NXP Semiconductors
Application Notes

Document Number: AN12176
Rev. 1, 12/2018

MPC5777C Clock Calculator Guide

How to use MPC5777C tool to easily calculate device frequency domains

by: NXP Semiconductors

Contents

1. Introduction

NXP's MPC5777C is a dual-core 32-bit microcontroller dedicated to high-performance powertrain applications. The MCU supports ASIL-D automotive safety rating and runs on two e200z7 Power Architecture cores. The MPC5777C features two versions, a standard version whose core runs up to 264 MHz and a performance version that supports up to 300 MHz. This application note will refer to these versions as "MPC5777C_264MHz" and "MPC5777C_300MHz", respectively, for version-specific information; general content that applies to both versions will refer to the device as simply "MPC5777C".

The MPC5777C supports an 8-44 MHz external oscillator (XOSC), a 16 MHz internal RC oscillator (IRC), and two phase locked loops (PLL). MPC5777C_264MHz's two PLLs support 200 MHz and 264 MHz, respectively; MPC5777C_300MHz's respective PLLs can run up to 240 MHz and 300 MHz. The IRC is selected out of reset so increasing the operating frequency from 16 MHz requires additional configuration. The MPC5777C Clock Calculator is meant to complement the reference manual. It seeks to simplify the clock configuration process by providing a graphical, interactive tool to help the user find the

correct register settings in order to achieve the desired clock frequencies.

Accompanying this application note is the clock calculator itself. You can download it from MPC5777C clock calculator.

The clock calculator makes use of macros to perform functions like resetting the spreadsheet to initial values, configuring all clock frequencies to the fastest allowable settings, and copying generated code. Macros must be enabled in the user's MS Excel settings to access these features. If macros are turned off, however, the tool will still be able to calculate clock frequencies, but the aforementioned features will be disabled. To turn on macros in MS Excel 2016, go to the *Developer* tab on the top toolbar and click on *Macro Security*. A popup window will appear. In it, select *Enable all macros*.

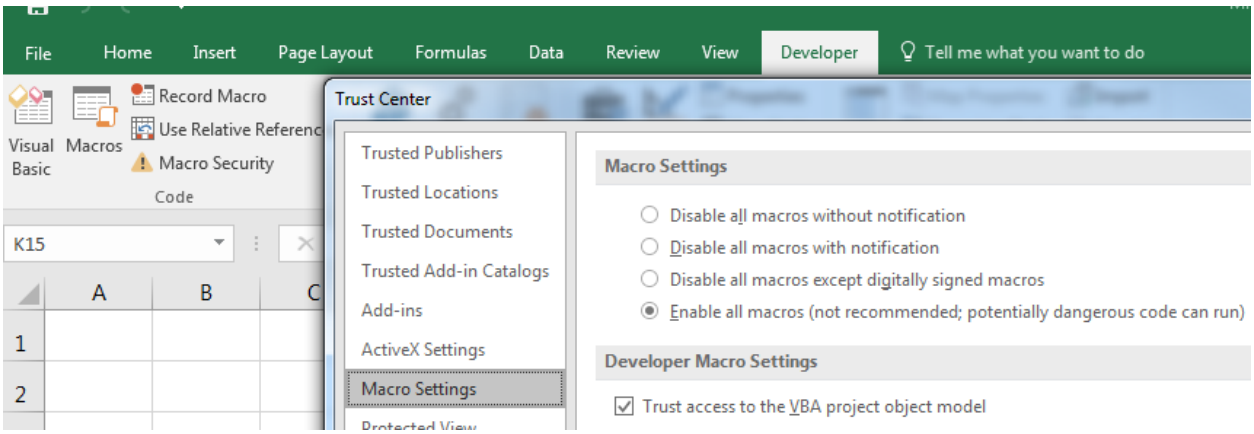


Figure 1. **Enable macros**

2. Clock calculator design

The MPC5777C clock calculator takes the form of an interactive Microsoft Excel spreadsheet organized into multiple tabs as shown in the following figure.

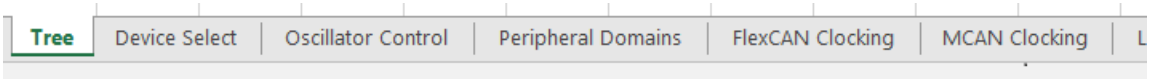


Figure 2. **MPC5777C Clock calculator setup**

Clock sources (e.g. oscillators and PLLs) propagate to the various clock domains from which the MCU modules take their clocks. Most cells representing clock domain frequencies are not to be modified manually. The user is meant to enter frequencies to the few select clock sources and all clock domain frequencies derive from these sources. Several clock domain inputs *are* meant to be modified manually as they represent external clocks that are driven into a pin. There are also input cells that set muxes and clock dividers. All cells that take entries have blue borders instead of black, as shown in the figure below.

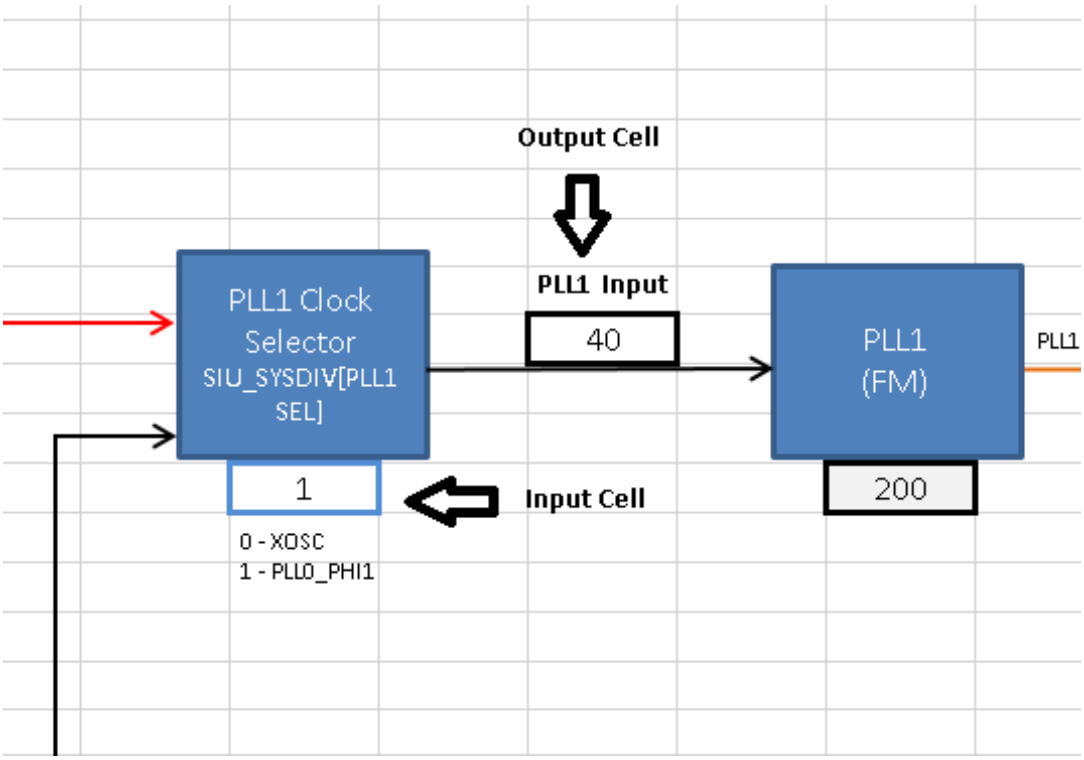


Figure 3. **Input cells vs. Output cells**

There are limits to what frequencies can be entered to the input frequency cells. Values that are out of range will be rejected and the user will receive an error message. Invalid clock domain frequencies that arise from valid input values and legal, but improper, dividers will be shaded in red, as will be explained in greater depth later in this application note.

Frequency values are linked across tabs, so *PER_CLK* in the *Tree* tab will always be the same as *PER_CLK* in the *Peripheral Domains* tab. Hyperlinks are provided to duplicate domain names to link back to their points of origin. For example, *PER_CLK* originates in *Tree*. So clicking the *PER_CLK* textbox in *Peripheral Domains* will take the user to *PER_CLK* in *Tree*. Textboxes that are links, when hovered over, will cause the mouse cursor to turn into a hand icon and a pop-up to appear, showing the destination address, as shown in the following figure.

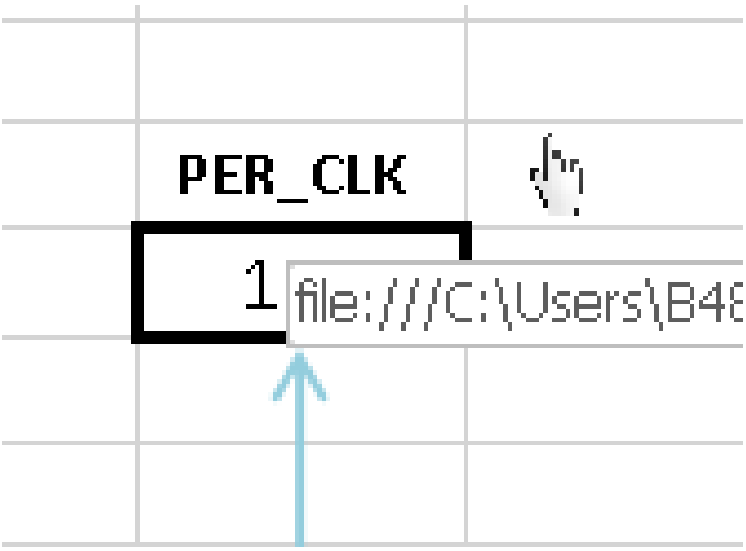


Figure 4. **Clicking on a link**

The following subsections will explain in depth the purpose of each tab.

2.1. Tree

Tree is the centerpiece of the tool. This tab is the starting point for all clock frequency calculations. It is organized to resemble the MPC5777C clock tree as presented in the following figure.

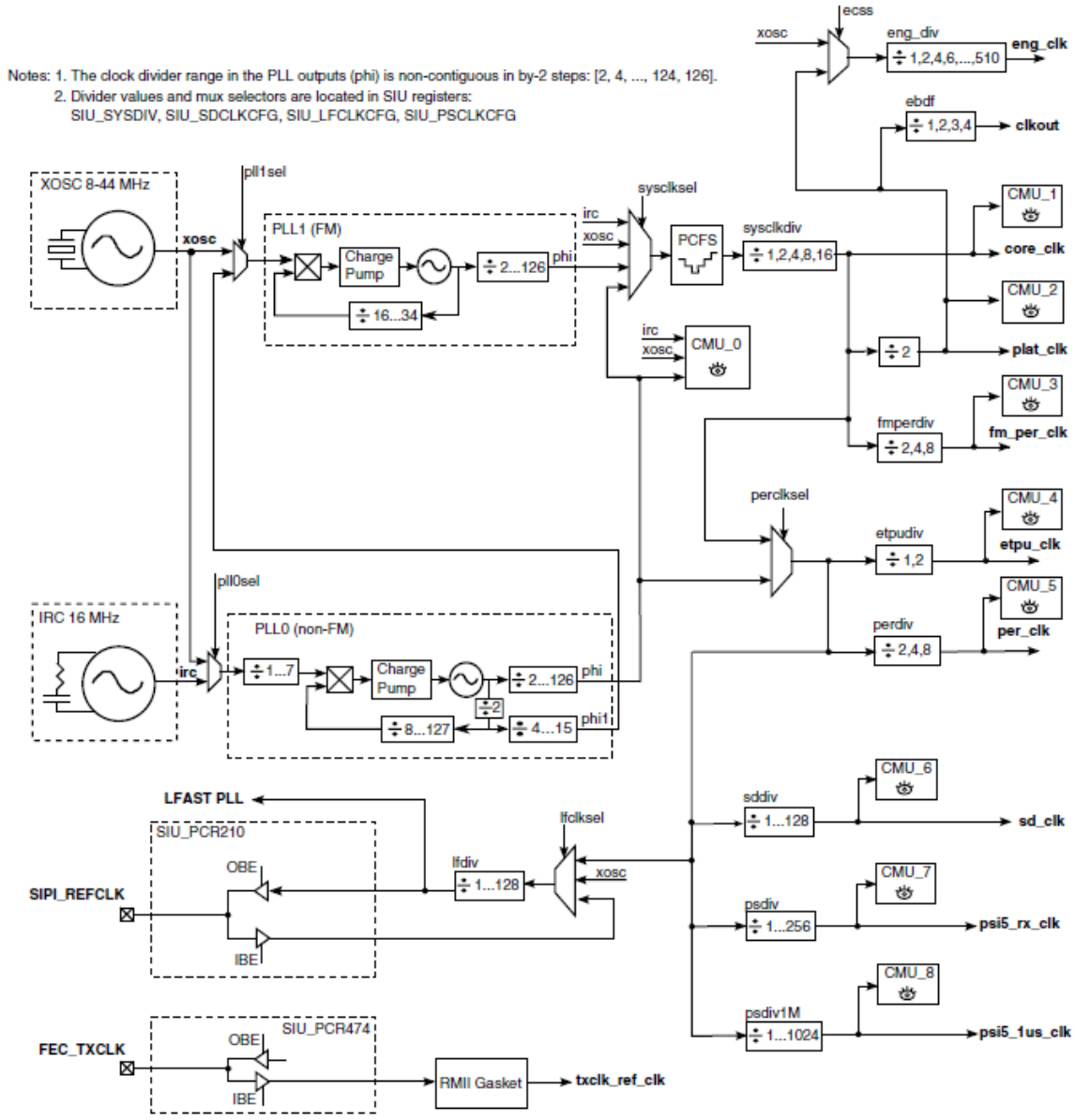


Figure 5. **MPC5777C Reference manual clock Tree**

Figure 6 shows, in part, the diagram's clock tool counterpart. The difference between the two is that the latter is interactive.

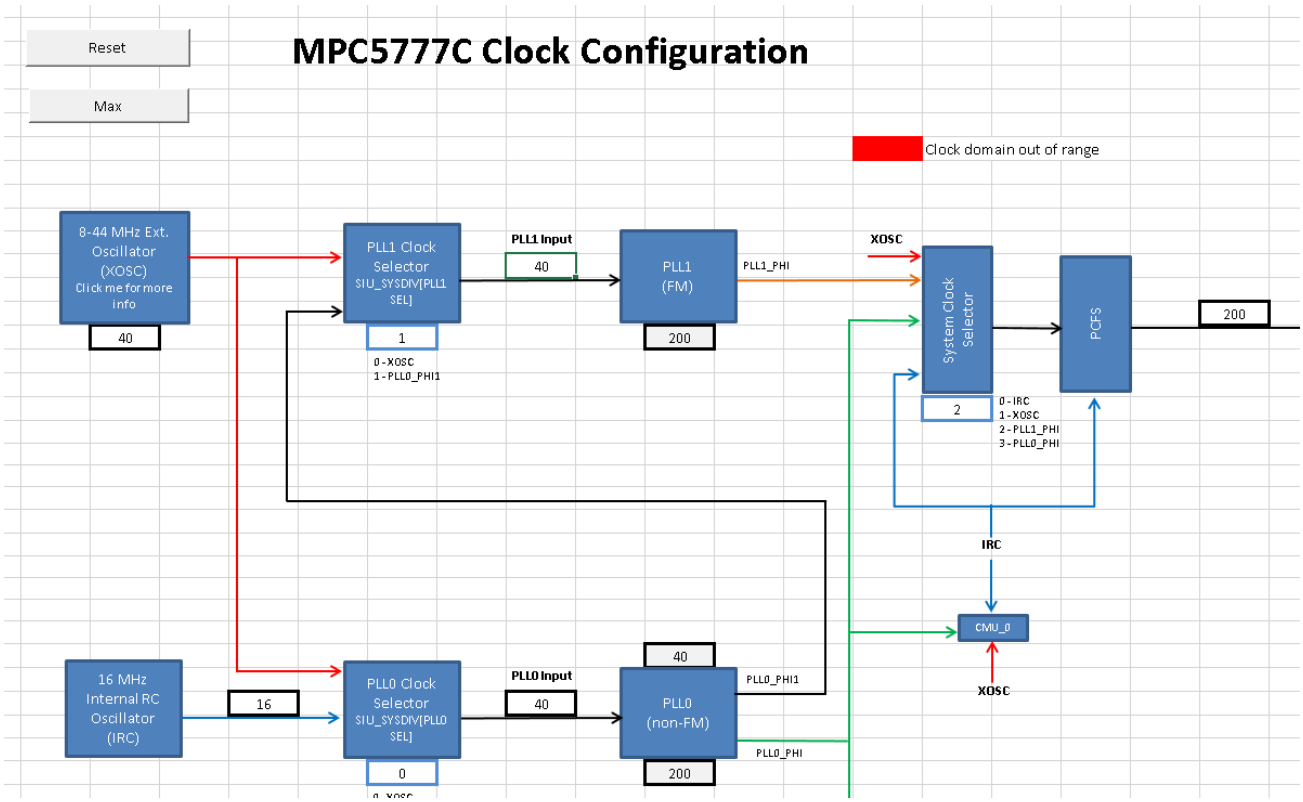


Figure 6. **Clock calculator Tree**

The flow of the diagram generally goes from left to right. On the left are the MPC5777C clock sources and on the right are the clock domains. MCU modules run on one or more of these clock domains.

Clock domain frequency values are displayed in the outlined cells next to their labels. Most cells are not meant to be written to; their values are dependent on the frequencies of preceding steps in the clock tree. Take FM_PER_CLK, for example: its value is sourced from either the system clock, which in turn comes from either the IRC, XOSC, PLL0_PHI, or PLL1_PHI. Now look at the XOSC block. XOSC depends on the settings from the Oscillator Control tab. So all clock domains ultimately derive from a few select inputs. The IRC is always on at 16 MHz, PLL0_PHI and PLL1_PHI are configured in the PLL0 and PLL1 tab, respectively. The system clock selects from these four clock sources by selecting the value of the System Clock Selector block. Then finally the selected signal is divided by the FM_PER_CLK prescaler value. It is important to note, though, that the user input for the divider field is not the desired divider, but the bitfield value that one would have to enter to achieve the desired divider. Some divider blocks like FM_PER_CLK's, provide a dropdown list with textbox description underneath. Others support too many input values to be put into a list. The user must provide the proper bitfield value, not the desired divisor. That is why such blocks are labelled something like "/(1+(0…63))" rather than simply "/1…64". The user provides a value between 0 and 63, to which the hardware automatically adds 1 to calculate a divider that is between 1 and 64.

Tree also features two buttons, Reset and Max. They only function when macros are enabled. Clicking on these buttons with macros disabled will return an error. If macros are enabled, the Reset button will set all blocks to their reset value, as described in the reference manual. The Max button sets all blocks in this tool to values that configure the system and auxiliary clock domains to their respective maximum allowable frequencies. Below is a screenshot of the buttons.

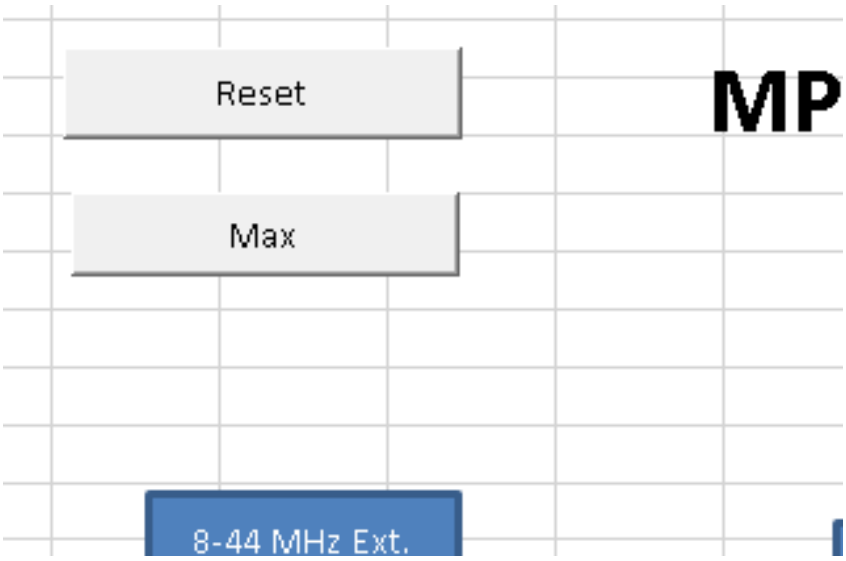


Figure 7. **Buttons**

2.2. Device select

The *Device Select* tab allows user to choose between MPC5777C_264MHz and MPC5777C_300MHz. This changes the hardware specification table used by the MPC5777C Clock Calculator. For example, if MPC5777C_300MHz is selected, *PLL1_PHI* will shade red starting at 300 MHz rather than 264 MHz. The structure of the tool remains unchanged, though, since the two MPC5777C versions share the same architecture, but with different operating speeds.

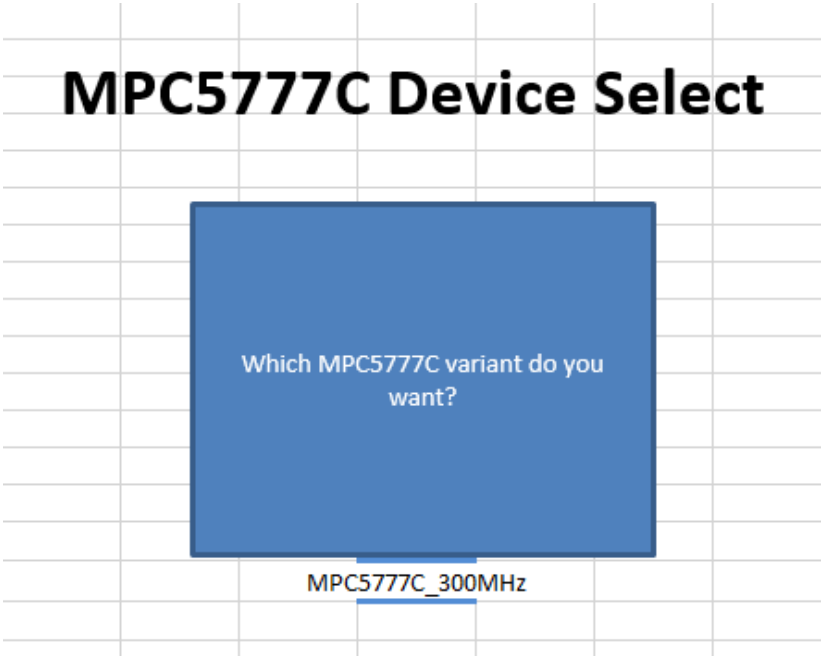


Figure 8. **Device select**

2.3. Oscillator control

Oscillator Control controls the generation of the external oscillator (XOSC) frequency. MPC5777C supports two ways of XOSC generation. The chip has two external oscillator pins, *XTAL* and *EXTAL*. An 8-44 MHz external oscillator can be connected to both pins. This external oscillator is also referred to simply as *XTAL*. If the *XOSC Enable/Select* block selects *XTAL*, XOSC will derive its frequency from the external oscillator (XTAL) block. Alternatively, a waveform can be driven directly to the *EXTAL* pin, also known as bypass mode. This signal is also referred to simply as *EXTAL*. When the *XOSC Enable/Select* block selects *EXTAL*, XOSC will derive its frequency from the *EXTAL* pin. Shown below is a screenshot of oscillator control.

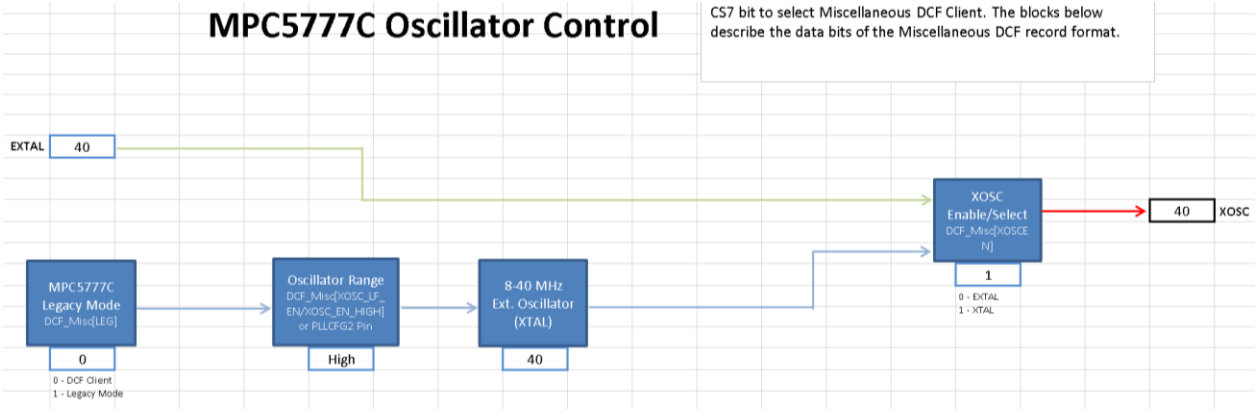


Figure 9. **Oscillator control**

2.4. Peripheral domains

Peripheral Domains is an in-depth diagram of MPC5777C modules. Where Tree leaves off at the clock domain level, Peripheral Domains picks up and progresses to the module level, as shown in the figure below.

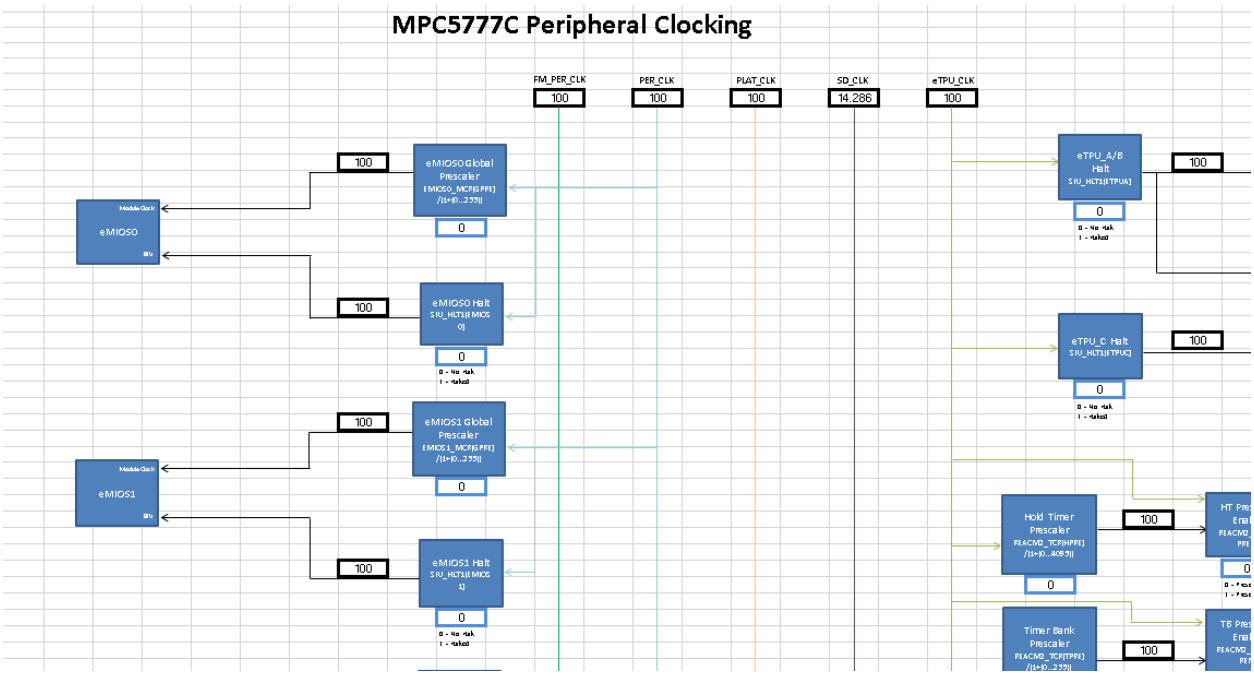


Figure 10. **Peripheral domains**

The clock domains are color-coded. Black lines are reserved for clock domains that only a few modules use. For example, many modules' BIU clocks are filtered through a Halt block. The output of each Halt block is colored black, since each only goes to at most a few modules. As a rule of thumb, clock domains are represented with black lines if all modules using it can fit within a single window without having to scroll. The frequencies on this tab are not meant to be modified and are dependent on frequency values in the Tree tab.

2.5. FlexCAN clocking and MCAN clocking

Some modules have so many clock configuration possibilities that they warrant their own dedicated sections, lest Peripheral Domains becomes too complicated. FlexCAN and MCAN uses their protocol clocks to generate can message time segments. Domains still hosts FlexCAN and MCAN blocks that show their input clocks and are hyperlinked to FlexCAN Clocking and MCAN Clocking, respectively, as shown in the below figure.

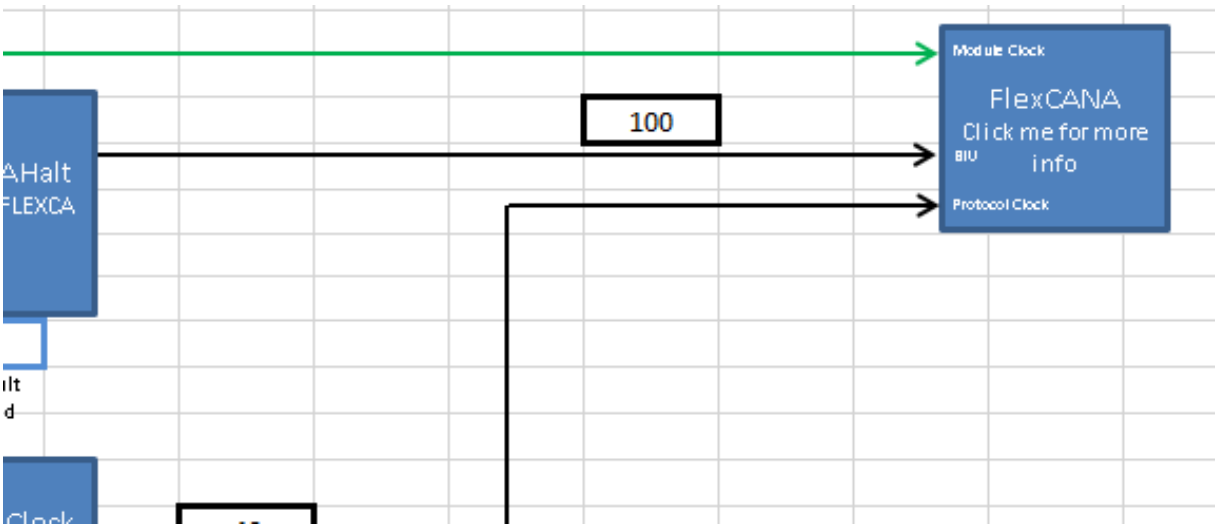


Figure 11. **FlexCAN in *Peripheral domains***

FlexCAN clocking and *MCAN clocking* provides options to configure the bit time segments, as shown in the figure below. The column on the right-hand side of the diagram shows the breakdown, in nanoseconds, of a CAN bit time, as configured by the FlexCAN or MCAN.

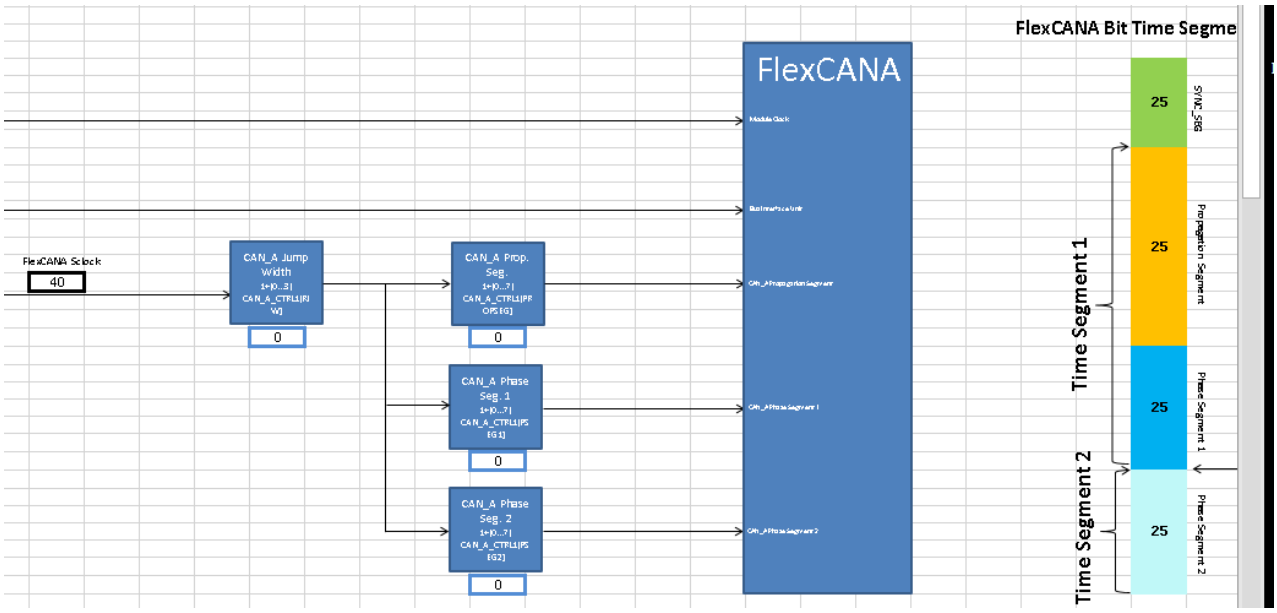


Figure 12. ***FlexCAN clocking***

2.6. LFAST clocking

The LFAST is the third module that has its own tab. It is a versatile, but intricate module that hosts its own PLL which generates multiple phases and generates a signal within specification only if its inputs are certain frequencies. These intricacies make it necessary to give LFAST its own dedicated tab.

LFAST clocking presents a block diagram of the module with various clocks going into it. It also supports LFAST_PLL configuration to increase the LFAST frequency up to 480 MHz. The LFAST also

supports a low-speed mode as well as a high-speed mode. This tool allows the user to select between the two modes. Below is a screenshot of LFAST clocking.

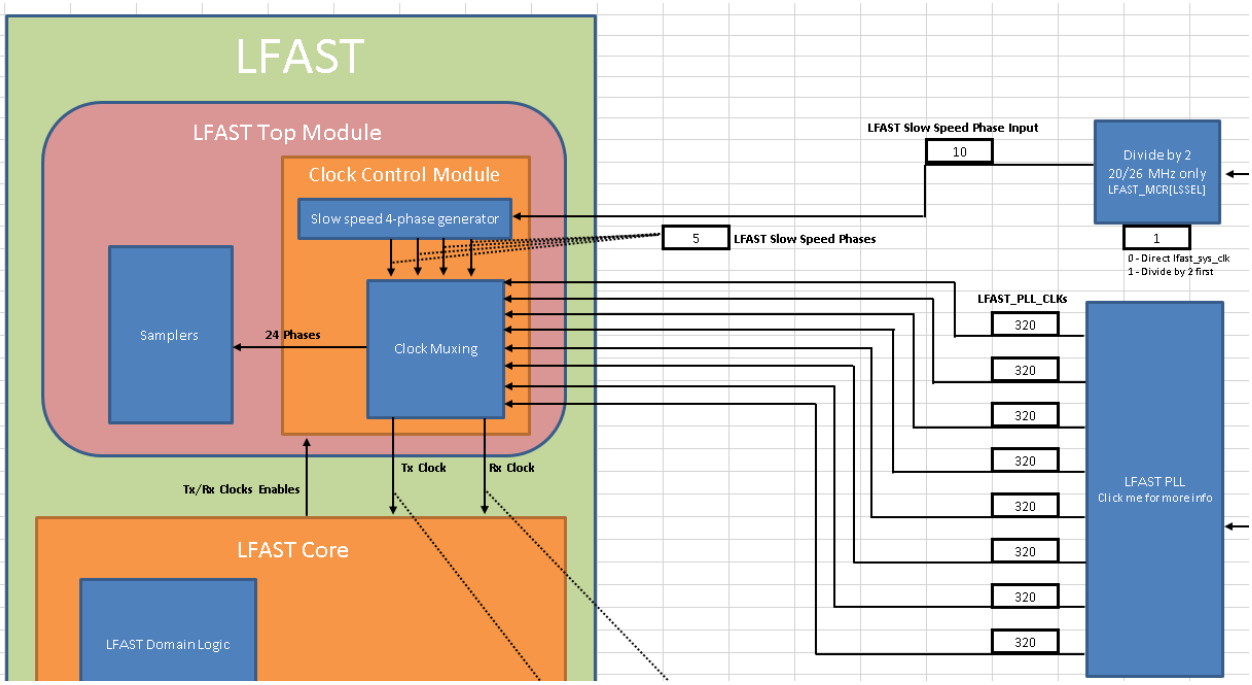


Figure 13. **LFAST block diagram**

Since the LFAST signal must be generated from an input clock of 10 or 20 MHz, this tool blocks any input from the signal *RF_REF* other than these two values. *RF_REF* can technically be set to any value, but the signal goes through the clock calculator's *LFAST Input Filter* block to become *lfast_sys_clk*, which in turn is the signal that gets fed into the *LFAST PLL* and phase generators, as shown in the following figure.

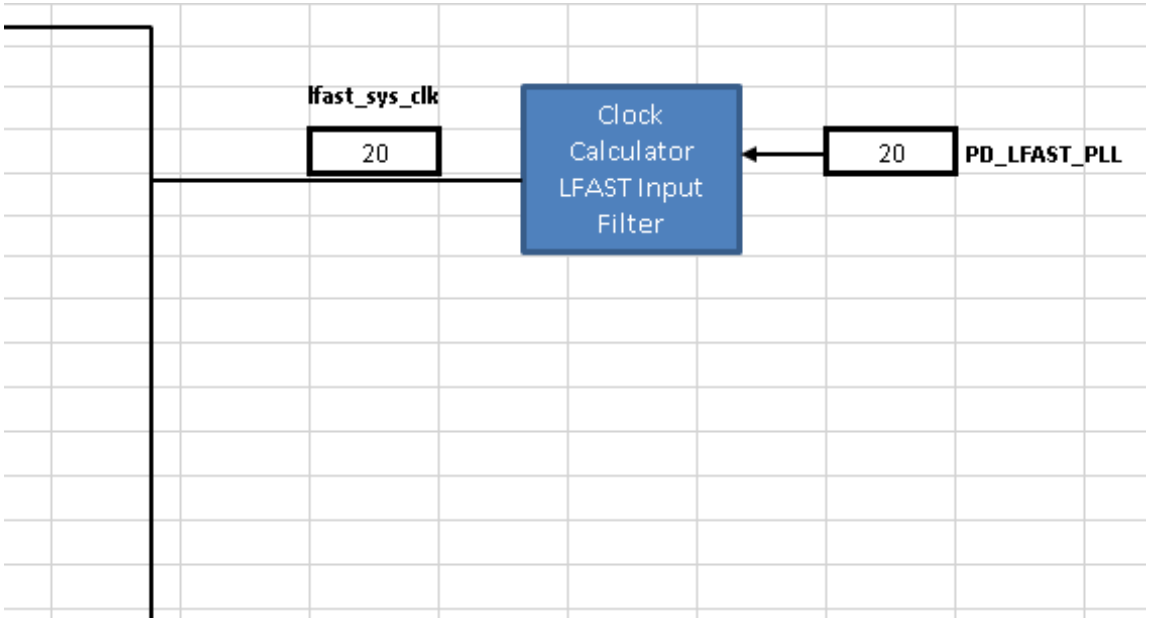


Figure 14. **LFAST clocking input filter**

If *RF_REF* is 10 or 20 MHz, *lfast_sys_clk* is the same; otherwise, *lfast_sys_clk* is 0. MPC5777C does not actually filter *RF_REF* the way this tool does. The purpose of the *LFAST Input Filter* block is to simulate how the user can technically set *RF_REF* to any value, but the resulting LFAST output would be unusable. Therefore, if a user were to enter an invalid input frequency (i.e. not 10 or 20 MHz), all subsequent frequencies would be 0, and the user would know to change the input.

2.7. PLLx

PLL0 and *PLL1* are visual abstractions of the PLL digital interface, as shown in the figure below.

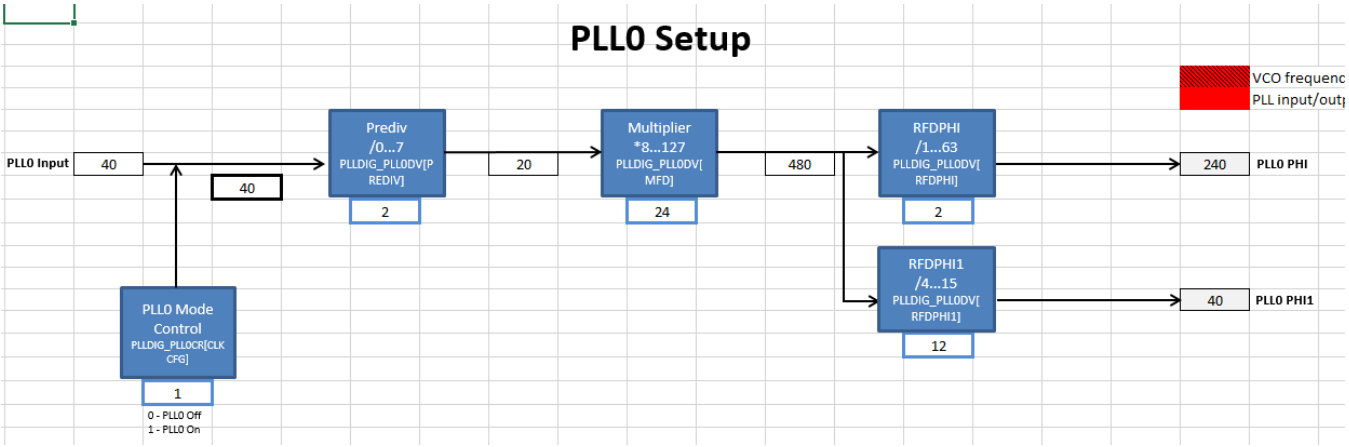


Figure 15. **PLL0 control**

The input source of PLL0 and PLL1 are selected by the *PLL0 Clock Selector* and *PLL1 Clock Selector* blocks in the *Tree* tab, respectively. Then, from the source, the dividers and multipliers located in the *PLL0* and *PLL1* tabs are set in order to achieve the PLL output frequencies. The PLL output frequencies are in turn propagated to the *PLLx_PHIn* clock domains in the *Tree* tab.

2.8. Reference tables (pll0_phi, pll0_phi1, and pll1_phi)

The three tabs *pll0_phi*, *pll0_phi1*, and *pll1_phi* are reference tables for the user to find the appropriate PLL dividers and multipliers to achieve the desired PLL frequency. There is a tab for each PLL output because input frequencies and the range of acceptable divider/multiplier values differ between each other. However, they all follow the same setup. Note that Columns A, B, and C of these tabs are frozen so if the table looks cut off, just scroll left or right.

PLL frequencies are calculated from a reference frequency, a reference divider (RFD), a multiplier (MFD), and in PLL0, a prescaler (PREDIV). The PLL reference is not manually configurable because there is a finite number of input values the PLL can take. For example, PLL0 can only reference either the 16 MHz IRC or the 8-44 MHz XOSC. PLL reference therefore comes from the *Tree* tab. Configure *PLL0 Clock Selector* and *PLL1 Clock Selector* in *Tree* for PLL0 and PLL1, respectively. Once the PLL reference frequency is configured, enter the desired PLL output frequency. Also, enter the PREDIV value when using *PLL0_PHI* or *PLL0_PHI1*. The reference table will then calculate the output frequency for each MFD and RFD setting. Like in the other sections, frequencies are color-coded to define which values are valid and which are not. Shading will change automatically once the output PLL frequencies are calculated. MFD and RFD settings that achieve the exact desired frequency will be

shaded in green; values that exceed the desired frequency, but are within MPC5777C hardware specifications are marked in yellow; and frequencies that exceed the MPC5777C hardware specification are colored red. Below is a screenshot of the reference table for *PLL0_PHI*.

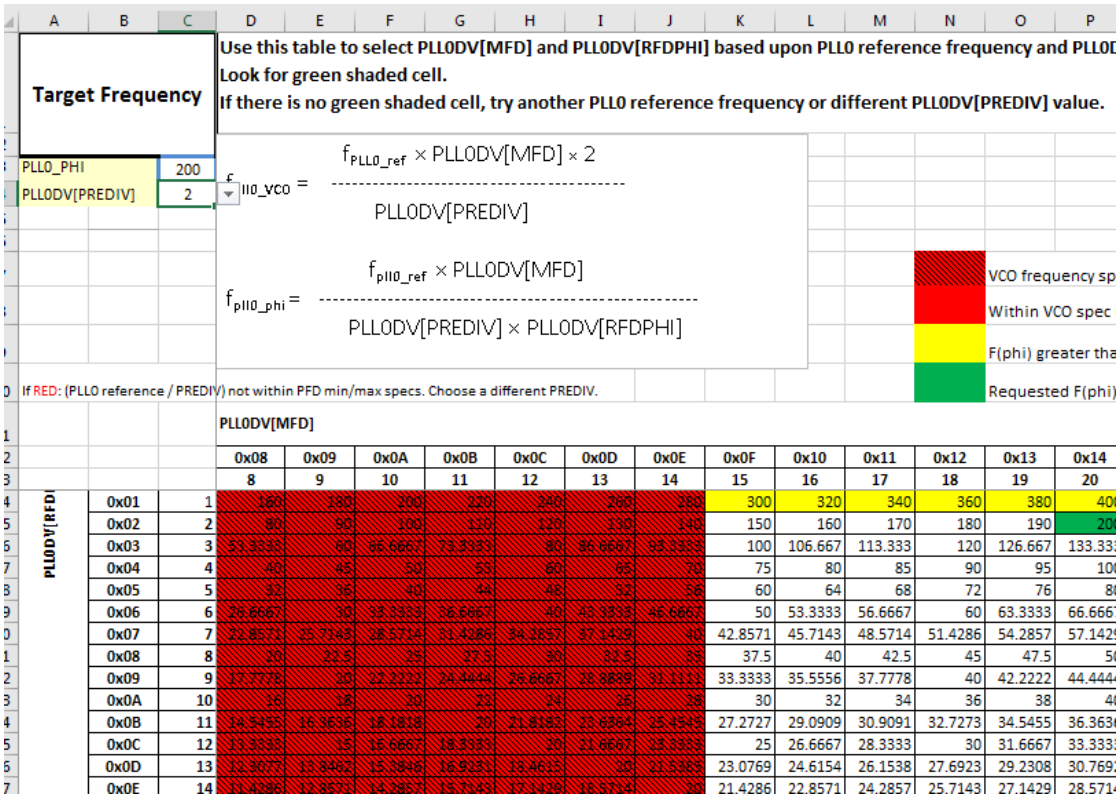
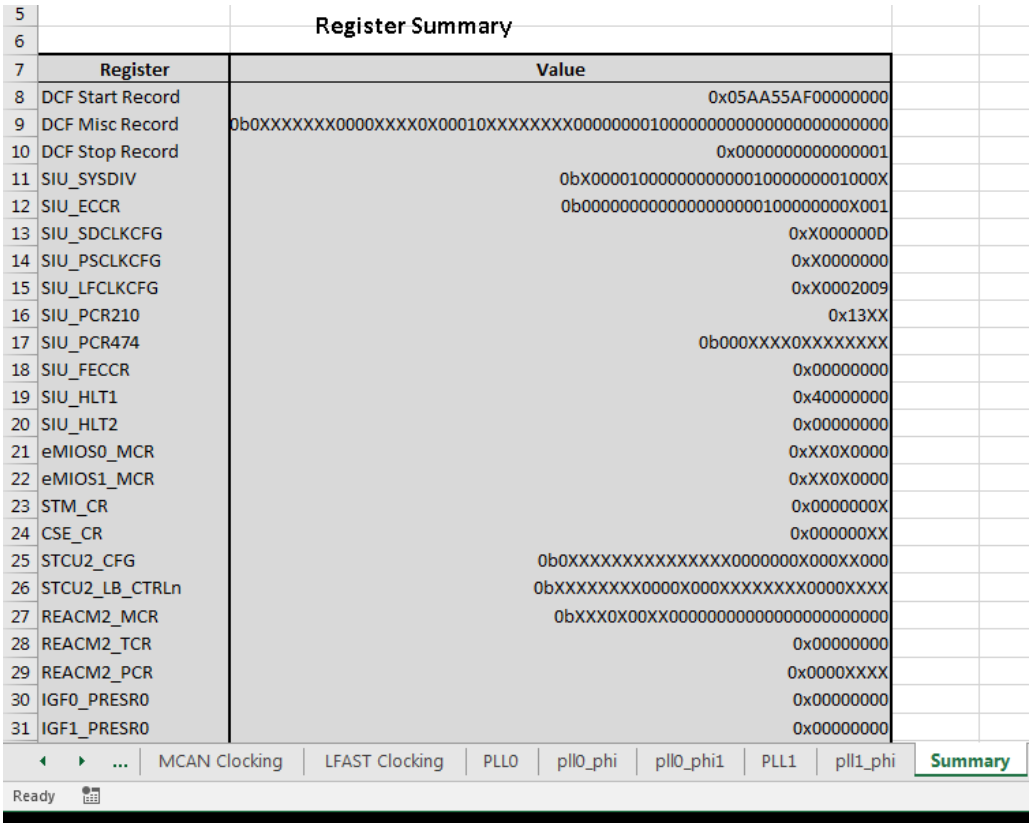


Figure 16. ***PLL0_PHI* Reference table**

2.9. Summary

Almost all blocks populating this clock calculator represent real register fields in silicon. The *Summary* tab collates all the information from the rest of the clock calculator into a list of register values, a screenshot of which is shown in Figure 16. The values in the register summary are interactive, updating automatically when the associated block is changed. Registers listed within *Summary* are only the ones whose values are affected by clock configuration, not every single register available in the chip.



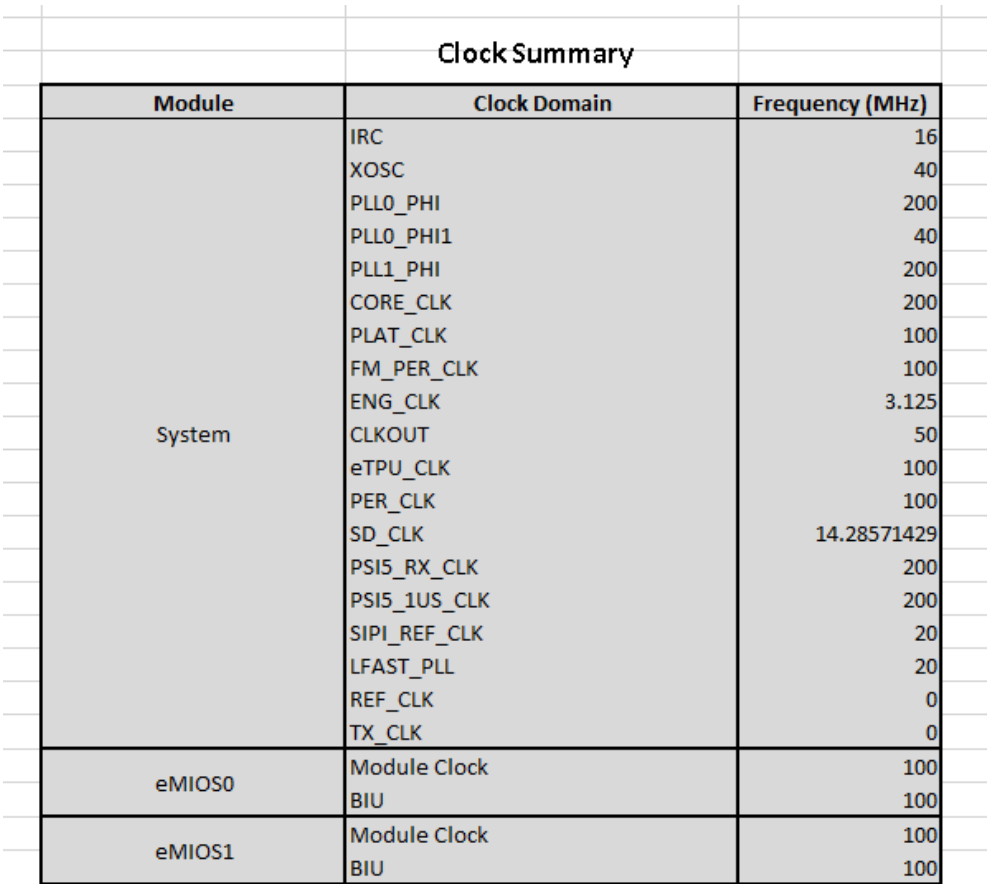
Register Summary

Register	Value
DCF Start Record	0x05AA55AF00000000
DCF Misc Record	0b0XXXXXXX0000XXXX0X00010XXXXXXXX00000000100000000000000000000000
DCF Stop Record	0x0000000000000001
SIU_SYSDIV	0bX00001000000000001000000001000X
SIU_ECCR	0b00000000000000000100000000X001
SIU_SDCLKCFG	0xX000000D
SIU_PSCLKCFG	0xX0000000
SIU_LFCLKCFG	0xX0002009
SIU_PCR210	0x13XX
SIU_PCR474	0b000XXXX0XXXXXXXX
SIU_FECCR	0x00000000
SIU_HLT1	0x40000000
SIU_HLT2	0x00000000
eMIOS0_MCR	0xXX0X0000
eMIOS1_MCR	0xXX0X0000
STM_CR	0x0000000X
CSE_CR	0x000000XX
STCU2_CFG	0b0XXXXXXXXXXXXXXX0000000X000XX000
STCU2_LB_CTRLn	0bXXXXXXXX0000X000XXXXXXX0000XXXX
REACM2_MCR	0bXXX0X00XX00000000000000000000000
REACM2_TCR	0x00000000
REACM2_PCR	0x0000XXXX
IGF0_PRESR0	0x00000000
IGF1_PRESR0	0x00000000

Figure 17. **Register summary table**

The register values are displayed in either hexadecimal or binary format, where a “0x” prefix represents hexadecimal and “0b” denotes binary. A capital “X” represents a “don’t care” bit/half-byte. These bits do not affect the clock frequency, so users can set these values to whatever suits their purposes. Users can best utilize *Summary* by setting the configuration they want in the clock calculator and then copying the resulting register value into code. For example, taking from the figure above, the register SIU_LFCLKCFG should be set to 0xX0002009. Assuming the instances of “X” are “0”, the resulting S32DS C code would be: “SIU.LFCLKCFG.R = 0x00002009;”.

Summary also includes an overview of the clock domain frequencies. Since this tool consists of multiple interdependent spreadsheets, it might be cumbersome for users to weave through them all to find a clock domain. This table provides a place where all of them can be found. The table is organized by module, followed by the clock type (i.e. BIU clock, peripheral clock, protocol clock, etc.), and finally the frequency, as currently configured. Below is a screenshot of clock summary table.



Clock Summary

Module	Clock Domain	Frequency (MHz)
System	IRC	16
	XOSC	40
	PLL0_PHI	200
	PLL0_PHI1	40
	PLL1_PHI	200
	CORE_CLK	200
	PLAT_CLK	100
	FM_PER_CLK	100
	ENG_CLK	3.125
	CLKOUT	50
	eTPU_CLK	100
	PER_CLK	100
	SD_CLK	14.28571429
	PSI5_RX_CLK	200
	PSI5_1US_CLK	200
	SIPI_REF_CLK	20
	LFAST_PLL	20
	REF_CLK	0
	TX_CLK	0
eMIOS0	Module Clock	100
	BIU	100
eMIOS1	Module Clock	100
	BIU	100

Figure 18. **Clock summary table**

This tool also supports a degree of code generation. *Summary* provides two sample clock initialization functions, *Sysclk_Init* for configuring oscillators and PLLs and *InitPeriClkGen* for providing sources/dividers to auxiliary clocks. The dynamic C code in these functions depend on tool settings just like the register summary. These functions can be copied and pasted to a source file via Ctrl+C/Ctrl+V or by clicking on the associated *Copy Code* button, if macros are enabled. The following figure shows *Sysclk_Init* and its *Copy Code* button.

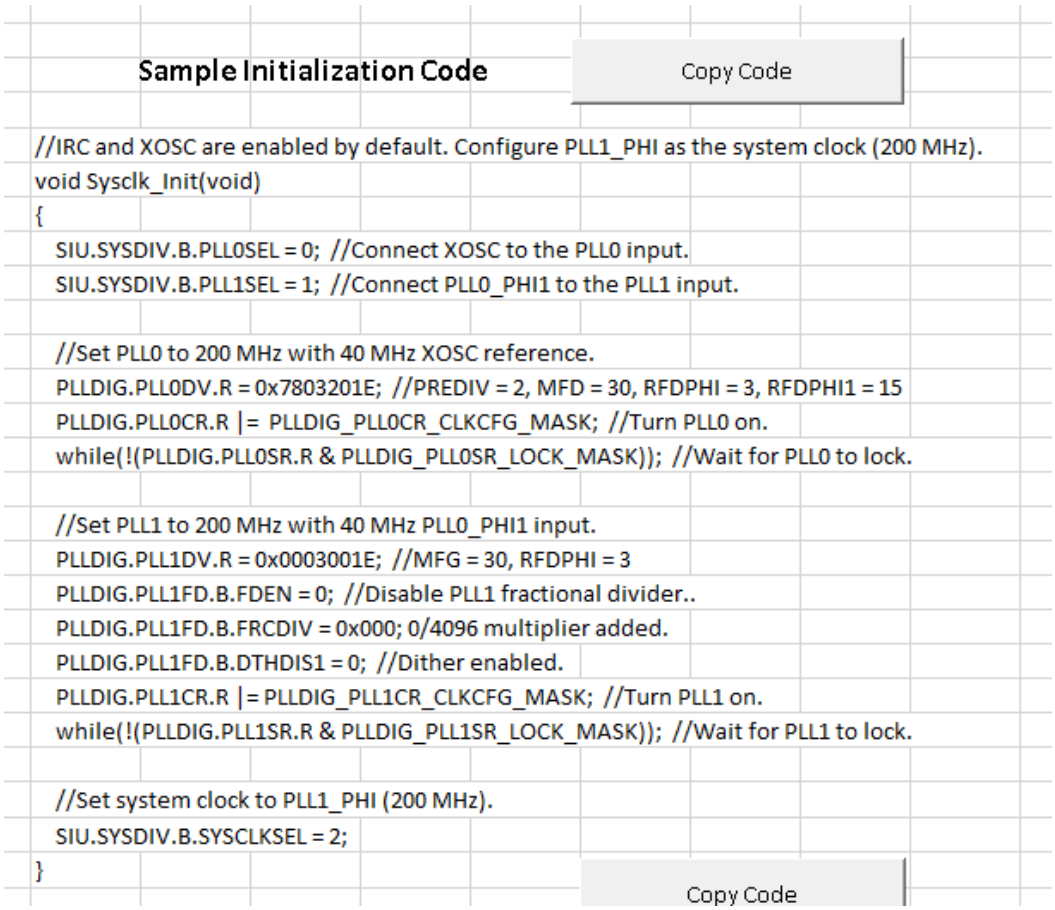


Figure 19. **Sample initialization code**

2.10. Limits

Limits is the reference tab for all the color-coding and other rules. The values in its tables are based on the MPC5777C's datasheet and reference manual and should not be modified by the user. The following figure is a screenshot of the *Limits* tab.

Do not change these numbers	
PLL0 Input (min)	8
PLL0 Input (max)	44
PFD (min)	8
PFD (max)	20
PLL0_VCO (min)	600
PLL0_VCO (max)	1250
PLL0_PHI (min)	4.762
PLL0_PHI (max)	200
PLL0_PHI1 (min)	38
PLL0_PHI1 (max)	78
PLL1 Input (min)	38
PLL1 Input (max)	78
PLL1_VCO (min)	600
PLL1_VCO (max)	1250
PLL1_PHI (min)	4.762
PLL1_PHI (max)	264

Clock Name	Max (MHz)	Min (MHz)
CORE_CLK	264	0
eTPU_CLK	200	0
PLAT_CLK	132	0
FM_PER_CLK	132	0
PER_CLK	132	0
EBI	66	0
SDCLK	16	4
eQADC ADCLK	33	2
MII	25	0
RMII	50	0
LFAST	26	10
DSPI_CLK	100	0
LFAST_PLL	240	10

Figure 20. **MPC5777C Frequency limits**

3. Clock tool example use case: Configure eMIOS to 60 MHz PLL1 with MPC5777C_264MHz

The following sections presents an example application of the MPC5777C Clock Calculator. This application note's example will configure eMIOS0's clocks to 60 MHz. The example will not only show the correct configurations but also how the tool responds if improper configurations are attempted.

When configuring clocks for a module, start at *Peripheral Domains*. As shown in the following figure, eMIOS0 follows only *PER_CLK* for both the bus interface unit and module clock.

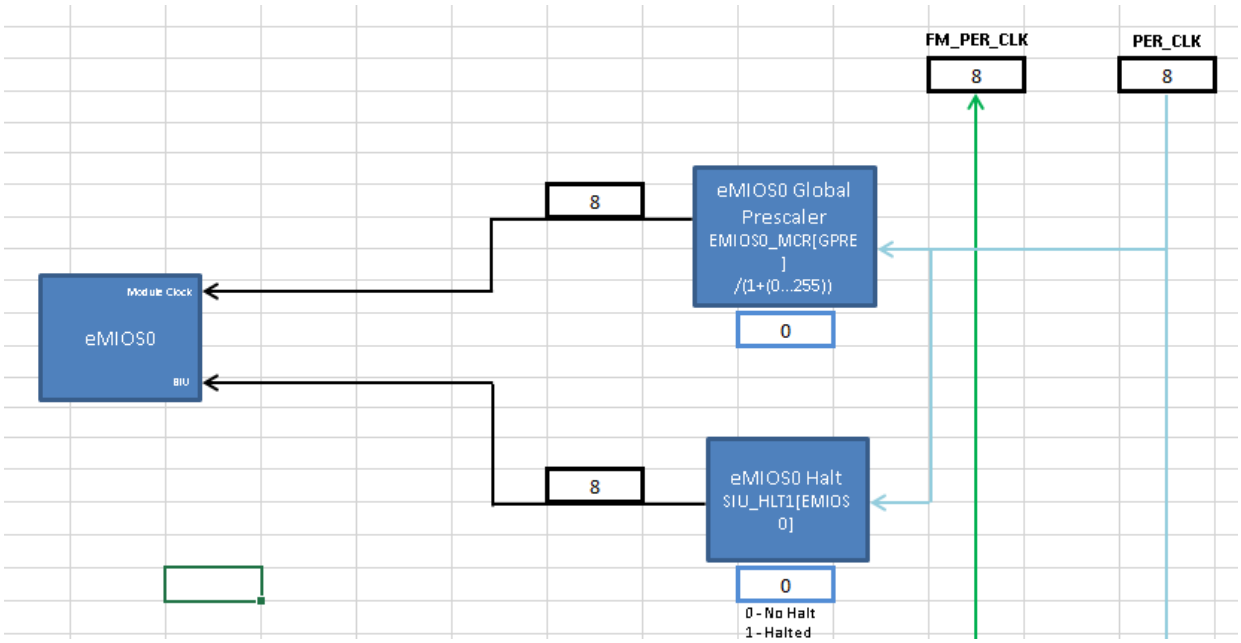


Figure 21. **eMIOS0 clocks**

PER_CLK is currently 8 MHz. If a module uses multiple clock domains, configuring the clock calculator can be done in any order.

3.1. Select the device

First choose the correct the device. Go to *Device Select* and set the device to *MPC5777C_264MHz*. This ensures that the MPC5777C Clock Calculator will use the spec sheet corresponding to the 264 MHz variant.

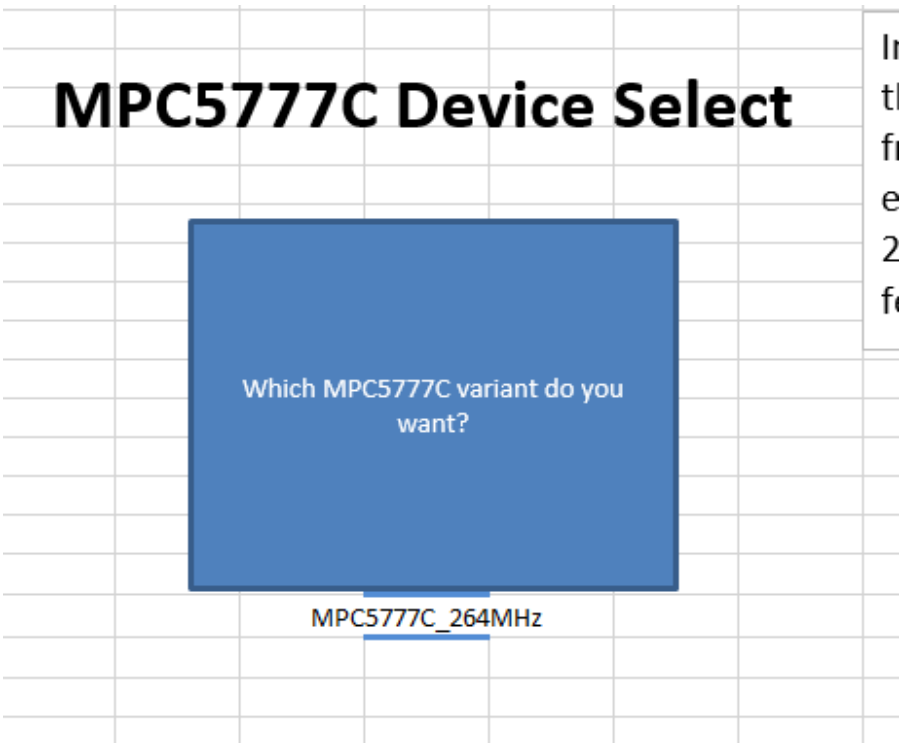


Figure 22. **Selecting MPC5777C_264 MHz**

3.2. Configure *PER_CLK*

Click on *PER_CLK* to forward to the *PER_CLK* cell of *Tree*, shown below.

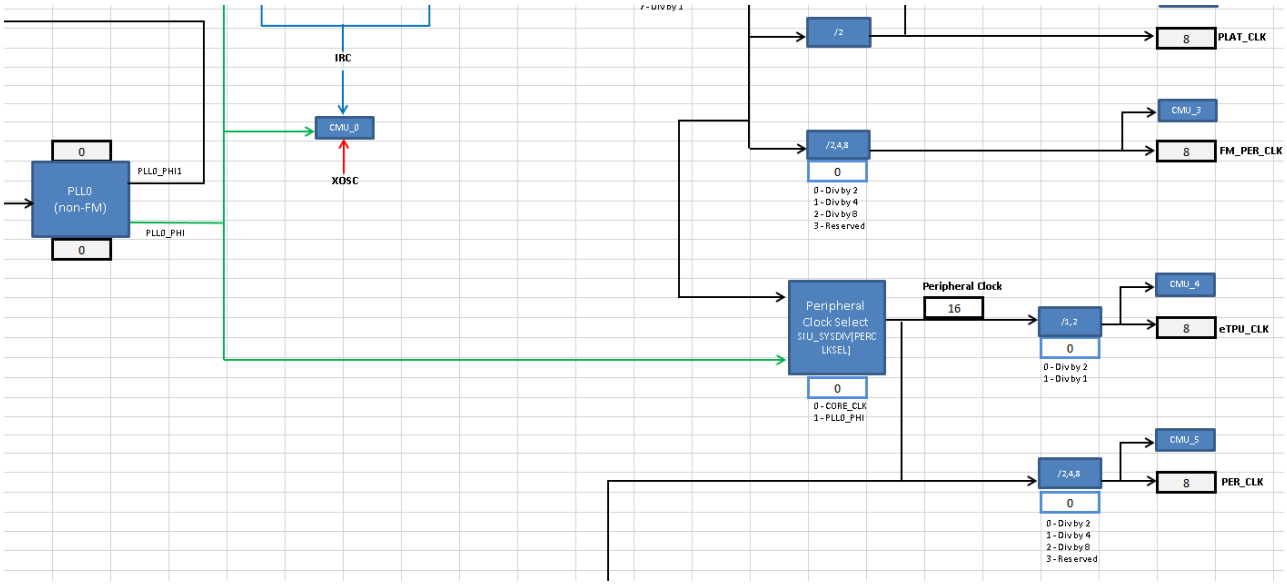


Figure 23. ***PER_CLK*, *Tree* tab**

Trace *PER_CLK* all the way back to its point of origin. As shown in the figure, *PER_CLK* is sourced from the 16 MHz *CORE_CLK* and divided by 2, hence 8 MHz. The cell is a drop-down menu and the

textbox explains what each available value represents. As shown in the figure, *CORE_CLK*, in turn, is currently sourced from the 16 MHz IRC, divided by 1, for a final frequency of 16 MHz.

Since the only way to achieve 60 MHz is through the PLL, one of the PLLs must be configured. This example will choose PLL1. Trace *PLL1_PHI* back to its own sources. PLL1 selects from either XOSC or *PLL0_PHI1* via *PLL1 Clock Selector*. These oscillators are the point of origin for all clock domains. The figure below shows *PER_CLK* being traced back to PLL1 and then finally to the oscillators.

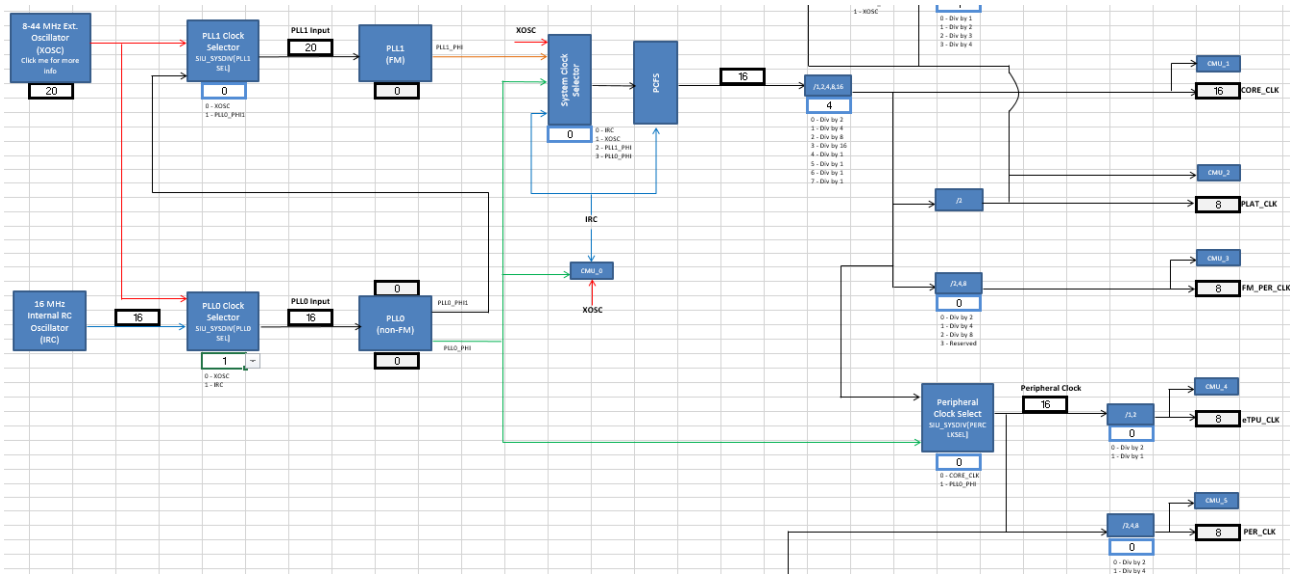


Figure 24. ***PER_CLK* to oscillators**

3.2.1. Configure the oscillator

Now start going downstream, configuring from the oscillator down to *PER_CLK*. The external oscillator frequency is application-dependent and can be any value between 8 MHz and 40 MHz. This tool has a safeguard to prevent invalid values from being entered. The Figure23 shows an attempt to enter 7 MHz to the XOSC frequency cell. A dialog box appears notifying the user that the value is not accepted when he/she tries to click away from the cell.

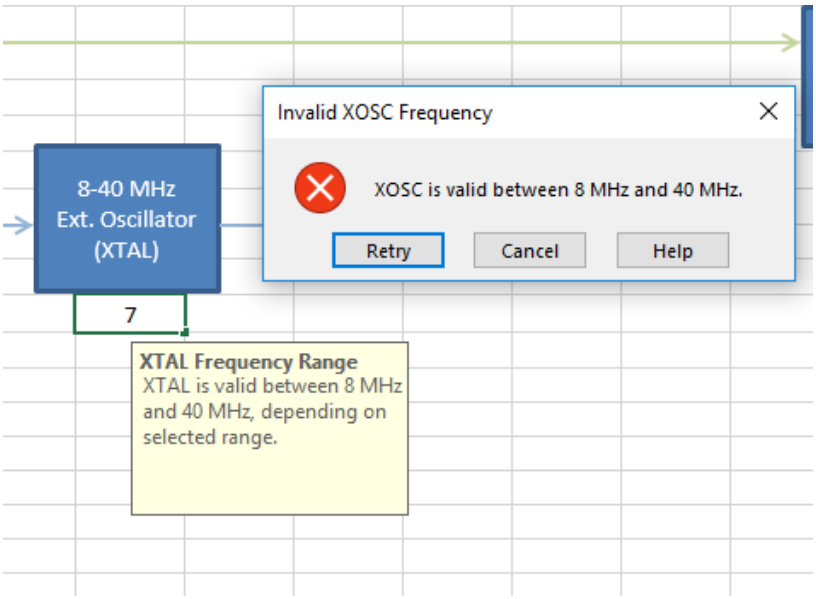


Figure 25. **Invalid frequency input**

Set MPC5777C to *DCF Client* mode, oscillator range to *High*, and XOSC frequency to 40 MHz. Next, set the value of the *XOSC Enable/Select* block to 1 to select XTAL, the external oscillator, as shown in the following figure.

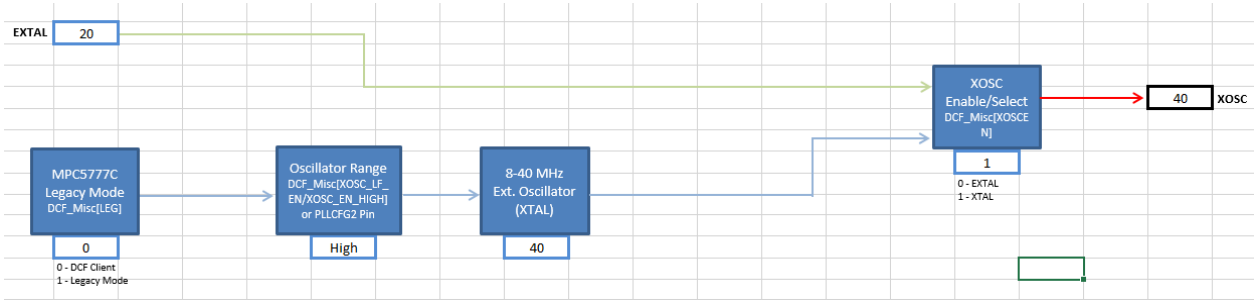


Figure 26. **Oscillator configuration**

3.2.2. Configure PLL0

Follow the XOSC path to *PLL0 Clock Selector*. Change the *PLL0 Clock Selector* value to 0, so that PLL0 sources from XOSC, as shown in the figure below.

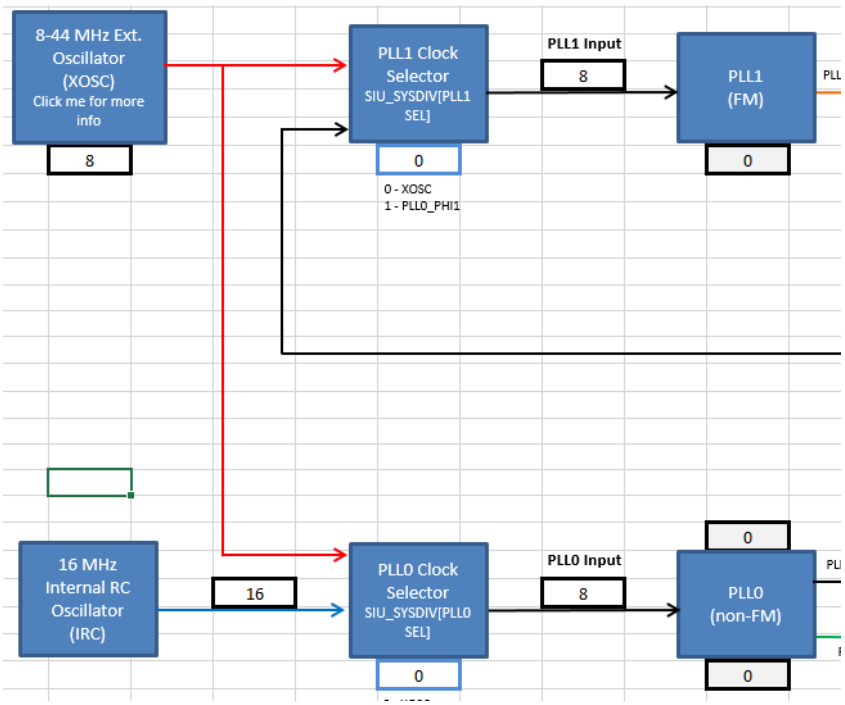


Figure 27. **PLL0 source to XOSC**

Next, configure PLL0. Click on the *PLL0* block to forward automatically to the *PLL0* tab. This is the tab that sets up the *PLL0_PHI* frequency. The *PLL0 Input* block of the figure below shows that PLL0 detects the 40 MHz XOSC as its source frequency.

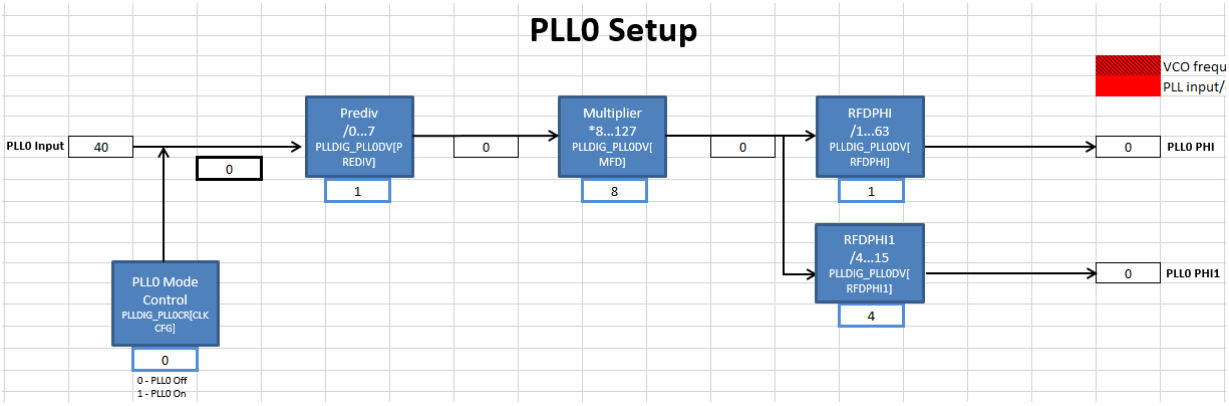


Figure 28. **PLL0 calculator**

Configure the dividers to achieve 200 MHz. The correct configuration can be achieved by trial and error, but the MPC5777C clock calculator provides a lookup table in the *pll0_phi* tab, as shown in the following figure.

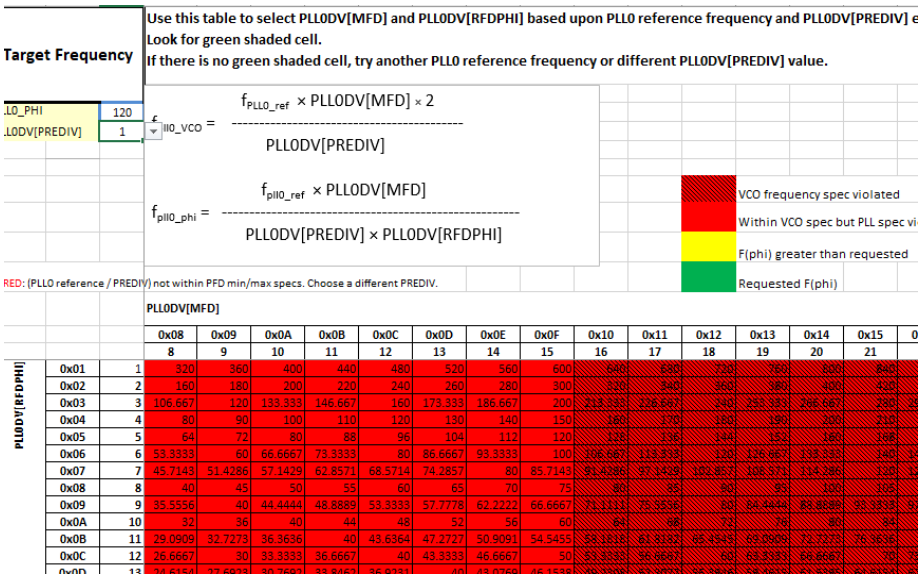


Figure 29. ***PLL0_PHI* reference table**

The *PLL0 reference* field is the frequency of the PLL0 input, in this case the 40 MHz XOSC. Set the target frequency and PREDIV values. This example will target 200 MHz and change PREDIV to 2. The values and shading in the lookup table will automatically change to fit these new settings. In the figure below, the table has changed and circled are the modified settings.

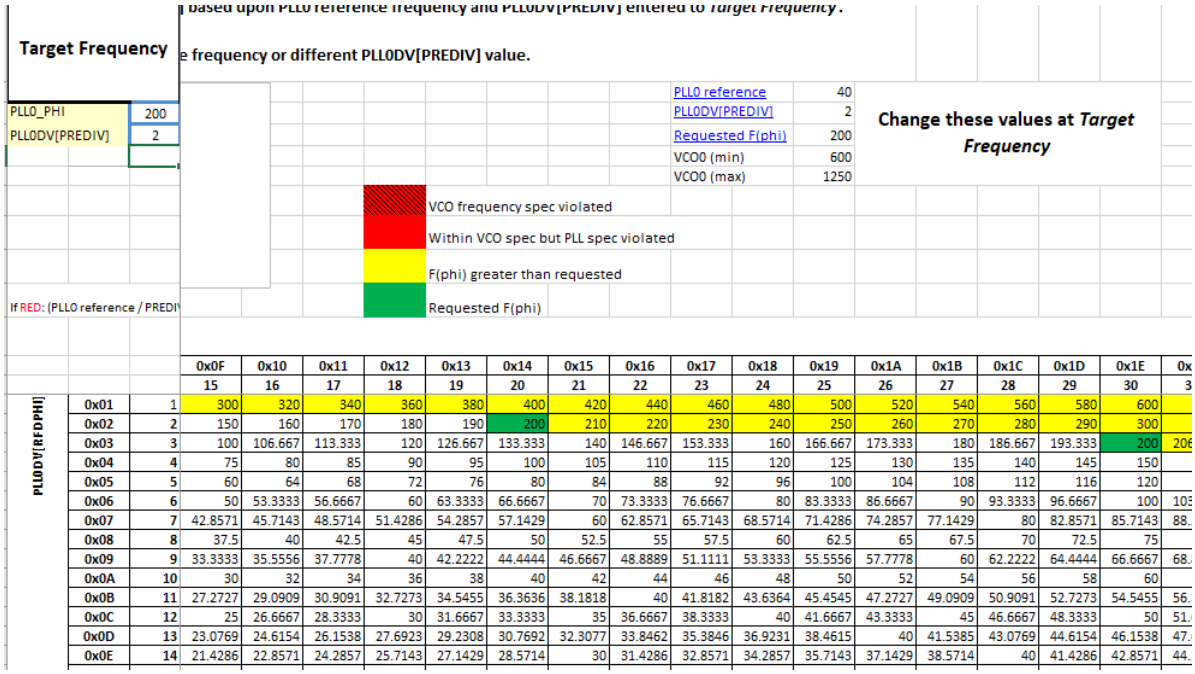


Figure 30. ***PLL0_PHI* table with new settings**

The cells shaded green means there are two divider combinations that can achieve exactly 200 MHz given an input frequency of 40 MHz and a PREDIV of 2. This example will use a MFD of 20 and a RFD of 2, but before configuring the *PLL0* tab, it is worth noting what happens if the output PLL frequency is out of range.

In the following figure, the PLL has been configured so that the output frequency is 5.08 GHz. This obviously exceeds the maximum hardware spec of 200 MHz. The associated voltage controlled oscillator (VCO) frequency, which can be back-calculated from *PLL0_PHI* also exceeds the maximum VCO spec of 1250 MHz. Therefore, the output is crosshatched and shaded red.

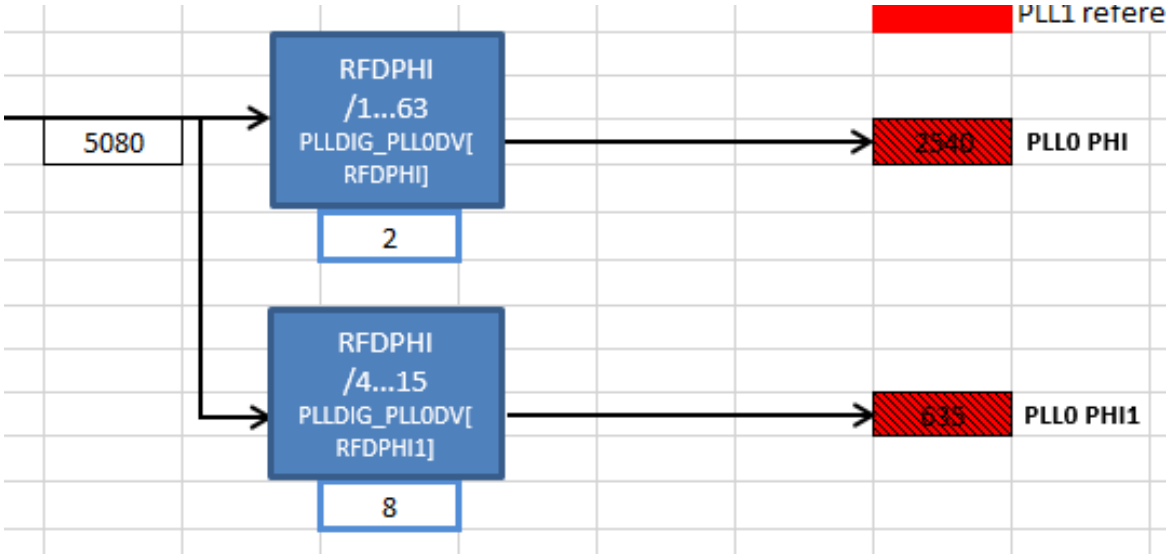


Figure 31. **When *PLL0_PHI* exceeds VCO and PLL spec**

Now let's configure the PLL correctly. Turn on the PLL in the *PLL0* tab by setting the *PLL0 Mode Control* block to 1, set *Prediv* to 2, *Multiplier* to 20, and *RFDPHI* to 2. As shown in the next figure, the output *PLL0_PHI* is 200 MHz and the cell remains unshaded, meaning the configuration fits within spec. Also, set *RFDPHI1* to 10 so that *PLL0_PHI1* is 40 MHz. This output will be the source of PLL1.

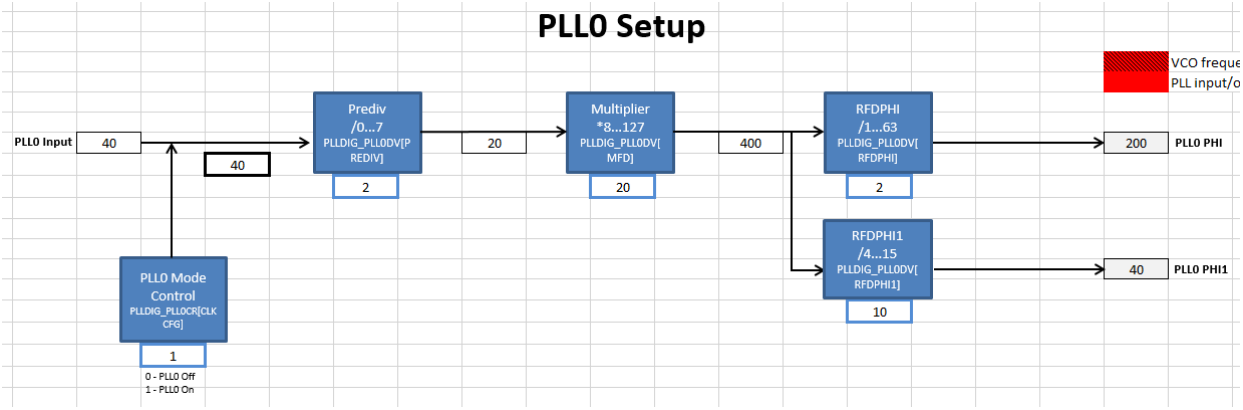


Figure 32. ***PLL0_PHI* configured to 200 MHz**

Go back to *Tree* to observe that the *PLL0_PHI* frequency is now 200 MHz and *PLL0_PHI1* is 40 MHz.

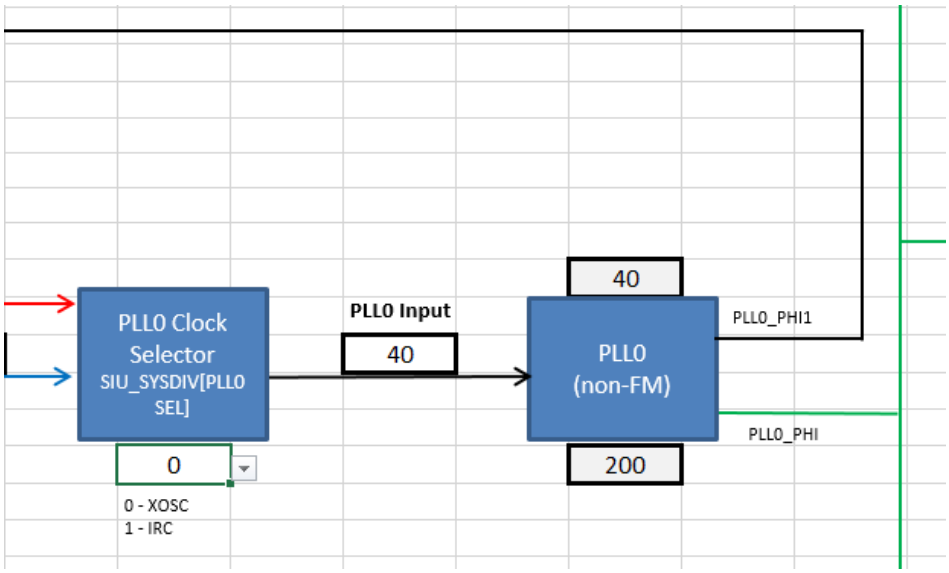
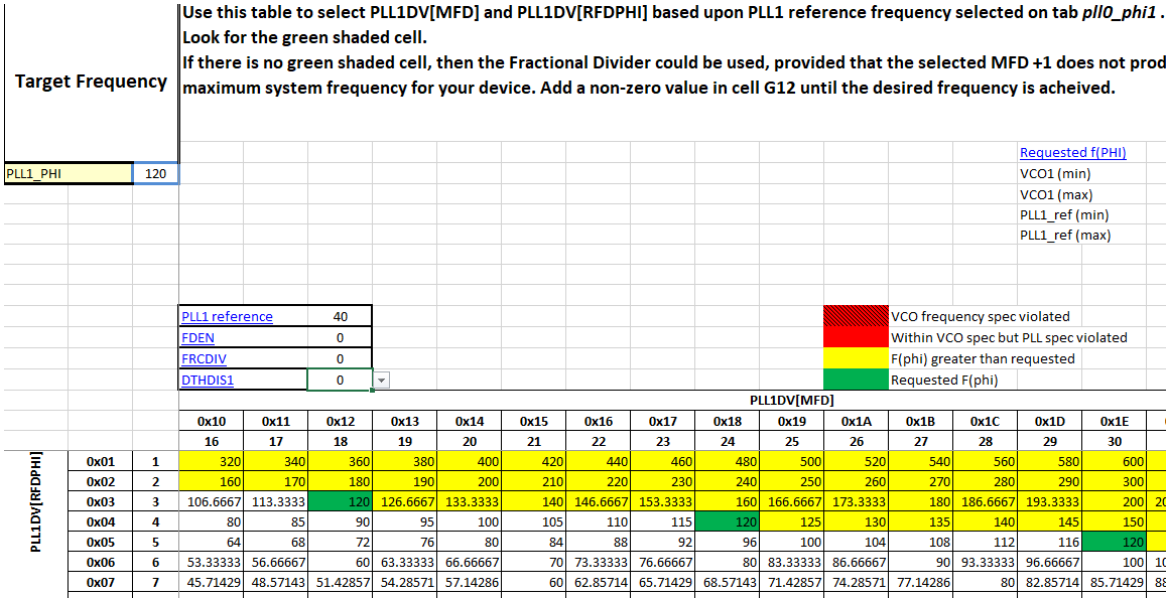


Figure 33. ***PLL0_PHI* propagated to *Tree***

3.2.3. Configure PLL1

Next, configure *PLL1_PHI* to 120 MHz using *PLL0_PHI1* as the source. Change the *PLL1 Clock Selector* block to 1 to select *PLL0_PHI1* as the source of PLL1. Much like the case with PLL0, you can use the *pll1_phi* reference table to find the proper dividers, as shown in the figure below.



Use this table to select PLL1DV[MFD] and PLL1DV[RFDPHI] based upon PLL1 reference frequency selected on tab *pll0_phi1* .
Look for the green shaded cell.
If there is no green shaded cell, then the Fractional Divider could be used, provided that the selected MFD +1 does not prod
maximum system frequency for your device. Add a non-zero value in cell G12 until the desired frequency is acheived.

Target Frequency

PLL1_PHI	120

Requested f(PHI)
VCO1 (min)
VCO1 (max)
PLL1_ref (min)
PLL1_ref (max)

PLL1 reference	40
FDEN	0
FRCDIV	0
DTHDIS1	0

VCO frequency spec violated
Within VCO spec but PLL spec violated
F(phi) greater than requested
Requested F(phi)

PLL1DV[MFD]

PLL1DV[RFDPHI]		0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
0x01	1	320	340	360	380	400	420	440	460	480	500	520	540	560	580	600
0x02	2	160	170	180	190	200	210	220	230	240	250	260	270	280	290	300
0x03	3	106.6667	113.3333	120	126.6667	133.3333	140	146.6667	153.3333	160	166.6667	173.3333	180	186.6667	193.3333	200
0x04	4	80	85	90	95	100	105	110	115	120	125	130	135	140	145	150
0x05	5	64	68	72	76	80	84	88	92	96	100	104	108	112	116	120
0x06	6	53.33333	56.66667	60	63.33333	66.66667	70	73.33333	76.66667	80	83.33333	86.66667	90	93.33333	96.66667	100
0x07	7	45.71429	48.57143	51.42857	54.28571	57.14286	60	62.85714	65.71429	68.57143	71.42857	74.28571	77.14286	80	82.85714	85.71429

Figure 34. ***PLL1_PHI* table for 120 MHz**

There are three settings that achieve exactly 120 MHz. This example will use an MFD of 18 and an RFDPHI of 3. In the PLL1 tab, turn on the PLL and enter these settings much in the same manner as for PLL0, as seen in the following figure.

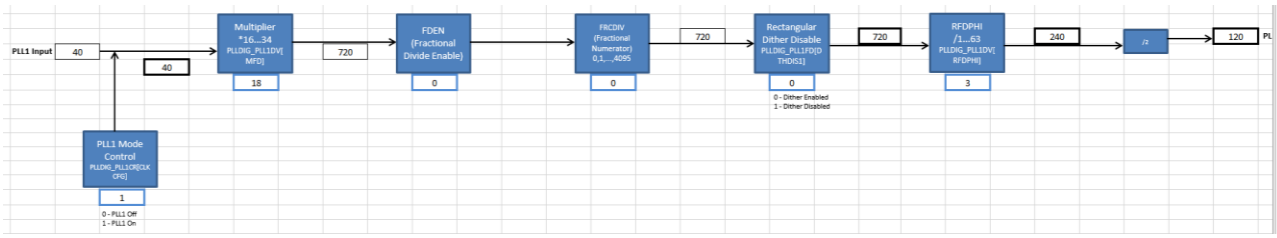


Figure 35. **PLL1 at 120 MHz**

3.2.4. Finish Setting *PER_CLK*

With the PLLs properly configured, change the system clock source to *PLL1_PHI* by changing the *System Clock Selector* value to 2.

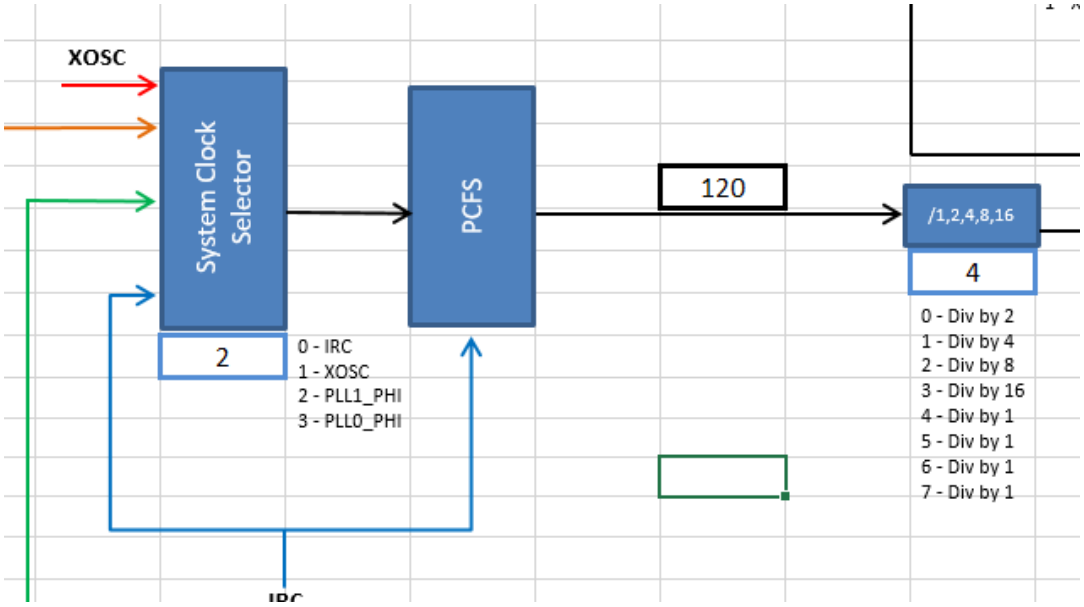


Figure 36. **System clock following PLL1_PHI**

The *PCFS* block stands for *Progressive Clock Frequency Switch*. This is a feature supported in the MPC5777C to smooth the transition of the system clock from one clock source to another. The block here is just a visual representation for the user to know that the system clock filters through the progressive clock switch before propagating to the various system clock domains. *PCFS* takes IRC in this diagram because its logic is organized in terms of IRC cycles. You can find more information on the progressive clock switch in the application note AN5304. The linked application note explains how to configure the MPC574xP, but its general principle can be extrapolated to the MPC5777C.

Set the system clock divider to divide by 1 so that *CORE_CLK* is 120 MHz. There are multiple bitfield values that does the job. This example will use the MPC5777C's reset value of 4. See the following figure.

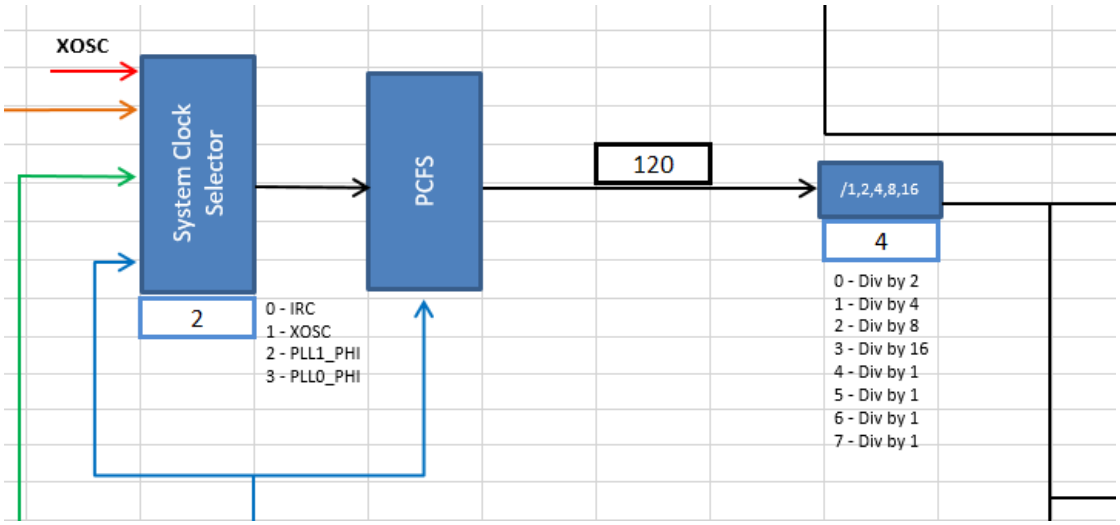


Figure 37. ***CORE_CLK* at 120 MHz PLL1**

Follow *CORE_CLK* down to the *Peripheral Clock Select* block and make sure its value is 0, so that *CORE_CLK* is selected.

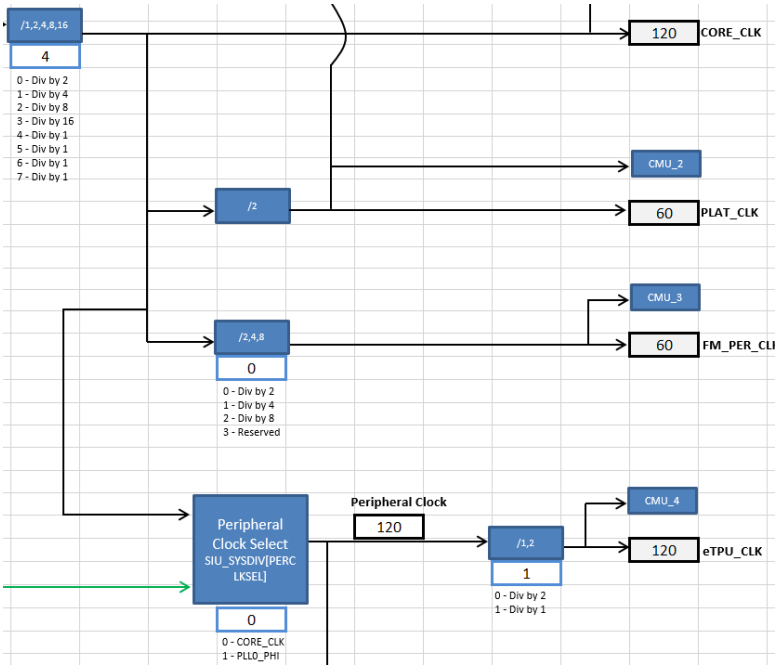


Figure 38. ***CORE_CLK* selected as peripheral clock source**

Navigate down to *PER_CLK* and set its divider field to 0, which corresponds to a divide by 2. Hence *PER_CLK* has been configured to 60 MHz.

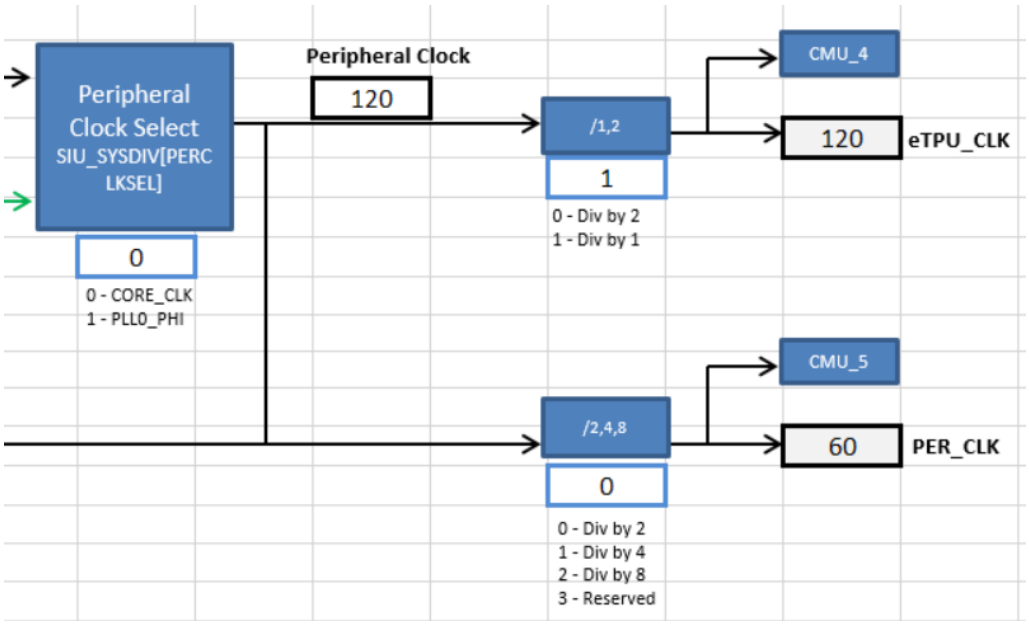


Figure 39. ***PER_CLK* at 60 MHz**

Return to the *Peripheral Domains* tab to verify that eMIOS0 is running at 60 MHz.

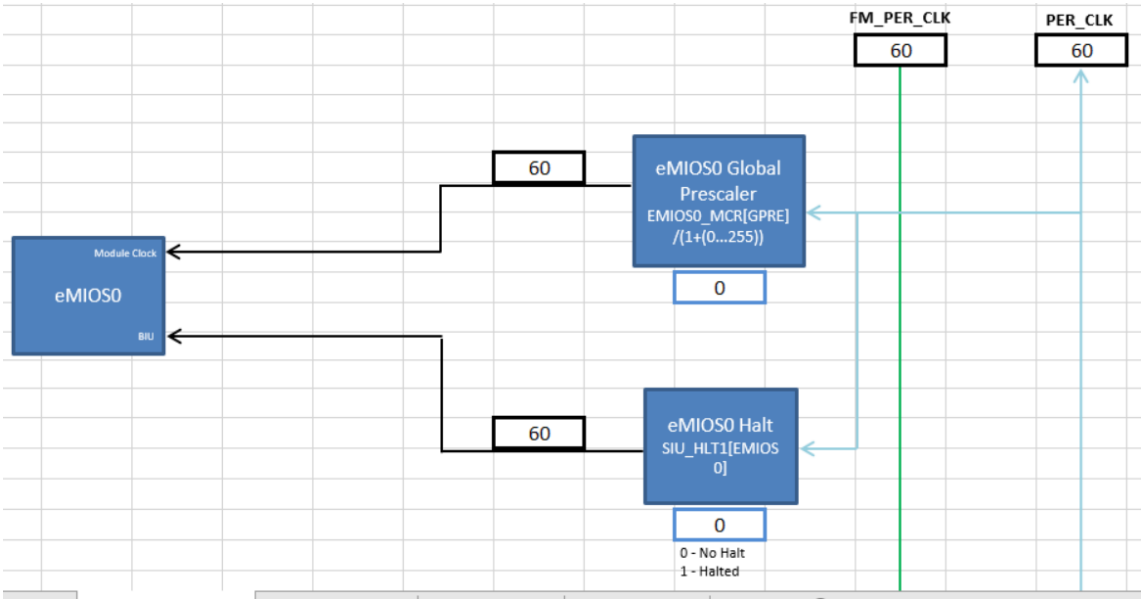


Figure 40. **eMIOS at 60 MHz**

Note that if a clock domain exceeds its hardware spec, the corresponding box will be shaded in red. As shown in the following figure, the max *SD_CLK* frequency is 16 MHz. Since 120 MHz, exceeds its limit, its cell is shaded red.

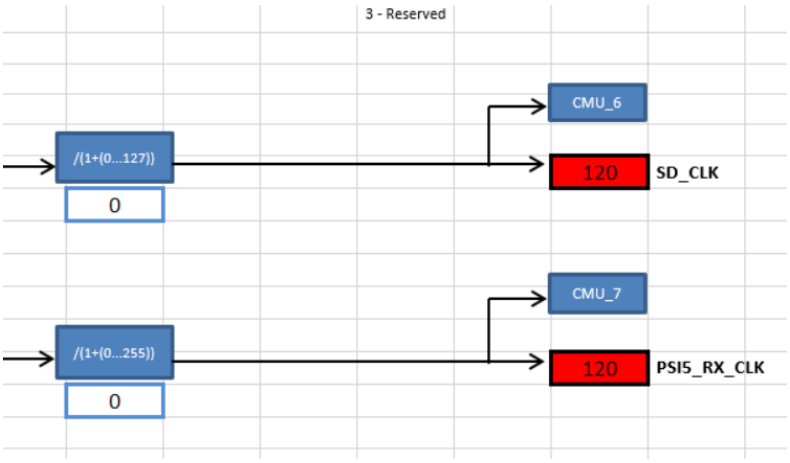


Figure 41. **Clock domain exceeds spec**

3.3. Observe the registers

The final register summary table, as displayed in *Summary*, is shown in the following figure. Note that most of these registers would not have to be written in code to achieve the setup that this example just configured. For example, the register SIU_LFCLKCFG would not have to be included, since the LFAST was untouched. Registers that would have to be written would be ones like PLLDIG_PLL0DV and SIU_SYSDIV.

Register Summary

Register	Value
DCF Start Record	0x05AA55AF00000000
DCF Misc Record	0b0XXXXXXX0000XXXX0X00010XXXXXXX00000000100000000000000000000000
DCF Stop Record	0x0000000000000001
SIU_SYSDIV	0bX00000000000000000100001000 1000X
SIU_ECCR	0b00000000000000000000100000000X001
SIU_SDCLKCFG	0xX0000000
SIU_PSCLKCFG	0xX0000000
SIU_LFCLKCFG	0xX0000000
SIU_PCR210	0xXXXX
SIU_PCR474	0b000XXXX0XXXXXXXX
SIU_FECCR	0x00000000
SIU_HLT1	0x40000000
SIU_HLT2	0x00000000
eMIOS0_MCR	0xXX0X0000
eMIOS1_MCR	0xXX0X0000
STM_CR	0x0000000X
CSE_CR	0x000000XX
STCU2_CFG	0b0XXXXXXXXXXXXXXX0000000X000XX000
STCU2_LB_CTRLn	0bXXXXXXXX0000X000XXXXXXX0000XXXX
REACM2_MCR	0bXXX0X00XX00000000000000000000000
REACM2_TCR	0x00000000
REACM2_PCR	0x0000XXXX
IGF0_PRESR0	0x00000000

Figure 42. **Register summary after configuration**

3.4. Copy the code

Sysclk_Init and *InitPeriClkGen* provide dynamic clock generation C code. The code will configure the clocks to the settings as configured in this clock calculator. It can be copied and pasted to a source file. The following figure shows *Sysclk_Init* as configured by this example. The solid-bordered highlight around the function means that the code has been copied with the *Copy Code* button; a regular Ctrl+C causes a dashed-bordered highlight. In both cases, the code can be pasted into a source with a regular Ctrl+V.

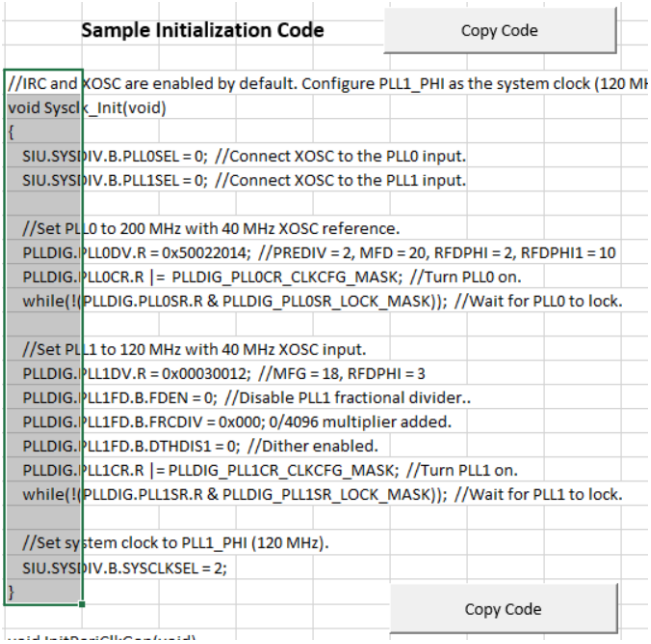


Figure 43. ***Sysclk_Init* after example**

To summarize, this example has achieved its goal: an eMIOS module whose bus interface and module clocks derive from a 60 MHz *PER_CLK*. *PER_CLK* is in turn sourced from the *CORE_CLK*, which comes from the 120 MHz *PLL1_PHI*. *PLL1_PHI* itself comes from PLL0, which are then ultimately sourced from a 40 MHz external oscillator.

4. Conclusion

This application note gives an overview of the MPC5777C interactive clock calculator. It seeks to simplify clock configurations in the form of a graphical tool so that a user can more easily visualize the device's clock signals' propagation. There are similar clock calculators for other NXP products, including the MPC574xG and S32K14x. Visit the NXP website to find more of these tools.

5. Revision History

Revision Number	Date	Substantive changes
0	04/2018	Initial release

Revision Number	Date	Substantive changes
1	12/2018	1. Extensively updated Introduction 2. Updated Figure 2 3. Added the following new sections: a. Device select b. Select the device 4. Updated Figure 15 5. Changed Section 3 heading to "Clock tool example use case: Configure eMIOS to 60 MHz PLL1 with MPC5777C_264MHz"

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN12176
Rev. 1
12/2018