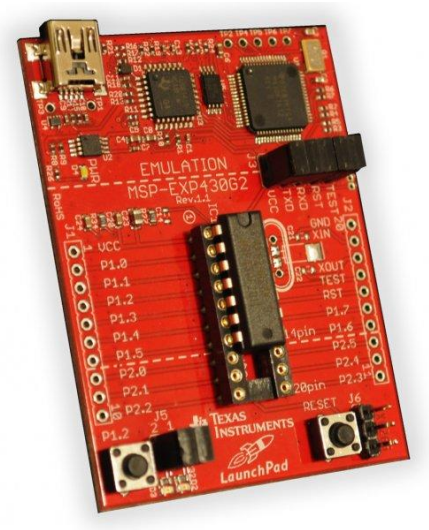


LaunchPad 之系统初始化及时钟配置

By: David Lee

TEXAS INSTRUMENTS

2011 年 11 月 9 日

声　　明

此文为个人学习心得，仅供学习交流之用，难免有错漏之处，请读者自行辨别。

此文发布于电子工程世界论坛 http://bbs.eeworld.com.cn/，本人论坛 ID：David_Lee，欢迎相互交流学习。

文中所涉及的例程，将提供 CCS v4.2 以及 IAR for MSP430 v5.2 的完整工程文件。

相关帖子列表：

手把手教你使用 Grace 开发 LaunchPad

http://bbs.eeworld.com.cn/viewthread.php?tid=303736&page=1&fromuid=194149#pid1178155

手把手教你使用 TI MSP430 LaunchPad

http://bbs.eeworld.com.cn/viewthread.php?tid=303599

大学堂之 IO 的使用

http://bbs.eeworld.com.cn/thread-306944-1-1.html

David Lee

2011 年 11 月 9 日

1. 概要

OS: Windows 7 Ultimate 32 bit

IDE: Code Composer Studio v4.2.3 & IAR for MSP430 v5.20

Board: TI LaunchPad MSP430G2231

本文主要介绍 MSP430G2231 的内部时钟结构，采用 Grace 进行该单片机的时钟配置为例，并介绍如何将 Grace 生成的初始化代码移植到 IAR form MSP430 中。

使用 IAR 时，记得选择器件型号为 MSP430G2231，并且修改仿真器为 FET Debugger。

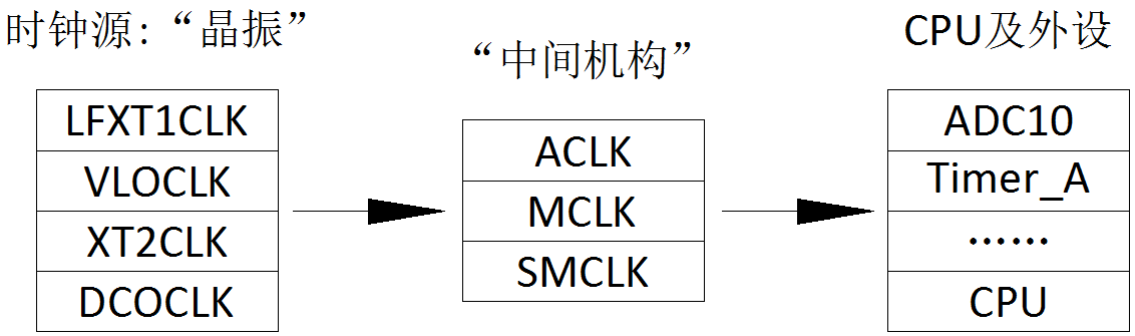
使用 CCS 时，记得选择器件型号为 MSP430G2231，仿真器使用默认的 TI MSP430 USB1 即可。

2. MSP430G2231 时钟结构

时钟是处理器的脉搏，没有了时钟，处理器将停止工作。至于为什么处理器一定要有时钟才能工作，这个我只能解释为这是组成 MCU 的硬件逻辑电路决定的。就像人的身体机能决定了如果没有脉搏，人也就 over 了。

MSP430G2231 基础时钟模块 Basic Clock Module+主要提供三个时钟输出：辅助时钟 ACLK（Auxillary Clock）、主系统时钟 MCLK（Main System Clock）和子系统时钟 SMCLK（Sub System Clock）。

上述三个时钟信号，是提供给 CPU 及其它功能模块使用的，它们有别于 LFXT1CLK、XT2CLK、DCOCLK、VLOCLK。LFXT1CLK、XT2CLK、DCOCLK、VLOCLK 这四个是时钟源，形象点说，这四个就相当于 4 个“晶振”，提供了不同的时钟。而 ACLK、MCLK、SMCLK 这三个是跟 CPU 或外设连接的，相当于连接在“晶振”和 CPU、外设之间的“中间机构”，这个“中间机构”决定给哪个“晶振”给 CPU 或外设使用。

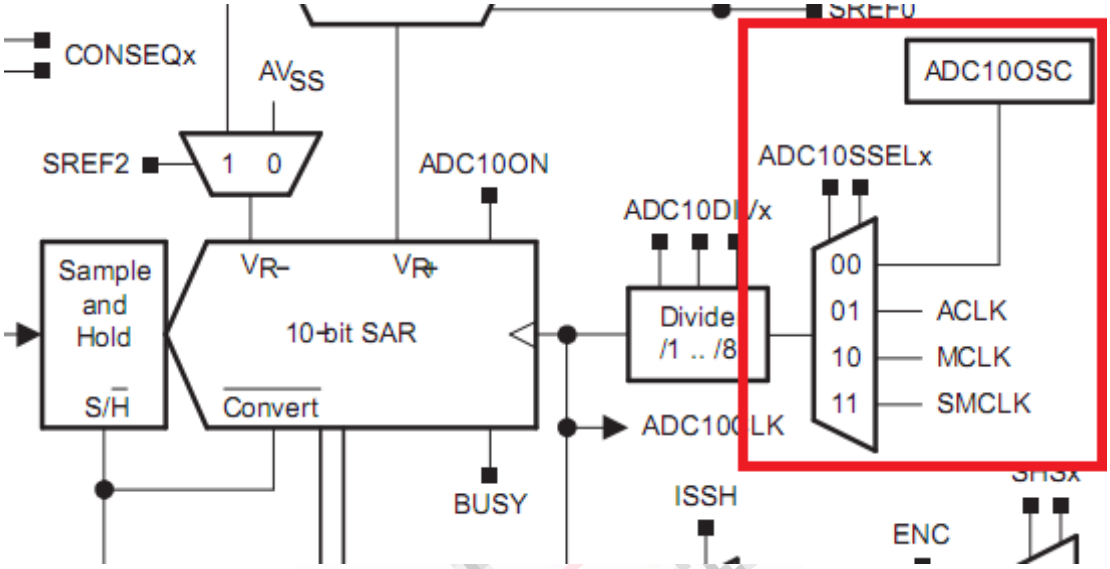


请注意，上图仅仅是为了标示 ACLK、MCLK、SMCLK 这三个基础时钟模块的时钟信号是有别于那四个时钟源 LFXT1CLK、XT2CLK、DCOCLK、VLOCLK 的。这个观念建立是很重要的，不然

往后那么多个带 CLK 后缀的东西，没几下就把自己给搞晕了。

“中间机构”提供了 3 种时钟输出，是不是外设可以在 ACLK、MCLK、SMCLK 中随便选呢？下面我们来看一个外设 ADC10 的时钟部分框图。

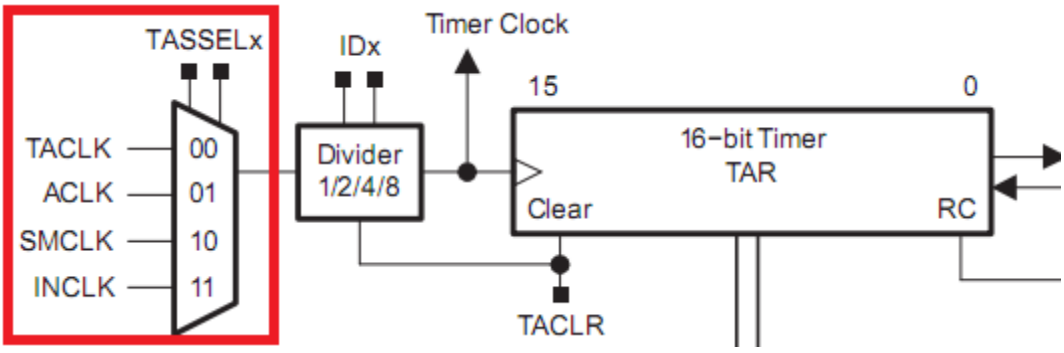
首先到 TI 官网下载 MSP430x2xx Family User's Guide（TI 文档编号：slau144h.pdf），这个 pdf 是 MSP430 的 x2xx 系列的手册（下面简称手册）。在手册的第 541 页的图 22-1 是 ADC10 的结构框图，这里截取了时钟部分：



由上图我们可以看到，ADC10 的时钟可以来源于四种：ADC10OSC 、ACLK、MCLK 或 SMCLK。ADC10 模块是可以在那三个“中间机构”里挑时钟的，并且它还有自己的 ADC10OSC 时钟源可以选择使用。怎么选择呢？ADC10SSELx 中写 00，则选择了 ADC10OSC 给 ADC10 使用。写 10，则选择了 MCLK 给 ADC10 使用。这里时钟的选择使用，并没有排它性，即若 ADC10 选择使用了 SMCLK，其它外设模块也可以同时选择使用 SMCLK。

此外，通过四选一，选择的时钟进入 ADC10 的分频器，可以通过 ADC10DIVx 写相应值进行 1~8 分频，把时钟频率降下来。

在手册第 359 页的图 12-1 是 Timer_A 的结构框图，这里截取了时钟部分：



由上图可以看到，Timer_A 的时钟选择也有四种，TACLK、ACLK、SMCLK 和 INCLK，这里并

不能选择 MCLK 作为 Timer_A 的时钟，之前 ADC10 特有的 ADC10OSC 时钟也不能选择作为 Timer_A 的时钟。但是，跟 ADC10 一样，Timer_A 也有自己特有的时钟源 TACLK 和 INCLK。

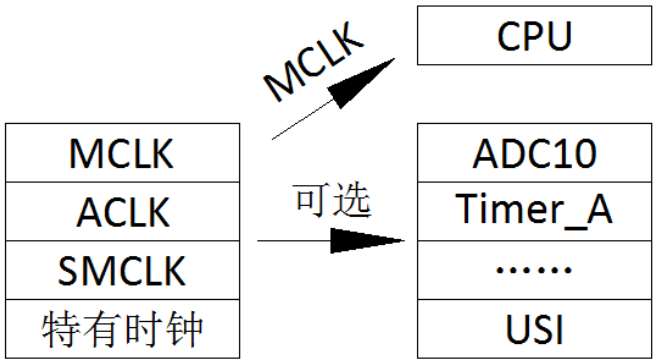
同样，选择了 Timer_A 的时钟之后，进入到 Timer_A 的分频器，这里的分频并不像 ADC10 那样可以进行 1~8 分频，Timer_A 只能选择 1、2、4 或者 8 分频。

其实，通过上述介绍，可以发现，我们学习一款 MCU 的时候，看它的结构框图是非常直观的，除了可以看到信号输入输出的路径，TI 还把相应的寄存器标示出来，我们只要给那些寄存器设置相应的值，就可以选择我们想要的功能。

此外，我们可以看到基础时钟模块 Basic Clock Module+虽然只提供了 ACLK、MCLK、SMCLK 三种时钟，但是有些外设模块是有自己特有的时钟源可以选择的，比如 ADC10 的 ADC10OSC。而且外设并不是基础时钟模块提供的所有三种时钟都可以选择的，比如 Timer_A 就不能使用 MCLK 的时钟。所以，外设可以使用什么时钟，我们得依据外设的时钟输入结构来确定。

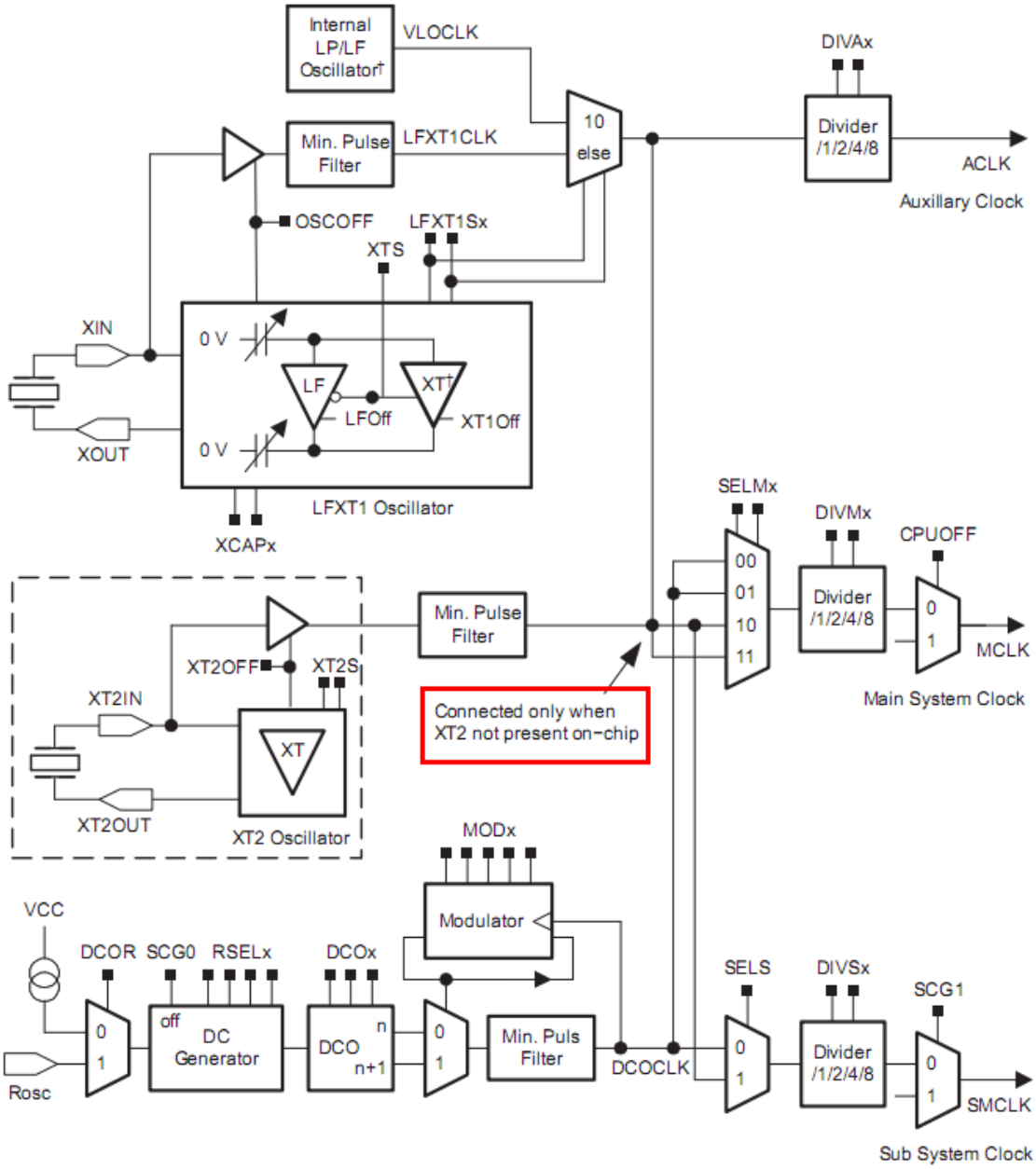
CPU 是处理器的核心部分，它使用的时钟始终是 MCLK。

MSP430G2231 的基础时钟模块和外设之间的关系大概可以用下面的图表示，其中 MCLK、ACLK、SMCLK 是由基础时钟模块提供的，特有时钟是部分外设模块特有的。



前面的内容，介绍了基础时钟模块这个“中间机构”与外设之间的关系。下面介绍那四个“晶振”与基础时钟模块提供的 ACLK、MCLK 和 SMCLK 之间的关系。

在手册第 275 页的图 5-1 是基础时钟模块的结构框图，如下图：



由上面的图，我们可以非常直观的看到，ACLK、MCLK、SMCLK 是通过一些时钟输入选择之后，基础时钟模块输出的三个时钟信号，这三个时钟信号最终将提供给 CPU 和外设使用。而这三个时钟的最初来源，就是来源于 VLOCLK、LFXT1CLK、XT2CLK 或 DCOCLK 的。

从上到下，观察基础时钟模块的结构框图，我们可以看到：ACLK 的时钟来源有 VLOCLK 或者外接的晶振 XT1 产生的 LFXT1CLK。

MCLK 的时钟来源可以是与 ACLK 相同的时钟（VLOCLK 或 LFXT1CLK）、XT2 产生的 XT2CLK 或者 DCOCLK。

框图中间红色那里有一段标注：Connected only when XT2 not present on-chip（当 XT2 不可用时，该节点连接），所以 SMCLK 的时钟来源可能是 DCOCLK、XT2CLK（当 XT2 可用时）或者

与 ACLK 相同的时钟（VLOCLK 或 LFXT1CLK，当 XT2 不可用时）。

当然，上述框图是针对整个 MSP430x2xx 系列的，具体到 MSP430G2231 的时钟，是怎么样的呢？

手册的图 5-1 基础时钟模块的结构框图下面有这么一段描述：

NOTE: † Device-Specific Clock Variations

All clock features are not available on all MSP430x2xx devices:
MSP430x20xx, MSP430G2xxx: LFXT1 does not support HF mode, XT2 is not present, ROSC is not supported.
MSP430x21x1: Internal LP/LF oscillator is not present, XT2 is not present, ROSC is not supported.
MSP430x21x2: XT2 is not present.
MSP430x22xx, MSP430x23x0: XT2 is not present.

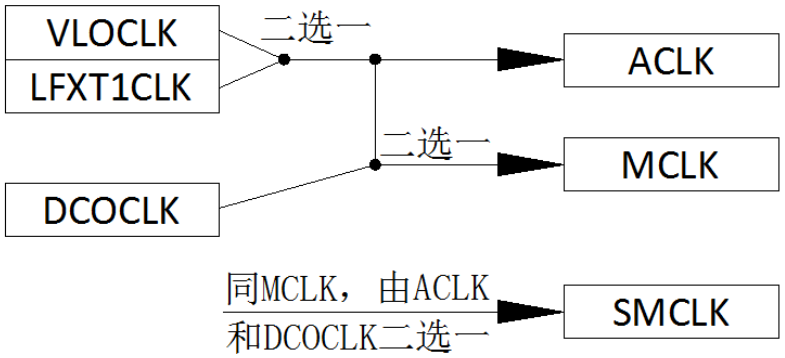
NOTE: † Device-Specific Clock Variations 这个是告诉我们带†标示的部分，不同器件参数可能不一样，比如内部低频振荡器 Internal LP/LF Oscillator 的频率，虽然同为系列 x2xx，但不同型号可能频率不一样。

MSP430x20xx, MSP430G2xxx: LFXT1 does not support HF mode, XT2 is not present, R_{OSC} is not supported. 从这个提示，我们可以知道，MSP430G2231 的 LFXT1 是不支持高频 HF（High Frequency）模式的，所以 8MHz、16MHz 等高频晶体就不要接到 XT1 上去用了，否则 MCU 运行情况是不可预料的。要使用 XT1 的时候，接个低频的晶振 32.768KHz 就可以了。

MSP430G2231 的 XT2 是不支持的，所以，框图上红色框框里面的那个节点是导通的，顺着那个节点的连接，我们可以看到 MSP430G2231 的 SMCLK 是可以选择 DCOCLK 或者与 ACLK 相同的时钟（VLOCLK 或 LFXT1CLK）。

MSP430G2231 的 R_{OSC} 是不可用的，不能通过外接 R_{OSC} 来调整 DCOCLK 产生的时钟，只能通过内部的 R_{SELx} 和 D_{COx} 来调整 DCOCLK 的频率。

至此 MSP430G2231 的基础时钟模块的结构已经简单介绍完毕，MSP430G2231 的时钟结构简化后，可以通过下图表示：

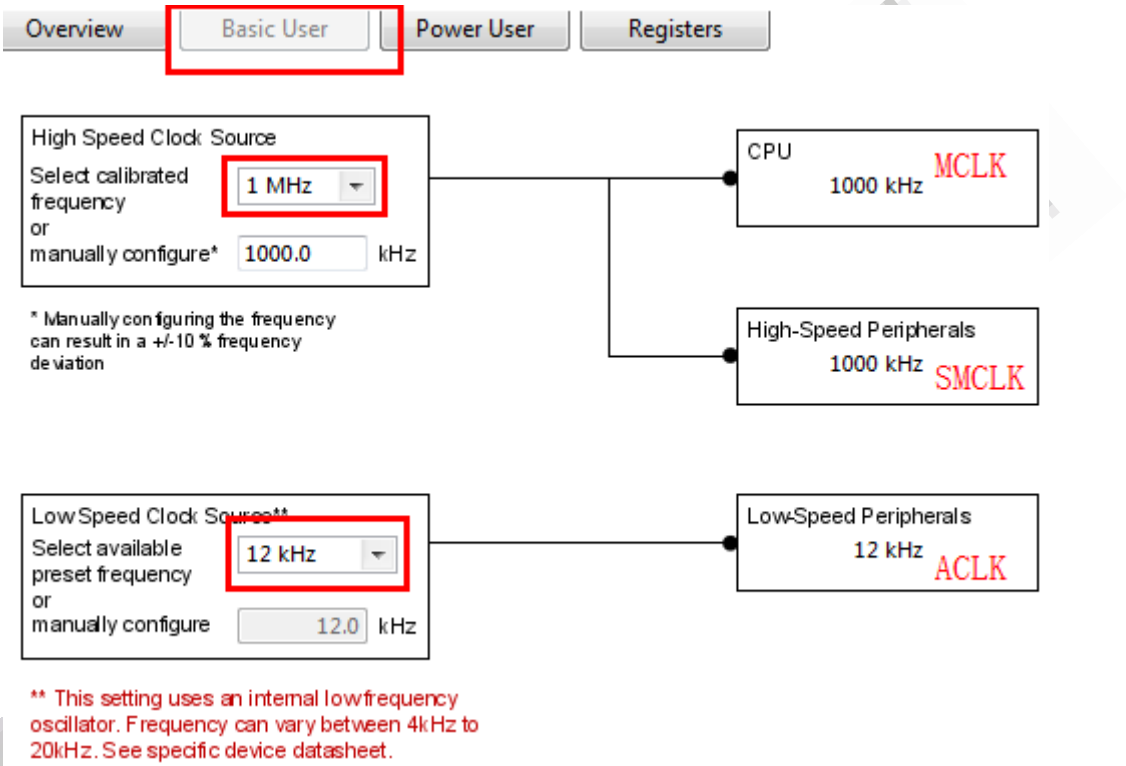


MSP430G2231 的 ACLK 时钟源可以选择外接低频晶体 XT1 产生的 LFXT1CLK 或者内部的振荡器 VLOCLK。

MCLK 和 SMCLK 的来源是类似的，都可以是 DCOCLK，或者是与 ACLK 相同的时钟源。

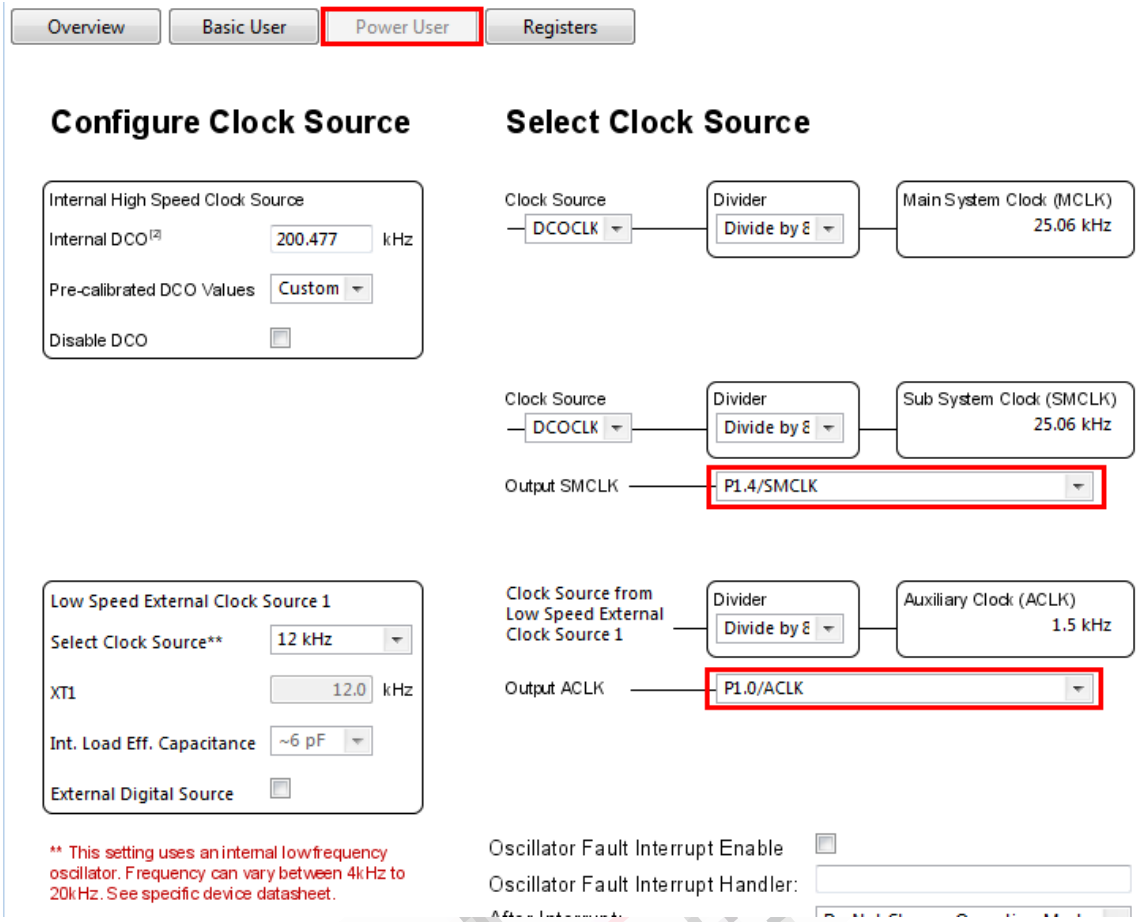
3. 时钟初始状态

上电后，系统默认使用的主系统时钟 MCLK 和子系统时钟 SMCLK 是同为 DCOCLK 产生的 1MHz 时钟，而辅助时钟 ACLK 则为内部 VLOCLK 产生的 12KHz 时钟。下图是 CCS 环境下的 Grace 对时钟的默认配置：



4. 时钟配置

在 CCS 新建一个空的 Grace 工程，Grace 的时钟配置界面里点击 Power User，进入到对时钟较为详细的配置，如果对 MSP430G2231 比较熟悉的话，也可以直接进入到 Registers 下，通过直接配置寄存器来实现对时钟的初始化。这里配置了 MCLK 和 SMCLK 为 25.06kHz，ACLK 为 1.5kHz，并且把 SMCLK 和 ACLK 直接输出到管脚 P1.4 和 P1.0。



这里，我通过 EK-LM3S811 写了一个简单的频率计程序，把测量结果通过 UART0 发送至串口调试助手，其中 MSP430G2231 的 P1.4 引脚 SMCLK 的实测数据为：26.2kHz 左右，如下图：

P1.0 引脚 ACLK 的实测数据为：1.43kHz 左右，如下图：

可以看到，使用内部的振荡器，配置出来的频率是有一定的波动范围的，并不是精确值，当然，这也可能是我写的的频率计程序由于中断用时等因素产生了一定的误差。建议有条件的网友通过示波器来观察 P1.0 和 P1.4 引脚的波形，检验时钟配置结果。

5. 时钟模块部分参数

在手册第 279 页的图 5-6 所示，DCOCLK 的输出频率 f_{DCO} 可以高达 20MHz，实际上 MSPG2231 的 MCLK 能不能跑到 20MHz 呢？

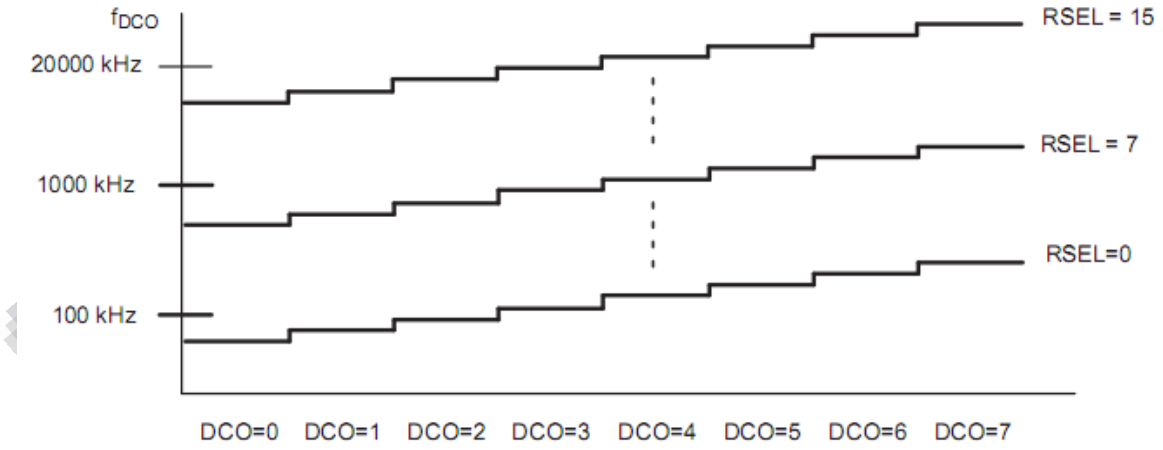
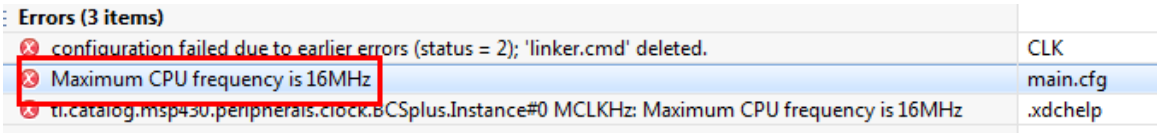
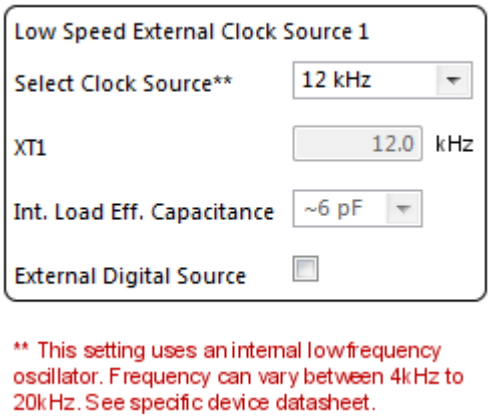


Figure 5-6. Typical DCOx Range and RSELx Steps

这边通过 Grace 配置 MCLK 使用 DCOCLK 的 20MHz，分频值 1，编译后 CCS 提示 CPU 的最高运行频率为 16MHz。

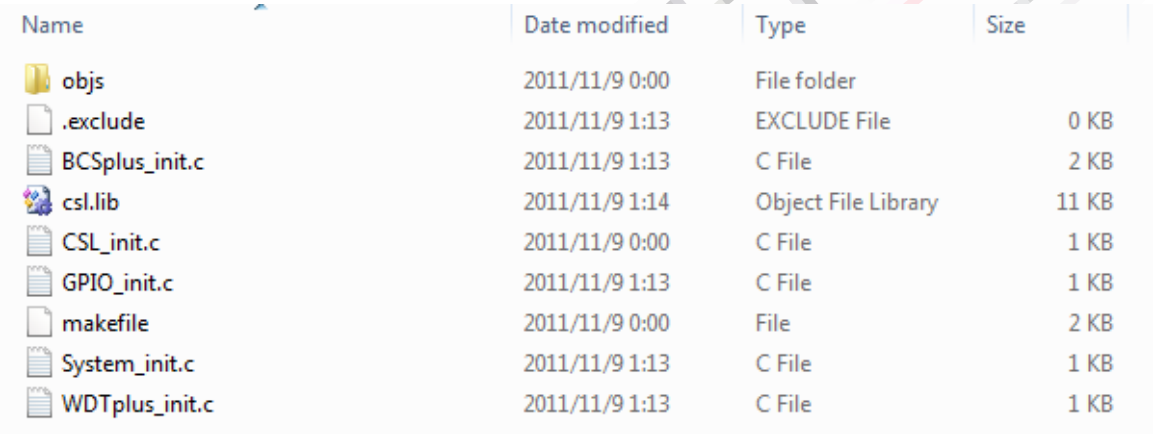


MSP430G2231 的 VLOCLK 频率为 12kHz，如下图所示：



6. Grace 代码移植

Grace 配置 MSP430 之后，会在工程目录下生成相应的源代码，这里的路径为：CLK\CCS\src\csl，在这个路径下，我们可以看到有几个.c 文件：



Name	Date modified	Type	Size
objs	2011/11/9 0:00	File folder	
.exclude	2011/11/9 1:13	EXCLUDE File	0 KB
BCSplus_init.c	2011/11/9 1:13	C File	2 KB
csl.lib	2011/11/9 1:14	Object File Library	11 KB
CSL_init.c	2011/11/9 0:00	C File	1 KB
GPIO_init.c	2011/11/9 1:13	C File	1 KB
makefile	2011/11/9 0:00	File	2 KB
System_init.c	2011/11/9 1:13	C File	1 KB
WDTplus_init.c	2011/11/9 1:13	C File	1 KB

在 CCS 工程的 main.c 里，我们可以看到有 CSL_init();// Activate Grace-generated configuration，这行代码完成了对 MSP430 所有 Grace 初始化代码的调用。

CSL_init();函数的原型在 CSL_init.c 里，直接用记事本打开可能会出现代码不分行显示的情况，建议直接使用 CCS 来 Open File…（也可以用其它 Edit Plus 等软件打开），用 CCS 打开之后，如下图所示：

```
/* external peripheral initialization functions */
extern void WDTplus_init(void);
extern void GPIO_init(void);
extern void BCSplus_init(void);
extern void System_init(void);

/*
 *  ======== CSL_init ========
 *  Initialize all configured CSL peripherals
 */
void CSL_init(void)
{
    /* initialize Config for the MSP430 WDT+ */
    WDTplus_init();

    /* initialize Config for the MSP430 GPIO */
    GPIO_init();

    /* initialize Config for the MSP430 2xx family clock systems (BCS) */
    BCSplus_init();

    /* initialize Config for the MSP430 System Registers */
    System_init();

}
```

这里我们可以看到 Grace 对 MSP430 的初始化流程，而我们移植的过程，就是把这些函数复制粘贴的过程，这几个函数的源码在 CLK\CCS\src\csl 的几个.c 文件里都可以找到，把它们处理之后，直接在 IAR 里新建工程，复制粘贴过去，即可使用。

新建一个 IAR 工程，在 Option 里设置相应的芯片型号 MSP430G2231，仿真器选择 FET Debugger。

第一步：IAR 使用的头文件用#include "io430.h"即可，其它头文件视乎需要添加。

第二步：把 CLK\CCS\src\csl\CSL_init.c 里的 CSL_init()里调用的函数列表都复制到 IAR 工程的 main()函数里。

IAR Information Center for MSP430 | main.c

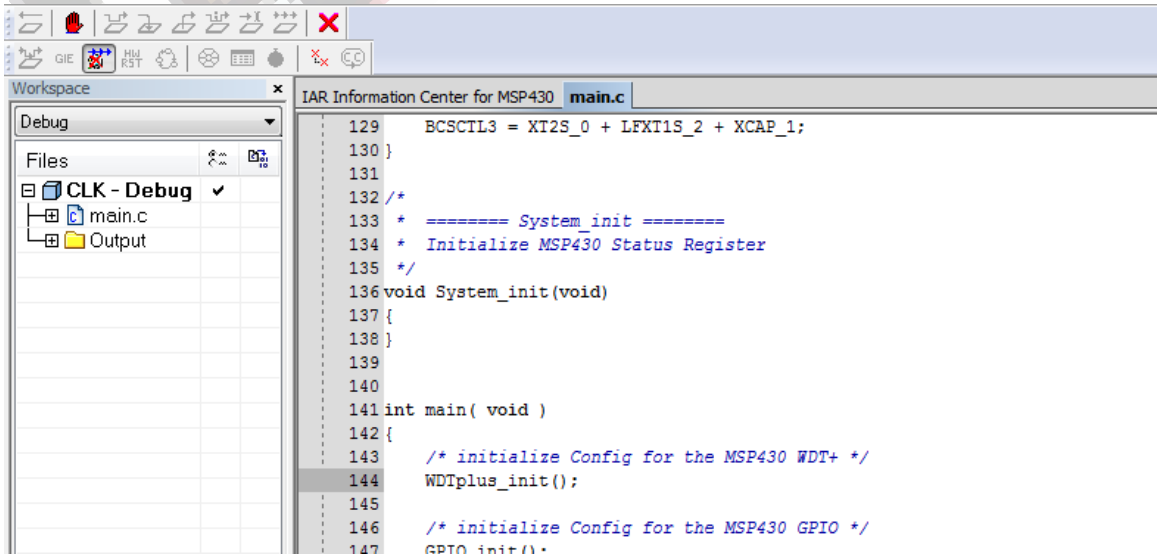
```
1
2 #include "io430.h"
3
4
5 int main( void )
6 {
7     /* initialize Config for the MSP430 WDT+ */
8     WDTplus_init();
9
10    /* initialize Config for the MSP430 GPIO */
11    GPIO_init();
12
13    /* initialize Config for the MSP430 2xx family clock systems (BCS) */
14    BCSplus_init();
15
16    /* initialize Config for the MSP430 System Registers */
17    System_init();
18
19    while(1);
20 }
21
```

第三步：依次把 main()中调用的那几个初始化函数复制到 main()之前，那几个初始化函数可以在 CLK\CCS\src\csl 相应的.c 文件找到源码。

第四步：编译程序，下载运行。

```
Messages
Building configuration: CLK - Debug
Updating build tree...
main.c
Linking

Total number of errors: 0
Total number of warnings: 0
```



至此，Grace 生成的 MSP430 初始化代码已经移植完毕，对于使用到中断的情况，记得把中断服务函数也添加上。

7. 总结

本文简单介绍了 MSP430G2231 的基础时钟模块的结构组成以及 Grace 代码在 IAR 下的移植。

MSP430 提供了非常灵活的系统时钟，通过选择不同频率的时钟，可以让外设工作在不同的频率，合理配置，降低功耗。

在学习使用 MSP430 的过程中，有几个东西是非常有帮助的：芯片手册、IDE 里对应芯片的.h 文件、网络上一些同系列寄存器的中文介绍、官方的例程。

End

电子发烧友 eeworld.com.cn