Introduction

This application note provides details on how to program the frequency for devices in the 3rd generation Universal Frequency Translator family. This guide will use the following example for the calculations:

Jitter Attenuator Example

- Input CLK0 frequency: 25MHz
- Output Frequency: 125MHz (integer output divider)
- Output 2 Frequency: 155.52MHz (fractional output divider)
- Crystal frequency: 38.88MHz
- Crystal Doubler: Enabled

(Please reference script "8T49N28X-register-calculations.py" for details on how to implement these calculations in Python).

Overview

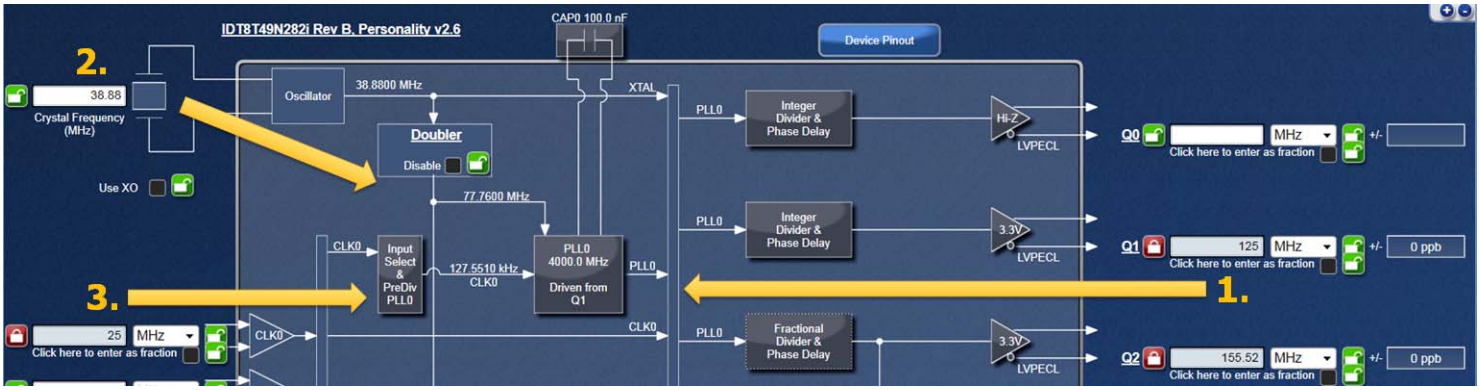
The process involves three main steps:

1) Calculate the possible output dividers and choose the highest value. This establishes the VCO frequency.
2) Based on the ratio between the VCO frequency and Crystal frequency, calculate the upper loop feedback divider settings.
3) If the device is operating in jitter attenuator mode, calculate the input clock pre-divider setting and the lower loop feedback divider setting.

A fourth, and optional step, is to calculate the translation error:

4) Use the register settings to calculate the actual output frequency. Compare that result to the target frequency.

Figure 1. Jitter Attenuator vs Synthesizer Mode



Jitter attenuator mode uses one or more input reference frequencies and translates them to new output frequencies while removing jitter at the same time.

Synthesizer mode generates output frequencies directly from the crystal input. In this mode of operation, the input pre-dividers and the lower loop are not used.

When the device is used in jitter attenuator mode, steps 1 through 3 are all necessary. When the device is used in synthesizer mode, only steps 1 and 2 apply.

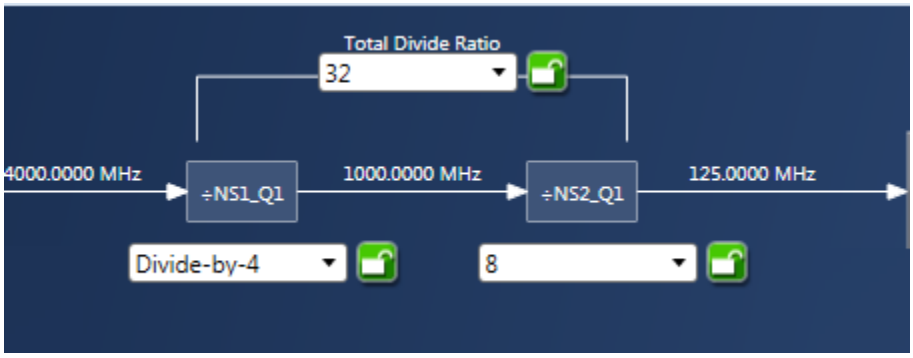
Calculate the Output Dividers

In this example, both the 125MHz and 155.52MHz output frequencies will be derived from the same VCO. The 125MHz output will be derived using an integer divider and 155.52MHz will be derived using a fractional divider. The VCO is first calculated using the integer divider path, then the fractional divider is calculated based on that VCO frequency.

Integer Divider Calculation

The integer output divider has two stages, NS1 and NS2. The total divide ratio is NS1*NS2.

Figure 2. Integer Output Divider Stages: NS1 and NS2



The valid settings for NS1 and NS2 are:

NS1=÷4, ÷5, or ÷6 (/1 option cannot be used when driving NS1 from the VCO)

NS2=÷1, ÷2, ÷2*n for n=2 to 2^16

Below is a tabular representation of the NS1 and NS2 settings:

Table 3. Q0, Q1, Q[4:7] Output Divide Ratios

1st-Stage Divide	2nd-Stage Divide	Total Divide	Minimum F_{OUT} MHz	Maximum F_{OUT} MHz
4	1	4	750	1000
5	1	5	600	800
6	1	6	500	666.7
4	2	8	375	500
5	2	10	300	400
6	2	12	250	333.3
4	4	16	187.5	250
5	4	20	150	200
6	4	24	125	166.7
...				
4	131,070	524,280	0.0057	0.0076
5	131,070	655,350	0.0046	0.0061
6	131,070	786,420	0.0038	0.0051

NOTE: Above frequency ranges for Q[4:7] apply when driven directly from PLL0 or PLL1.

The output frequency is related to the VCO frequency as follows:

Fout=Fvco/N

where

Fout= Output frequency

Fvco= VCO frequency

*N= NS1*NS2*

The valid VCO range is 3000-4000MHz, meaning that dividers that result in a VCO frequency outside that range are invalid. In order to obtain the list of valid output dividers, the VCO max and min frequencies are divided by the output frequency. Note that the dividers must be integers so it will be necessary to perform rounding to determine the limiting divider values, as follows:

OutDivMin = math.ceil (VCOmin / Fout) – minimum divider value requires rounding UP

OutDivMax = math.floor (VCOmax / Fout) – maximum divider value requires rounding DOWN

Next, iterate through the possible NS1 and NS2 combinations that fall within the OutDivMin and OutDivMax range. Choose the highest valid divider and keep this as the default (N_default). This establishes the VCO frequency as follows:

In this example,

OutDivMin = math.ceil(3000/125) = 24,

OutDivMax = math.floor(4000/125) = 32

From 24 to 32, the following valid NS1 & NS2 combinations are possible:

NS1 = 4, NS2 = 6, N=4*6 = 24,

NS1 = 4, NS2 = 8, N=4*8 = 32,

NS1 = 5, NS2 = 6, N = 5*6 = 30

NS1 = 6, NS2 = 4, N=6*4 = 24.

So, the valid output dividers are 24, 30, and 32. Choosing the highest for the default, N_default=32. Therefore,

*Fvco= 125MHz*32= 4000MHz*

The other valid output dividers can be kept as alternates in case a different VCO frequency is required.

This code below can be used to map the N_default value to the NS1 and NS2 register values:

```
############################################################
#INTEGER DIVIDER: Determine NS1 register setting
############################################################
if N_default < 4:               # Only use the divide-by-1 option for really small divide ratios -
                                # note that this option will never be on the list for the Q0 - Q3 dividers
        best_setting = 3
if N_default == 4 or N_default %  8 == 0:       # Make sure we can divide the ratio by 4 in NS1
                                                #and by 1 or an even number in NS2
        NS1_reg_setting = 2                     #"Divide by 4 register selection
if N_default == 5 or N_default % 10 == 0:       # Make sure we can divide the ratio by 5 in NS1
                                                #and by 1 or an even number in NS2
        NS1_reg_setting = 0                     # Divide by 5 register selection
if N_default == 6 or N_default % 12 == 0:       # Make sure we can divide the ratio by 6 in NS1
                                                #and by 1 or an even number in NS2
        NS1_reg_setting = 1  #Divide by 6 register setting
```

```
############################################################
#INTEGER DIVIDER: Determine NS2 register setting
############################################################
if   NS1_reg_setting == 0 : NS1_ratio = 5
elif NS1_reg_setting == 2 : NS1_ratio = 4
elif NS1_reg_setting == 1 : NS1_ratio = 6
elif NS1_reg_setting == 3 : NS1_ratio = 1      # This is the bypass (divide-by-1) option
else : NS1_ratio = 6
    NS2_ratio = int ( math.floor ( N_default / NS1_ratio ) )
    NS2_reg_setting = int(math.floor ( NS2_ratio / 2 ))
```

Fractional Divider Calculation

Now that the VCO frequency has been established, the effective fractional divider for the second output frequency is calculated as follows:

FracDiv=Fvco/Fout2

For this example:

FracDiv=4000MHz/155.52MHz= 25.7201646090534979

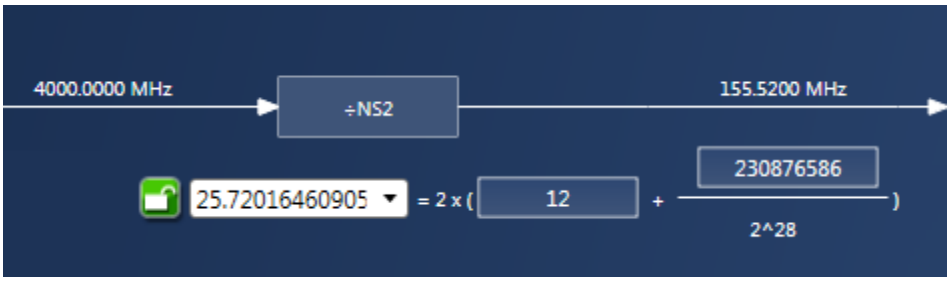
The fractional output divider ratio is composed of an integer and fractional component:

FracDiv = 2(N.F)*

N = Integer Part: 4, 5, ...(2^18-1)
F = Fractional Part: [0, 1, 2, ...(2^28-1)]/(2^28)

For integer operation of these outputs dividers (i.e. when fractional portion is 0), N = 3 is also supported

Figure 3. Fractional Divider Calculation for the Second Output Frequency



N is written directly to the registers. The NFRAC register setting is derived as follows:

*NFRAC= rounding of F*2^28 to the close integer*

In the example,

2(N.F) = 25.7201646090534979*

N.F=12.86008230452674895

N=12
F=0.86008230452674895

*NFRAC= rounding of F*2^28 to the closest integer*

*0.86008230452674895*2^28=230876585.6131687185907712*

NFRAC=230876586

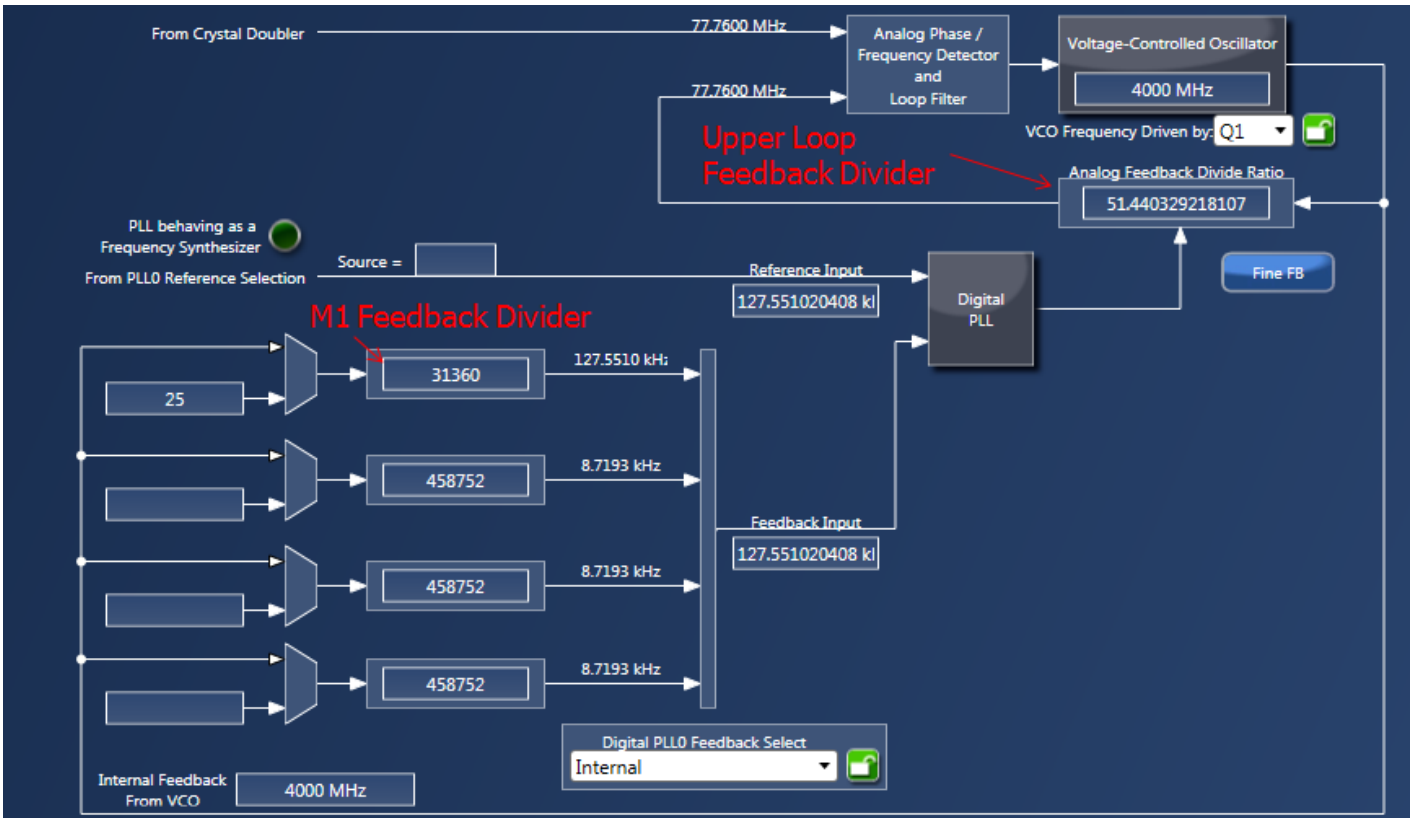
This code can be used for the calculations:

```
frac_numerator= round ((( FracDiv / 2.0 ) - int(FracDiv/2.0) ) * (2**28))
#This is the case where the fractional portion is 0.
if frac_numerator>=268435456 or FracDiv/ 2.0==int(FracDiv/ 2.0):      #Due to precision limitations, sometimes fractional portion of the Effective divider gets rounded to 1.  This checks for that condition.
        N_Q2= int(round(FracDiv / 2.0))
        NFRAC_Q2=0  #Set fractional portion to 0
#This is the case where the fractional portion is not 0.
else:
        N_Q2= int(math.floor ( FracDiv/ 2.0 ) )
        NFRAC_Q2=int(round ((( FracDiv / 2.0 ) - int(FracDiv/2.0) ) * (2**28)))
Fin=25          #Input Frequency
Fout=125        #Output frequency, Int Divider
Fout2=155.52    #Output frequency, Frac Divider
Fxtal=38.88     #Xtal doubler
XTAL_Doubler=1  #Register
```

Calculate the Upper Loop Feedback Divider

The 8T49N28X device has an upper loop (where the analog PLL and VCO reside) and a lower loop (where the DPLL resides). The upper loop feedback divider is calculated first.

Figure 4. Calculating the Upper Loop Feedback Divider



The feedback divider for the upper loop is the ratio between the Crystal Input frequency and the VCO frequency. If enabled, the Crystal doubler multiplies the crystal input by a factor of 2. The calculation for the upper loop feedback divider is:

*UpperFBDiv= Fvco/ (Fxtal*XTAL_Doubler)*

When mapped to the registers, the feedback divider (sometimes called the M1 divider) setting has two components:

DSMint.DSMFrac

where

DSMint = integer UpperFBDiv

*DSMFrac= fraction of UpperFBDiv * 2^21, rounded to the closest integer*

For the example, the crystal doubler is enabled, so the analog feedback divider is:

*UpperFBDiv= Fvco/ (Fxtal*XTAL_Doubler)= 4000/(38.88*2)= 51.440329218107*

DSMint=51

*DSMFrac=round 0.440329218107*2^21= 923437*

Calculate the Lower Loop Feedback Divider

The device has an input pre-divider (P) that scales the input clock signal going into the digital PFD (phase frequency detector). It also has a feedback divider ratio (M1) that scales the feedback clock signal going into the same PFD. Ideally, both inputs into the PFD should have the same frequency. That relationship is expressed as follows:

Fpfd =Fin/P = Fvco/M1

where

Fpfd= phase frequency detector input frequency
Fin= input frequency
M1= lower loop feedback divider
P= input clock predivider
Fvco= VCO frequency

P and M1 must both be integers, so the relationship between P and M1 is the following:

*M1= P*Fvco/Fin, rounded to the nearest integer*

For each P value, a corresponding M1 value can be determined while the valid ranges for P and M1 are taken into consideration.

The maximum PFD input frequency is 128kHz, or 0.128MHz. The recommended minimum PFD input frequency is 500Hz. From this, the minimum and maximum P values can be calculated as follows:

Pmin= Fin/PFDmax, (if this result is <1, then set Pmin=1)
Pmax= Fin/PFDmin (if this result is >2^20, then set Pmax=2^20)

M1 must then be calculated for the range, starting with Pmin in order to target the lowest input divider, ie, highest PFD input frequency for best performance.

Pmin<P <Pmax

When iterating through the P values, each P/M1 ratio should be compared to the ratio Fin/Fvco. Choose the first result that provides an error of 0. If such result is not found, then choose the result with the lowest error and a PFD frequency of >500Hz and <128KHz.

For the example, the resulting P and M1 values are:

P =195
M1=31200
Fin/P = 25/195 = 0.128205128205128205128205128205128205
Fvco/M1=4000/31200= 0.128205128205128205128205128205128205

P can be directly written to the register for the appropriate input. M1 can also be directly written to the register associated with the corresponding clock input.

This calculation needs to be repeated for each input reference that may be used to drive that VCO.

Calculate the Output Frequency Error

Within the device there are 3 sources of frequency error. Which of these sources may affect the output frequency depends on the configuration of the device.

1) **Upper loop feedback fraction (Synth)** – the upper PLL loop makes use of a Delta-Sigma Modulator (DSM) to create a feedback ratio that consists of an integer and a fractional portion. Due to the limited number of bits available to represent the fraction portion (21 bits), there will be rounding error in some cases. Note that this rounding error will not affect the output frequency in Jitter Attenuator mode because the lower-loop will 'steer' the DSM fraction to exactly track the input reference. This error is typically <7.45ppb, which is insignificant when compared to the accuracy of a crystal (which is usually in the 10's of ppms). If a very accurate source is used and an error accuracy of less than 1ppb is desired, please consult the document *8T49N28x Fine Fractional Feedback Adjustment* for instructions on how to fine tune the fractional feedback divider.
2) **Lower loop PFD Frequency mismatch (JA)** – the Phase / Frequency Detector (PFD) circuit in the lower loop compares a pre-divided input reference frequency to the VCO frequency divided by M1. As discussed above, since both P and M1 values are integers, there will be cases where the two input frequencies to the PFD are not exactly matched. The error is proportional to the mismatch between the two frequencies. Note that this error will not affect the output frequency in Synthesizer mode because the lower-loop is not used.
3) **Fractional Output Divider rounding (FOD)** – a fractional output divider has a limited number of bits (28 bits) to represent the fractional portion of the desired output divider ratio. Because of this, rounding error may affect the output frequency. Note that this rounding error will not affect the output frequency when the Fractional Output Divider (Output Q2 or Q3) is programmed with a purely integer value (i.e. fraction is 0) or is not part of the output path (the Q2 or Q3 divider may be selected as a frequency source for any of the Q4-7 outputs and may introduce this rounding error to those output frequencies if used in this way).

The net error can be calculated by using the register settings to calculate the actual output frequency, based on the input and comparing it to the desired output frequency. The calculations are:

*Fout_lower_calc = Fin/P * M1 /N_default (JA error)*
*Fout_upper_calc=Fxtal*XTAL_Doubler/(DSMInt+DSMFrac/2^21))/N_default (Synth error)*

*Fout2_lower_calc=Fin/P * M1 /(2*(N_Q2+ float(NFRAC_Q2)/2**28)) (JA + FOD errors)*
*Fout2_upper_calc=Fxtal*XTAL_Doubler/(DSMInt+DSMFrac/2^21))/ (2*(N_Q2+ float(NFRAC_Q2)/2**28)) (Synth + FOD errors)*

*(Fcalc-Ftarget)/Ftarget*1e9 (scale result to parts-per-billion)*

For the example:

*Fout_lower_calc= 25/195*31200/32= 125.0, (0ppb)*
*Fout_upper_calc= 38.88*2*(51+923437/2097152)/32= 124.99999651908874511718 75, (-2.78ppb)*

*Fout2_lower_calc= 25/195*31200/(2*(12+230876586/2^28))= 155.51999998257294 (-0.112ppb)*
*Fout2_upper_calc= 38.88*2*(51+923437/2097152)/(2*(12+230876586/2^28))=155.519999549492 (-2.897ppb)*

Each of these errors affects the output frequency differently:

Fout_lower_calc ppb error: this is the actual translation error in a particular output frequency when used in Jitter Attenuator mode with integer dividers only in the output path.

Fout_upper_calc ppb error: If the device is operating in synthesizer mode, this is the error in the ratio between the Crystal and the Output Frequency and will show up on the output. In this case, since this DSM error is on the scale of ppb, it may be negligible compared to the frequency accuracy of the Crystal or TXCO input.

Fout2_lower_calc ppb error: this is the actual translation error in a particular output frequency when used in Jitter Attenuator mode with a Fractional Output divider, programmed with a non-zero fraction in the output path.

Fout2_upper_calc ppb error: this is the actual translation error in a particular output frequency when used in Synthesizer mode with a Fractional Output divider, programmed with a non-zero fraction in the output path.

Again, reference script “8T49N28X-register-calculations.py" for details on how to implement these calculations in Python or to double-check any manual calculations performed.

For details on how to adjust the feedback divider settings without triggering a recalibration, refer to the document titled *8T49N28x Fine Fractional Feedback Adjustment*.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) 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 who are designing with Renesas 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. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.