss of lock | | | 1 | | 1 | | 1 |
| 5 | pllc_outer_lol | pllc outer loss of lock | | | 2 | | 2 | | 2 |
| 6 | plld_outer_lol | plld outer loss of lock | | | 3 | | 3 | | 3 |
| 9 | plla_ho_freeze | plla hold over freeze | 0 | 0x0a | 0 | 0x0b | 0 | 0x0c | 0 |
| 10 | pllb_ho_freeze | pllb hold over freeze | | | 1 | | 1 | | 1 |
| 11 | pllc_ho_freeze | pllc hold over freeze | | | 2 | | 2 | | 2 |
| 12 | plld_ho_freeze | plld hold over freeze | | | 3 | | 3 | | 3 |
| 15 | plla_inner_lol | plla inner loop loss of lock | 0 | 0x92 | 0 | 0x93 | 0 | 0x94 | 0 |
| 16 | pllb_inner_lol | pllb inner loop loss of lock | | | 1 | | 1 | | 1 |
| 17 | pllc_inner_lol | pllc inner loop loss of lock | | | 2 | | 2 | | 2 |
| 18 | plld_inner_lol | plld inner loop loss of lock | | | 3 | | 3 | | 3 |
| 21 | eeprom_ctrl_eeprom_read_done | eeprom ctrl eeprom read done | 0 | 0x96 | 0 | 0x97 | 0 | 0x98 | 0 |
| 22 | eeprom_ctrl_crc_deft | eeprom ctrl crc defect | | | 1 | | 1 | | 1 |

| Sr No | Name of Signal | Description | Pg no | Status Register | | Notify Register | | Mask Register | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Register Address | Bit No | Register Address | Bit No | Register Address | Bit No |
| 23 | eeprom_ctrl_arb_lost_deft | eeprom ctrl arb lost defect | | | 2 | | 2 | | 2 |
| 24 | eeprom_ctrl_page_id_deft | eeprom ctrl page id defect | | | 3 | | 3 | | 3 |
| 25 | eeprom_ctrl_dev_id_deft | eeprom ctrl dev id defect | | | 4 | | 4 | | 4 |
| 26 | eeprom_ctrl_eeprom_size_deft | eeprom ctrl eeprom size defect | | | 5 | | 5 | | 5 |
| 27 | eeprom_ctrl_word_add_deft | eeprom ctrl word add defect | | | 6 | | 6 | | 6 |
| 28 | eeprom_ctrl_dev_add_deft | eeprom ctrl dev add defect | | | 7 | | 7 | | 7 |
| 29 | eeprom_ctrl_start_timeout_deft | eeprom ctrl start timeout defect | 0 | 0x9a | 0 | 0x9b | 0 | 0x9c | 0 |
| 30 | eeprom_ctrl_data_deft | eeprom ctrl data defect | | | 1 | | 1 | | 1 |
| 31 | eeprom_ctrl_stop_deft | eeprom ctrl stop defect | | | 2 | | 2 | | 2 |
| 32 | eeprom_ctrl_start_deft | eeprom ctrl start defect | | | 3 | | 3 | | 3 |
| 33 | eeprom_ctrl_read_done_timeout_deft | eeprom ctrl read done timeout defect | | | 4 | | 4 | | 4 |
| 34 | plla_phase_loss_of_lock | plla phase loss of lock | 0 | 0x0a | 6 | 0x0b | 6 | 0x0c | 6 |
| 35 | pllb_phase_loss_of_lock | pllb phase loss of lock | | | 7 | | 7 | | 7 |
| 36 | pllc_phase_loss_of_lock | pllc phase loss of lock | 0 | 0x92 | 6 | 0x93 | 6 | 0x94 | 6 |
| 37 | plld_phase_loss_of_lock | plld phase loss of lock | | | 7 | | 7 | | 7 |
| 40 | plla_tdc_data_ready | plla tdc data ready | 0 | 0x02 | 6 | 0x03 | 6 | 0x04 | 6 |
| 41 | pllb_tdc_data_ready | pllb tdc data ready | | | 7 | | 7 | | 7 |
| 42 | pllc_tdc_data_ready | pllc tdc data ready | 0 | 0x06 | 6 | 0x07 | 6 | 0x08 | 6 |
| 43 | plld_tdc_data_ready | plld tdc data ready | | | 7 | | 7 | | 7 |
| 44 | clk0p_freq_fine_drifted | clk0p freq fine drifted | 6 | 0x02 | 0 | 0x03 | 0 | 0x04 | 0 |
| 45 | clk0p_freq_coarse_drifted | clk0p freq coarse drifted | | | 1 | | 1 | | 1 |
| 46 | clk_in0p_loss | clk in0p loss | | | 2 | | 2 | | 2 |
| 47 | clk_in0p_loss_with_FD | clk in0p loss with FD | | | 3 | | 3 | | 3 |
| 48 | clk1p_freq_fine_drifted | clk1p freq fine drifted | | | 4 | | 4 | | 4 |
| 49 | clk1p_freq_coarse_drifted | clk1p freq coarse drifted | | | 5 | | 5 | | 5 |
| 50 | clk_in1p_loss | clk in1p loss | | | 6 | | 6 | | 6 |
| 51 | clk_in1p_loss_with_FD | clk in1p loss with FD | | | 7 | | 7 | | 7 |
| 52 | clk2p_freq_fine_drifted | clk2p freq fine drifted | 6 | 0x06 | 0 | 0x07 | 0 | 0x08 | 0 |
| 53 | clk2p_freq_coarse_drifted | clk2p freq coarse drifted | | | 1 | | 1 | | 1 |
| 54 | clk_in2p_loss | clk in2p loss | | | 2 | | 2 | | 2 |
| 55 | clk_in2p_loss_with_FD | clk in2p loss with FD | | | 3 | | 3 | | 3 |
| 56 | clk3p_freq_fine_drifted | clk3p freq fine drifted | | | 4 | | 4 | | 4 |
| 57 | clk3p_freq_coarse_drifted | clk3p freq coarse drifted | | | 5 | | 5 | | 5 |
| 58 | clk_in3p_loss | clk in3p loss | | | 6 | | 6 | | 6 |
| 59 | clk_in3p_loss_with_FD | clk in3p loss with FD | | | 7 | | 7 | | 7 |
| 60 | clk4p_freq_fine_drifted | clk4p freq fine drifted | 6 | 0x0a | 0 | 0x0b | 0 | 0x0c | 0 |
| 61 | clk4p_freq_coarse_drifted | clk4p freq coarse drifted | | | 1 | | 1 | | 1 |
| 62 | clk_in4p_loss | clk in4p loss | | | 2 | | 2 | | 2 |
| 63 | clk_in4p_loss_with_FD | clk in4p loss with FD | | | 3 | | 3 | | 3 |
| 64 | ocxo_in5_freq_fine_drifted | ocxo in5 freq fine drifted | | | 4 | | 4 | | 4 |

| Sr No | Name of Signal | Description | Pg no | Status Register | | Notify Register | | Mask Register | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Register Address | Bit No | Register Address | Bit No | Register Address | Bit No |
| 65 | ocxo_in5_freq_coarse_drifted | ocxo in5 freq coarse drifted | | | 5 | | 5 | | 5 |
| 66 | clk_ocxo_in5_loss | clk ocxo in5 loss | | | 6 | | 6 | | 6 |
| 67 | clk_ocxo_in5_loss_with_FD | clk ocxo in5 loss with FD | | | 7 | | 7 | | 7 |
| 68 | clk0n_freq_fine_drifted | clk0n freq fine drifted | 6 | 0x92 | 0 | 0x93 | 0 | 0x94 | 0 |
| 69 | clk0n_freq_coarse_drifted | clk0n freq coarse drifted | | | 1 | | 1 | | 1 |
| 70 | clk_in0n_loss | clk in0n loss | | | 2 | | 2 | | 2 |
| 71 | clk_in0n_loss_with_FD | clk in0n loss with FD | | | 3 | | 3 | | 3 |
| 72 | clk1n_freq_fine_drifted | clk1n freq fine drifted | 6 | 0x92 | 4 | 0x93 | 4 | 0x94 | 4 |
| 73 | clk1n_freq_coarse_drifted | clk1n freq coarse drifted | | | 5 | | 5 | | 5 |
| 74 | clk_in1n_loss | clk in1n loss | | | 6 | | 6 | | 6 |
| 75 | clk_in1n_loss_with_FD | clk in1n loss with FD | | | 7 | | 7 | | 7 |
| 76 | clk2n_freq_fine_drifted | clk2n freq fine drifted | 6 | 0x96 | 0 | 0x97 | 0 | 0x98 | 0 |
| 77 | clk2n_freq_coarse_drifted | clk2n freq coarse drifted | | | 1 | | 1 | | 1 |
| 78 | clk_in2n_loss | clk in2n loss | | | 2 | | 2 | | 2 |
| 79 | clk_in2n_loss_with_FD | clk in2n loss with FD | | | 3 | | 3 | | 3 |
| 80 | clk3n_freq_fine_drifted | clk3n freq fine drifted | | | 4 | | 4 | | 4 |
| 81 | clk3n_freq_coarse_drifted | clk3n freq coarse drifted | | | 5 | | 5 | | 5 |
| 82 | clk_in3n_loss | clk in3n loss | | | 6 | | 6 | | 6 |
| 83 | clk_in3n_loss_with_FD | clk in3n loss with FD | | | 7 | | 7 | | 7 |
| 84 | clk4n_freq_fine_drifted | clk4n freq fine drifted | 6 | 0x9a | 0 | 0x0b | 0 | 0x9c | 0 |
| 85 | clk4n_freq_coarse_drifted | clk4n freq coarse drifted | | | 1 | | 1 | | 1 |
| 86 | clk_in4n_loss | clk in4n loss | | | 2 | | 2 | | 2 |
| 87 | clk_in4n_loss_with_FD | clk in4n loss with FD | | | 3 | | 3 | | 3 |

## 13.2 Examples for Live Status Read Back

Some examples are presented based on the table above for reading the live status of the defects.

In the pseudo code presented below:

wr_cmd(address, data): refers to a “Write Command” where the corresponding data is written in to the specified register address

x= rd_cmd(address): refers to a “Read Command” where the corresponding data is read from the specified register address and stored in the variable ‘x’

y >> x: denotes a bit wise right shift on the number y by x bit locations

y << x:denotes a bit wise left shift on the number y by x bit locations

& is the logical AND operation (bit wise)

Dynamic registers to read the various alarm registers in the RealTime page.

1. Input Clocks CL and FD Related Real Time live status read back:

wr_cmd(0xff, 0x06)  Program the CLKMON_SYS page number

### Clock Monitor dynamic status

clock_mon_dyn_status = rd_cmd(0x02) & 0xff

| | | |
|---|---|---|
| (clock_mon_dyn_status >> 0) & 0x01 | INP0 Status for FD (fine), | Read bit position [0] |
| (clock_mon_dyn_status >> 1) & 0x01 | INP0 Status for FD (coarse), | Read bit position [1] |
| (clock_mon_dyn_status >> 2) & 0x01 | INP0 Status for CL, | Read bit position [2] |
| (clock_mon_dyn_status >> 3) & 0x01 | INP0 Status for CL with FD, | Read bit position [3] |
| (clock_mon_dyn_status >> 4) & 0x01 | INP1 Status for FD (fine), | Read bit position [4] |
| (clock_mon_dyn_status >> 5) & 0x01 | INP1 Status for FD (coarse), | Read bit position [5] |
| (clock_mon_dyn_status >> 6) & 0x01 | INP1 Status for CL, | Read bit position [6] |
| (clock_mon_dyn_status >> 7) & 0x01 | INP1 Status for CL with FD, | Read bit position [7] |

### 2. PLL Related Real Time live status read back:

wr_cmd(0xff, 0x00)  Program the MAIN_SYS page number, Page 0

### PLL Lock Loss dynamic status

pll_outer_lol_dyn_status = rd_cmd(0x06) & 0xff

| | |
|---|---|
| (pll_outer_lol_dyn_status >> 0) & 0x01 | PLLA Status for Outer LL, Read bit position [0] |
| (pll_outer_lol_dyn_status >> 1) & 0x01 | PLLB Status for Outer LL, Read bit position [1] |
| (pll_outer_lol_dyn_status >> 2) & 0x01 | PLLC Status for Outer LL, Read bit position [2] |
| (pll_outer_lol_dyn_status >> 3) & 0x01 | PLLD Status for Outer LL, Read bit position [3] |

### Holdover Status

pll_ho_freeze_dyn_status = rd_cmd(0x0a) & 0xff

| | |
|---|---|
| (pll_ho_freeze_dyn_status >> 0) & 0x01 | PLLA Status for HO, Read bit position [0] |
| (pll_ho_freeze_dyn_status >> 1) & 0x01 | PLLB Status for HO, Read bit position [1] |
| (pll_ho_freeze_dyn_status >> 2) & 0x01 | PLLC Status for HO, Read bit position [2] |
| (pll_ho_freeze_dyn_status >> 3) & 0x01 | PLLD Status for HO, Read bit position [3] |

### 3. XO clock loss Related Real Time live status read back:

### CLOS_XAXB, XO Clock Loss

wr_cmd(0xff, 0x00)  Program the MAIN_SYS page number, Page 0

clos_xAxB = rd_cmd(0x02) & 0x10 // XO CL Status, Read bit position [2]

## 13.3 Examples of Sticky Bit Clearing

As described earlier, the sticky notify bits are cleared by writing a '1' to the corresponding notify bit itself. The notify bit by itself is enabled by writing a '1' to the corresponding mask bit.

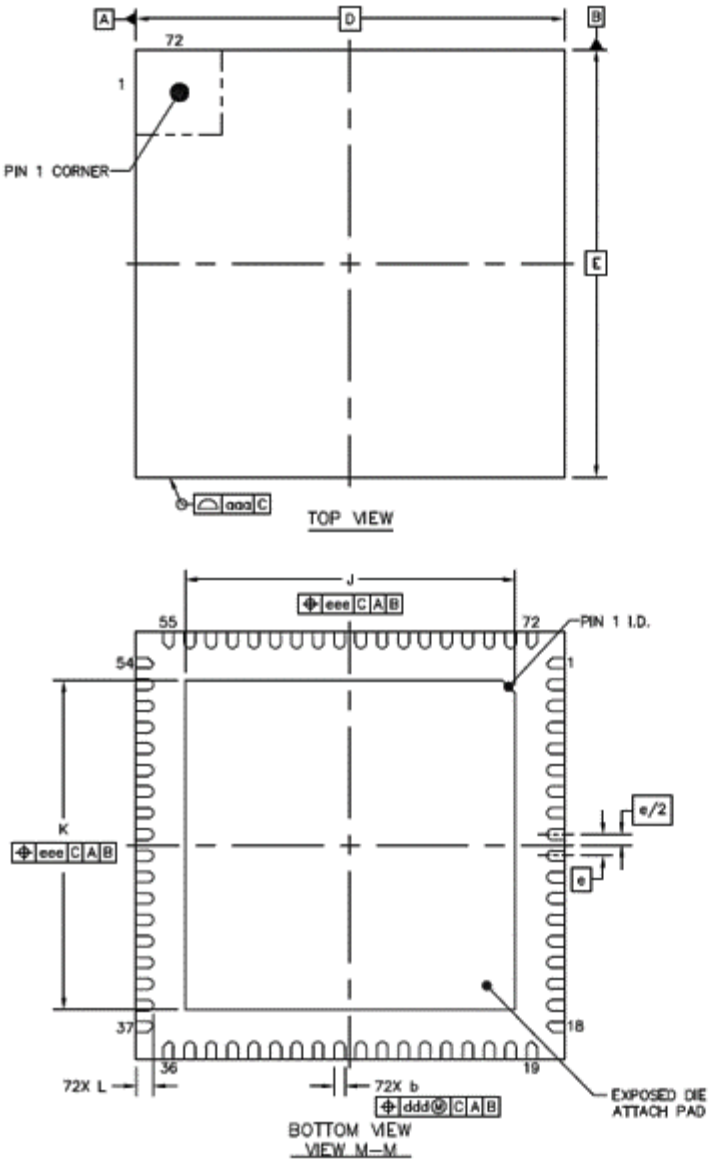
In the pseudo code presented below,

rmw_cmd(addr,bit_loc,no_of_bits,data): denotes the read/modify/write operation where no_of_bits number of bits at bit_loc location (denoted as 7:0) is replaced with the data at address location addr.

```
def clr_intb_CMON_IN0P():
    This function is used to clear the sticky notify for clear clock mon notify for IN0P
    Write the page number
    wr_cmd(0xff, 0x06)
     Information to clr ** Page6: reg03[3:0]=0x0f **
    addr       = 0x3
    bit_loc    = 3
    no_of_bits = 4
    data       = 0x0f
    rmw_cmd(addr,bit_loc,no_of_bits,data)


def clr_intb_PLL_OUTER_LOL():
    This function is used to clear the sticky notify for outer loss of lock notify for all PLLs
    Write the page number
    wr_cmd(0xff, 0)
     Information to clr ** Page 0: reg07[3:0] = 0x0f  **
    addr       = 0x7
    bit_loc    = 3
    no_of_bits = 4
    data       = 0x0f
    rmw_cmd(addr,bit_loc,no_of_bits,data)


def clr_intb_PLL_HO():
    This function is used to clear the sticky notify for outer loss of lock notify for all PLLs
    Write the page number
    wr_cmd(0xff, 0)
     Information to clr ** Page 0: reg0b[3:0] = 0x0f  **
    addr       = 0x7
    bit_loc    = 3
    no_of_bits = 4
    data       = 0x0f
    rmw_cmd(addr,bit_loc,no_of_bits,data)


def clr_intb_XO_CL():
     This function is used to clear the sticky notify for XO Clock Loss
    Write the page number
    wr_cmd(0xff, 0)
     Information to clr ** Page 0: reg03[4]=1 **
    addr       = 0x3
    bit_loc    = 4
    no_of_bits = 1
    data       = 1
    rmw_cmd(addr,bit_loc,no_of_bits,data)


def clr_intrb():
    '''
    This is the main clear function
```

```
 which calls the 4 clear functions
 '''
clr_intb_CMON_IN0P()
clr_intb_PLL_OUTER_LOL()
clr_intb_PLL_HO()
clr_intb_XO_CL()
```

## 14 Package Information




| | | SYMBOL | MIN | NOM | MAX |
|---|---|---|---|---|---|
| TOTAL THICKNESS | | A | 0.8 | 0.85 | 0.9 |
| STAND OFF | | A1 | 0 | 0.035 | 0.05 |
| MOLD THICKNESS | | A2 | --- | 0.65 | --- |
| L/F THICKNESS | | A3 | | 0.203 REF | |
| LEAD WIDTH | | b | 0.2 | 0.25 | 0.3 |
| BODY SIZE | X | D | | 10 BSC | |
| | Y | E | | 10 BSC | |
| LEAD PITCH | | e | | 0.5 BSC | |
| EP SIZE | X | J | 7.6 | 7.7 | 7.8 |
| | Y | K | 7.6 | 7.7 | 7.8 |
| LEAD LENGTH | | L | 0.35 | 0.4 | 0.45 |
| PACKAGE EDGE TOLERANCE | | aaa | | 0.1 | |
| MOLD FLATNESS | | bbb | | 0.1 | |
| COPLANARITY | | ccc | | 0.08 | |
| LEAD OFFSET | | ddd | | 0.1 | |
| EXPOSED PAD OFFSET | | eee | | 0.1 | |

**Figure 70 Package Information**

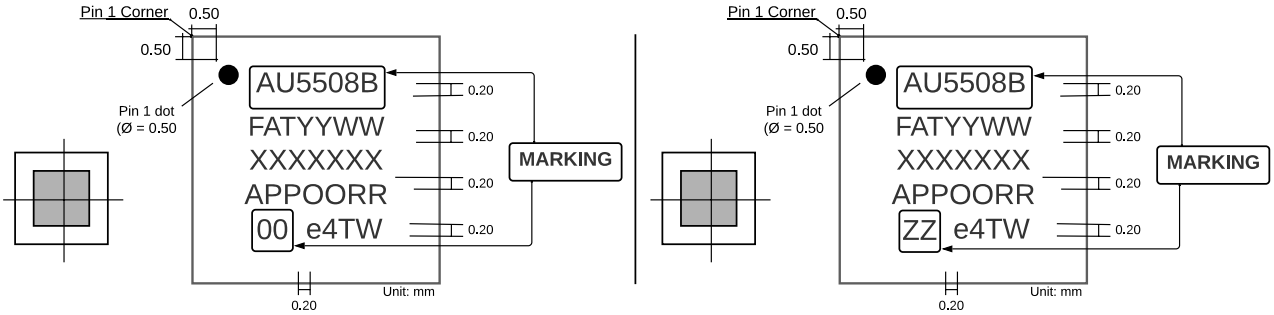
# 15 Ordering Information

**Table 33 Ordering Information for AU5508**

| Ordering Part Number (OPN) | Marking | No. of Input/ Output Clocks | Output Clock Frequency Range (MHz) | Supported Frequency Synthesis modes | Package | Temp Range |
|---|---|---|---|---|---|---|
| AU5508B00-QMR[1,2] | AU5508B | 5/12 | 0.5 Hz-2.94912 GHz | Integer and Fractional | 72-QFN 10x10 mm | –40 to 85 °C |
| AU5508B00-QMT[1,2] | AU5508B | 5/12 | 0.5 Hz-2.94912 GHz | Integer and Fractional | 72-QFN 10x10 mm | –40 to 85 °C |
| AU55xx-EVB | | — | — | — | Evaluation Board | — |

Notes:

1. Add an R at the end of the OPN to denote tape and reel ordering option. Add a T at the end of the OPN to denote tray ordering option.
2. Custom and factory pre-programmed devices are available. Ordering part numbers are assigned by Aurasemi Semiconductor, please contact local sales to request the unique part number.
3. Please refer to 5th row of the die Marking to differentiate between the generic part and pre-programmed part.



# 16 Revision History

**Table 34 Revision History**

| Revision | Date | Description | Author |
|---|---|---|---|
| 1.0 | 18th Jan 2022 | AU5508 Data Sheet Production Version Released | Aurasemi |
| 1.1 | 13th Apr 2022 | 1. Section 4.1: Added delay information between power up to I2C transaction from master driver when EEPROM is not connected<br>2. Section 4.2: Added information about GPIO latching enabled part needs custom profile and part number.<br>3. Section 5: Modified recommended resistor values in Crystal to device interface for better slew rate | Aurasemi |
| 1.2 | 8th Aug 2022 | 1. Table 7- Note 6 updated<br>2. Table 10 - Note 2 and Note 4 updated<br>3.Figure 4 and Figure 5 added to Section 2<br>4. Section 4.2.3 updated in the Input modes.<br>5. Figure 39 typo in VDDO value fixed ( ’18 V’ changed to ‘1.8 V’).<br>6. Figure 42 typo in VDDO value fixed (18V to 1.8 V)<br>7. In Section 15, Table 33 updated with the Ordering Information. | Aurasemi |

## IMPORTANT NOTICE AND DISCLAIMER

AURASEMI PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Aurasemi products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Aurasemi grants you permission to use these resources only for development of an application that uses Aurasemi products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Aurasemi intellectual property or to any third-party intellectual property. Aurasemi disclaims responsibility for, and you will fully indemnify Aurasemi and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Aurasemi products are provided only subject to Aurasemi Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Aurasemi resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

## Contact Information

For more information visit www.aurasemi.com

For sales related information please send an email to sales@aurasemi.com

## Trademarks