恩智浦半导体
应用笔记

文档编号：AN5408
第6版，2018年9月

S32K1xx 时钟计算器指南

如何使用S32K1xx工具轻松计算各个频率

作者：恩智浦半导体

1 简介

S32K1xx 是恩智浦面向汽车和工业应用的 32位通用 MCU 系列。我们的产品结合了最新的 90 纳米技术，因此客户不必牺牲性能来换取低功耗。S32K1xx 基于ARM Cortex-M4® 架构，运行频率高达 112MHz。该器件系列包含两个子系列：S32K14x 和 S32K11x。S32K14x 系列注重于性能，包括器件S32K142、S32K144、S32K146 和 S32K148；而 S32K11x（S32K116 和 S32K118）是低成本子系列，适用于低价格功能要求相对不多的用户。为简单起见，本应用笔记将 S32K1xx 系列称为“S32K”。

目录

该器件支持四个时钟振荡器，在 S32K14x 中，支持一个系统锁相环（SPLL），总共最多五个时钟源。还有多个输入引脚，通过这些引脚可以将外部时钟信号送到 MCU 中。在四个振荡器中，有一个系统振荡器 （SOSC）、一个 48MHz 快速内部 RC 振荡器（FIRC）、一个 2-8MHz 慢速内部 RC 振荡器（SIRC）和一个 128kHz 低功耗振荡器（LPO）。SOSC 可以来自 EXTAL 引脚信号或连接到 XTAL 和 EXTAL 引脚的晶体振荡器（以下简称“XTAL”）。EXTAL 最高可支持 50MHz，而根据配置，XTAL 允许有两个范围：4-8MHz 或 8-40MHz；FIRC 可以调整到 48MHz；SIRC 可以是 2MHz 或 8MHz。此外，S32K14x 器件上的 SPLL 支持 90MHz 至 160MHz 的频率。有关摘要，请参阅下表。

表1. S32K时钟频率

时钟源	允许的频率
FIRC	48 MHz
SIRC	可选：2 MHz或8 MHz
LPO	128 kHz
SPLL (仅限S32K14x)	90-160 MHz
SOSC	可选：XTAL或EXTAL
XTAL	可选范围：4-8 MHz或8-40 MHz
EXTAL	最高50 MHz

时钟设置是几乎所有应用中的必要步骤。S32K 时钟计算器旨在通过提供图形化的交互式工具来补充参考手册中的配置说明，以帮助用户找到正确的寄存器配置，以实现所需的时钟频率。

本应用笔记附带时钟计算机。您可以从 S32K1xx_Clock_Calculator 下载它。

时钟计算器使用宏来执行功能，例如将电子表格重置为初始值、将所有时钟频率配置为最大允许设置以及复制生成的代码。用户必须在MS Excel 中启用宏才能访问这些功能。如果宏被关闭，该工具仍然能够计算时钟频率，但上述功能将被禁用。要在 MS Excel 2016 中打开宏，请转到顶部工具栏上的“开发人员”选项卡，然后单击“宏设置”。将出现一个弹出窗口。在其中，选择启用所有宏。

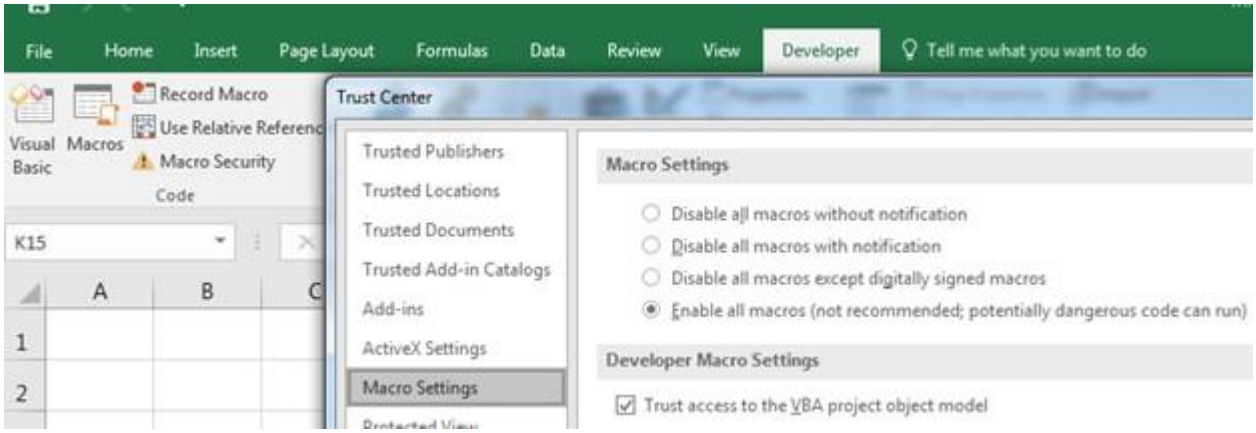


图1. 启用宏

2 时钟计算器设计

S32K 时钟计算器采用交互式 Microsoft Excel 电子表格的形式，组织成多个选项卡，如下图所示。

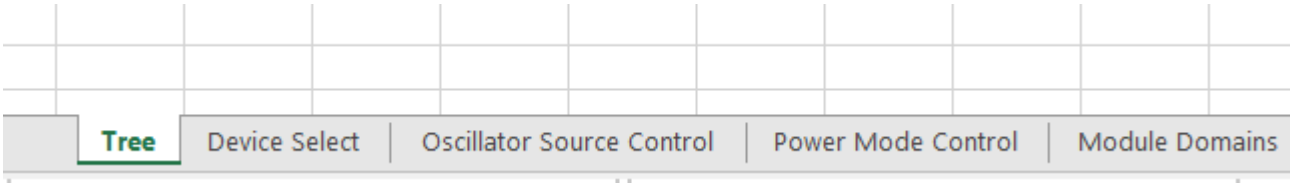


图2. S32K1xx 时钟计算机设置

时钟源（例如振荡器、SPLL、外部输入引脚）输入到MCU的不同时钟域，从而为MCU的各个模块提供时钟。代表时钟域频率的大多数单元不需要手动修改。用户只需为少数选定时钟源输入频率，并且所有时钟域频率均源自这些时钟源。几个时钟域的输入需要手动修改，因为它们代表输入到芯片中的外部时钟。还有用于设置多路复用器和时钟分频器的输入单元。所有用户可以输入的单元格都是蓝色边框而不是黑色边框，如下所示。需要输入的块中还显示了其代表的寄存器字段。

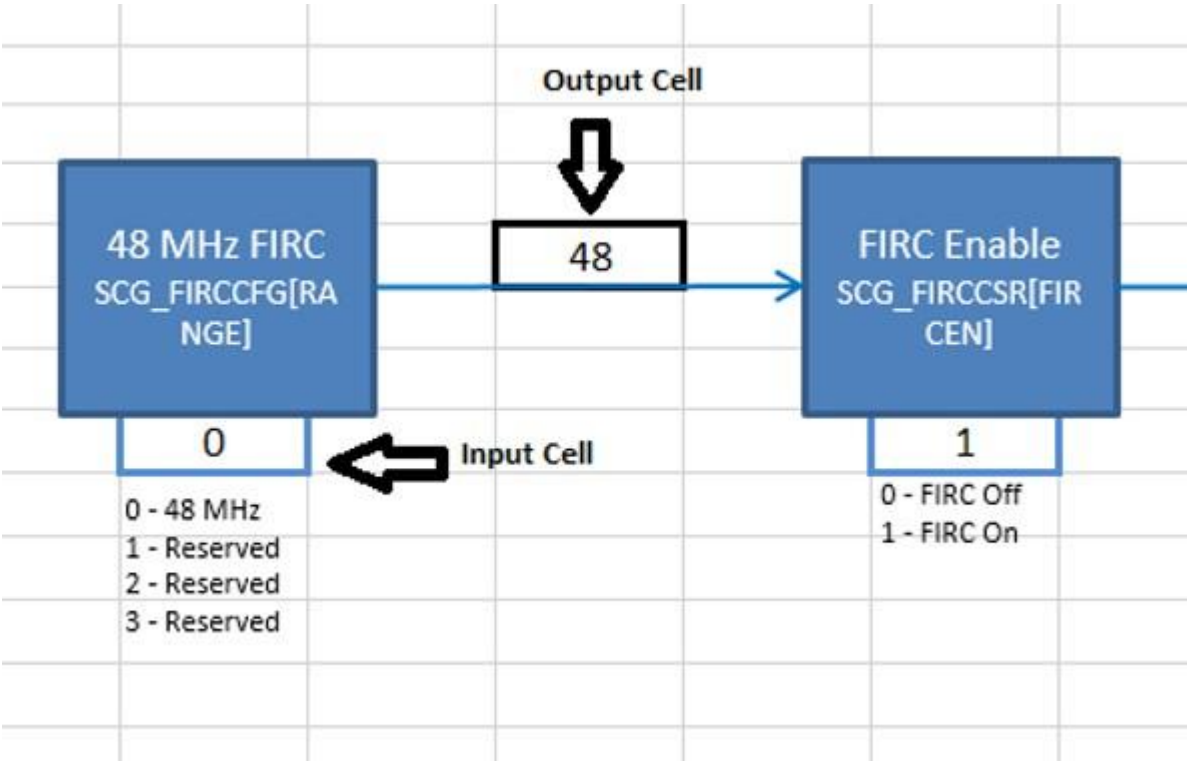


图3. 输入单元与输出单元

可以将哪些频率输入到输入频率单元格是有限制的。超出范围的值将被拒绝，用户将收到一条错误消息。由有效输入值和合法但不正确的分频器产生的无效时钟域频率将用红色阴影表示。本应用笔记稍后将对此进行更深入的解释。

频率值会跨选项卡链接，因此树选项卡中的 BUS_CLK 将始终与模块域选项卡中的 BUS_CLK 相同。同样的时钟域名会提供超链接以链接回其来源。例如，BUS_CLK 起源于Tree。因此，单击 Module Domains 中的 BUS_CLK 文本框会将用户带到 Tree 中的 BUS_CLK。作为链接的文本框，当鼠标悬停在上面时，会使鼠标光标变成手形图标并弹出一个显示目的地地址的弹出窗口，如下图所示。

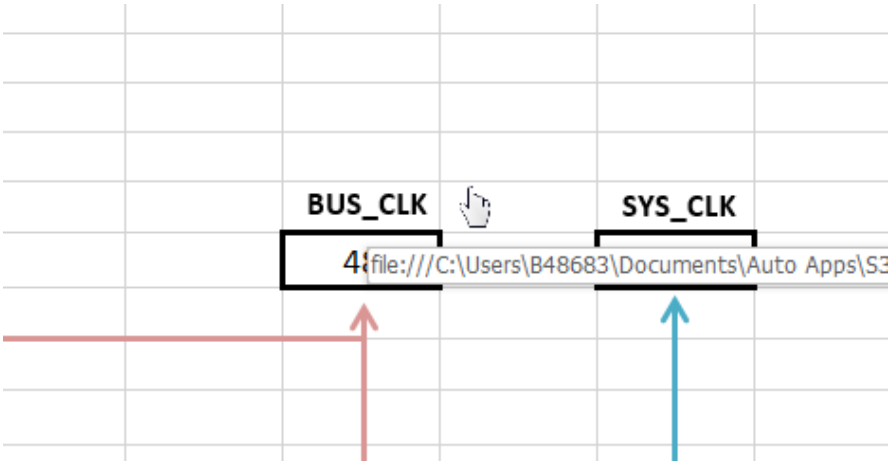


图4. 单击链接

以下小节将深入解释每个选项卡的用途。

2.1 树状图

树是该工具的核心。此选项卡是所有时钟频率计算的起点。它的组织方式类似于 S32K 时钟树，如下图所示。

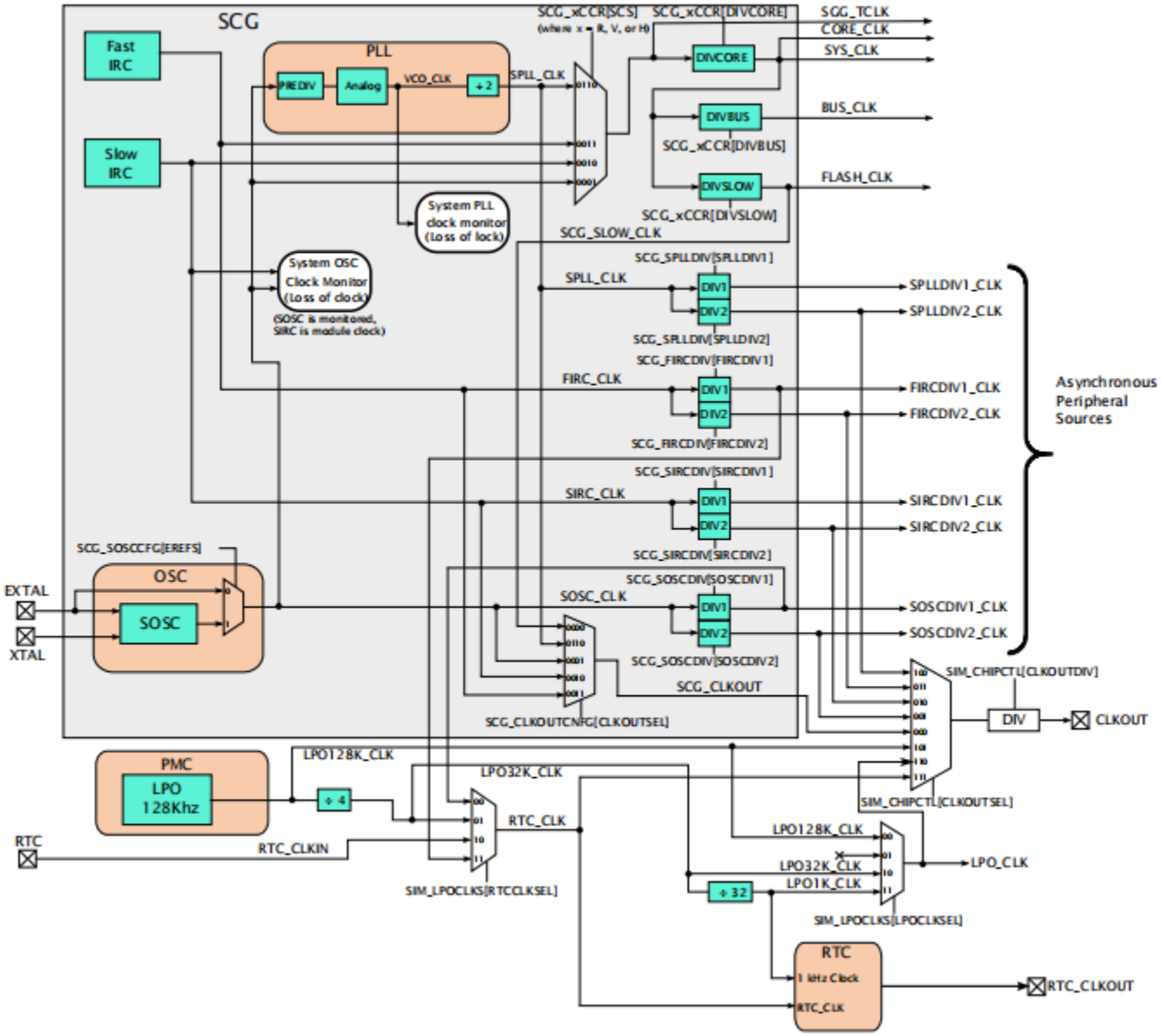


图5. S32K参考手册时钟树状图

图 5 部分，显示了图表的时钟工具副本。对树状图进行了添加，以反映参考手册图形中未显示的细微差别。为简单起见，参考手册图形仅显示基本功能。该工具将所有时钟选项整合到一个平台中。

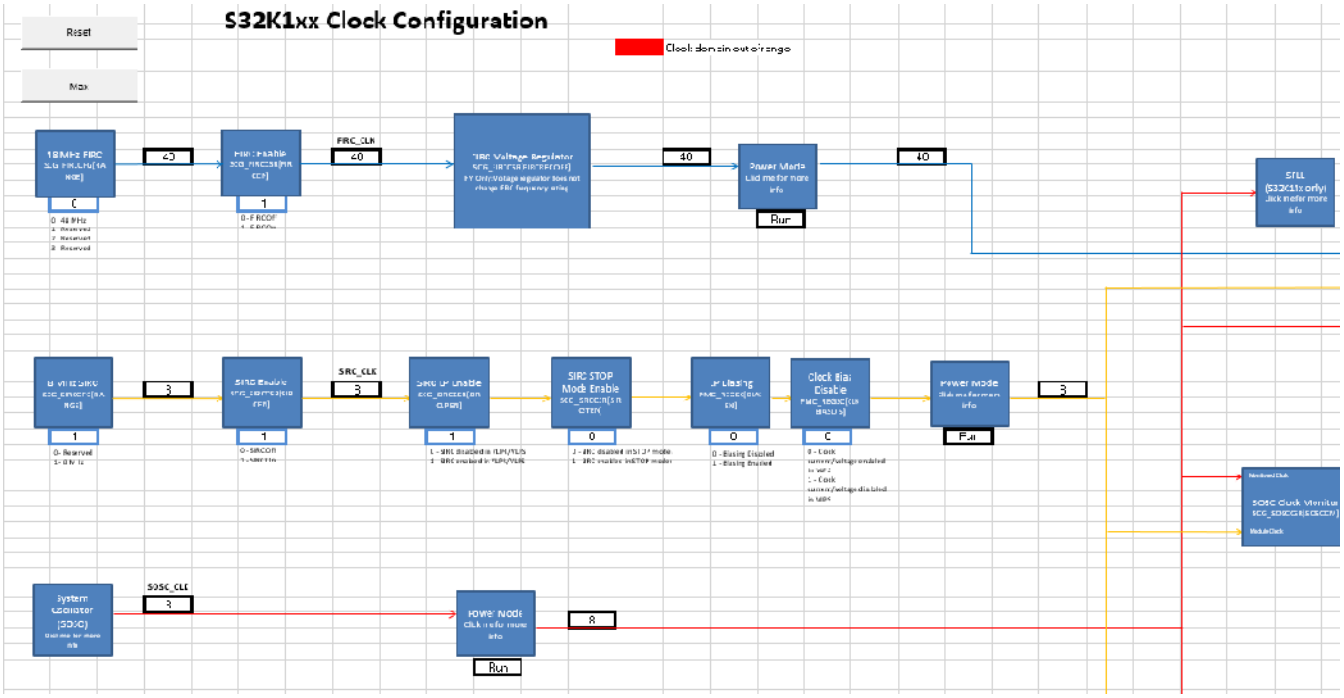


图6. 时钟计算器树状图

这个工具的版本显然比参考手册中的要复杂得多。事实上，屏幕截图只能合理地显示图表的左上部分。图的流程一般从左到右。左边是 S32K 时钟源，右边是时钟域。MCU 各个模块在这些时钟域的一个或多个上运行。

时钟域频率值显示在其标签旁边的空心单元格中。大多数单元不需要写入，它们的值取决于时钟树中前面步骤计算的频率。以 BUS_CLK为例：它的值取决于系统电源模式、内核时钟分频器、系统时钟选择器以及系统时钟选择器选择的源选择控制器。系统时钟选择器可以选择 SOSC、SIRC、FIRC 或 SPLL 的输出。现在看看其中一个源，FIRC。FIRC 被调整到 48MHz，但它的频率是否分配下去取决于下一个块，FIRC Enable。因此，以 FIRC 为源的模块接收到的实际输入频率是 48MHz 的 FIRC 频率，由 FIRC Enable 决定。SOSC、SIRC 和 LPO 也是如此。SPLL 输出的在 SPLL 选项卡中配置。BUS_CLK 通过配置系统时钟选择器模块的值从这四个时钟源中进行选择。然后最终选择的信号除以内核时钟预分频器值，并由系统模式决定。

此选项卡还有两个按钮，重置和最大。它们仅在启用宏时起作用。在禁用宏的情况下单击这些按钮将返回错误。如果启用了宏，则重置按钮会将所有块设置为其重置值，如参考手册中所述。Max 按钮将此工具中的所有模块设置为将系统和辅助时钟域配置为其各自最大允许频率的值。下面是按钮的屏幕截图。

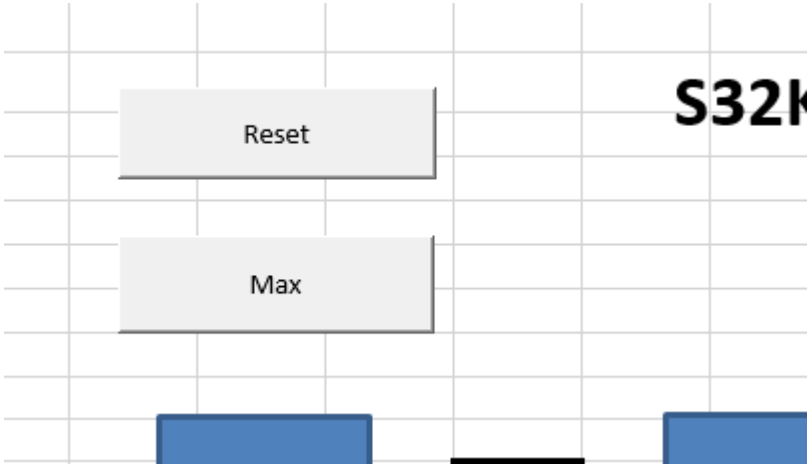


图7. 按钮

2.2 设备选择

设备选择选项卡在两个 S32K 子系列之间进行选择。与 S32K14x 相比，S32K11x 缺少 SPLL、HSRUN 电源模式和几个模块。由于此工具可配置全功能的 S32K14X，因此选择 S32K11X 时，S32K14x 的独有功能会关闭。这意味着 SPLL 输出将设置为 0 并且不能用作时钟源，如果选择了 HSRUN 电源模式，电源模式块将显示为红色，并且仅限 S32K14x 的外设也将其时钟归零。

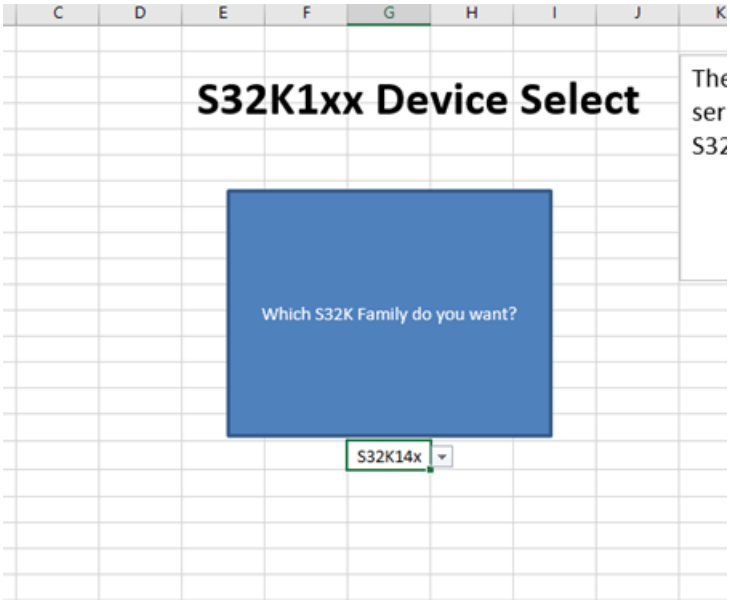


图8. S32K设备选择

2.3 振荡器源控制

S32K 的外部振荡器有单独的选项卡来配置。这些功能反映在 S32K 时钟计算器的'振荡器源控制选项'卡中。振荡器源控制包含 SOSC 和 LPO 的选项。下面是该选项卡的屏幕截图。

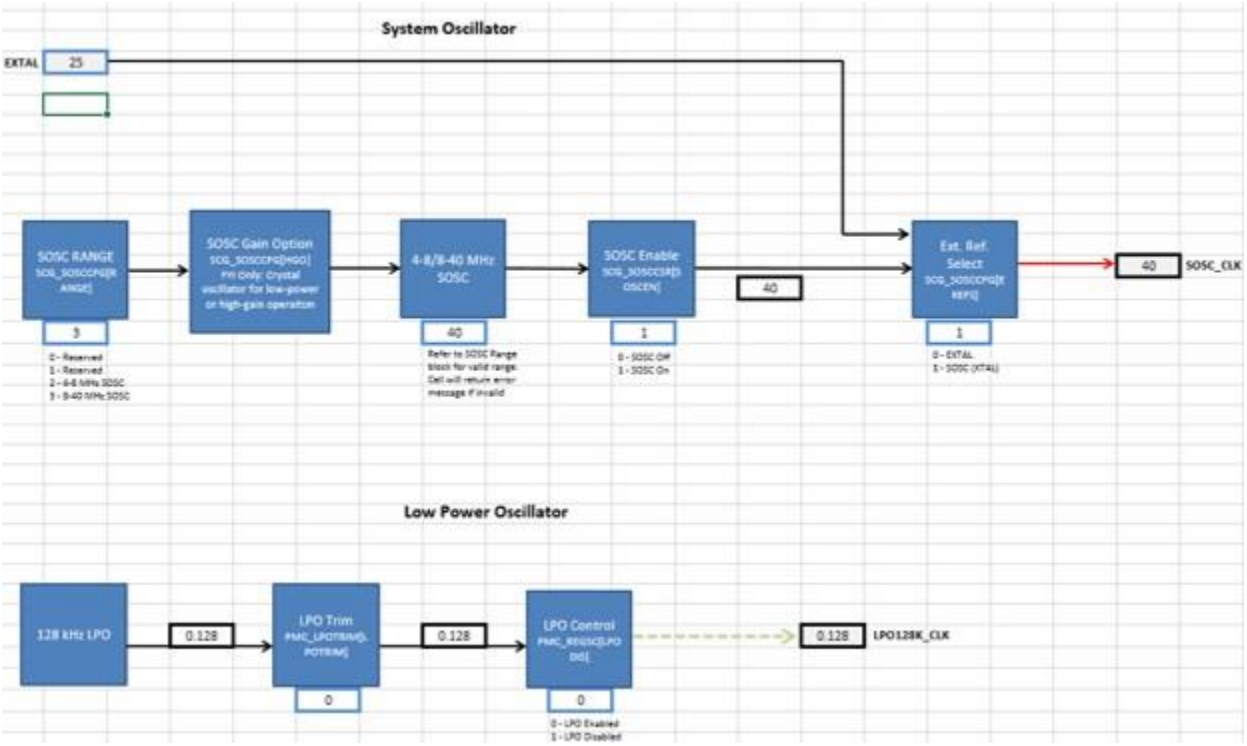


图9. 振荡器源控制

对于系统振荡器，此选项卡提供用于选择频率范围、启用/禁用振荡器以及在 XTAL 和 EXTAL 之间进行选择的选项。LPO的 控制允许频率校准，其额定值为 128kHz，但可以在 113kHz 和 139kHz 之间变化。

2.4 功耗模式控制

由于许多时钟域受 S32K 系统功耗模式的影响，因此功耗模式控制选项需要有自己的选项卡。下图显示了功率模式控制表。

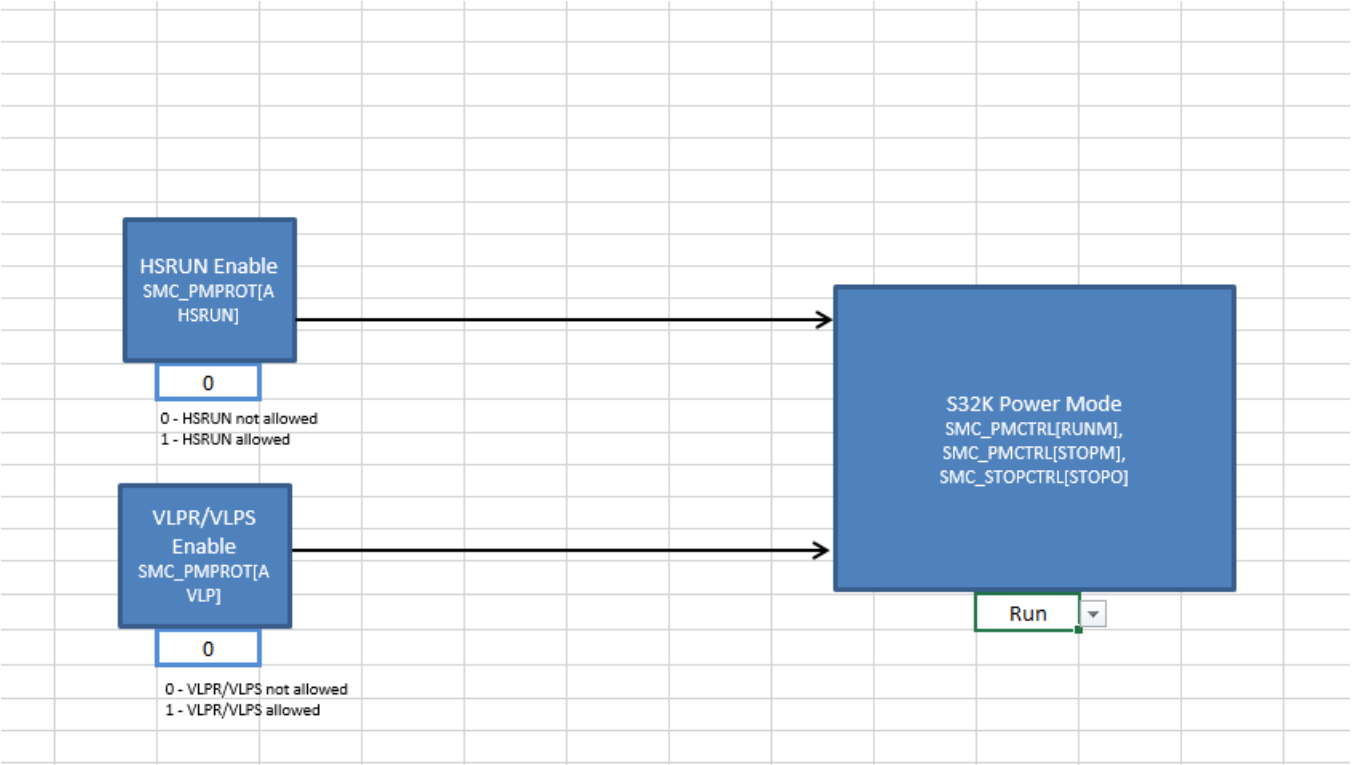


图10. S32K 功耗模式控制

HSRUN 及低功耗模式VLPR 和 VLPS 需要在它们各自的块中启用，如上，反映 了S32K 电源管理设计。S32K Power Mode 的选项列表将根据 HSRUN Enable 和 VLPR/VLPS Enable 的设置而改变。请注意，S32K11x 缺少 HSRUN 模式，因此如果在 Device Select 选项卡中选择了 S32K11x，则 HSRUN Enable 块无效，并且 HSRUN 将不可用。

2.5 模块域

模块域选项卡是 S32K 各个模块时钟的进一步表示。Tree 实在在时钟域的级别，Module Domain 选项卡开始进入各个模块的时钟级别。下图显示了模块域的屏幕截图。

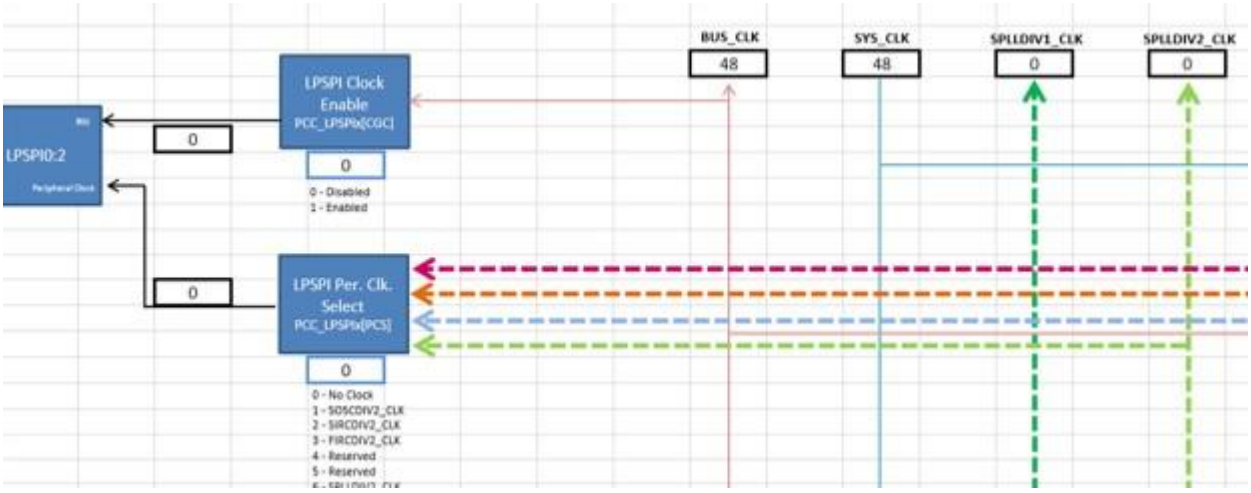


图11. 模块域

时钟域采用颜色编码。黑线为本地时钟节点保留。例如，BUS_CLK 分支到 LPSPI，但通过 LPSPI 时钟使能模块使能。LPSPI 时钟使能模块后的箭头颜色更改为黑色，表示与该黑线关联的频率值仅适用于 LPSPI。一般来说，如果所有使用时钟域的模块都可以容纳在一个窗口中而无需滚动，则时钟域用黑线表示。

2.6 SPLL

SPLL是SPLL数字接口的可视化抽象，如下图所示。

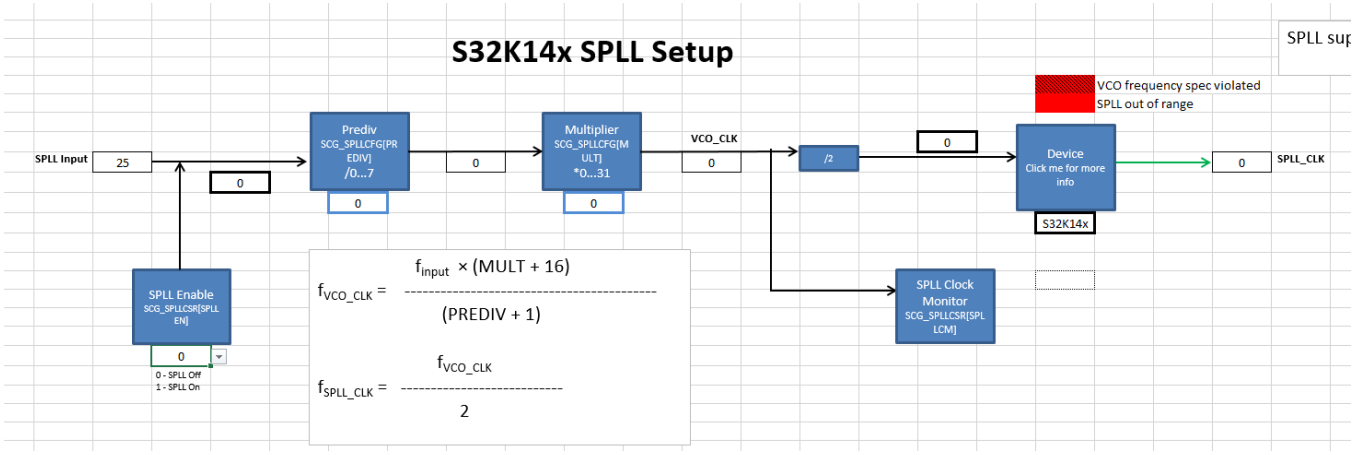


图12. SPLL 控制

SPLL 的输入源是 SOSC。然后，从源头设置位于 SPLL 选项卡中的分频器和乘法器，以实现 SPLL 输出频率。SPLL 输出频率依次输入到树选项卡中的 SPLL_CLK 时钟域。如前几节所述，S32K11x 缺少PLL，因此如果在Device Select中选择 S32K11x，则 SPLL_CLK 将始终为 0。

2.7 spll 时钟

选项卡 spll_clk 是一个参考表，供用户查找合适的 SPLL 分频器和乘法器以实现所需的 SPLL 频率。请注意，这些选项卡的 A、B 和 C 列已冻结，因此如果表格看起来不完整，只需向左或向右滚动即可。

SPLL 频率是根据参考频率、乘法器（MFD）和预分频器（PREDIV）计算得出的。SPLL 参考不可手动配置，因为 SPLL 可以采用有限数量的输入值；SPLL 将是 SOSC 配置的频率。因此，SPLL 参考来自树选项卡。选择 SPLL 参考频率后，输入所需的 SPLL 输出频率。然后参考表将计算每个有效 MFD 和 PREDIV 设置的输出频率。与其它部分一样，频率用颜色编码来定义哪些值有效，哪些无效。一旦计算出输出 SPLL 频率，阴影将自动改变。达到确切所需频率的 MFD 和 PREDIV 设置将以绿色阴影显示，超过所需频率但在 S32K 硬件规范内的值以黄色标记，超过 S32K 硬件规范的频率以红色标记。下面是参考表的屏幕截图。

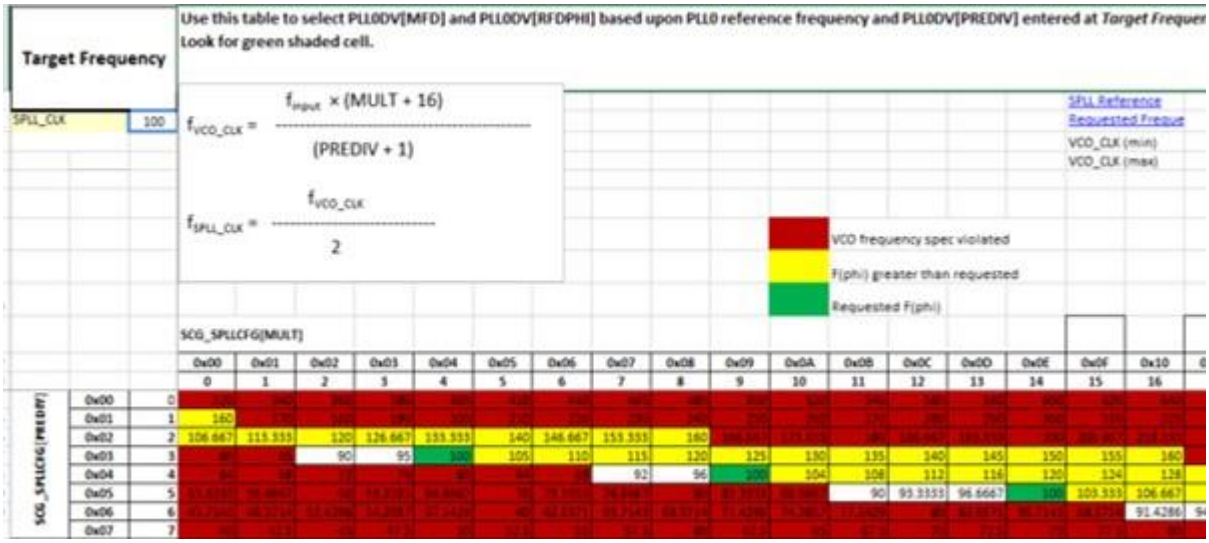


图13. SPLL_CLK参考表

2.8 详细模块图（RTC、SAI、QSPI、ENET、FlexCAN）

一些模块（例如 FlexCAN 和 QSPI）具有附加时钟配置选项，这些选项太大而无法放入 Module Domains 选项卡中。因此，模块 RTC、SAI、QSPI、ENET 和 FlexCAN 都有自己的专用选项卡。以下部分显示了 RTC。它的概念可以外推到其他上述外围设备。Module Domains 中的 RTC 块是指向 RTC Clocking 选项卡的超链接，如下所示。

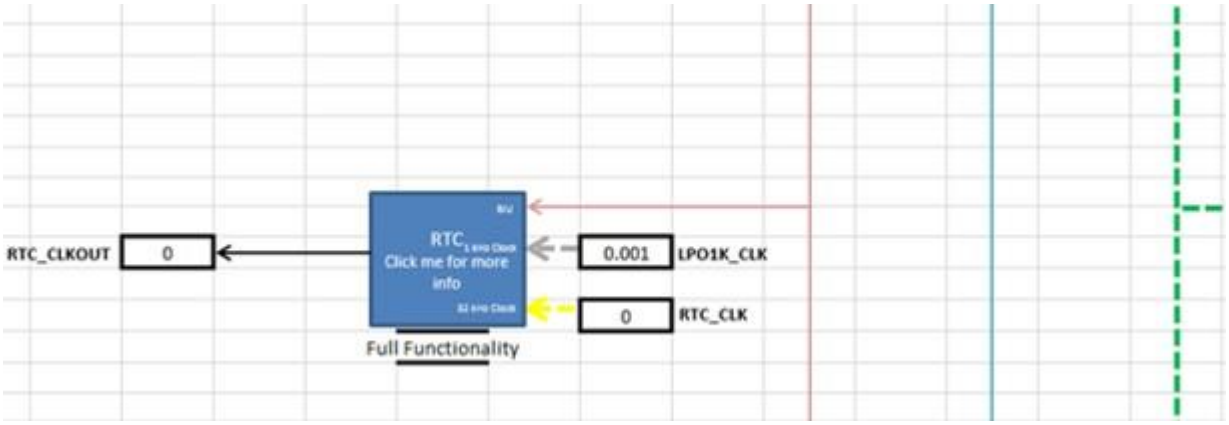


图14. 模块域中的RTC模块

如上图显示，RTC 模块使用 *BUS_CLK*、*LPO1K_CLK*、*RTC_CLK* 时钟源做输入，输出 *RTC_CLKOUT*。RTC 时钟选择包含处理这三个输入以生成 *RTC_CLKOUT* 的实际 RTC 设置选项。下面是 *RTC 时钟*选项卡的屏幕截图。

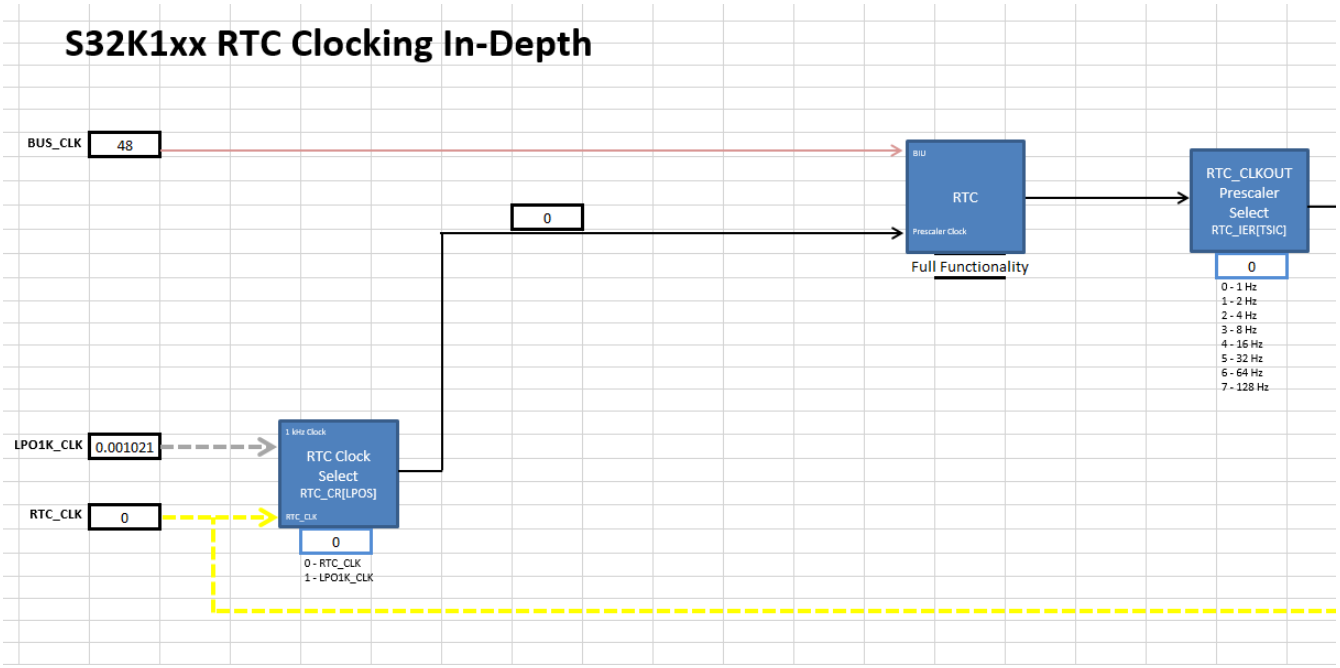


图15. S32K RTC 时钟

2.9 总结

几乎所有构成此时钟计算器的模块都代表MCU中的真实寄存器字段。 “摘要”选项卡将来自时钟计算器其余部分的所有信息整理到寄存器值列表中，其屏幕截图如下图所示。寄存器摘要中的值是交互式的，当相关块更改时会自动更新。摘要中列出的寄存器只是其值受时钟配置影响的寄存器，而不是 SoC 中可用的每个寄存器。

S32K1xx Summary

Register Summary

Register	Value
SCG_FIRCCFG	0x00000000
SCG_FIRCCSR	0b00000X11X0000000000000XX0000X001
SCG_FIRCDIV	0x00000000
SCG_SIRCCFG	0x00000001
SCG_SIRCCSR	0x0XX00005
SCG_SIRCDIV	0x00000000
SCG_SOSCCFG	0b00000000000000000000000000001X000
SCG_SOSCCSR	0b00000X00X00000XX0000000000000X0000
SCG_SOSCDIV	0x00000000
SCG_SPLLCFG	0x00000000
SCG_SPLLDIV	0x00000000
SCG_SPLLCSR	0b00000XXXX00000XX0000000000000000
SMC_PMPROT	0x00000000
SMC_PMCTRL	0b00000000000000000000000000000X000
SMC_STOPCTRL	0x00000003
PMC_LPOTRIM	0x1C
PMC_REGSC	0b0X000X00
SCG_RCCR	0x03000001
SCG_VCCR	0x00000000
SCG_HCCR	0x00000000
SCG_CLKOUTCNFG	0x03000000
SIM_LPOCLKS	0x00000003
SIM_CHIPCTL	0b00000000000XXXXXX00XX00000000XXXX

图16. 寄存器汇总表

寄存器值以十六进制或二进制格式显示，其中“0x”标头代表十六进制，“0b”代表二进制。大写的“X”代表“无关”位/半字节。这些位确实会影响时钟频率，因此用户可以将这些值设置为适合其目的的值。用户可以通过在时钟计算器中设置他们想要的配置，然后利用摘要中的寄存器值将结果寄存器值复制到代码中。例如，从上图中可以看出，寄存器 SCG_SIRCCSR 应设置为 0x0XX00001。假设“X”为“0”，则生成的 S32DS C 代码将为“SCG->SIRCCSR = 0x00000001”。

摘要还包括时钟域频率的概述。由于该工具由多个相互依赖的电子表格组成，因此用户将它们全部联系起来以找到时钟域可能会很麻烦。这张表提供了一个可以找到所有这些的地方。该表按模块组织，然后是时钟类型（即 BIU 时钟、外设时钟、协议时钟等），最后是当前配置的频率。下面是截图。

Module	Clock Domain	Frequency (MHz)
System	FIRC	48
	SIRC	8
	SOSC	8
	LPO128K_CLK	0.128
	LPO32K_CLK	0.032
	LPO1K_CLK	0.001
	SPLL_CLK	96
	CORE_CLK	48
	SYS_CLK	48
	BUS_CLK	48
	FLASH_CLK	24
	SPLLDIV1_CLK	0
	SPLLDIV2_CLK	0
	FIRCDIV1_CLK	0
	FIRCDIV2_CLK	24
	SIRCDIV1_CLK	0
	SIRCDIV2_CLK	0
	SOSCDIV1_CLK	0
	SOSCDIV2_CLK	0
	CLKOUT	0
	LPO_CLK	0.128
	RTC_CLKOUT	0
LPSPI0:2	BIU	48
	Peripheral Clock	24

图17. 时钟汇总表

该工具还支持一定程度的代码生成。摘要提供了两个时钟初始化的样例函数，*SysClk_Init* 用于配置振荡器和 PLL，*InitPeriClkGen* 用于为辅助时钟提供源/分频器。这些函数中的动态 C 代码取决于工具设置，就像寄存器摘要一样。这些函数可以通过 Ctrl+C/Ctrl+V 复制粘贴到源文件中，或者如果启用了宏，则单击相关的复制代码按钮。下图显示了 *SysClk_Init* 及其复制代码按钮。

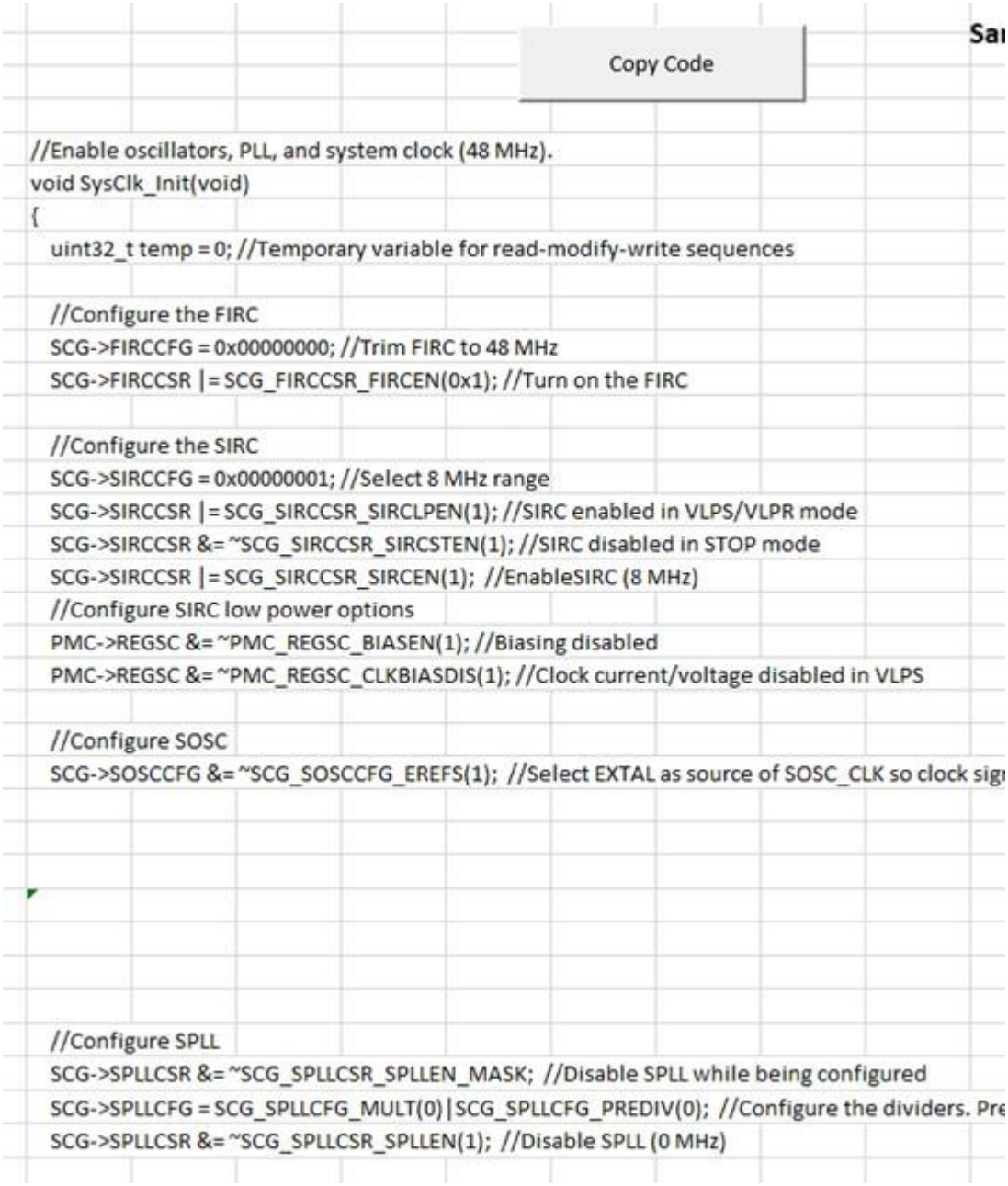
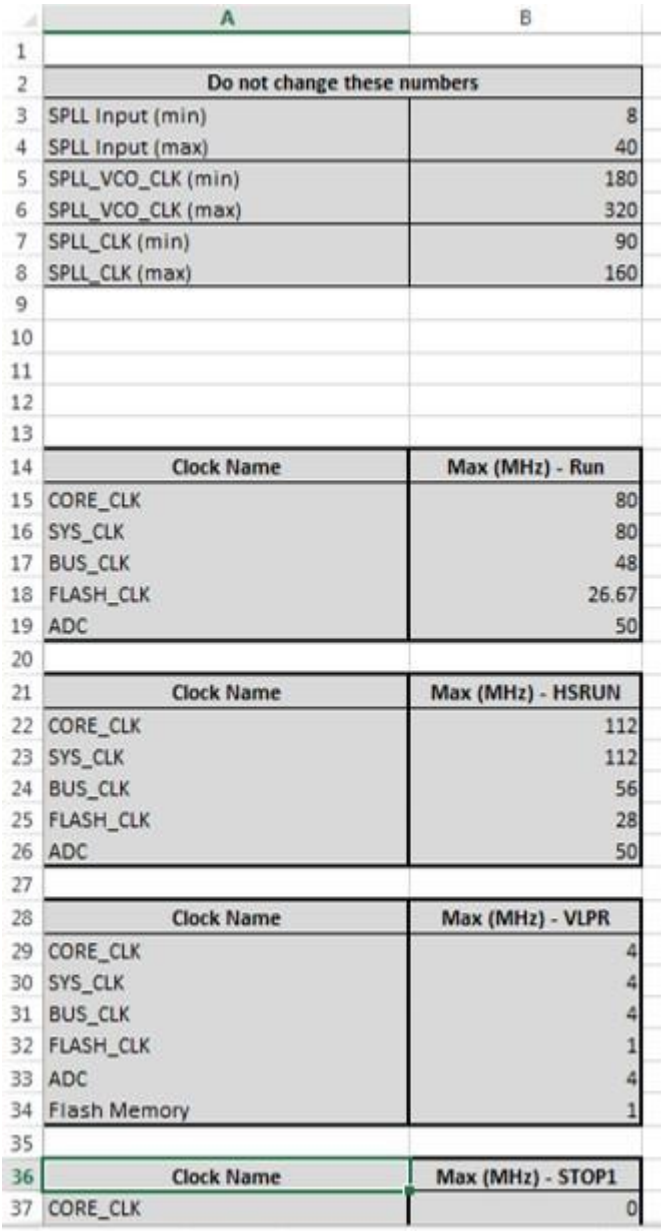


图18. 示例初始化代码

2.10 限制

限制是所有颜色编码规则的参考选项卡。其表中的值基于 S32K 数据表和参考手册，因此用户不要修改。下图是“限制”选项卡的屏幕截图。



	A	B
1		
2	Do not change these numbers	
3	SPLL Input (min)	8
4	SPLL Input (max)	40
5	SPLL_VCO_CLK (min)	180
6	SPLL_VCO_CLK (max)	320
7	SPLL_CLK (min)	90
8	SPLL_CLK (max)	160
9		
10		
11		
12		
13		
14	Clock Name	Max (MHz) - Run
15	CORE_CLK	80
16	SYS_CLK	80
17	BUS_CLK	48
18	FLASH_CLK	26.67
19	ADC	50
20		
21	Clock Name	Max (MHz) - HSRUN
22	CORE_CLK	112
23	SYS_CLK	112
24	BUS_CLK	56
25	FLASH_CLK	28
26	ADC	50
27		
28	Clock Name	Max (MHz) - VLPR
29	CORE_CLK	4
30	SYS_CLK	4
31	BUS_CLK	4
32	FLASH_CLK	1
33	ADC	4
34	Flash Memory	1
35		
36	Clock Name	Max (MHz) - STOP1
37	CORE_CLK	0

图19. S32K频率限制

3 时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

以下部分将介绍 S32K 时钟计算器的示例应用。本应用笔记的示例将 LPSPI 总线接口时钟配置为 40MHz 的 SPLL，将 LPSPI 外设时钟配置为 24MHz 的 FIRC。它不仅会显示正确的配置，还会显示工具在尝试不正确的配置时如何响应。为模块配置时钟时，首先查看该模块。对于此示例，请在模块域中找到 LPSPI0:2。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

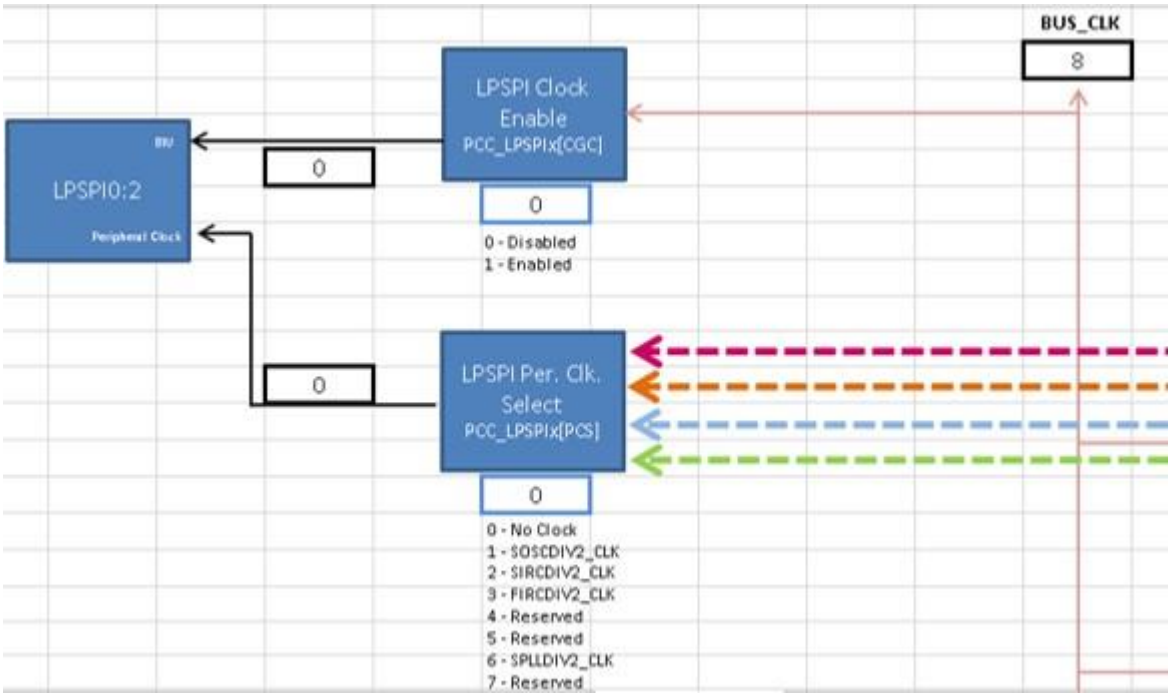


图20. LPSPI时钟

在模块域中，该模块图显示 BUS_CLK 作为总线接口时钟，而 SOSCDIV2_CLK、SIRCDIV2_CLK、FIRCDIV2_CLK 或 SPLLDIV2_CLK 可以作为 LPSPI 外设引擎时钟。LPSPI 总线接口时钟 BUS_CLK 当前为 8MHz；LPSPI 外设时钟为 0MHz，因为模块 LPSPI Per.Ckl.Selet 的值为0，表示没有选择时钟。配置时钟计算器可以按任何顺序进行，本示例将从 BUS_CLK 开始。

3.1 设置设备

首先，确保选择了正确的 S32K MCU。此示例开始配置 S32K14x，因此请转到“Device Select”选项卡并将设备更改为 S32K14x。

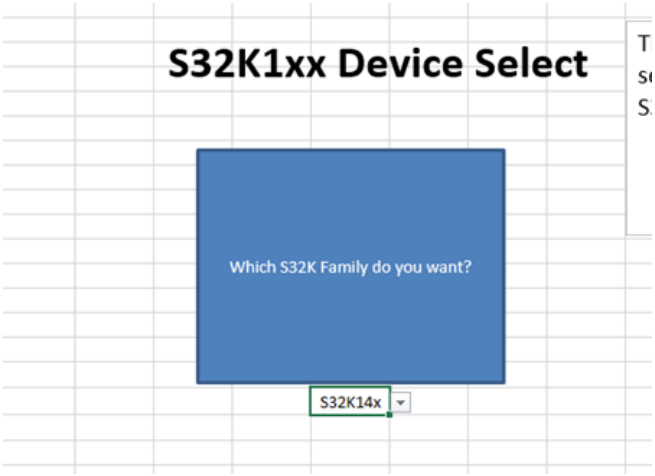


图21. 选择 S32K14x

3.2 设置电源模式

接下来确保系统处于运行模式。转到 Power Mode Control 选项卡并将 S32K Power Mode 块设置为 Run，如下图所示。

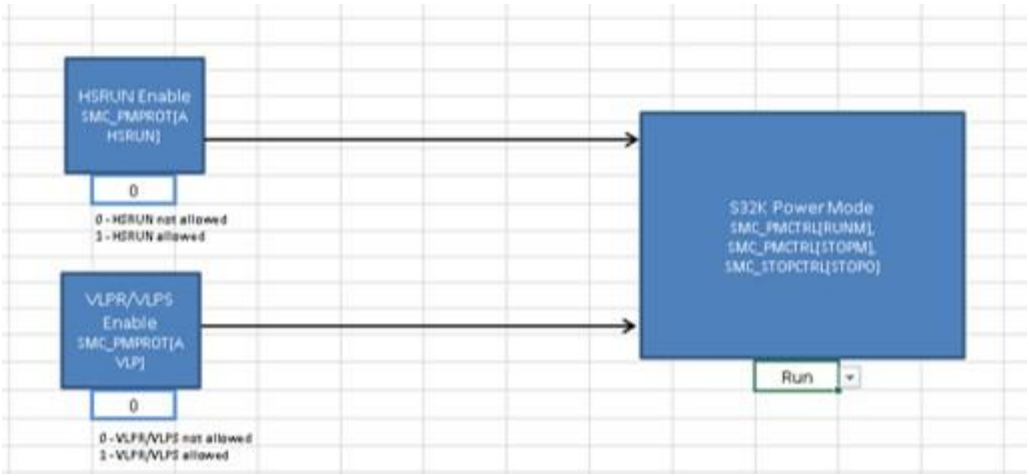


图22. 运行模式下的 S32K

3.3 配置BUS_CLK

返回 Module Domains 选项卡并单击 *BUS_CLK*；会跳到Tree 选项卡的 *BUS_CLK*，如下所示。

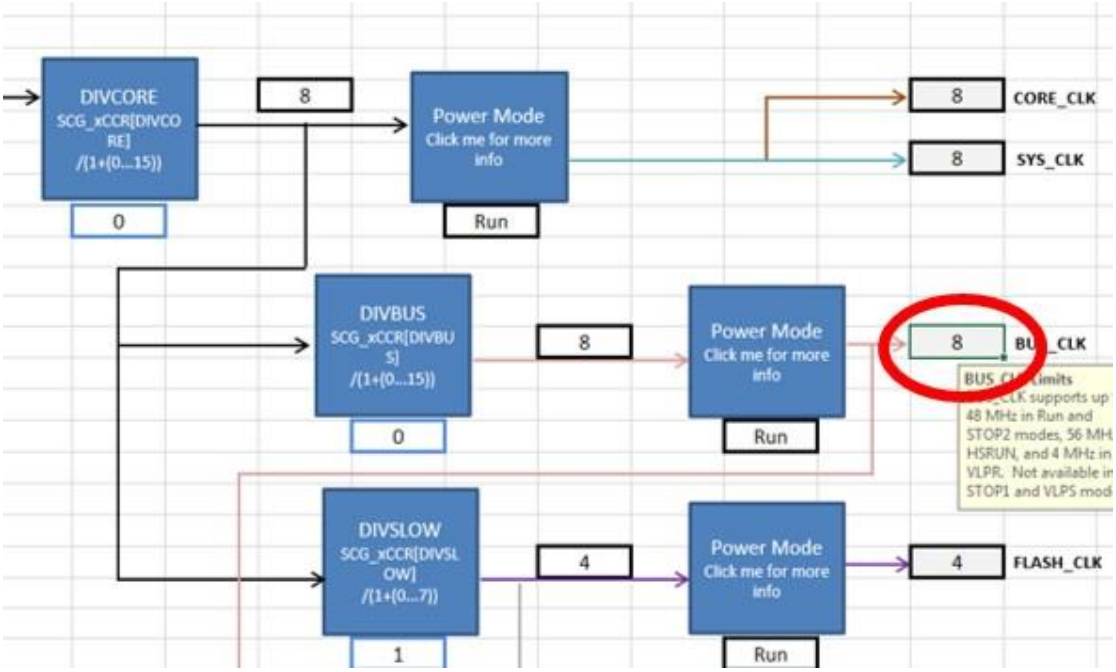


图23. BUS_CLK树状图选项卡

一直找回到BUS_CLK的源头。首先找到电源模式块，然后是分频器 DIVBUS，再到 DIVCORE，最后是系统时钟选择器，其当前值为2。单元格是一个下拉菜单，文本框解释了每个可用值与什么相关联。

由于我们的目标是将 *BUS_CLK* 配置为 SPLL，因此将 SPLL 追溯到其自己的源。SPLL 源来自 SOSC。振荡器 FIRC、SIRC、SOSC 和 LPO 是所有时钟域的原点。下图显示了从 SPLL 到振荡器的回溯。

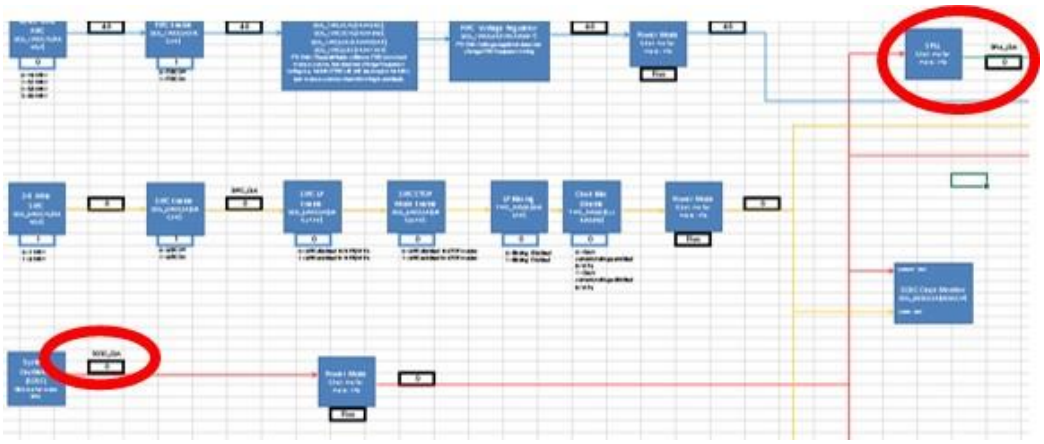


图24. SPLL到SOSC

3.3.1 配置振荡器

现在开始下行，从振荡器配置到 *BUS_CLK*。要为 SPLL 提供源，要从 SOSC 开始。单击 *SOSC_CLK* 文本框以转至振荡器源控制选项卡。

SOSC_CLK 可以来自外部振荡器 XTAL 或引脚 EXTAL 的信号。XTAL的设置和应用相关，可以是 4MHz 到 8MHz 或 8MHz 到 40MHz 之间的任何值，具体取决于 XTAL 配置。EXTAL 必须低于 50MHz。将 SOSC Range 块设置为 3 以选择 8-40MHz 范围，如下图所示。4-8/8-40MHz SOSC 模块现在可以采用 8 到 40MHz 之间的任何值。

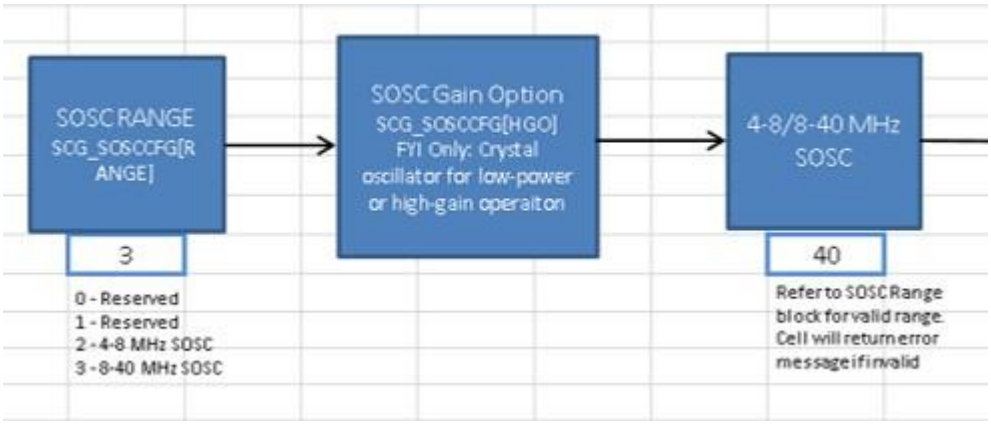


图25. SOSC设置为高范围

此工具具有防止输入无效值的保护措施。下图显示了向 SOSC 频率单元输入 7MHz 的尝试。当用户尝试单击其它单元格时，会出现一个对话框，通知用户该值未被接受。

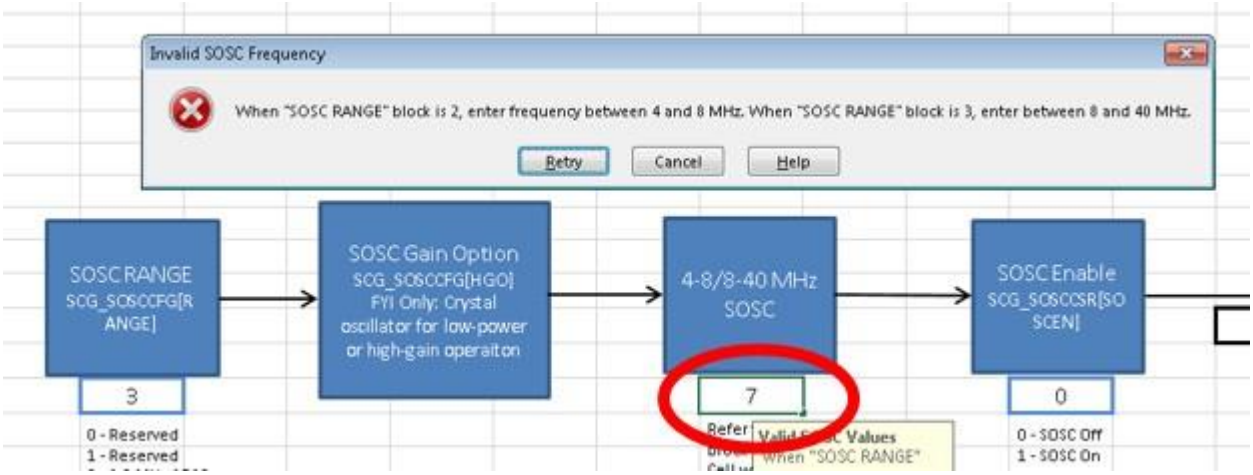


图26. 无效频率输入

将 SOSC 频率设置为 8MHz。从4-8/8-40MHz SOSC 模块向前跟踪到 SOSC Enable。将 SOSC Enable 设置为1以启用8MHz SOSC 向下游传播，如下所示。

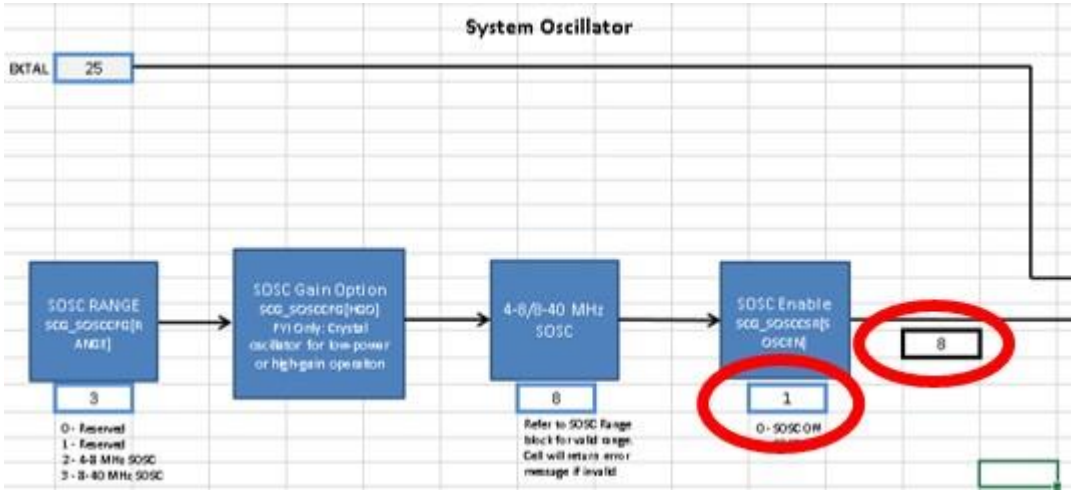


图27. SOSC打开

接下来，配置 Ext.Ref. 选择1以选择 XTAL 而不是 EXTAL。SOSC_CLK 将来自 8MHz 的系统振荡器而不是 EXTAL 引脚。如下。

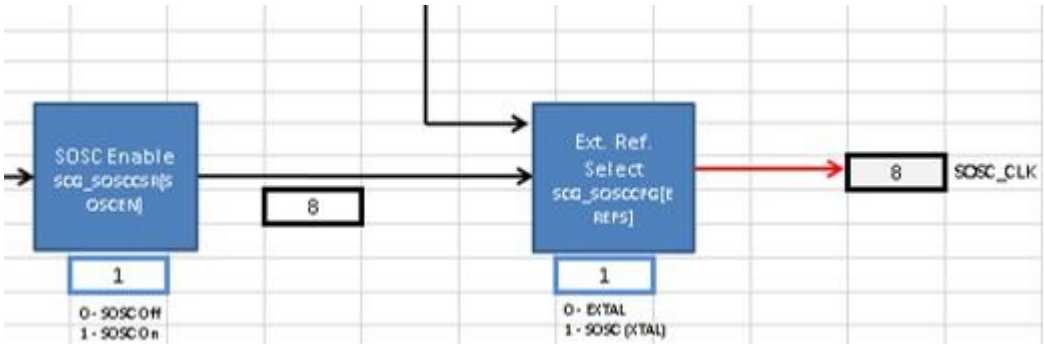


图28. SOSC_CLK 配置为跟随 40MHz 的外部振荡器

3.3.2 配置SPLL

现在 SOSC_CLK 设置为 8MHz，返回 Tree 选项卡并追踪 SOSC_CLK 进入 SPLL 模块，如下图所示。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

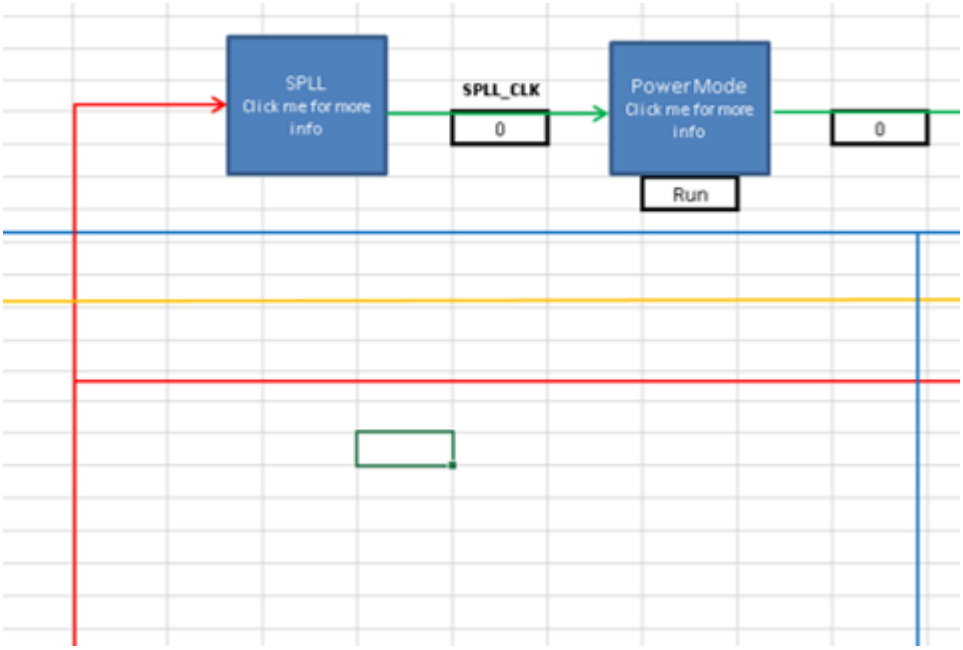


图29. SPLL

单击 SPLL 块以自动跳转到 SPLL 选项卡。这是设置 SPLL_CLK 频率的选项卡。下图的输入时钟模块显示 SPLL 检测 8MHz SOSC_CLK 作为其源频率。

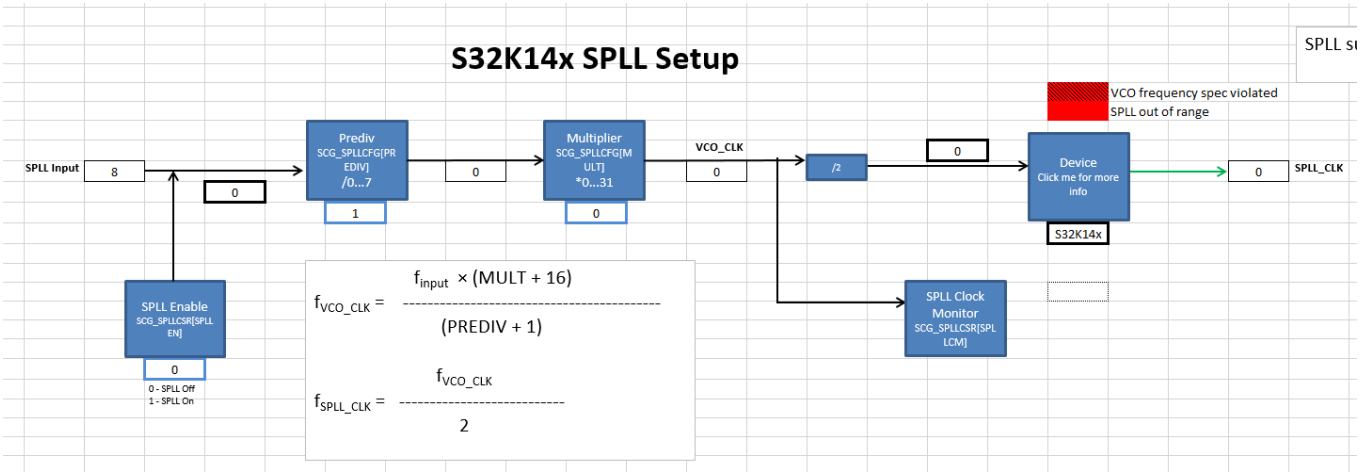


图30. SPLL计算器

配置分频器以达到 96MHz；该频率稍后将被划分为 48MHz。正确的配置可以通过反复试验来实现，但是 S32K 时钟计算器在 spll_clk 选项卡中提供了一个查找表，如下所示。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

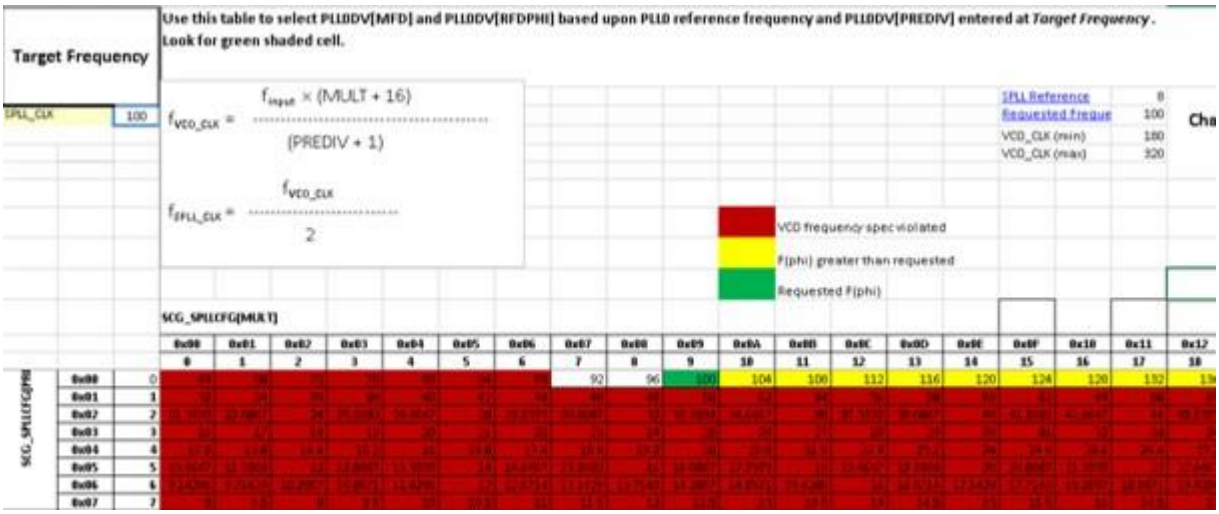


图31. spll_clk参考表

SPLL 参考字段是 SPLL 输入的频率，在本例中为 8MHz SOSC。设置目标频率。此示例将针对 96MHz。查找表中的值和阴影将自动更改以适应这些新设置。下图中，表格发生了变化，圈出的是修改后的字段。

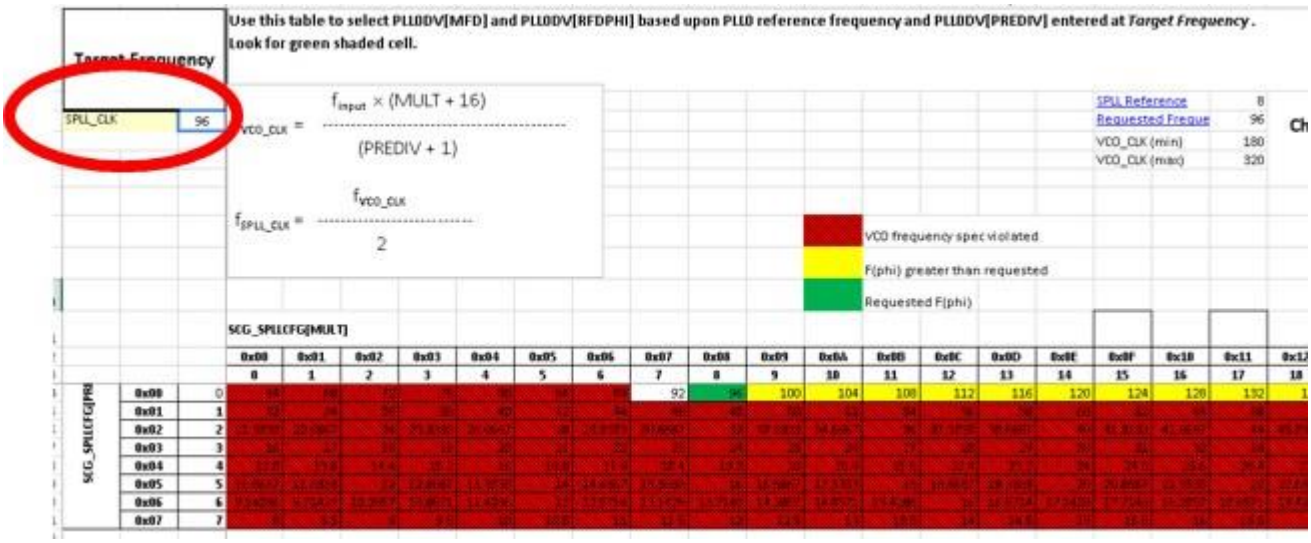


图32. 带有新设置的 spll_clk 表

带绿色阴影的单元格表示在输入频率为 8MHz 的情况下，有一个分频器组合可以准确实现 96MHz。在这种情况下，MFD 为 8 且 PREDIV 值为 0 即可完成这项工作。但是，值得注意的是如果输出 SPLL 频率超出范围会发生什么。

在下图中，SPLL 已配置为输出频率为 188MHz。这显然超过了 160MHz 硬件规范的最大值。相关的压控振荡器（VCO）频率（可以从 *SPLL_CLK* 反向计算）也超过了 320MHz 的最大 VCO 频率。因此，输出是交叉影线和红色阴影。

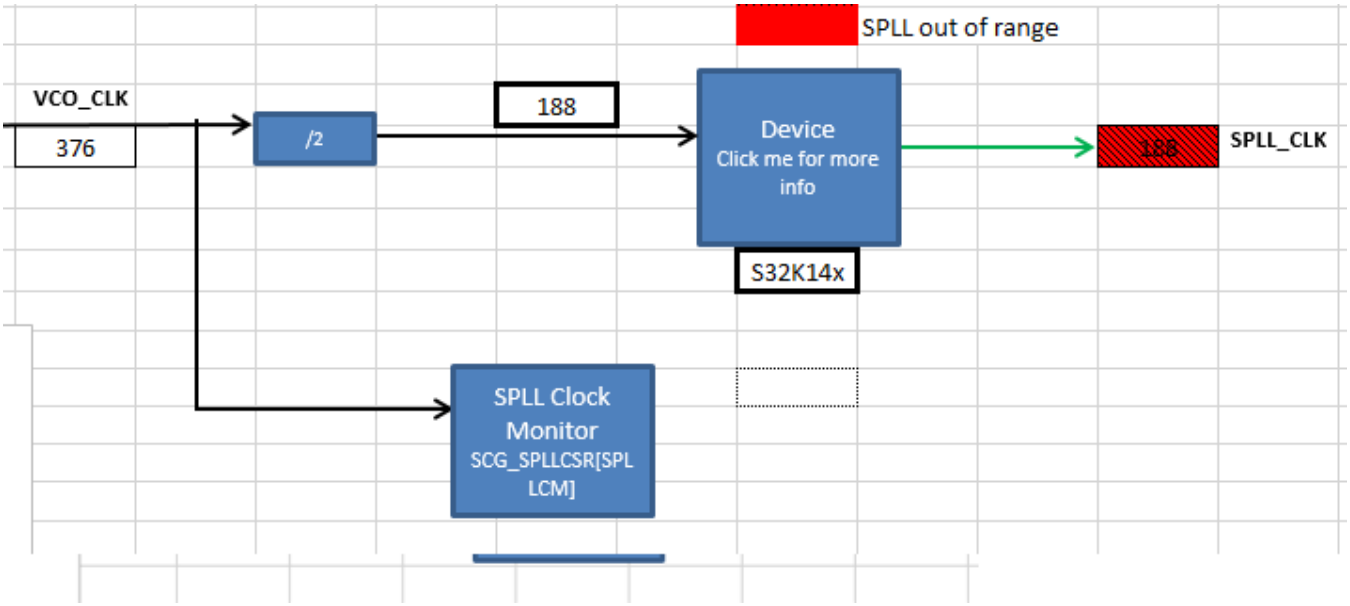


图33. 当SPLL_CLK超过VCO和PLL规格时

现在让我们正确配置 SPLL。通过将 SPLL Enable 模块设置为 1，打开 SPLL 选项卡中的 SPLL，然后将 Prediv 设置为 0，Multiplier 设置为 8。如下图所示，输出 *SPLL_CLK* 为 96MHz，并且单元保持无阴影，这意味着配置 符合规范。

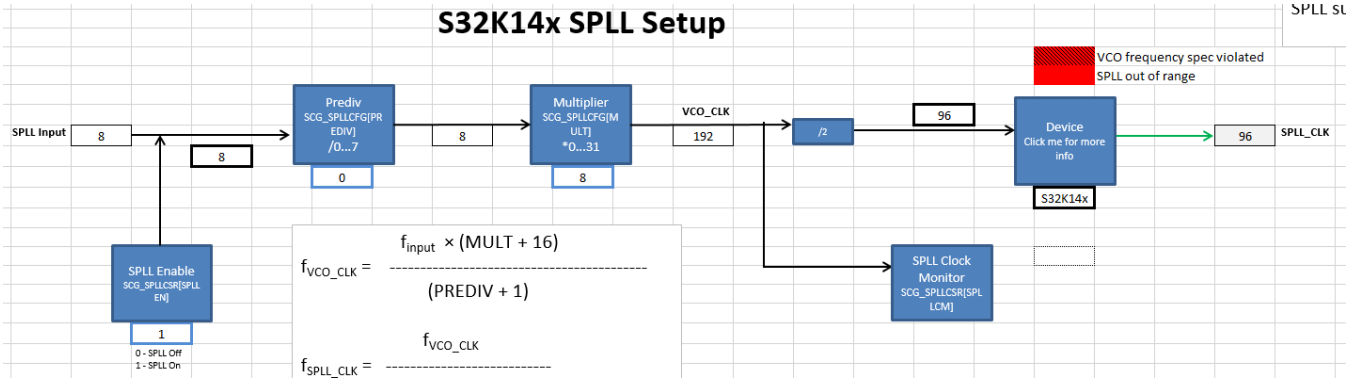


图34. SPLL_CLK 配置为 96MHz

返回 Tree 选项卡观察 *SPLL_CLK* 频率现在为 96MHz。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

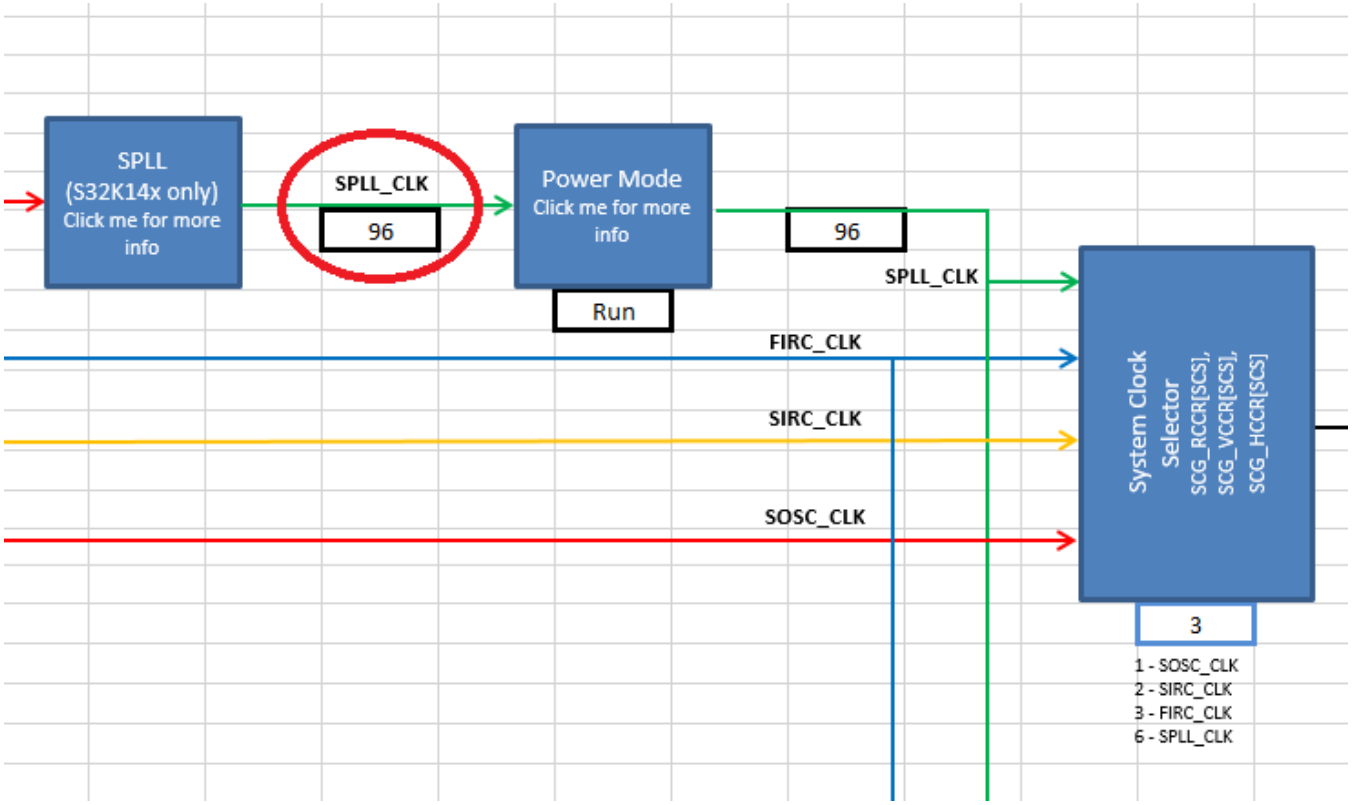


图35. SPLL_CLK 传播到树状图

3.3.3 BUS_CLK设置完成

BUS_CLK 是系统时钟之一。因此，跟随 *SPLL_CLK* 信号向下到达系统时钟选择器。*SIRC_CLK* 是现在系统时钟的时钟源。将 System Clock Selector 的值更改为 6，以使系统时钟配置成 *SPLL_CLK*，如下所示。

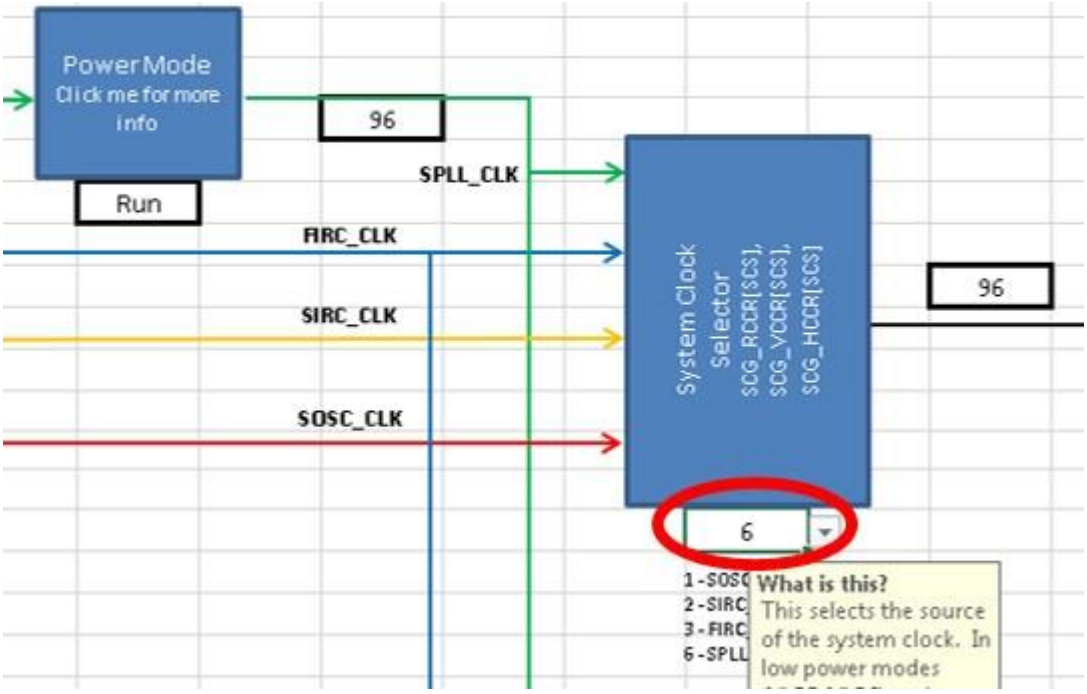


图36. 系统时钟更改为 FMPL

此后，跟随系统时钟输出到 DIVCORE。在运行模式下，CORE_CLK 和 SYS_CLK 的最大频率为 48MHz，因此将 DIVCORE 从 0 设置为 1。这会将 96Mhz 信号除以 2，从而将 CORE_CLK，SYS_CLK 以及到 DIVBUS 模块的输入设置为 48MHz，其输出为 BUS_CLK。见下图。

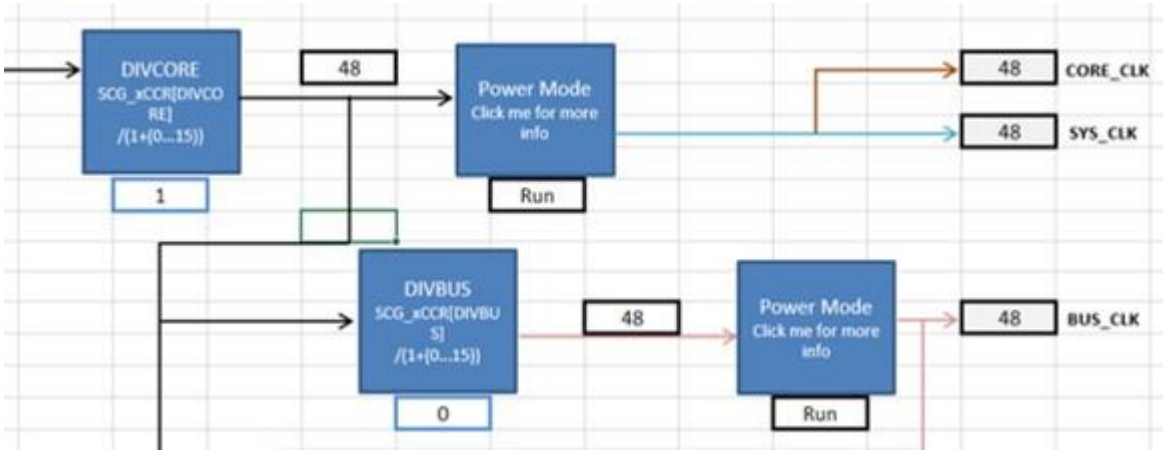


图37. 2处的DIVCORE

这些字段的输入不是所需的分频器的值，而是为实现所需的分频器而必须输入的位域值。这就是为什么 DIVCORE 块描述声明 “/(1+(0…15))”而不是简单的“/1…16”。用户输入的 0 到 15 之间的值，硬件会自动将其加 1 以计算 1 到 16 之间的分频系数。

例如，如果 DIVCORE 设置为 0，这对应于 1 的分频器，则 CORE_CLK 和 SYS_CLK 将为 96MHz，这将超过它们的最大允许频率 48MHz。该工具将它们的单元格突出显示为红色，以表示不允许这样的频率，如下所示。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

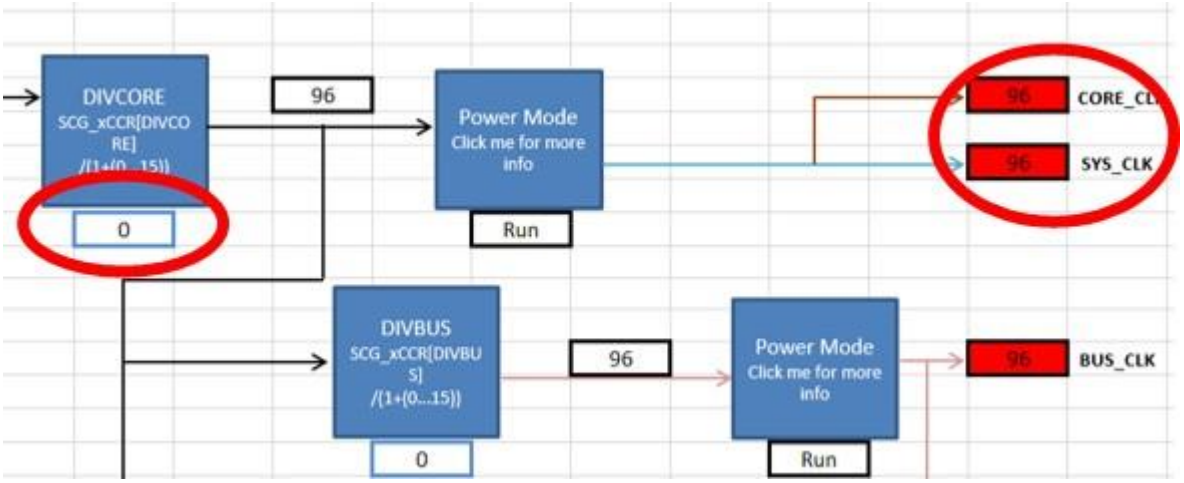


图38. 频率超过规格时的系统时钟

将 DIVCORE 设置回 1 并将 *DIVBUS* 保留为 0，以将 *BUS_CLK* 保持在 48MHz。*BUS_CLK* 现在已配置为 48MHz SPLL，如下图所示。

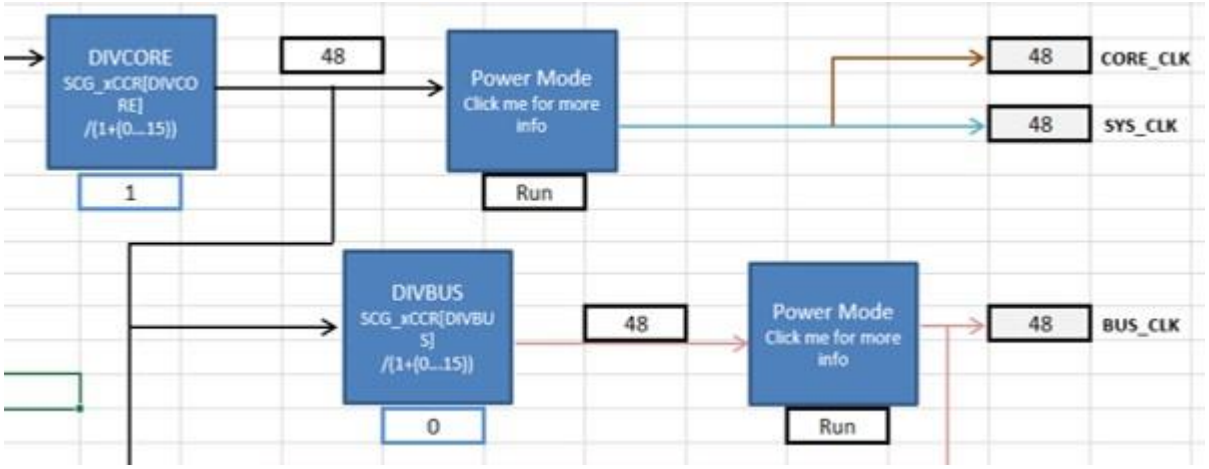


图39. BUS_CLK正确配置

3.4 配置LPSPI外设时钟FIRCDIV2_CLK

LPSPI 使用 BUS_CLK 作为其总线接口时钟，但外设时钟可以是 *SOSCDIV2_CLK*、*SIRCDIV2_CLK*、*FIRCDIV2_CLK* 或 *SPLLDIV2_CLK*。此示例将外设时钟设置为 24MHz 的 FIRCDIV2_CLK。树选项卡中找到 48MHz FIRC 块。S32K 的 FIRC 只能修整到 48MHz，因此将 48MHz FIRC 块值设置为 0，并将 FIRC Enable 设置为 1 以使信号传播，如下所示。

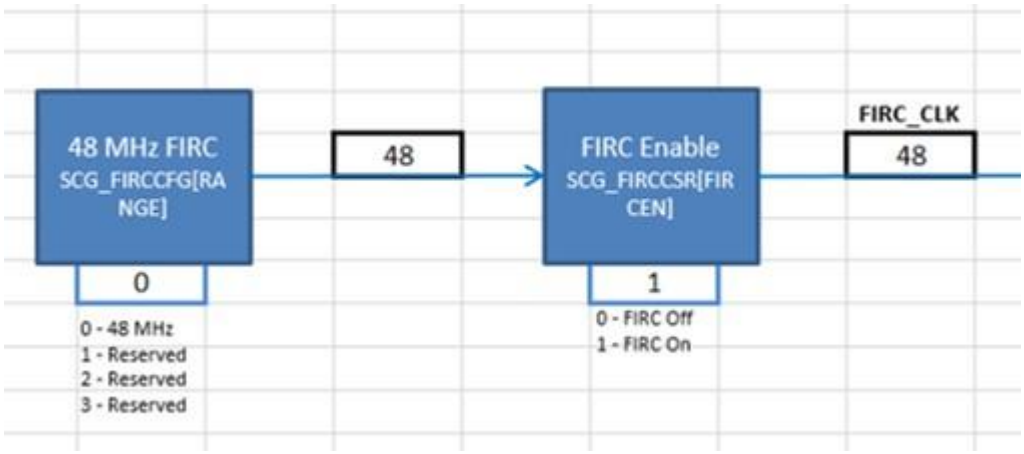


图40. FIRC 在 48MHz

追踪 FIRC 时钟信号到 Tree 选项卡的 FIRCDIV2 块并将该块设置为2。这将使能 *FIRCDIV2_CLK* 并将 48MHz FIRC 信号除以2，从而实现 24MHz 的 FIRCDIV2_CLK 域。请参见下图。

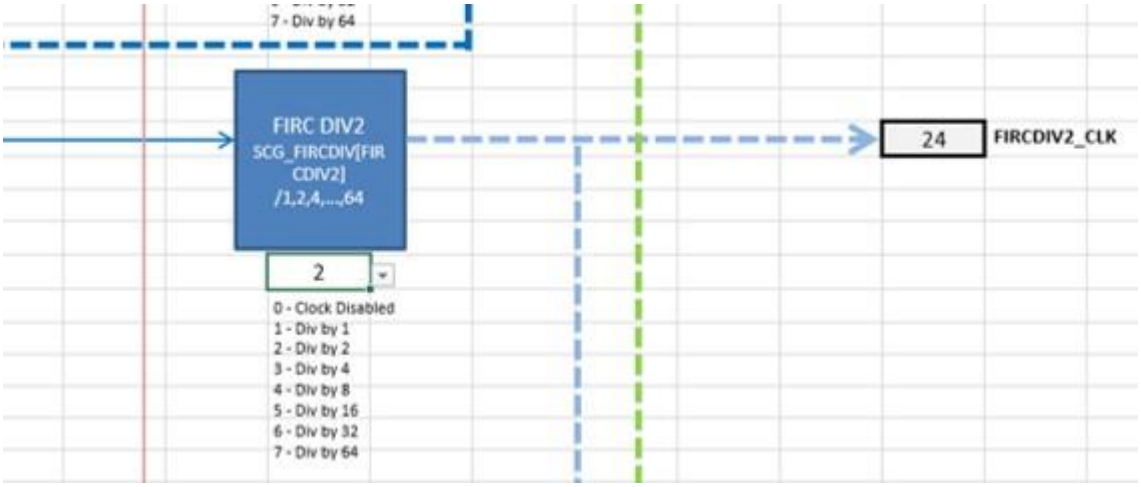


图41. FIRCDIV2_CLK 设置为 30MHz

3.5 配置LPSPI时钟

返回到模块域选项卡。将 LPSPI 时钟启用模块设置为 1 以启用 *BUS_CLK* 信号。LPSPI 总线接口时钟现在是 48MHz *BUS_CLK*。将LPSPI外设时钟配置为 *FIRCDIV2_CLK*，设置 LPSPI Per.Clk.Select 的值为3。LPSPI 如下图所示。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

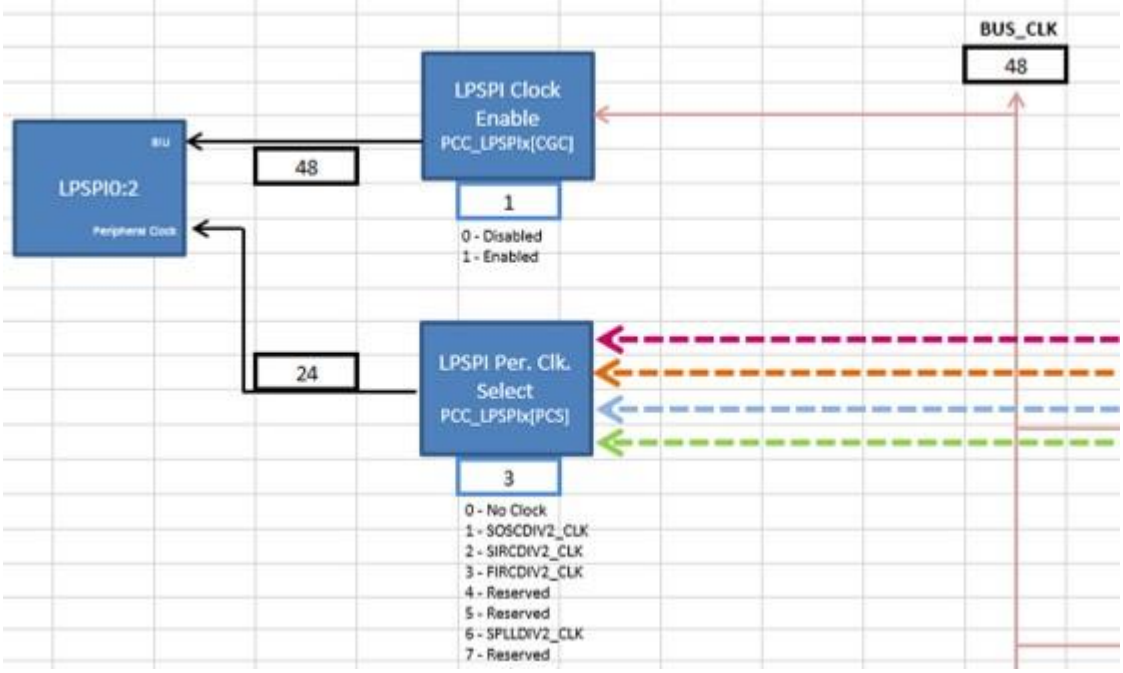


图42. LPSPI最终配置

3.6 观察寄存器

最终寄存器汇总表，如 Summary 中所示，如下图所示。请注意，要实现本样例的配置，大多数寄存器是不需要配置的。例如，不必包含寄存器 PCC_FlexIO，因为 FlexIO 模块未受影响。必须写入的寄存器将是 SCG_FIRCDIV 和 PCC_LPSPIx 之类的寄存器（“x”表示您选择的 LPSPI 模块）。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

Register	Value
SCG_FIRCCFG	0x00000000
SCG_FIRCCSR	0b00000X11X0000000000000XX0000X001
SCG_FIRCDIV	0x00000200
SCG_SIRCCFG	0x00000001
SCG_SIRCCSR	0x0XX00001
SCG_SIRCDIV	0x00000000
SCG_SOSCCFG	0b00000000000000000000000000011X100
SCG_SOSCCSR	0b00000X00X00000XX000000000000X0001
SCG_SOSCDIV	0x00000000
SCG_SPLLCFG	0x00080000
SCG_SPLLDIV	0x00000000
SCG_SPLLCSR	0b00000XXXX00000XX0000000000000001
SMC_PMPROT	0x000000A0
SMC_PMCTRL	0b00000000000000000000000000000X000
SMC_STOPCTRL	0x00000003
PMC_LPOTRIM	0x00
PMC_REGSC	0b0X000X00
SCG_RCCR	0x06010001
SCG_VCCR	0x00000000
SCG_HCCR	0x00000000
SCG_CLKOUTCNFG	0x03000000
SIM_LPOCLKS	0x00000003
SIM_CHIPCTL	0b00000000000XXXXXX00XX00011001XXXX
PCC_LPSPIx	0xC3000000
PCC_LPIT	0x80000000
PCC_FlexIO	0x80000000
FLEXIO_CTRL	0bXX000000000000000000000000000XX
PCC_LPI2Cx	0x80000000
PCC_LPUARTx	0x80000000
PCC_EWM	0x80000000

图43. 配置后的寄存器摘要

3.7 复制代码

SysClk_Init 和 *InitPeriClkGen* 提供动态时钟生成 的 C 代码。该代码会将现在的时钟，配置为此时钟计算器中配置的值。它可以复制并粘贴到源文件中。下图显示了本示例配置的 *SysClk_Init*。函数周围的实心边框高亮表示代码已被复制代码按钮复制；常规的 Ctrl+C 会导致虚线边框突出显示。在这两种情况下，都可以使用常规的 Ctrl+V 将代码粘贴到源中。

时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟

Sample Initialization Code

Copy Code

```
//Enable oscillators, PLL, and system clock (48 MHz).
void SysClk_Init(void)
{
  uint32_t temp = 0; //Temporary variable for read-modify-write sequences

  //Configure the FIRC
  SCG->FIRCCFG = 0x00000000; //Trim FIRC to 48 MHz
  SCG->FIRCCSR |= SCG_FIRCCSR_FIRCEN(0x1); //Turn on the FIRC

  //Configure the SIRC
  SCG->SIRCCFG = 0x00000001; //Select 8 MHz range
  SCG->SIRCCSR |= SCG_SIRCCSR_SIRCLPEN(1); //SIRC enabled in VLPS/VLPR mode
  SCG->SIRCCSR &= ~SCG_SIRCCSR_SIRCSTEN(1); //SIRC disabled in STOP mode
  SCG->SIRCCSR |= SCG_SIRCCSR_SIRCEN(1);  //EnableSIRC (8 MHz)
  //Configure SIRC low power options
  PMC->REGSC &= ~PMC_REGSC_BIASEN(1); //Biasing disabled
  PMC->REGSC &= ~PMC_REGSC_CLKBIASDIS(1); //Clock current/voltage disabled in VLPS

  //Configure SOSC
  SCG->SOSCCFG |= SCG_SOSCCFG_EREFS(1);  //Select external oscillator hooked to XTAL and EXTAL pins as source of S
  //Select the range of the external oscillator
  temp = SCG->SOSCCFG;  //read-modify-write for multi-bit RANGE field
  temp &= ~SCG_SOSCCFG_RANGE_MASK;  //Clear field
  temp |=  SCG_SOSCCFG_RANGE(1);  //8-40 SOSC range
  SCG->SOSCCFG = temp;  //Write new value to register
  SCG->SOSCCSR |= SCG_SOSCCSR_SOSCEN(1); //EnableSOSC (8 MHz)

  //Configure SPLL
  SCG->SPLLCSR &= ~SCG_SPLLCSR_SPLLEN_MASK;  //Disable SPLL while being configured
  SCG->SPLLCFG = SCG_SPLLCFG_MULT(8)|SCG_SPLLCFG_PREDIV(0);  //Configure the dividers. Prediv = 0 Mult = 8.
  SCG->SPLLCSR |= SCG_SPLLCSR_SPLLEN(1);  //Enable SPLL (96 MHz)

  //Configure SCG_CLKOUT
  SCG->CLKOUTCNFG = 0x03000000;  //Select FIRC_CLK as source of SCG_CLKOUT

  //Configure the low power oscillator
  PMC->LPTRIM = 0x00;  //Configure the LPO trim value. LPO is 128 kHz.
  PMC->REGSC &= ~PMC_REGSC_LPODIS(1); //Enable the LPO
  //Configure the LPO_CLK source
  temp = SIM->LPOCLKS;  //Read-modify-write
  temp &= ~(SIM_LPOCLKS_LPOCLKSEL_MASK|SIM_LPOCLKS_LPO32KCLKEN_MASK|SIM_LPOCLKS_LPO1KCLKEN);  //
  temp |=  SIM_LPOCLKS_LPO32KCLKEN(1); //Enable LPO32K_CLK
  temp |=  SIM_LPOCLKS_LPO1KCLKEN(1);  //Enable LPO1K_CLK
  temp |=  0;  //Select LPO128K_CLK as source of LPO_CLK
  SIM->LPOCLKS = temp;  //Write to register

  //Configure the system clock source and dividers, depending on power mode
  SCG->RCCR = SCG_RCCR_SCS(6)|SCG_RCCR_DIVCORE(1)|SCG_RCCR_DIVBUS(0)|SCG_RCCR_DIVSLOW(1);

  //Configure the S32K power modes
  SMC->PMPROT = 0x00000000;  //HSRUN disabled and VLPR/VLPS disabled.
  //Switch to Run.
  SMC->PMCTRL = SMC_PMCTRL_RUNM(0);  //Switch to normal Run mode

}
```

Copy Code

图44. SysClk_Init示例后

所以，总而言之，这个例子已经实现了它的目标：一个总线接口时钟，其信号由 *BUS_CLK* 提供，频率为 48MHz。*BUS_CLK* 来自于 8MHz SOSC，产生的SPLL 96MHz 的输出，然后 SPLL 以 48MHz 输出给 *BUS_CLK*。最后是一个由 24MHz *FIRCDIV2_CLK* 驱动的外设时钟，其时钟源是从 48MHz FIRC 分频出来的。

4 结论

本应用笔记概述了 S32K 交互式时钟计算器。它以图形工具的形式辅助进行时钟配置，以便用户可以更轻松地通过可视化工具的配置，理解MCU时钟信号路径及配置。其他 NXP 产品也有类似的时钟计算器，包括 MPC574xP 和 MPC574xG。访问 NXP website 网站以查找更多这些工具。

5 修订历史

版本号	日期	内容更改
1	05/2017	• 在第11页的总结中，添加了文本“摘要中还包含.....截图”，并在第13页中添加了图17。 • 更新了S32K14x_时钟_计算器表，请参阅附件。
2	08/2017	• 在简介中添加了文字“时钟计算器.....启用所有宏”和“附加到此.....附件”。添加了时钟计算器和查找工具的图。 • 在第4页的树状图中添加了文本“此选项卡.....按钮”，并增加了按钮的图。 • 将小节名称从“RTC时钟”更改为“详细模块图（RTC、SAI、QSPI、ENET、FlexCAN）”，并更新了此小节。 • 在总结中添加了文本“此工具还.....复制代码按钮”，并添加了示例初始化代码的图。 • 增加了“复制代码”小节。 • 添加了更新的S32K14x_Clock_Calculator_Rev3
3	11/2017	更新了关联的 S32K14x_Clock_Calculator 文件
4	01/2018	更新了关联的 S32K14x_Clock_Calculator 文件

表格在下一页继续.....

表格从上一页继续.....（续表）

版本号	日期	内容更改
5	08/2018	• 添加了S32K11x系列的信息。 • 更新了第1页的介绍 • 在第6页增加了设备选择 • 在第16页增加了设置设备 • 更改了第15页的内容：时钟工具配置样例：在S32K14x的RUN模式下，将LPSPI配置为48MHz的SPLL BUS_CLK和24MHz FIRC的外设时钟 • 更新了SPLL Calculator, When SPLL_CLK exceeds VCO and PLL spec, SPLL_CLK propagated to Tree 和 SPLL_CLK configured to 96 MHz. • 更新了寄存器汇总表 • 更新了S32K RTC 时钟 • 更新了SPLL 控制 • 更新了S32K 功耗模式控制 • 更新了时钟计算器树状图 和 按钮 • 更新了S32K1xx 时钟计算机设置
6	09/2018	更新了关联的 S32K14x_Clock_Calculator 文件

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number: AN5408
Rev. 6, 09/2018