Lime Microsystems Limited

Surrey Technology Centre
Occam Road
The Surrey Research Park
Guildford, Surrey GU2 7YG
United Kingdom

Tel: +44 (0) 1483 685 063
e-mail: enquiries@limemicro.com

LimePSB-RPCM GPSDO

- Design description-

Version: 1.00 Last modified: 29/01/2025

REVISION HISTORY

The following table shows the revision history of this document:

Date	Version	Description of Revisions
28/01/2025	1.00	Initial version

Table of Contents

1 Introduction

The LimePSB-RPCM v1.3 board features a Lattice iCE5LP4K FPGA, where the GPS Disciplined Oscillator (GPSDO) logic is implemented. The FPGA receives a 1PPS (Pulse Per Second) signal from the GPS module or other sources, as well as a clock signal from a Voltage Controlled Temperature Compensated Crystal Oscillator (VCTCXO). The system calculates the frequency error based on the 1PPS signal using three averaging intervals: 1 second, 10 seconds, and 100 seconds.

Communication between the host and the FPGA is facilitated via an SPI interface, allowing the host to enable the GPSDO module, configure its parameters, and retrieve frequency error information from the FPGA.

2 GPSDO designs structure

GPSDO implementation is shown in Figure 1. It consist of three main modules:

- **NEO430** – softcore CPU with control algorithm for VCTCXO TAMER module and SPI bus for controlling VTCXO DAC
- **VCTCXO TAMER** – contains actual counters to measure frequency error compared to time pulse.
- **gpsdocfg** - configuration registers accessible via SPI bus.

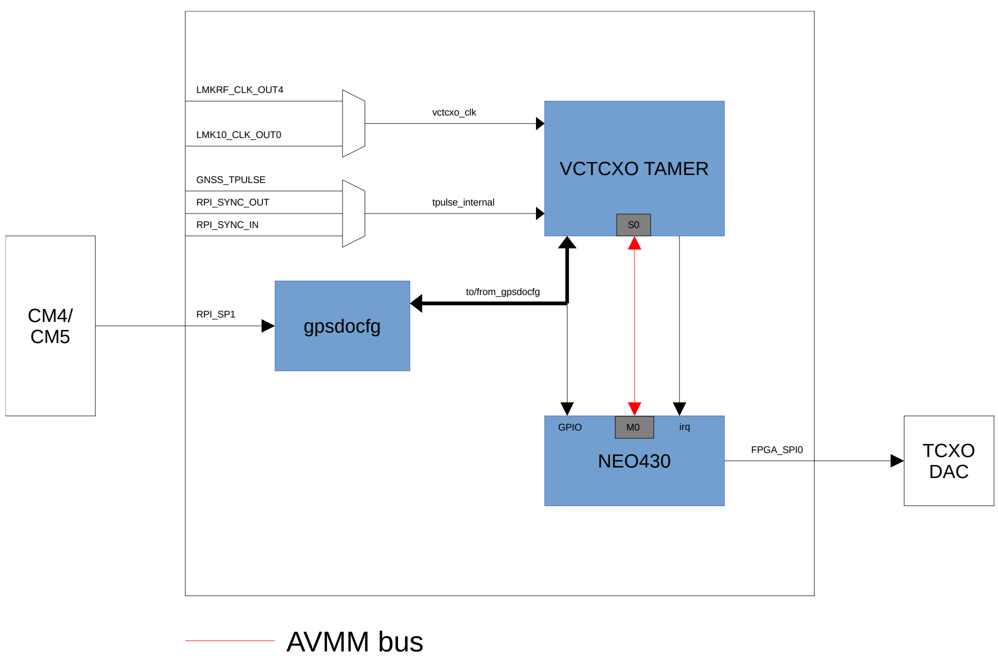

Figure 1: GPSDO implementation block diagram

2.1 Module gpsdocfg

Module gpsdocfg is simple SPI accessible module with configuration registers. SPI interface is connected to RPI_SPI1 lines and uses RPI_SPI1_SS1 chip select. SPI interface features:

- All registers are 16bit wide;
- Operating mode 0 (data is captured on the clock rising edge, while data is shifted on falling edge).
- 32 serial clock cycles are required to complete read/write operations.

A write/read sequence consist of 16-bit instruction followed by a 16-bit data. The MSB of the instruction bit stream is used as SPI command where CMD=1 for write and CMD=0 for read. Basic write sequence can be found in Figure 2 and read sequence in Figure 3

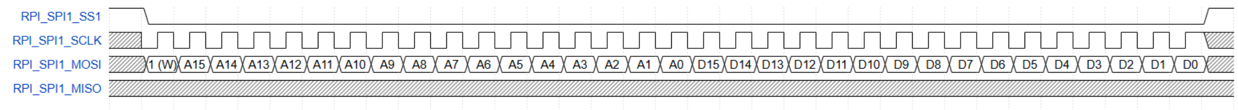

Figure 2: SPI write sequence

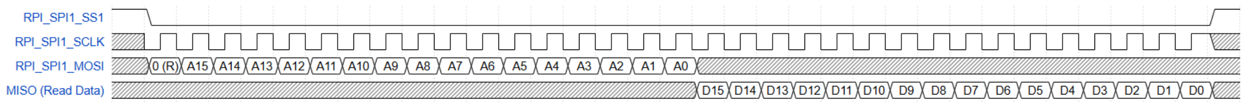

Figure 3: Read sequence

Register description can be found in Table 1. All values should be represented in HEX number format.

Address	Def. value	Bits	Type	Name	Description
					Control
0x0000	0000	15-4		**Reserved**	
		3-2	R/W	**TPULSE_SEL**	00 – GNSS_TPULSE, 01 – RPI_SYNC_OUT, 10 – RPI_SYNC_IN, 11 – Reserved
		1	R/W	**CLK_SEL**	1 - LMK10_CLK, 0 – LMKRF_CLK
		0	R/W	**EN**	1 - Enabled, 0 – Disabled
					Tune settings
0x0001	-	15-0	R/W	**PPS_1S_TARGET_L**	Frequency target value in 1s period (32 bit value, L – lower 16 b, H – upper 16b).
0x0002	-	15-0	R/W	**PPS_1S_TARGET_H**	
0x0003	-	15-0	R/W	**PPS_1S_ERR_TOL**	Error tolerance value in 1s period (16 bit value).
0x0004	-	15-0	R/W	**PPS_10S_TARGET_L**	Frequency target value in 10s period (32 bit value, L – lower 16 b, H – upper 16b).
0x0005	-	15-0	R/W	**PPS_10S_TARGET_H**	
0x0006	-	15-0	R/W	**PPS_10S_ERR_TOL**	Error tolerance value in 10s period (16 bit value).
0x0007	-	15-0	R/W	**PPS_100S_TARGET_L**	Frequency target value in 100s period (32 bit value, L – lower 16 b, H – upper 16b).
0x0008	-	15-0	R/W	**PPS_100S_TARGET_H**	
0x0009	-	15-0	R/W	**PPS_100S_ERR_TOL**	Error tolerance value in 100s period (16 bit value).
					Current tune error values
0x000A	0000	15-0	R	**PPS_1S_ERR_L**	Error count in 1s period (32 bit signed value, L – lower 16 b, H – upper 16b)
0x000B	0000	15-0	R	**PPS_1S_ERR_H**	
0x000C	0000	15-0	R	**PPS_10S_ERR_L**	Error count in 10s period (32 bit signed value, L – lower 16 b, H – upper 16b)
0x000D	0000	15-0	R	**PPS_10S_ERR_H**	
0x000E	0000	15-0	R	**PPS_100S_ERR_L**	Error count in 100s period (32 bit signed value, L – lower 16 b, H – upper 16b)
0x000F	0000	15-0	R	**PPS_100S_ERR_H**	
					Status
0x0010	0000	15-0	R	**DAC_TUNED_VAL**	Tuned DAC value
0x0011	0000	15-9		**Reserved**	
		8	R	**TPULSE_ACTIVE**	0- Timepulse is not active, 1- Timepulse is active
		7-4	R	**ACCURACY**	"0000" - tune disabled or lowest accuracy, 0001 – 1s tune , 0010 – 2s tune , 0011 – 3s tune (highest accuracy).
		3-0	R	**STATE**	"0000" - Coarse Tune, "0001" - Fine tune

Table 1: gpsdocfg registers

2.2 NEO430 CPU

The NEO430 soft-core CPU in its minimal configuration is used to implement the GPSDO control algorithm. More information about the CPU can be found at https://github.com/stnolting/neo430.

The CPU continuously monitors the GPIO pins where the enable signal from gpsdocfg is connected. Depending on the signal's state, the CPU enables or disables the VCTCXO TAMER module via the AVMM (Avalon Memory-Mapped interface). Additionally, it reads frequency error values from the VCTCXO TAMER and performs the necessary adjustments for the VCTCXO DAC. More details about the implemented control algorithm can be found in the following sections.

2.2.1 GPSDO control algorithm

General GPSDO control algorithm is shown in Figure 1. It consist of three main stages:

1. **Initialization** – tune mode is set and VCTCXO TAMEER module is enabled.
2. **Coarse tune** - DAC control values are set to minimum and maximum values and frequency counter errors are captured from 1s period after each tune. At the end of the course tune stage DAC control value is calculated and set by formula described in 2.2.2 Calculating DAC control value (Coarse tune) chapter. At this stage VCTCXO frequency is adjusted using only 1s time interval and should give ± 1Hz precision.
3. **Fine tune** – minor adjustments are made to fine tune VCTCXO frequency and gives better precision close to 0.01Hz. To calculate VCTCXO DAC control value 2.2.3 Calculating DAC control value (Fine tune) should be followed.

VCTCXO DAC adjustement values are calculated using two point line equation formula, more details can be found in following sections.

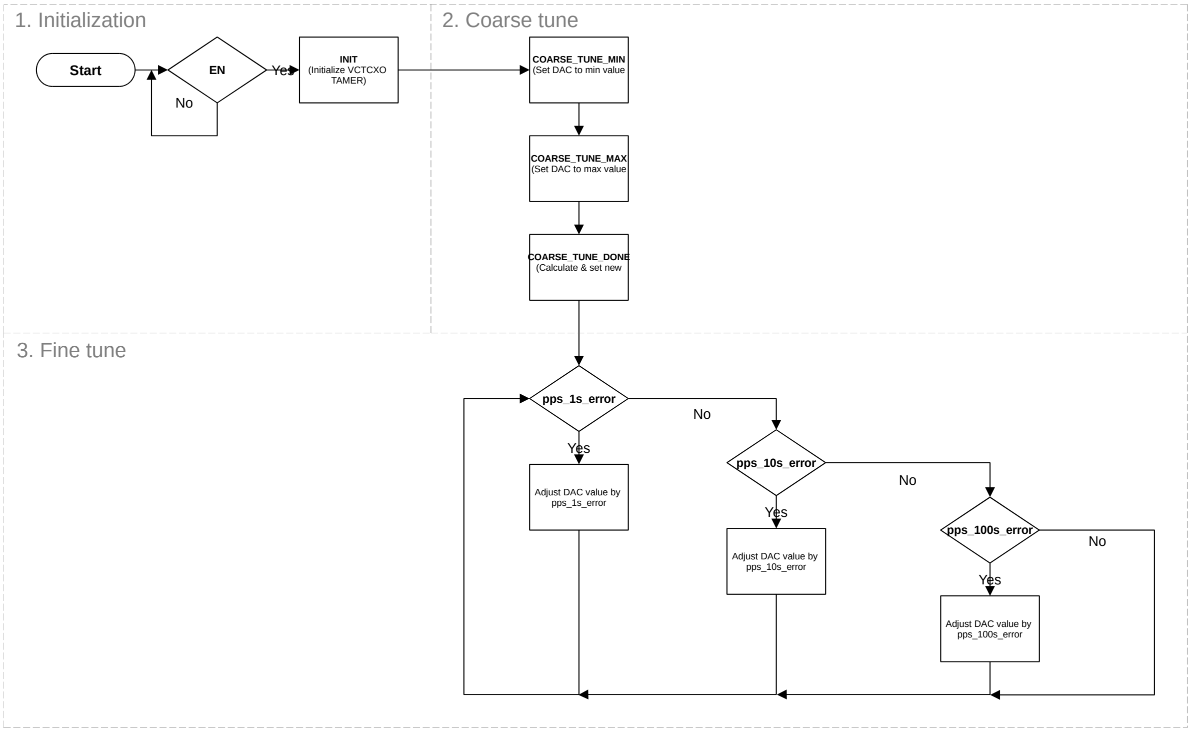

Figure 4: GPSDO control algorithm

2.2.2 Calculating DAC control value (Coarse tune)

DAC control value in coarse tune state is calculated by using two point line equation and finding y-intercept (where line crosses y axis). Graphical representation can be found in Figure 5, where y axis is DAC control value and x axis is frequency counter error.

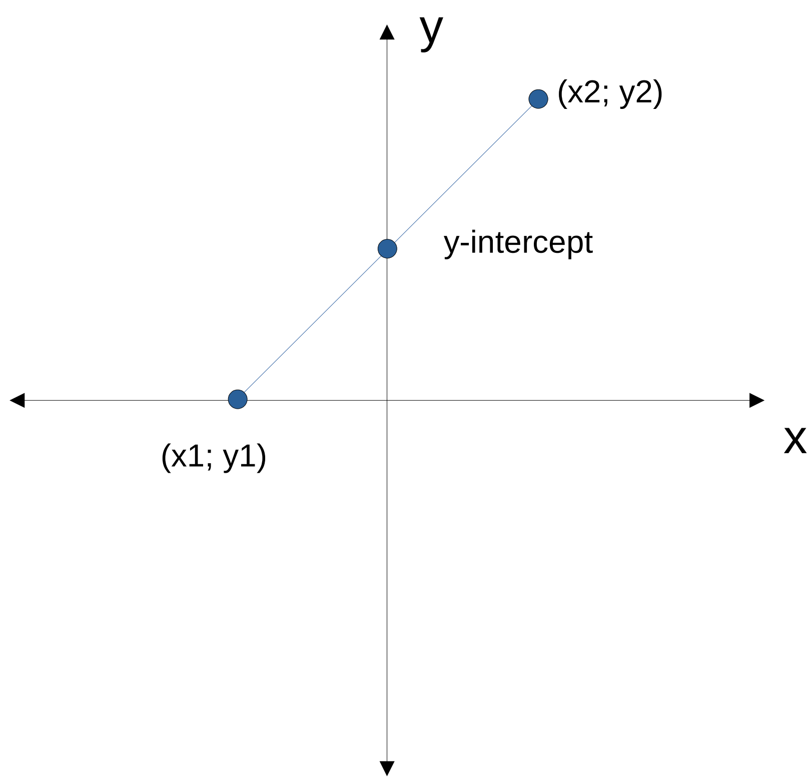

Figure 5: Two point line graphical representation

In order to find DAC control value where frequency counter error would be equal to zero following formulas can be used. First two point line slope is calculated:

$$m = \frac{y2 - y1}{x2 - x1}$$

where *m* – line slope, *x1 y1, x2 y2* – line point coordinates.

Then y-intercept can be calculated:

$$b = y1 - x1 * m$$

where *b* – y intercept, *x1 y1* – line point coordinates and *m* – line slope.

2.2.3 Calculating DAC control value (Fine tune)

VCTCXO DAC value set in fine tune stage is being adjusted by checking all three frequency counter intervals at this stage. Check the magnitude of the errors starting with the one second count. If an error is greater than the maximum tolerated error, adjust the trim DAC by the error multiplied by the slope and scale the result by the precision interval (e.g. 1s, 10s, 100s):

$$new\,value = current\,value - \frac{error * m}{scale}$$

where $error$ - frequency counter error greater than maximum tolerated error, m – line slope calculated in previous stage , $scale$ - 1, 10 or 100 depending which frequency counter exceeded the set tolerance.

2.3 VCTCXO TAMER module

The VCTCXO TAMER implementation on an FPGA uses three 32-bit counters to monitor and calculate the frequency error of a reference clock over three time periods: 1 second, 10 seconds, and 100 seconds.

Key Features:

- **Reference Time Period (1PPS):** A 1 Pulse Per Second (1PPS) signal serves as the reference for measuring the time periods, ensuring accurate error calculation.
- **32-bit Counters:** Three independent counters track the clock signal over the respective time periods (1s, 10s, 100s).
- **Frequency Error Calculation:** The module computes the frequency error by comparing the measured clock counts to the expected values for each time period.
- **Error Interrupt:** If the calculated error exceeds a user-configured tolerance threshold, the module generates an interrupt signal for the CPU, enabling corrective action or reporting.
- **AVMM Interface:** The module features an Avalon Memory-Mapped (AVMM) interface, which allows the CPU to Read the computed error values for diagnostics or adjustments.

Simplified functional diagram can be found in Figure 3.

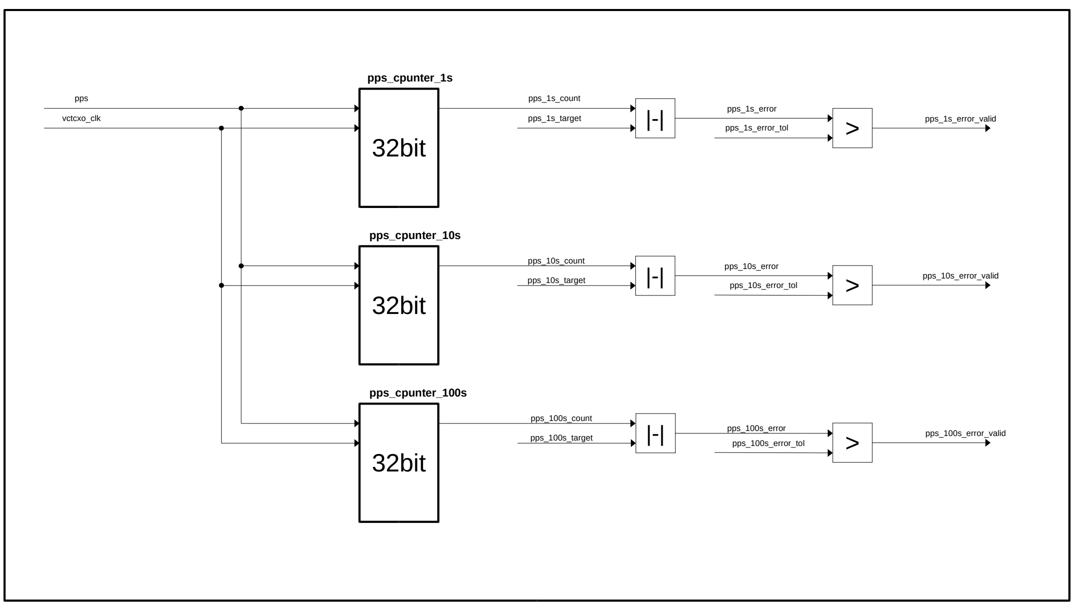

Figure 6: Simplified functional diagram of VCTCXO TAMER

3 Getting started with GPSDO

This chapter provides a short guide on how to use the **GPSDO** implementation in LimePSB-RPCM v1.3 board. It covers the necessary steps to configure and operate the module.

1. Connect GPS antenna to GPS/GNSS(J44) connector
2. Configure PPS_TARGET and PPS_ERR_TOL registers in gpsdocfg module (see register description in Table 1). Values can be calculated:

   $$PPS\_TARGET = f * s;$$

   $$PPS\_ERROR\_TOL = round(s * f * ppb * 1e\text{-}9)$$

   where *s* – time in seconds, *f* – frequency in Hz, ppb – required stability in points per billion

   Example register values calculated for 30.72MHz VCTCXO clock and 20ppb error tolerance:

   ```
   0x0001 = 0xC000
   0x0002 = 0x01D4
   0x0003 = 0x0001
   0x0004 = 0x8000
   0x0005 = 0x124F
   0x0006 = 0x0006
   0x0007 = 0x0000
   0x0008 = 0xB71B
   0x0009 = 0x003D
   ```

3. Enable VCTCXO tamer module by setting `0x0000[0]` register bit to 1.
4. LED7 should start blinking in red color once per second, this indicates that there is active 1PPS signal and VCTCXO TAMER module is enabled.
5. Observe VCTCXO TAMER module tune state and accuracy register values of address `0x0011`, after few minutes it should become `0x0031` (fine tune state and 3s tune highest accuracy, see Table 1 for register description).