



TI-EEWORLD
联手打造

玩转

TI MSP430 LAUNCHPAD



序

旁人也许很难理解我看到这本电子书时的欣喜。

十年了，作为TI MSP430技术支持工程师中的一员，或许当初对这个选择还有些许的不确定，而现在我则庆幸当年的决定。的确，MSP430本身是一个相对简单易用的产品，而LaunchPad更是一块MSP430的敲门砖。然而这十年间，我看着一批又一批的人对MSP430从陌生到熟悉，从饶有兴趣到用其打造出真实的产品，从“小荷才露尖尖角”到企业成为国际上的领头羊……而他们中的一些人也已经从普通的工程师变身为技术总监，还有一些则跳出了研发的范畴，从事系统咨询等更高视角的工作。

可以说，在为MSP430做支持的过程中，我收获了巨大的成就感，因为我看到的是中国电子产业的蓬勃发展、中国客户进步、越来越多的青年才俊加入到电子设计的行业中，同时我也看到了TI的进步。

MCU的发展对中国电子产业的重要性毋需多言，也许大家知道，TI的Jack Kirby是集成电路的发明人并因此获得了诺贝尔奖；但少有人知道，拥有第一件微处理器专利的公司是TI，TI的Gary Boone在1973年获得第一个单片微处理器的美国专利授权。这些年来，TI一直都在逐步加大对MCU产品的投入，而产品技术上的进步只是一个方面，更重要的是，TI在思考如何改变自己以适应客户的需求，如何让客户更成功。

熟悉TI的朋友都知道，一直以来，TI在技术培训上面投入了很大的人力物力，我们也不停在思考，如何给客户提供更好的培训。从2012年5月开始，我们联合EEWORLD进行在线培训，为的就是能够让大家不受地域、时间的限制，很高兴看到了大家对培训的热情和赞扬，这也是对我们在支持和培训领域不断创新的一种肯定。

然而让我意想不到的是，很多网友在学完课程后，利用MSP430 LaunchPad开发板进行了许多极具创新的尝试，并无私地分享在EEWORLD论坛上。而这些实际应用案例和TI讲述产品基本概念的在线培训结合起来，对用户产生的帮助会是最大的。

也正因为如此，当EEWORLD提出要将TI的在线培训课程和网友的心得体会结合成一本电子书时，我们予以全力支持，因为我们清楚这件事本身对于用户的价值。

顺便说一句，我就是MSP430 LaunchPad这门课程的讲师，说实话，当时在面对镜头时心里多少还是有点儿紧张，因为我希望可以将最有价值的东西呈现给大家。所以，也恳请广大读者以及工程师批评指正，以便在以后的版本中及时修正。在此也想对部分已经观看过培训视频、并给出很多积极反馈的工程师朋友表示感谢。希望更多工程师朋友加入到与我们互动的行列中，分享你的创意。

MSP430 LaunchPad是我们的第一期培训课程，也是第一本电子书，之后TI还会继续推出更多更好的培训及MCU系列电子书以飨工程师朋友们。

谁又知道读这本小书的人中，不会诞生中国未来的乔布斯呢？

丁京柱 Jeffrey Ding
TI微控制器产品技术经理

前言

对于广大电子技术领域的从业工程师和相关专业的在校大学生而言,熟练掌握MCU(微控制器,俗称“单片机”)系统的应用设计技术是职业发展的必由之路。长期起来,因种种原因,国内的单片机入门教材多基于Intel公司于1980年推出的MCS51体系,虽然,MCS51系列单片机也得到了长足的发展,但毕竟是三十多年前的设计,在追求高性能以及绿色环保的今天,可选择的应该更多。

TI的MSP430 Value系列是针对当今市场流行趋势采用前瞻性设计理念和先进半导体工艺推出的高性能、低功耗16位精简指令集型单片机,更有价值的是,针对MSP430 Value系列单片机, TI推出了配套的高性能廉价开发、实验平台MSP430 LanuchPad,广大初学者和电子工程师们可以用非常低廉的价格获得这一单片机入门学习及应用开发的利器,仅仅是一顿快餐的代价,这在过去是根本无法想像的。

在我的印象中,相信也是绝大多数人的看法,单片机入门教材或工具书之类都是由资深专业人士编写,而令我感到非常高兴和欣慰的是,本书中大量精彩内容来自于EEWorld电子论坛的网友们,他们分享了众多在MSP430 LanuchPad学习过程中获得的知识和经验,这些都是来自于实践的一手真知,非常的有价值。互联网改变世界,谁说不是呢?

为了促进MSP430 LanuchPad的推广和帮助广大初学者们更快、更好的学习、掌握单片机入门知识,EEWorld站方非常用心的做了大量幕后工作,包括与TI官方合作推出了LanuchPad入门培训系列视频讲座等,EEWorld电子论坛已成为人气最旺的单片机网上学习园地,推出本书既是对前期MSP430 LanuchPad网上学习活动的阶段性总结,也是继续的推进。从EEWorld站方在这一系列的学习促进活动中的表现可以看出,EEWorld是一支充满活力和热情的团队,这也是广大网友们的福音,相信EEWorld在以后更多的类似活动中会做的更好、更成熟。有感于此,当向农提出想要我帮忙为本书写一点寄言时,我毫不犹豫的答应了,只是最近事情比较多,时间上多少有些耽搁,对此表示很惭愧。

在这里,还是要跟网友们再多啰嗦几句:电子技术是一门实践性很强的学科,要想成为一名优秀的电子工程师,尽量多的动手实践、用心思考以及彼此交流是非常非常重要的,建议网友们在学习单片机基础知识以及阅读本书的过程中务必勤动手、多动脑,如果遇到任何问题最好及时在EEWorld电子论坛中发帖交流,如果有自己的学习体会和实践经验也不妨亮出来晒晒,其过程本身也是一种收获,而且也将有机会入选本书的下一版(这个算是提前“剧透”一下吧)。

最后,祝大家学习进步,都有一个灿烂的前程。

EEworld网友 chunyang

目录

第一章 菜鸟基础篇

1.1 LaunchPad开发板介绍	5
1.2 LaunchPad初体验	6
1.3 LaunchPad板上资源解读	7
1.4 LaunchPad之系统初始化及时钟配置	8
1.5 LaunchPad-ADC10 介绍	14
1.6 实现带有TimerA 的UART功能	16
1.7 LaunchPad另一种uart的实现-串行接口	20
1.8 LaunchPad风火轮触摸板初体验	21
1.9 LaunchPad_实验板触摸感应子卡使用指南	23
1.10 MSP430 LaunchPad触摸板试用心得	27

第二章 官网课程篇

2.1 课程前言	29
2.2 Value Line产品的介绍	30
2.3 集成开发环境CCS	34
2.4 CPU 与基本时钟模块	38
2.5 中断与GPIO口	43
2.6 定时器A与增强型看门狗模块	47
2.7 MSP430低功耗设计的实践基础	54
2.8 10位ADC和增强型的比较器	64
2.9 串行通信模块	68
2.10 Grace软件	72
2.11 基于430实现电容式触摸按键的解决方案	77

第三章 高手实战篇

3.1 基于LaunchPad的【低功耗时钟】	80
3.2 触摸按键MSP430低功耗时钟	82
3.3 MSP430G2231简单温度计	84
3.4 MSP430 LaunchPad 心率仪	85
3.5 TI Launchpad 音乐《欢乐颂》	87
3.6 Launchpad + 呼吸灯这样才好玩~	88
3.7 分享一下自制的触摸摁键，有程序，有视频，有真相	89
3.8 木有风火轮，用锡箔做电容触摸感应	90
附录一—Launchpad 资料汇总	91
附录二玩转LaunchPad实用问答	92
附录三:编委信息与后记	93
附录四:版权说明	94
附录五:EE团优惠券	95

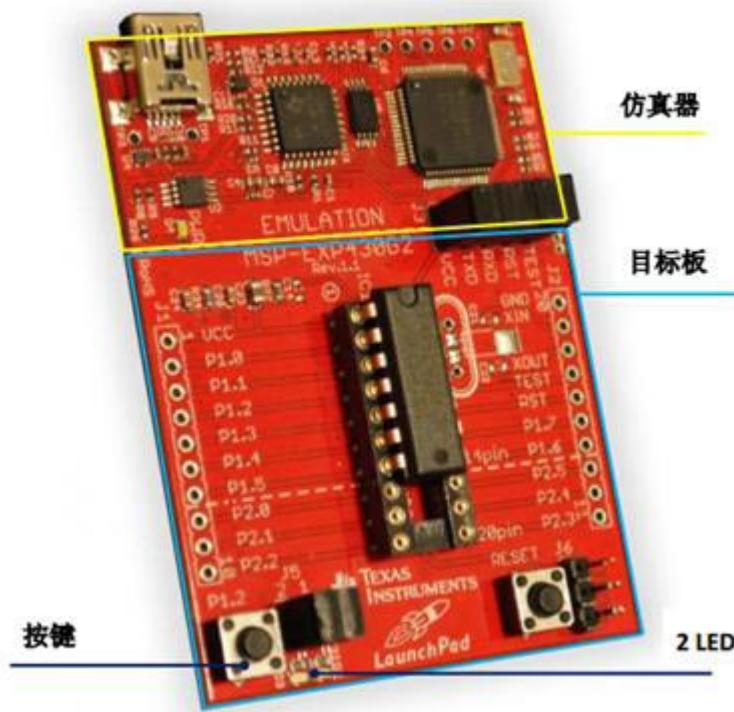
第一章 菜鸟基础篇

1.1 LaunchPad开发板介绍

MSP-EXP430G2 LaunchPad 是一款易于使用的闪存编程器和调试工具，它提供了在 MSP430 超值系列器件上进行开发所需的一切内容。它提供了具有集成仿真功能的 14/20 引脚 DIP 插座目标板，可通过 Spy Bi-Wire (2 线 JTAG) 协议对系统内置的 MSP430 超值系列器件进行快速编程和调试。由于 MSP430 闪存的功耗极低，因此无需外部电源即可在数秒内擦除闪存并对其进行编程。

LaunchPad 将 MSP430 器件与 Code Composer Studio 版本 4 或 IAR 嵌入式工作平台等集成软件环境相连接。MSP430 超值系列器件上的这些 IDE 是免费且非受限的软件。LaunchPad 支持所有采用 14 或 20 引脚 DIP 封装 (TI 封装代码:N) 的 MSP430G2xx 闪存器件。

LaunchPad 还采用用于定制项目和应用的板载可编程 LED 和按钮！10 引脚 PCB 连接器还可用于连接 LaunchPad 和附加器件。

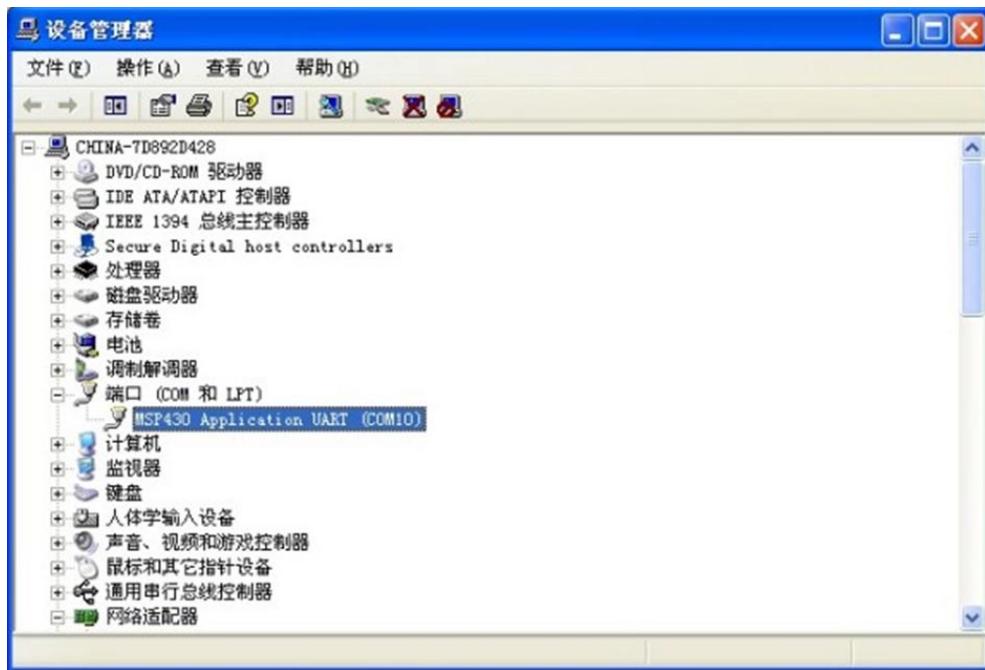


更多技术资料：<http://www.ti.com.cn/tool/cn/msp-exp430g2>

1.2 LaunchPad初体验

前不久一个偶然的机会获得MSP-EXP430G2 LaunchPad 实验板一块：LaunchPad板子上已经插有一片具有：2kB 闪存、128B RAM、10 GPIO、1 个 16 位定时器、WDT、BOR、1 个 USI (I2C/SPI) 8通道 10位ADC 的MSP430G2231IN14单片机，并预置内部温度测量上位机显示程序，所以上电即可演示。在小板上电运行之前首先要详细看一下使用说明书，并下载LaunchPad演示时所需的相关驱动和软件，这些驱动和软件在TI的官方网站：<http://www.ti.com.cn/tool/cn/msp-exp430g2> 都可以下载的到！

(1)、安装驱动，这个应该很简单。安装完之后用LaunchPad板子附件中的数据线将小板与连接电脑，等待识别后板子下方的红色LED(LED1)和绿色LED(LED2)会交替闪烁，打开电脑设备管理器中会显示如下信息：



(2)、按下LaunchPad板子左下方的S2按键，红色LED(LED1)和绿色LED(LED2)停止闪烁，LaunchPad 开始运行演示程序。

(3)、运行电脑中的LaunchPad_Temp_GUI.exe 程序，运行之前要确定你的电脑中是否已经安装java程序，否则会出错。

(4)、根据设备管理器中所分配的串口号，与下图程序界面中的串口号对应，选择串口号前面的数字，故在 # = 后面输入 1 并按回车键：

(5)、上位机程序中显示的温度如下，因为此显示温度为华氏温度，想知道摄氏温度得需要自己换算一下：公式 $^{\circ}\text{C} = (\text{F}-32) * 5/9$

更多详情：<http://bbs.eeworld.com.cn/thread-309771-1-1.html>

1.3 LaunchPad板上资源解读

MSP- EXP430G2 LaunchPad features:

USB debugging and programming interface featuring a driverless installation and application UART serial communication with up to 9600 Baud 板上的串口转usb 最多只能到9600的波特率。

仿真连接：

The power supply and the Spy-Bi-Wire JTAG signals TEST and RST must be connected with jumper J3 to allow the onboard emulation connection to the device. Jtag看来只需要TEST 和 RST两个接口,要断开仿真只需断开这两个跳线。

The jumper 5 VCC also must be opened if the LaunchPad board is powered with an external power supply over J6. 如果在J6接外部电源,J3 Jumper 5的VCC必须断开。

Table 1. Jumper Connection J3 Between Emulator and Target

Jumper	Signal	Description
1	TEST	Test mode for JTAG pins/Spy-BI-Wire test Clock input during programming and test
2	RST	Reset/Spy-bi-Wire test data input/output during programming and test
3	RXD	UART transmit dataoutput
4	TXD	UART transmit data output
5	VCC	Target socket power supply voltage(power consumption test jumper)

可通过5脚来用电流表测整个板(包括外设)的耗电量

更多详情：<http://bbs.eeworld.com.cn/thread-306923-1-1.html>

1.4 LaunchPad之系统初始化及时钟配置

1. 概要

OS: Windows 7 Ultimate 32 bit

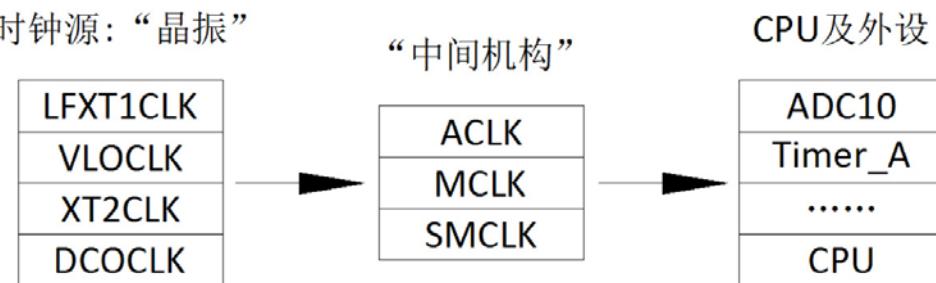
IDE: Code Composer Studio v4.2.3 & IAR for MSP430 v5.20

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 for 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

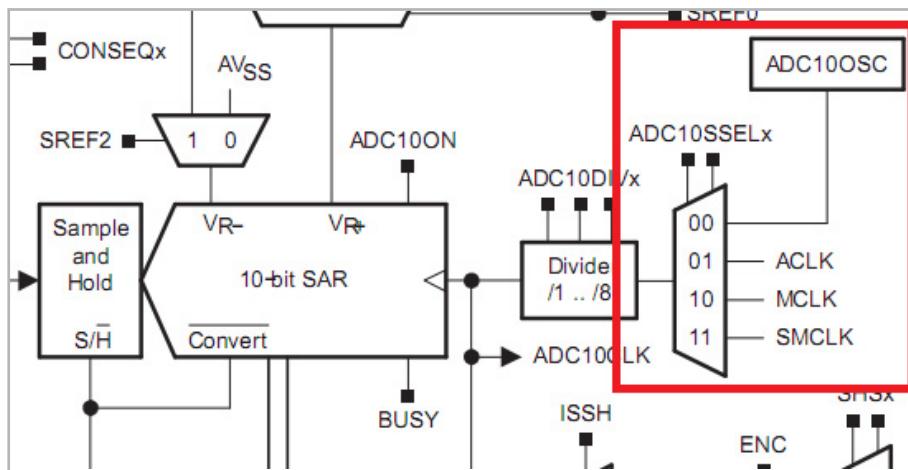
MSP430G2231基础时钟模块Basic Clock Module+主要提供三个时钟输出:辅助时钟ACLK(Auxillary Clock)、主系统时钟MCLK(Main System Clock)和子系统时钟SMCLK(Sub System Clock)。上述三个时钟信号,是提供给CPU及其它功能模块使用的,它们有别于LFXT1CLK、XT2CLK、DCOCLK、VLOCLK。LFXT1CLK、XT2CLK、DCOCLK、VLOCLK这四个是时钟源,形象点说,这四个就相当于4个“晶振”,提供了不同的时钟。而ACLK、MCLK、SMCLK这三个是跟CPU或外设连接的,相当于连接在“晶振”和CPU、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?

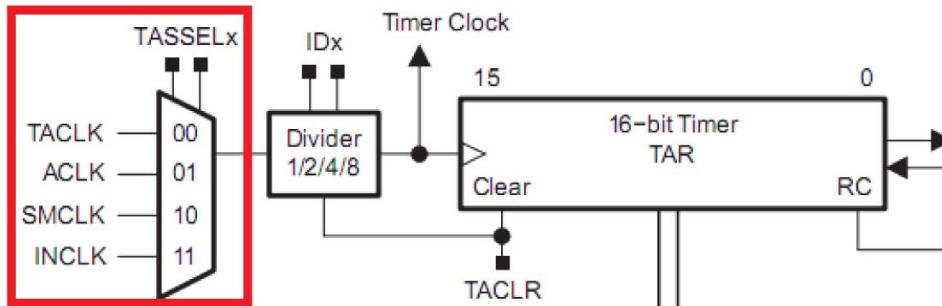
下面我们来看一个外设ADC10的时钟部分框图。

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？ADC10SELx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：

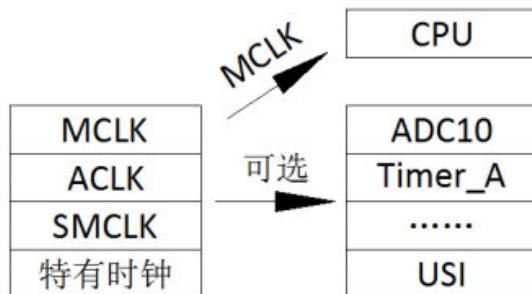


由上图可以看到，Timer_A的时钟选择也有四种，TACLK、ACLK、SMCLK和INCLK，这里并不能选择MCLK作为Timer_A的时钟，之前ADC10特有的ADC10OSC时钟也不能选择作为Timer_A的时钟。但是，跟ADC10一样，Timer_A也有自己特有的时钟源TACLK和INCLK。

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。

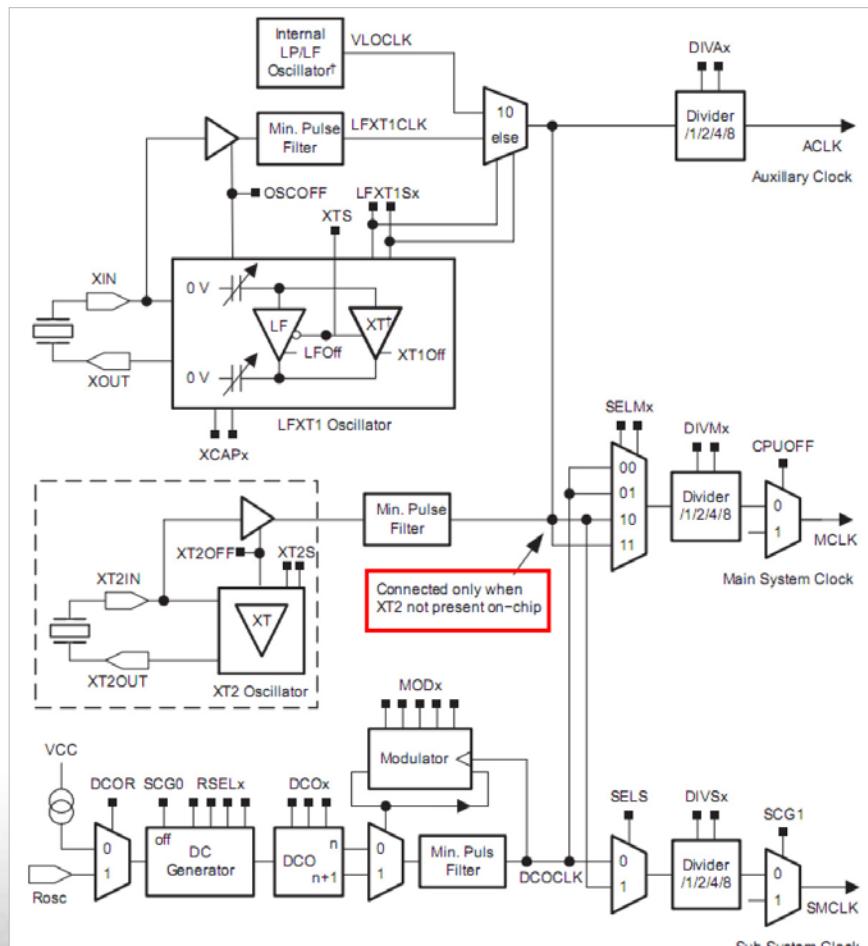
此外，我们可以看到基础时钟模块Basic Clock Module+虽然只提供了ACLK、MCLK、

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：

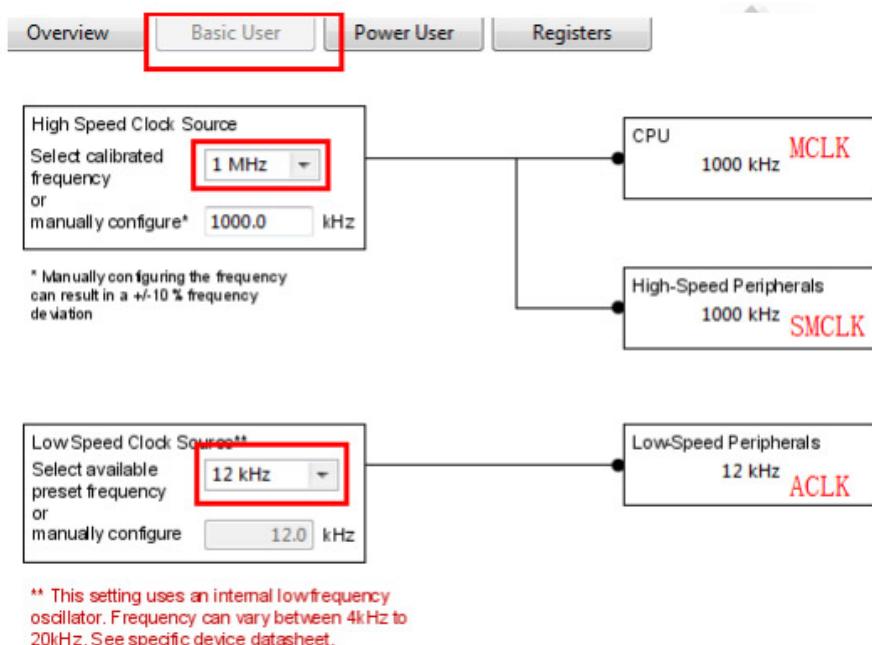


由上面的图，我们可以非常直观的看到，ACLK、MCLK、SMCLK是通过一些时钟输入选择之后，基础时钟模块输出的三个时钟信号，这三个时钟信号最终将提供给CPU和外设使用。而这三个时钟的最初来源，就是来源于VLOCLK、LFXT1CLK、XT2CLK或DCOCLK的。从上到下，观察基础时钟模块的结构框图，我们可以看到：ACLK的时钟来源有VLOCLK或者外接的晶振XT1产生的LFXT1CLK。

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）。

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。P1.0引

脚ACLK的实测数据为：1.43kHz左右。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fDCo可以高达20MHz，实际上MSPG2231的MCLK能不能跑到20MHz呢？

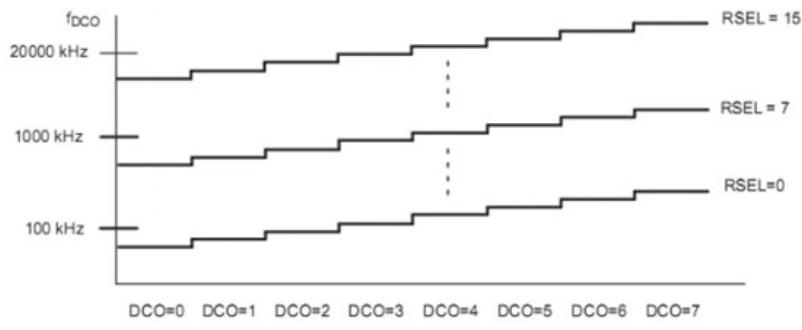
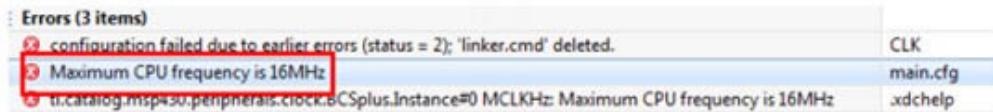
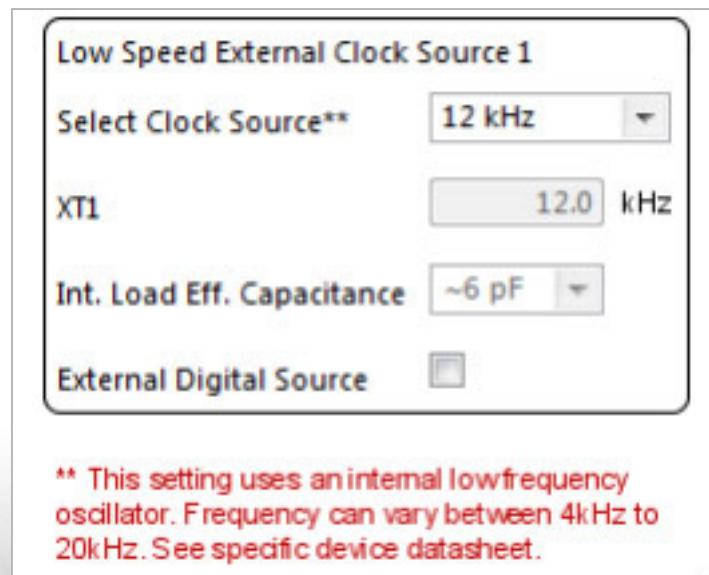


Figure 5-6. Typical DCOx Range and RSELx Steps

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）。

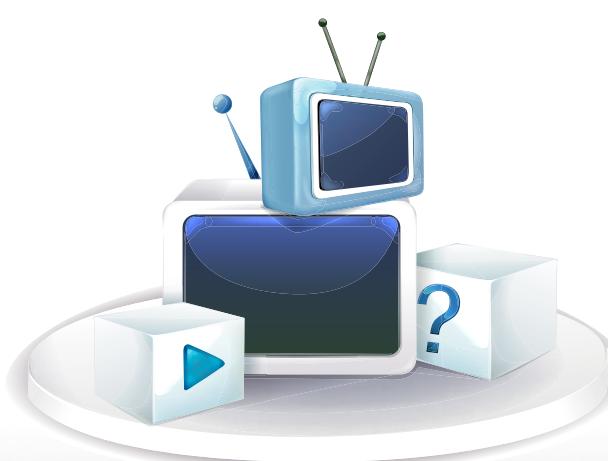
我们可以看到Grace对MSP430的初始化流程，而我们移植的过程，就是把这些函数复制粘贴的过程，这几个函数的源码在CLK\CCS\src\csl的几个.c文件里都可以找到，把它们处理之后，直接在IAR里新建工程，复制粘贴过去，即可使用。

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。

详情：<http://bbs.eeworld.com.cn/thread-314351-1-1.html>



1.5 LaunchPad-ADC10 介绍

1. ADC10 简介

ADC10 是MSP430单片机的片上模数转换器，根据其命名大家知道转换位数为10比特。该模块内部是一个 SAR 型的 AD内核，可以在片内产生参考电压，并且具有数据传输控制器。数据传输控制器能够在CPU 不参与的情况下，完成 AD数据向内存任意位置的传输。

2. ADC 特性

- * 最大转换速率大于 200kHz 。
- * 转换精度为 10位。
- * 采样保持器的采样周期可编程设置。
- * 利用软件或者 TimerA 设置转换初始化。
- * 编程选择片上电压参考源(2.5V 或者1.5V)。
- * 编程选择内部或者外部电压参考源。
- * 8 个外部输入通道(在MSP430x22xx 上有 12个)。
- * 具备对内部温度传感器、供电电压 VCC 和外部参考源的转换通道。
- * 转换时钟源可选择。
- * 多种采样模式：单通道、重复单通道、排序和重复排序
- * ADC 的内核和参考源可分别单独关闭
- * 用于自动控制数据传输的数据传输控制器

要掌握一个外设的使用，首先就需要清楚外设特点，以上就是MSP430自带的 10位AD的特性，这些特性将直接影响该 AD的应用场合，希望初学的朋友们对这些特点在开始的时候能有个感性的认识。

3. ADC10 操作

1、ADC10的转换值怎么计算

ADC10 通过两个可编程的参考电压 V_{r+} 和 V_{r-} 来定义转换电压的上下限。当输入电压大于等于 V_{r+} ，输出转换值为 3FF；当输入电压小于 V_{r-} ，转换值为 0。计算公式为：

$$N = (V_{in} - V_{r-}) / (V_{r+} - V_{r-})$$

2、ADC10的转换时钟选择

ADC10 的时钟源可以是 SMCLK、MCLK、ACLK 和内部的振荡器 ADC10OSC，可以用 ADC10SELx 来选择时钟源，并且设置 ADC10DIVx 可实现 1 到 8 的分频。振荡器 ADC10OSC 根据不同的器件有所不同，这个需要根据具体的器件选择。ADC 的时钟必须在转换的过程中保持有效，如果在转换的过程中时钟失效，转换结果是无效的。

4. ADC10 模拟引脚选择

这个操作也很简单，模拟输入引脚和普通的 IO 是管脚复用的，所以当模拟信号输入的时候普通 IO 的功能其实对信号是有影响的。你要做一件事把普通引脚功能关掉，使能 ADC10AE_x。

5. ADC10 的参考电压源

ADC10 是有内部参考源的，REFON = 1 时被打开。当 REF2_5V = 1 时，参考源为 2.5V；为 REF2_5V = 0 的时候 1.5V。当 REFOUT = 0 时，内部参考源可以输出到 Vref+。

当然 MSP430 肯定是可以使用外部参考源的，这个时候可以关闭内部参考源来降低功耗。MSP430 还有自动关闭的功能，当 ADC10 的内核在没有进行转换的时候，会自动关闭；ADC10OSC 也会自动关闭来降低功耗。

详情：<http://bbs.eeworld.com.cn/thread-308773-1-1.html>



1.6 实现带有TimerA 的UART功能

1. 摘要

本应用报告介绍了如何使用Timer_A实现UART功能。该包括例子是专门为MSP430x11x家庭,但他们能适应任何MSP430家族成员纳入Timer_A。使用硬件UART的功能在Timer_A功能和软件。执行是半双工的,事件驱动,它支持的8N1波特率协议从1200到115200或者更快。

2. 简介

异步串行通信,可以添加到MSP430x11x应用程序综合Timer_A模块硬件的功能。这份报告提供了一些的UART函数实例演示了一对一MSP430F1121闪存RS232接口单片机与PC机串行端口。一个描述如何使用Timer_A1硬件提供自动启动位检测,波特率生成和数据位锁存是详细。Timer_A硬件特性的软件,大大降低CPU开销通常与微控制器软件UART实现。硬件的功能也让Timer_A的UART操作作为背景的作用,同时与其他实时系统的任务。

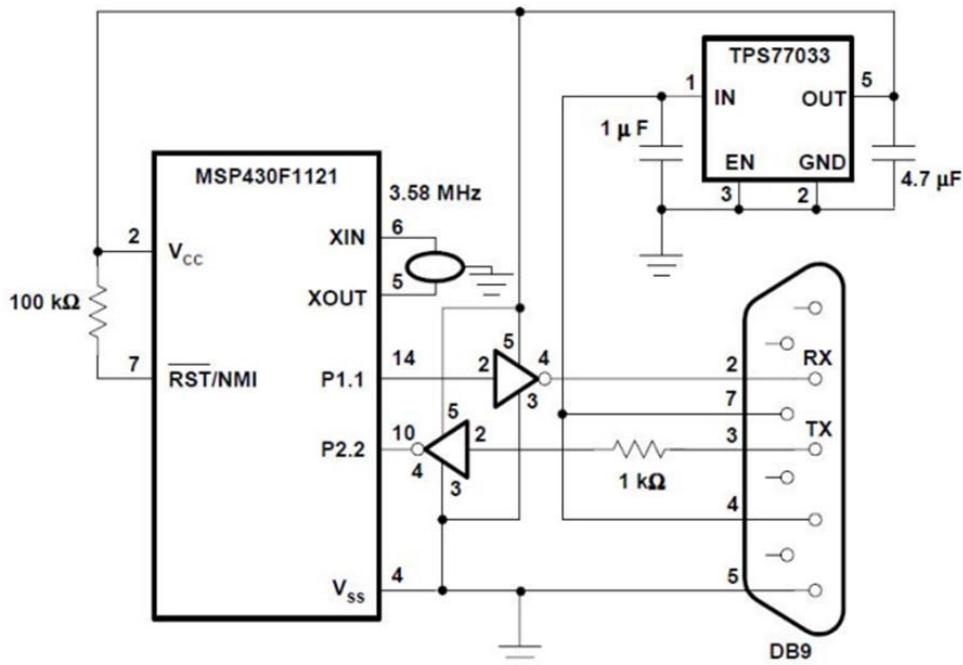


Figure 1. MSP430x11x(1) UART Demonstration Circuit

MSP430x11x Timer_A的UART描述:串行通信的MSP430F1121与另一个系统,在这报告电脑使用RS232接口。特点是两个系统之间交换通过三线:接收,传输和共同点。协议使用的字符是8N1:8个数据位,无奇偶校验,一个停止位。用户可以修改UART的功能支持其他协议和波特率,奇偶校验和包括一个9个位寻址。UART的功能描述使用捕获比较寄存器0(CCR0),Timer_A的三个可用寄存器之一。CCR0用于起始位检测,波特率生成和数据位锁存。另外两个捕获比较寄存器可用于其他事。该CCR0选择是任意的。任何或所有

CCR_x寄存器可用于UART的功能。端口引脚P1.1和P2.2是外设选项选择的关于Timer_A CCR0。 P1.1用于传输，P2.2用于接收。外设选项选定为使用外围设备选项选择引脚寄存器，P1SEL和P2SEL。由于P1.1置为输出时，该引脚必须配置为使用输出端口方向寄存器1 (P1DIR)。 P2.2作为运行需要输入。这是一个MSP430的端口引脚的默认。 Timer_A配置为运行在连续模式，允许定时器资源用于同时与UART等功能可用。中央处理器寄存器R4是为RXTXData -缓冲区使用的UART数据或出位的变化。 R5的CPU寄存器用于BitCnt，有点跟踪登记。 R4和R5的选择是任意的。任何CPU寄存器或RAM字节可以用于这些功能。

在接收模式下，捕获比较控制寄存器0 (CCTL0) 的初始配置，使得在下降沿接收引脚 P2.2 CCR0捕获。由于接收线路空闲高，一降边指示起始位开始。当UART功能已准备好接收数据，没有开销放在CPU的功能，即使是准备接收任何一个字符时间。 CPU资源执行后，起始位下降沿上P2.2发生。下降对P2.2边沿捕获了自由运行Timer_A计数器寄存器(焦油)的电流值，CCR0与任何其他运行时的活动无关。捕获是通过Timer_A硬件，而不是由软件。同时发出一个中断给CPU。中断的延迟并不大关注确切时间的下降沿触发中断正在CCR0存储，独立其他活动。启动后位的边缘检测，软件重新配置，使CCR0的CCTL0在第一种模式的比较发生在第一个数据位的中间。一个1.5位偏移量添加到CCR0，定位比较下的第一个数据位的中间。接收到的数据是同步的捕获比较输入 (SCCI) Timer_A锁存硬件。 SCCI是可读的CCTL0闩锁。在UART的功能，SCCI捕获的逻辑电平同步与CCR0输入P2.2进行比较。 UART的功能是从SCCI中接收锁存的数据。软件不直接测试P2.2。

RX_Bit	bit.w #SCCI,&CCTL0	; Get bit waiting in SCCI
	rrc.b RXTXData	; Store received bit

第一个数据位后，1位长的偏移量添加到CCR0定位在未来捕捉中间的每一位。八位连续的数据被锁存，并收到来自SCCI软件到RXTXData位的数据。发射模式任务是简单，因为MSP430在决定时，传输数据，没有启动位边缘检测的必要性。在缓冲区中存储的数据 RXTXData传输引脚P1.1使用Timer_A CCR0输出至硬件。

CCR0是预先在比较模式下位传输使用输出模式控制位在CCTL0。模式控制位被预先配置为复位模式 (逻辑0)，或设置模式 (逻辑1)发生在未来比较CCR0。当比较时，在一开始位传输的数据，自动输出CCR0硬件数据配置P1.1置位并发出一个中断。随着CCR0硬件自动输出数据上P1.1置位，软件中断延迟和位定时的关注减少。这不是必要的软件准备CCR0输出锁存器在一个确切的时间内。 UART的功能软件不直接对P1.1置输出数据，而是预装在CCR0输出数据位输出至锁存器的硬件。

```

TX_Bit      rra.b    RXTXData           ; LSB is shifted to carry
            jc       TX_Mark            ; Jump if bit = 1
TX_Space    bis.w    #OUTMOD2,&CCTL0   ; TX Space
            reti
TX_Comp    bic.w    #CCIE,&CCTL0      ; All Bits RX, disable interrupt
TX_Mark    bic.w    #OUTMOD2,&CCTL0   ; TX Mark
            reti

```

每个位输出之前，软件循环查询RXTXData到进位。适当的跳转提出和CCTL0输出模式准备和1位长的偏移量添加到CCR0。

3. 波特率计算

Timer_A CCR0用于波特率产生。根据所需波特率，间隔Bitime的计算方法。Bitime是Timer_A位和PC之间的计数长度时间间隔Timer_A闩锁在接收和发送出的数据位。Bitime是为Timer_A计算时钟的波特率分源。

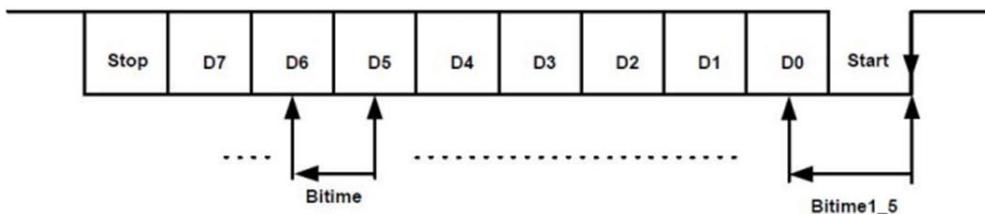


Figure 2. UART8N1 Character

Timer_A有几个可用的时钟源和除法(参见器件表具体数据)。可用于MSP430x11x Timer_A模块的时钟源包括辅助时钟(ACLK),子系统时钟(SMCLK)和两个外部时钟。

4. 软件开销

固件工程师可能会关切的是使用中UART的硬件/软件系统开销，而不是一个专用的串行端口。UART的功能在本报告所述用途timer_A硬件的功能，最大限度地减少CPU的负担。Timer_A硬件记录起始位优势和锁存和输出自动位的数据。该软件需要有一个26位的每接收或发送的最大周期包括中断服务程序。开销是CPU的时钟(MCLK)和波特率功能。使用3.58 - MHz的9600波特率例如上面的MCLK，开销计算公式为：

$$\text{软件开销} = (26 \text{个CPU周期}) \times (9600) / 3\,579\,545 = 6.9\%$$

5. 演示电路

图1的示例电路与调控，从直接供电由一个PC串口3.3 - V的TPS76033低压降稳压器。出于演示的目的，串口接口是通过使用两个TI SN74AHC1G04逆变器。如果一个完全兼容的RS232接口是必需的，如TI的低功耗3 - V的MAX2331集成电路都可以使用。复位拉高和3.58 MHz的陶瓷谐振器用于产生时钟使用。

6. 示例代码

所包含的例子11x1_uart1.s43使用图1中的电路，并提供了基本的回声功能。接收到一个字符从PC和回显。在初始化过程中，端口引脚配置和所有时钟都同步到LFXT1晶体振荡器。LFXT1是配置为在高速运行(高频)模式。如果在这个例子中，MCLK的源是一个外部高频晶体，晶体必须是稳定或振荡器故障安全模式会自动使用FOR MCLK的DCOCLK。该OSCFAULT可以查询，以确保稳定的晶体前选择此MCLK的时钟源。

```

SetupBC    bis.b  #XTS,&BCSCTL1           ; ACLK = LFXT1 HF XTAL
SetupOsc   bic.b  #OFIFG,&IFG1            ; Clear OSC fault flag
           mov.b  #0FFh,R15
SetupOsc1  dec.b  R15                  ; Ddelay to ensure startup
           jnz    SetupOsc1
           bit.b  #OFIFG,&IFG1            ; OSC fault flag set?

```

振荡器故障安全模式中描述的MSP430x1xx系列用户指南(SLAU049)。该主循环调用子程序RX_Ready UART接收准备，然后在低功耗模式0等待(LPM0)与CPU关闭。只有Tmer_A1和ACLK活跃。即使在CPU中关闭主循环中，UART接收中断操作功能在后台驱动。后UART功能接收到RXTXData，UART的中断处理程序的完整的字符返回到主循环活动的CPU。发送子程序被称为未来，又呼应RXTXData回接收到的字符。循环重复，另一个字符等待收到。该示例使用的编码速度优化技术研究。内CCR0_ISR，BitCnt用于自动递增的间接寻址，直接程序流的确切节ISR的规定办理接收或发送位。软件是不需要投票登记标志或递减，以确定采取何种行动。自动递增寻址使用一个查找表，直接的方案立即流。

```

add.w  #Bitime,&CCR0          ; Bitime till next bit
br     @BitCnt+               ; Branch To Routine

```

该自动递增寻址的优点是速度和CPU的周期计算效率在需要处理为查找表所需的费用在ISR码字。

更多详情：<http://bbs.eeworld.com.cn/thread-214427-1-1.html>

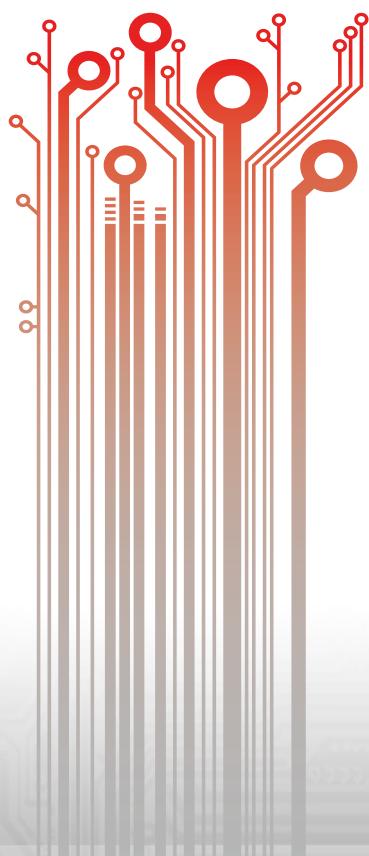
1.7 LaunchPad另一种uart的实现-串行接口

首先声明，所谓的另一种是，是相对于官方温度传感例程里面的方法来说的。官方程序里面大致是这样的，设为连续计数模式，TXD,RXD作为TA的捕获/比较功能引脚，当捕获到起始位或要发送数据的时候，在当前计数值的基础上加上一位的宽度来设定下一个中断时间，然后根据要发送的值来确定TXD功能引脚的输出。这种方法充分利用了硬件资源，但是相对应的要设置的寄存器和中断要处理的稍多点，使用起来稍显复杂。

其实uart通讯很简单，起始位加8bit数据加上停止位（不是特别的应用或高速率传输时一般可以省掉校验位）就好了，在起始位（高电平向低电平转换时刻即下降沿）开始按数据置高或拉低一个数据位宽度（由波特率决定）即可。

所以这里就简单的利用crr0的up mode中断，产生一个数据位的时长，我们只需要把要发送的数据，前后分别加上起始位和停止位按低位在前，高位在后的顺序发出去就好了。

更多详情：<http://bbs.eeworld.com.cn/thread-309735-1-1.html>



1.8 LaunchPad风火轮触摸板初体验

在与MSP-EXP430G2 LaunchPad 实验板初体验之后,我又在TI官网看到一个关于LaunchPad实验板很好玩的视频,就是将风火轮触摸板置于LaunchPad 实验板之上进行触摸操作,不仅能实验风火轮演示,还能通过触摸板控制电脑中的默认视频播放器。体验步骤:

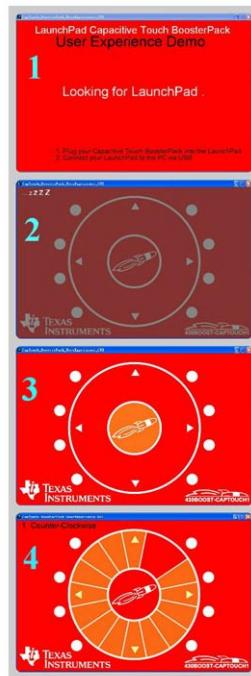
①、硬件准备:将MSP-EXP430G2 LaunchPad 实验板附件中的两条排针分别焊在实验板的J1和J2上,将实验板上的J5的两短路片取下,将风火轮触摸板中附带的M430G2452(TI的20脚单片机,内部已经装有实验程序了)插入实验板的IC1中,最后将风火轮触摸板正确插入实验板上。



②、软件准备:上TI官网下载MSP-EXP430G2 LaunchPad 实验板驱动、风火轮触摸板上位机模拟演示程序、风火轮触摸板MediaPad驱动和风火轮触摸板使用说明书,并安装LaunchPad 实验板驱动。

③、我们首先演示风火轮触摸板通过上位机显示触摸按键及范围,点击运行官方网站下载的演示程序:

CapTouch_BoosterPack_UserExperience_GUI.exe 会出现下图1界面,将LaunchPad 实验板通过数据线与电脑连接,会出现下图2界面,这时用手指触摸风火轮中间位置,会出现下图3界面,线续触摸风火轮小圆外大圆内位置,会出现右图4界面。



④、演示风火轮触摸板控制电脑中的Windows Media Player播放器。在演示之前我们首先要点击运行官方网站下载的程序:430Boost_CapTouchMediaPad.exe,之后将LaunchPad实验板通过数据线与电脑连接,按照下图标注功能进行演示。

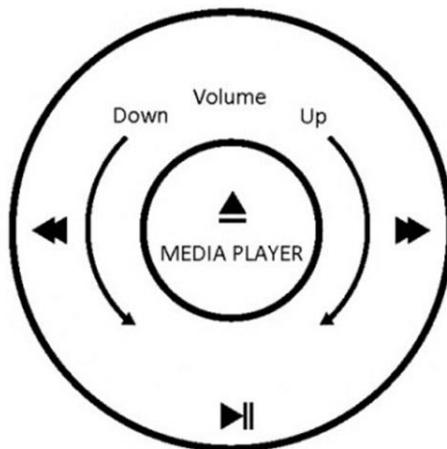


Figure 5. MediaPad

活动详情: <http://bbs.eeworld.com.cn/thread-310408-1-1.html>



1.9 LaunchPad_实验板触摸感应子卡使用指南

1. 概述

LaunchPad Touch的触摸感应子卡是一个基于Texas Instruments MSP430G2xx 系列的完整的触摸感应应用开发方案。触摸感应子卡可以被插入现有的MSP-EXP430G2 LaunchPad 实验板来实现一个触摸感应的应用。进行简单的硬件配置和使用不同的软件，可以实现按键/ 滑条/ 拨号盘的不同应用。在MSP-EXP430G2 LaunchPad实验板套件中采用了MSP430G2211 微控制器；利用 MSP430G2211 的内置比较器来完成触摸感应，同时利用计时器来控制LED 闪烁频率以及LED 的亮度作为反馈。

MSP-EXP430G2可以运用IAR Embedded workbench 集成开发环境 (IDE) 或 Code Composer Studio (CCS) 来进行代码编写，程序下载和调试。调试程序非常简单易用，不需要任何额外的硬件资源，允许用户全速运行程序，设定硬件断点以及单步执行等。

触摸感应子卡特性：

- * 实现触摸按键/滑块/拨号盘的功能
- * 通过2 个通用数字I/O 口来驱动2 颗LED 用于视觉反馈
- * 支持所有PDIP14 封装的具有片内比较器模块的MSP430G2x11 和MSP430F20x1 型号
- * 能轻松连接到MSP-EXP430G2 LaunchPad 实验板用于调试

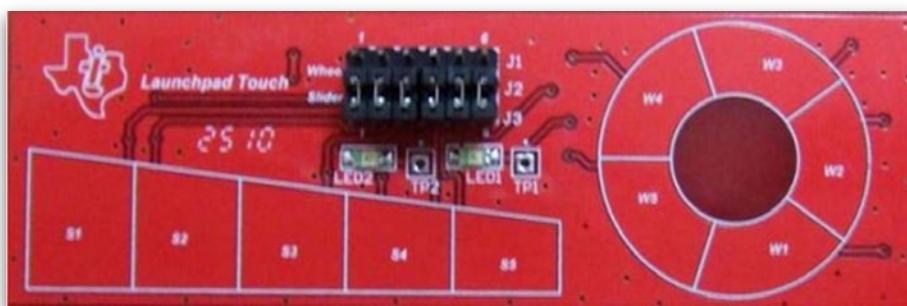


图1. 触摸感应子卡外观图

套件内容

触摸感应子卡套件包含了以下一些配件：

触摸感应子卡

另外，套件附送的MSP430G2211 芯片可以帮助您马上体验EXP430G2的各项功能。MSP430G2211 是超低功耗16位微控制器MSP430 中的一员，拥有片内比较器，2k字节程序空间和128 字节的SRAM 。

2. 安装

硬件安装

首先,请准备好MSP-EXP430G2 LaunchPad 实验板。

移除(1)(2)(3) 和(4)号跳接帽,参考图2。

用烙铁移除电容(5)和电阻(6),参考图2。

将两个公的10脚连接器焊在MSP-EXP430G2 LaunchPad 实验板上。按照正确的方向将你的MSP430G2211 芯片放入插座中!插入触摸感应子卡,确认连接正常,参考图3。

将mini-USB连接线插入LaunchPad板上。如果你已正确安装已认证的 VCP驱动在您的电脑上, FET 将不会要求Launchpad 板的手动安装:

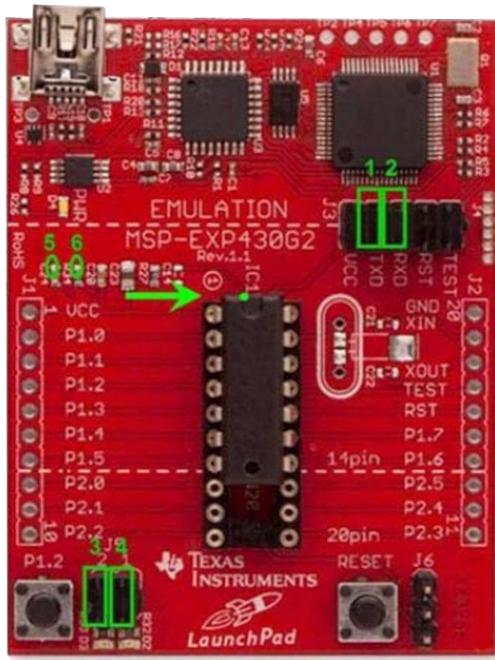


图2. MSP-EXP430G2 LaunchPad 实验版设置示意图

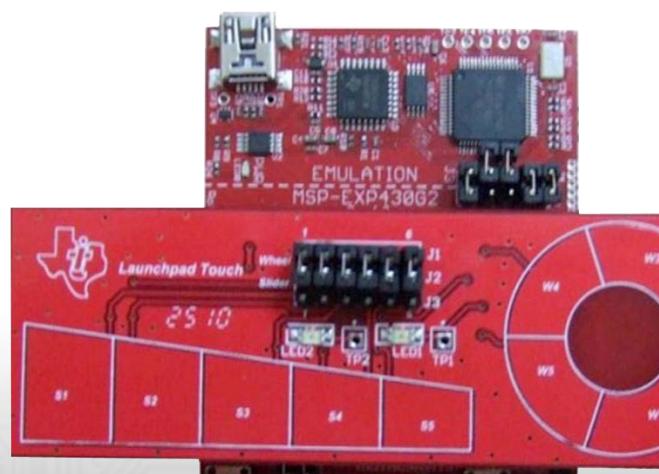


图3. 已安装在 LaunchPad实验板上的触摸感应子卡

软件安装

下载IAR KickStart 或CCS V4 中的任何一种集成开发环境软件,他们都提供了在用于仿真MSP-EXP430 LaunchPad所需的驱动程序。一旦安装完成,IDE 会将MSP-EXP430G2 LaunchPad识别为USB:HID调试界面。至此,基于MSP430G2xx ,将其触摸感应子卡接入LaunchPad实验板的触摸感应运用开发环境已准备就绪。

3. 开始您的触摸感应用

首先,将MSP-EXP430G2 LaunchPad用自带的mini USB连接线接入计算机一个空闲的USB接口。以下的章节将会

演示如何建立一个MSP-EXP430G2 LaunchPad 的工程,并将相关的应用程序下载到MSP430G2211 。 在例子程序中LED 将会随着对触摸感应子卡上W1到W5或S1到S5的触摸进行闪烁并进行亮度调节。我们将会基于IAR kickstart来演示这个应用程序。请按照下面的顺序打开IAR Embedded Workbench ;打开Windows 的Start 菜单→ Programs → IAR Systems → IAR Embedded Workbench Kickstart for MSP430 Vx.x → IAR Embedded Workbench)。

1. 建立一个新工程:通过Use Project → Create new project 来设置你新工程的工程名以及在电脑上的保存目录。最重要的是,要使用→ Add files 来添加工程项目Cap_Sensing_Touch_Key_V1.0.c ;

2.选择正确的芯片型号:通过点击Projects → Options → General Options → Target 并在设备列表中选择MSP430G2211 来设置正确的芯片型号。

3.设定正确的调试工具:在 Options窗口里,进入 FET Debugger → Setup → Connection 并选择TI USB FET来使用USB接口。

4.保存设定好的工程:使用File → save the workspace 来保存你自己的workspace 文件。

5.编译工程:使用 Project → Rebuild All来编译和关联源代码。你可以通过双击project,然后再双击显示的源文件来读取源代码。

6.下载代码到芯片中:使用Project → Download & Debug来开始C-SPY调试器。C-SPY 会擦除设备的闪存,然后将程序文件下载到设备闪存里。

7.调试程序:使用 Debug → Go 来开始程序。LED 会随着触摸W1到W5而闪烁! LED1 会变化闪烁的频率;LED2会随着相应的按键显示不同的亮度。

8.退出调试:使用Debug → Stop Debugging来停止调试,退出C-SPY且回到Workbench。

9.退出开发环境：使用File → Exit 来退出Workbench。

恭喜您，您已经顺利创建并且完成了您的第一个MSP430 触摸感应应用。

通过将6 个跳接帽的位置从滑轮边改变到滑块边，然后通过复位按键对芯片进行复位，您就可以测试S1到S5的触摸键功能了。你不需要下载任何代码，LED 会随着触摸S1到S5而闪烁！LED1 会变化闪烁的频率；LED2 会随着相应的按键显示不同的亮度。

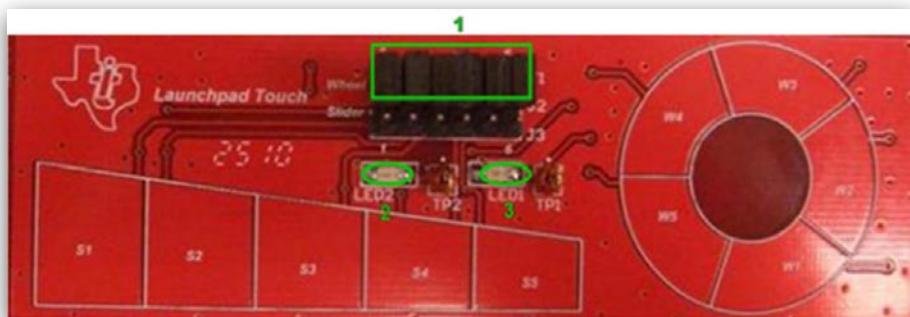


图4. 触摸感应子卡设置示意图

开始触摸感应滑块/ 拨号盘的应用

重复上面的操作来设置一个新的MSP-EXP430G2 工程并将Cap_Sensing_Slider_Wheel_V1.0.c 下载到MSP430G2211。你将可以体验触摸感应滑块/ 拨号盘的应用。

更多详情：<http://bbs.eeworld.com.cn/thread-309941-1-1.html>



1.10 MSP430 LaunchPad触摸板试用心得

1. 硬件环境搭建

(1)、将开发板上的G2231芯片换成附赠的G2211，因为G2231不带比较器，只有G2211才带，2231带SPI和串口等。

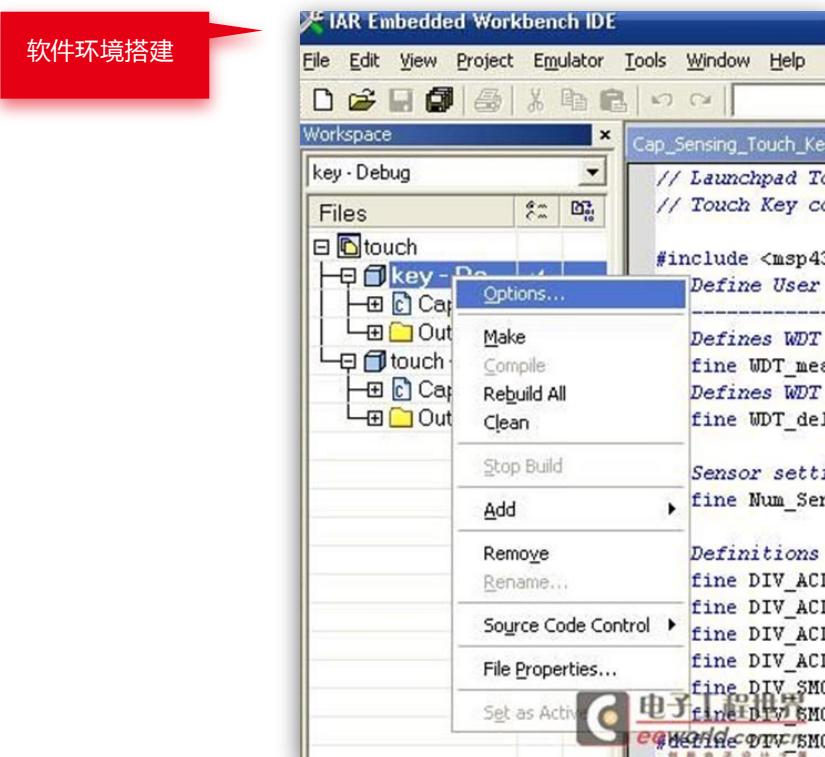
(2)、将开发板上的R34和C24焊下，焊接时小心，我就是焊接时把PCB焊坏了，这两个贴片是焊补上去了，到时只有飞线了，郁闷:(。

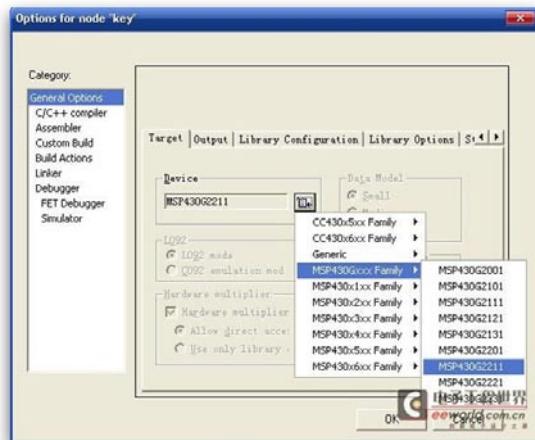
(3)、改跳线，将开发板的J3的RXT、TXD断开，将J5的两个跳线断开。

(4)、将LaunchPad TOUCH子板插入到开发板上。

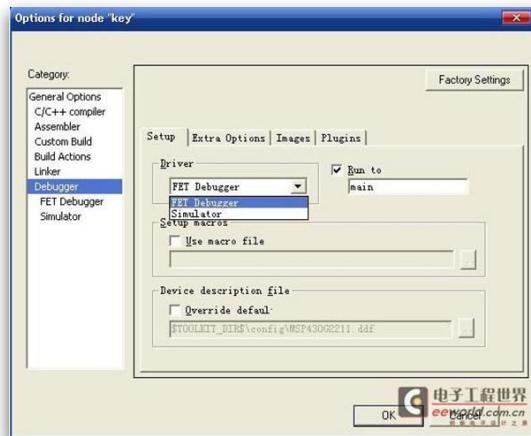
2. 软件环境搭建

我是使用的IAR5.1，没有使用CCS，安装好后，新建一个WORKSPACE，再新建一个PROJECT，加入现有文件将TI的例程Cap_Sensing_Touch_Key_V1.0.c加入，修改项目的选项。





编译程序，主要修改仿真项
如下，不然程序
只是在电脑上
仿真不能下载
到板子上。



好了，现在大家可以在线仿真测试触摸板的魅力了，感觉LED闪屏的频率分级不明显，一共两个工程，另一个需要跳触摸屏子板上的跳线，下面分享我已经建好的工程文件和我的参考资料。



触摸感应测试工程.rar (49.19 KB)



LaunchPad实验板触摸感应子卡使用指南.pdf (1.05 MB)



G2用户手册.pdf (534.17 KB)



官方示例源码.zip (264.01 KB)

更多详情：<http://bbs.eeworld.com.cn/thread-221644-1-1.html>

第二章 官网课程篇

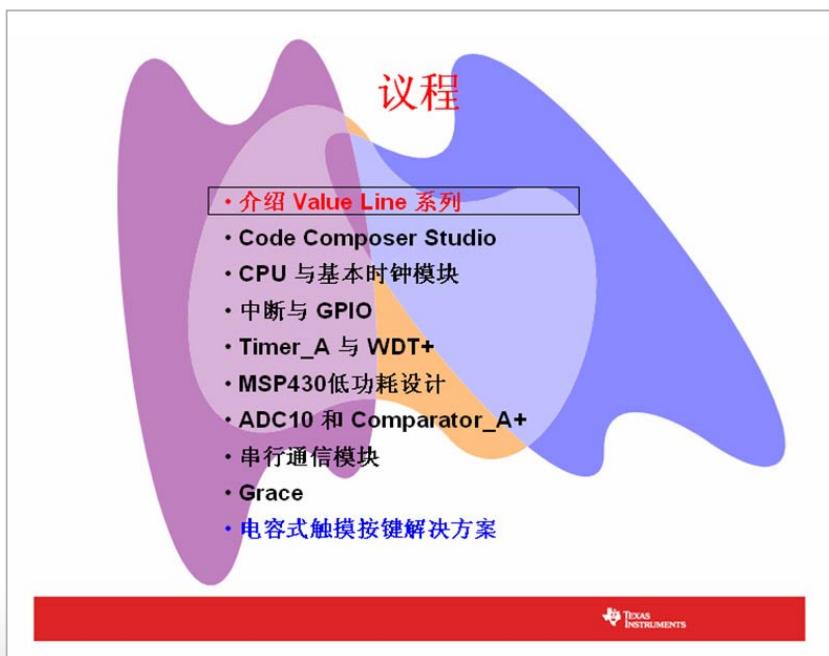
2.1 课程前言

微控制器产品门类浩瀚如海。一直以来，工程师对如何选择适合的微控制都十分头疼，而使用惯了一个系列的产品以后，也不愿意更换不熟悉的平台。我们需要快速并深入的了解更多微控制器产品，以备不时只需。

作为全球领先的半导体厂商TI(德州仪器)的微控制器产品大家已经耳熟能详，主要包括低功耗的MSP430，适合于实时控制的C2000，基于ARM的Stellaris产品，以及最新推出的定位于安全应用的Hercules产品。本次为大家着重介绍的是MSP430产品，并用物美价廉的430 Launchpad试验版，和大家一起通过动手实验来掌握和学习使用MSP430。

本书为TI MCU产品技术经理丁京柱基于MSP430内核和MSP430LaunchPad的开发讲解课程，对于已经有微控制经验的工程师来讲，您可以从本次培训中来建立对430的了解，并很快可以通过动手做一些简单的实验，并为以后做一些简单的项目打下坚实的基础。

本书主要包括Value Line产品的介绍；集成开发环境CCS；CPU与基本时钟模块；中断与GPIO；Timer-A与增强型的看门狗；430低功耗设计的实践基础；以及外设10位的ADC和增强型的比较器；串行通信模块的介绍；TI的图形化编程工具Grace软件；以及基于430 LaunchPad和电容式触摸按键解决方案。



2.2 Value Line产品的介绍

MSP430系列MCU产品，目前已经有将近400款产品，价格最低的只需要25美分，产品特性最高可以达到256KB flash, 18KB RAM, 有25种封装可供选择，最高可以支持113pin，拥有非常高的集成度。MSP430是一款16位RISC CPU，包含了很多的片上外设，比如16位的定时器，片上的看门狗，各种智能的片上的模拟和数字外设。

目前，最新推出的MSP430微控制器是基于FRAM的产品，可以让目前MSP430的功耗比现在的功耗降低一半。比如目前MSP430的功耗大部分是每MIPS 250 μ A，而我们基于FRAM的MSP430系列产品，每MIPS只需要100 μ A左右，具有超低功耗的特性，也可以说是目前星球上最低功耗的微控制器。

MSP430 系列MCU产品

MSP430 | Ultra-Low Power is our DNA

MSP430 Portfolio at a glance

300+ Ultra-Low Power Devices Starting @ \$0.25USD
Featuring: Up to 256kB Flash, 18kB RAM, 25+ Package Options, Up to 113 pins, High integration

— Ultra-Low Power Performance —		— Analog Integration —		— Easy-to-Use —	
MSP430 16-bit RISC CPU All devices feature: • 16-bit timers • Watchdog timer • External Digital Controlled Oscillator • On-chip 2.5V reference • On-chip crystal oscillator • On-chip temperature sensor • 16-bit wide ADC	L092 Speed 4MHz ROM to 256B RAM to 256B GPIO 11 F1xx Speed 8MHz Flash 1-60kB RAM to 8kB GPIO 14-48 F2xx Speed 16MHz Flash 1-120kB RAM to 8kB GPIO 10-64	G2xx Speed 16MHz Flash 0.5-16kB RAM to 256kB GPIO 10-16 F4xx Speed 8/12MHz Flash 1-120kB RAM to 8kB GPIO 14-80	F5xx Speed 25MHz Flash 8-256kB RAM to 16kB GPIO 14-80	CC430 Speed 20MHz Flash 8-256kB RAM to 4kB GPIO 48 CC430 Speed 20MHz Flash 8-256kB RAM to 4kB GPIO 48 Comp. B RTC, AP	
ROM I ₂ C SPI PWM ADC DAC PWM DAC SPI USART	ROM I ₂ C SPI PWM ADC DAC PWM DAC SPI USART	ROM I ₂ C SPI PWM ADC DAC PWM DAC SPI USART	ROM I ₂ C SPI PWM ADC DAC PWM DAC SPI USART	ROM I ₂ C SPI PWM ADC DAC PWM DAC SPI USART	ROM I ₂ C SPI PWM ADC DAC PWM DAC SPI USART
All Devices Some Devices					

另外，MSP430还有有低电压的L092系列产品，它的工作电压是0.9V到1.6V，速度最高可达到4M，ROM最高是2K，RAM最高也是2K。支持11个GPIO口。因为低电压的特性非常适合于支持单节电池供电的应用，在这一系列产品中，包含通用产品的F1系列和F2系列。F1系列的速度支持8M，F2系列最高可以支持到12-16M。F1系列最高达到60KB的片上flash，

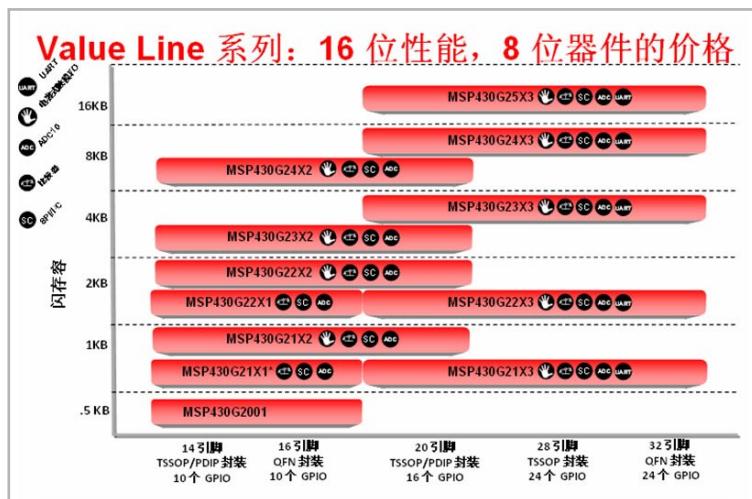
MSP430 支持的应用

公用事业计量 电能表 燃气表 流量表 智能计量仪表	MSP430 Ultra-Low Power MCUs 	便携式医疗 血糖计 温度计 心率监测计 可植入装置
无线应用 远程传感器 通讯控制器 RFID	MSP430 MCU 可支持数以计的应用 凭借 MSP430 MCU 的超低功耗性能、高集成度模拟与数字外设、以及易用的工具，客户可方便地实现其产品的差异化	传感器与安全 烟雾探测器 运动探测器 振动检测器 智能传感器
消费类电子 便携式电子产品 遥控器 个人保健 PC 外设	能量收集 可再生能源 无电池设备 太阳能、热能、振动能，等等	个人健康与健身 运动手表 计步器 热量计 潜水手表

F2系列最高可以支持120KB的片上flash。

本书的重点是G2XX系列，也是Value Line系列产品。

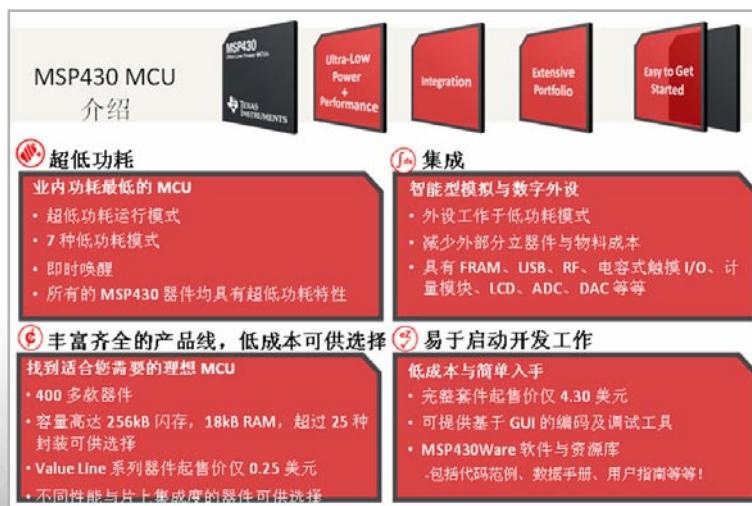
Value Line是指低成本的系列，其速度可以最高达到16M，flash容量最高达到16KB，GPIO口可以从最低10到最高24个。



MSP430家族有通用的5系列和6系列产品，是目前市场中广泛运用的产品。针对一些ASSP的需要，如水表、气表、热表，以及医疗产品血糖仪等，MSP430家族还有F4系列产品提供支持。F4系列产品的最大特点是具有片上LCD模块，整个系列产品最高有两种速度，8M和16M，flash最高达到120KB，RAM最高可以达到8K。

由于产品的革新，MSP430在5系列产品中也内置了LDO模块，可以让CPU内核工作在更低电压，其速度可达到25M，flash可以最高达到256KB。另外，512KB的5系列产品也将很快面试。针对一些特殊应用，430的产品序列中也有CC430系列产品，它提升了片上transceiver芯片，可以很容易的实现无线射频的连通性。

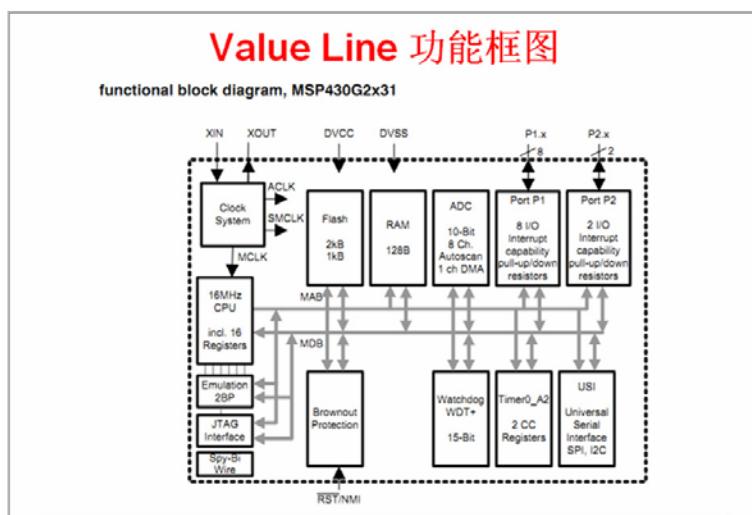
430微控制器总体而言，拥有四大优势：第一，超低功耗；第二，丰富齐全的产品线，可选择性高；第三，具有非常高的集成度；第四，非常易于启动我们的开发工作。



430的超低功耗前文已有所谈及，每MIPS的功耗约为 $250\mu\text{A}$ ，最新推出的FRAM系列产品，功耗达到每MIPS $100\mu\text{A}$ ，同时有多种超低功耗模式可供大家选择，如LMP0、LMP3、LMP4等。从低功耗到快速运行的模式，能够实现快速快减的功能。

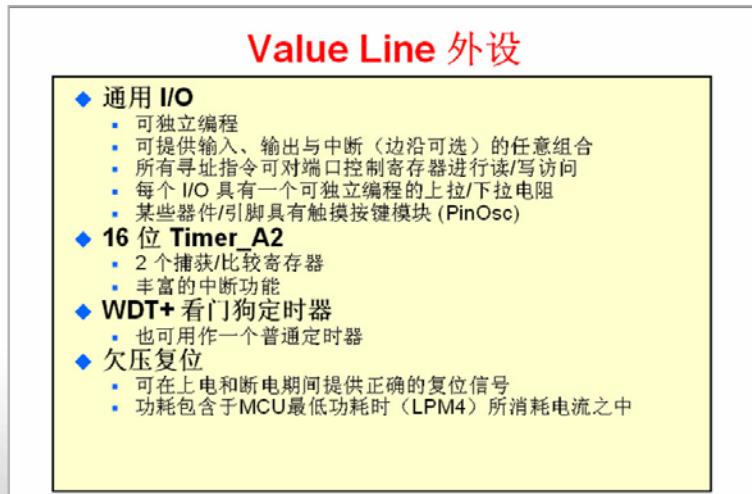
430家族目前已经拥有400多颗器件，容量最高可达256K闪存，18K RAM，超过25种封装，另外也提供超小型的芯片尺寸封装。

MSP430 Value Line系列产品最低的起售价25美分，在高集成度的同时拥有不同的性能特性，不同的产品可以满足不同设计的要求。



MSP430片上的集成度包括智能的模拟数字外设。所谓的智能，是指MSP430的外设是根据低功耗设计可以自动开启和关断，所有外设可工作在低功耗模式。采用片上外设可以有效减少外部分立器件与物料成本，从而可以让系统功耗更低。

MSP430包含的外设包括FRAM、USB Device接口、射频接口、电容式触摸I/O即通常所说的Cap Touch I/O、通常用在电表计量的ESP模块、可以很方便的实现段码式LCD显示的LCD module、片上有ADC和DAC。其中ADC是MSP430最大产品特色，目前有10位、12位包含16位的ADC，近期也在推出24位的片上ADC的芯片。

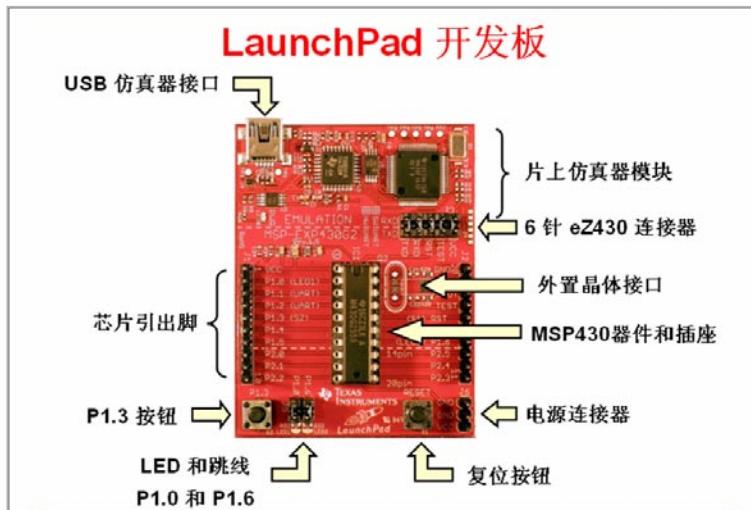


Value Line 外设

- ◆ 串行通信
 - 支持 I2C 和 SPI 的 USI
 - 支持 I2C、SPI 以及 UART 的 USCI
- ◆ Comparator_A+
 - 可设定反相和同相输入
 - 可选的 RC 输出滤波器
 - 可直接输出至 Timer_A2 捕获输入
 - 具有中断能力
- ◆ 8 通道/10 位 200 ksps SAR ADC
 - 8 个外部通道（取决于器件）
 - 内置电压和温度传感器
 - 可编程的参考电压
 - DTC可在无需 CPU 干预的情况下将结果发送至存储器
 - 具有中断能力

MSP430的开发成本极低，这包括TI提供的低成本的开发板，例如LaunchPad开发板的最低售价只需要4.3美金，以及基于功率的编程和调试工具，如基于CCS的Grace软件。

TI最新推出了430ware软件，它可以让开发者非常方便的使用TI提供的的代码范例，包含数据手册以及用户选择指南。



430微控制器可以支持数以千计的应用，超低功耗特性和高集成度的外设，可以满足产品差异化的设计原则。在一些典型应用如各种的公用事业计量，包括电能表、燃气表、流量表以及各种智能的计量仪表等领域；无线应用包括常用的RKE和PKE、RFID，包括目前热点的物联网领域，也是430非常好的应用环境和应用场合。

同时，430也被广泛应用在消费电子产品中，如说各种便携式电子产品，各种遥控器、个人保健产品、以及各种PC外设等。

最后，在便捷式医疗领域中，430也有不俗表现。在当今世界大家非常注重健康的环境下，430非常适合一些便携式的医疗产品设计，例如血糖仪等产品。

2.3 集成开发环境CCS

无论做任何设计，我们都需要一个开发环境，这是我们最基本的工具。

CCS是TI独有的集成开发环境，支持TI的所有嵌入式产品，TI全线产品都可以采用CCS进行开发设计。

CCS集调试器、编译器、仿真器、编辑器和操作系统于一体，属于集成型开发环境(IDE)，该开发环境是基于Eclipse的开源式软件框架，由TI进行扩展，支持TI做嵌入式产品，不仅包括MSP430系列，也包括C2000、Stellaris、Hercules，甚至包括TI的C5000、C6000 DSP产品，甚至包括OMAP等各种高端的嵌入式控制器等，均可以用CCS进行开发。

目前，TI的CCS版本主要为CCSv5，CCSv5是基于成熟的Eclipse的平台，我们现在是基于Eclipse version3.7的版本，也会随着Eclipse版本的升级进行提升，CCS将使用最新的版本Eclipse平台。

由于CCS是基于开源的Eclipse平台，用户能够充分的体会到Eclipse的最新特性，比如说对中文的支持，而在之前的CCS版本，比如在最早的3.3或者更高的版本中则很难支持中文；基于Eclipse平台的另一个优势是可以支持更多的操作系统，如支持Linux、Android等多种系统均可支持；另外，CCS还有一些更为优越的工具，如代码分析及各种源的控制。

对于开发者选择开发环境而言，CCS最大的好处是，相对于市面上常见的一些集成开发环境而言，能够以较低的价格满足用户的要求。CCS目前有两个价格版本，一个是495美金，一个是445美金，对于开发环境而言，可谓物美价廉。

图2.1的右下角可以看到CCS启动界面，一个小小的魔方框图代表了CCS的LOGO。



图2.1

从图2.2中我们可以看到，CCS的界面非常友好，对常见任务的处理也非常简单。创建工程项目可以使用模板轻松实现对器件的创建新项目工作。而从编译的Build的过程来看，也可以进行各种配置实现Build的各种过程。

另外,CCS在使用过程中不需要对CCS不停地升级,可以减少用户在升级过程中的麻烦。它的升级是依赖于用户对不同器件、编译器的升级而升级。

举例来看,例如目前用户使用的CCS为5.1版本,内置的430C编译器是4.0版本,假如TI发布新的430器件,用户只需要升级内置的430C编译器即可,不需要升级CCS,这样的设计让用户用起来更加的方便。

此外,CCS有更加易用的共享项目功能,可以实现不同用户和不同应用模块中间进行共享。如,我们最常见的版本控制,同时可以进行解剖操作实现共享链接资源等。

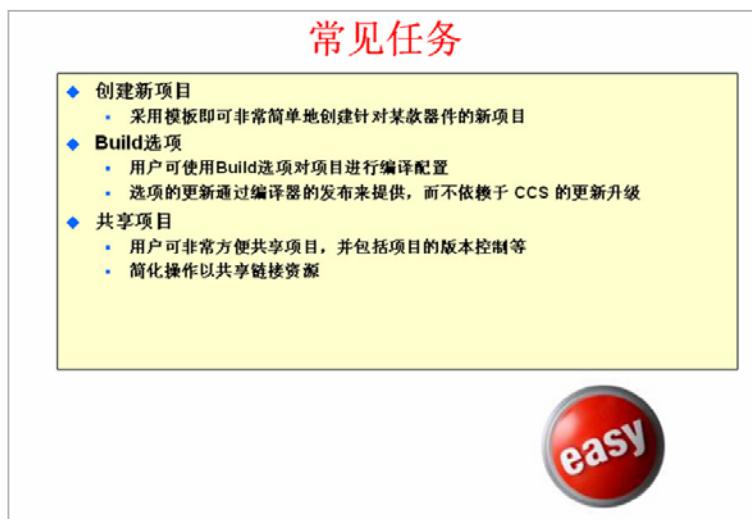


图2.2

图2.3介绍了CCS的管理方式,清楚的展现出CCS工作空间、项目、文件之间的关系。

在CCS中,首先会生成一个工作空间,通过工作空间可以产生不同的项目。如第一个、第二个、第三个项目,不同的项目可以进行不同的设定和偏好,其中包含项目的链接。

在项目中,通常包括源文件、头文件和库文件,并可以设置不同的Build的选项。项目通常是由源文件、头文件和库文件组成,这种管理是通过曲线链接的方式实现。

图2.3中表示出,对于链接的操作,无论是在工作空间还是在项目之间,通常我们在应用过程中,如果只删除了链接,而不会删除设计文件,这一点对于工作空间和项目都是如此。

我们在设计过程中,做一些编辑的过程中,删除的通常只是链接,而不是设计文件,这样的设定能够对我们的知识产权起到很好的保护作用。

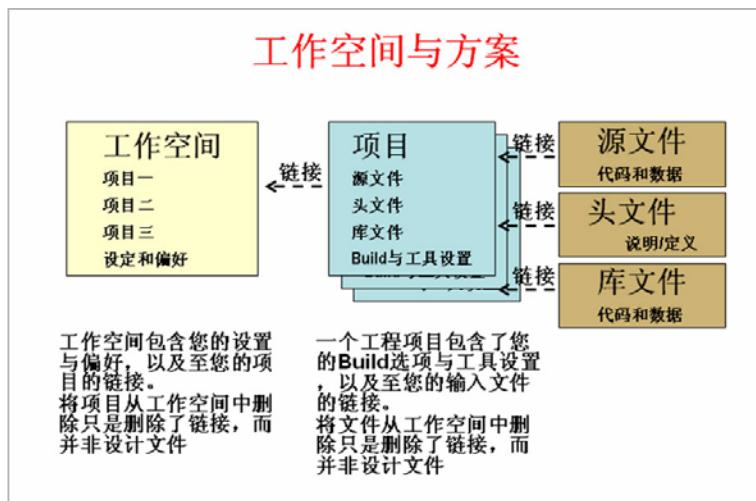


图2.3

如何去创建一个项目？

CCS有一个简单易用的项目向导，就像一个魔术棒一样，很快而方便的建立基于某个器件的项目向导。

通常而言，CCS有单页向导可以满足大部分情况的使用要求，我们在操作过程中只需要按“下一步”这样简单的方式就可以完成一个工程项目的建立。另外，每个过程、器件选择好之后都包含相应所生成的CCXML文件，也就是对器件资源的配置。

通常在项目向导中并不需要进行特别配置，如果需要使用一些高级选项，可以在高级的配置中，比如Advanced Settings依次配置编译器的版本、字节存储器等项目。

需要特别强调的是，在我们初步使用过程中，不需要做特别的设置，只有我们对现有的项目了解非常透彻之后，才需要进入到这种高级配置选项下去做选择。

图2.4是项目向导简单的框图，只要一个界面就可以完成一个工程项目的建立。

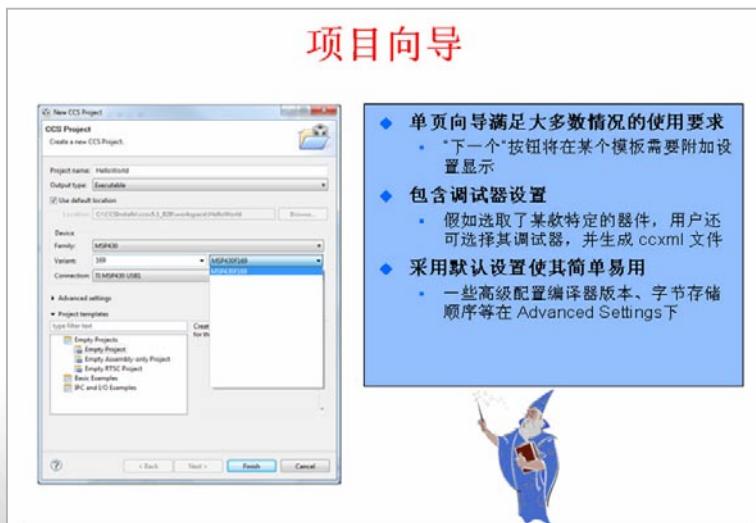


图2.4

TI有不同的嵌入式产品,包括不同的微控制器,目前所有嵌入式产品都可以基于CCS进行开发。对于微控制器产品而言,TI还可以满足一些小型化项目的需求,提供一些免费的集成开发环境和各种IDE。除了上述的标准版CCS以外,TI的CCS也可以提供免费版供,如免费的16KB代码空间限制的版本,以及根据特殊的需要,提供在限定时间的情况下短时限版本可供选择。

图2.5所示,除TI提供的集成开发环境CCS之外,TI还与第三方合作,如IAR公司,TI的430产品可以使用IAR工具进行开发。IAR是一个功能强大的C语言编译器工具供应公司,它的IAR集成开发环境简单易用。IAR Embedded Workbench配有项目管理工具和编辑器,适用于所有430家族的所有器件。同时,TI的网站上也提供一些有空间限制的版本供下载。对一些有全功能要求的用户而言,也可以提供30天试用期限的版本,从IAR的网站上可以下载。



图2.5

如果对编译器的成本有更高要求,TI在GNU平台上基于430免费开源的GCC工具链。比如有GNU的C编译器,汇编器和链接器等等,另外该工具可以在Windows、Linux、BSD及其他大多数Unix版本的各种操作系统上使用。

更多的详情请访问MSPGCC网站。如果想了解更多关于430开发工具包括集成开发环境更多的信息,请大家访问www.ti.com/msp430tools网站。

2.4 CPU 与基本时钟模块

本章将为大家介绍MSP430 Value Line G2XX系列的CPU与基本时钟模块。

图3.1是MSP430 Value Line G2XX系列结构框图。该系列产品具有如下特性：

第一，超低功耗，该系列产品包含0.1uA的断电模式，在这一模式下功耗只需要0.1uA，而待机模式下的功耗为0.8uA，全速运行下每MIPS是需要220uA。另外，具有从启动待机到唤醒时间小于1us的启动时间，小于50nA的端口漏电流及零功耗欠压复位(BOR)功能。

第二，超灵活，所谓灵活就是我们可选择的空间范围很大，比如从flash 0.5kB至16kB的片内可编程闪存可选择空间。16位的定时器，包括SPI、I2C、10位ADC和嵌入式仿真功能。

图3.1中，右侧是Value Line系统框图。

该框图中左边的部位还是RISC架构的16位CPU。紧邻是JTAG调试窗口，左上角依次是时钟、闪存、RAM；最下面是通过ACLK和SMCLK做时钟源的数字外设和模拟外设。

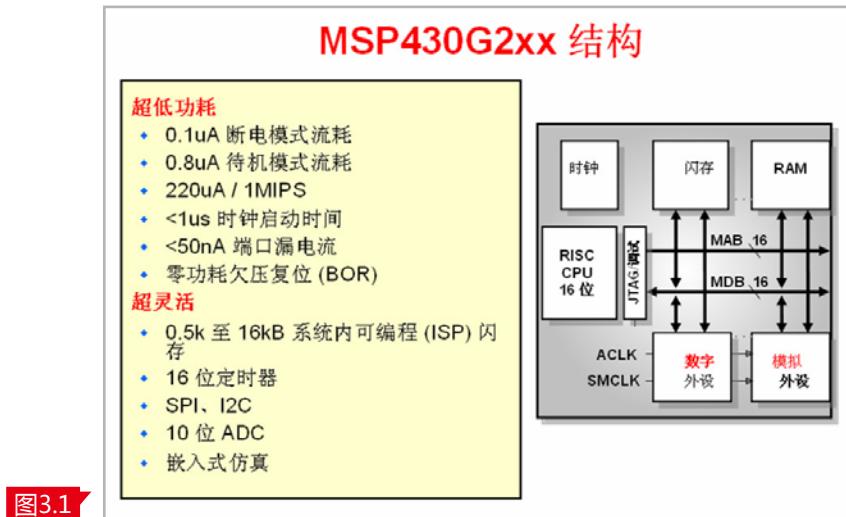


图3.1

G2XX架构

图3.2是G2XX系列16位RISC CPU的架构图。从改架构图中可以看到，它是单周期寻址寄存器文件，包含4个特殊功能寄存器。如常用的R0是PC指针；R1为堆栈指针；R2为状态寄存器；R3为常数发生器，R4到R15为12个通用寄存器。

从CPU的架构上来看，16位RISC CPU没有51CPU的累加器瓶颈，可以让运算速度和执行速度更快！同样由于是RISC架构的原因，所有指令支持正交架构。这种现代的CPU架构只需要27条的内核核心指令和24条仿真指令，从汇编的角度而言只有51条指令，且支持7种寻址模式。另外，G2XX可以支持Atomic内存至内存的寻址，也就是无缝的内存寻址方式。

G2XX整个CPU支持字、字节和字的处理，其中常数发生器也是MSP430的一大特色，比如我们在控制里面经常用到-1、0、8、4、2、1几个常数，在编译器产生过程中，可以不需要嵌入flash当中，大大节省flash的空间。

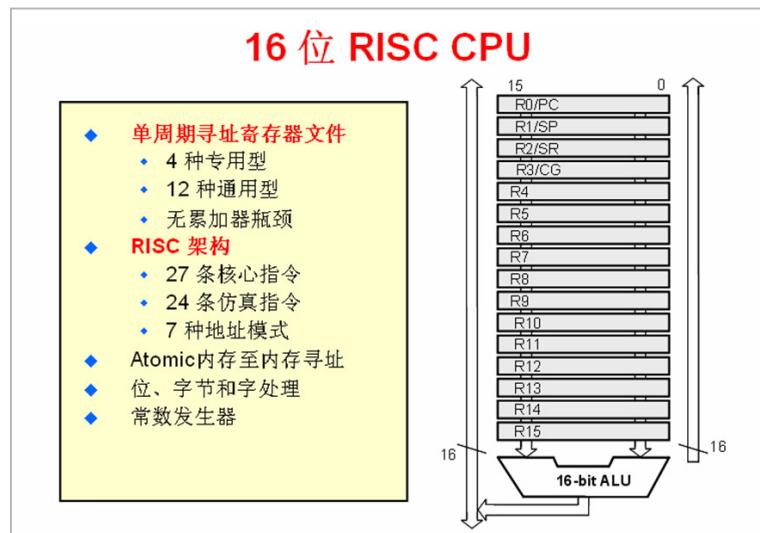


图3.2

MSP430存储器映射

图3.3是以G2231为例，0h到0Fh是微控制器8位特殊功能寄存器空间；010h到0FFh是8位外设和16位外设空间；从0200h到027Fh是RAM空间；01000h到010FFh为信息存储器的存储空间；F800h到FFDFh为存储的单元；从0FFC0h到0FFFFh是中断矢量表，单片机复位也是从0FFFEh开始执行，自动的进入到Reset状态。整个空间完全基于flash，可以通过JTAG或者系统编程（ISP编程）来实现。

需要特别强调的是信息存储器空间，该空间一般而言会分为A、B、C、D 4个区域，组成信息存储器，通常Section A会包含一些专用校准数据，且这个数据可以通过lock位锁住，这是个非常有价值的数据，在应用过程中要特别注意对该数据的保护。曾经用户因为在生产过程中没有刻意的去保护数据，而用到了这些数据，当生产过程中数据被误擦除之后出现了很大不良的问题，也是对数据保护没有做到足够的重视。

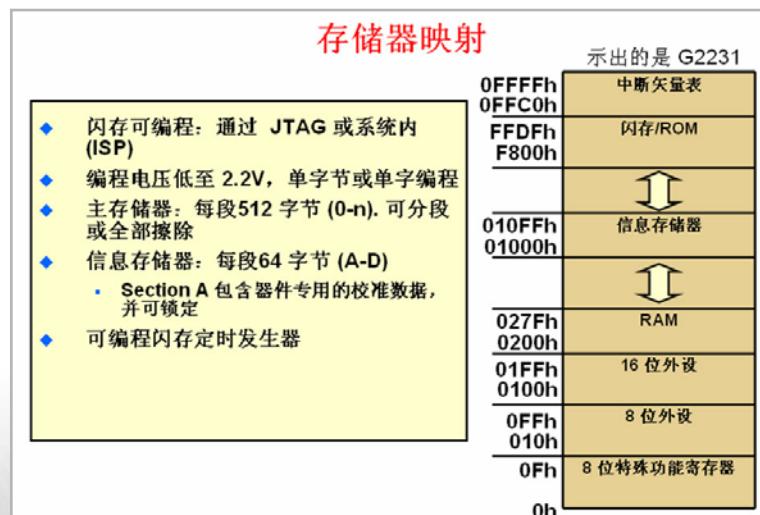


图3.3

G2231的编程电压从原来的1系列、4系列的2.7V降到2.2V。0.5V的电压变化在实际应用中大大节省了能源,让电池的寿命更加长久。我们通常意义上的电池的寿命是0.9V-1.5V,这一改善能够有效的通过缩短低电压编程空间,来延长电池的使用寿命。

另外,G2231的主程序存储器分为若干个段,即segment,每个segment为512字节。例如一个4K的flash通常来讲是由8个segment,即从segment0到segment7来组成主程序存储器。

MSP430时钟系统

图3.4描绘了MSP430的时钟系统。在框图中,红色的部分我们通常叫做“源”,也就是作为时钟输入的部分,其中包括内置的16M的DCO(Digital Controlled Oscillator,数字控制振荡器),以及VLO(Very Low Frequency Oscillator,低频振荡器)。同时,在外接晶体,如低频晶体LFXT1,通常外接晶体含有故障保护功能及脉冲滤波器,可以减少外部震荡过程中产生的一些错误。

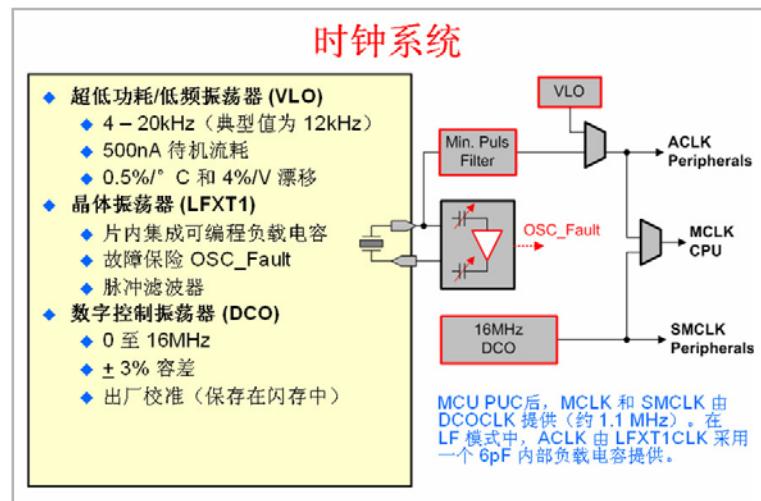


图3.4

“源”作为时钟的输入部分,系统中有三个时钟的作为输出,我们通常叫做ACLK、MCLK和SMCLK。ACLK是Auxiliary clock即辅助系统时钟的简称;MCLK是Main clock,也就是系统CPU运行的主要时钟选择;SMCLK是Sub-main clock,也就是次系统时钟。如何来区分这三种时钟?ACLK通常针对低速外设,通常而言,它的时钟频率比较低,例如系统待机经常工作在ACLK模式下;MCLK是CPU运行的时钟;针对高速外设,通常选择SMCLK来提供。

DCO校正

G2231的数字振荡器(DCO)性能极佳,能够满足0-16MHz的工作空间,且内部容差只有±3%。另外,前文提到的在information memory存放了一些出厂校验好的数据,也能够通过DCO进行校正。

采用数字振荡器的好处是：

第一，能够快速实现从待机到全速运行的快速唤醒功能；

第二，可以大大节省外部所用晶体的成本，来满足简化系统成本的设计。如前文所述存储于information memory中的校验数据，放在内部的存储单元中，减少外部校验晶体。根据具体所选用器件的Datasheet或者数据手册可以查看到，典型器件内存储了1M、8M、12M、16M的数据，通过分频的方式可以很容易的选择所需要频率的数据。

图3.5中间的这段源代码是在进行编程过程中所遇到的，需要根据选择的频率把相应数据存入相应位置。比如说希望用到1M的数据，只需要把calibration 1M的数据放到basic control 1(BCSCTL1)，也就是basic clock system control1寄存器和DCO control(DCOCTL)寄存器中，就很简单地完成了主频1M的设定。这里特别要强调的是，我们的G2xx1系列在我们的calibrate数据里面只存放了1M的数据。在G2xx2和G2xx3里面具有4个DCO校正数据可供大家使用。

G2xxx –DCO校正

DCO Calibration Data (provided from factory in flash info memory segment A)			
DCO Frequency	Calibration Register	Size	Address
1 MHz	CALBC1_1MHz	byte	010FFh
	CALDCO_1MHz	byte	010F Eh
8 MHz	CALBC1_8MHz	byte	010FDh
	CALDCO_8MHz	byte	010FCCh
12 MHz	CALBC1_12MHz	byte	010FBh
	CALDCO_12MHz	byte	010FAh
16 MHz	CALBC1_16MHz	byte	010F9h
	CALDCO_16MHz	byte	010F8h

```

// Setting the DCO to 1MHz
if (CALBC1_1MHz == 0xFF || CALDCO_1MHz == 0xFF)
    while(1); // Erased calibration data? Trap!
BCSCTL1 = CALBC1_1MHz; // Set range
DCOCTL = CALDCO_1MHz; // Set DCO step + modulation

```

◆ G2xx1 器件只具有1MHz DCO 校正参数。若需要较高的频率，客户必须自行校准。
 ◆ G2xx2 和 G2xx3 具有所有 4 个DCO校正参数校准值。

图3.5

VLO校准

MSP430 G2XX系列产品的VLO的数据为4kHz~20kHz，这是一个较大的范围，典型值是12kHz。由于VLO的正负偏差相对较大，如何用好VLO是一个比较困难的问题。在这样情况下，通常可以采用DCO进行校准。从图3.6中可以看到，首先是DCO是校准过的数据作为时钟源，然后去捕捉VLO也就是在运行期间可对VLO进行校准，采用校准的1MHz DCO为Timer-A提供时钟，利用VLO提供的ACLK/8，捕获其上升沿，通过运算就很容易得到相对准确的VLO的数据。

需要注意的事，不同的芯片VLO的数值可能是不一样的，通过校准以后的VLO可以保证相对准确，可以满足不需要任何外部晶振实现低功耗的设计。这一部分的设计TI官网中有相应代码，文件的编号是(SLAA340)，可以参考范例的VLO校准是如何实现的。

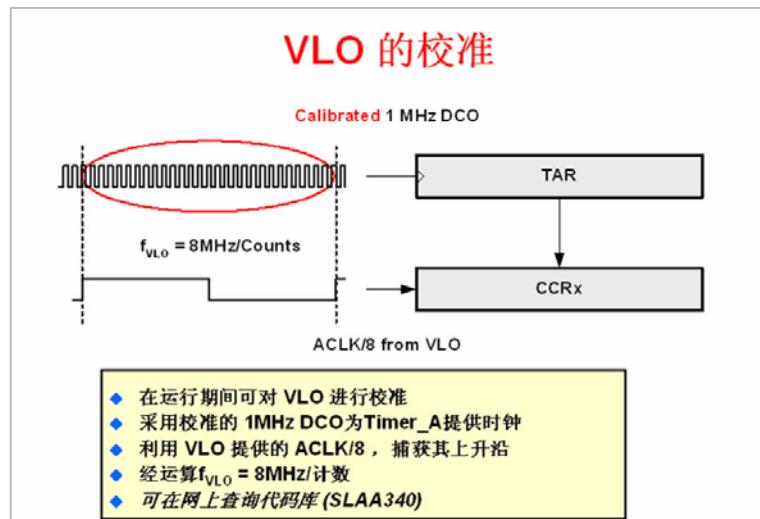


图3.6

系统MCLK和Vcc的关系

开发者在进行开发时要注意的MCLK和Vcc要有合适的比例关系。从图3.7中可以看到,纵轴是系统的频率,也是系统工作频率,从6MHz到16MHz;横轴是工作电压,从1.8V到3.6V。

当系统工作到1.8V的情况下,我们可以工作在6MHz。但是,假设希望系统工作在12MHz,却只提供系统1.8V电压的情况下,是无法保证系统稳定而可靠的工作。所以,在进行开发过程中时钟速度必须与Vcc保证准确匹配。

对于G2XX而言,图3.7中灰色阴影部分的工作电压是1.8V到2.2V,这是芯片的flash编程电压所不能做到的,但只要让整个系统工作在3.3V的条件下,就能够满足最高工作频率16MHz的要求。

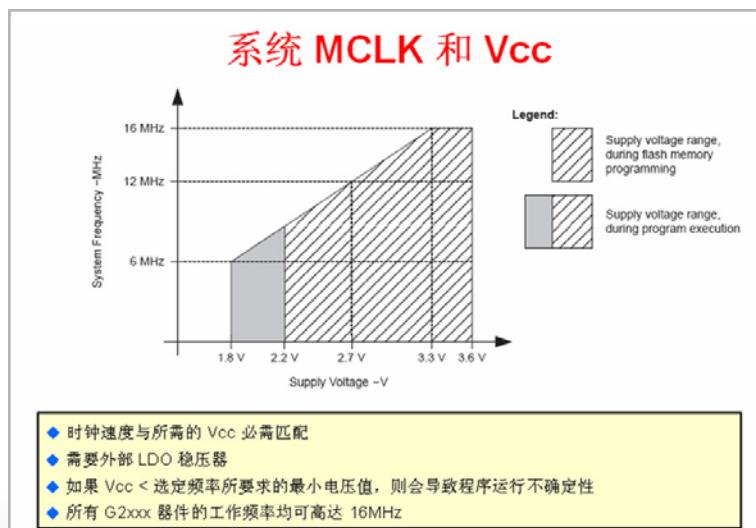


图3.7

2.5 中断与GPIO口

中断和堆栈

如何来进入一个中断服务程序？

在当前执行的指令已经完成时，指向下一条指令的PC指针会推送至堆栈上，状态寄存器SR(state register)也会被推送至堆栈上，此次程序会选择具有最高优先级的中断。同时，中断请求标志为单源中断标志位的会自动被清除为零，如多源中断标志的则会保持于设定状态，由进入中断服务程序的软件来控制。同时，MSP430是靠低功耗模式来进行编程的，进入到中断服务程序一般都是在低功耗模式下。进入到中断服务程序以后，会自动终止任何低功耗的模式，也就是SR会被清零。

另外，除SR会被清零之外，GIE位也会被清零，GIE是Global Interrupt Enabled。因此，在中断服务程序里面也会禁止执行更多的中断。同时，中断向量的内容将会被装入PC指针，程序将利用位于该地址的中断服务程序例程继续执行中断服务程序。

从图4.1中可以看到在中断前堆栈的模式，以及中断之后PC和状态寄存器被送到栈顶。

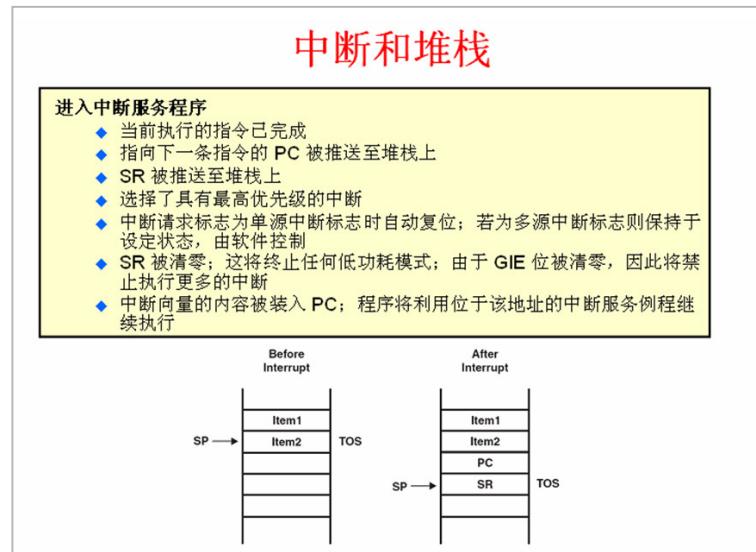


图4.1

中断向量

图4.2为G2231的中断向量表。从中我们可以看到优先级和中断源，包括各种中断标志位和各种系统中断以及相应的地址。

通常对单片机系统的安全性而言，如Power-up，外部的Reset、看门狗、定时器、以及Flash，出错的优先级是最高的。相应的中断源也有中断标志位，例如Power-up的PORIFG、Reset有RSTIFG，看门狗有WDTIFG等等。

一些外设中断如看门狗中断、定时器A2中断、ADC10中断、USI中断、I/O口中断等，其中断优先级相对较低。

同时，我们也要注意到该表当中，不同的中断有不同中断源的选择。如对于P1口而言有P1.0到P1.7，通常是一个单源中断，在进入中断服务程序后，通过软件来控制实现不同Port引脚或者是Pin脚数中断的判识。

Vector Table – G2231				
Interrupt Source	Interrupt Flag	System Interrupt	Word Address	Priority
Power-up External Reset Watchdog Timer+ Flash key violation PC out-of-range	P0UFG RSTIFG WDTIFG KEYV	Reset	0FFF Eh	31 (highest)
IMI Oscillator Fault Flash memory access violation	IMIFG OFIFG ACCVIFG	Non-maskable Non-maskable Non-maskable	0FFF Ch	30
			0FFFA h	29
			0FFF8 h	28
			0FFF6 h	27
Watchdog Timer+	WDTIFG	maskable	0FFF4 h	26
Timer_A2	TACCR0 CCIFG	maskable	0FFF2 h	25
Timer_A2	TACCR1 CCIFG TAIFG	maskable	0FFF0 h	24
			0FFE Eh	23
			0FFEC h	22
ADC10	ADC10IFG	maskable	0FFE Ah	21
USI	USIIFG USISTIFG	maskable	0FFE8 h	20
I/O Port P2 (2)	P2IFG.6 P2IFG.7	maskable	0FFE6 h	19
I/O Port P1 (8)	P1IFG.0 to P1IFG.7	maskable	0FFE4 h	18
			0FFE2 h	17
			0FFE0 h	16
Unused			0FFDE h to 0FFCD h	15 - 0

图4.2

中断处理函数编程

我们以一个实际案例进行讲解，用看门狗定时器做一个按键扫描消抖的一个中断服务程序，以此来来介绍中断处理函数编程。

图4.3是一个典型的看门狗中断服务程序编程，一个#pragma vector用来标识我们所列矢量的中断服务程序，同时-_interrupt void用来识别中断服务程序的名称。

一般情况下，中断服务程序不需要特别的返回值，而图中的源代码需要特别强调的是，一般情况下进入到一个中断服务程序当中，前一两句程序通常是disable应该中断，然后清除到该中断的标志位，从而来进行中断以后的各种编程。

中断处理函数编程

```
#pragma vector=WDT_VECTOR
_interrupt void WDT_ISR(void)
{
    IE1 &= ~WDTIE;           // disable interrupt
    IFG1 &= ~WDTIFG;         // clear interrupt flag
    WDTCTL = WDTPW + WDTHOLD; // put WDT back in hold state
    BUTTON_IE |= BUTTON;     // Debouncing complete
}
```

#pragma vector — 下面的函数是一个用于所列矢量的 ISR
 _interrupt void — 识别 ISR 名称
 无特别需要的返回值

图4.3

GPIO口的设置与应用

仍然以MSP430G2231为例，G2231是一个14Pin引脚的单片机，第1Pin脚是DVCC，第14Pin脚是DVSS，第12、13Pin是XIN和XOUT，一般情况下，第10Pin脚和第11Pin脚是TEST和RST引脚。

特别强调的是第10Pin脚和第11Pin脚是SBWTCK和SBWTDIO。通过第1Pin脚、第14Pin脚、第10Pin脚、第11Pin脚，可以很轻松的建立起Spy-bi-wire接口，也就是说用4根导线就能轻松的实现430的仿真和调试。

图4.4展示了G2231的GPIO的寄存器。每一个GPIO的寄存器通常包含很多种分类，例如：输入寄存器包括了PxIN，比如P1IN和P2IN，P1IN包含从P1.0到P1.7，P2IN包含从P2.0到P2.1；以及出寄存器、方向寄存器等。

如何来设定一个I/O口为输入或者为输出？可以通过PxDIR也就是direction来设定输入输出。如前文介绍过G2231的I/O口具有上下拉电阻，通过PxREN可以选择上拉或者下拉电阻。另外，G2231的I/O口具有多种功能复用功能，可以通过PxSEL或者PxSEL2来选择不同功能的实现。

GPIO的中断，如果需要通过GPIO口来实现中断，可以设置中断寄存器来实现。G2231的一个GPIO口有多种中断寄存器需要设置，例如中断边缘，就是选择上升沿和下降沿，中断使能PxIE，就是Interrupt Enable，中断标志位PxIFG，就是Interrupt Flag。

GPIO的编程，以图4.4的两段例子来进行简单说明。例程中，P1DIR=BIT4，实际上是将P1DIR设1.4为高，若P1SEL是写入的唯一输入，那实际上就是从P1.4的功能上选择SMCLK的功能。P1.0为输出把P1 OUT 1.0，即将P1.0设置为输出为高，就可以看到一个从低到高的过程，图中的示例就是选择系统SMCLK输出的一个例子，非常简单易用。

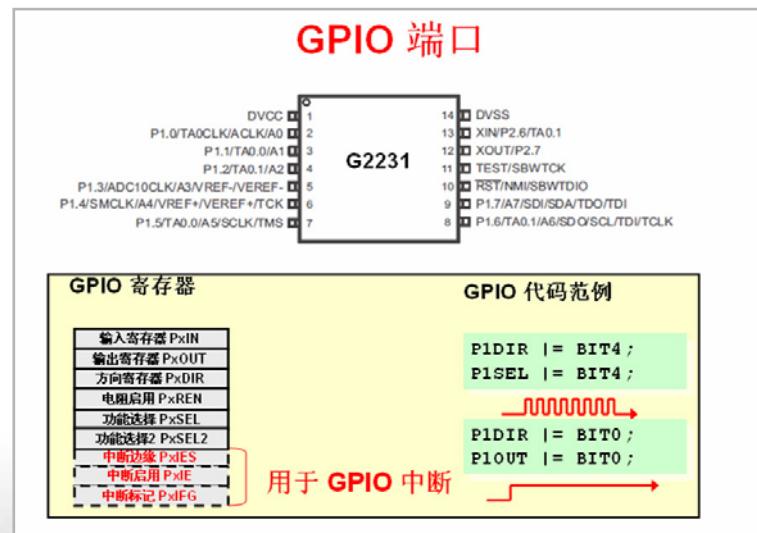


图4.4

引脚复用

G2231的引脚复用的选择和实现，以图4.4中P1.4为例，P1.4的引脚从Pin内部可以看到具有多种功能，从图表最左边来看，既可以实现P1.4GPIO口的功能；也可以实现SMCLK即子系统时钟输出功能，包括TA0.2也就是Timer-A的输入功能；还可以作为VREF₊也就是ADC的参考源；该引脚也可以作为ADC的输入；还可以做比较器的输入、JTAG TCK以及可以做Cap touch I/O。如此多的功能在选择上就需要新型复用选择。从图表中可以看到实现该功能的复用选择方法。例如说要实现SMCLK的功能，只要设置P1DIR为1，相应的P1.4为1，那P1SEL为1，就选择了SMCLK的功能。

如果需要选择A4的功能，甚至都可以不去关注P1DIR、P1SEL、P1SEL2等设置，只需要在ADC相应的寄存器设置ADC10 AE，也就是ADC10的Analog enable就可以选择P1.4引脚模拟量输入的功能作为ADC的输入。图4.5通过一个简单的表格来举例，将G2231的引脚复用功能是如何选择实现的进行了列举，更详细的不同430的器件及选择，需要查看相应的数据手册。

引脚复用

Table 16. Port P1 (P1.4) Pin Functions

PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS / SIGNALS ⁽¹⁾					
			P1DIR.x	P1SEL.x	P1SEL2.x	ADC10AE.x (INCH.x=1) ⁽²⁾	JTAG Mode	CAPD.y
P1.4/ SMCLK/ TA0.2/ VREF ₊ ⁽²⁾ / VEREF ₊ ⁽²⁾ / A4 ⁽²⁾ / CA4/ TCK/ Pin Osc	4	P1.x (I/O)	I: 0, O: 1	0	0	0	0	0
		SMCLK	1	1	0	0	0	0
		TA0.2	1	1	1	0	0	0
		TA0.CC02A	0	1	1	0	0	0
		VREF ₊	X	X	X	1	0	0
		VEREF ₊	X	X	X	1	0	0
		A4	X	X	X	1 (y = 4)	0	0
		CA4	X	X	X	0	0	1 (y = 4)
		TCK	X	X	X	0	1	0
		Capacitive sensing	X	0	1	0	0	0

- ◆ 每个引脚具有多项功能
- ◆ 在对应的寄存器选择相应的引脚功能
- ◆ 具体详见各器件的数据手册

图4.5

2.6 定时器A与增强型看门狗模块

Timer_A

所有430器件都有一个Timer_A模块，该模块有一个非常强大的定时计数器模块，它可以选择不同的时钟源，如TACLK、ACLK、SMCLK，也可以选择外置的INCLK。该模块内置的16位定时器是一个异步16位定时器/计数器，支持多种工作模式。

图5.1中可以看到它有Count Mode，支持连续、递增-递减、递增计数器的模式。Timer_A模块可以有多个捕获/比较模块，如CCR0、CCR1、CCR2。在图5.1中，我们以Timer_A3为例，有CCR0到CCR2三个模块，而Value Line产品只有两个模块，即CCR0、CCR1。该定时器有多个功能，它可以实现多个捕获/比较寄存器的功能，可以实现自动的PWM输出，同时其中断向量寄存器可以用于实现中断快速响应，该定时器也可以用于触发DMA传输。

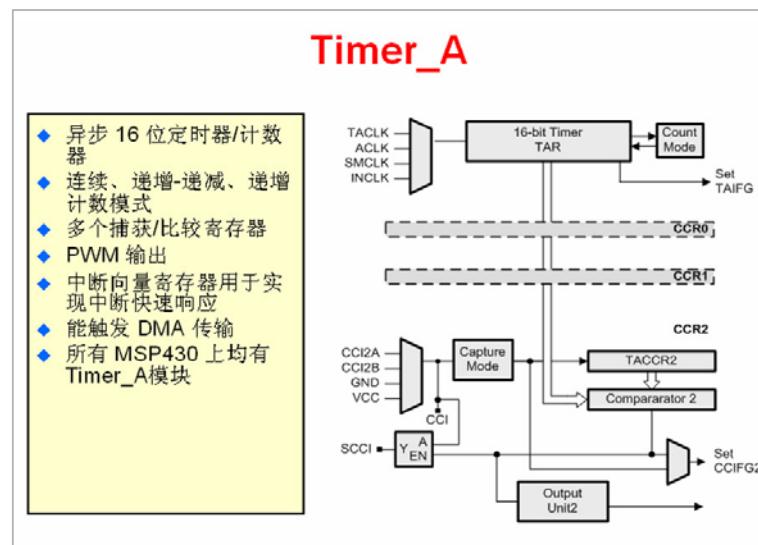


图5.1

Timer_A计数器

图5.2展示出Timer_A的计数模式。第一种是定时器停止和暂停的模式，第二种是连续计数的模式，第三种是递增模式，第四种是递增/递减模式。

从连续奇数模式中可以看到从0到0FFFFh，就是连续计数的模式，从0开始累加到65535，然后再回到0，再重新计数到65535完成一个连续的模式；递增模式中，我们可以设定一个值，从0计数到我们的设定值，比如CCR0，然后再回到0，再计数到我们的设定值，形成一个递增的模式，递增模式常用于PWM自动输出功能情况下；对于递增-递减定时器则会在0-CCR0的值再回到0的模式下，来实现一个递增/递减的计数模式。

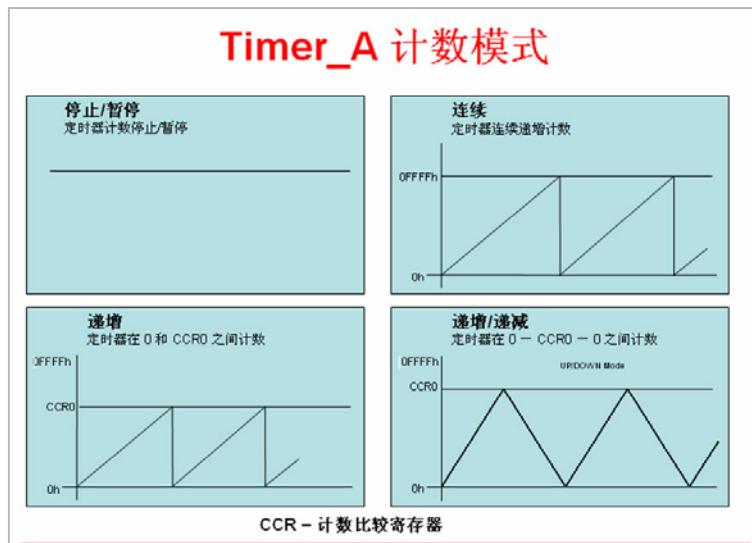


图5.2

Timer_A中断

Timer_A作为互补型的PWM的输出模式是比较常用的方式，对于Timer_A中断而言是一个相对复杂的中断判识过程。首先，CCR0有一个单独的中断向量，TACCR0 CCIFG，在整个Timer_A定时器A中是一个相当高的中断优先级，而且它不需要额外的处理程序就可以单独地响应中断和执行中断。

CCR1和CCR2对于Timer_A的中断标的进行处理时，需要对Timer_A中断进行优先级处理，并采用Timer_A中断向量寄存器(TAIV)将之组合成一个中断向量，这是一个多源中断。

图5.3中可以看出TAIV Timer_A中断向量表包含了TACCR1的中断向量、TACCR2的中断以及溢出中断，同时组成了一个TIMERA1的中断向量。由于中断向量的多样性，因此在中断服务程序中必须包含一个处理程序，以确定触发的是哪个TIMERA1的中断。对于TIMERA有TIMERA0_VECTOR和TIMERA1_VECTOR，对于TIMERA1_VECTOR，中断服务程序必须有相应的处理程序来判识中断源的发生。

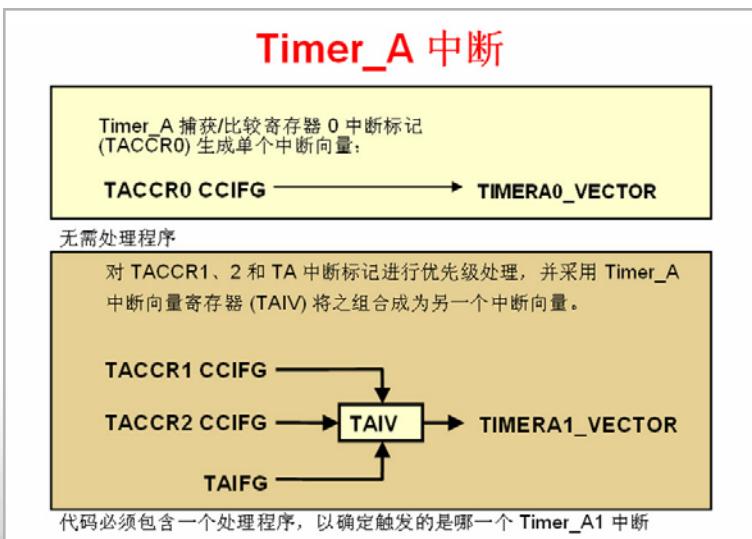


图5.3

图5.4是Timer_A中断向量处理程序的实例。

可以看到TAIV寄存器位打着4个X，这是用来标识产生CCR1、CCR2或者溢出中断相应的标志位。从右上角可以看到在产生TAIV的时候，产生02h的Offset偏移量时，是TACCR1的中断向量；产生04h的Offset偏移量时，是TACCR2的中断向量，产生0Ah，也就是通常说的10时，是TAIFG的中断向量。左下角是常用的C代码程序，其本征函数`_even_in_range(TAIV, 10)`，通过2、4、10判识，来表明不同中断产生的判识。

图5.4右侧是汇编程序代码，其方式也非常简单，可通过产生相应的中断后在内部进行比较和判识，输出到相应的中断服务程序中。汇编程序代码和C代码的对应性是非常强，从这个角度可以看到430的编程C语言和汇编语言都具有非常高的效率。

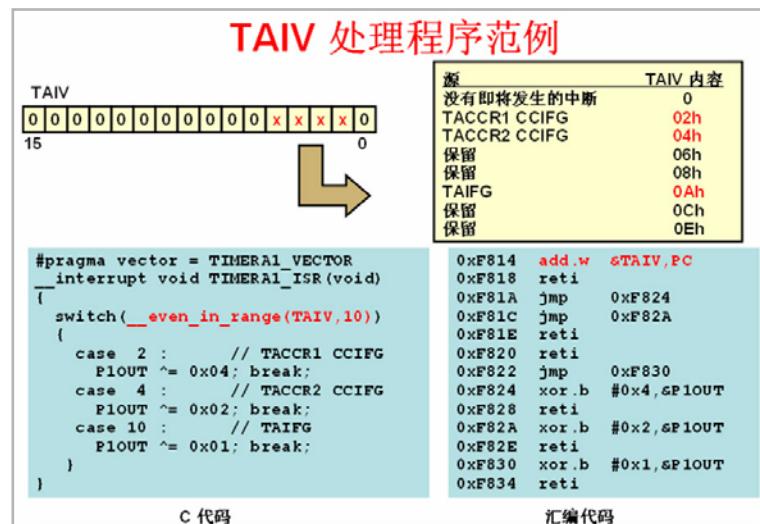


图5.4

Timer_A PWM范例

Timer_A能够自动产生PWM的功能，当CCR0置成UP模式，即指定的一个递增模式，即是设定完成一个PWM的周期寄存器，同时设定CCR1和CCR2相应的值，从而产生不同占空比的PWM，这样就自动完成了不需要CPU干预的PWM的生成。

图5.5是使用MSP430F11x1一个芯片的引脚来完成的实例，在430系列中，无论是哪一颗芯片的PWM都是自动发生的，如果想找到更多的关于PWM的例程请查阅430的相关网址，找到相应的代码范例。

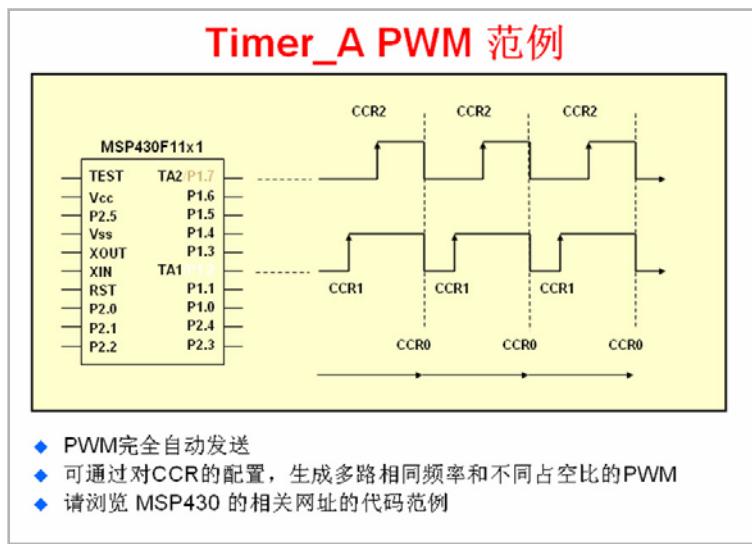


图5.5

采用Timer_A的直接硬件控制功能

在MSP430中，当使用Timer_A触发ADC时，不需要CPU干预ADC，它可以通过外部模拟信号实现一个连续自动的采样。例如，我们需要产生一个稳定的时间，参考源是一个17毫秒的稳定时间，那只用在黄色的阴影时间内去设定一个TACCR1，这样17毫秒就自动产生。其中黄色部分是初始化所设定的时间，也就是我们需要CPU干预的时间，这一部分非常小。

当参考源足够稳定的时候，芯片可以启动AD转换，此时触发ADC、处理ADC结果，并让ADC的基准关断，我们同样可以看到在转换过程中，CPU的运作时间也是非常短的，在两秒的时间周期内，用Timer_A来控制ADC，可以让CPU在非常短的时间内完成一个周期的转换，大部分时间CPU都在休眠状态，可以用Timer_A很好地实现直接硬件控制，来满足对低功耗的要求。

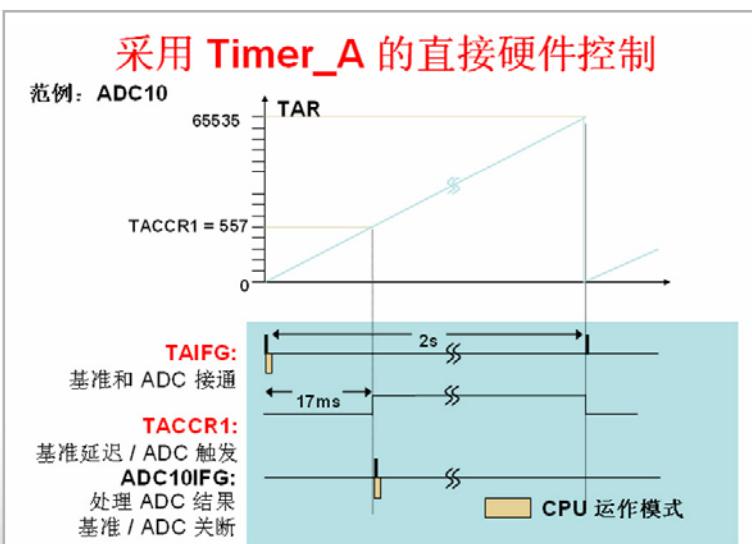


图5.6

增强型看门狗模块

所有430器件上均有WDT看门狗。分为两种，一种是普通的WDT，另外一种是增强型看门狗。

对于看门狗而言，需要在程序发生异常的时候去喂狗，然后让单片机回到复位状态。因此，看门狗模块可以用做时间间隔寄存器，由于该模块的重要性，它的访问需受到密码保护。

看门狗模式和定时器模式对于该模块有不同的单独的中断向量，可以满足不同需要的应用要求。另外，考虑到时钟源对时间间隔和看门狗应用的不同，时钟源可以由ACLK和SMCLK来单独设定，同时还可以控制RST/NMI也就是非屏蔽中断的引脚模式。

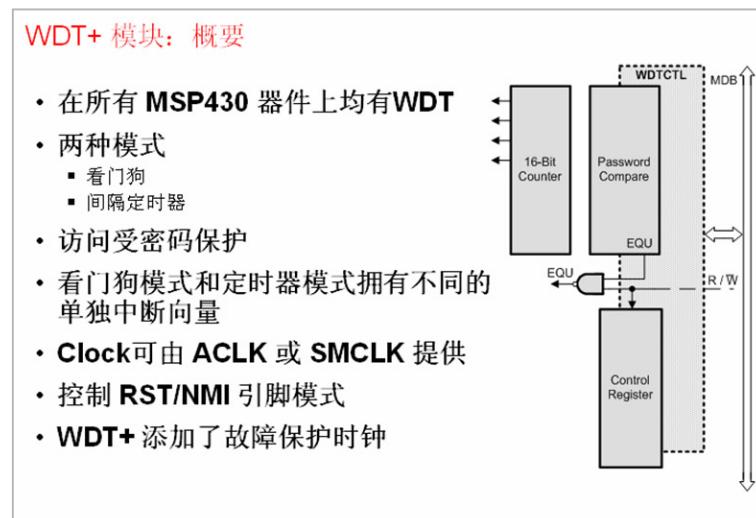
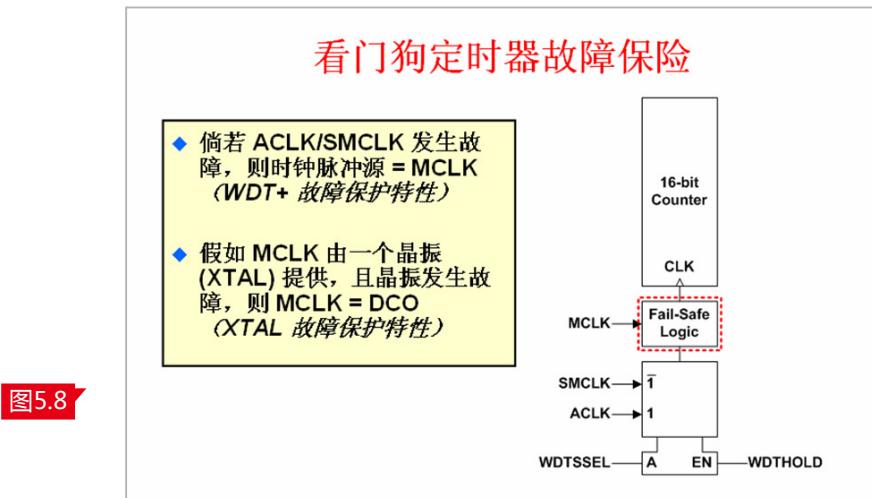


图5.7

增强型看门狗模块增加了各种保护时钟功能，可以实现时钟的自动保护，具有时故障保护功能。例如，若ACLK和SMCLK发生故障的时候，时钟源会自动切换到MCLK，这样就实现了WDT增强型看门狗故障保护的特性。

故障保护特性举例而言，假如MCLK由一个晶振外置的crystal提供，且晶振发生故障，看门狗的时钟MCLK也会自动的切换到DCO，从而实现对外部crystal故障保护的特性。这是一个非常好的Fail-Safe的逻辑模式，在代码例程库里经常可以找到相应的代码例程来查看看门狗是如何实现这种Fail-Safe的逻辑保护功能的。

图5.8画出了看门狗定故障保护的运作方式。



对于看门狗设计过程中常用的一些问题，以及一些解决办法，通常有如下问题。

通常程序能够对其自身进行复位，这是看门狗最常见的功能。如果程序动作反常，可以判断执行是否达到清除WDT的位置，可以在靠近main函数的起点来设置一个中断，来察看代码是否重新启动。

还有一些应用，CPU甚至在到达第一条指令之前似乎就出现了冻结现象。举例而言，我们用一个相对全局变量非常大的一个函数在调用过程中，似乎我们很难实现真正的调试。因此开发者需要判断，C程序是否是具有大量的初始化数据，就像一些全局变量要通过JTAG调用到CPU内部。这种故障通常只会在拥有非常大的RAM空间器件上出现，解决办法也很简单，在TI的函数库里提供了_low_level_init()函数，在这个函数中可以关闭看门狗来完成这个功能。

图5.9的源代码就是在通常调试过程中，如何去停掉看门狗。在WDTCTL寄存器送入WDTPW+WDTHOLD功能，就实现了停掉看门狗的功能。

WDT：常见设计问题

- 程序保持对其自身的复位
- 程序动作反常 – 执行是否到达清除WDT的位置？
 - 设置一个靠近 main() 起点的中断，以查看代码是否重新启动
- CPU 甚至在到达第一条指令之前似乎就出现了冻结现象
 - C 程序是不是具有大量初始化数据？
 - 通常只会在拥有超大RAM空间器件上出现
 - 解决方案：在 __low_level_init() 函数中关闭看门狗

```

void main(void)
{
    WDTCTL = WDTPW+WDTHOLD;           // Stop the dog
    .
}
```

图5.9

如前文所述，WDTCTL可用做时间间隔定时寄存器。图5.10中可以看到我们画了一个标准的闹钟，如果采用WDT做时间间隔寄存器的话，就像闹钟一样相比于以前的沙漏计时而言是非常精准的。

当定时产生时，看门狗相对而言是没有PUC(Power Up Clear)产生的，如果WDTIE和GIE在到达间隔时被设定，则自动生成一个WDT间隔中断，这个时候强调的不是复位中断，而是一个时间间隔的中断。

定位的时间和定时时间的间隔可以通过编程的方式来选择，选择时钟源的不同，间隔时间的长短可以任意去调整。

WDT：间隔定时器功能

- 当定时到达时没有 PUC 产生
- 如果 WDTIE 和 GIE 在到达间隔时被设定，则生成一个 WDT 间隔中断（而不是复位中断）
- 定时间隔可编程选择



图5.10



2.7 MSP430低功耗设计的实践基础

MSP430可以说是为低功耗而生，血脉中流淌的是低功耗的特性。在本章正式介绍之前，有一个MSP430和苹果不得不说的故事。

这是个基于MSP430F413设计的开发板，左下角包含一颗TITMP100的温度传感芯片，以及一个段式的LCD Module，通常会用一颗纽扣电池供电。图6.1我们要讲述的故事是，拿掉纽扣电池之后，把一个苹果切成三瓣，每瓣苹果插上铜芯用导线串联起来，代替纽扣电池实现对整个PCB板的供电。

实验进行的非常顺利，LCD的显示非常的准确、清楚。如果我们用万用表测量三瓣苹果能提供大约3V左右的电压，当然，它很不稳定但能够满足整个PCB板供电的要求。

如果这个时候把万用表串接在整个电源回路当中去，可以从万用表观察到整个PCB板所需要的功耗大约只有在2微安左右。

最后，图6.1还展示了MSP430的BOR功能。一个水果电池它的内阻远大于纽扣电池，带载能力比较差电源稳定性低，正是430片上的BOR功能能够保证整个系统可靠而稳定的工作。这也是MSP430低功耗要求的一个很好的例证。

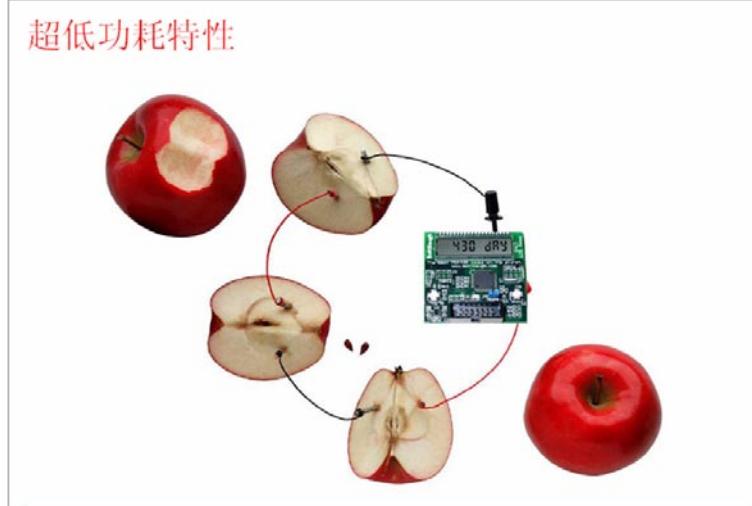


图6.1

超低功耗是430的DNA

MSP430从一开始就专为超低功耗而设计，并形成一个缩写ULP，ULP就是超低功耗(Ultra Low Power)。MSP430所有的外设设计是专为减少功耗和最大限度来降低CPU的开销或者占用而优化的。所有的外设都是智能型能够独立CPU而工作，在系统不需要该模块工作时，能够自动关闭且较长时间可以处于较低功耗的模式下。

TI网站www.ti.com/ulp详细的介绍了430所有低功耗的特点，其中包括可以实现多种的操作模式，如 $0.1\mu A$ 断电模式，RAM能够保持通常所说的LPM4的模式。

$0.3\mu A$ 的待机模式，通常称为实时时钟模式。在运行时，若程序在RAM运行，每MIPS为 $110\mu A$ ；用flash运行，每MIPS为 $220\mu A$ 。

作为低功耗系统设计，即时可稳定工作的高速时钟非常重要，英文常描述为insist high-speed clock。它能够小于1微秒的时钟，将系统从没有到完全稳定下来，这一点是设计低功耗系统非常重要的数据。满足于电池应用的1.8到3.6V单电源系统，是零功耗、始终工作的BOR功能，小于 $50nA$ 的引脚漏电流。所以，从执行运算速度的角度，它可以很大限度的减少每项任务执行的周期的CPU，就是具有较高运算速度的CPU。

低功耗智能外设能够自动传输数据，前面已经介绍了带有DTC功能的ADC模块，在我们的数据手册上你很难查到Timer_A的功耗，因为它的功耗微乎其微。模拟比较器的功耗大约在 $100nA$ 左右。综上，430可以保证在对低功耗要求所有下的性能的要求。



图6.2

超低功耗的工作模式

图6.3介绍超低功耗MSP430是如何工作的，它的工作模式是什么。

图6.3中红色部分的最底下有一条水平线，通常为待机模式，或者叫做超低功耗模式。上面像脉冲的图形是一个运行模式，整个430就工作在待机、运行这种间歇的模式下。所以它的平均功耗很低。可以看到，绿色水平线代表了平均功耗，要特别强调一下的是，MSP430一个小于1微秒的启动时间。

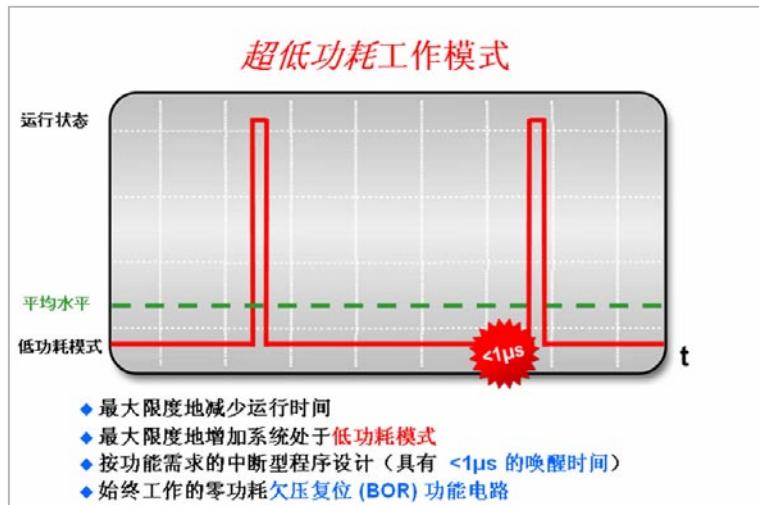


图6.3

如何来做低功耗？

如图6.4所示，首先，可以最大限度的减少运行时间，也就是说让脉冲时间越少系统功耗就越小。另外，要最大限度的增加系统处于低功耗模式的时间，同时按需中断程序设计。因为单片机具有小于1μs的唤醒时间，如何来理解这句话呢？我们设计一个系统，并使系统工作在待机模式，而需要去处理一些程序的情况下，迅速地唤醒系统，来实现程序的功能。举例来讲，烟雾探测器，我们看到它的指示灯在间隔一段时间闪烁一次，间隔一段时间闪烁一次，在闪烁的时间也就是说是烟雾探测器在工作的时间，在它不闪烁的时间，实际上整个系统都是在待机状态。如果从待机到工作时间越短，能够让它迅速的完成所要处理的数据任务，也可以让系统的平均功耗更低。同时，430芯片具有零功耗的BOR的功能电路，能够实现稳定的上电、掉电的保护功能。

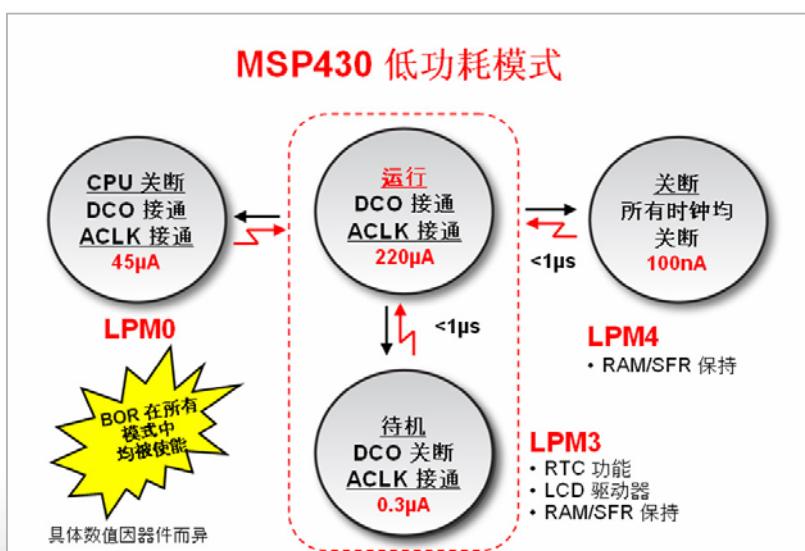


图6.4

低功耗模式配置

从图6.5可以看到,有LPM0、LPM3和LPM4,为什么没有LPM1和LPM2?这是因为相比于设计一个低功耗系统,LPM1和LPM2的功耗是在LPM0、LPM3、LPM4之间,是一个低功耗系统中很少用到的两种功耗模式,所以常用到的是LPM0、LPM3、LPM4以及Active模式。

LPM0的状态是CPU关断、DCO接通、ACLK接通,整个系统功耗大约在45 μ A;LPM3是待机状态,DCO关断、ACLK唤醒,大约整个芯片系统是0.3 μ A;LPM4所有全部关掉,只有RAM和特殊功能寄存器保持。

另外,我们的I/O口可以唤醒整个系统,整个芯片系统大概只有0.1 μ A。全速运行的时候,ACLK、MCLK、包括DCO全部运行,包括CPU,整个单片机系统的功耗是220 μ A左右。

BOR在所有的模式中均被使能。同时,各个模式之间可以通过状态寄存器互相进行切换,切换的时间是小于1微秒的。

来如何实现低功耗的配置?如何来实现各种模式间切换?如何来实现LPM0、LPM3、LPM4?

前文中CPU部分谈及了R2,CPU内部R2寄存器也是一个状态寄存器SR(state register),里面有SCG1、SCG0、OSCOFF和CPUOFF等相应的位,通过这种位的组合就可以实现不同低功耗模式的配置。比如运行模式4位大家可以看到4位全部是零,这时候系统功耗是250 μ A;LPM0需要CPU OFF置1,相当于把CPU关掉,那系统功耗是35 μ A;若LPM3只ACLK打开,也就是Oscillation OFF被关闭,所以整个系统功耗是0.8 μ A;LPM4相应的4位全部置高,整个单片机系统功耗大约0.1 μ A。

如何用汇编语言来实现呢?图6.5最下面一条语句,一句话就可以让整个单片机进入到一个低功耗模式里。bis.w,然后把CPUOFF写给SR寄存器,就可以让整个单片机进入到低功耗模式里。

低功耗模式配置									
保留	V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C
R2/SR									
运行模式	0	0	0	0		约 250 μ A			
LPM0	0	0	0	1		约 35 μ A			
LPM3	1	1	0	1		约 0.8 μ A			
LPM4	1	1	1	1		约 0.1 μ A			

```
bis.w #CPUOFF,SR ; LPM0
```

采用汇编程序的 LPM

图6.5

低功耗模式的堆栈操作

从图6.6右边整个源程序来看，这是一个典型的低功耗系统的实现。从一开始，设置中断向量，设置指针来做一个看门狗的时间间隔，进入到主循环，单片机进入到一个CPUOFF，也就是前面所说的LPM0的模式。最下面是中断向量表，以及相应的WDT_ISR中断服务程序。左边是堆栈指针。特别强调两个数值，一个是SR=0018，一个是SR=0008，“8”实际上是1000，也就是说global IE全局中断使能。

0和1的变化就是单片机从低功耗和active的模式，是低功耗模式0, 1是全速运行的模式。所以，从低功耗的模式来看，堆栈操作是相互关联的，是很容易的实现低功耗的操作，因此ULP低功耗简单易用。

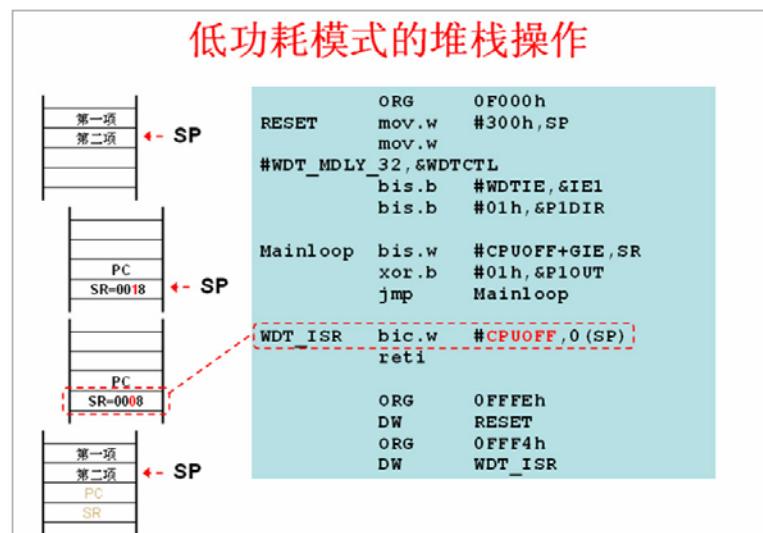


图6.7中，采用C语言来实现整个低功耗的模式，整个程序流程如下：配置main函数、WDT使能。整个过程当中，`_bis_SR_register`使整个系统进入到LPM3的模式，这时候启用中断，单片机进入休眠状态，等待中断产生。

考虑到PC指针下移，可以用`activeMode`让整个程序循环。中断服务程序如下：`WDT_VECTOR`，`_interrupt void watchdog_timer(void)`。在中断服务程序里面`bic_SR_register_on_exit(LPM3)`，实际上是从SR里清除LPM3位，退出LPM3，从而进入到运行模式。所以，从整个程序来看就是一个待机到全速运行、在中断服务程序里运行、退出LPM3、再回到待机这样一个反复的模式，从而让系统工作在待机、运行、待机、运行的模式。

ULP 简单易用！

使用我们的低功耗模式很容易

```

void main(void)
{
    WDT_init(); // initialize Watchdog Timer
    while(1)
    {
        __bis_SR_register(LPM3_bits + GIE); // 进入LPM3, 启用中断
        activeMode(); // 运行的代码
    }
}

#pragma vector=WDT_VECTOR
_interrupt void watchdog_timer (void)
{
    __bic_SR_register_on_exit(LPM3_bits); // 从0(SR)清除LPM3位, 退出LPM3, 进入运行模式
}

```

图6.7

430低功耗系统的实时时钟，用一颗纽扣电池可以让其工作10年。图6.8中以20x1开发的图例，外接一个32.768K的时钟，采用一个CR2032纽扣电池供电。其工作模式从左下角图中可以看到，整个基准是低功耗模式，脉冲模式是在进行秒增加、分钟增加和小时增加模式。

对于功耗的计算，从图6.8左上角可以看到LPM3，对应左下角的图是待机模式，功耗是 $0.8\mu\text{A}$ 。考虑到系统被激活，因为时间间隔很短，大概一秒钟只需要100微秒来计算，所以整体功耗大概在 $0.03\mu\text{A}$ ，系统的平均功耗是 $0.83\mu\text{A}$ 。考虑到纽扣电池2032，220毫安时的容量，可以保证超过10年的寿命。

可连续工作 10 年的嵌入式实时时钟

$$\begin{aligned}
 &= \text{LPM3} + \text{RTC_Function} \\
 &0.80\mu\text{A} + 250\mu\text{A} * \frac{100\mu\text{s}}{1000000\mu\text{s}} \\
 &0.80\mu\text{A} + 0.030\mu\text{A} = 0.83\mu\text{A}
 \end{aligned}$$

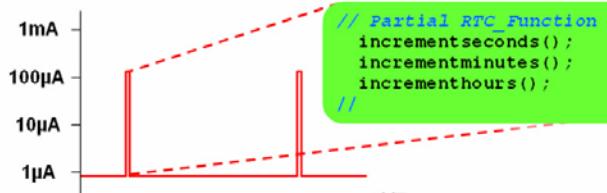
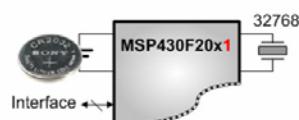


图6.8

低功耗操作

如何通过低功耗操作，让系统更加节能？对于低功耗有较高要求的产品，可以最大限度的减少瞬间吸收电流，也就是说让工作时间相对缩短，同时最大限度的增加系统处于低功耗

模式的时间。

由于430具有固有的低功耗特性，所以您的设计如何让整个系统更为优化，也对低功耗特性有很大的影响。也就是说，软件、硬件的设计对整个系统、产品的低功耗起到巨大作用。所以，正确的低功耗设计的方法，对整个系统的功耗的优化非常的重要。

图6.9可以看到待机模式中，Always-on也是一个一直打开的ACLK时钟系统，比如说用32768来做时钟或者用片上的VLO来做时钟。同时对于工作时间，或者我们叫做active模式的情况下，MCLK是通过DCO来提供的，它是一个On demand也就是按需打开模式。只有我需要计算的时候，才会打开这个时钟，而这个时钟能够做到即时稳定，即能够小于1微秒的时间来稳定，从而让系统整个功耗做到最低、最优化的状态。

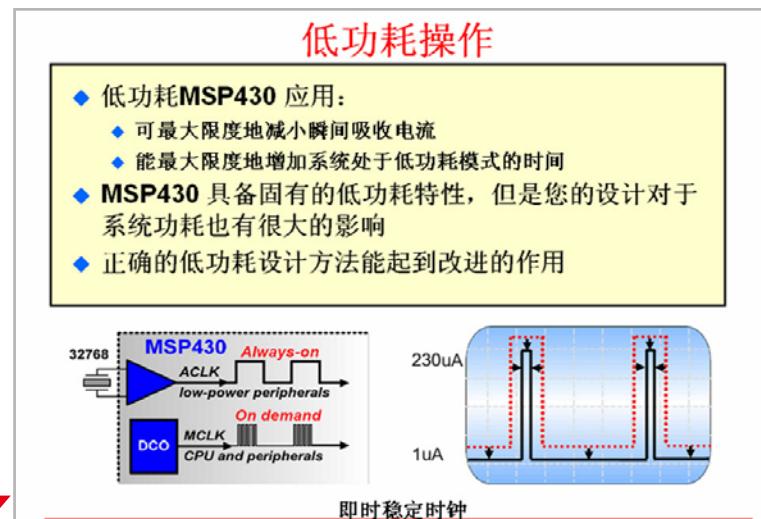


图6.9

如果在做系统设计过程中，把一些软件功能用外设实现，可以让系统的功耗更低。

前文曾有描述，430的外设是专为低功耗设计的，所以用430片上外设来做一些软件功能，可以使系统的功耗得到更好优化。

图6.10以一个P1.2来输出一个脉冲来做举例。可以看到假如我们用active的模式，让CPU去切换一个I/O口的变化，此时CPU负荷为100%。如果使用器件内的定时器和比较器，则会自动输出一个脉冲，代码CCTL1=OUTMOD0_1中，是设定一个脉冲自动输出，然后通过_BIS_SR(CPUOFF)进入低功耗模式，整个CPU不需要唤醒，可以保持睡眠状态，所以CPU的功耗非常低，因此系统功耗也非常低。

将软件功能移至外设来完成

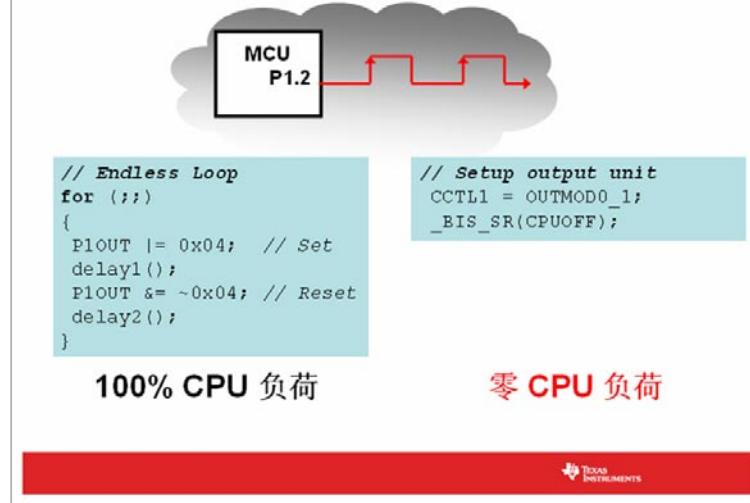


图6.10

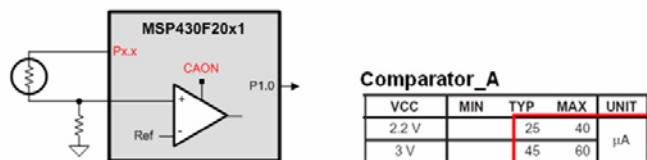


针对片上外设，也有方法能够让系统的功耗更低。

从图6.11的右侧可以看到，比较器在工作过程中，在2.2V的条件下是25 μ A的功耗，在3V条件下是45 μ A的功耗，但比较器并不是实时去做一些功能，那如何做？很简单，在做一些运算的过程中，可以利用PORT口，把比较器的正输入端接到某个PORT口，当用比较器的时候，通过PORT口来进行充放电，来实现电阻值的测量，来满足整个低功耗的设计，这样就可以做到按需启动比较器。

在图6.11所示的程序中也可以看到，设置P1OUT=0x02，实际上是给这个传感器加电来启动整个测量，最后用P1OUT清零。同时，外加传感器是一个掉电的过程，所以让整个系统功耗能够做到更低。

片上模拟外设的电源管理



```
P1OUT |= 0x02; // Power divider
CACTL1 = CARSEL + CAREF_2 + CAON; // Comp_A on
if (CAOUT & CACTL2)
    P1OUT |= 0x01; // Fault
else
    P1OUT &= ~0x01;
    P1OUT &= ~0x02; // de-power divider
    CACTL1 = 0; // Disable Comp_A
```

图6.11

外部设备的电源管理

我们都知道，没有外设或者是不需要供电的外设才会是功耗最低的系统，所以需要一个外部设备的电源管理方式。图6.12右侧的例子中，选择一个低功耗的运放，其静态模式和运行模式只需要 $1\mu A$ ，考虑到它的供电跟MCU一样都是由一个 $3.3V$ 来供电，那整个运放所产生的功耗大约在 $1\mu A$ 。但是如果使用I/O口管理外部器件的电源供应，可以让系统的平均功耗更低。

例如选择一个具有外部shutdown功能的运放TLV 2760，当需要对外部传感进行测量的时候，通过Port来使能或shutdown，通过这种方式430能够工作在间歇式工作模式下，系统的平均功耗只有 $0.06\mu A$ ，虽然它的运行模式是 $20\mu A$ ，但是平均功耗很低。所以，这也是我们在做一些低功耗系统设计过程中的一些小的技巧，让具有关断模式的运算放大器选择像这种器件，可以使系统的总的功耗可以降低20倍。

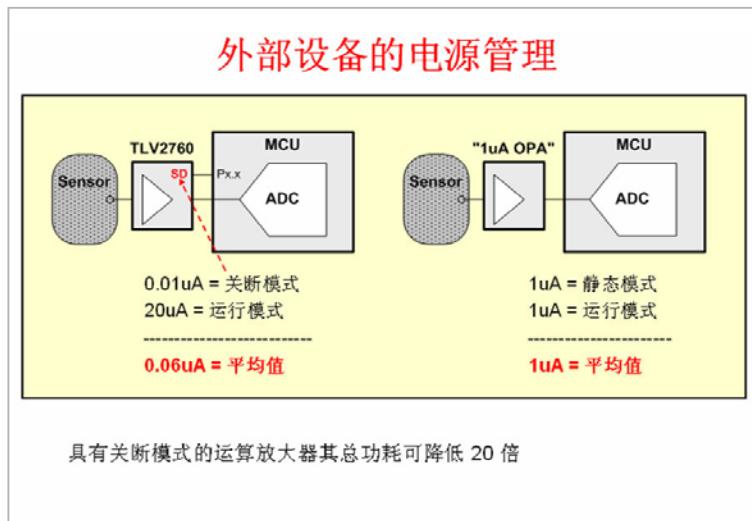


图6.12

另外一个技巧是关于引脚的设计，通常我们使用的单片机并不一定所有的引脚都会用得到，对于没有用到的引脚该如何设计？MSP430相应器件的User's guide文档中，通常在第二章的最后一页会介绍在设计过程中没有用到的引脚，该如何去设计可以让系统的功耗做到更低。

通常，未用的GPIO口设为输出，也可以由一个外围线路连接至Vcc或GND，或可接一个上拉/下拉电阻，但是这里特别要强调的是，千万不要让这个I/O口设为输入且为浮动状态。一般单片机的I/O口或者430的I/O口是一个推挽来输出的，当我们的I/O口置于输入，假如说外部受到一个 $1/2Vcc$ 左右电压的情况下，会造成整个推挽或者我们通常来讲上桥臂和下桥区的直通，会造成功耗的剧增。

另外一个也有可能把整个器件会击穿，产生很高的击穿电流，让系统的产品产生不可靠的一种变化。

未用引脚接口

- ◆ 数字输入引脚需防止遭受击穿电流的影响
 - ◆ 如果引脚设为输入且浮动，则当输入电压在 V_{IL} 与 V_{IH} 之间时的会产生击穿电流。
- ◆ 未用的GPIO 应当：
 - ◆ 设为输出
 - ◆ 可由一个外围线路连接至 V_{CC} 或 GND
 - ◆ 或可接上一个上拉/下拉电阻

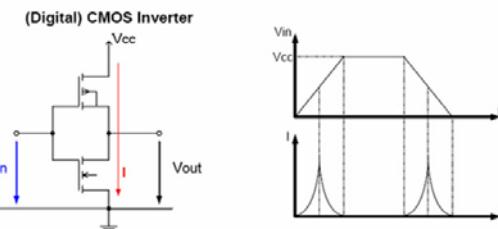


图6.13

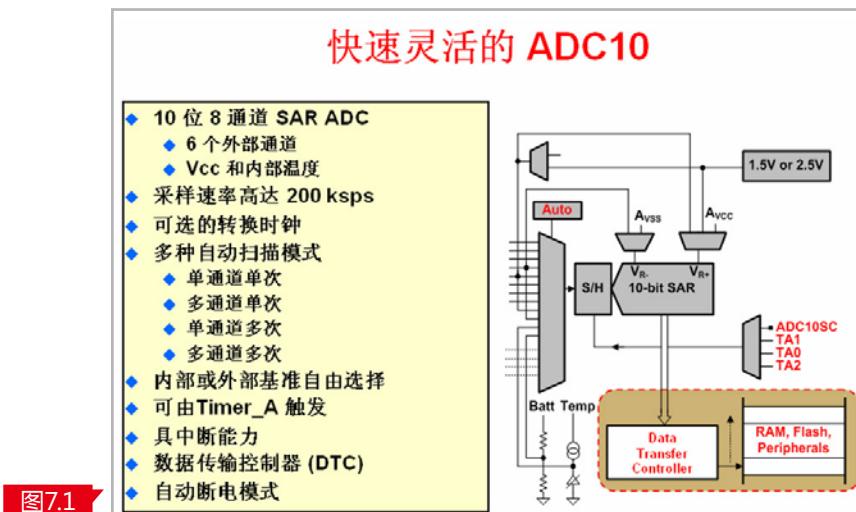


2.8 10位ADC和增强型的比较器

快速灵活的ADC10

430Value Line提供了一个10位的ADC，具有快速灵活的特点。它是10位8通道逐次逼近型的ADC，有6个外部通道，内部可以实现VCC和内部温度的计量和监测。它的采样速率或者转换速率高达200ksps，有多种可选转换时钟模式，如单通道单次、多通道单次、单通道多次、多通道多次等等。

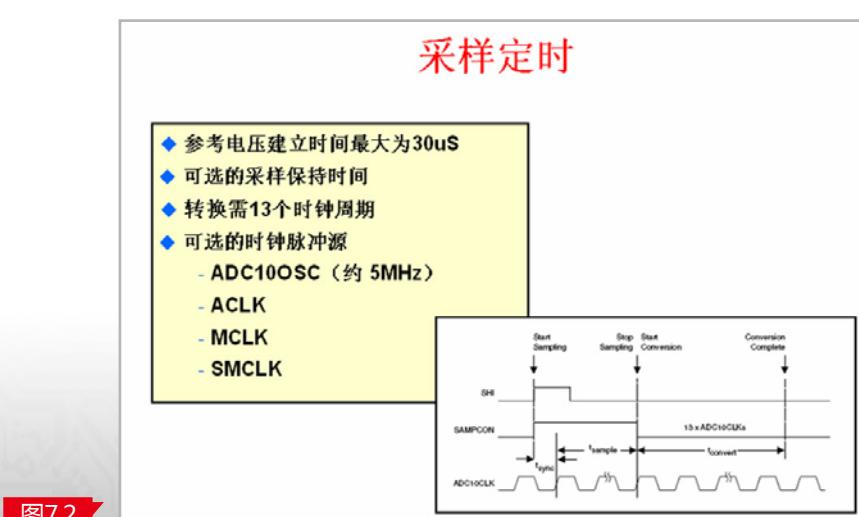
ADC内部基准和外部基准可自由选择，其中内部基准电压1.5V或2.5V可选。可以由Timer_A触发，具有中断能力。图7.1中可以看到ADC10有DTC功能，可以不需要CPU干预来实现外部模拟量，完成系统转换，并把数据转存到指定的RAM、Flash以及相应外设中。同时，在完成整个ADC转换的过程中，ADC10能够实现自动断电来满足低功耗要求。



ADC的采样定时

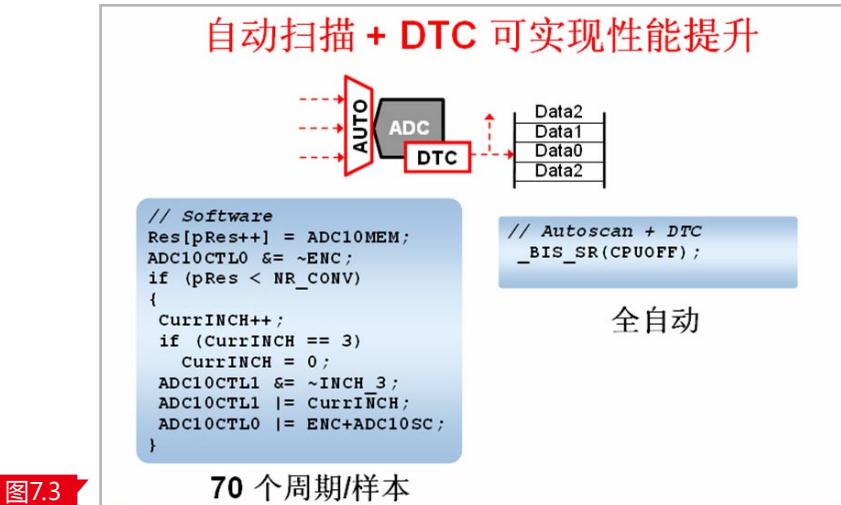
关于ADC采样定时，如果使用内部参考源参考电压，建立时间最大约为30 μ s，同时考虑到采样保持时间，可选定不同的采样保持时间，在转换过程中，当采样保持时间维持模拟量需要进行转换的时候，整个转换过程需要大约13个时钟周期。

对于ADC而言有4个时钟源可供选择，一个是内置的ADC10 OSC，它的默认频率约5MHz，以及可选ACLK、MCLK以及SMCLK作为不同的时钟源。图7.2下角是转换过程的图例，从Sample、Hold还有SAMPON，包括ADC转换的13CLK，这是一次ADC转换的大概过程。



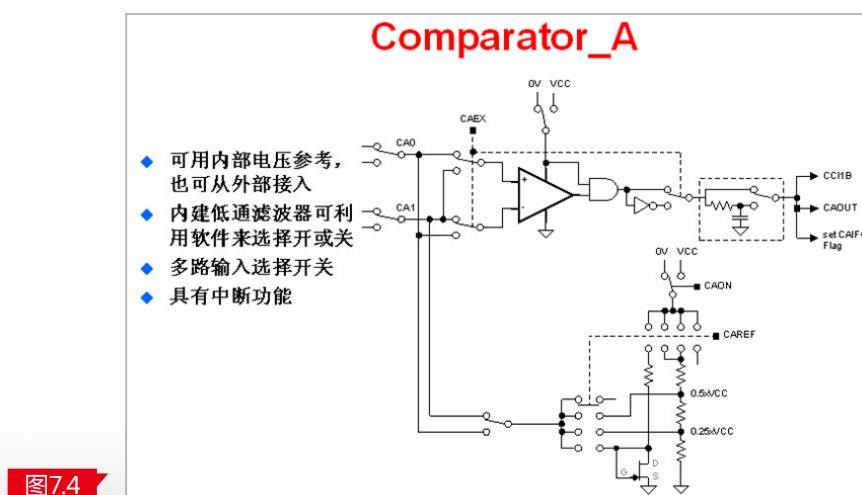
前文多次提到DTC功能，ADC10的自动扫描加上DTC可以将ADC转换性能进一步提升，并降低系统功耗。从图7.3中我们可以看到，AUTO可以自动的把现实世界中的模拟信号通过ADC转换成数字信号，DTC可以把转换的数字信号放到指定的任意空间去。

图7.3中也进行了对比，左边是要我们通过软件的方式实现ADC转换，其一次转换需要70个周期；而启动DTC后，Autoscan功能能够实现多个通道自动扫描+DTC，此时可以将CPU进入到睡眠状态，整个过程是全自动的，不需要CPU干预从而降低系统功耗。



比较器A

MSP430的比较器A框图参考图7.4，核心是一个比较器，可以选择不同的参考源。比如说0.5Vcc或1/2*Vcc和1/4*Vcc。同时，个片上置入约为0.55V的内置参考。CAR可以来打开和关闭整个比较器模块，正向输出和反向输出可以做一些切换，在比较器输出有可开和关的滤波电路，可直接达到比较器输出也可以做不同的中断标志。还可以直接输入到Timer的模块，这是一个功能非常强大的比较器。

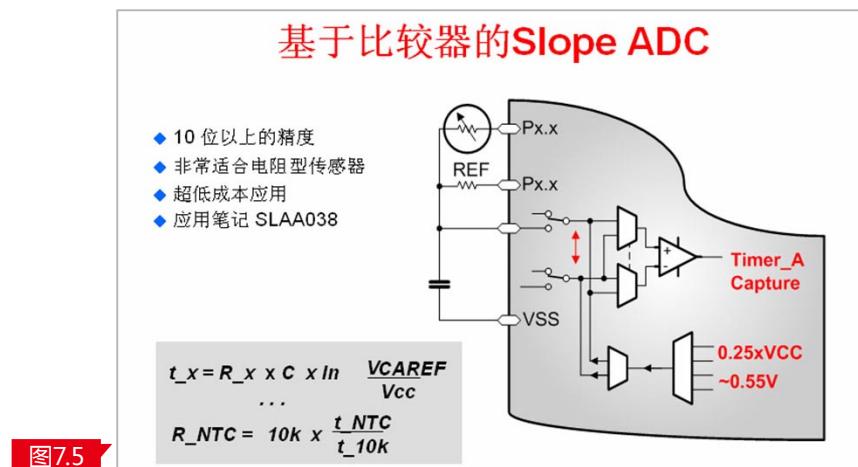


也许很多人用过430，一直在奇怪为什么会有个叫做Slope ADC的模块。事实上，

Slope ADC可以叫做斜边ADC,它是通过一种充放电的方式,对一种阻值型的传感器进行测量。

这种采样方式成本非常低缺可以实现10位以上的精度。图7.5右边的框图可以看到一个参考电阻,一个NTC电阻及一个电容,通过比较器进行充放电,同时把我们的比较器输出直接送到Timer_A的Capture单元。图7.5左下角是进行Slope ADC的计算公式,RTC的阻值 $=10k \times t_{NTC} \times t_{10k}$,该阻值的测量精度非常高,可以消除Vcc等一系列偏差的影响。参考电阻就是标准的充电时间和标准的放电时间,公式中用 t_{5tu} , t_{10k} 来表示,首先是 t_{5tu} 的充电时间然后是放电时间不同,因为阻值不同所以放电时间不同,因而得到这一推导公式。

采样的整个过程,可以参考TI官网的应用笔记application notes SLAA038。有括源代码、原理图,辅助开发者使用Slope ADC。



举例来说明Slope ADC,如图7.6所示,我们用Bit Set ADC soft conversion送到该寄存器,就可以开始整个转换过程。整个过程中,约触发采样16个CPU的cycle,所以没有软件等待的循环,因此能够做到更加精准和更低功耗。



MSP430的片上ADC

图7.7中展示，MSP430除了ADC10之外，还有包括ADC12, 12位逐次逼近型ADC、16位SD16, Sigma-Delta型ADC、增强型的SD16A，以及可能推出的24位的片上AD。

如此丰富的资源对于选择造成了一定的困扰。图7.7中可以看到不同ADC的架构，性能、精度的不同，可以根据需求进行筛选。

首先是Slope ADC是属于转换速率和精度相对较差，逐次逼近即SAR的ADC特性是转换速率很高，但精度只提供10位和12位。

其次是Sigma-Delta的ADC，最高可以提供24位的精度，包括有16位、增强型16位以及24位。针对如何选择，TI提供了一个表格，针对开发应用所需的电压范围、模拟量输入最大频率，以及设计所需要的分辨率，决定ADC的转换速率。同时，需要根据应用考虑是否需要差分输入、测量基准的范围以及应用是否有需要有多个通道。通过这些组合来对应ADC选择的表格，来选择满足需要的ADC。

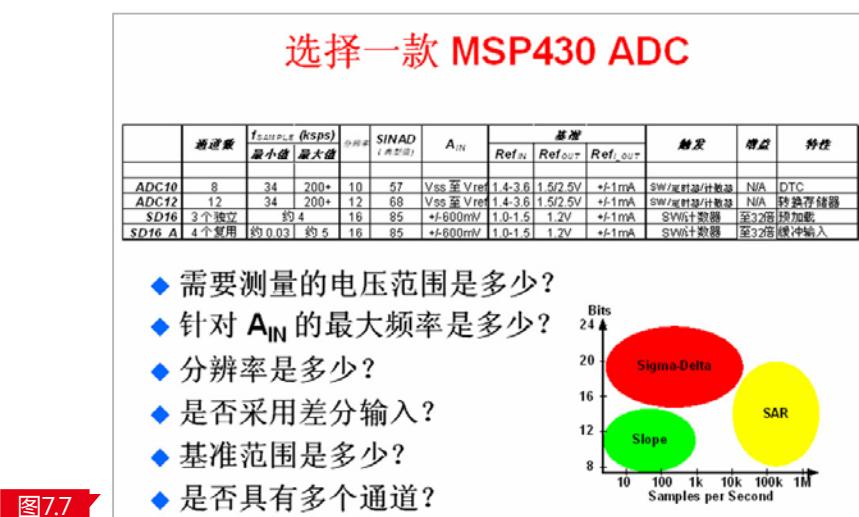


图7.7



2.9 串行通信模块

430的Value Line片上的串行通信模块包含USI模块、USCI模块，以及可以应用Timer_A实现更多UART接口。

430的USI

图8.1右边的框图，是USI系统框图，可以很方便的实现I2C和SPI。USI在430 Value Line的G2xx1/2器件里面可以支持可变长度的移位寄存器、支持I2C，包括通常所要求的START/STOP检测、SCL在计数器溢出之后的保持以及丢失检测机制。同时还可以支持8/16位的SPI，包含8/16位的移位寄存器，并可以设定MSB优先或者LSB优先选项，同时也支持中断的功能。

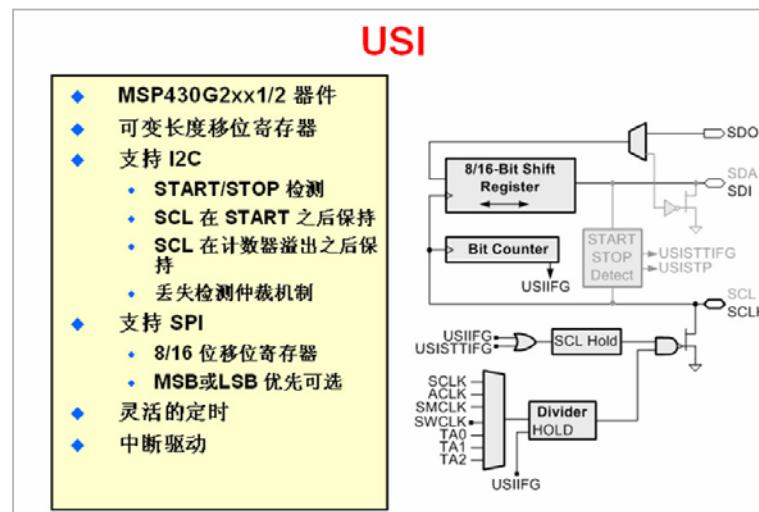


图8.1

图8.2展示USI的数据IO。数据移位寄存器可支持多达16位，发送和接收的位数受控于一个位计数器。从这个角度来看，这是一个位计数的串行通信模块，其发送和接收是同时进行的，用户的I/O或者数据I/O是由用户定义，可设定MSB或者LSB优先。

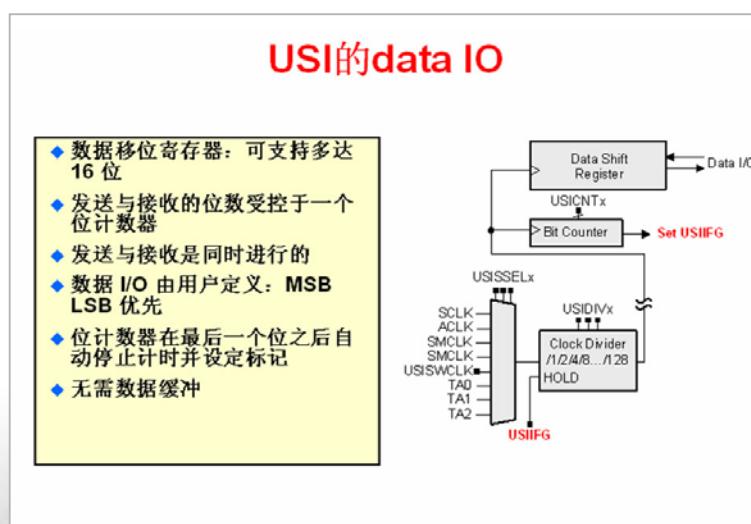
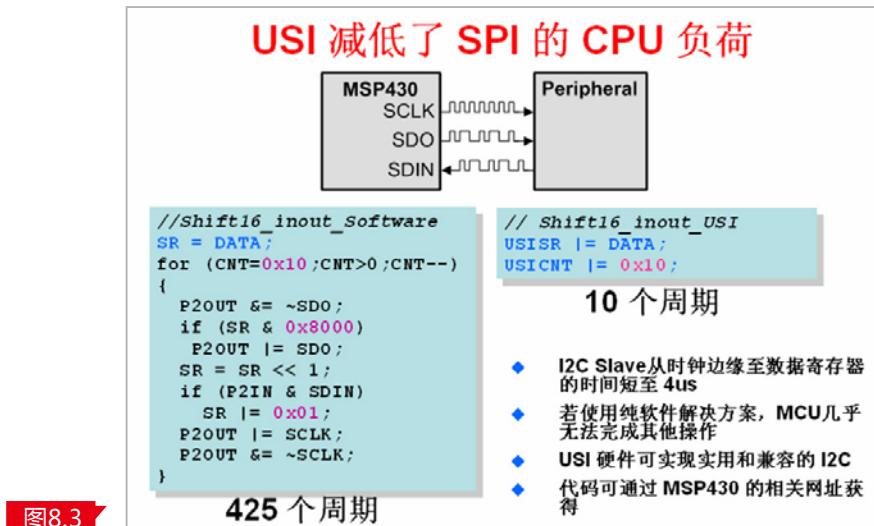


图8.2

位计数器在最后一个位之后自动停止计时并设置相应的标记位。同时，不需要特别的数

据缓冲，使用过程非常简单易用。USI的存在大幅降低了SPI的CPU负荷，这是专为低功耗设计而设计的。

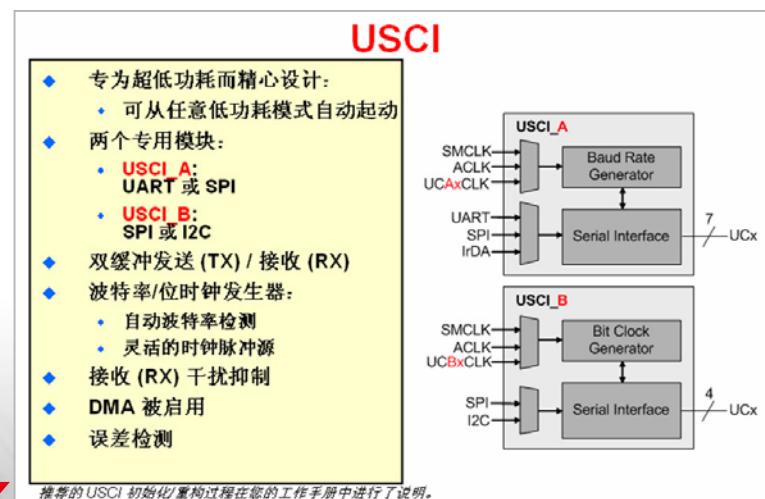
图8.3中左侧是一个需要CPU干预的SPI的编程方法，也是通常所说的用I/O口模拟SPI的通信过程；右边是用USI模块来实现SPI。针对CPU功耗进行比较，可以发现纯软件基于430实现的方式需要425个周期；而使用USI实现SPI只需要10个周期。因此，USI可以实现实用和兼容，实现代码可以通过430相关网站找到很多关于USI实现SPI和I2C的代码。



USCI

USCI相比USI而言功能更加强大。在G2xx3系列中，片上的USCI模块包含两个分立模块，一个是USCI_A，可以实现UART和SPI的功能；另一个是USCI_B，可以实现SPI和I2C的功能。

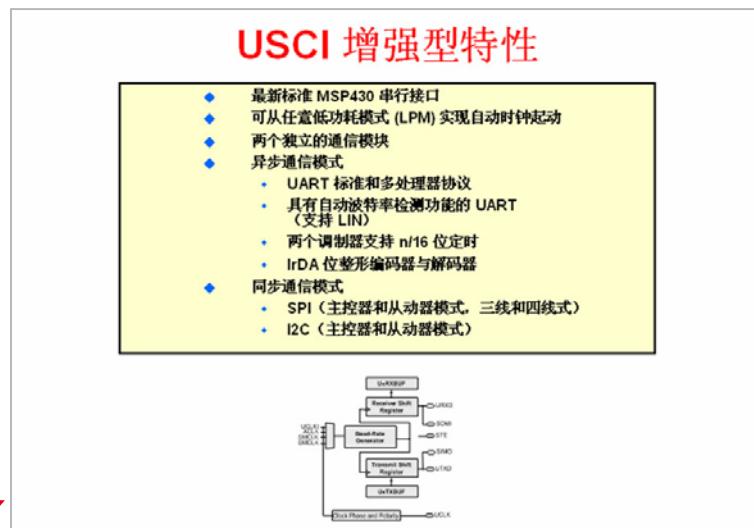
在G2xx3中如果使用SPI接口，可以同时实现两个SPI接口。USCI是专为低功耗设计而设计，可以从任意低功耗模式自动启动，USCI支持双缓冲发送/接收，可以实现波特率位时钟发生器的自动波特率检测，以及灵活的时钟脉冲源的设计。同时，USCI还包含接收干扰抑制，可以与DMA结合使用，同时包含误差检测的功能。图8.4介绍了USCI_A的模块和USCI_B的模块，不同的功能模块主要的异同点。



USCI还有一些增强特性。

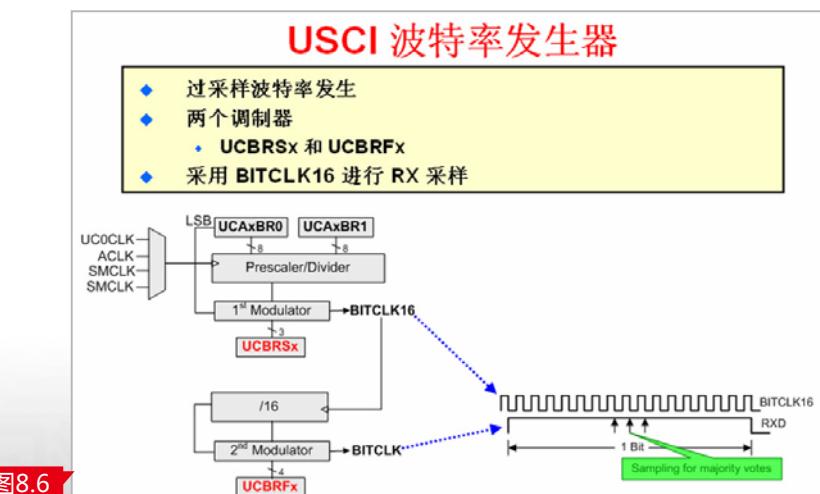
首先, USCI是后续所有新增430产品的标准的模块, 也可以说是最新标准的430的一个串行接口。它可以从任意低功耗模式LPM实现自动时钟起动, 包含两个独立的通信模块, 同时支持异步通信模式。

UART的标准和多处理器协议方面, USCI可以自动支持波特率检测功能的UART, 同时也支持LIN BUS。包含两个调制器, 支持N/16位定时, 同时支持IrDA位整形编码器与解码器。作为同步通信模式, USCI可以支持SPI, 包括主控器和从动模式、三线和四线式的SPI, 同时执行做主、做从的I2C模式。



USCI波特率发生器

过采样波特率发生器可以实现两个调制器, 通过这种majority的检测方法能够保证波特率很准确的进行。USCI内置两个调制器, UCBRSx和UCBRFx, 可以采用BITCLK16进行RX采样, 可以保证误差率非常低的波特率来实现UART。



Value Line的通信模块

Value Line的通信模块包括USI和USCI。USI即在G2xx1/2系列产品中，USCI在G2xx3产品中。USCI支持UART，两个定时器、两个调制器，支持N/16的定时，自动波特率检测、IrDA编码器包括LIN BUS等等，有两个通道，USCI_A和USCI_B。

USI模块不支持UART，USCI支持SPI，USI也支持SPI，但是USCI的SPI模式功能更为强大，包括支持Master和Slave模式，包括支持三线、四线模式，USI的SPI只能提供一个SPI。USCI的I2C可以简化中断用法，支持Master和Slave模式，最高速度可以400kbps。USI的I2C需要软件，也就是state machine状态机的控制，但它同时也支持Master和Slave的模式。

Value Line 通信模块	
	USCI USI
UART	通用串行通信接口 G2xx3
SPI	两个调制器：支持 N/16 定时 - 自动波特率检测 - IrDA 编码器与解码器 - 同时 USCI_A 和 USCI_B (两通道)
I2C	两个 SPI (USCI_A 和 USCI_B 上各一个) - 支持 Master 和 Slave 模式 - 三线和四线模式 - 简化的中断用法 - 支持 Master 和 Slave 模式 - 高达 400kbps

图8.7

前文所述的很多应用需要很多UART口，但同时又对低功耗有较高的要求，如何来做到这一点呢？可以通过Timer_A轻松实现。

首先，这是一个100%硬件实现锁存与输出的方式，而且同时能够依靠LPM3和LPM4实现全速性能，实现非常低的CPU功耗，从图8.8中的框图可以看到，有SCCI和Output模式，可以完全自动的实现移位和输出，非常便于使用。整个用法CPU的功耗非常低为6.7%，具体的实现方法，可以参考Application Notes SLAA078。

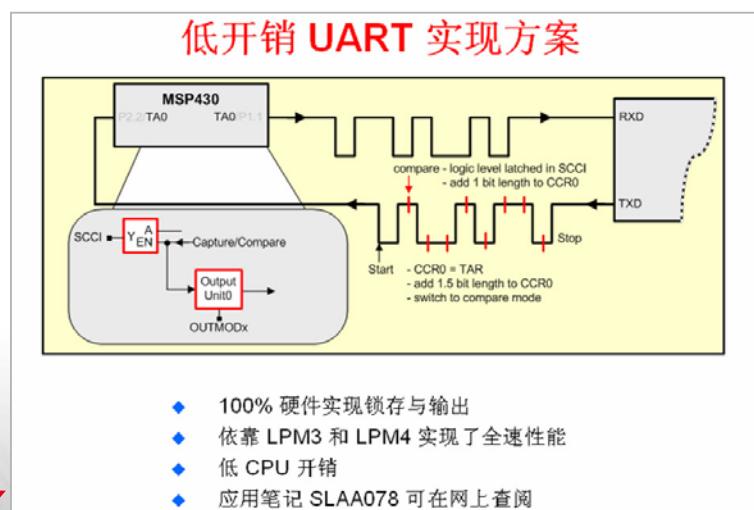


图8.8

2.10 Grace软件

什么是Grace?

Grace是一种免费的图形化界面,它是什么的简称? Grace是Graphical Code Engine。这是一款免费的图形用户界面,可生成源代码,免除手动设置外设。



图9.1



图9.2

需要强调的是,它不是一种图形应用程序编制器,不是用来生成图形的,它是利用TI的集成开发环境CCS生成430外设初始代码,可以优化代码,可以使使用430的新用户在15分钟内运行程序的开发工具,它更专注于易用性。

TI有计划进一步扩展Grace的应用,希望能覆盖所有的MCU器件。

Grace可以直观的使用和配置430的外设,图9.3中我们可以看到,很像430的Device框图。下面是一个外设,ADC10 10位的SAR的Overview。以这种图形化的形式,通过拖拽的方式就可以完成对ADC的匹配配置。

所以,开发人员可用按钮下拉菜单或者文本字段进行交互,轻而易举的完成对底层寄存器的设置。同时Grace可生成经全面注释的C代码,适用于MSP430产品线所有的F2xx和G2xx的Value Line微控制器。

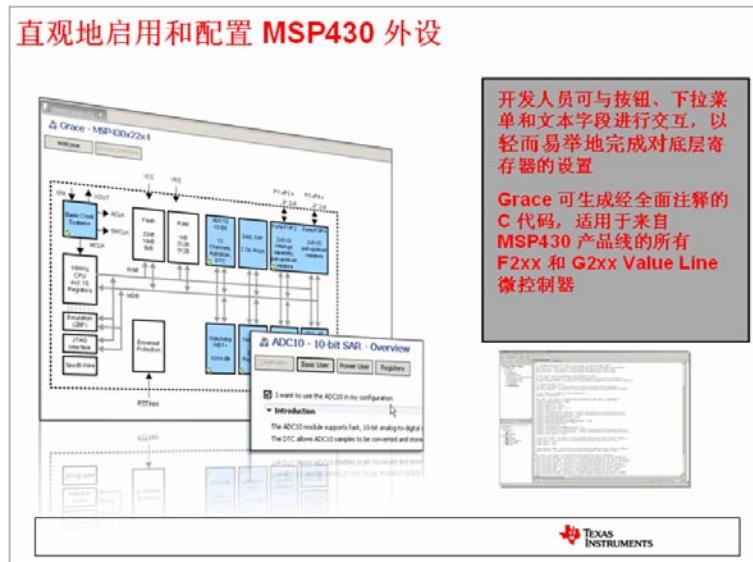


图9.3

图9.4是用Grace来做项目结构和构建的流程,通常会用C/C++完成的源文件。但对于单片机而言,可以通过某个Device的cfg文件,就是器件外设的配置文件,采用图形化的方式,也就是Grace视图的方式进行编辑。这样在Debug或在Release文件夹内部就自动生成包含用于所有已经配置的外设进行初始化MSP430的C代码,然后通过编译器把应用程序和我们外设配置文件编译到一起,最终通过链接器生成.out文件,也就是最终可以执行的430的输出文件。

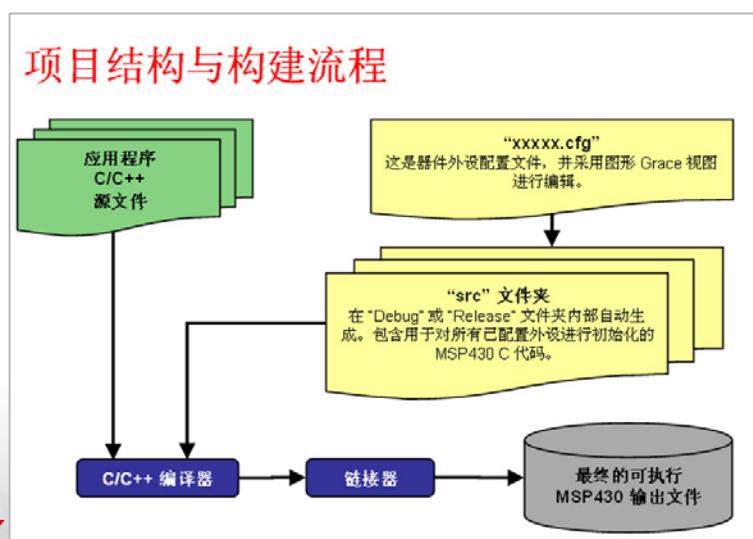


图9.4

Grace最大的特点是用户可以把大部分的时间花在和专注在自己的应用程序上，而不是如何去用MCU和如何配置MCU上去。图9.5是开发代码框架示例。在自动生成的源程序可以看到，include 430.h，是我常用的430通用头文件，也会包含CSL文件。主控制器包括运行所有Grace关联的文件，同时会形成CSL initial函数，这是执行所有由Grace配置的外设函数。

从图9.5的方框中可见，所有算法、创新、和独有IP放在用户代码开始部分，这样就很容易完成了单片机的初始化，其余工作完全由Grace来完成。



图9.5

如何在Grace添加一个外设？非常的简单，只需要启动Grace这个图形化的界面在CCS里面，然后在外设单击右键并选择“使用”就可以使用这个模块即可完成，所有标有蓝色的模块均可进行配置和使用，如图9.6所示。

Grace – 添加一个外设

- 在该外设上单击右键并选择“使用 (Use)”
- 所有着蓝色的模块均可配置

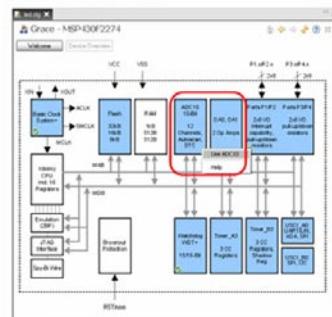


图9.6

那么怎样告知添加了一个外设？

可以在CSL视图上观察该外设的左下角,如果此外设被初始化,则会显示一个绿色的对钩标记。很容易的可以判断出该部分已经被配置过了,如果我们再去重新配置,点击进入进行相应的配置。

怎样告知添加了一个外设?

- 在CSL视图上观察该外设的左下角,如果此外设被初始化,则将显示一个绿色的钩型标记。

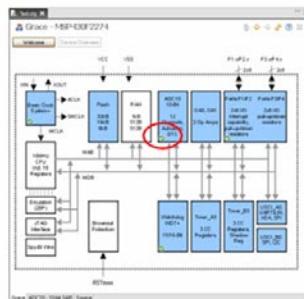


图9.7

如图9.8所示,在Grace界面的右上角会看到有一些图标,所以在某个外设上单击左键以导航至其细节,可以看到从Overview到非常detail的一些情况。同时也可以使用主画面按钮以返回至顶层器件的Overview的视图,同时还有前进/后退按钮来完成对Grace软件的使用,通过导航来完成对不同模块的配置和选用。

Grace – 导航

- 在某个外设上单击左键以导航至其细节视图
- 使用主画面按钮以返回至顶层器件视图
- 还可使用前进/后退按钮

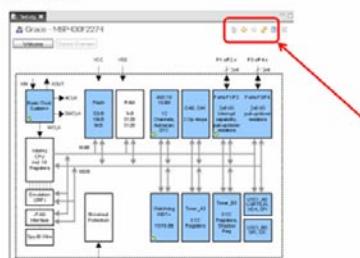


图9.8

Grace来配置一个外设也很简单,例如使用BCS Basic Clock System的Overview的情况大概有4种,一个Overview的大概的外设的简介,有Basic User就是基本用户,只做high level,

从很高高度来做一些基本配置。对于Power User也就是高级用户而言，同时还有Registers的这种选项，可以详细到每个寄存器进行配置。

编译时可以编译其中的任意一个，他们之间是相互通联的。无论是用Basic User来进行配置，还是用最高级的Registers的方式进行配置，实际上能得到相同的结果。完成好相应的配置之后，通过点击图9.9中红色箭头所指向的Refresh来确定当前的配置，就完成了对当前外设的配置。

Grace – 配置一个外设

- 每个外设具有4种不同的表示：
 “Overview”（概要）
 “Basic User”（基本用户）
 “Power User”（高级用户）
 “Registers”（寄存器设置）
- 您可以编辑其中的任一个，它们是完全连通的
- 通过点击“Refresh”来确认当前的配置



图9.9

如何移除一个外设？

在Grace的视图，如图9.10单击所选定的模块，右击选择StopUsing进行停止使用，就可以把前文所述的绿色对钩选项去掉，就停止了对该外设的配置。

Grace – 移除一个外设

- 在该外设上单击右键并选择“Stop Using”（停止使用）

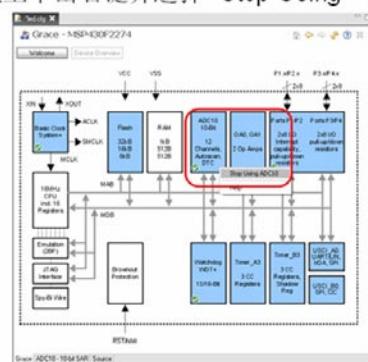


图9.10

2.11 基于430实现电容式触摸按键的解决方案

什么是电容式触摸？

平板电容的原理是，当一个正极一个负极两个电极之间会有电动势存在。当我们用导体或者半导体去接近于改变这个电动势时，会改变这个感应电容或者这个平板电容容值的变化。

如图10.1，在PCB上画一个类似于触摸按键的形式，可以看到在中间的焊盘与周围的地方会形成不同的电容，比如说C1、C2。当我们用手指去靠近所谓的电容按键的时候，会改变这个电容的变化。从原理上，会增加C3和C4，从而导致整个触摸按键的电容： $C_1 + C_2 + C_3 \parallel C_4$ 的容值增大。这就变成了一个触摸按键至大地自由空间耦合通路的一部分，从而使电容值产生变化。

那么如何来实现这种触摸式按键呢？可以通过一些方法把这种电容的变化转变成频率的变化或者是脉冲的变化，从而实现检测手指按与不按或手指在与不在形成电容式触摸按键的过程，就叫做电容式触摸按键。

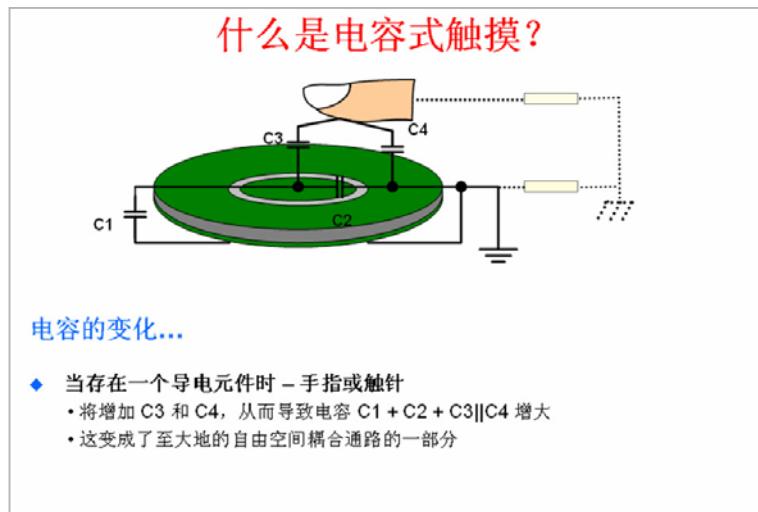


图10.1

MSP430电容式触摸检测方法

基于430实现电容式触摸按键有很多种方法。例如Value Line系列产品G2xx2和G2xx3，内部有Pin oscillation，称之为cap touch RO的器件。它有一个Pin oscillation的方法，可以自动的把外部电容的变化转变成内部频率的变化，通过技术的方式，实现触摸按键的检测。其最大的好处是不需要任何的外部组件就可以实现触摸按键。目前G2xx2和G2xx3的产品支持Pin oscillation的方法，每个按钮的平均功耗约3微安。

第二种是RO方法，实际上是relax oscillation，就是张弛振荡器的简称。通过部分430产品带有比较器，通过比较器构建一个张弛振荡器，来实现把电容的变化转变成频率的变化，从而实现按键的检测。RO方法的最大优势在于它有很强的抗干扰能力，因为比较阈值

会在 $1/3V_{cc}$ 和 $2/3V_{cc}$ 之间，有较强的干扰能力。同时，使用内部定时器和比较器，还需要一些外部电阻网络，所以需要较多的外部元器件。同时，也需要具有比较器的430器件支持，如G2xx1有部分器件。

第三种是RC方法。RC方法属于充电和放电方式，charge和discharge。是功耗最低的检测方法，最多可以支持16个按键，只需要GPIO和定时器，任何430的器件都支持，只要包含Port1, Port2, 16个具有中断能力的IO口都可以使用RC的方法，在所有的方法中功耗是最低的，每个按钮约 $1\mu A$ 。相对于其他的两个方法，它只需要对电容进行一次充电和放电检测，相对抗干扰能力较弱，常适合于电池供电、对功耗要求较高的场合。

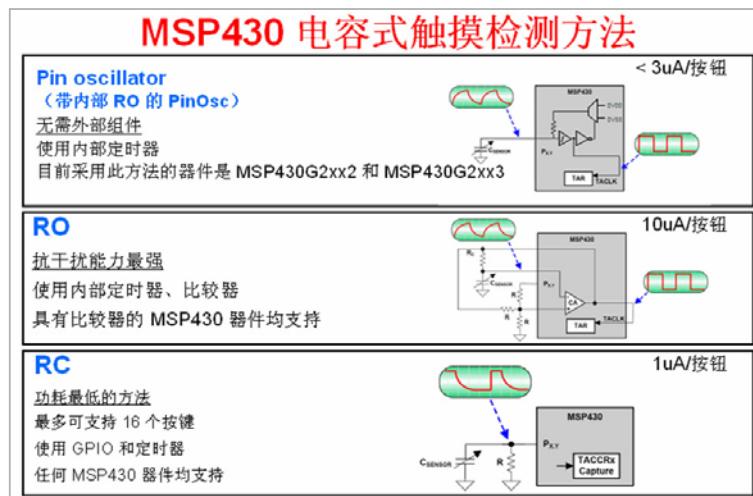


图10.2

基于上述方法，TI从满足大家易用的角度出发，建立了CAP TOUCH LIBRARY也就是触摸按键的使用结构库，它通过Element、Sensor、Schedule，通过API的函数的方法很容易的调用相应的RC、RO包括CAP TOUCH L的方法来满足实现触摸按键的功能。

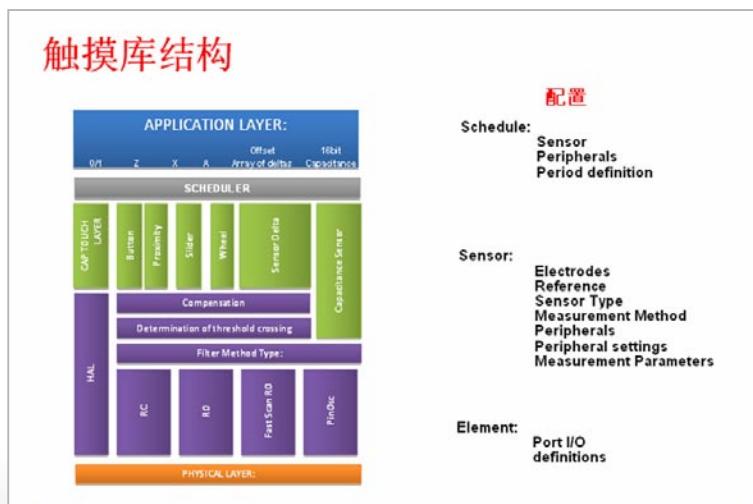


图10.3

基于Launch Pad的电容式触摸实验板

TI的基于LaunchPad电容触摸板叫做BoosterPack。它是可以通过链接器链接到Launch Pad上实现电容式触摸按键功能。它可以实现触摸按纽、滚轮和接近式感应。通常会用一颗带有电容式触摸I/O，即CAP TOUCH I/O的G2xx2的芯片进行控制。同时，触摸板针对滚轮及接近感应有很好的设计范例，并它支持前文所述的电容式触摸感应代码库，可以利用代码库设定各种已经验证的功能，整体价格约10美金。



图10.4



● 第三章 高手实战篇

3.1 基于LaunchPad的【低功耗时钟】

自从买了LaunchPad就一直想做点什么。想做一个低功耗时钟，正好看到手边有一个废弃的电子表，就拿来拆掉改装。

这次DIY涉及到的内容：GPIO操作，时钟的操作，中断应用，看门狗作定时器，段式液晶的直接驱动，触摸按键的操作。说一下关于段式液晶和触摸按键的心得体会吧！

之前我想大家用的最多的显示器件大体可以分为两类：带控制器的液晶和LED数码管。数码管的原理我想很容易理解，就是在某一段的两段通上电压就会亮。段式液晶其实也是这样，只是它需要的不是直流电，而是交替变化的电压。举个例子，假如某一个公共端的时序是0-1-0-1，那么若想让该公共端控制的一个段显示，则在另一端上要加的时序就是1-0-1-0，不显示就是0-1-0-1。

但是同时问题也来了，由于IO口只有两个状态，当一个公共端正在使用的时候，别就不能起作用，但是它的状态要么和正在使用的相同，要么就相反。相同肯定不行，但是如果相反的话，原来不想显示的段就会显示出来了。所以要想个办法给不使能的公共端找个位置，这就是“1/2 Bias”的目的了。让不使能的公共端的电压保持在0.5，这样别的段不管是0还是1，和它的压差只有0.5，虽然理论上还是会有显示，但是相对于压差为1的就不明显了，实际的结果也说明这种方法是有效的。当然还有1/3、1/4的驱动方法，但是用普通的IO口已经不可能做到了。

那么最关键的，如何用一个IO口产生1/2电压呢？其实说起来也简单，只要把IO口设为输入，然后在外部用两个电阻分压就行了。还有就是频率，段式液晶一般在30~60Hz，这个说的是全部扫描一遍的时间。就实际应用来说，我的液晶有两个公共端，11个段选，所以用了四个电阻，其他的都直接接到了单片机的IO上。

在程序执行时，用看门狗做定时器，每8ms产生一个中断，这样四个中断完成一个显示周期，频率是31Hz，勉强可以。本来我是用TimerA做5ms中断的，这样频率是50Hz，比现在的效果要好。但是后来开始做触摸按键才发现竟然要用两个定时器，TimerA和看门狗，这就不没办法计时和显示了吗...后来在资料里找来找去才找到一个只用TimerA的，但是看门狗只能产生固定周期的中断，所以就只好用31Hz的了。

具体在程序里，每一个显示周期分成了四个节拍，两个公共端的时序分别是1-0-1/2-1/2和1/2-1/2-1-0；在第1、3节拍里各个段选根据要显示的内容输出0（显示）或1（不显示），在第2、4节选只要简单的把使能的位选和各个段选取反就可以了。

触摸按键的使用:

触摸按键估计是这次LaunchPad最吸引人的地方了吧，当初有多少人是冲着那块触摸板才买的啊....嘿嘿。那么这次既然是用LaunchPad做DIY，不做一个触摸按键恐怕有点说不过去吧，所以就照着触摸板的原理图加了三个触摸按键。

硬件上很简单，就是三块覆铜直接连到了P2口的IO口上。看技术文档介绍在背面应该做一些敷铜接地，但是不做好像问题也不大。

触摸按键的使用主要是在软件上做文章，这也是我这次DIY花费时间最多的地方，首先说一下原理吧，其实原理也不复杂。当激活了P2口某个引脚的第二功能后，该引脚立即就会产生1M左右的震荡，这个频率是和该引脚上的电容直接相关的。而触摸与该引脚相连的覆铜就足以产生使频率改变的电容变化。所以我们需要做的就是测量出频率的变化值。

具体的应用上TI已经提供了详细的Capacitive Touch Library，里面有很多做触摸感应的方案，以及操作的API函数。

我们需要做的就是现在structure.c和structure.h里定义好我们要使用的元件(element)和传感器(sensor)，传感器指的是种类，比如button、weel等，同一个传感器可以包含多个元件，比如button1、button2等。

同时在定义传感器的时候也需要定义要使用的测量方案，针对不同的方案还有一些参数需要设置，比如用TimerA做计数，看门狗做门限的话需要设置看门狗的溢出周期，如果是用软件做门限时的话就要设置软件技术的计数值。

在定义元件的时候也有针对各个元件的参数设定，主要是最大计数和门限计数。这两个参数是给API函数判断是否按下用的。

具体的信息在Capacitive Touch Library都有很详细的介绍，只是要耐下心来一点一点看。内容上没有什么难度，关键的是耐心。

还有如果碰到问题的话可以多参考一下TI提供的触摸板的例程，看一下他操作的方法和各个部分定义的顺序也会很有帮助。再一个就是头文件，单片机上所有的资源都会在头文件里提到，仔细找一找也会很有收获。

关于低功耗:

说实话最近还没来得及仔细研究低功耗方面的问题，只是直接测了一下工作的电流，一开始没加休眠，主函数里的while(1)是一直在空转着的，测的电流大概在500uA左右。然后我在里面加了一句__bis_SR_register(CPUOFF); 不执行中断的时候就把CPU关掉，再一测，电流直接掉到200uA左右了。看来确实还有很大水分可以压出来啊。

详情：<http://bbs.eeworld.com.cn/thread-309750-1-1.html>

3.2 触摸按键MSP430低功耗时钟

1. 功能

通过MSP430读取DS1302，获得时间信息，并将其通过液晶显示器进行显示。

2. 硬件信息

(1) 主芯片：MSP430G2231->LAUCHPAD

(2) 按键：(可以改下, 只用一个按键, 用来切换日期和时间, 留个IO口驱动蜂鸣器, 闹钟的时间可以在下载程序的时候就设置好。就是只好通过重新修改程序来改变。)

S1 -----> P1.5(外接) || Mode Select

S2 -----> P1.3(LAUCHPAD 自带) || Change Value

(3) 显示：16段液晶显示器

(4) 显示驱动：HT1621B

DAT -----> P1.0

WR#-----> P1.1

CS# -----> P1.2

(5) 实时时钟信号：DS1302

SCLK-----> P1.7

SDA -----> P1.6

RST -----> P1.5

3. 实物图：

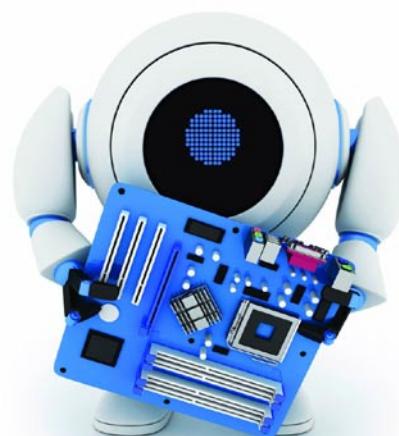
(1) 整体图



(2) 初步效果图

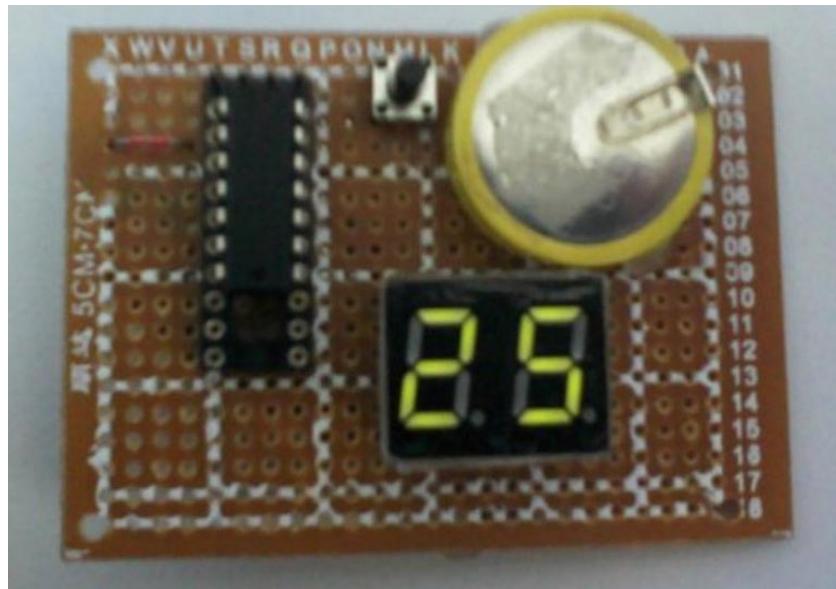


更多详情：<http://bbs.eeworld.com.cn/thread-307515-1-10.html>



3.3 MSP430G2231简单温度计

有幸团购到了LaunchPad团购。第一次接触到了MSP430单片机。也没太多时间，就做了一个简单的温度计。开始使用了ADC0外接NTC热敏电阻，其他IO口驱动了一个2位数码管，外加一个复位开关和电池。后来发现MSP430G2231有内部温度传感器，就改成内部温度传感器了。温度显示一段时间后，进入待机模式，按复位开关后重新显示温度。



附件为基本代码  TEST.rar (1.41 KB)

更多详情：<http://bbs.eeworld.com.cn/thread-309948-1-1.html>

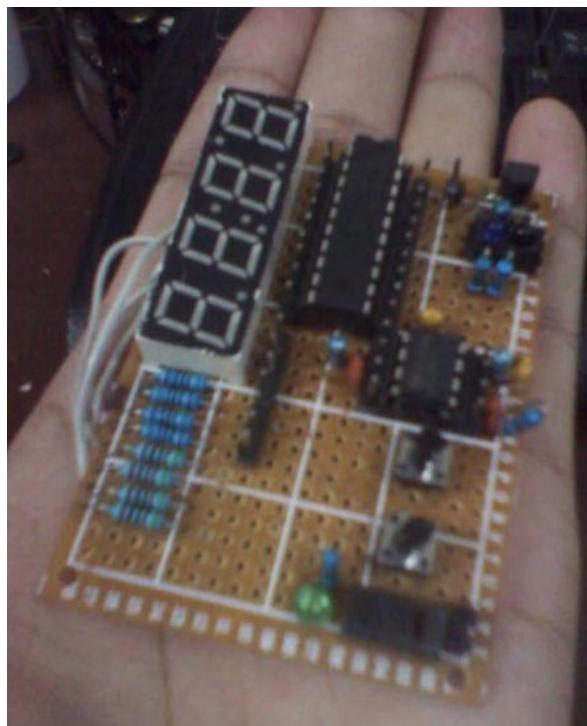


3.4 MSP430 LaunchPad 心率仪

在团购的群里看到有人给了一个用PIC单片机制作心率仪链接，做得很不错，是外国人做的！拿到MSP430 LaunchPad本来是想做一块触摸板的，可是图不好画，后面又补发了触摸板，所以原来计划作罢。前段时间比较闲，看看有网友翻译了那篇用PIC单片机制作心率仪，就模仿做了一个。水平有限，请大家拍砖！

在原来的基础上我做了一些该进，我的是四位数的，最高位是时间提示位-----也就是提示被测的人已经按住多少时间按了。15秒一到，显示被测者的心率。显示部分我本来想用段码液晶的，可手头上没有合适的屏。把四级听力的耳机拆了，弄到一个屏可不懂怎么控制。弄明白了再改进。

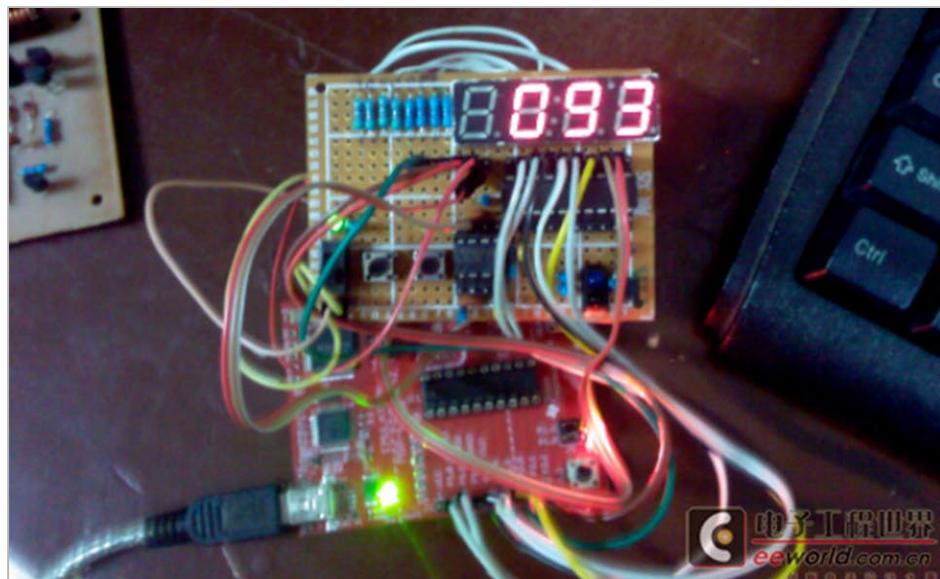
不足之处，误差比较大，有时会弄100以上，这次93还是比较好的，我在程序里允许有3次误差-----也就是显示93实际测到有96。



整机图



开始计时时



整体效果

更多详情：<http://bbs.eeworld.com.cn/thread-309073-1-2.html>



3.5 TI Launchpad 音乐 《欢乐颂》



如图, TI Launchpad 驱动蜂鸣器(喇叭)演奏音乐《欢乐颂》;代码见附件。

运行主频为 1MHz ; 所用芯片为g2553,当然其他芯片也可以,仅需修改芯片型号即可, P1.0端口接蜂鸣器。

精准延时运用:

```
#define CPU_F ((double)1019000) //1 MHz
#define delay_us(x) __delay_cycles((long)(CPU_F*(double)x/1000000.0))
#define delay_ms(x) __delay_cycles((long)(CPU_F*(double)x/1000.0))
//MCLK = SMCLK = CALxxx_1MHZ = 1MHz
延时100毫秒
delay_ms(100);
延时100微妙
delay_us(100);
```

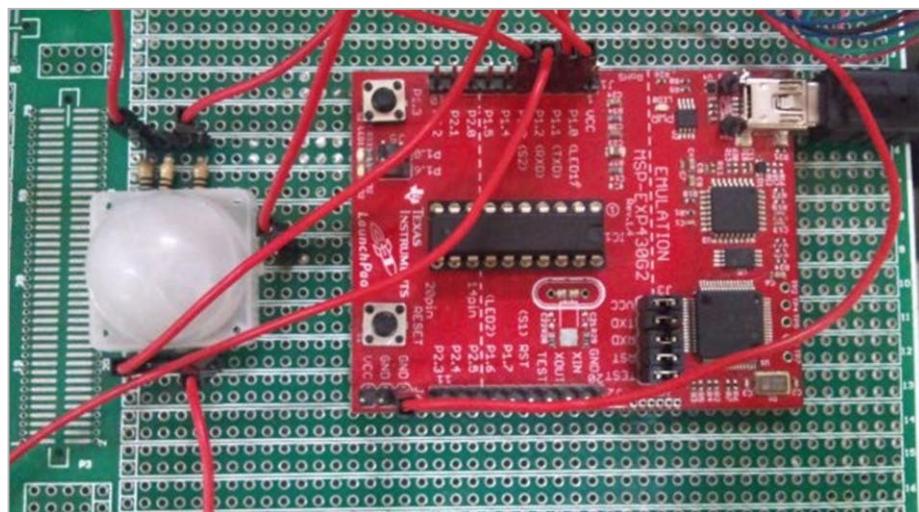
每个音符对应固定的频率,而频率则由上述延时实现。

源文件  欢乐颂.rar (846 Bytes)

更多详情: <http://bbs.eeworld.com.cn/thread-319288-1-1.html>

3.6 Launchpad + 呼吸灯这样才好玩~

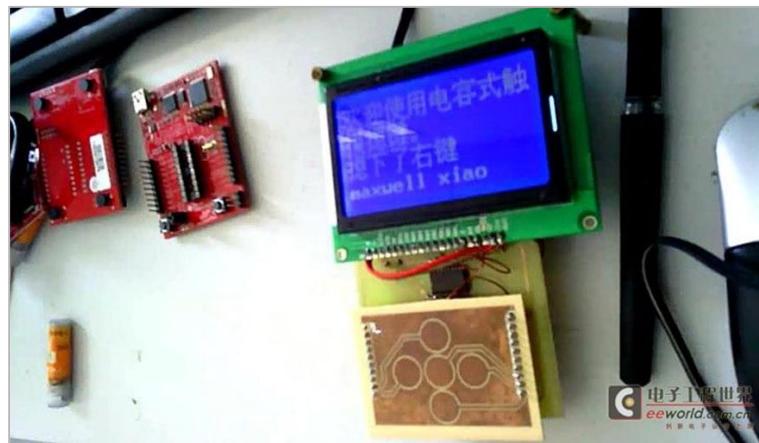
实物图：



更多详情：<http://bbs.eeworld.com.cn/thread-333772-1-1.html>



3.7 分享一下自制的触摸摁键，有程序，有视频，有真相



分享自制的触摸摁键，上图诱惑下大家，自拍了个视频，效果可能不是很好。

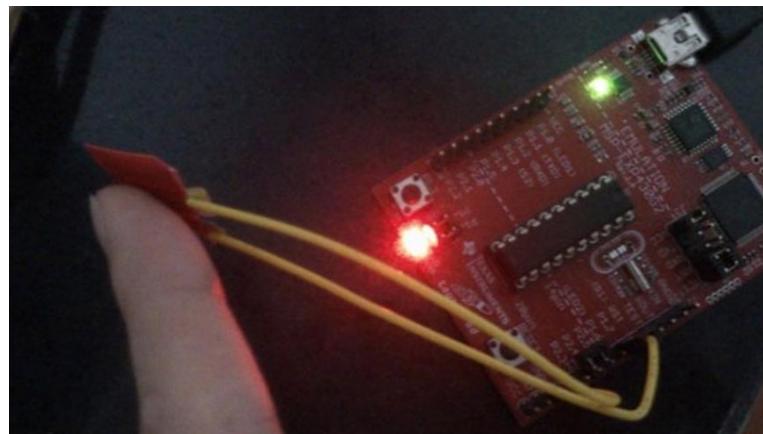
首先说明下我在structure.c做了点接口BIT位的小改动。为的是适应板子。

更多详情: <http://bbs.eeworld.com.cn/thread-344006-1-1.html>



3.8 木有风火轮，用锡箔做电容触摸感应

不得不感叹一下TI产品的设计周到，RO的电路都内置了，外部只需要一个感应电容。



从TI的触摸库中抽取了一段代码简单改了下面的函数：

```
unsigned int TI_CTS_RO_PINOSC_TA0_HAL(unsigned char bit, unsigned int accumulationCycles);
```

调用即可获得P1的bit端口(直接连Csensor)的周期计数，放大accumulationCycles倍。由于触摸会增大Csensor，减小频率，所以触摸是这个函数返回的值会变小。

更多详情：<http://bbs.eeworld.com.cn/thread-332984-1-1.html>



▶ 附录一 Launchpad 资料汇总

👉 手把手教你使用Grace开发LaunchPad

<http://bbs.eeworld.com.cn/thread-303736-1-1.html>

👉 手把手教你使用TI MSP430 LaunchPad

<http://bbs.eeworld.com.cn/thread-303599-1-1.html>

👉 LaunchPad之IO 的使用

<http://bbs.eeworld.com.cn/thread-314354-1-1.html>

👉 MSP430 LaunchPad演示应用：内部温度测量

<http://bbs.eeworld.com.cn/thread-303601-1-1.html>

👉 MSP430 LaunchPad学习第一记

<http://bbs.eeworld.com.cn/thread-205918-1-1.html>

👉 【玩转LaunchPad】应用文档指南

<http://bbs.eeworld.com.cn/thread-314437-1-1.html>

👉 【玩转LaunchPad】例程及共享代码

<http://bbs.eeworld.com.cn/thread-314443-1-1.html>

👉 基于MSP430的触摸按键

<http://bbs.eeworld.com.cn/thread-330697-1-1.html>

▶ 附录二 玩转LaunchPad实用问答

- 👉 1、MSP430 LaunchPad在哪里购买?
<http://bbs.eeworld.com.cn/viewthread.php?tid=294629>
- 👉 2、哪里能够获得LaunchPad开发板原理图?
<http://bbs.eeworld.com.cn/thread-221644-1-1.html>
- 👉 3、LaunchPad实验板触摸感应到底是怎么回事?
<http://bbs.eeworld.com.cn/viewthread.php?tid=238170>
- 👉 4、MSP430 LaunchPad为什么没给光盘呢?
<http://bbs.eeworld.com.cn/viewthread.php?tid=305761>
- 👉 5、Win7下,我的LaunchPad驱动安装不了啊?
<http://bbs.eeworld.com.cn/viewthread.php?tid=309135>
- 👉 6、win7 串口安装不上
<http://bbs.eeworld.com.cn/viewthread.php?tid=309140&extra=&page=1>
- 👉 7、不知道LaunchPad能不能给F2XX系列的单片机下载程序呢?
<http://bbs.eeworld.com.cn/viewthread.php?tid=307239>
- 👉 8、LaunchPad上板载的仿真器支持其他系列的430吗?
<http://bbs.eeworld.com.cn/viewthread.php?tid=303148>
- 👉 9、LaunchPad能不能仿真149?
<http://bbs.eeworld.com.cn/viewthread.php?tid=305648>
- 👉 10、MSP430 LaunchPad支持那些20Pin的IC?
<http://bbs.eeworld.com.cn/viewthread.php?tid=305692>
- 👉 11、LaunchPad仿真器升级方法
<http://bbs.eeworld.com.cn/viewthread.php?tid=303567>

► 附录三:编委信息与后记

《玩转——TI MSP 430 LAUNCHPAD》通过对TI教室MSP430 Launchpad 视频培训教程的整理，同时结合了EEWORLD社区资深工程师对TI网管资料的整合以及一手的评测报告。帮助更多工程师更快完成设计。

在此特别感谢：

► EEWORLD社区 (<http://bbs.eeworld.com.cn/forum-192-1.html>) 坛友对我们活动的支持与关注，产生了大量新鲜、一手的应用经验。

TI公司的大力支持

► 希望《玩转——TI MSP 430 LAUNCHPAD》，能够为电子工程师设计工作提速！

EEWORLD社区

2012.10.21

► 附录四:版权说明

- 1、《玩转——TI MSP 430 LAUNCHPAD》著作权属TI和EEWORLD共同所拥有；
- 2、本着开源思想，我们授权任何对TI MSP 430 LAUNCHPAD有兴趣的工程师免费下载、复制、传播该书；
- 3、如用于商业用途须经EEWORLD书面同意。

► 附录五:EE团优惠券



👉 <http://item.taobao.com/item.htm?spm=0.0.0.0.J1AWbQ&id=18477696550>