Rev. 2,11/2021

S32G 优化 EMI 的软件改频与展频

by John Li (nxa08200)
GSM CAS

S32G支持除了外设PLL外的，内部加速器PLL，ARM_PLL和DDR_PLL的展频功能。通常情况下，加速器模块和CPU的PLL是不需要展频的，因为是内部模块，封装在芯片内部。而外设PLL又不支持展频。所以可以提供的主动改善EMI的方法主要是DDR_PLL的展频功能，因为DDR是有可能通过比如说端接电阻，表面走线辐射到空间中的。

除了展频外，可能还需要调节频率及其产生的谐波中心点，以避开一些敏感的比如说导航卫星信号频率，所以也需要调节DDR_PLL的频率。而事实上，因为S32G仅支持中心展频，所以为了不超过最高要求频率，都是会先降一点频率再展频的。

S32G默认的BSP是通过已经加载到内部SRAM的uboot来初始化DDRC控制器的，所以可以通过C代码来设置相关PLL寄存器来实现以上功能。

自主设计的bootloader可能通过M7的代码直接操作DDRC控制器，则对相关clock寄存器设置可以根据本文所述方法自行设置。

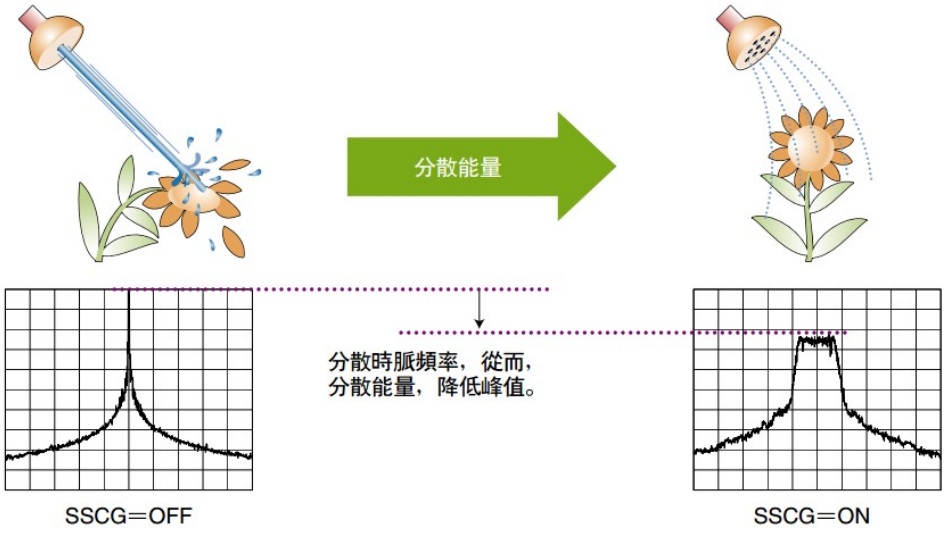
历史	说明	作者
V1	● 创建本文	● John.Li
V2	● 根据数据和芯片手册 V4 内容修改	● John.Li

目录

1 展频的基本概念

EMI，即电磁干扰，是指电路系统通过传导或者辐射的方式，对于周边电路系统产生的影响。时钟信号常常是电路系统中频率最高和边沿最陡的信号，多数 EMI 问题的产生和时钟信号有关。降低 EMI 的方法有许多种，包括屏蔽、滤波、隔离、铁氧体磁环、信号边沿控制以及在 PCB 中增加电源和 GND 层等等。在应用中可以灵活使用以上方法，其中屏蔽是相对简单的机械学方法，成本较高，不适用于手持和便携式设备；滤波和信号边沿控制对于低频信号有效，不适合当前广泛应用的高速信号。另外，使用 EMI/RFI 滤波器这些被动元器件，会增加成本；通过 LAYOUT 技巧降低 EMI 显然比较费时，而且因设计的不同，手段也不尽相同。

展频时钟（Spread Spectrum Clocking）是另一种有效降低 EMI 的方法，时钟展频通过频率调制的手段将集中在窄频带范围内的能量分散到设定的宽频带范围，通过降低时钟在基频和奇次谐波频率的幅度（能量），达到降低系统电磁辐射峰值的目的。



时钟展频通过特定方式调制原始时钟信号。Linear 和 Hershey Kiss 是常用的调制方式。一般嵌入式芯片上仅支持线性展频。SSCG 通过时钟内部集成电路调制频率的手段来达到抑制 EMI 峰值的目的。SSCG 不仅调制时钟源，其它的同步于时钟源的数据、地址和控制信号，在时钟展频的同时也一并得以调制，整体的 EMI 峰值都会因此减小，所以说，时钟展频是系统级的解决方案。这是 SSCG 相比其它抑制 EMI 措施的最大优势。

时钟展频有三个主要的控制参数：调制速度（Modulation Rate）、调制深度（Modulation Depth）和调制方式（Modulation Profile）

1. 调制速度

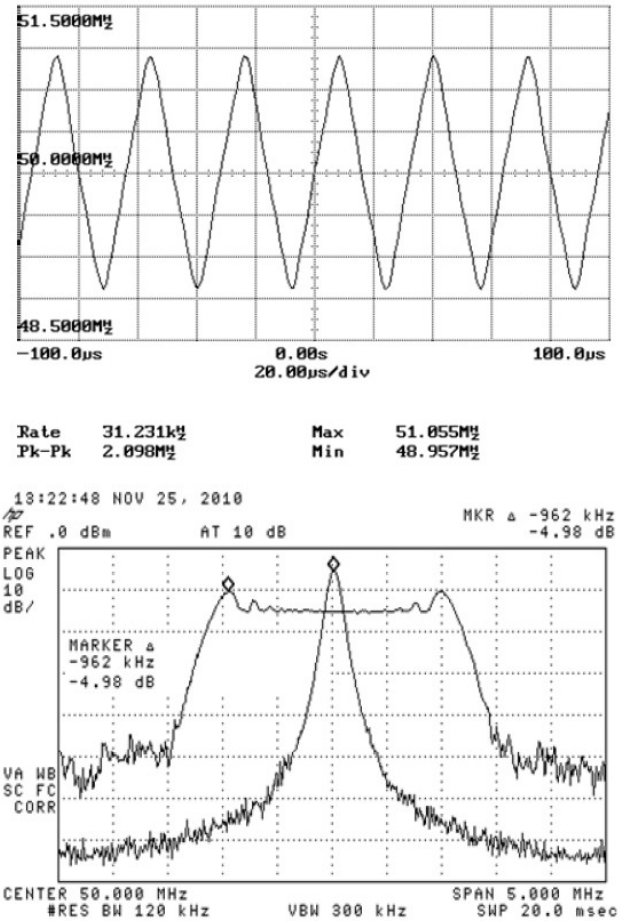
调制速度（MR）是指输出时钟频率 fo 在设定的调制频率范围内的变化速度。调制速度应远小于源时钟的频率 fc 以免引起时序问题（建立/保持时间等），同时应当高于人耳可识别的声音的频率范围（20Hz~20KHz）以免产生噪音。在实际应用中，调制速度一般选择 30KHz~120KHz。

2. 调制深度

调制深度是指展频后输出时钟频率 fo 以调制速度 MR 偏移源时钟频率 fc 的大小。调制深度以偏移（Δf）源时钟频率的百分比（%）来表示。调制深度决定降低 EMI 峰值的大小。通常调制深度越大，EMI 峰值越低。在应用时，需要合理预计系统可接受的频率调制范围。

3. 调制方式

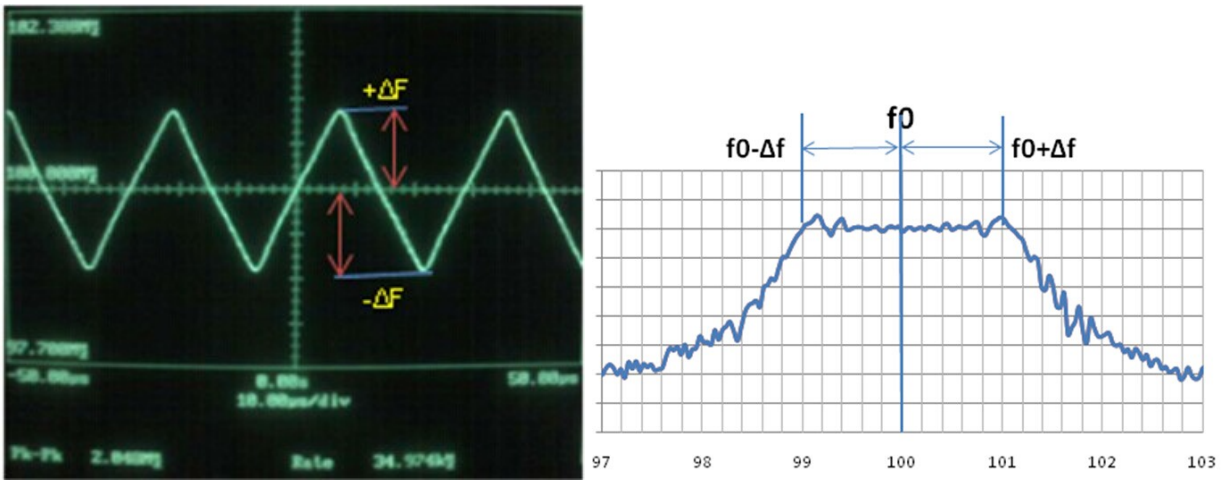
调制方式（Modulation Profile）决定 EMI 峰值的表现形式。Linear 和 Hershey Kiss 是 SSCG 常用的两种调制方式。线性调制相对简单，顾名思义，线性调制后输出的时钟频率是线性变化的。这种调制方法的缺点，如下图所示，输出频谱旁瓣比中间频率幅度高 1-2dB，如前文讨论，在任何频率 EMI 的失效也就意味着整个 EMI测试的失败。旁瓣的辐射峰值可能超出 SPEC 范围，设计者需要考虑到这点的影响。



按展频时钟相对源时钟偏移的不同，展频分为三类：中间展频; 向下展频; 向上展频。其中间展频是指：中间展频（Centre Spread）是指展频时钟的平均频率和源时钟频率相同的展频方式。在未调制输出时钟频率等于输入时钟频率的系统中，展频后输出时钟频率 fo 按调频方式（Linear 或 Hershey Kiss）决定的波形以 MR 速度，在（fc-Δf）到（fc+Δf）范围内变化，即：

$$fo = fc \pm \Delta f$$

例如，100MHz 时钟以±1%调制深度中间展频后，输出时钟的变化范围为 99MHz~101MHz。



时钟展频的一大弊端是不能用于对时钟精度敏感的应用，如以太网和 CAN 总线。在选择时钟展频和调制深度时，设计人员需要特别注意展频引入额外的 Jitter，并可能由此引起的建立/保持时间问题、高误码率和 PLL 失锁问题。

当展频的时钟输出到下游的 PLL 时，注意 PLL 表现为低通滤波器，即允许输入的低频部分通过，同时衰减其中的高频部分。展频时钟输入 PLL 时，PLL 可能出现无法锁住频率的问题。务必确保 PLL 必须能检测展频时钟的频率变化并允许展频时钟通过。以上取决于 PLL 的带宽，如果带宽太低，PLL 可能无法可靠地侦测输入时钟，造成侦测偏差，给系统引入更大的 Jitter

2 获取测试用 uboot 源代码

根据文档《S32G_Uboot_BSPxx_Vx-xxxxxxxx.doc》获得 Uboot 的源代码并创建 standalone 的编译环境，本文使用 BSP30 的 Uboot。

注意还有一种做法是 M7 去初始化 DDRC 控制器，然后把 Uboot 直接从储存设备放在外部 LPDDR4 中，而不是默认 BSP 的把 Uboot 放在内部 SRAM 中，如果那样，展频与改频代码需要在 M7 的代码中实现，这个不在本文讨论范围，本文讨论默认 BSP 使用的方法，启动后放在内部 SRAM 的 Uboot 来初始化 DDRC 的情况。

3 DDR_PLL 的改频

改频的目的主要是避免和一些敏感的如定位卫星信号重叠，如下：

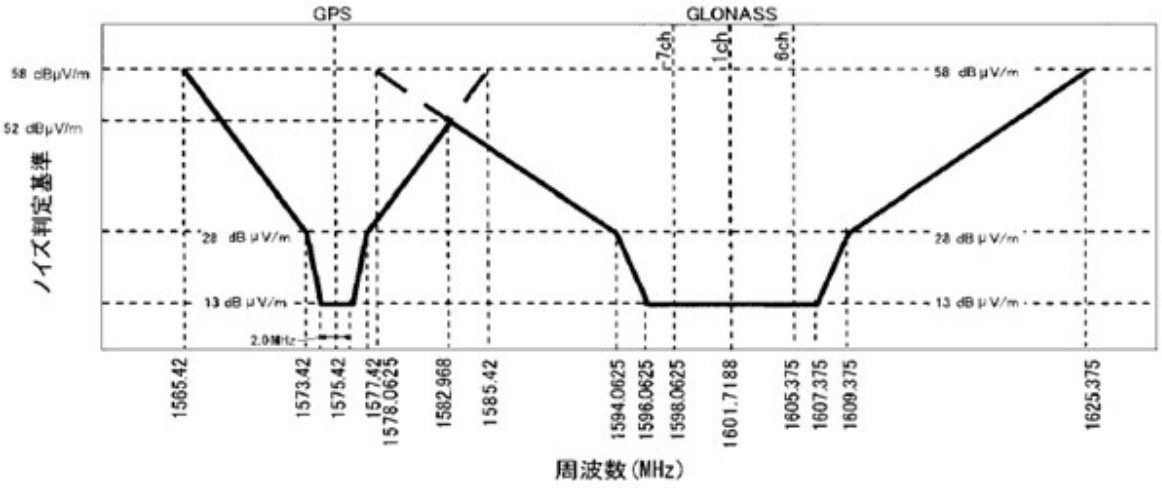


図3 GPS-GLONASS帯の目標値（電界アンテナ法 アベレージ検波）（補正値を用いた場合）

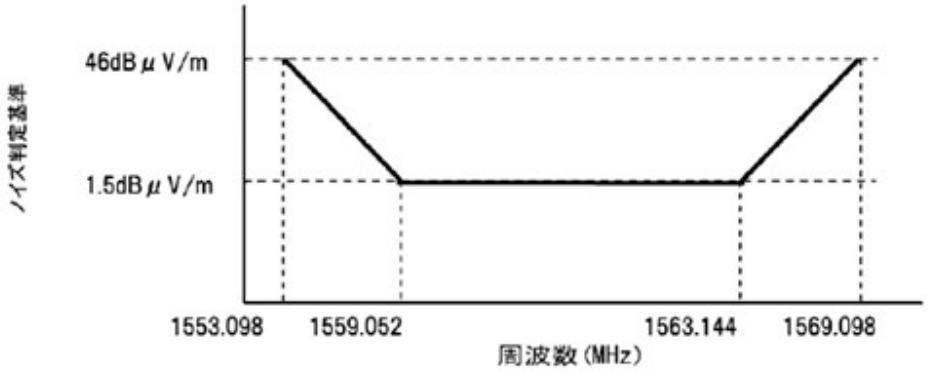


図4 中国BDS帯の目標値（電界アンテナ法 アベレージ検波）（補正値用いた場合）

- GPS 频段为 (1575.42 +/-10 MHz 1565.42~1585.42)。
- 北斗频段为(1561.098 +/-8 MHz 1553.098~1569.098)。
- 格星频段为(1578.0625~1625.375)。

可见北斗，GPS 和格星是重合并覆盖了从 1553~1625 的频段(倒梯形边缘有一些三角形的区域)。所以如果要完美的避开此频段，DDR clock 需要设置到比 1553 更小，(更大一般不支持，超出了 spec 要求的最高频率)。

S32G 的晶体频率是 40M，DDR clock 默认是 40X40=1600Mhz，所以需要修改为 40X38=1520Mhz(注意，此处只考虑了整数倍频的情况，如果加上小数倍频，是可以更加接近 1553M 的，但是考虑到改频后，往往还有展频的要求，所以取整数倍频比较合理，实事上在 Uboot 中，小数倍频目前的代码支持还有问题)。

另外请注意卫星信号的频谱要求是一个倒梯形，所以在每一种定位卫星信号频谱要求之间也有一定的空间可以通过调节改频和展频来满足更加接近 1600Mhz 的要求。

所以 DDR 频率可以做如下修改：

```
#define S32G274A_REV2_FC_DDR_PLL_VCO_FREQ                (1520 * MHZ)
#define S32G274A_REV2_FC_DDR_FREQ                        (760 * MHZ)
```

修改编译后，Uboot 启动后如下测试结果：

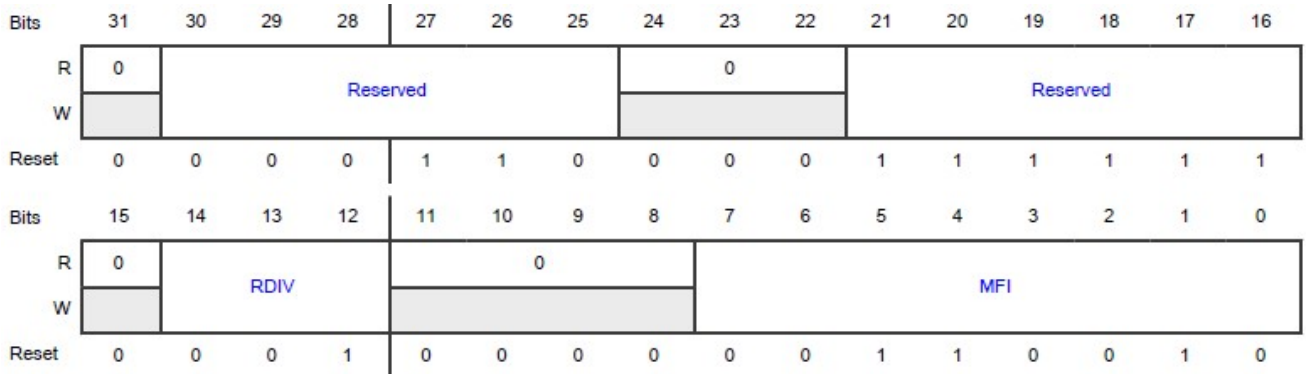
```
=> clk dump
…
DDR_PLL_MUX                : 40000000 Hz
DDR_PLL_VCO                : 1520000000 Hz
DDR_PLL_PHI0               : 760000000 Hz
MC_CGM5_MUX0               : 760000000 Hz
DDR                        : 760000000 Hz
…
=>md 0x40044008 1
40044008: 00001026
```

所以 MFI 由原来的 40(0x28)变成了 38(0x26)

DDR_PLL base address: 4004_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	PLL Control (PLLCR)	32	RW	8000_0000h
4h	PLL Status (PLLSR)	32	W1C	0000_0300h
8h	PLL Divider (PLLDV)	32	RW	0C3F_1032h
Ch	PLL Frequency Modulation (PLLFM)	32	RW	4000_0000h
10h	PLL Fractional Divider (PLLFD)	32	RW	0000_0000h
20h	PLL Clock Multiplexer (PLLCLKMUX)	32	RW	0000_0000h
80h	PLL Output Divider (PLLODIV_0)	32	RW	0000_0000h

Register	Offset
PLLDV	8h

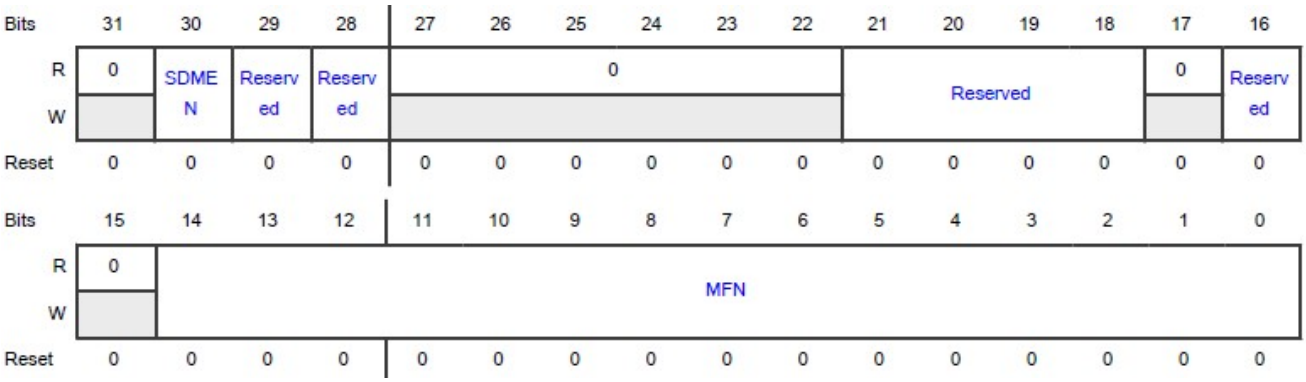


=> md 0x40044010 1

40044010: 40000000

而 MFN 依然=0。

Register	Offset
PLLFD	10h



Field	Function
14-0 MFN	Numerator Of Fractional Loop Division Factor Sets the numerator of the fractional loop division factor. You must write a value of less than 18432 to this field. When Fractional mode is disabled, you must write 000_0000_0000_0000b to this field.

根据 PLL 的倍频公式：

- Integer-only mode:

— When PLLDV[RDIV] is 0:

$$f_{pll\_VCO} = f pll\_ref \times PLLDV[MFI]$$

Equation 1. PLL VCO frequency in integer-only mode when PLLDV[RDIV] is 0

— When PLLDV[RDIV] is not 0:

$$f_{pll\_VCO} = \frac{f_{pll\_ref}}{PLLDV[RDIV]} \times PLLDV[MFI]$$

Equation 2. PLL VCO frequency in integer-only mode when PLLDV[RDIV] is not 0

Fractional mode:

— When PLLDV[RDIV] is 0:

$$f_{pll\_VCO} = f_{pll\_ref} \times \left(PLLDV[MFI] + \frac{PLLFD[MFN]}{18432}\right)$$

Equation 3. PLL VCO frequency in Fractional mode when PLLDV[RDIV] is 0

— When PLLDV[RDIV] is not 0:

$$f_{pll\_VCO} = \frac{f_{pll\_ref}}{PLLDV[RDIV]} \times \left(PLLDV[MFI] + \frac{PLLFD[MFN]}{18432}\right)$$

Equation 4. PLL VCO frequency in Fractional mode when PLLDV[RDIV] is not 0

所以出来的频率 fpll_vco=((fpll_ref_40M/(rdiv=1))*(MFI=38))=40*38=1520。

内核启动后的测试结果如下：

```
root@s32g274ardb:~# cat /sys/kernel/debug/clk/clk_summary |grep ddr
   ddrpll_sel                0     0     0    40000000          0     0  50000
      ddr_part_block         0     0     0    40000000          0     0  50000
      ddrpll_vco             0     0     0  1520000000          0     0  50000
      ddrpll_phi0            0     0     0   760000000          0     0  50000
   ddr                       0     0     0   760000000          0     0  50000
```

实际上使用频谱分析仪(如 agilent E4404B 9KHz~6.7GHz ESA-E serials spectrum analyzer)(未来增加)

4 DDR_PLL 的展频

参考 S32G 芯片手册说明：

26.7.4 Frequency modulation

In Frequency Modulation mode, PLL generates a frequency-modulated clock. The modulation depth and modulation frequency are calculated using the equations shown in Frequency modulation programming.

Write 0 to PLLFM[SPREADCTL] to select center-spread modulation. See Figure 130 that shows an example of center-spread modulation.

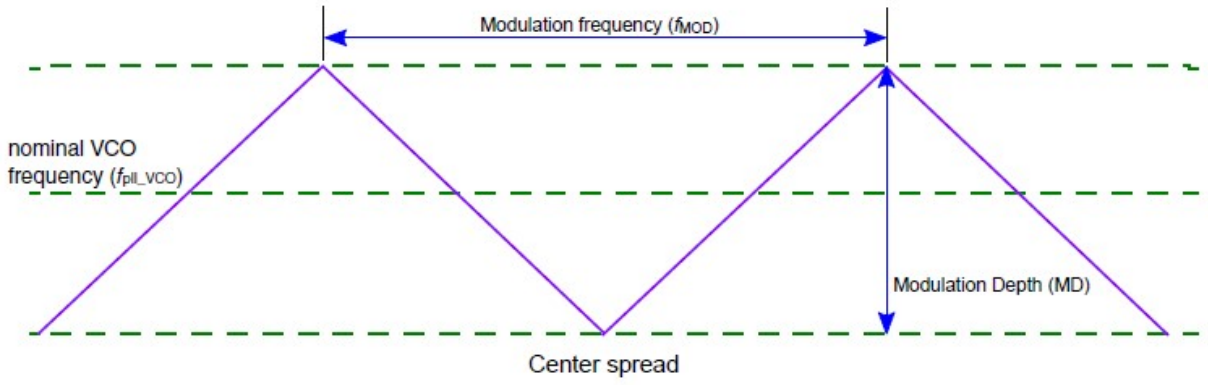


Figure 130. Frequency modulation

26.7.4.1 Frequency modulation programming

Modulation depth and modulation frequency programming uses step number (PLLFM[STEPNO]) and step size (PLLFM[STEPSIZE]). The table below shows variables used during calculations when programming PLL for frequency modulation.

Table 113. Variables for configuring modulation depth and frequency

Variable	Description
f_{REF}	Input clock frequency
f_{MOD}	Expected modulation frequency
MD	Expected modulation depth in percentage
LDF	Loop division factor

Use the following equations to configure PLL for frequency modulation.

$$LDF = PLLDV[MFI] + \frac{PLLFD[MFN]}{18432}$$

Equation 6. LDF

$$PLLFM[STEPNO] = \frac{f_{REF}}{\left(2 \times f_{MOD} \times PLLDV[RDIV]\right)}$$

Equation 7. Step number

$$PLLFM[STEPSIZE] = \frac{MD \times LDF}{100 \times PLLFM[STEPNO]} \times 18432$$

Equation 8. Step size

Frequency modulation is only possible if the condition shown in Equation 9 is met.

$$(\text{PLLFM[STEPSIZE]} \times \text{PLLFM[STEPNO]}) < 18432$$

Equation 9. Requirement to achieve FM

You must write 0 to PLLFM[SSCGBYP] and write 1 to PLLFD[SDMEN] to enable frequency modulation.

$$\text{Max (MD \%)} = \frac{(\text{Fref} \times 100)}{\text{PLLDV[RDIV]} \times \text{Fpll\_vco}}$$

Equation 10. Maximum possible modulation depth

CAUTION

The effective modulation depth may differ from the intended modulation depth because of rounding operations applied to PLLFM[STEPSIZE] and PLLFM[STEPNO].

26.8 Initialization information

Perform the following steps to initialize PLL:

1. Confirm that PLLODIV_*n*[DE] is 0 for all dividers.
2. Confirm that PLLCR[PLLPD] is 1.
3. Program PLLCLKMUX to select the appropriate reference clock.
4. Program the following as needed:
   - PLLDV
   - PLLFD
   - PLLFM to the desired value
5. Program PLLODIV_*n*[DIV] to the desired value.
6. Wait for the PLL reference clock to be stable.
7. Write 0 to PLLCR[PLLPD].
8. Wait for PLLSR[LOCK] to be 1.
9. Write 1 to PLLODIV_*n*[DE].

Perform the following steps to shut down PLL:

1. Write 0 to PLLODIV_n[DE] for all dividers.
2. Write 1 to PLLCR[PLLPD].

由于 S32G 仅支持中心展频，所以展频前需要先将中心频点向低移动至少半个展频深度，以防止在向上展频时超出 spec 要求范围，如下， DDR_VCO 频率不能高于 1600Mhz。

fPLL_DDR_VCO	DDR PLL VCO Frequency Range [6, 7]	1300	—	1600	MHz	without center-spread SSCG enabled	—
fPLL_DDR_PHI0	DDR PLL PHI0 Frequency [6, 8]	800	—	800	MHz	DDR_CLK (3200 MT/s), without center-spread SSCG enabled	—

目前 uboot 代码是自动推算倍频系数的，实际测试中发现如果是使用带小数的倍频会导致 uboot 启动 halt。所以此处仍以整数倍频来说明这种情况：

39X40Mhz=1560Mhz，38X40Mhz=1520Mhz。

所以我们需要设置的参数有：

- fMOD: 调制频率

参考 S32G 数据手册说明 ：

fPLL_MOD	Spread Spectrum Clock Modulation Frequency [13]	30	—	64	KHz	—	—

在 30~64KHZ 之间。

- MD:调制深度的百分数

参考 S32G 芯片手册说明：

$$\text{Max (MD \%)} = \frac{(\text{Fref} \times 100)}{\text{PLLDV[RDIV]} \times \text{Fpll\_vco}}$$

Equation 10. Maximum possible modulation depth

所以对于:

- 1560Mhz: Max(MD %)=40*100/1560=2.564%, 则展频深度为 1560X2.564%=40Mhz(峰峰值，则最大频率为 1560+40/2=1580Mhz<1600Mhz)
- 1520Mhz: Max(MD %)=40*100/1520=2.6316%, 则展频深度为 1560X2.6316%=40Mhz(峰峰值，则最大频率为 1520+40/2=1540Mhz<1600Mhz)

注意一下在芯片手册中 MD 按照图示是峰峰值，所以计算最高频率要除 2。另外如果有在最大展频深度情况下，频率会高出 1600Mhz 的时候，最大展频需要减少以防止超过 spec 要求。

所以调制频率是从 30~64Khz，我们选择 fref=40Mhz 的整数除倍数 fmod=40K: 则：

对于改频后再做展频，改频为 1520Mhz(mfi=38)

ldf=mfi(38)+(mfn(0)/18432)=mfi=38;

stepno=Fref(40*MHZ)/(2*S32G274A_REV2_SSC_MOD(40Khz)*rdiv(1))=500

stepsize=(S32G274A_REV2_SSC_MD(26)*ldf(38)*18432)/(1000*stepno(500))~=36(36.421632) //千分之 26 的最大展频，保证整数乘。

由于 stepno/stepsize 只能取整数值，所以当配置为 MD=2.6%, Fmod=40K 时，可见对于 1520(mfi=38)的 VCO，圆整为 stepsize=36 较为接近。

而对于 1560 的展频，频率为 1576(mfi=39)

ldf=mfi(39)+mfn(0)/18432=39

stepno=Fref(40*MHZ)/(2*S32G274A_REV2_SSC_MOD(40Khz)*rdiv(1))=500

stepsize=(S32G274A_REV2_SSC_MD(25)*ldf(39)*18432)/(1000*stepno(500))~=35(35.9424)

相对更加不接近整数(注意，因为 C 语言圆整的原因，对于大于 0.5 的数值，实际数值还是整数值，分数部分都圆整了，所以可能需要+1 来更加接近要求值，但是要注意最大展频值不能超界，需要验证)

注意：

- 实际上是用 setpno/setpsize 来推导出 MD 和 Fmod 的，所以对寄存器的整数的设置会影响到 MD 和 Fmod 的值，比如说：

  对 1520Mhz+2.57%展频：S32G274A_REV2_SSC_MD=36X100X500/(18432*38)=2.57，而 1520 展频深度要求是如上 2.6316%,满足要求。

  对 1560Mhz+2.43%展频：S32G274A_REV2_SSC_MD=35X100X500/(18432*39)=2.43，而 1560 展频深度要求是如上 2.564%,满足要求。

  所以在 stepno 和 stepsize 会整数值的情况下，要让 MD 和 Fmod 符合 spec 要求，而在符合要求的情况下，尽量让 Fmod 取整数。

- 如何选取展频深度与展频步数，要与实际的 EMI 测试结果配合，没有一定之规，需要实际测试并自行决定，本文也不做规定。

展频加改频为 1520Mhz+2.57%的测试结果如下：

- Uboot 启动

```
=> md 0x40044000 7
40044000: 00000000 00000704 00001026 002401f4    ........&.......
40044010: 40000000 00000000 00000000             ...@........
```

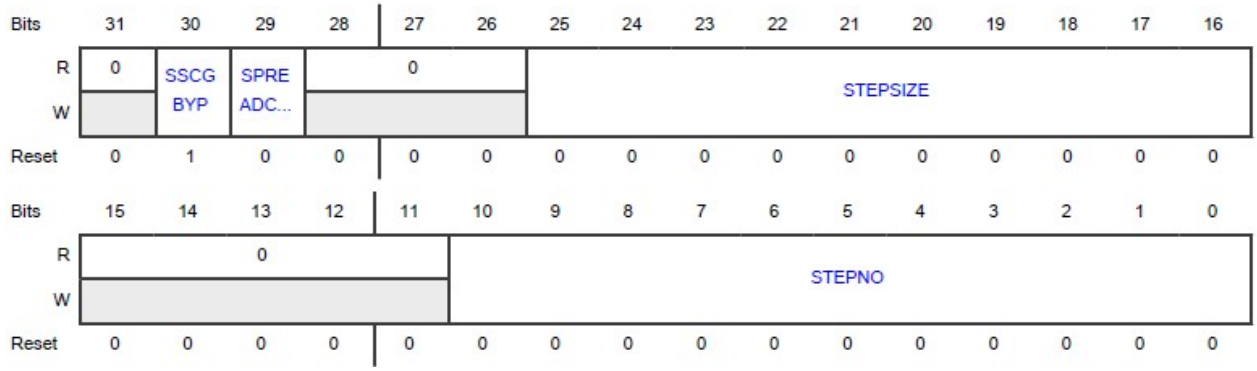
所以是：

Register	Offset
PLLFM	Ch

Function

Configures PLL frequency modulation parameters.

Diagram



Field	Function
30 SSCGBYP	Frequency Modulation (Spread Spectrum Clock Generation) Bypass Bypasses frequency modulation. 0b - Not bypassed 1b - Bypassed
29 SPREADCTL	Modulation Type Selection Indicates that the modulation is centered around the nominal frequency. 0b - Centered around nominal frequency 1b - Reserved

SSCGBYP=0 展频功能不 bypass

SPREADCTL=0 设置为中心展频

25-16 STEPSIZE	Frequency Modulation Step Size Provides the step size for modulation depth and frequency in Frequency Modulation mode (see Frequency modulation).
15-11 —	Reserved
10-0 STEPNO	Number Of Steps Of Modulation Period Or Frequency Modulation Provides the number of steps to achieve modulation depth in Frequency Modulation mode (see Frequency modulation).

SETPNO=0x1f4=500 步

STEPSIZE=0x24=36:

相比较不展频改频之前的 PLL 相关寄存器值如下：

```
=> md 0x40044000 7
40044000: 00000000 00000704 00001028 40000000    ........(......@
40044010: 40000000 00000000 00000000             ...@........
=>
```

展频加改频为 1560Mhz+2.43%的测试结果如下：

代码修改为：

```
#ifdef CONFIG_EMI_FC
#define S32G274A_REV2_FC_DDR_PLL_VCO_FREQ (1560 * MHZ) //(1560 * MHZ)
#define S32G274A_REV2_FC_DDR_FREQ (780 * MHZ) //(780 * MHZ)
#ifdef CONFIG_EMI_SSC
#define KHZ (1000UL)
#define S32G274A_REV2_SSC_MD (25) //26%0 for 1520Mhz, 25%0 for 1560Mhz,
#define S32G274A_REV2_SSC_MOD (40 * KHZ)
#endif
#endif
```

- Uboot 启动

```
=> clk dump
…
DDR_PLL_MUX                    : 40000000 Hz
DDR_PLL_VCO                    : 1560000000 Hz
```

```
DDR_PLL_PHI0                : 780000000 Hz
MC_CGM5_MUX0                : 780000000 Hz
DDR                         : 780000000 Hz

=> md 0x40044000 7
40044000: 00000000 00000704 00001027 002301f4    ........'.......
40044010: 40000000 00000000 00000000             ...@........
```

实际上使用频谱分析仪(如 agilent E4404B 9KHz~6.7GHz ESA-E serials spectrum analyzer)(未来增加)

5 总结修改后的源代码

Uboot 的 ddr clock 初始化流程是：

```
init_sequence_f(common/board_f.c)
|->arch_cpu_init(arch/arm/cpu/armv8/s32/cpu.c)
|   |-> enable_early_clocks(drivers/clk/s32/early_clocks.c)
|   |   |->enable_ddr_clock
|   |   |   |->s32gen1_enable
最终会从源到头找到 PLL
|   |   |   |   |->enable_module
|   |   |   |   |   |->enable_pll
|   |   |   |   |   |   |->program_pll
```

修改的源代码如下:

Configs/s32g274ardb2_defconfig

```
CONFIG_EMI_FC=y //控制是否打开改频
CONFIG_EMI_SSC=y //控制是否打开展频，默认我们都打开。
```

Drivers/clk/Kconfig

```
config EMI_FC
        bool "enable S32G DDR PLL frequency changing "
        help
          Enable S32G DDR PLL frequency changing.

config EMI_SSC
        bool "enable S32G DDR PLL frequency spread spectrum clocking "
```

```
    help
      Enable S32G DDR PLL frequency spread spectrum clocking.

config EMI_DEBUG
    bool "enable S32G DDR PLL EMI DEBUG "
    help
      Enable S32G DDR PLL EMI DEBUG.
```

uboot/include/dt-bindings/clock/s32gen1-clock-freq.h

```
#ifdef CONFIG_EMI_FC
#define S32G274A_REV2_FC_DDR_PLL_VCO_FREQ (1520 * MHZ) //(1560 * MHZ)
#define S32G274A_REV2_FC_DDR_FREQ (760 * MHZ) //(780 * MHZ)
#ifdef CONFIG_EMI_SSC
#define KHZ (1000UL)
#define S32G274A_REV2_SSC_MD (26) //26%0 for 1520Mhz, 25%0 for 1560Mhz,
#define S32G274A_REV2_SSC_MOD (40 * KHZ)//modulation frequency range is 30K to 64K from datasheet
#endif
#endif
```

Uboot/drivers/clk/s32/early_clocks.c

```
static int enable_ddr_clock(void)
{…
        ddr_pll_freq = S32GEN1_DDR_PLL_VCO_FREQ;
        ddr_freq = S32GEN1_DDR_FREQ;
    }
#if defined(CONFIG_EMI_FC)
   ddr_pll_freq = S32G274A_REV2_FC_DDR_PLL_VCO_FREQ; //改频或改频作为展频基础
   ddr_freq = S32G274A_REV2_FC_DDR_FREQ;
#else
…
```

Uboot/arch/arm/include/asm/arch-s32/s32-gen1/mc_cgm_regs.h

```
//johnli for emi //设置FM寄存器的宏
//johnli for emi
#if defined(CONFIG_EMI_SSC)

#define PLLDIG_PLLFM_SSCGBYP_SET(val )(PLLDIG_PLLFM_SSCGBYP_MASK & \
                                        ((val)<< PLLDIG_PLLFM_SSCGBYP_OFFSET))


#define PLLDIG_PLLFM_SPREADCTL_OFFSET       (29)
#define PLLDIG_PLLFM_SPREADCTL_MASK  (0x20000000)
#define PLLDIG_PLLFM_SPREADCTL_SET(val)     (PLLDIG_PLLFM_SPREADCTL_MASK & \
                                        ((val)<< PLLDIG_PLLFM_SPREADCTL_OFFSET))

#define PLLDIG_PLLFM_STEPSIZE_OFFSET    (16)
#define PLLDIG_PLLFM_STEPSIZE_MASK      (0x03FF0000)
#define PLLDIG_PLLFM_STEPSIZE_SET(val)  (PLLDIG_PLLFM_STEPSIZE_MASK & \
                                            ((val)<< PLLDIG_PLLFM_STEPSIZE_OFFSET))
```

```
#define PLLDIG_PLLFM_STEPNO_OFFSET      (0)
#define PLLDIG_PLLFM_STEPNO_MASK        (0x000007FF)
#define PLLDIG_PLLFM_STEPNO_SET(val)    (PLLDIG_PLLFM_STEPNO_MASK & \
                                            ((val)<< PLLDIG_PLLFM_STEPNO_OFFSET))
#endif
//end
```

Uboot\drivers\clk\s32\enable_clk.c

```
static int program_pll(struct s32gen1_pll *pll, void *pll_addr,
                       struct s32gen1_clk_priv *priv, u32 clk_src)
{
        …
        u32 rdiv = 1, mfi, mfn;
        //johnli for emi
#if defined(CONFIG_EMI_SSC)
        u32 stepno, stepsize,;
        float ldf;
#endif
        //end
…
        writel(PLLDIG_PLLFD_MFN_SET(mfn) |
            PLLDIG_PLLFD_SMDEN, PLLDIG_PLLFD(pll_addr));  // PLLDIG_PLLFD_SMDEN already set to 1
        //johnli for emi
        #if defined(CONFIG_EMI_SSC)

if(0x40044000 == pll_addr)//仅对DDR PLL展频
        {

    ldf=(float)mfi+((float)mfn/18432); //for 1520Mhz clock, mfi=38(38X40=1520), mfn=0, so ldf=38
    stepno=(40*MHZ)/(2*S32G274A_REV2_SSC_MOD*rdiv); // 40M/(2X40K*1)=500
    stepsize=(uint)((S32G274A_REV2_SSC_MD*ldf*18432)/(1000*stepno)); // mdX38X18432/(100X500)= ?
//md/1000

        writel(PLLDIG_PLLFM_SSCGBYP_SET(0)|PLLDIG_PLLFM_SPREADCTL_SET
(0)|PLLDIG_PLLFM_STEPSIZE_SET (stepsize)|PLLDIG_PLLFM_STEPNO_SET(stepno),
PLLDIG_PLLFM(pll_addr));
}
        #endif
        //end
        ret = adjust_odiv_settings(pll, pll_addr, priv, odivs_mask, old_vco);
…
```