专题7-系统时钟初始化

一、概念解析

1.1、时钟脉冲信号

时钟脉冲信号：按一定的电压幅度，一定的时间间隔 连续发出的脉冲信号。时钟脉冲信号是时序逻辑的基础，它用于决定逻辑单元中的状态何时更新。数字芯片中众多的晶体管都工作在开关状态，它们的导通和关断动作无不是按照时钟信号的节奏进行的。

1.2、时钟频率

时钟脉冲频率：在单位时间（如1秒） 内产生的时钟脉冲个数。

1.3、时钟源

1.3.1、晶振

晶振全称晶体振荡器，是用石英晶体经精密切割磨 削并镀上电极焊上引线做成。这种晶体有一个很重要的特性，如果给他通电，他就会产生机械振荡，他们有一个很重要的特点，其振荡频率与他们的形状，材料，切割方向等密切相关。由于石英晶体化学性能非常稳定，热膨胀系数非常小，其振荡频率也非常稳定，由于控制几何尺寸可以做到很精密，因此，其谐振频率也很准确。

晶体振荡器时钟的优点包括结构简单和噪声低，以及可为客户提供精确的定制频率等方面；但另一方面，它的缺点也比较明显，例如其频率仅由晶体决定，通常是特定晶体被制成客户所需的振荡器，导致生产成本高、交货周期较长，不利于客户加快产品上市时间，而且难以获得非标准的频率。

1.3.2、锁相环

PLL(锁相环)合成器是一种更为复杂的系统 时钟源。通用PLL合成器需要一个外部晶体并包含一个能够对晶体的特定频率加倍或分频的集成锁相环（PLL）电路。

1.3.3、对比

典型的系统时钟振荡器源通常采用石英晶振，而更复杂的系 统时钟振荡器源则是由PLL合成器提供。

1. 对于特定的时钟频率，采用PLL合成器可使用较便宜以 及较低频率晶振来代替昂贵的高频晶振；

2. 对于需要多个时钟频率的系统，采用PLL合成器通过分 频即可实现，而此时采用晶振模块则需要多个不同频率的晶振。

因此相对于晶体振荡器模块，通过PLL合成器提供精确时钟 具有成本更低、占板面积更小等一系列优点。

二、时钟体系（6410）

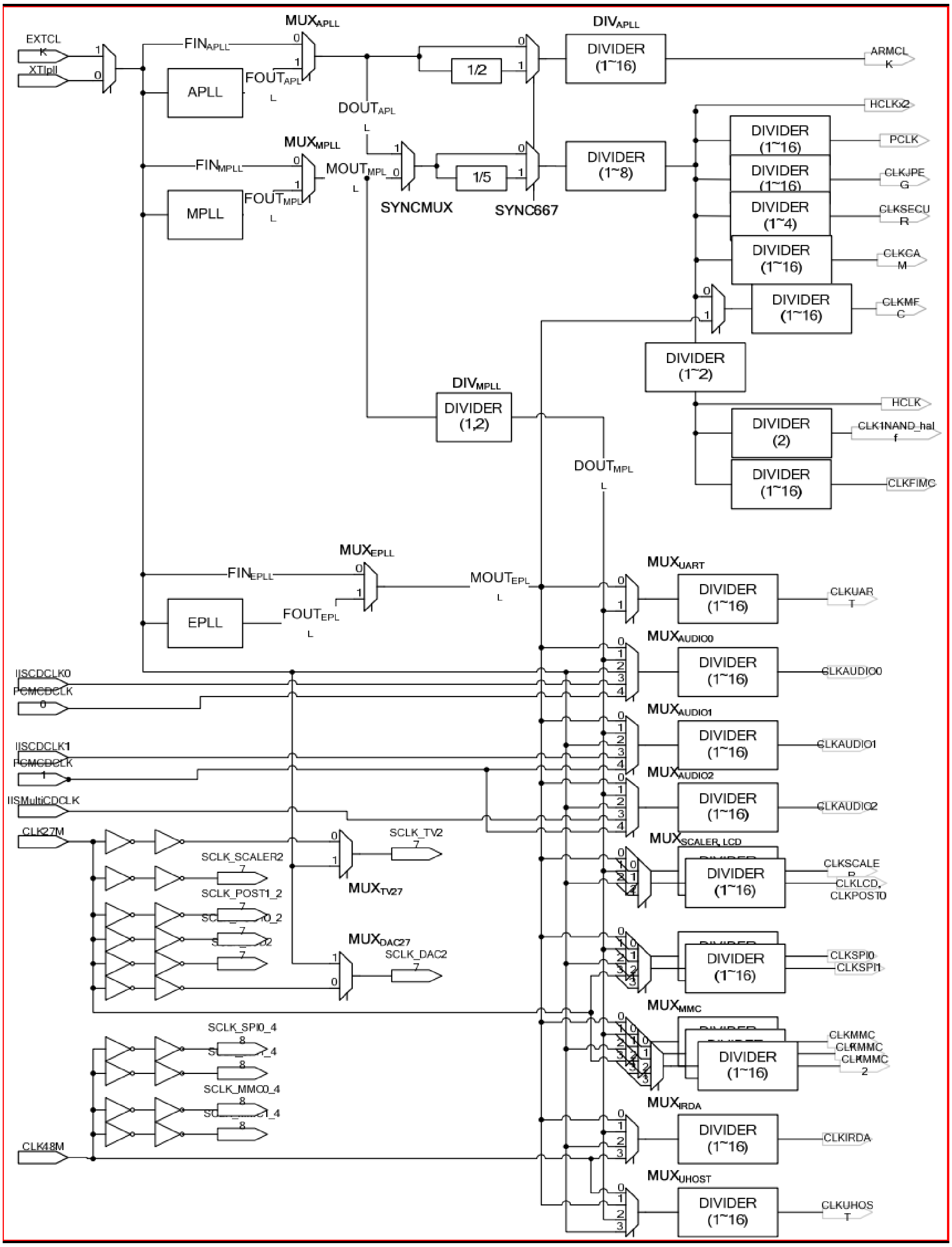


Figure 3-2. The block diagram of clock generator

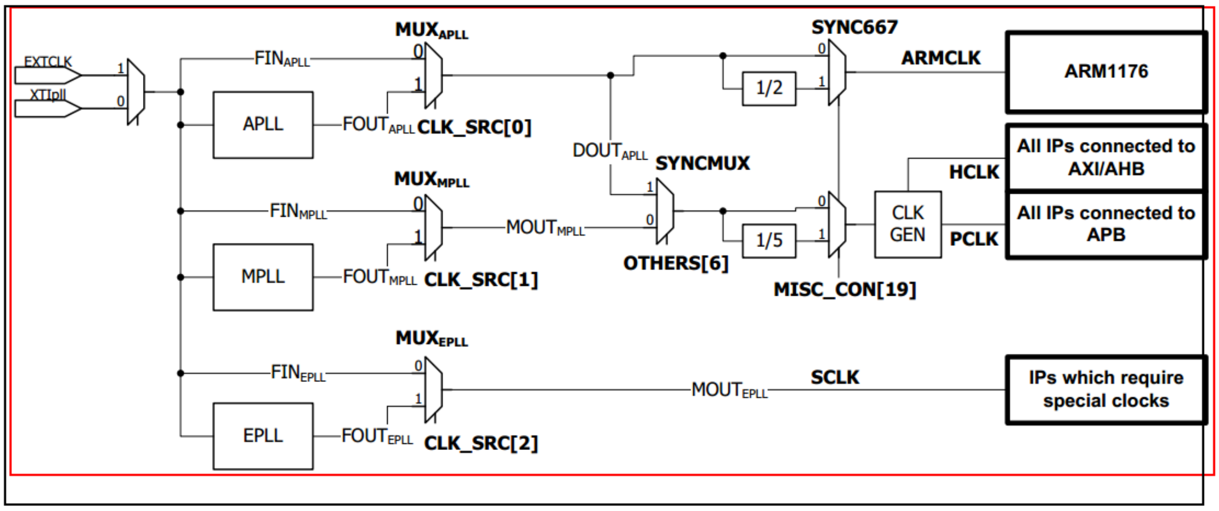


Figure 3-4. Clock generation from PLL outputs

2.1、晶振
12MHz

2.2、3个PLL
APLL，MPLL，EPLL

2.3、时钟

时钟	应用场合	应用举例	所属PLL
ACLK	处理器	ARM11	APLL
HCLK	AHB总线	LCD,DMA...	MPLL
PCLK	APB总线	UART,GPIO...	MPLL
SCLK	USB总线	USB主从口	EPLL

AHB(Advanced High performance Bus)系统总线
APB(Advanced Peripheral Bus)外围总线

三、上电时序图

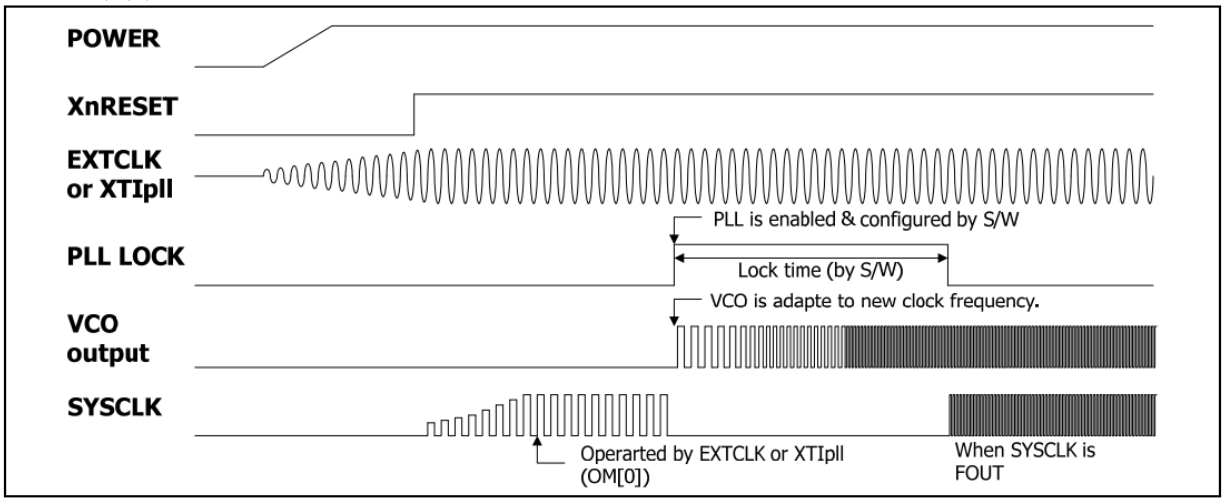
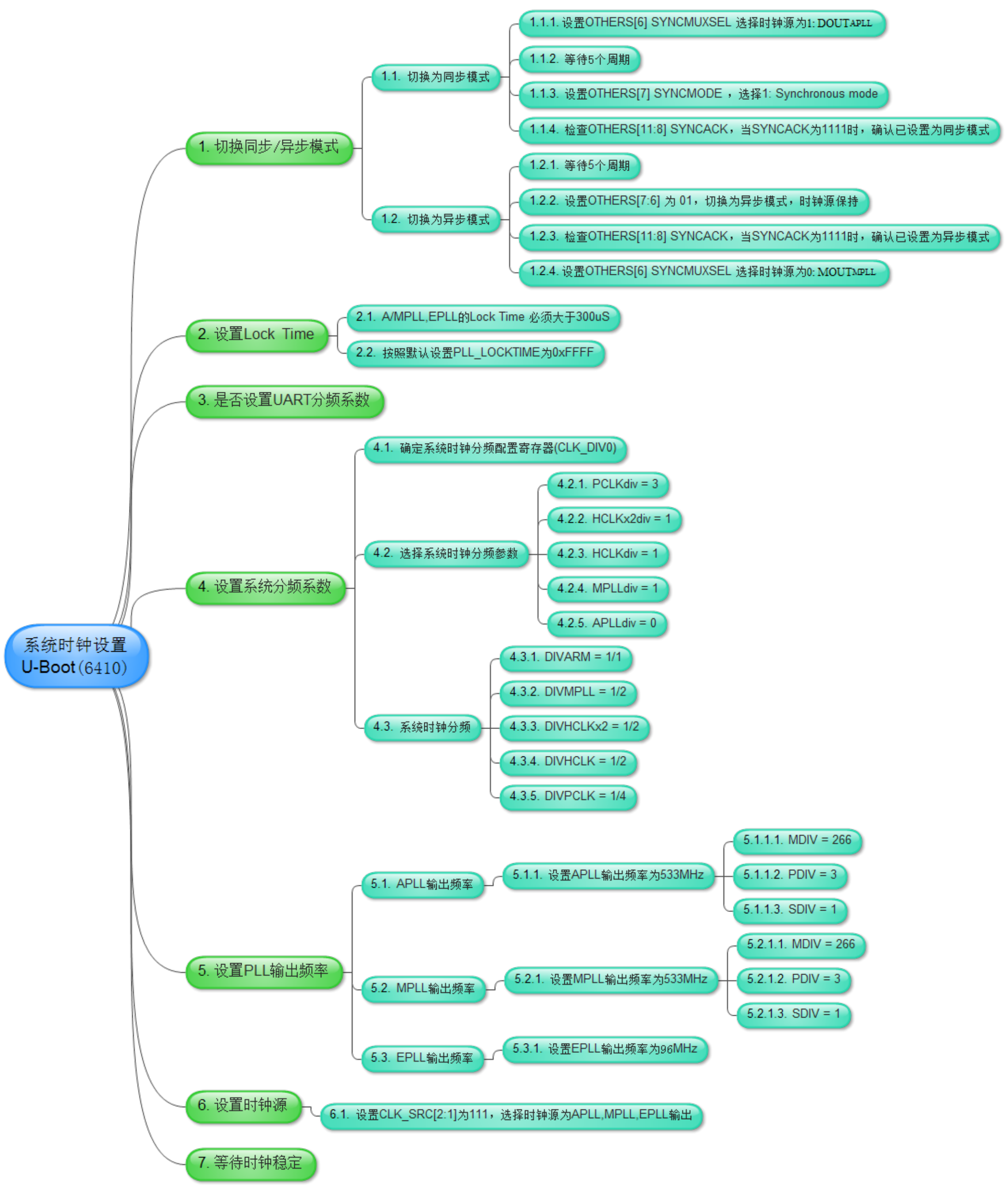

POWER
XnRESET
EXTCLK or XTIpll
PLL is enabled & configured by S/W
PLL LOCK
Lock time (by S/W)
VCO is adapte to new clock frequency.
VCO output
SYSCLK
Operarted by EXTCLK or XTIpll (OM[0])
When SYSCLK is FOUT

Figure 3-17. Power-on reset sequence

3.1、锁相环 Lock Time
用于锁相环配置后时钟稳定。

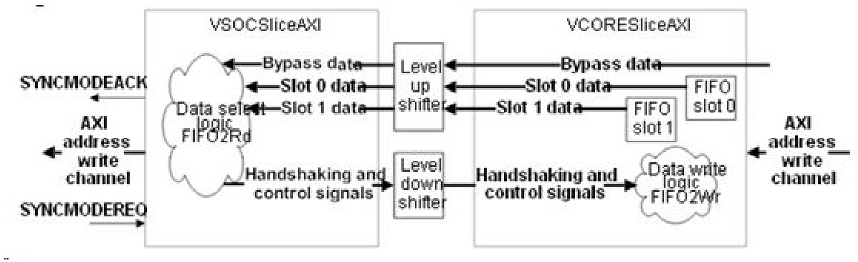
四、配置流程
4.1、U-Boot下时钟配置流程


系统时钟设置
U-Boot(6410)
1. 切换同步/异步模式
1.1. 切换为同步模式
1.1.1. 设置OTHERS[6] SYNCMUXSEL 选择时钟源为1: DOUTAPLL
1.1.2. 等待5个周期
1.1.3. 设置OTHERS[7] SYNCMODE ，选择1: Synchronous mode
1.1.4. 检查OTHERS[11:8] SYNCACK，当SYNCACK为1111时，确认已设置为同步模式
1.2. 切换为异步模式
1.2.1. 等待5个周期
1.2.2. 设置OTHERS[7:6] 为 01，切换为异步模式，时钟源保持
1.2.3. 检查OTHERS[11:8] SYNCACK，当SYNCACK为1111时，确认已设置为异步模式
1.2.4. 设置OTHERS[6] SYNCMUXSEL 选择时钟源为0: MOUTMPLL
2. 设置Lock Time
2.1. A/MPLL,EPLL的Lock Time 必须大于300uS
2.2. 按照默认设置PLL_LOCKTIME为0xFFFF
3. 是否设置UART分频系数
4. 设置系统分频系数
4.1. 确定系统时钟分频配置寄存器(CLK_DIV0)
4.2. 选择系统时钟分频参数
4.2.1. PCLKdiv = 3
4.2.2. HCLKx2div = 1
4.2.3. HCLKdiv = 1
4.2.4. MPLLdiv = 1
4.2.5. APLLdiv = 0
4.3. 系统时钟分频
4.3.1. DIVARM = 1/1
4.3.2. DIVMPLL = 1/2
4.3.3. DIVHCLKx2 = 1/2
4.3.4. DIVHCLK = 1/2
4.3.5. DIVPCLK = 1/4
5. 设置PLL输出频率
5.1. APLL输出频率
5.1.1. 设置APLL输出频率为533MHz
5.1.1.1. MDIV = 266
5.1.1.2. PDIV = 3
5.1.1.3. SDIV = 1
5.2. MPLL输出频率
5.2.1. 设置MPLL输出频率为533MHz
5.2.1.1. MDIV = 266
5.2.1.2. PDIV = 3
5.2.1.3. SDIV = 1
5.3. EPLL输出频率
5.3.1. 设置EPLL输出频率为96MHz
6. 设置时钟源
6.1. 设置CLK_SRC[2:1]为111，选择时钟源为APLL,MPLL,EPLL输出
7. 等待时钟稳定

4.2、切换同步/异步模式

4.2.1、同步异步模式概念

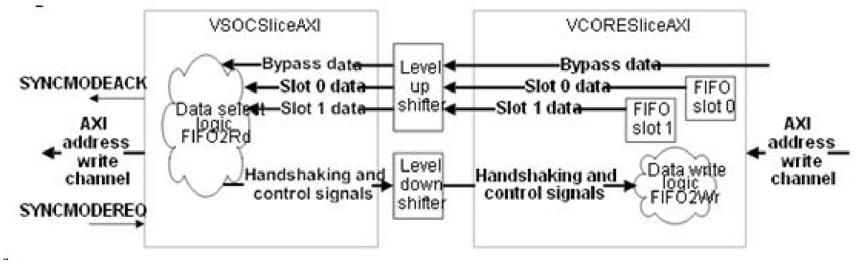
在AXI/AHB/APB总线配置有一个同步或异步模式选择，通过OTHER[6] = 0，或OTHER[6] = 1来确定是同步还是异步模式，关于同步或异步模式在《ARMApplication Note 172》有描述，当需要系统最大性能需求时，6410内部的AXI RegisterSlices工作在同步模式，AXI Register Slices由不同深度的FIFO组成。工作同步模式就是AXI Register Slices（包括VCORE和VSOC两边的），AXI RegisterSlices数据不先经过FIFO缓存，而是直接由 VSOC AXI与VCORE AXI经过“Level shift r wrappers”直接到达对方。



摘自《ARM1176 Technical Reference manual》

当AXI地址写通道从VCORE到VSOC，异步模式下，VCORE需要经过FIFO slot0 和FIFO slot1缓存，若为同步模式，则不需要FIFO缓存。若要从异步模式切换到同步模式，使SYSCMODEREQ = 1，等待当SYNCMODEACK = 1，表示FIFO内容已经被“抽干”并且接下来FIFO都不起作用，即已经切换同步模式。

从同步模式切换到异步模式：使SYSCMODEREQ= 0，等待当SYNCMODEACK= 0，即表示已经切换到异步模式。



摘自《ARM1176 Technical Reference manual》

Switchingbetween asynchronous and synchronous mode is controlled by the SYNCMODEREQinput on each register slices. This input must be driven to a logic 1 to requestsynchronous mode, and logic 0 for asynchronous mode. The AXI AsynchronousRegister Slice also has an output SYNCMODEACK to indicate the currentoperating status. Logic 1 on SYNCMODEACK indicates that all FIFOs within the slices havebeen drained, and the slice is operating in synchronous mode. Logic 0 on SYNCMODEACKindicates that the slice operates asynchronously.…

摘自《ARM Application Note 172》

在6410芯片中同步模式切换是通过设置OTHER[6]=1，等待当OTHER[11:8]=1111，表示已经切换到同步模式，之所以需要4bit 的SYNCMODECACK，我觉得可能是与ARM1176JZF-S内核有Peripheral AXI、DMA AXI、Data AXI及Instruction AXI 等四组VCORE AXI Register Slices有关。

CMU（clock management unit）时钟管理单元

3.4.2.14 Others control register

REGISTER	ADDRESS	R/W	DESCRIPTION	RESET VALUE
OTHERS	0x7E00_F900	R/W	Others control register	0x0000_801E

OTHERS	BIT	DESCRIPTION	RESET VALUE
RESERVED	[31:24]	RESERVED	0x0000
STABLE COUNTER TYPE	[23]	Indicate OSC_STABLE, PWR_STABLE counter type 0 : Exponential Scale, 1 : Set by SFR	0
RESERVED	[22:17]	DO NOT CHANGE	0x0000
USB_SIG_MASK	[16]	USB signal mask to prevent unwanted leakage. (This bit must set before USB PHY is used.)	0
RESERVED	[15:14]	RESERVED	0x2
CLEAR_DBGACK	[13]	Clear DBGACK signal when this field has 1. ARM1176 asserts DBGACK signal to indicate the system has entered Debug state. If DBGACK is asserted, this state is store in SYSCON until software clear it using this field.	0
CLEAR_BATF_INT	[12]	Clear interrupt caused by battery fault when this bit is set.	0
SYNCACK	[11:8]	SYNC mode acknowledge (Read Only)	0x0
SYNCMODE	[7]	SYNCMODEREQ to ARM 0: Asynchronous mode, 1: Synchronous mode	0
SYNCMUXSEL	[6]	SYS CLOCK SELECT IN CMU 0: MOUT$_{MPLL}$, 1: DOUT$_{APLL}$	0
RESEVED	[5:3]	DO NOT CHANGE	0x3
RESERVED	[2]	Should be ‘1’	1
RESERVED	[1]	Should be ‘1’	1
CP15DISABLE	[0]	Disables write asses to some system control processor registers of ARM1176. (0: enable, 1: disable)	0

4.2.2、切换为同步模式

从异步模式切换为同步模式，需要先将时钟源选择同一个时钟源，然后进行模式切换。并检查模式切换是否成功。

```
#define SYS_CTL_BASE 0x7e00f000
#define OTHERS_OFFSET 0x900
#define CLK_DIV0_OFFSET 0x20
#define CLK_DIV_VAL ((3<<12)|(1<<9)|(1<<8)|(1<<4)|(0<<0))
#define APLL_VAL ((1<<31)|(266<<16)|(3<<8)|(1<<0))
#define MPLL_VAL ((1<<31)|(266<<16)|(3<<8)|(1<<0))
#define APLL_CON_OFFSET 0x0c
#define MPLL_CON_OFFSET 0x10
#define CLK_SRC_OFFSET 0x1c
system_clock_init:
 ldr r0, =SYS_CTL_BASE

#ifdef CONFIG_SYNC_MODE /* SYNC Mode */
 ldr r1, [r0, #OTHERS_OFFSET]
 mov r2, #0x40
 orr r1, r1, r2
 str r1, [r0, #OTHERS_OFFSET]

 nop
 nop
 nop
 nop
 nop

 ldr r2, =0x80
 orr r1, r1, r2
 str r1, [r0, #OTHERS_OFFSET]

check_syncack:
 ldr r1, [r0, #OTHERS_OFFSET]
 ldr r2, =0xf00
 and r1, r1, r2
 cmp r1, #0xf00
 bne check_syncack
```

4.2.2、切换为异步模式

从同步模式切换为异步模式，需要先进行模式切换，并检查模式切换是否成功，然后切换时钟源。

```
#else /* ASYNC Mode */
 nop
 nop
 nop
 nop
 nop
 ldr r1, [r0, #OTHERS_OFFSET]
 bic r1, r1, #0xc0
 orr r1, r1, #0x40
 str r1, [r0, #OTHERS_OFFSET]

wait_for_async:
 ldr r1, [r0, #OTHERS_OFFSET]
 and r1, r1, #0xf00
 cmp r1, #0x0
 bne wait_for_async

 ldr r1, [r0, #OTHERS_OFFSET]
 bic r1, r1, #0x40
 str r1, [r0, #OTHERS_OFFSET]
#endif
```

4.3、设置Lock Time

因为一般使用默认的时间，所以一般不修改。

APLL_LOCK / MPLL_LOCK / EPLL_LOCK	BIT	DESCRIPTION	RESET VALUE
RESERVED	[31:16]	RESERVED	0x0000
PLL_LOCKTIME	[15:0]	Required period to generate a stable clock output	0xFFFF

PLL	Max. LockTime(us)	PLL_LOCK(FIN:12MHz)
APLL	300	0xE11
MPLL	300	0xE11
EPLL	300	0xE11

PLL_CON register controls the operation of each PLL. If ENABLE bit is set, the corresponding PLL generates output after PLL locking period. The output frequency of PLL is controlled by the MDIV, PDIV, SDIV, and KDIV values. APLL_LOCK, MPLL_LOCK, and EPLL_LOCK fields denote the number of external clock. User can adjust this fields, which must be larger than maximum lock time (A/MPLL is 300us, EPLL is 300us).

4.4、设置系统时钟分频系数

4.4.1、系统时钟分频配置寄存器

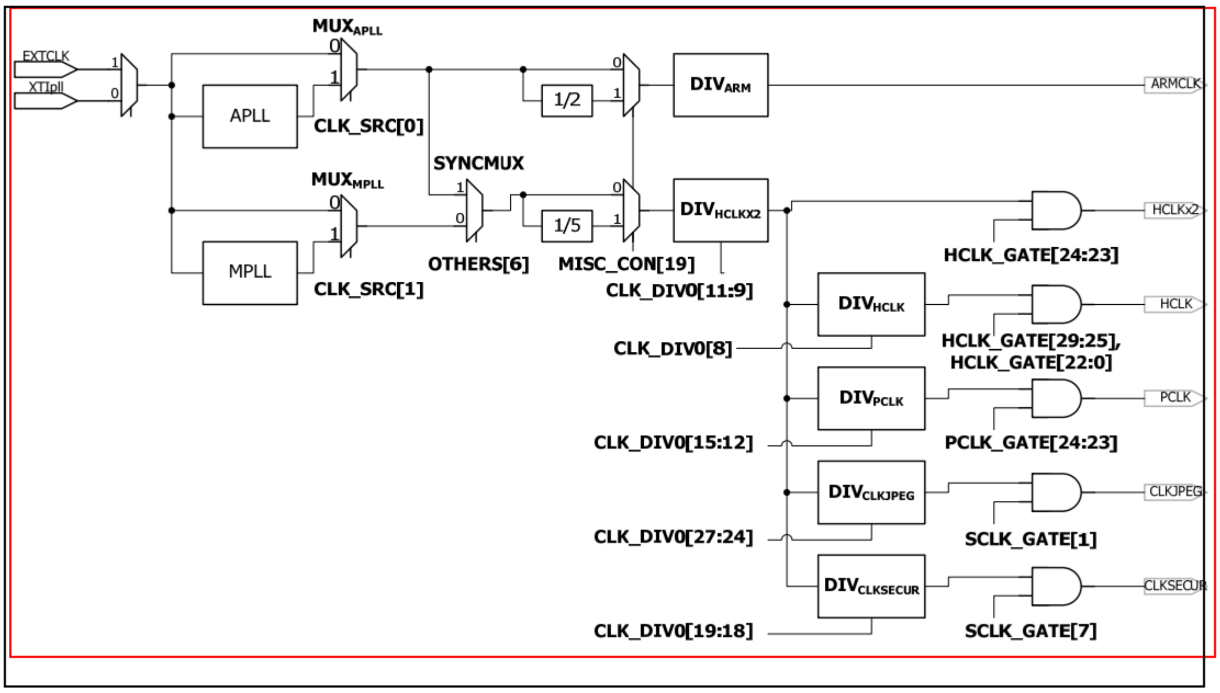
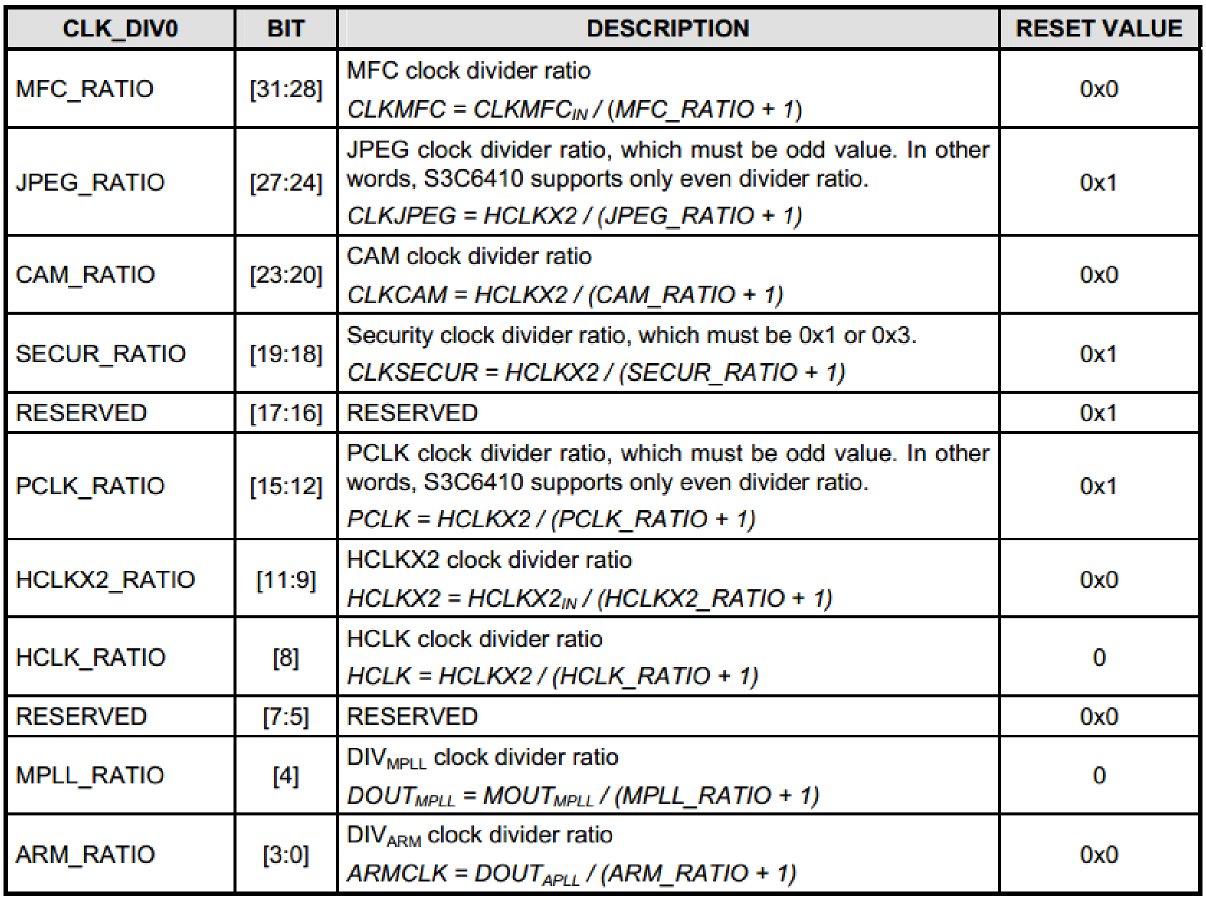


Figure 3-5. ARM and Bus clock generation

Table 3-2. Typical value setting for clock dividers (SFR setting value / output frequency)

APLL	MPLL	DIV_{ARM}	DIV_{HCLKX2}	DIV_{HCLK}	DIV_{PCLK}	$DIV_{CLKJPEG}$	$DIV_{CLKSECUR}$
266MHz	266MHz	0 / 266MHz	0 / 266MHz	1 / 133MHz	3 / 66MHz	3 / 66MHz	3 / 66MHz
400MHz	266MHz	0 / 400MHz	0 / 266MHz	1 / 133MHz	3 / 66MHz	3 / 66MHz	3 / 66MHz
533MHz	266MHz	0 / 533MHz	0 / 266MHz	1 / 133MHz	3 / 66MHz	3 / 66MHz	3 / 66MHz
667MHz	266MHz	0 / 667MHz	0 / 266MHz	1 / 133MHz	3 / 66MHz	3 / 66MHz	3 / 66MHz

The divider for ARM independently uses the output clock of APLL and there is no constraint for clock divider value as described in the above table.



CLK_DIV0	BIT	DESCRIPTION	RESET VALUE
MFC_RATIO	[31:28]	MFC clock divider ratio *CLKMFC = $CLKMFC_{IN}$ / (MFC_RATIO + 1)*	0x0
JPEG_RATIO	[27:24]	JPEG clock divider ratio, which must be odd value. In other words, S3C6410 supports only even divider ratio. *CLKJPEG = HCLKX2 / (JPEG_RATIO + 1)*	0x1
CAM_RATIO	[23:20]	CAM clock divider ratio *CLKCAM = HCLKX2 / (CAM_RATIO + 1)*	0x0
SECUR_RATIO	[19:18]	Security clock divider ratio, which must be 0x1 or 0x3. *CLKSECUR = HCLKX2 / (SECUR_RATIO + 1)*	0x1
RESERVED	[17:16]	RESERVED	0x1
PCLK_RATIO	[15:12]	PCLK clock divider ratio, which must be odd value. In other words, S3C6410 supports only even divider ratio. *PCLK = HCLKX2 / (PCLK_RATIO + 1)*	0x1
HCLKX2_RATIO	[11:9]	HCLKX2 clock divider ratio *HCLKX2 = $HCLKX2_{IN}$ / (HCLKX2_RATIO + 1)*	0x0
HCLK_RATIO	[8]	HCLK clock divider ratio *HCLK = HCLKX2 / (HCLK_RATIO + 1)*	0
RESERVED	[7:5]	RESERVED	0x0
MPLL_RATIO	[4]	DIV_{MPLL} clock divider ratio *$DOUT_{MPLL}$ = $MOUT_{MPLL}$ / (MPLL_RATIO + 1)*	0
ARM_RATIO	[3:0]	DIV_{ARM} clock divider ratio *ARMCLK = $DOUT_{APLL}$ / (ARM_RATIO + 1)*	0x0

4.4.2、系统时钟分频参数与分频对照表

APLL	MPLL	HCLKx2	HCLK	PCLK	
0	1	1	1	3	
1/1	1/2	1/2	1/2	1/4	
ARMCLK	HCLKx2_IN	HCLKx2	HCLK	PCLK	DOUTMPLL
533MHz	533MHz	266MHz	133MHz	66MHz	266MHz

```
ldr r1, [r0, #CLK_DIV0_OFFSET]
bic r1, r1, #0x30000
bic r1, r1, #0xff00
bic r1, r1, #0xff
ldr r2, =CLK_DIV_VAL
orr r1, r1, r2
str r1, [r0, #CLK_DIV0_OFFSET]
```

4.5、设置PLL输出频率

4.5.1、PLL频率设置寄存器

3.4.2.1 PLL Control Registers

S3C6410 has three internal PLLs, which are APLL, MPLL, and EPLL. They are controlled by the following seven special registers.

REGISTER	ADDRSS	R/W	DESCRIPTION	RESET VALUE
APLL_LOCK	0x7E00_F000	R/W	Control PLL locking period for APLL	0x0000_FFFF
MPLL_LOCK	0x7E00_F004	R/W	Control PLL locking period for MPLL	0x0000_FFFF
EPLL_LOCK	0x7E00_F008	R/W	Control PLL locking period for EPLL	0x0000_FFFF
APLL_CON	0x7E00_F00C	R/W	Control PLL output frequency for APLL	0x0190_0302
MPLL_CON	0x7E00_F010	R/W	Control PLL output frequency for MPLL	0x0214_0603
EPLL_CON0	0x7E00_F014	R/W	Control PLL output frequency for EPLL	0x0020_0102
EPLL_CON1	0x7E00_F018	R/W	Control PLL output frequency for EPLL	0x0000_9111

A PLL requires locking period when input frequency is changed or frequency division (multiplication) values are changed. PLL_LOCK register specifies this locking period, which is based on PLL's source clock. During this period, output will be masked '0'.

4.5.2、APLL,MPLL频率设置

APLL_CON / MPLL_CON	BIT	DESCRIPTION	RESET VALUE
ENABLE	[31]	PLL enable control (0: disable, 1: enable)	0
RESERVED	[30:26]	RESERVED	0x00
MDIV	[25:16]	PLL M divide value	0x190 / 0x214
RESERVED	[15:14]	RESERVED	0x0
PDIV	[13:8]	PLL P divide value	0x3 / 0x6
RESERVED	[7:3]	RESERVED	0x00
SDIV	[2:0]	PLL S divide value	0x2 / 0x3

The reset value of APLL_CON / MPLL_CON generate 400MHz and 133MHz output clock respectively, if the input clock frequency is 12MHz.

NOTE1:

The output frequency is calculated using the following equation:

$$F_{OUT} = MDIV \times F_{IN} / (PDIV \times 2^{SDIV})$$

where, MDIV, PDIV, SDIV for APLL and MPLL must meet the following conditions :

MDIV: $64 \leq MDIV \leq 1023$

PDIV: $1 \leq PDIV \leq 63$

SDIV: $0 \leq SDIV \leq 5$

F_{VCO} (=MDIV X F_{IN} / PDIV): $800MHz \leq F_{VCO} \leq 1600MHz$

F_{OUT}: $40MHz \leq F_{VCO} \leq 1600MHz$

F_{IN} : $10MHz \leq F_{IN} \leq 20MHz$

Don't set the value P and M to all zeros.

NOTE2:

Although there is the equation for choosing PLL value, we recommend only the values in the PLL value recommendation table. If you have to use another value, please contact us.

FIN (MHz)	Target FOUT (MHz)	MDIV	PDIV	SDIV
12	100	400	3	4
12	200	400	3	3
12	266	266	3	2
12	400	400	3	2
12	533	266	3	1
12	667	333	3	1

在此设置APLL,MPLL输出频率都为533MHz

```
ldr r1, =APLL_VAL
str r1, [r0, #APLL_CON_OFFSET]
ldr r1, =MPLL_VAL
str r1, [r0, #MPLL_CON_OFFSET]
```

4.5.3、EPLL频率设置

EPLL_CON0	BIT	DESCRIPTION	RESET VALUE
ENABLE	[31]	PLL enable control (0: disable, 1: enable)	0
RESERVED	[30:24]	RESERVED	0x00
MDIV	[23:16]	PLL M divide value	0x20
RESERVED	[15:14]	RESERVED	0x0
PDIV	[13:8]	PLL P divide value	0x1
RESERVED	[7:3]	RESERVED	0x00
SDIV	[2:0]	PLL S divide value	0x2

EPLL_CON1	BIT	DESCRIPTION	RESET VALUE
RESERVED	[30:18]	RESERVED	0x0
RESERVED	[17:16]	Should be 0x0	0x0
KDIV	[15:0]	PLL K divide value	0x9111

The reset value of EPLL_CON0 / EPLL_CON1 generate 97.70MHz output clock respectively, if the input clock frequency is 12MHz.

NOTE1:

The output frequency is calculated by the following equation:

$$F_{OUT} = (MDIV + KDIV / 2^{16}) \text{ X } F_{IN} / (PDIV \text{ X } 2^{SDIV})$$

where, MDIV, PDIV, SDIV for APLL and MPLL must meet the following conditions :

MDIV: $16 \leq MDIV \leq 255$

PDIV: $1 \leq PDIV \leq 63$

KDIV: $0 \leq KDIV \leq 65535$

SDIV: $0 \leq SDIV \leq 4$

F_{VCO} (= $(MDIV + KDIV / 2^{16})$ X F_{IN} / PDIV) : $300MHz \leq F_{VCO} \leq 600MHz$

F_{OUT} : $20MHz \leq F_{OUT} \leq 600MHz$

F_{IN} : $10MHz \leq F_{IN} \leq 20MHz$

Don't set the value P and M to all zeros.

NOTE2:

Although there is the equation for choosing PLL value, we recommend only the values in the PLL value recommendation table. If you have to use another value, please contact us.

FIN (MHz)	FOUT (MHz)	MDIV	PDIV	SDIV	KDIV
12	36	48	1	4	0
12	48	32	1	3	0
12	60	40	1	3	0
12	72	48	1	3	0
12	84	28	1	2	0
12	96	32	1	2	0
12	32.768	43	1	4	45264
12	45.158	30	1	3	6903
12	49.152	32	1	3	50332
12	67.738	45	1	3	10398
12	73.728	49	1	3	9961

4.6、设置时钟源

通过配置CLK_SRC寄存器选择时钟源为PLL输出

3.4.2.2 Clock source control register

S3C6410 has many clock sources, which include three PLL outputs, the external oscillator, the external clock, and other clock sources from GPIO configuration. CLK_SRC register controls the source clock of each clock divider.

REGISTER	ADDRESS	R/W	DESCRIPTION	RESET VALUE
CLK_SRC	0x7E00_F01C	R/W	Select clock source	0x0000_0000
CLK_SRC2	0x7E00_F10C	R/W	Select Audio2 clock source	0x0000_0000

		(00:48MHz, 01:$MOUT_{EPLL}$, 10: $DOUT_{MPLL}$, 11:FIN_{EPLL})	
MFCCLK_SEL	[4]	Control MUXMFC, which is the source clock of MFC	0
RESERVED	[3]	RESERVED	0
EPLL_SEL	[2]	Control MUX_{EPLL} (0:FIN_{EPLL}, 1:$FOUT_{EPLL}$)	0
MPLL_SEL	[1]	Control MUX_{MPLL} (0:FIN_{MPLL}, 1:$FOUT_{MPLL}$)	0
APLL_SEL	[0]	Control MUX_{APLL} (0:FIN_{APLL}, 1:$FOUT_{APLL}$)	0

```
ldr r1, [r0, #CLK_SRC_OFFSET]
ldr r2, =0x7
orr r1, r1, r2
str r1, [r0, #CLK_SRC_OFFSET]
```

4.7、等待时钟稳定

```
   /* wait at least 200us to stablize all clock */
   mov    r1, #0x10000
1:    subs    r1, r1, #1
   bne    1b
```