AN520: AS500X DCXO reference manual

1. Introduction

This document is an application note describing detail configuration and usage of As5000 family DCXO (Digitally Controlled XO) feature. It is a complement to the I²C controlled As5003 device datasheet.

2. Table of Contents

3. I²C Control

The DCXO feature is fully configurable and controlled using I²C interface. The DCXO feature must be configured and enabled by the I²C master before use. It is also possible to factory configure the DCXO featured such that the device will come up with DCXO full configured and enabled.

The As5000 family I²C interface description is duplicated here from As5003 datasheet for convenience.

3.1. I²C Interface

As5000 I²C interface is fully electrically and timing compatible with the “UM10204 I²C-bus specification and user manual, Rev. 6 — 4 April 2014” with the exceptions described in Table 3.1.

Table 3.1 I²C Compatibility

I2C	Speed	Compliance	Notes
Standard	100 kHz	Compliant	N/A
Fast	400 kHz	Compatible	The following I2C Fast requirements are not met: 1. There is no SDA falling edge control, it can be faster than 20 ns required. I2C SDA driver has typical ~50 Ω output impedance. 2. If the device loses power, the connected I2C bus SDA and SCL signals will be forced to ground rather than floating as required by the standard.
Fast+	1 MHz	Compatible	SDA pad is not I2C Fast+ electrically compliant: Same issues 1. and 2. as described in Fast mode above apply here. Fast+ standard requires 20 mA pull down ability, device can only pull 6 mA. If the bus is not heavily loaded then the weaker driver generates correct Fast+ timing on the bus and the device would work at 1 MHz.

3.2. I²C Register Write and Read Protocol

As5003 implements 8 bit I²C addressable address space with 256 addressable byte locations. Not all locations are used and some are reserved for factory use. The user should not attempt to write to the register addresses not specifically described in the As5003 datasheet.

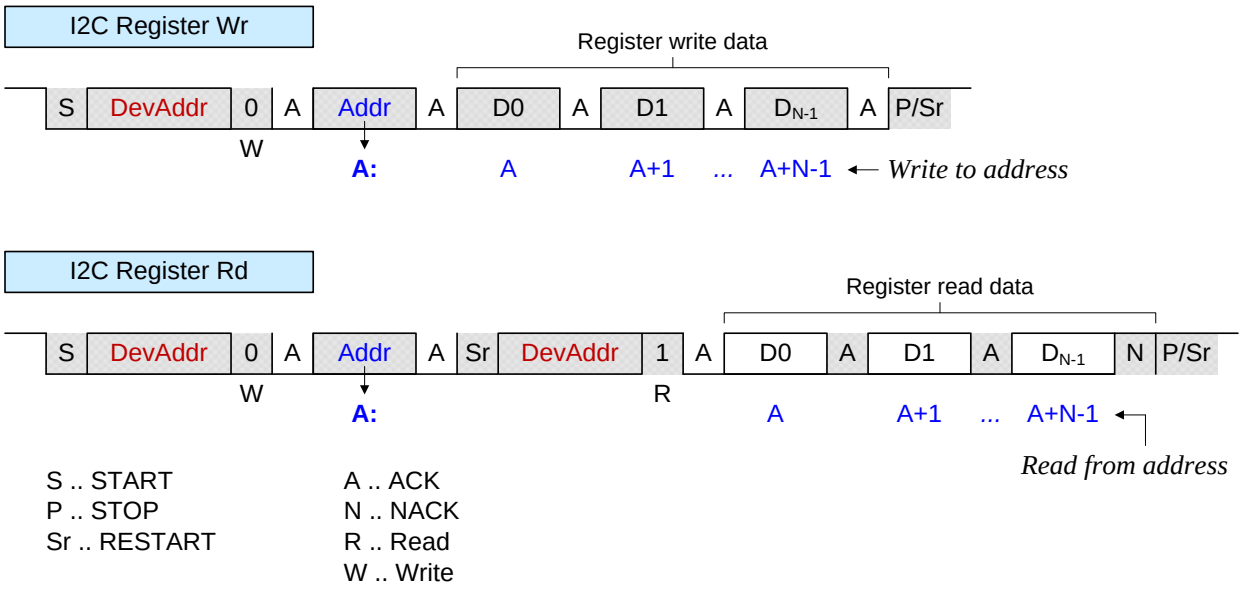

Figure 3.1 I²C write and read transactions

DevAddr is 7 bit device address as the device appears on the I²C bus set in the factory per user specification. Allowed values are in the range **16** to **119**, inclusive. Both write and read register transactions with register address autoincrement enabled are shown in Figure 3.1. Write register transaction is an I²C write transaction for which data byte stream must start with the 8 bit register address followed by one or more register data bytes. Read register sequence starts with I²C write transaction to set the read register address followed by the I²C read transaction to read one or more data bytes.

The register address autoincrement is enabled by default after power up. When the maximum address 0xFF of the I²C register space is reached, it saturates at 0xFF and is not incremented further. The I²C register address autoincrement therefore does not wrap around to 0x00 address.

The register address autoincrement can be disabled by writing register **bI2C_INC_DIS**=1 and re-enabled again by writing **bI2C_INC_DIS**=0. When the register address autoincrement is disabled all bytes in the I²C transactions are written to or read from the same address. Having the register autoincrement disabled is required for DCXO streaming mode.

Data and address bytes appear on the I²C SDA line with most significant bit (MSB) first in time per I²C standard. During I²C transactions SCL clock line is never stalled by the device.

4. I²C DCXO Control Registers

This section is a DCXO feature related excerpt from the overall I²C register description in As5003 datasheet.

The I²C interface is a byte oriented interface. Registers wider than 8 bits are required to be split into multiple bytes located on subsequent register addresses.

Signed 32 bit long integer, prefix **j**, is organized in big endian fashion, such that the most significant byte (MSB) is located at the lowest I²C register address.

Single unsigned byte registers have prefix **b**.

The field access **Type** has the following values:

R/W .. read/write field

R .. read only field

W1 .. writing 1 triggers an associated event, writing 0 has no effect, read always returns 0.

The **Endian** column indicates endian for registers wider than one byte.

B .. big endian. The most significant register byte (MSB) appears at the lowest address.

L .. little endian. The least significant register byte (LSB) appears at the lowest address.

The **Rst** column indicates the field reset value after powerup.

Table 4.1 DCXO control register summary and field list

Register	Addr	Bytes	Endian	Field	Bits
bI2C_INC_DIS	0x06	1	--	i2c_inc_dis	[0]
bDCXO_SYNC	0x1a	1	--	dcxo_stream_sync	[0]
				dcxo_clr_sec	[1]
				dcxo_sat_neg	[6]
				dcxo_sat_pos	[7]
bDCXO_SHIFT	0x1b	1	--	dcxo_shift	[4:0]
bDCXO_CTRL	0x1c	1	--	dcxo_size	[2:0]
				dcxo_stream_mode	[4]
				dcxo_rel_mode	[5]
				dcxo_ena	[6]
				dcxo_clr	[7]
jDCXO_DATA	0x1d	4	B	dcxo_data	[31:0]
bVC_LPF_BW_DIR	0x41	1	--	vc_lpf_bw_dir	[2:0]
bVC_LPF_ABS_SAT	0x42	1	--	vc_lpf_abs_sat	[7:0]
bVC_VCTRL	0x43	1	--	vc_vctrl_dis	[0]

DCXO register and field details are in the following tables.

Table 4.2 DCXO control

Register	Addr	Field	Bits	Type	Rst	Description
bDCXO_SYNC	0x1a	dcxo_stream_sync	[0]	W1	0	Writing 1 to this bit clears the streaming mode synchronization logic. The next write to the bDCXO_DATAL register byte will be treated as the MSB byte of the burst. This bit allows to forcibly synchronize the streaming operation when in doubt. Note that any write to bDCXO_CTRL register also clears the streaming mode synchronization logic.
		dcxo_clr_sec	[1]	W1	0	Secondary duplicated clear pulse bit for convenience, same function as bDCXO_CTRL.dcxo_clr bit. See the bDCXO_CTRL description. The clear affects only the DCXO data processing and not the DCXO I2C control registers. All the current DCXO settings as set in bDCXO_CTRL and bDCXO_SHIFT registers will stay intact.
		dcxo_sat_neg	[6]	R	0	If 1 then DCXO frequency deviation internal value has saturated to minimum negative value. Register bit updated every time the 32 bit DCXO internal value is updated. The bit is not sticky. Cleared by bDCXO_SYNC.dcxo_clr_sec=1 or bDCXO_CTRL.dcxo_clr =1.
		dcxo_sat_pos	[7]	R	0	If 1 then DCXO frequency deviation internal value has saturated to maximum positive value. Register bit updated every time the 32 bit DCXO internal value is updated. The bit is not sticky. Cleared by bDCXO_SYNC.dcxo_clr_sec=1 or bDCXO_CTRL.dcxo_clr =1.
bDCXO_SHIFT	0x1b	dcxo_shift	[4:0]	R/W	0	Number of bits to shift the input jDCXO_DATA value to the left before internal processing. The left bit shift is always applied at the input DCXO value in all modes and configurations. The shift value is in the range of <0,

Register	Addr	Field	Bits	Type	Rst	Description
						24> bits. If the shift number is greater than 24 then it is internally forced to 24.
bDCXO_CTRL	0x1c	dcxo_size	[2:0]	R/W	0	Number of streamed bytes per input value or number of bottom jDCXO_DATA bytes to be used as input value in direct register access mode. The valid values are 0, 1, 2, 3, and 4 bytes. Default value 0 is an alias for value 4. The value greater than 4 is internally forced to 4. All values are internally sign extended to 57 bits before the value is used. The bDCXO_SHIFT left shift is applied on the internally sign extended 57 bit number. For streamed mode the number denotes the size of ordered byte burst in the I2C transaction: 4 .. {H|M|N|L} .. 32 bits, the last L byte is always LSB byte 3 .. {M|N|L} .. 24 bits 2 .. {N|L} .. 16 bits 1 .. {L} .. 8 bits For direct register mode this number denotes the number of LSB bytes from the jDCXO_DATA used to form a number to be then internally sign extended to 57 bits. The number of bytes and their order is the same as in the streaming table above. The {H} denotes the MSB byte at byte address jDCXO_DATA while the {L} denotes the LSB byte at the byte address jDCXO_DATA + 3. When used as 4 bytes the jDCXO_DATA is a 32 bit signed integer stored in a big endian fashion, MSB at lower address. The bDCXO_DATAL byte at byte address jDCXO_DATA + 3 is always the LSB byte for all widths, and could be viewed as the value alignment anchor.

Register	Addr	Field	Bits	Type	Rst	Description
		dcxo_stream_mode	[4]	R/W	0	DCXO frequency offset value input mode. 0 .. regular direct register mode. For situations when I2C address autoincrement is active, which is a default mode. The jDCXO_DATA four byte array can be written any time. Whenever the bDCXO_DATAL location at the jDCXO_DATA + 3 byte address is written the number of bytes selected by bDCXO_CTRL.dcxo_size number is taken from the bottom of the jDCXO_DATA value, aligned towards bDCXO_DATAL location as LSB byte. Those selected bytes are sign extended to 57 bits and left shifted by the bDCXO_SHIFT value. The resulting number is then either used to add to the existing internal DCXO value in 57 bit addition and the sum is then sign saturated to bottom 32 bits to be used as a new DCXO value, or the value is sign saturated to 32 bits to be used directly as the new DCXO value. 1 .. streaming mode. For situations when I2C register address autoincrement is disabled. User must write a register bI2C_INC_DIS=1 to actively disable the I2C register address autoincrement. The setting is global and applies to all I2C register accesses, not only to the DCXO data registers. The user must choose to stream 4 {H|M|N|L}, 3 {M|N|L}, 2 {N|L}, or 1 {L} byte by setting the bDCXO_SIZE register before starting streaming operation. The stream byte order is from left to right and the LSB will always come last in the value burst.

Register	Addr	Field	Bits	Type	Rst	Description
						All data is written to bDCXO_DATAL register only at address jDCXO_DATA + 3. Writes to other jDCXO_DATA register bytes are ignored. The streamed byte value is then treated the same way as in the regular direct register mode described above. The streaming mode is useful for very fast DCXO value updates, possibly in single, very long, I2C transaction.
		dcxo_rel_mode	[5]	R/W	0	DCXO frequency offset value usage mode. 0 .. absolute mode. The user DCXO signed input value obtained using the current access mode is sign extended to 57 bits and then left shifted by bDCXO_SHIFT bits. The shifted signed value is sign saturated to bottom 32 bits and then used as the internal DCXO frequency offset value. 1 .. relative mode. The user DCXO signed input value obtained using the current access mode is sign extended to 57 bits and then left shifted by bDCXO_SHIFT bits. The shifted signed value is added to the current internal 32 bit DCXO value resulting in 57 bit internal signed addition number. The 57 bit signed sum is sign saturated to bottom 32 bits and then used as the internal DCXO frequency offset value. If the bottom 32 bit saturation happened the read only saturation flags bDCXO_SYNC.dcxo_sat_pos and bDCXO_SYNC.dcxo_sat_neg will reflect the saturation operation of the last update of the internal DCXO frequency offset value.
		dcxo_ena	[6]	R/W	0	Enable the DCXO operation.

Register	Addr	Field	Bits	Type	Rst	Description
						If this bit is 1 then the DCXO operation is enabled. If this bit is 0 then the DXCO operation is disabled. All control bits in bDCXO_CTRL register are ignored with the exception of bDCXO_CTRL.dcxo_clr, which can be written as 1 to clear DCXO processing module any time.
		dcxo_clr	[7]	W1	0	Write 1 to this bit clears all the DCXO internal frequency offset value registers, all DCXO value holding registers, and streaming mode synchronization logic. All mentioned registers are cleared even if bDCXO_CTRL.dcxo_ena=0. The clear does not affect bDCXO_SHIFT register. Note that the streaming mode synchronization logic is cleared every time the bDCXO_CTRL register is written, even if the bDCXO_CTRL.dcxo_ena=0 or bDCXO_CTRL.dcxo_clr=0. It is highly recommended that this bit is written as 1 whenever the DCXO is being enabled by setting bDCXO_CTRL.dcxo_ena=1 to clear the internal processing registers.
jDCXO_DATA	0x1d	dcxo_data	[31:0]	R/W	0	DCXO data register through which the actual 32 bit signed DCXO control data is applied. Increasing value will increase the output frequency. The 32 bit input value represents 11.21s signed 2's complement number in [0.9536743164 ppm] units as a relative deviation from the current central frequency. The 4 register bytes can be viewed separately as bDCXO_DATAH,

Register	Addr	Field	Bits	Type	Rst	Description
						bDCXO_DATAM, bDCXO_DATAN, and bDCXO_DATAL data bytes and writing to them depends on the write mode selected. bDCXO_DATAH .. jDCXO_DATA address + 0 bDCXO_DATAM .. jDCXO_DATA address + 1 bDCXO_DATAN .. jDCXO_DATA address + 2 bDCXO_DATAL .. jDCXO_DATA address + 3 .. always LSB The bDCXO_DATAL LSB byte refers to the jDCXO_DATA[7:0] bits, which is the byte at the highest address (jDCXO_DATA address + 3) since the register value is represented in big endian, MSB at lower address. There are holding registers for 3 MSB upper value bytes and the written value will get applied internally only if the bDCXO_DATAL byte is written. In regular direct register access mode, bDCXO_CTRL.dcxo_stream_mode=0, any byte in the four byte jDCXO_DATA register can be written at any time. The most convenient way is to write it is as a burst of bytes with incrementing byte address after each byte write. The written value is applied internally only after the last, bDCXO_DATAL, byte is written. The user DCXO input value is formed as a signed selection of bottom N bytes (N=DCXO_CTRL.dcxo_size) where bDCXO_DATAL is always the LSB byte of the signed number. In streaming access mode, bDCXO_CTRL.dcxo_stream_mode =1, the bDCXO_DATAL byte is used to stream the data. Writing to the rest of the bytes in the jDCXO_DATA register is ignored in streaming mode. The user

Register	Addr	Field	Bits	Type	Rst	Description
						could decide to stream 4 {H|M|N|L}, 3 {M|N|L}, 2 {N|L}, or 1 {L} byte, all in big endian fashion. After the last byte of the value burst is written, the signed numerical value represented by 4, 3, 2, or 1 bytes based on the streaming number configuration form the input user DCXO value. Reading from jDCXO_DATA register always returns current signed internal 32 bit DCXO value as applied to the input of the DCXO frequency control low pass filter in [0.9536743164 ppm] units as a relative deviation from current central frequency. The read value does not depend on any write mode configuration and is always the current internal DCXO frequency offset value. This also applies in the streaming mode. The streaming mode applies only for value writes. To read the current 32 bit internal value even in streaming mode the user needs to read all 4 bytes of this register to get the 32 bit current internal value. See Figure 5.1.

Table 4.3 DCXO low pass filter configuration

Register	Addr	Field	Bits	Type	Rst	Description
bVC_LPF_BW_DIR	0x41	vc_lpf_bw_dir	[2:0]	R/W	7	Frequency offset low pass filter bandwidth control. The LPF bandwidth formula: BW = 1166 * 2vc_lpf_bw_dir [Hz] which corresponds to the following values: 0 .. 1.2 kHz 1 .. 2.3 kHz 2 .. 4.7 kHz 3 .. 9.3 kHz 4 .. 18.7 kHz 5 .. 37.3 kHz 6 .. 74.6 kHz 7 .. pass through .. default value

Register	Addr	Field	Bits	Type	Rst	Description
bVC_LPF_ABS_SAT	0x42	vc_lpf_abs_sat	[7:0]	R/W	0	Frequency offset unsigned absolute saturation value to user limit the maximum frequency offset swing. The unsigned 8 bit value is shifted left by 13 bits to form 10.11u unsigned absolute saturation value of the 11.11s output. The 22 bit signed output will be saturated to fall in the value range <-vc_lpf_abs_sat, + vc_lpf_abs_sat > * 213. See Figure 5.1. The value 0 squelches the frequency offset and forces central frequency while value 255 allows full frequency offset range to be utilized.
bVC_VCTRL	0x43	vc_vctrl_dis	[0]	R/W	0	Disable the DCXO frequency control output and force the value to 0. This feature is to provide temporary disable for DCXO datapath to see how the device is doing while keeping the DCXO setting unchanged and datapath active. See Figure 5.1.

5. DCXO Control Description

The device implements I²C controlled Digitally Controlled XO (DCXO) feature to allow seamless and fast setting of the frequency deviation from the center frequency.

The device implements 32 bit signed **jDCXO_DATA** register in **11.21s** signed bit format which represents frequency offset from the current center frequency in **fUSER_FREQ** register in **[0.9536743164 ppm]** units where [ppm] is a frequency step equal to 10^{-6} · **fUSER_FREQ** in [Hz]. The scaling factor is **0.9536743164** = $10^6/2^{20}$.

It is possible to move the output frequency relative to the center frequency within the range **±975 ppm**.

The DCXO feature must be configured before use by writing **bDCXO_CTRL** and **bDCXO_SHIFT** registers with additional configuration provided by **bVC_LPF_BW_DIR** and **bVC_LPF_ABS_SAT** registers.

Whenever the new frequency deviation register **jDCXO_DATA** is written the output signal changes to a new frequency seamlessly and glitchlessly both in time and frequency without output signal interruption. The settlement time from the end of last data bit of the byte written by I²C to bDCXO_DATAL register to a new output frequency is less than **8 us**.

The DCXO data path configuration and control is schematically shown in Figure 5.1. Numerical values are shown in **NN.MMs** notation where NN is the number of MSB bits representing integer part of the number while MM represents number of LSB bits representing fractional part of the number. The letter '**s**' means signed, the letter '**u**' unsigned value. Signed values are represented in binary 2's complement. The values are labeled in **[ppm]** in Figure 5.1 for simplicity, but the numerical binary values must be multiplied by the **0.9536743164** factor to represent the real **[ppm]** frequency deviation value from the current center frequency.

I²C write to 32 bit **jDCXO_DATA** register normally requires I²C transaction of 6 bytes. Every update of the DCXO value would take 6 byte write on I²C bus, which takes ~56 us if using I²C Fast+ 1 MHz speed and ~140 us if using I²C Fast 400 kHz speed. In some application faster update speed may be required while the full 32 bit frequency offset resolution is not needed. To accommodate faster DCXO offset updates using I²C the DCXO has the following configurable features:

Data size: bDCXO_CTRL.dcxo_size={0|1|2|3|4}
Select to write only 1, 2, 3, or all 4 bytes signed numbers, 0 means 4.

Left shift: bDCXO_SHIFT=<0, 24> range

Configurable shift of the entered input value to the left up to 24 bits for scaling small numbers to proper position within 32 bit number.

- **Data input:**

Direct register mode: bDCXO_CTRL.dcxo_stream_mode=0
Write the input value to the **jDCXO_DATA** bytes directly by I²C.

Streaming mode: bDCXO_CTRL.dcxo_stream_mode=1
Use **only LSB** byte of **jDCXO_DATA** at I²C byte address **0x20** as a streaming window. It is required that I²C register address autoincrement is disabled by writing **bI2C_INC_DIS**=1 to allow DCXO frequency deviation updates in one very long I²C transaction feeding only data bytes to the same register address.

- **Data application:**

Absolute mode: bDCXO_CTRL.dcxo_rel_mode=0
Apply the input value directly as a frequency deviation. The input jDCXO_DATA value gets interpreted as a frequency offset from fUSER_FREQ frequency.

Relative mode: bDCXO_CTRL.dcxo_rel_mode=1
Apply the input value as a relative change from the current DCXO frequency deviation in the internal 32 bit register, different from the **jDCXO_DATA** input register. This allows applying only incremental steps to change the internal frequency deviation register. The input **jDCXO_DATA** value gets interpreted as an incremental step to be added to the current internal register frequency deviation from **fUSER_FREQ**.

DCXO register description in Figure 5.1, Table 4.2, and Table 4.3Table 4.3 provide more details.

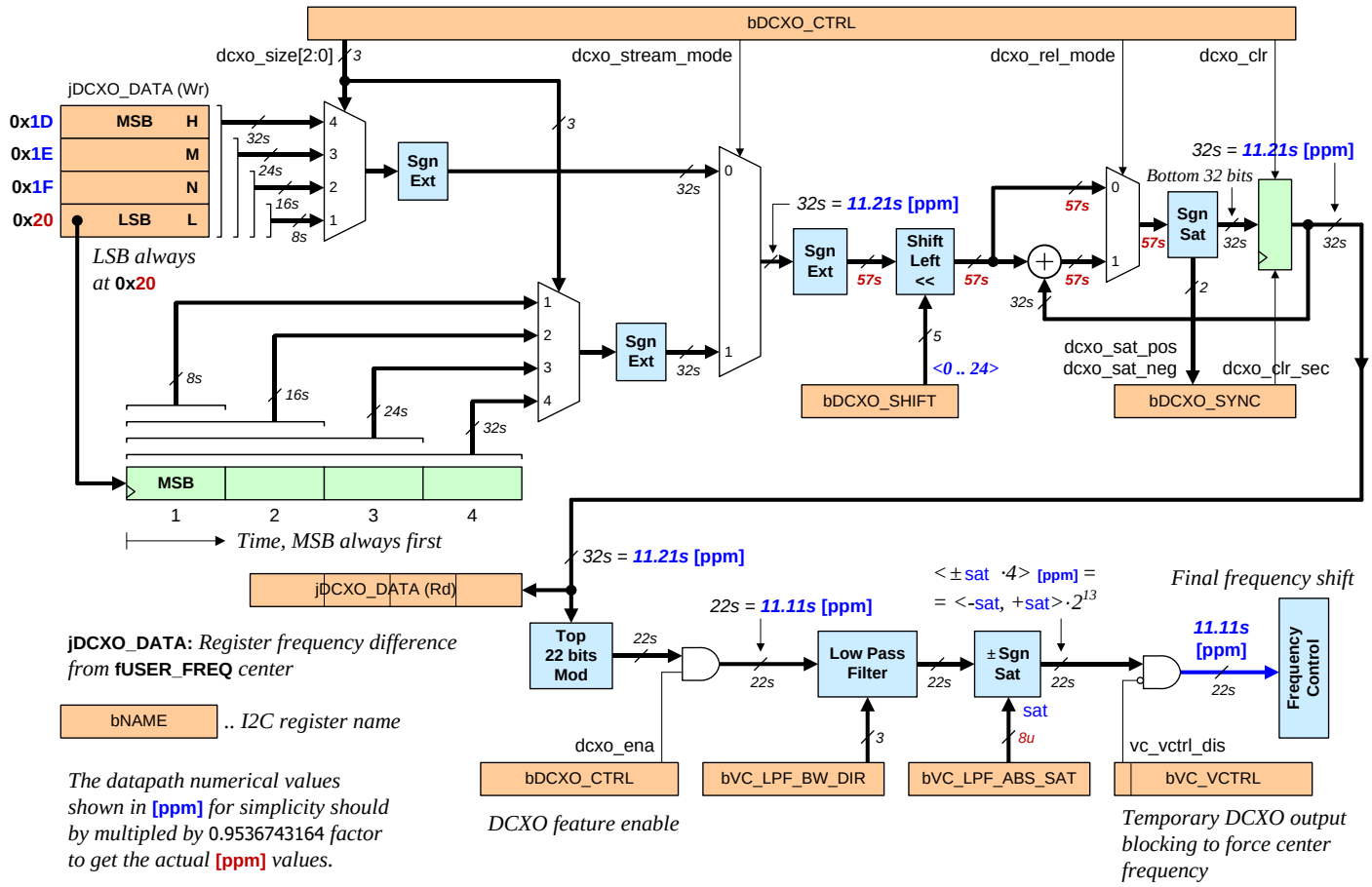

Figure 5.1 DCXO control and data processing

6. DCXO Control Configuration

Even though the section above provides complete set of details how to configure and use the DCXO feature, it could be daunting to determine all the configuration details. This section provides a guide to suggested DCXO configuration steps and examples of DCXO use.

Notation:

Dppm .. desired frequency deviation in parts per million [ppm] (10^{-6})

Dppb .. desired frequency deviation in part per billion [ppb] (10^{-9})

DmaxPpm .. maximum one sided desired frequency deviation in [ppm]

DmaxPpb .. maximum one sided desired frequency deviation in [ppb]

LsbPpm .. desired **jDCXO_DATA** deviation step when LSB bit changes in [ppm]

LsbPpb .. desired **jDCXO_DATA** deviation step when LSB bit changes in [ppb]

LsbActPpm .. actual **jDCXO_DATA** deviation step when LSB bit changes in [ppm]

LsbActPpb .. actual **jDCXO_DATA** deviation step when LSB bit changes in [ppb]

DsatPpm .. desired one sided saturated frequency deviation [ppm]

DsatActPpm .. actual one sided saturated frequency deviation [ppm]

The [**ppb**] denotes parts per billion and it is a frequency step equal to 10^{-9} · **fUSER_FREQ** in [Hz]

The DCXO feature configuration can be split into two groups:

1. Common configuration for all modes (**bVC_LPF_BW_DIR**, **bVC_LPF_ABS_SAT**)
2. Configuration of DCXO data size, data shift, data input (**Direct**, **Streaming**), and data application (**Absolute**, **Relative**)

6.1. Common Configuration

The user is required to configure the saturation level and the low pass filter bandwidth no matter what the DCXO deviation update strategy is used. Those are controlled by the following registers:

bVC_LPF_BW_DIR

bVC_LPF_ABS_SAT

It is required to configure the low pass filter by setting up **bVC_LPF_BW_DIR** register. The reset value **0** corresponds to the 1.2 kHz bandwidth. The value **7** means pass through and therefore no filtering. See the register table other bandwidth values.

It is required to configure the saturation register **bVC_LPF_ABS_SAT** register. The reset value of **bVC_LPF_ABS_SAT** is **0**, which means that the output of internal DCXO datapath is blocked. It is required to set the register to proper value. The maximum value of **255** means no saturation and full **±975 ppm** frequency deviation swing can be utilized.

If there is a desire to limit the maximum deviation swing by hardware, the saturation hardware can be configured. Note that the saturator follows the internal frequency deviation register which saturates at the maximum swing **±975** ppm. The output saturator just selects the symmetrical band of values from that register. This can have consequences if saturating when in **Relative** mode.

To configure the saturation register based on the user desire to limit the frequency deviation swing to **±DsatPpm**, configure the register:

bVC_LPF_ABS_SAT = min(255, floor(**DsatPpm** * $2^{18}/10^6$))

If there is a desire to go to the next bigger value to always include the desired saturation level and add most additional 4 ppm, use **ceiling** function instead of a **floor** function in the equation above. The **min** function picks the minimum out of the two values in the parenthesis.

To calculate the actual maximum saturated swing **±DsatActPpm** in [ppm] use the following:

DsatActPpm = **bVC_LPF_ABS_SAT** * $10^6/2^{18}$

6.2. Simple DCXO Use

Let us consider the simplest case when the speed of the DCXO frequency deviation update is not critical. The situation can be described as follows:

1. DCXO deviation update is not time critical and is much slower than I²C speed.
2. All 4 bytes of **jDCXO_DATA** register will be written to the device in **Direct** register mode for each deviation update.
3. **Absolute** mode is used, which means that new I²C transaction is used for each new deviation update, resulting in 6 bytes traffic on I²C bus which is about ~56 I²C SCL clock cycles.

The explanation of this scenario will serve as a foundation for the more generic DCXO setup described in the next section.

If frequency deviation update speed in not critical then the **Direct** register mode, writing the full **jDCXO_DATA** register in **11.21s** format, unshifted with **bDCXO_SHIFT**=0, is the simplest, but also slowest, mode of operation. All other modes are there to speed up the DCXO frequency deviation update rate.

For this mode of operation the user needs to configure:

bDCXO_SHIFT = 0 .. no jDCXO_DATA shift

bDCXO_CTRL.dcxo_stream_mode = 0 .. **Direct** register mode

bDCXO_CTRL.dcxo_rel_mode = 0 .. **Absolute** data mode

bDCXO_CTRL.dcxo_size = 0 (or 4) .. all 4 bytes of jDCXO_DATA are used

With this configuration the jDCXO_DATA represents the frequency deviation value in a fractional **11.21s** signed bit width format in [**0.9536743164 ppm**] or [**953.6743164 ppb**] units. To calculate what the frequency deviation single LSB bit change of **jDCXO_DATA** represents, the values above needs to be divided by 2^{21}. The minimum frequency deviation achieved by changing single LSB bit of **jDCXO_DATA** in this configuration is:

953.6743164 ppb / 2^{21} = **0.00045475 ppb**

This is the minimum theoretically achievable frequency step by changing the LSB by **jDCXO_DATA**.

To calculate the **jDCXO_DATA** for a given desired signed frequency deviation **Dppm** in [ppm] or in **Dppb** in [ppm] use the following formula:

jDCXO_DATA = round(**Dppm** * $2^{41}/10^6$) = round(**Dppb** * $2^{41}/10^9$)

The scaling factor $2^{41}/10^6$ = **2199023.255552** exactly.

The calculation of the **jDCXO_DATA** must be done such that the result is 32 bit signed integer with negative numbers represented in 2's complement.

6.3. Generic DCXO Use

With **bDCXO_SHIFT**=0 above it is required to calculate the **jDCXO_DATA** such that the LSB of the **jDCXO_DATA** number represents a frequency shift of 0.00045475 ppb. In the real application two scenarios might happen:

1. The master computer calculates the frequency deviation in specific quantization step, say 1 ppb, and there is a desire to use that number directly without recalculation.
2. There is a desire of faster deviation update by limiting number of bytes used to update **jDCXO_DATA** register at the expense of the resolution of the minimum frequency deviation step.

When configuring the DCXO for generic use the user needs to determine the following:

1. Determine the output saturation range **bVC_LPF_BW_DIR** described above.
2. Determine the low pass filter bandwidth **bVC_LPF_ABS_SAT** described above.
3. Determine the **Data application** mode: **Absolute** or **Relative**. For the **Relative** mode the data calculation below refer to the maximum relative step from the current deviation. **Relative** mode could reduce the duration of the deviation update even further since single **jDCXO_DATA** might suffice in some applications.
4. Determine the maximum desired frequency deviation **±DmaxPpm** or **±DmaxPpb**. Those values must not exceed the maximum allows **±975** ppm.
5. Determine the frequency deviation desired step **LsbPpm** or **LsbPpb** when changing LSB bit of **jDCXO_DATA**.
6. From **±DmaxPpm** and **LsbPpm** or their [**ppb**] equivalents determine the value to be set in **bDCXO_SHIFT** register. This also determines the minimum number of **jDCXO_DATA** bits/bytes to be used to achieve the desired maximum

frequency deviation and frequency deviation step. If the maximum number of desired **jDCXO_DATA** bytes to be used is specified this step could be an iterative process to determine the **bDCXO_SHIFT** register value and **LsbPpm** or **±DmaxPpm** , depending on the situation.

7. Determine the **Data input mode**: **Direct** or **Streaming**.

- Generic formula for **bDCXO_SHIFT** calculation:

bDCXO_SHIFT = floor($\log_2$(**LsbPpm** * $2^{41}/10^6$)) = floor($\log_2$(**LsbPpb** * $2^{41}/10^9$))

Instead of the **floor** function it is possible to use **round** to get to the closest step or **ceil** to have at least the step desired. The check is required to make sure that calculated result obeys the following limitation:

bDCXO_SHIFT ≤ 24

Due to the conversion to an integer using **floor**/**round**/**ceil** function the desired **LsbPpm** or **LsbPpb** step might not be matched exactly. To calculate the exact frequency deviation step when LSB bit of **jDCXO_DATA** changes for a given the **bDCXO_SHIFT** value use the following:

LsbActPpb = $10^9 / 2^{(41\text{-}\mathbf{bDCXO\_SHIFT})}$

LsbActPpm = $10^6 / 2^{(41\text{-}\mathbf{bDCXO\_SHIFT})}$

A general purpose formula for required minimum number of bottom input bits $\mathbf{N_{bits}}$ or $\mathbf{M_{bytes}}$ to be written to **jDCXO_DATA** for one sided maximum desired frequency deviation **DmaxPpm** in [ppm] to achieve **±DmaxPpm** deviation at the output from central frequency:

$\mathbf{N_{bits}}$ = ceil($\log_2$(**DmaxPpm** * $2^{(41\text{-}\mathbf{bDCXO\_SHIFT})}/10^6$)) + 1

$\mathbf{M_{bytes}}$ = ceil($\mathbf{N_{bits}}$ / 8)

The $\mathbf{N_{bits}}$ wide number is a signed number which includes the sign. To following formula reverses the calculation and calculates the **DmaxPpm** from the specified $\mathbf{N_{bits}}$ used to write to **jDCXO_DATA**:

DmaxPpm = $10^6 / 2^{(42\text{-}\mathbf{bDCXO\_SHIFT}\text{-}\mathbf{Nbits})}$

Obviously, the usable frequency deviation maximum is **DmaxPpm** ≤ 975 ppm. To complete the formula set, the following calculates the **bDCXO_SHIFT** from the user specified **DmaxPpm** and the desired $\mathbf{N_{bits}}$:

bDCXO_SHIFT = 42 - $\mathbf{N_{bits}}$ - floor($\log_2$(10^6/**DmaxPpm**))

Instead of the **floor** function it is possible to use **round** or **ceil** functions as in the case of calculating **bDCXO_SHIFT** from **LsbPpm** input above. Again, the check is required to make sure that calculated results obeys the following limitation:

bDCXO_SHIFT ≤ 24

To calculate the **jDCXO_DATA** for a given desired signed frequency deviation **Dppm** in [ppm] or in **Dppb** in [ppb] use the following formula:

jDCXO_DATA = round(**Dppm** * $2^{(41\text{-}\mathbf{bDCXO\_SHIFT})}/10^6$) = round(**Dppb** * $2^{(41\text{-}\mathbf{bDCXO\_SHIFT})}/10^9$)

where the **Dppm** must always be within the user specified interval <**-DmaxPpm**, **+DmaxPpm** > ppm and the same applies to the [ppb] equivalents.

7. DCXO Usage Notes

7.1. Reading Internal Frequency Deviation

The current internal frequency deviation can be read by I²C reading the **jDCXO_DATA** register. The value read is the frequency deviation value of the internal register before the final saturation. There is no streaming mode for reading the register. All 4 register bytes needs to be read. If **bI2C_INC_DIS**=1 and the user does not want to change that setting it can be accomplished by reading them separately in four different I²C transactions setting the specific byte register addresses **0x1d** to **0x20** manually, or in burst mode when the b **bI2C_INC_DIS**=0 setting the register address as **0x1d** and reading four subsequent data bytes in big endian (MSB is read first) fashion.

There is no way to read the frequency deviation after the final saturation **bVC_LPF_ABS_SAT**. Consult Figure 5.1 for data flow details.

7.2. Clearing Datapath

To clear all the internal datapath registers while keeping the DCXO configuration intact, write:

bDCXO_SYNC.dcxo_clr_sec = 1

which results in writhing the whole register with the value:

bDCXO_SYNC = 0x02

Writing the 1 to the **dcxo_clr_sec** field clears the internal datapath. The field is write 1 only, there is no need to write 0 there. Writing 1 to the **dcxo_clr_sec** field also clears the streaming logic so the next I²C byte written to address **0x20** is the MSB byte of the next **jDCXO_DATA** value burst.

7.3. Clearing Streaming Logic

When operating in **Streaming** mode the input **jDCXO_DATA** value bytes are written to a single byte register at address **0x20**. The number of bytes per each write must match exactly the configuration set in the **bDCXO_CTRL** register. Any extra or less byte during transaction will cause the streaming logic out of synchronization. To clear the streaming logic and put it to initial state when it expects first MSB byte of the first data value write:

bDCXO_SYNC.dcxo_stream_sync = 1

which results in writhing the whole register with the value:

bDCXO_SYNC = 0x01

Writing the 1 to the **dcxo_stream_sync** field forces the synchronization logic to be cleared and expects the MSB byte of the next value. The field is write 1 only, there is no need to write 0 there. If **dcxo_stream_sync** field is written as 1 only the state of he streaming logic is cleared. No other internal state or any configuration is changed.

The streaming logic is also cleared by the following:

- Writing **bDCXO_SYNC**.dcxo_clr_sec = 1
- Writing any value to **bDCXO_CTRL** register

7.4. Relative Mode Notes

If the DCXO is configured in **Relative** mode by setting **bDCXO_CTRL**.dcxo_rel_mode=1 the input **jDCXO_DATA** number is not used directly as a frequency deviation, but is added to the internal register which holds the current actual frequency deviation from the central frequency.

This mode allows controlling the frequency deviation **jDCXO_DATA** using representing relative steps from the current frequency deviation.

During **Relative** data mode the user may desire the following operations:

- Read the current internal frequency deviation
- Clear the internal frequency deviation register to get back to the central frequency and start over from there

How to accomplish either of these case is described above.

7.5. Relative Mode Saturation

As shown in Figure 5.1 there is an main 32 bit wide internal frequency deviation register (green color from where **jDCXO_DATA (Rd)** value is read), followed by low pass filter and output saturator controlled by **bVC_LPF_ABS_SAT**.

The internal register saturates at its maximum 32 bit wide values always representing the full frequency deviation range **±975** ppm and is independent of the output saturator.

This arrangement might present a subtle issue when using **Relative** mode. For example, let us assume that the output saturator **bVC_LPF_ABS_SAT** is configured to saturate the actual frequency deviation at **±100** ppm. Assume that the DCXO is configured to work in **Relative** mode and starts with the datapath cleared. The following table shows the sequence of input **jDCXO_DATA** values written to the device, the actual internal register, and the actual output frequency deviation from the center frequency after each **jDCXO_DATA** write:

DCXO Relative Mode Sequence Example

jDCXO_DATA [ppm]	Internal register [ppm]	Output frequency deviation [ppm]
--	0	0
+20	+20	+20
+50	+70	+70
+40	+110	+100
+20	+130	+100
+80	+210	+100
-10	+200	+100
-50	+150	+100
+20	+170	+100
+900	+975	+100
+220	+975	+100
+100	+975	+100
-800	+175	+100
-85	+90	+90
-20	+90	+90
-110	-20	-20
-200	-220	-100
-30	-250	-100
+80	-170	-100
-600	-770	-100
-400	-975	-100
-150	-975	-100
+300	-675	-100
+595	-80	-80
+20	-60	-60

The orange color indicates positive value saturation, the blue color indicates negative value saturation.

As can be observed from the data the relative input data is always applied to the internal register which saturates at the full range **±975** ppm. The output saturator then selects the window out of the full range internal register.

If the internal value is outside of the window of the output saturator the output will saturate at user specified level until the internal value falls back to the output saturator pass value range.

8. DCXO Configuration Examples

8.1. Streaming Absolute Mode with ~1 ppm Number Resolution

Requirements:

- Deviation number should have its LSB corresponding to ~1ppm: **LsbPpm** = 1
- Maximum input deviation used ± 600 ppm: **DmaxPpm** = 600.
- Saturate the datapath to ± 600 ppm: **DsatPpm** = 600.
- Low pass filter bandwidth should be set to pass through.
- **Absolute** mode required: The input frequency deviation corresponds to the actual frequency deviation from central frequency.
- **Streaming** mode of operation. This allows setting of multiple frequency deviations in a single I^2C transaction.
- Minimum number of **jDCXO_DATA** bytes should be used per one frequency deviation update.

Solution:

Start with calculating **bDCXO_SHIFT** based on the **LsbPpm**:

bDCXO_SHIFT = floor($\log_2$(**LsbPpm** * $2^{41}/10^6$)) = **21**

Check that it is less than 24:

bDCXO_SHIFT = 21 ≤ 24 .. success

Calculate the actual LSB step:

LsbActPpm = $10^6 / 2^{(41\text{-}bDCXO\_SHIFT)}$ = 0.9536743164 ppm

This is a close as we can get to the desired 1 ppm LSB step.

Calculate the minimum number of **jDCXO_DATA** bits based on **bDCXO_SHIFT** and **DmaxPpm**:

N_{bits} = ceil($\log_2$(**DmaxPpm** * $2^{(41\text{-}bDCXO\_SHIFT)}/10^6$)) + 1 = **10 bits**

M_{bytes} = ceil(N_{bits} / 8) = **2 bytes**

This means that two bytes of input data will be enough to represent the desired ±600 ppm deviation.

Calculate saturator setting with at least ±600 ppm range → use **ceiling** function:

bVC_LPF_ABS_SAT = min(255, ceil(**DsatPpm** * $2^{18}/10^6$)) = **158**

Calculate the actual saturation limit:

DsatActPpm = **bVC_LPF_ABS_SAT** * $10^6/2^{18}$ = 602.7 ppm.

Low pass filter setting set as pass through from register table:

bVC_LPF_BW_DIR = 7

Based on the calculations above, the configuration DCXO operation control register settings should be as follows:

bDCXO_CTRL.dcxo_size = 2

bDCXO_CTRL.dcxo_stream_mode = 1

bDCXO_CTRL.dcxo_rel_mode = 0 .. **Absolute** mode

bDCXO_CTRL.dcxo_ena = 1 .. enable DCXO operation

bDCXO_CTRL.dcxo_clr = 1 .. clear the DCXO hardware when 1 is written to this field

This corresponds to the **bDCXO_CTRL** register value:

bDCXO_CTRL = 1101_0010B = 0xd2

Other register settings:

bDCXO_SHIFT = 21

bVC_LPF_BW_DIR = 7 .. low pass filter set to pass through

bVC_LPF_ABS_SAT = 158

Since the streaming mode was set, the I²C address autoincrement must be disabled during the **jDCXC_DATA** writes:

bI2C_INC_DIS = 1 .. required for streaming operation

Do not forget to enable it back again for regular operation later on if needed.

For the actual DCXO frequency deviation streaming operation there will be 2 bytes for each deviation update sent in big endian fashion (MSB byte first) written to the I²C register address **0x20** as shown in Figure 5.1.

To calculate the two byte value of **jDCXO_DATA** based on the desired deviation **Dppm** the I²C master must perform the following calculation:

jDCXO_DATA = round(**Dppm** * $2^{20}/10^6$)

where in this case the **Dppm** must always be within the user specified interval <-600, +600> ppm.

The following figure shows fast updated deviation back to back to +100 ppm, +405 ppm, and -352 ppm:

+100 ppm: jDCXO_DATA = 105 = [0x00, 0x69]

+405 ppm: jDCXO_DATA = 425 = [0x01, 0xa9]

-352 ppm: jDCXO_DATA = -369 = [0xfe, 0x8f]

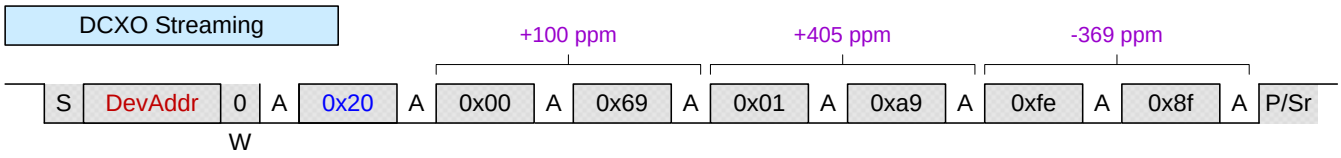

Figure 8.1 DCXO data Streaming example

Since the I²C master can control the bus for as long as necessary there could be gaps as long as needed in between the frequency deviation byte pairs.

8.2. Direct Absolute Mode with ~1 ppm Number Resolution

To contrast the **Streaming** data write with the **Direct** register data write with otherwise same settings as above we need to make only two changes in the configuration setting:

bDCXO_CTRL.dcxo_stream_mode = 0 .. **Direct** register mode

bI2C_INC_DIS = 0 .. keep register address autoincrement for **Direct** register operation

Keeping all the other settings the same, the I²C traffic for the same three frequency deviation updates would look as follows:

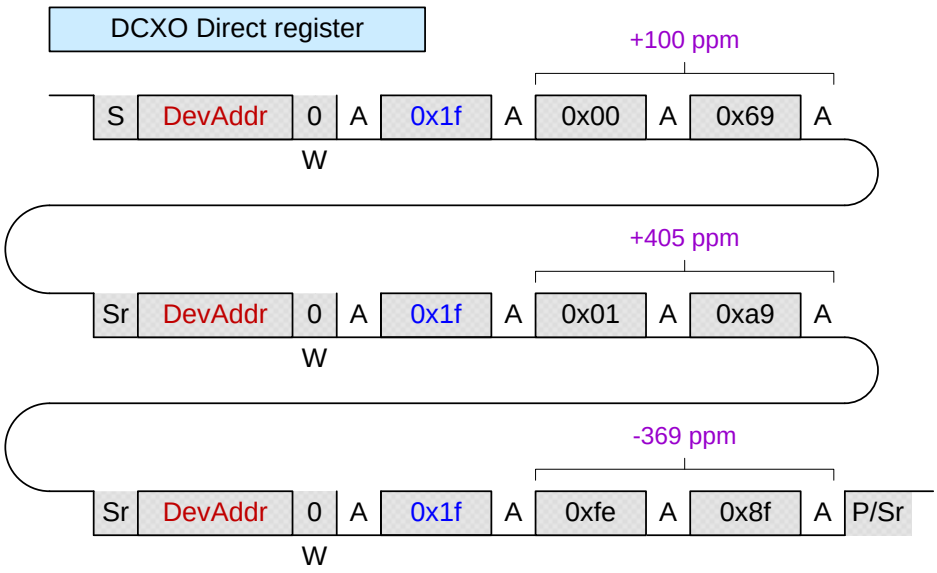

Figure 8.2 DCXO data Direct example

Notice the **0x1f** register address, which corresponds to the last 2 bytes of the **jDCXO_DATA** register. The last byte of the register at address **0x20** must always be written last with LSB byte of the data.

Each frequency deviation update requires new I^2C transaction with both device address **DevAddr** setting and the register address **0x1f** setting.

8.3. Streaming Relative Mode with ~1 ppb Number Resolution

Requirements:

- Deviation number should have its LSB corresponding to ~1ppb: **LsbPpb** = 1
- Maximum relative incremental deviation step ±20 ppm: **DmaxPpm** = 20.
- Saturate the datapath to ± 420 ppm: **DsatPpm** = 420.
- Low pass filter bandwidth should be set to pass through.
- **Relative** mode required: The input frequency deviation will be added to internal register which holds the current actual frequency deviation from central frequency.
- **Streaming** mode of operation. This allows setting of multiple frequency deviations in a single I^2C transaction.
- Minimum number of **jDCXO_DATA** bytes should be used per one frequency deviation update.

Solution:

Start with calculating **bDCXO_SHIFT** based on the **LsbPpm**:

bDCXO_SHIFT = floor($\log_2$(**LsbPpb** * $2^{41}/10^9$)) = **11**

Check that it is less than 24:

bDCXO_SHIFT = 11 ≤ 24 .. success

Calculate the actual LSB step:

LsbActPpb = $10^9 / 2^{(41-\text{bDCXO\_SHIFT})}$ = 0.93132257 ppb

This is a close as we can get to the desired 1 ppb LSB step.

Calculate the minimum number of **jDCXO_DATA** bits based on **bDCXO_SHIFT** and **DmaxPpm**:

N_{bits} = ceil($\log_2$(**DmaxPpm** * $2^{(41-\text{bDCXO\_SHIFT})}/10^6$)) + 1 = **16 bits**

M_{bytes} = ceil(N_{bits} / 8) = **2 bytes**

This means that two bytes of input data will be enough to represent the desired ±20 ppm deviation incremental step with ~1 ppb resolution of the number.

Calculate saturator setting with at least ±420 ppm range → use **ceiling** function:

bVC_LPF_ABS_SAT = min(255, ceil(**DsatPpm** * $2^{18}/10^6$)) = **111**

Calculate the actual saturation limit:

DsatActPpm = **bVC_LPF_ABS_SAT** * $10^6/2^{18}$ = 423.4 ppm.

Low pass filter setting set as pass through from register table:

bVC_LPF_BW_DIR = 7

Based on the calculations above, the configuration DCXO operation control register settings should be as follows:

bDCXO_CTRL.dcxo_size = 2

bDCXO_CTRL.dcxo_stream_mode = 1

bDCXO_CTRL.dcxo_rel_mode = 1 .. **Relative** mode

bDCXO_CTRL.dcxo_ena = 1 .. enable DCXO operation

bDCXO_CTRL.dcxo_clr = 1 .. clear the DCXO hardware when 1 is written to this field

This corresponds to the **bDCXO_CTRL** register value:

bDCXO_CTRL = 1111_0010B = 0xf2

Other register settings:

bDCXO_SHIFT = 11

bVC_LPF_BW_DIR = 7 .. low pass filter set to pass through

bVC_LPF_ABS_SAT = 111

Since the streaming mode was set, the I^{2}C address autoincrement must be disabled during the **jDCXC_DATA** writes:

bI2C_INC_DIS = 1 .. required for streaming operation

Do not forget to enable it back again for regular operation later on if needed.

For the actual DCXO frequency deviation streaming operation there will be 2 bytes for each deviation update sent in big endian fashion (MSB byte first) written to the I^{2}C register address **0x20** as shown in Figure 5.1.

To calculate the two byte value of **jDCXO_DATA** based on the desired deviation **Dppm** the I^{2}C master must perform the following calculation:

jDCXO_DATA = round(**Dppm** * $2^{30}/10^6$)

where in this case the **Dppm** must always be within the user specified interval <-20, +20> ppm.

9. Revision History

Rev	Date	Description
1.0	Mar 2021	- Initial release