Introduction

This application note describes the feature of moving the frequency by a small amount within the VersaClock® 5 (VC5) and VersaClock 6 (VC6) device families. Details are stated on how the frequency changes occur in the device and how to program the frequency changes. Additional measurements made with a Keysight oscilloscope DSO-S 404A illustrate the functionality of this feature. From a few ppm change to a percentage of frequency, the IC can do it all in a seamless programmable fashion.

Diverse applications require glitchless frequency change like the ones mentioned below:

- **Frequency Margining** which is a process where the nominal oscillator frequency is shifted higher, or lower, to understand and evaluate the robustness of a digital system. By increasing or decreasing the frequency, the system designer can test the limits (in other words, margin) of their system.

- Small frequency changes can be made in video applications to enable gunlock synchronization.

- Some applications using spread spectrum may also need small frequency changes.

**VersaClock 5/6 Programmable Clocks FOD**

IDT’s VersaClock 5 and VersaClock 6 programmable clocks are able to perform a glitchless frequency changes via their FOD (Fractional Output Divider) block.

The resolution will be $1/((N \times 2^P)$ where $N$ is the programmed integer and $P$ is the amount of bits for the fraction. The smallest $N$ is 4 and with 24 bits fractional; the worst case resolution is $1/(4 \times 2^{24}) = 1.5E-8$ or 0.015ppm. The 24-bit accurate FODs are capable of accuracy better than 0.015ppm.

The FOD can be programmed to get an output from 1MHz to 350MHz for the same VCO frequency. Total FOD division ratio will be $(\text{Int} + \text{Frac}/2^{24}) \times 2$.

The block diagram in Figure 1 shows the details of the VersaClock 5 and VersaClock 6 output architecture. Each output has its own FOD which enables a flexibility to program different frequency margining changes for each output. The FODs are programmable through an I2C interface. A full description is available in VC5 and VC6 register programming (see https://www.idt.com/document/mau/versaclock-5-family-register-descriptions-and-programming-guide).

**Figure 1. VersaClock 5/6 Family Block Diagram**
Frequency Margin Programming Example

Below is an example of programming the FOD1 for output 1, starting with 100MHz and moving the frequency by +1% and -1%:

Table 1. FOD1 Settings for OUT1 = 100MHz

<table>
<thead>
<tr>
<th>FOD1 settings</th>
<th>Output (MHz)</th>
<th>FOD Value Decimal</th>
<th>Integer Registers 0x2D 0x2E</th>
<th>Offset Value Decimal</th>
<th>Offset Registers 0x22 0x23 0x24 0x25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting frequency</td>
<td>100</td>
<td>14</td>
<td>00 E0</td>
<td>0</td>
<td>00 00 00 00</td>
</tr>
<tr>
<td>+1%</td>
<td>101</td>
<td>13.86138614</td>
<td>00 D0</td>
<td>0.86138614</td>
<td>03 72 0F 34</td>
</tr>
<tr>
<td>-1%</td>
<td>99</td>
<td>14.14141414</td>
<td>00 E0</td>
<td>0.14141414</td>
<td>00 90 CE E0</td>
</tr>
</tbody>
</table>

The VCO frequency or PLL block output frequency is 2800MHz in this case. The PLL programming and original FOD setting can be very easily executed with the Timing Commander software supporting the VC5 family (see Figure 2 below).


Figure 2. Timing Commander Settings for 100MHz Output Clock

After programming the device via I2C to the desired frequency, with all PLL settings and other output type settings, the FOD register can be specifically reprogrammed to change it to a different frequency. The bit setting change can be done with a different I2C Software interface of one's choice, or, contact IDT to obtain access to the internal I2C utility tool. Table 2 below shows an example of the FOD1 bit register change.

Table 2. FOD1 Register Change from 100MHz to 99MHz

<table>
<thead>
<tr>
<th>Output (MHz)</th>
<th>X22</th>
<th>X23</th>
<th>X24</th>
<th>X25</th>
<th>X26</th>
<th>X27</th>
<th>X28</th>
<th>X29</th>
<th>X2A</th>
<th>X2B</th>
<th>X2C</th>
<th>X2D</th>
<th>X2E</th>
<th>X2F</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>E0</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>0</td>
<td>90</td>
<td>CE</td>
<td>E0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>E0</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>
After the FOD1 reprogramming occurred, a perfect frequency transition from 100 MHz to 99MHz can be seen without any glitch, ringing or overshoot as shown in Figure 3.

- The green trace shows the final edge in the \( I^2C \) clock signal that indicates a “stop” of the \( I^2C \) data transfer.
- The yellow trace shows the output1 clock signal.
- The purple trace is a trend of the output 1 frequency, showing the actual transition.

As seen on the oscilloscope, it takes 1.5\( \mu \)s after the last \( I^2C \) transaction is done to change to the new frequency. The frequency transition itself, from 100MHz to 99MHz, is almost instantaneous, without any sign of a glitch or overshoot. Nevertheless, the whole sequence of sending new values to the FOD via \( I^2C \) (until the frequency changes) depends of the \( I^2C \) speed used by the user and the \( I^2C \) controller.

In the IDT lab, the \( I^2C \) transaction took approximately 12ms with the selected speed setting of 100kHz.

Figure 3. Frequency Transition from 100MHz to 99MHz
Programming 100ppm Offset from 100MHz Clock
A change in 100ppm will be performed in same manner as a 1% change. The programming of the FOD1 register is shown in Table 3.

Table 3. FOD1 Settings for OUT1 = 100MHz

<table>
<thead>
<tr>
<th>FOD1 settings</th>
<th>Output (MHz)</th>
<th>FOD Value Decimal</th>
<th>Integer Registers 0x2D 0x2E</th>
<th>Offset Value Decimal</th>
<th>Offset Registers 0x22 0x23 0x24 0x25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting frequency</td>
<td>100.00</td>
<td>14.25</td>
<td>00 E0</td>
<td>0.25</td>
<td>01 00 00 00</td>
</tr>
<tr>
<td>+100 ppm</td>
<td>100.01</td>
<td>14.248575142</td>
<td>00 E0</td>
<td>0.248575142</td>
<td>00 FE 8A 78</td>
</tr>
</tbody>
</table>

The VCO frequency or PLL block output frequency is 2850MHz in this case. The transition time from 100MHz to 100.01MHz (+100ppm) will take the same amount of time as programming a 1% change (~1.5μs).

Frequency Change with Spread Spectrum
This section illustrates how to program the VersaClock 5 or VersaClock 6 FOD with a frequency change of 1% and spread spectrum enabled. Figure 5 below shows an example of programming FOD output1 with SSC bit on, and the modulation rate is at 31.5KHz with -0.5% spread.
Figure 5. Timing Commander Example of Programming FOD1

Table 4. FOD1 Settings to 100MHz with Frequency Change and -0.5% Down Spread

<table>
<thead>
<tr>
<th></th>
<th>Output (MHz)</th>
<th>Divider (real)</th>
<th>Integer Registers</th>
<th>Period (hex)</th>
<th>Step (real)</th>
<th>Step (hex)</th>
<th>Offset (real)</th>
<th>Offset Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>100.0</td>
<td>14.25</td>
<td>00 E0</td>
<td>633</td>
<td>4.48696E-05</td>
<td>2F1</td>
<td>0.25</td>
<td>01 00 00 02</td>
</tr>
<tr>
<td>+0.5% Change</td>
<td>100.5</td>
<td>14.17910448</td>
<td>00 E0</td>
<td>63B</td>
<td>4.44866E-05</td>
<td>2E9</td>
<td>0.17910448</td>
<td>00 B7 67 2A</td>
</tr>
</tbody>
</table>

OUT1, -0.5% Spreading
Address: 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
100.0MHz: 00 81 01 00 00 02 00 02 F1 31 9C 00 00 00 E0 00
100.5MHz: 00 81 00 B7 67 2A 00 02 E9 31 DC 00 00 00 E0 00

The spread is programmed on top of the margining frequency.

Figure 6 shows an oscilloscope capture of the output clock with -0.5% spread with a frequency change of +0.5%. The original frequency being 100MHz is 0.5% down-spread and the ending frequency is 100.5MHz with 0.5% down-spread.

After the FOD value change has been initiated, the frequency transition will happen when the spread step is at its maximum (top of the triangle). The purple line shows the frequency transition.
Figure 6. Oscilloscope Image of Output Frequency Transition

Conclusion

The different examples that were given in this application note illustrate how easy it is to perform any frequency changes with VersaClock 5/6 Fractional Output Divider programming. From a few ppm to percentage of frequency, the changes can be made without any glitch, ringing or overshoot.