MSP-EXP430F5529 开发板实验指导书 (草稿)

前言

MSP430F5529 是最新一代的具有集成 USB 的超低功耗单片机，可以应用于能量收集、无线传感以及自动抄表等场合，是最低工作功耗的单片机之一。MSP430F5529 开发板(MSP-EXP430F5529)是 MSP430F5529 单片机的开发平台，由电源选择开关、RF 射频接口、microSD Card 插槽、MSP430F5529 芯片及引出引脚、USB 接口、JTAG 仿真接口、齿轮电位计、电容触摸按键、LED、按钮、EZ-FET 内置仿真器、102x64 点阵 LCD 和三坐标轴加速度计组成。该开发板将 I/O 引脚接出来，方便用户进行实验操作，既可用于科研开发，又适合实验教学、课程设计、毕业设计等，为广大高校师生提供了良好的实验开发环境，同时也是广大电子爱好者学习、开发 MSP430 系列单片机的良好平台。

该实验指导书共有 9 章，分为两大部分。第一部分为第 1 章至第 2 章，介绍开发板的硬件和软件资源，以及 CCSv5.1 的安装和使用。第二部分为第 3 章至第 9 章，介绍 7 个实验。按原开发板的板载程序，可以做 6 个实验：时钟实验、游戏实验、功耗测试实验、应用程序演示实验、SD 卡内存读取实验和开发板设置实验。我们以各主要模块应用为线索，对这 6 个实验进行重新整合分类，形成了 7 个实验：时钟实验、触摸按键应用实验、加速度计应用实验、USB 通讯实验、SD 卡应用实验、功耗测试实验和综合实验，并将原来的 1 个软件工程，整理成 7 个独立的软件工程，调试通过，便于读者实验。

此外，与该实验指导书配套的还有 PPT 和实验视频材料，以便读者更好地利用开发板进行学习和实验。

该实验指导书、程序和相关材料是在 TI 公司大学计划的资助下，由合肥工业大学电气与自动化工程学院 DSP 实验室任保宏编写，由徐科军审阅。在编写过程中，得到 TI 大学计划黄争经理的指导，以及合肥工业大学电气与自动化工程学院 DSP 实验室胡小玲和邵春莉等的帮助。在此，表示衷心的感谢。由于时间和水平有限，书中可能存在错误和不妥之处，敬请广大读者批评指正。

合肥工业大学电气与自动化工程学院 DSP 实验室
2012 年 8 月

目录

第一部分：实验设备软硬件安装及说明

第一章 MSP-EXP430F5529 开发板概述

1.1 MSP430F5529 微控制器特性

◆低工作电压：1.8V 到 3.6V；
◆超低功耗：
--活动模式（AM）：所有系统时钟活动
290 μA/MHz 在 8MHz，3.0V，Flash Program
150 μA/MHz 在 8MHz，3.0V，RAM Program
--待机模式(LPM3)：
实时时钟、看门狗、电源监控、RAM 数据保持、快速唤醒：
1.9μA 在 2.2V，2.1μA 在 3.0V（典型）
低功耗振荡器、通用计数器、看门狗、电源监控、RAM 数据保持、快速唤醒：
1.4 μA 在 3.0V（典型）
--关闭模式（LPM4）：
RAM 数据保持，电源监控，快速唤醒：1.1μA 在 3.0V（典型）
--关断模式（LPM4.5）：0.18μA 在 3.0V（典型）
◆从待机模式下唤醒时间在 3.5μs 内（典型）；
◆16 位 RISC 结构，可拓展内存，高达 25-MHZ 的系统时钟；
◆灵活的电源管理系统：
--核心供电电压可编程调节的内置 LDO
--电源电压监控、监测及掉电检测
◆UCS 统一时钟系统：
--频率稳定的 FLL 控制回路
--低功率或低频率内置时钟源（VLO）
--修整后的低频内置参考源（REFO）
--32KHZ 低频晶振（XT1）
--高达 32MHZ 高频晶振 (XT2)
◆具有五个捕获/比较寄存器的 16 位定时器 TA0，Timer_A；
◆具有三个捕获/比较寄存器的 16 位定时器 TA1，Timer_A；
◆具有三个捕获/比较寄存器的 16 位定时器 TA2，Timer_A；
◆具有七个捕获/比较映射寄存器的 16 位定时器 TB0，Timer_B；
◆两个通用串行通讯接口：
--USCI_A0 和 USCI_A1，每个支持：增强 UART、IrDA、同步 SPI
--USCI_B0 和 USCI_B1，每个支持：I^2C、同步 SPI
◆全速 USB：
--集成 USB-PHY
--集成 3.3V/1.8V USB 电源系统
--集成 USB-PLL

--8 输入，8 输出端点

◆具有内部基准电压，采样和保持及自动扫描功能的 12 位 ADC(MSP430F552X 系列仅有)；

◆比较器；

◆支持 32 位运算的硬件乘法器；

◆串行系统编程，无需添加外部编程电压；

◆三通道内部 DMA；

◆具有实时时钟功能的基本定时器。

1.2 MSP430F5529 引脚图及结构框图

MSP430F5529 的引脚图如图 1.1 所示，结构框图如图 1.2 所示。

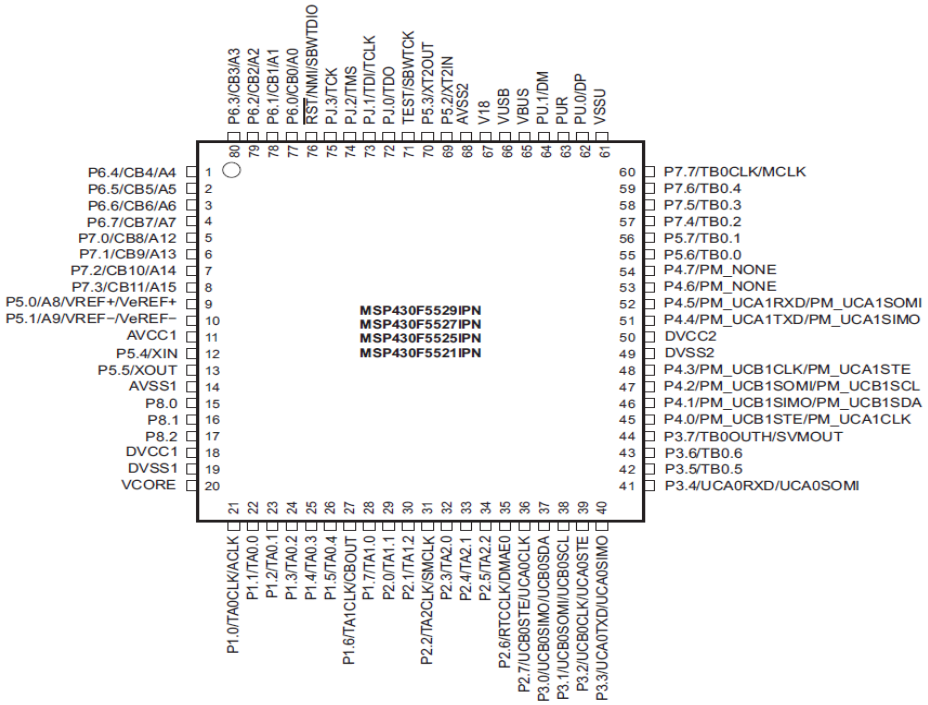


图 1.1 MSP430F5529 引脚图

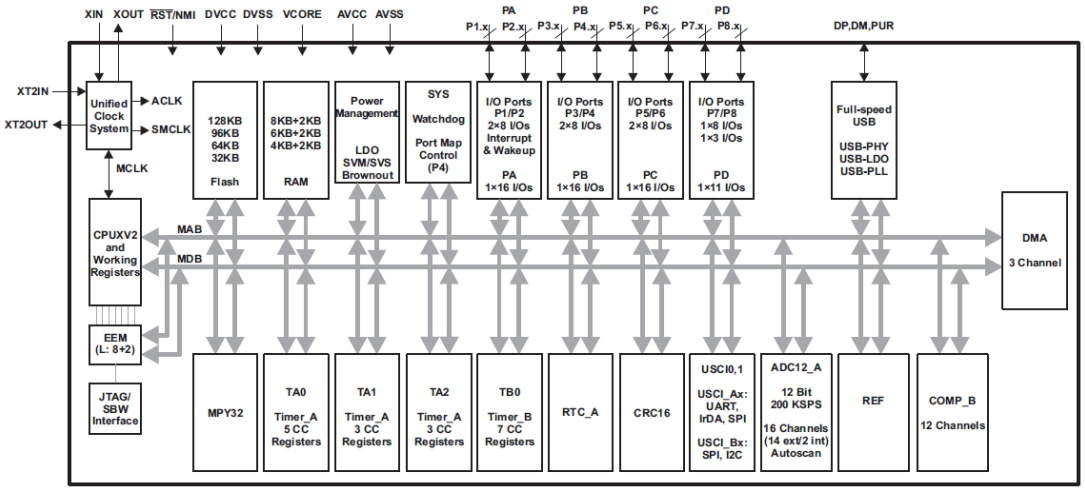


图 1.2 MSP430F5529 结构框图

注：其引脚具体功能请参考 MSP430F5529 数据手册

1.3 MSP-EXP430F5529 开发板硬件及软件资源概述

1.3.1 MSP-EXP430F5529 开发板硬件资源概述

MSP430F5529 开发板(MSP-EXP430F5529)是 MSP430F5529 器件的开发平台，为最新一代的具有集成 USB 的 MSP430 器件。该开发板与 CC2520EMK 等众多 TI 低功耗射频无线评估模块兼容。开发板能帮助设计者快速使用 F5529 MCU 进行学习和开发，其中 F5529 MCU 为能量收集、无线传感以及自动抄表基础设施 (AMI) 等应用提供了业界最低工作功耗的集成 USB、更大的内存和领先的集成技术。

MSP430F5529 开发板的结构组成如图 1.3 所示。我们从左上角开始，按顺时针方向介绍：MSP-EXP430F5529 集成了电源选择开关（4 种电源选择方式）、电池或外部电源接口、1 个 RF 射频接口、Micro SD Card 插槽（附 1G 内存卡）、MSP430F5529 引脚接口、5529USB 接口、JTAG 仿真接口、1 个齿轮电位计、5 块电容触摸按键、9 个 LED、4 个按钮（2 个用户配置按钮、1 个复位按钮、1 个 BSL 按钮）、eZ-FET 内置仿真器、1 块 102x64 点阵 LCD、1 个三坐标轴加速度计以及 MSP430F5529 芯片。

该开发板将 F5529 部分引脚接出来，方便用户进行实验操作，既适合科研开发，又适合实验教学、课程设计、毕业设计等，为广大高校师生提供了良好的实验开发环境，同时也是广大电子爱好者学习、开发 MSP430 系列单片机的良好平台。

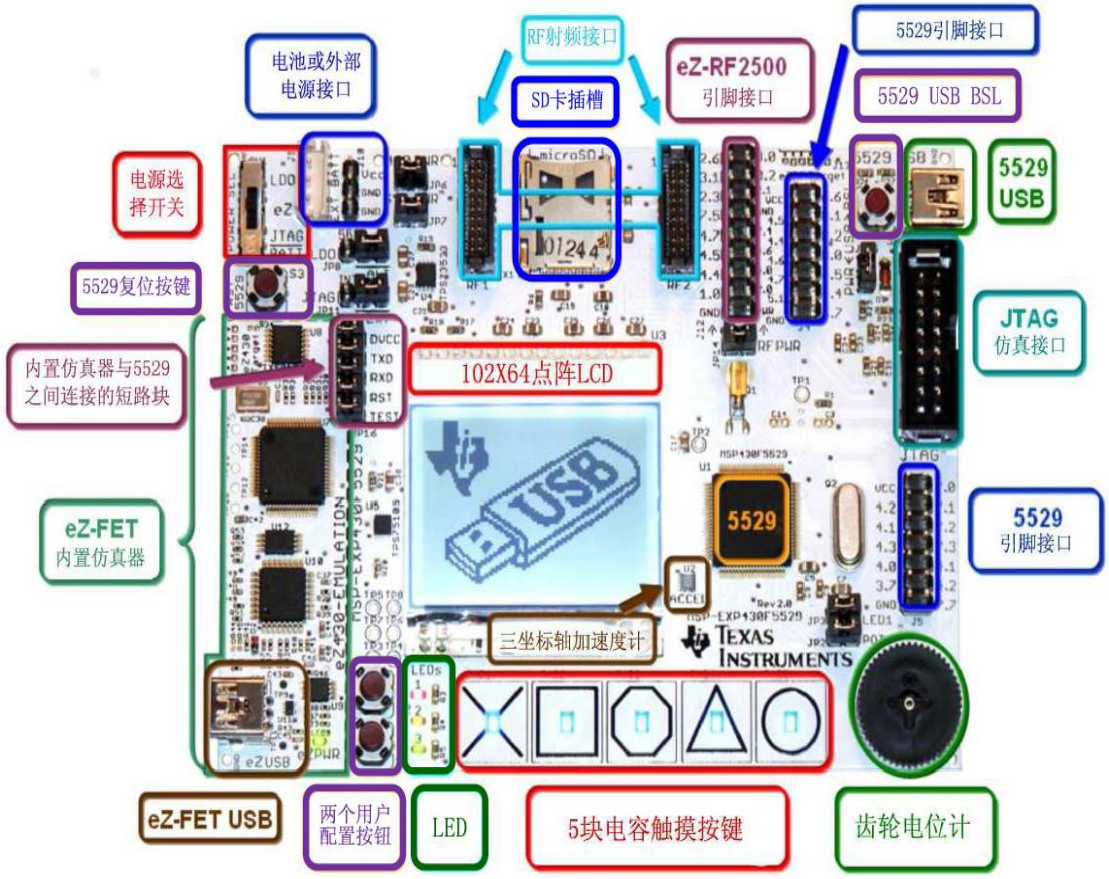


图 1.3 MSP-EXP430F5529 实物硬件资源图

附：具体的硬件电路图可通过以下途径获得：http://www.ti.com/lit/zip/slar055。

1.3.2 MSP-EXP430F5529 开发板实验程序资源概述

MSP-EXP430F5529 开发板的实验程序资源包含在名为 MSP-EXP430F5529 LAB CODE 的文件夹内，表 1.1 为各实验文件夹（LABx）内资源描述列表。

表 1.1 实验文件夹内资源描述列表

文件名称	描述
CTS	包含触摸按键应用程序资源库
Drivers	包含 USB 通信实验硬件驱动
F5xx_F6xx_Core_Lib	包含部分 MSP430 片内外设程序资源库
FatFs	包含开源的 FATFS 系统文件
MSP-EXP430F5529_HAL	包含 MSP-EXP430F5529 开发板硬件模块程序资源库
USB	包含 USB 应用程序资源库
UserExperienceDemo	包含开发板示例程序代码
LABxmain.c(x=1～7)	包含各实验主函数
labx.h/.c(x=1～7)	包含各实验菜单函数及实验程序

MSP-EXP430F5529 开发板各实验主函数（LABxmain.c 中）流程图如图 1.4 所示。

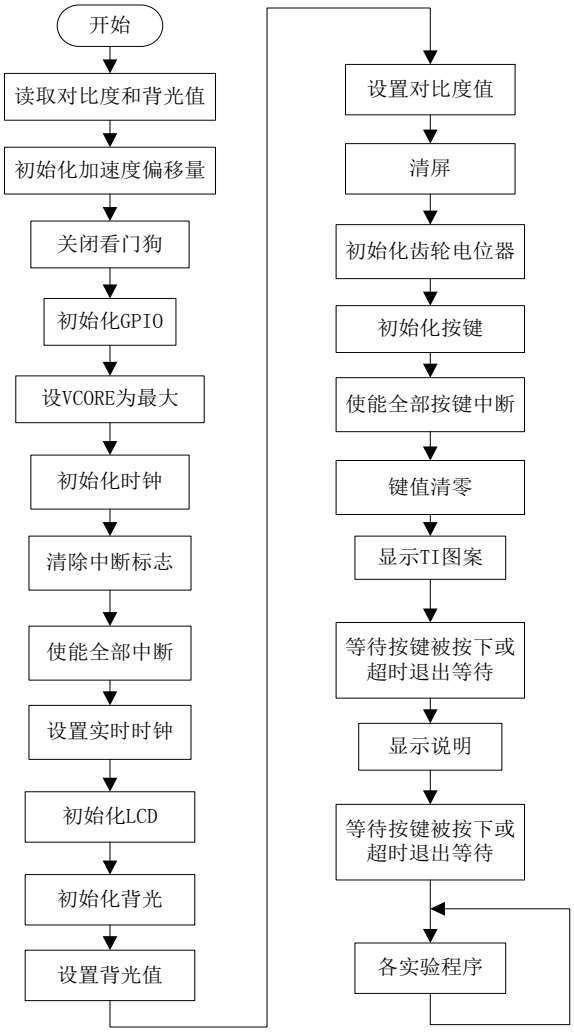


图 1.4 各实验程序主函数流程图

1.4 MSP-EXP430F5529 开发板供电方案分析

1.4.1 MSP-EXP430F5529 开发板供电方案实物分析

MSP-EXP430F5529 开发板有四种供电方案，在实物硬件连接上如图 1.5 所示：

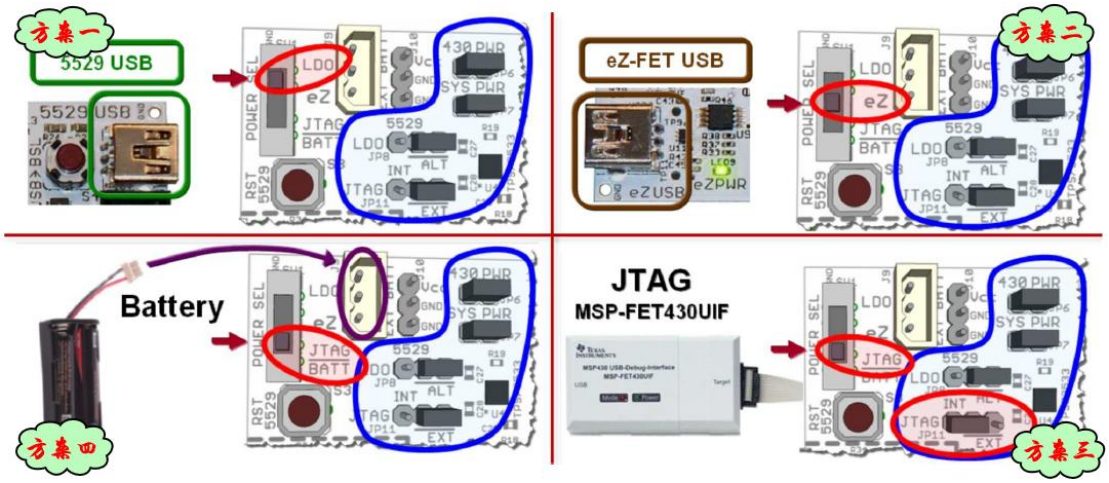


图 1.5 开发板供电方案实物连接图

供电方案设置：

方案一：开发板由 5529USB（开发板右上角）供电，将电源拨码开关打到 LDO 档位，电源选择短路块保持原始位置不变。

方案二：开发板由 EZ-FET USB（开发板左下角）供电，将电源拨码开关打到 eZ 档位，电源选择短路块保持原始位置不变。

方案三：开发板由 JTAG 仿真接口供电，将电源拨码开关打到 JTAG/BATT 档位，注意：将 JP11 短路块由左边两个端口短接变为右边两个端口短接，其他电源选择短路块保持原始位置不变。

方案四：开发板由两节电池供电，将电源拨码开关打到 JTAG/BATT 档位，电源选择短路块保持原始位置不变，并将电池连线接口插入电池或外部电源选择插槽中。

1.4.2 MSP-EXP430F5529 开发板供电方案电路分析

MSP-EXP430F5529 开发板四种供电方案电路如图 1.6 所示：

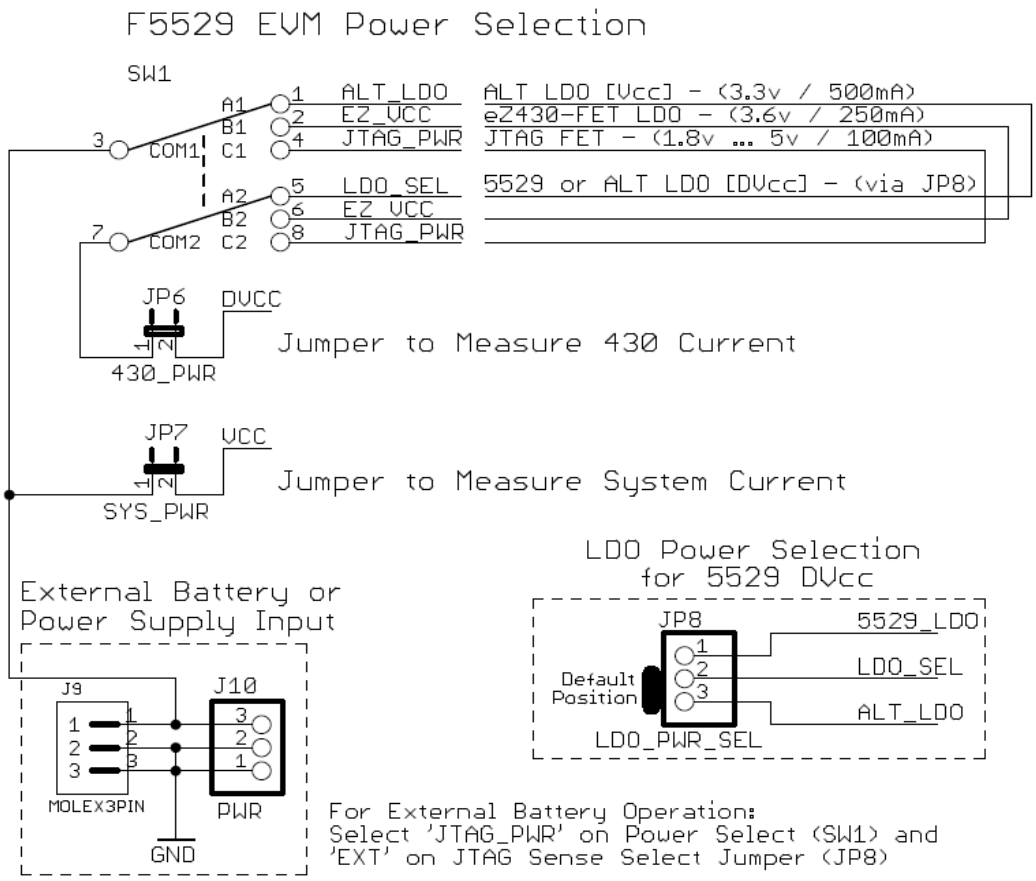


图 1.6 MSP-EXP430F5529 开发板电源选择电路图

图中，SW1 代表电源选择拨码开关；DVCC 电源为 MSP430F5529 微控制器供电，测试该路电流，即可得到 MSP430F5529 微控制器的功耗（利用短路块 JP6，后面将会介绍）；VCC 电源为除 MSP430F5529 微控制器外其他模块供电，测试该路电流，即可得到系统的功耗。

（1）当采用方案一供电时，即将拨码开关拨至最上面的档位。该方案供电来自于右上角 F5529USB 接口，供电电压为 3.3V，供电电流为 500mA。由图 1.6 中 JP8 短路块可知 LDO_SEL 和 ALT_LDO 短路，由图 1.7 可见，ALT_LDO 为由 5529_VBUS 经 TPS73533 芯片电压转换而来，又由图 1.8 可知，5529_VBUS 由右上角 Mini-USB 传输线上电源线所得，电压值为 5V。

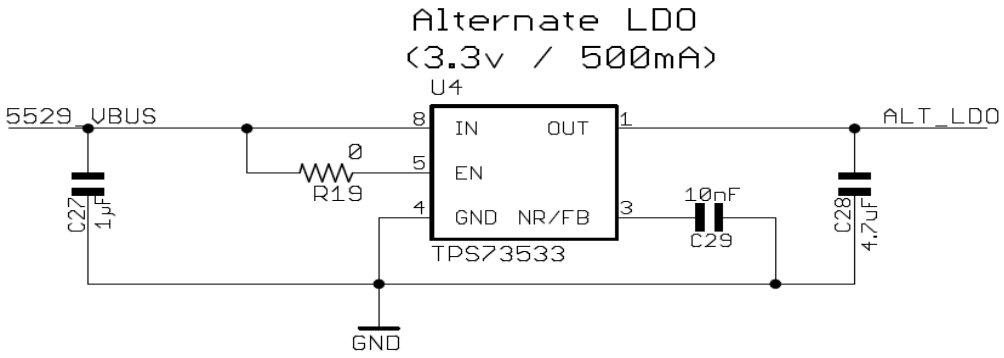


图 1.7 LDO 电平转换电路

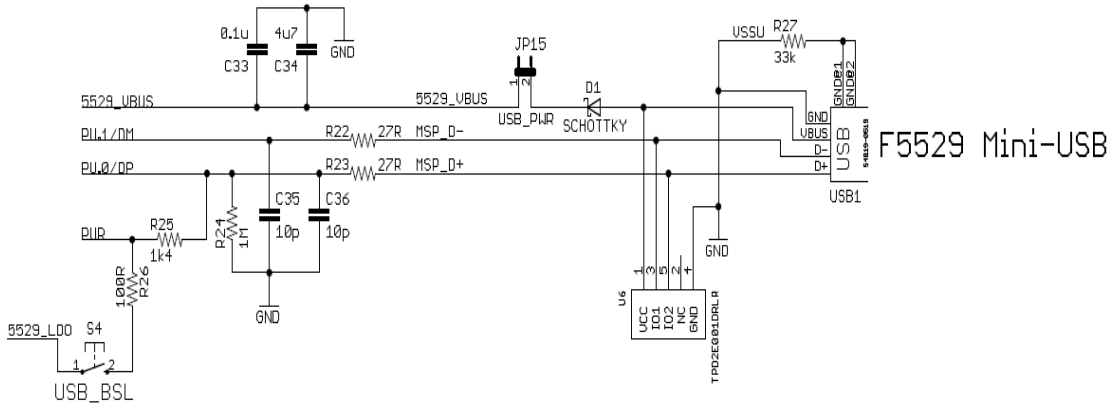
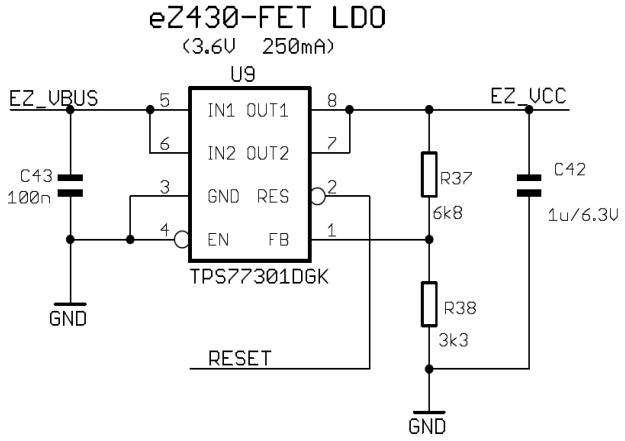


图 1.8 F5529 Mini-USB 电路

（2）当采用方案二供电时，即将拨码开关拨至中间的档位。该方案供电来自于左下角 EZ430-FET USB 接口，供电电压为 3.6V，供电电流为 250mA。由图 1.9 可见，该方案的供电电源 EZ_VCC 由 EZ_VBUS 经 TPS77301DGK 芯片电平转换而来，又由图 1.10 可见，EZ_VBUS 由左下角 Mini-USB 传输线上电源线所得，其电压值也为 5V。



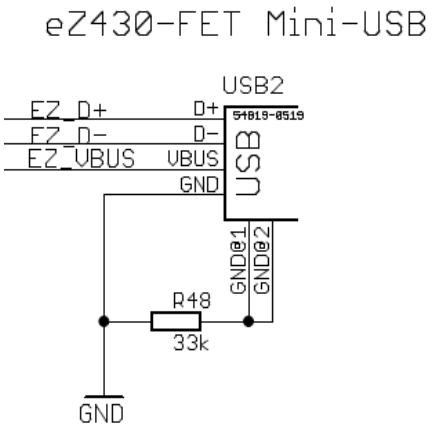


图 1.9 EZ430-FET LDO 电平转换电路　　图 1.10 EZ430-FET Mini-USB 接口电路

（3）当采用方案三供电时，即将拨码开关拨至最下面的档位。该方案的供电来自于 JTAG 仿真接口，供电电压为 1.8V～5V 之间，供电电流为 100mA。由图 1.11 可知该方案的供电电源 JTAG_PWR 来自于 JTAG 接口电路上的电源引脚。

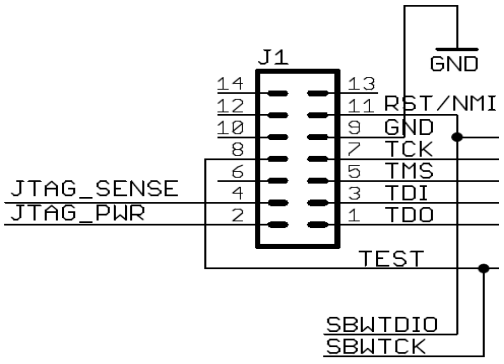


图 1.11 JTAG 接口电路

（4）当采用方案四供电时，也将拨码开关拨至最下面的档位，但是，该方案的供电电源来自于外部电池或其他的外部电源输入。由图 1.6 中可见，若将两节干电池的连线插入 J9 的插槽中，即可为整个系统供电；或者利用 J10 的插针引入外部适当电源，也可为整个系统供电。

开发板电能供应如图 1.12 所示。

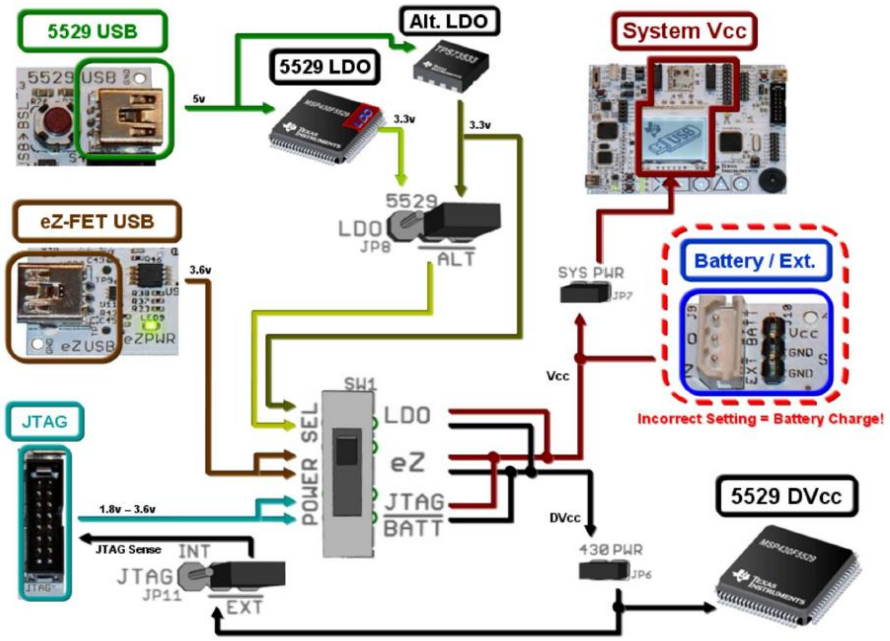


图 1.12 电能供应图

1.5 MSP-EXP430F5529 开发板仿真方案分析

MSP-EXP430F5529 开发板具有三种仿真方案，前两种方案介绍如图 1.13 所示：

方案一：将 Mini-USB 线与 MSP-EXP430F5529 开发板左下角 USB 接口连接,并将短路块和电源拨码开关正确配置，采用内置仿真器 EZ-FET 进行程序下载仿真。该方案无需安装仿真器驱动，程序可直接下载调试。

方案二:将 MSP-FET430UIF 与 JTAG 接口连接,并将短路块和电源拨码开关正确配置，采用外置仿真器进行程序下载仿真。该方案需安装仿真器驱动，才可进行程序的下载调试。

方案三：采用 MSP430 BSL 进行仿真，仿真方法可以参考：USB Field Firmware Updates on MSP430™ MCUs。

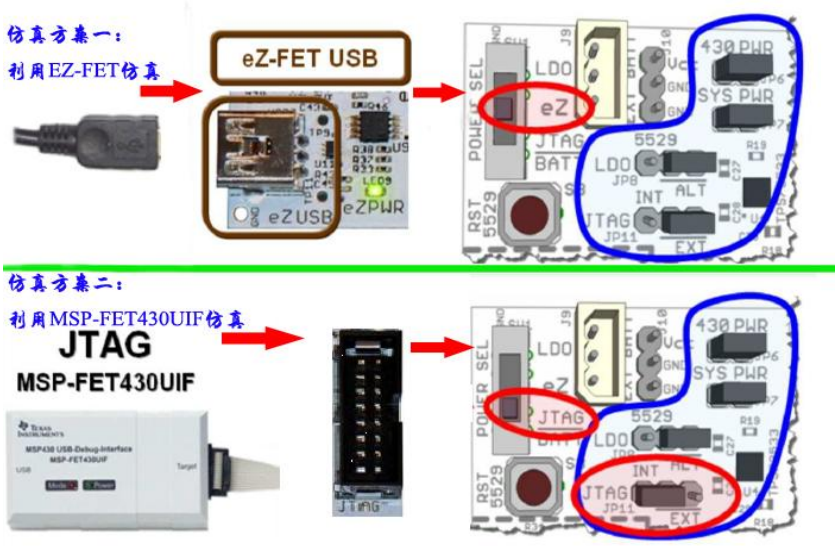


图 1.13 开发板仿真方案实物连接图

1.6 MSP-EXP430F5529 开发板短路块设置及功能介绍

短路块	连接短路块的功能	去除短路块的功能
JP2-POT	连接齿轮电位计和 P8.0	断开齿轮电位计和 P8.0 的连接
JP3-LED1	连接 LED1 和 P1.0	断开 LED1 和 P8.0 的连接
JP6-430 PWR	对 MSP430F5529 提供电源，可以用来测试单片机功耗，注意：430 PWR 通孔与 JP6 引脚相连。	断开 MSP430F5529 的电源
JP7-SYS PWR	对开发板提供电源，可以用来测试开发板系统功耗。	断开开发板电源
JP8-LDO	仅在 5529 USB 提供电源时有用。 ALT（默认）：连接 LDO（TPS73533）与 MSP430 VCC； INT：连接 F5529LDO 与 MSP430 VCC	5529 USB 无法供电
JP11-JTAG	仅在 JTAG 供电时有用 EXT（默认）：JTAG 对系统供电； INT：JTAG 不对系统供电	JTAG 无法对系统供电
JP14-RF PWR	连接 VCC 和 RF 接口:J12，J13 和 RF2	RF 接口无电源供应
JP15-USB PWR	将 USB 5V 电源供给 MSP430F5529 和 LDO（TPS73533）	USB 5V 电源无法供给系统
JP16-ez-FET	DVCC：连接 MSP430 VCC 与 Ez-FET； TXD/RXD：在 F5529 与 Ez-FET 之间连接 UART； RST/TEST：在 F5529 与 Ez-FET 之间连接 JTAG。	无法连接 F5529 与 Ez-FET

1.7 MSP-EXP430F5529 开发板各接口引脚介绍

（1）J4 接口引脚连接列表

引脚描述	引脚（左）	引脚（右）	引脚描述
Vcc	VCC	P6.6	CB6/A6
UCA1RXD / UCA1SOMI	P4.5	P8.1	GPIO – LED2
UCA1TXD / UCA1SIMO	P4.4	P8.2	GPIO – LED3
GPIO	P4.6	P8.0	GPIO – 齿轮电位计
GPIO	P4.7	P4.5	UCA1RXD / UCA1SOMI
A9 / VREF- / VeREF-	P5.1	P4.4	UCA1TXD / UCA1SIMO
GND	GND	P6.7	CB7 / A7

（2）J5 接口引脚连接列表

引脚说明	引脚（左）	引脚（右）	引脚说明
VCC	VCC	P7.0	CB8 / A12
UCB1SOMI / UCB1SCL - SD	P4.2	P7.1	CB9 / A13
UCB1SIMO / UCB1SDA - LCD/SD	P4.1	P7.2	CB10 / A14
UCB1CLK / UCA1STE - LCD/SD	P4.3	P7.3	CB11 / A15
UCB1STE / UCA1CLK - RF	P4.0	P4.1	UCB1SIMO / UCB1SDA - LCD/SD
TB0OUTH / SVMOUT - SD	P3.7	P4.2	UCB1SIMO / UCB1SDA - LCD/SD
GND	GND	P7.7	TB0CLK / MCLK

（3） J12 接口引脚连接列表

引脚说明	引脚（左）	引脚（右）	引脚说明
(RF_STE)	P2.6	P3.0	(RF_SIMO)
(RF_SOMI)	P3.1	P3.2	(RF_SPI_CLK)
TA2.0	P2.3	P2.1	TA1.2
TB0.3	P7.5	GND	GND
GPIO	P4.7	P2.4	TA2.1
(RXD)	P4.5	P4.6	GPIO
(TXD)	P4.4	P4.0	UCx1xx
(LED1)	P1.0	P2.0	TA1.1
GND	GND	RF_PWR	RF_PWR

1.8 MSP-EXP430F5529 开发板资源下载途径

（1）MSP-EXP430F5529 官方网站：
http://www.ti.com/tool/msp-exp430f5529&DCMP=MSP430&HQS=Other+OT+usbexp
（2）MSP-EXP430F5529 开发板用户指导手册：http://www.ti.com/lit/pdf/SLAU330
（3）MSP-EXP430F5529 开发板硬件电路图：http://www.ti.com/lit/zip/slar055
（4）MSP430x5xx/x6xx 用户指导：www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=slau208&fileType=pdf&track=no
（5）MSP430F552x 数据手册：www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=slas590&fileType=pdf&track=no
（6）MSP430F552X 例程：http://www.ti.com/lit/zip/slac300
（7）CCSv5 下载途径：http://processors.wiki.ti.com/index.php/Download_CCS
（8）USB 开发资源库下载途径：
http://www.ti.com/tool/msp430usbdevpack?DCMP=53xx663x&HQS=msp430usbdevpack-pr-tf
（9）电容触摸资源库下载途径：http://www.ti.com/tool/capsenselibrary#1

第二章 软件的安装与应用

CCS（Code Composer Studio）是 TI 公司研发的一款具有环境配置、源文件编辑、程序调试、跟踪和分析等功能的集成开发环境，能够帮助用户在一个软件环境下完成编辑、编译、链接、调试和数据分析等工作。CCSv5.1 为 CCS 软件的最新版本，功能更强大、性能更稳定、可用性更高，是 MSP430 软件开发的理想工具。

2.1 CCSv5.1 的安装

（1）运行下载的安装程序 ccs_setup_5.1.1.00031.exe，当运行到如图 2.1 处时，选择 Custom 选项，进入手动安装选择通道。

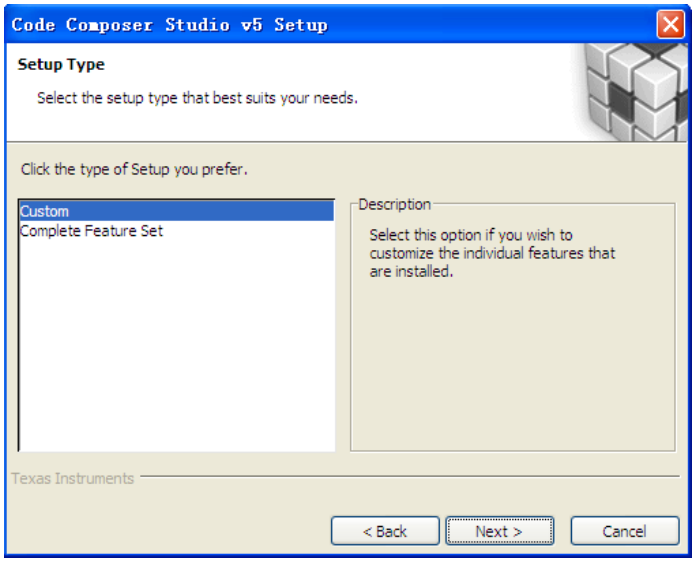


图 2.1 安装过程 1

（2）单击 Next 得到如图 2.2 所示的窗口，为了安装快捷，在此只选择支持 MSP430 Low Power MCUs 的选项。单击 Next，保持默认配置，继续安装。

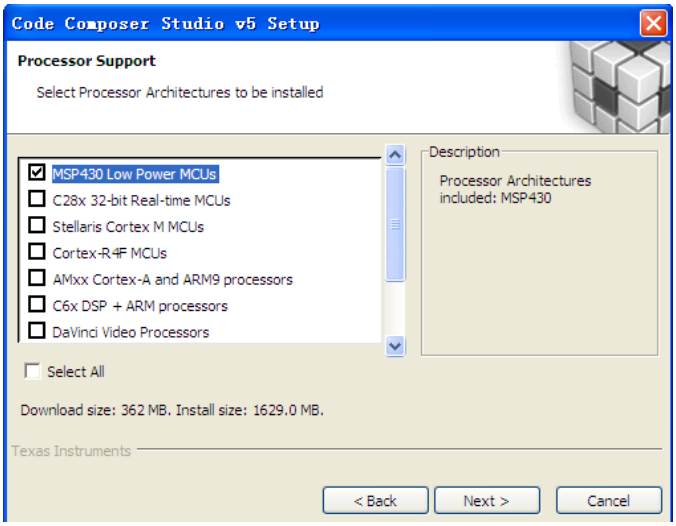


图 2.2 安装过程 2

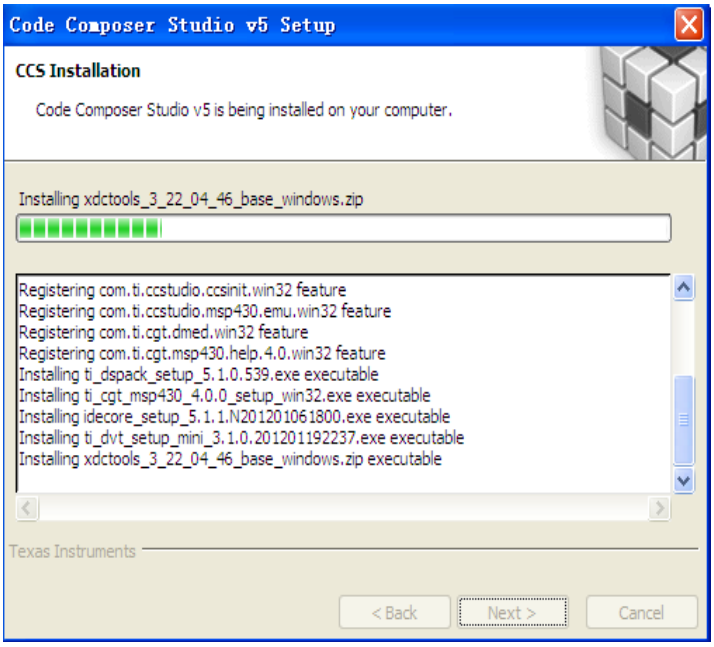


图 2.3 软件安装中

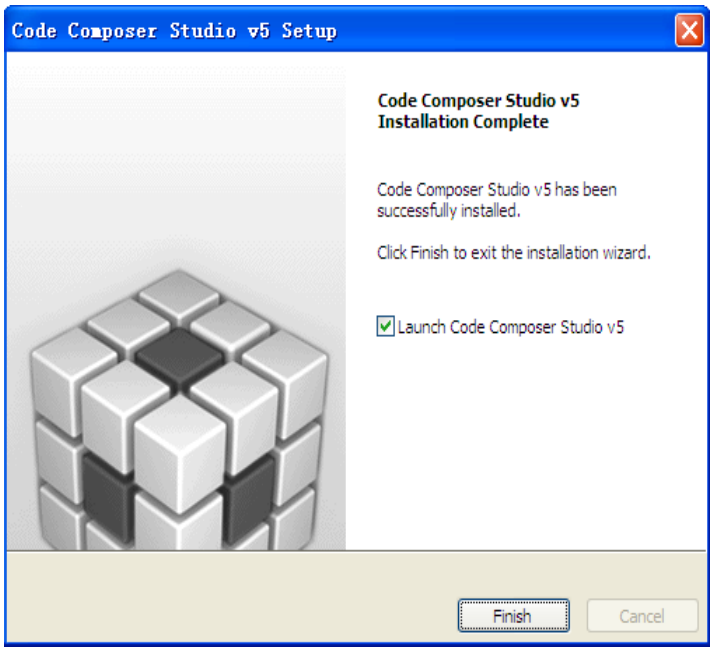


图 2.4 软件安装完成

（3）单击 Finish，将运行 CCS，弹出如图 2.5 所示窗口，打开“我的电脑”，在某一磁盘下，创建以下文件夹路径：-\MSP-EXP430F5529\Workspace，单击 Browse，将工作区间链接到所建文件夹，不勾选"Use this as the default and do not ask again"。

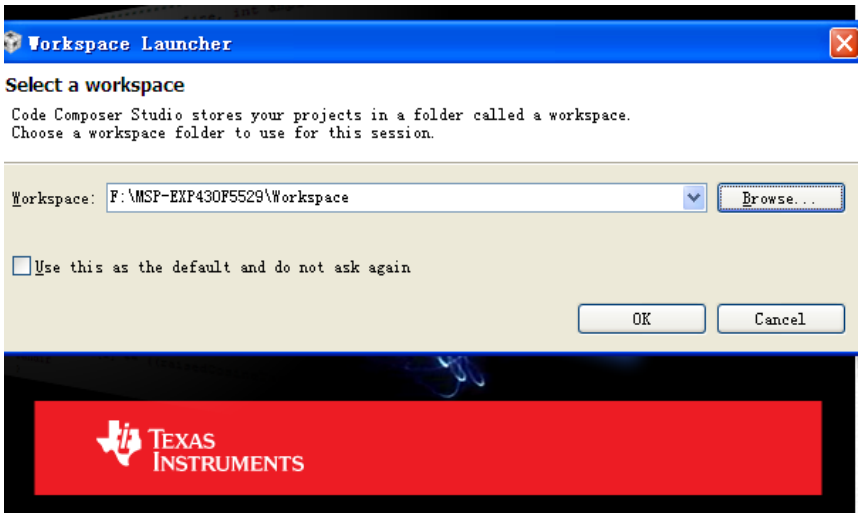


图 2.5　Workspace 选择窗口

（4）单击 OK，第一次运行 CCS 需进行软件许可的选择，如图 2.6 所示。

在此，选择 CODE SIZE LIMITED(MSP430)选项，在该选项下，对于 MSP430，CCS 免费开放 16KB 的程序空间；若您有软件许可，可以参考以下链接进行软件许可的认证：http://processors.wiki.ti.com/index.php/GSG:CCSv5_Running_for_the_first_time，单击 Finish 即可进入 CCSv5.1 软件开发集成环境，如图 2.7 所示。

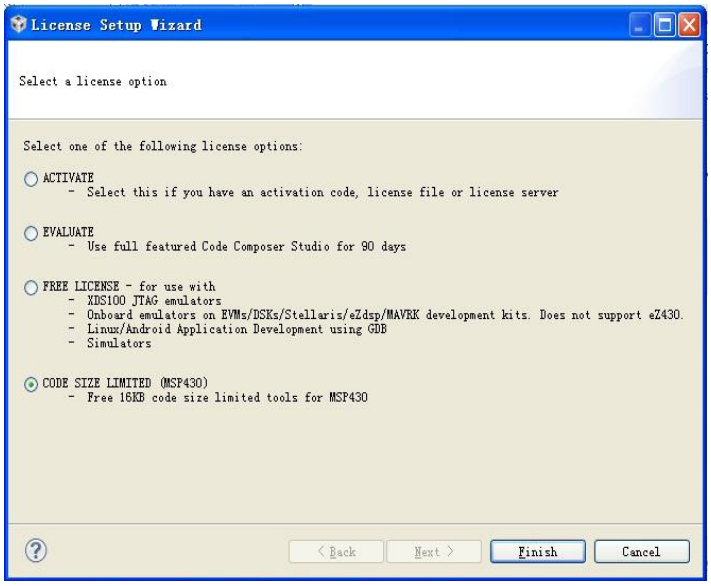


图 2.6 软件许可选择窗口

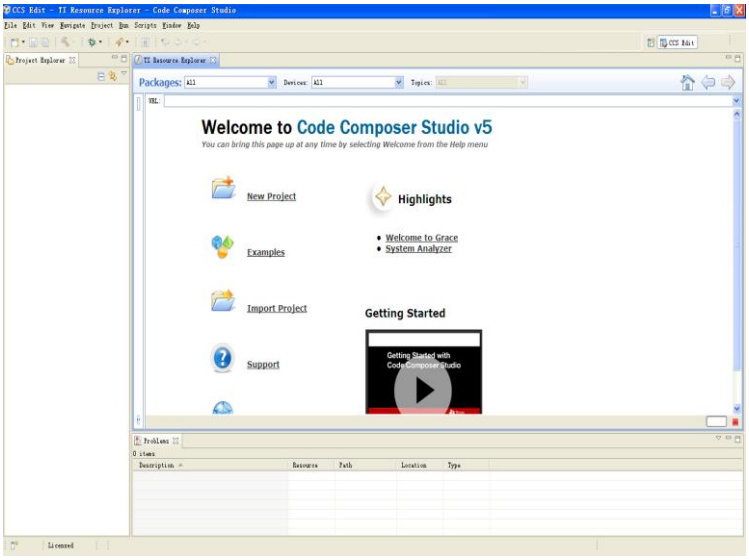


图 2.7 CCSv5 软件开发集成环境界面

2.2 利用 CCSv5.1 导入已有工程

（1）在此以实验一的工程为例进行讲解，首先打开 CCSv5.1 并确定工作区间：F\MSP-EXP430F5529\Workspace，选择 File-->Import 弹出图 2.8 对话框，展开 Code Composer Studio 选择 Existing CCS/CCE Eclipse Projects。

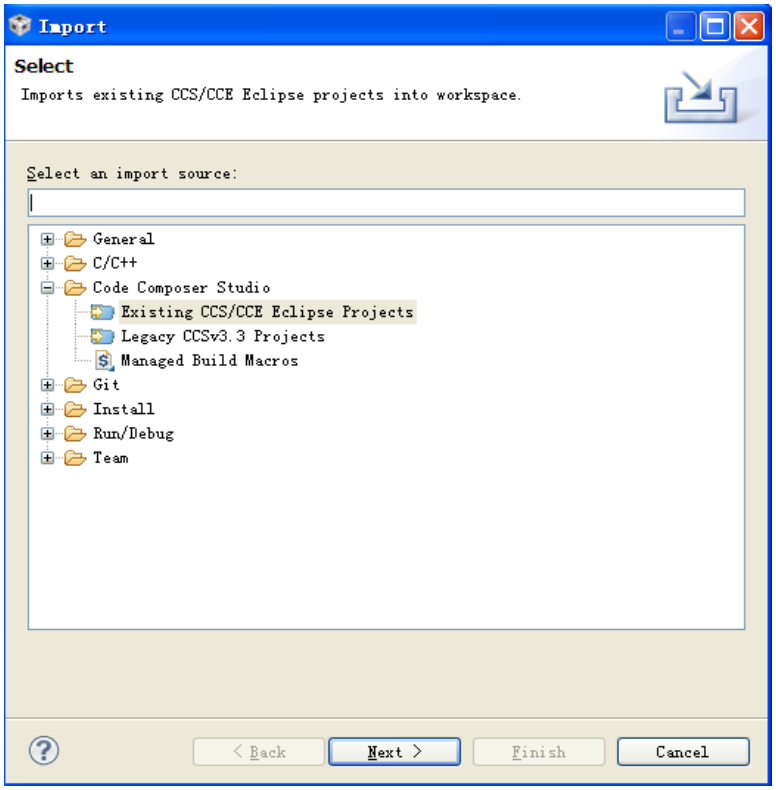


图 2.8 导入新的 CCSv5 工程文件

（2）单击 Next 得到图 2.9 对话框。

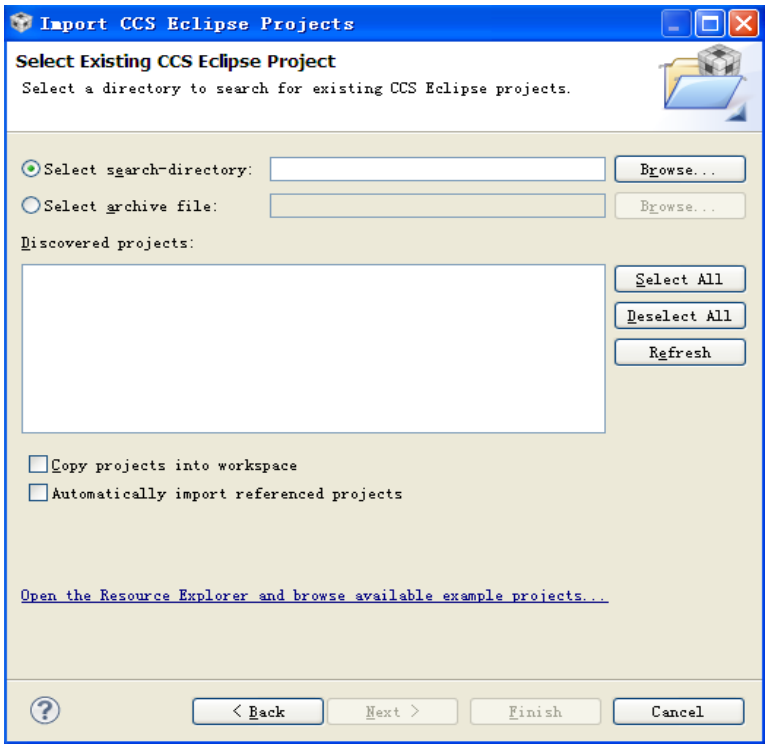


图 2.9 选择导入工程目录

（3）单击 Browse 选择需导入的工程所在目录，在此，我们选择：F\MSP-EXP430F5529\Workspace\MSP-EXP430F5529 LAB CODE\LAB1（需在此之前，将实验代码复制到工作区间下），得到图 2.10。

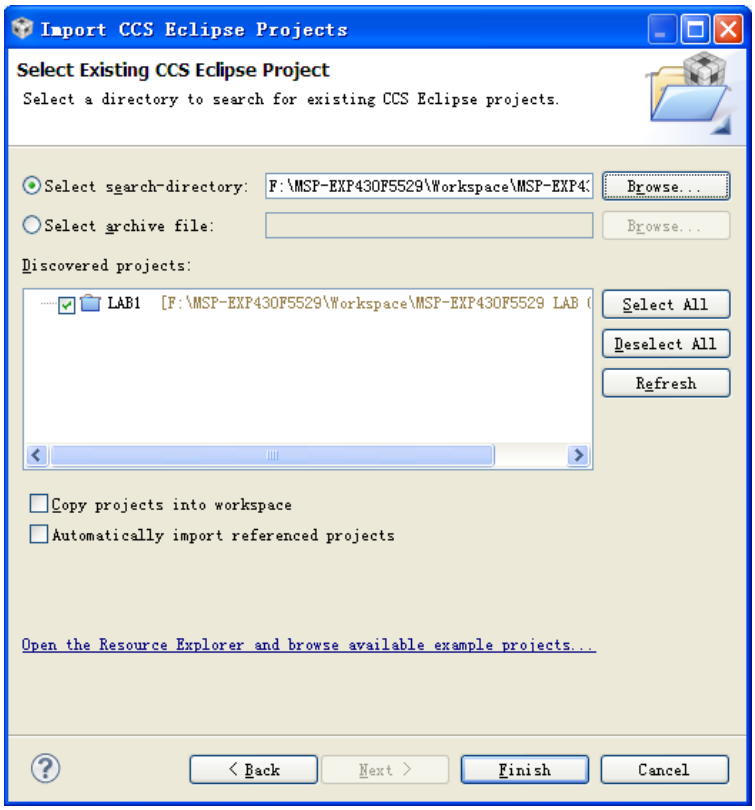


图 2.10 选择导入工程

（4）单击 Finish，即可完成既有工程的导入。

2.3 利用 CCSv5.1 新建工程

（1）首先打开 CCSv5.1 并确定工作区间，然后选择 File-->New-->CCS Project 弹出图 2.11 对话框。

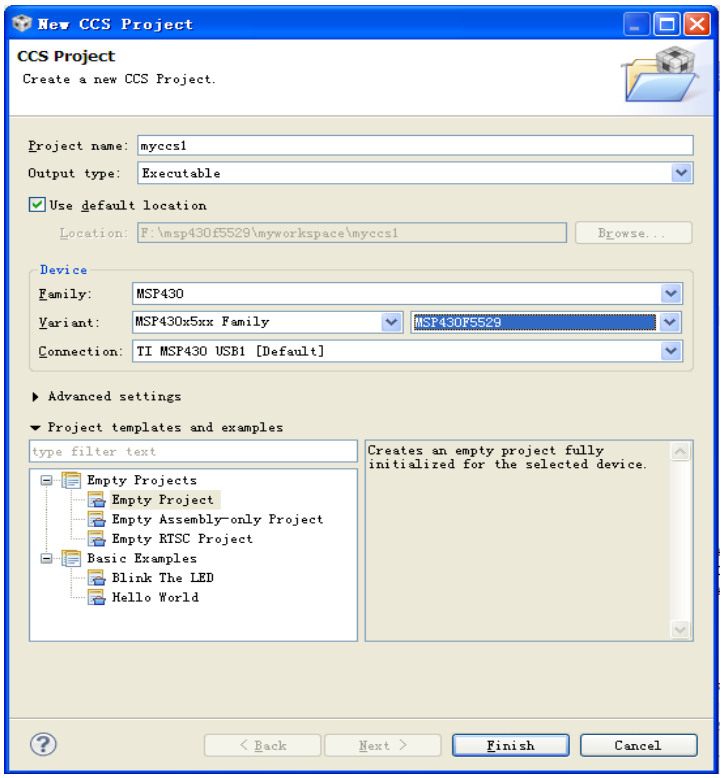


图 2.11 新建 CCS 工程对话框

（2）在 Project name 中输入新建工程的名称，在此输入 myccs1。

（3）在 Output type 中有两个选项：Executable 和 Static library，前者为构建一个完整的可执行程序，后者为静态库。在此保留：Executable。

（4）在 Device 部分选择器件的型号:在此 Family 选择 MSP430；Variant 选择 MSP430X5XX family，芯片选择 MSP430F5529；Connection 保持默认。

（5）选择空项目，然后单击 Finish 完成新项目的创建。

（6）创建的项目将显示在 Project Explorer 中，如图 2.12 所示。

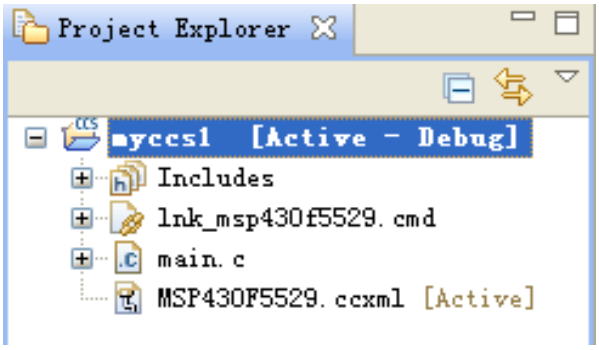


图 2.12 初步创建的新项目

特别提示：若要新建或导入已有.h 或.c 文件，步骤如下：

（7）新建.h 文件：在工程名上右键点击，选择 New-->Header File 得到图 2.13 对话框。

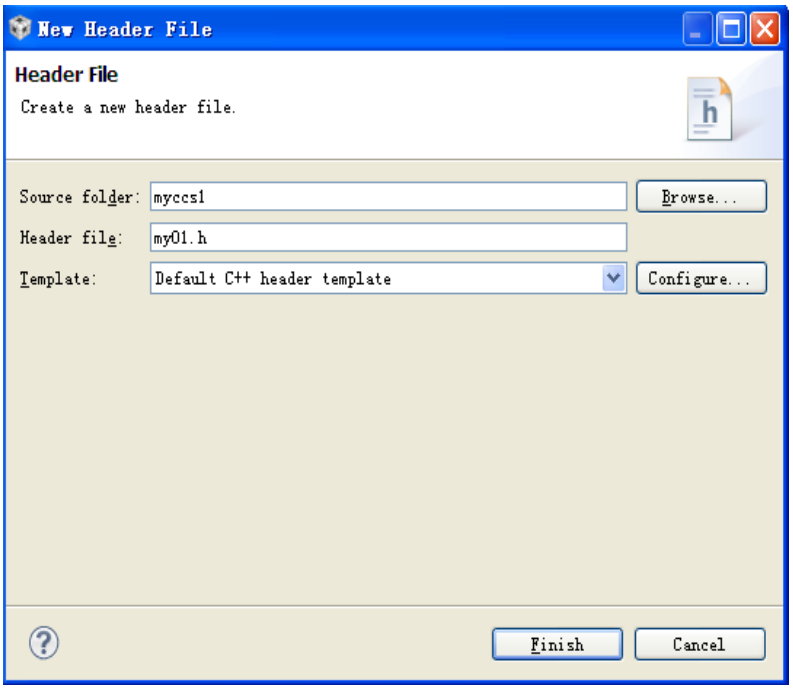


图 2.13 新建.h 文件对话框

在 Header file 中输入头文件的名称，注意必须以.h 结尾，在此输入 myo1.h。

（8）新建.c 文件：在工程名上右键单击，选择 New-->source file 得到图 2.14 对话框。

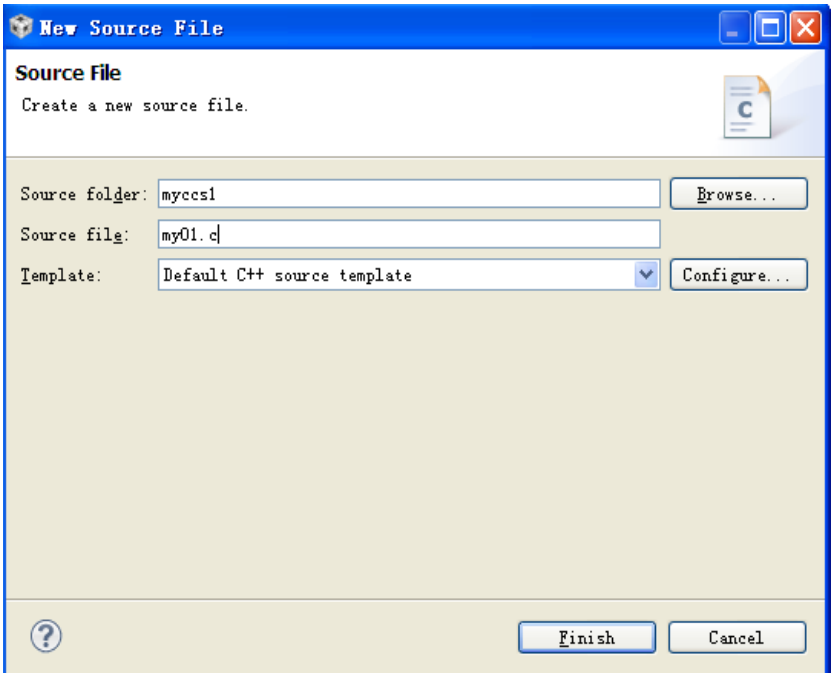


图 2.14 新建.c 文件对话框

在 Source file 中输入 c 文件的名称，注意必须以.c 结尾，在此输入 my01.c。

（9）导入已有.h 或.c 文件：在工程名上右键单击，选择 Add Files 得到如 2.15 对话框。

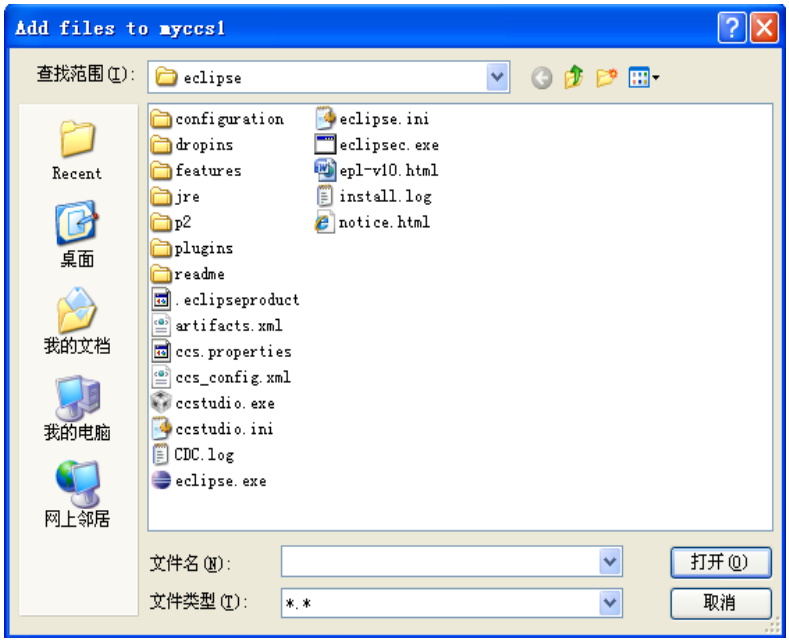


图 2.15 导入已有文件对话框

找到所需导入的文件位置，单击打开，得到图 2.16 对话框。

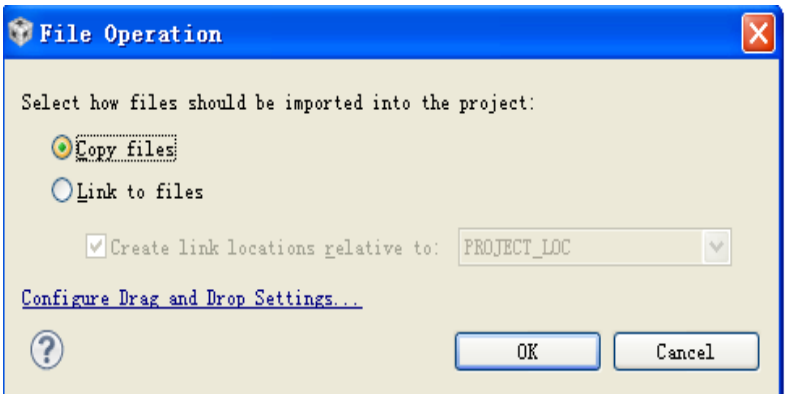


图 2.16 添加或连接现有文件

选择 Copy files，单击 OK，即可将已有文件导入到工程中。

若已用其它编程软件（例如 IAR），完成了整个工程的开发，该工程无法直接移植入 CCSv5，但可以通过在 CCSv5 中新建工程，并根据步骤（7）、（8）和（9）新建或导入已有.h 和.c 文件，从而完成整个工程的移植。

2.4 利用 CCSv5.1 调试工程

2.4.1 创建目标配置文件

（1）在开始调试之前，有必要确认目标配置文件是否已经创建并配置正确。在此以实验一为例进行讲解：首先导入实验一的工程，导入步骤请参考 2.2 节，如图 2.17 所示，**其中 MSP430F5529.ccxml 目标配置文件已经正确创建，即可以进行编译调试，无需重新创建；若目标配置文件未创建或创建错误，则需进行创建**。为了讲解目标配置文件创建过程，在此对 LAB1 的工程再次创建目标配置文件。

图 2.17 LAB1 工程浏览器

（2）创建目标配置文件步骤如下：右键单击项目名称，并选择 NEW --> Target Configuration File。

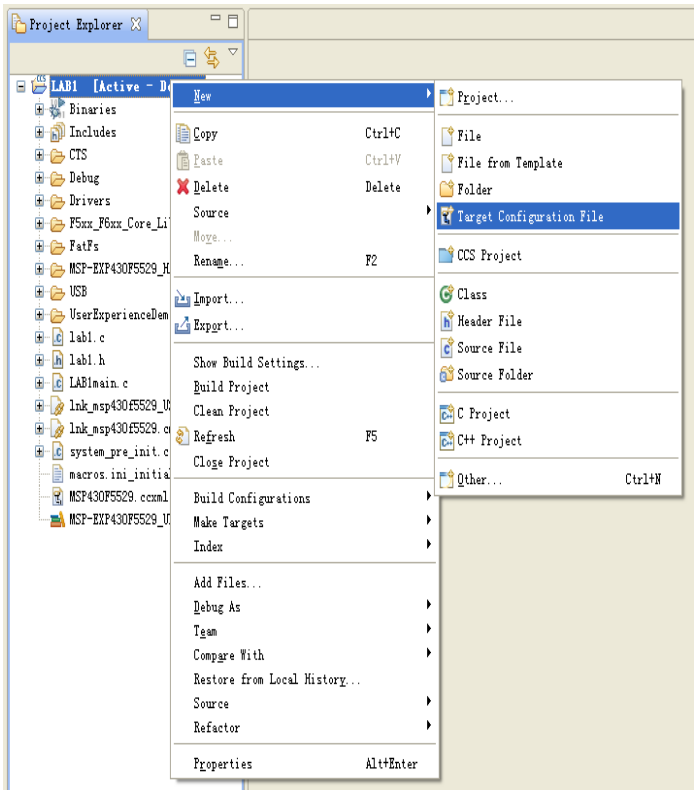


图 2.18 创建新的目标

（3）在 File name 中键入后缀为.ccxml 的配置文件名，由于创建 F5529 开发板的目标配置文件，因此，将配置文件命名为 MSP-EXP430F5529.ccxml，如图 2.19 所示。

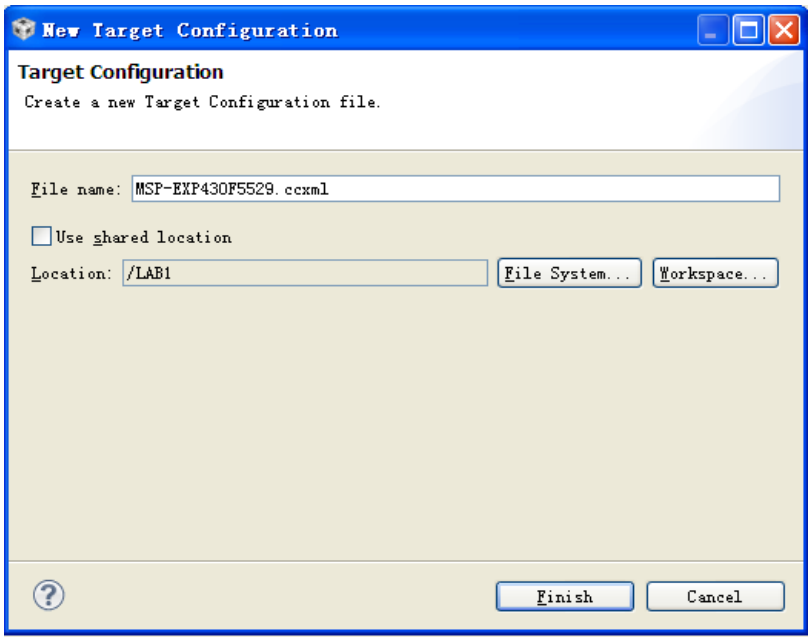


图 2.19 目标配置文件名

（4）单击 Finish，将打开目标配置编辑器，如图 2.20 所示。

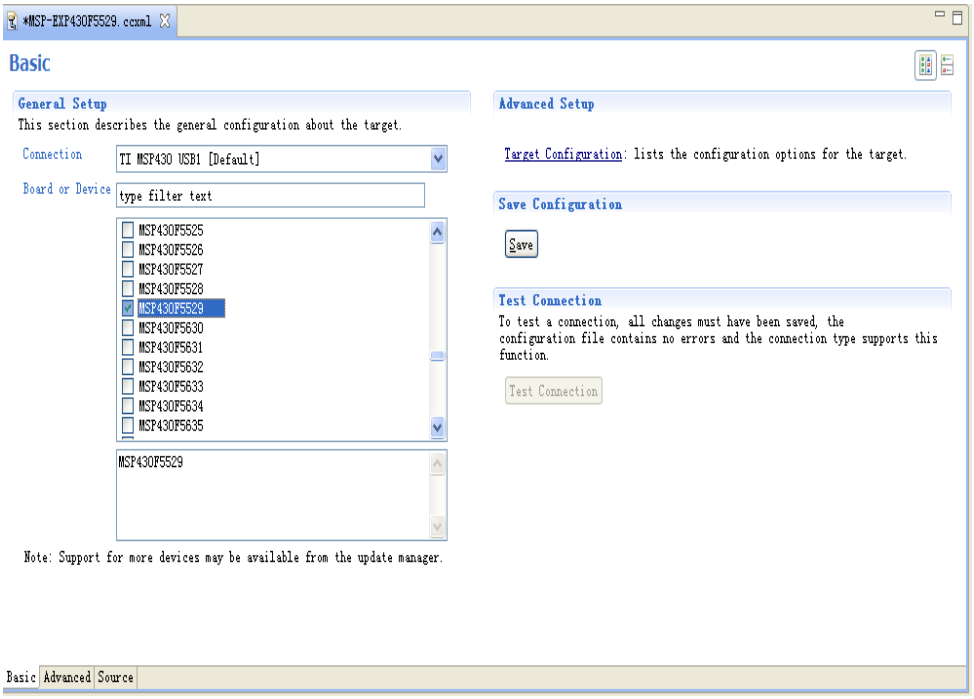


图 2.20 目标配置编辑器

（5）将 Connection 选项保持默认：TI MSP430 USB1 (Default)，在 Board or Device 菜单中选择单片机型号，在此选择 MSP430F5529。配置完成之后，单击 Save，配置将自动设为活动模式。如图 2.21 所示， 一个项目可以有多个目标配置，但只有一个目标配置在活动模式。要查看系统上所有现有目标配置，只需要去 View --> Target Configurations 查看。

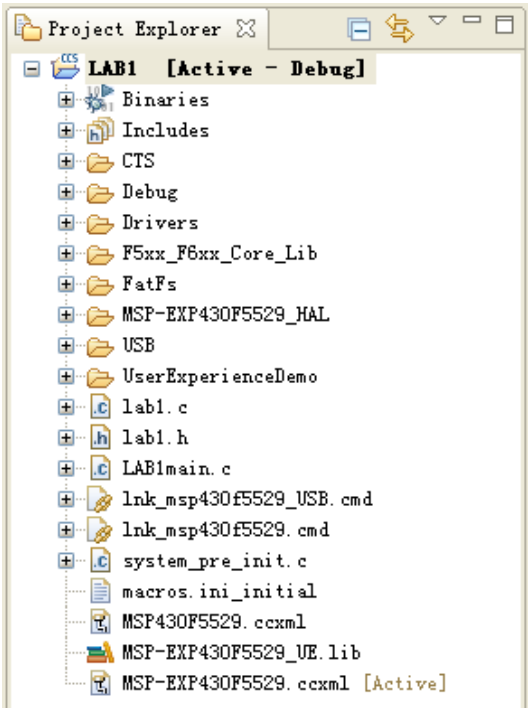


图 2.21 项目与配置后的目标文件

2.4.2 启动调试器

（1）首先将 LAB1 工程进行编译通过：选择 Project-->Build Project，调试目标工程，调试结果，如图 2.22 所示，表示编译没有错误产生，可以进行下载调试；如果程序有错误，将会在 Problems 窗口显示，根据显示的错误修改程序，并重新编译，直到无错误提示。

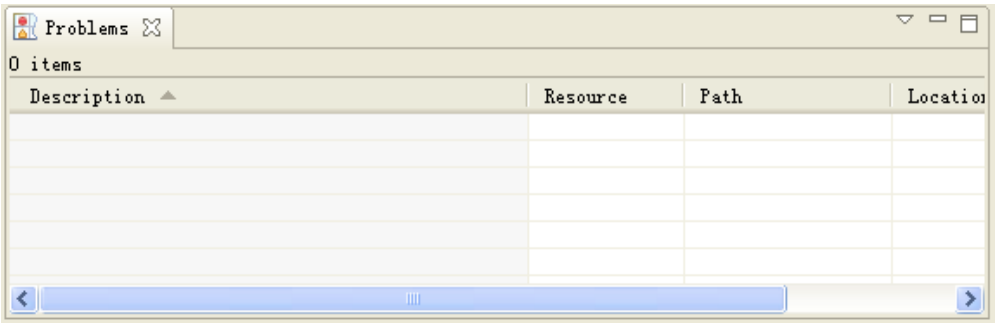


图 2.22 LAB1 工程调试结果

（2）单击绿色的 Debug 按钮  进行下载调试，得到图 2.23 所示的界面。

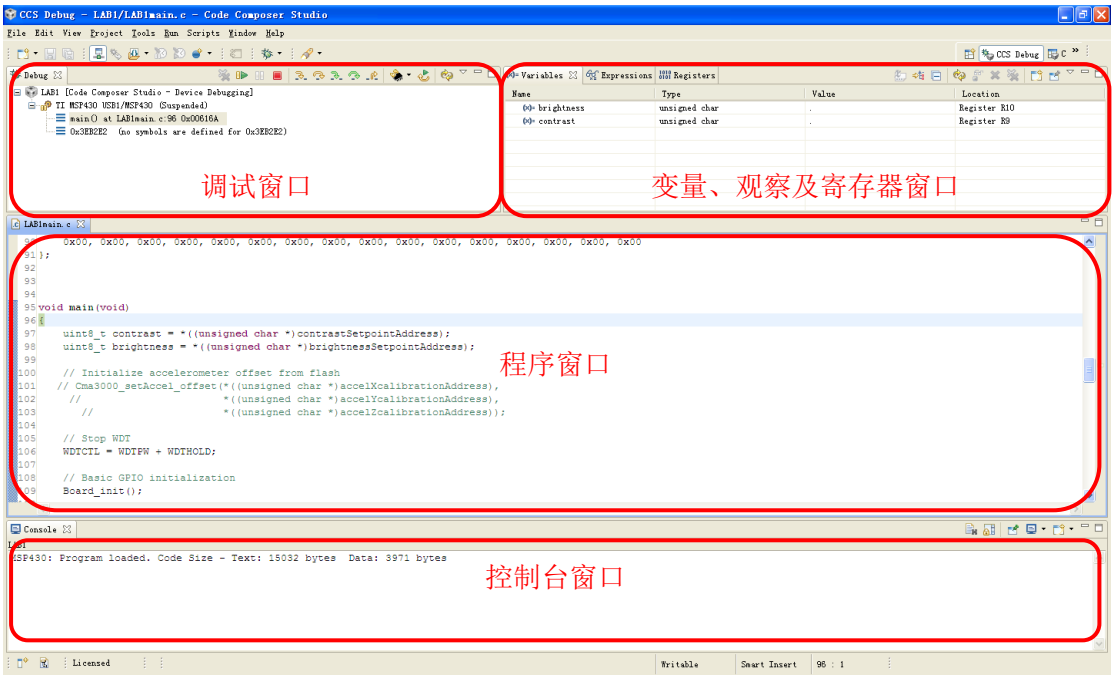


图 2.23 调试窗口界面

（3）单击运行图标 运行程序，观察显示的结果。在程序调试的过程中，可通过设置断点来调试程序：选择需要设置断点的位置，右击鼠标选择 Breakpoints→Breakpoint，断点设置成功后将显示图标，可以通过双击该图标来取消该断点。程序运行的过程中可以通过单步调试按钮 配合断点单步的调试程序，单击重新开始图标 定位到 main()函数，单击复位按钮 复位。可通过中止按钮 返回到编辑界面。

（4）在程序调试的过程中，可以通过 CCSV5.1 查看变量、寄存器、汇编程序或者是 Memory 等的信息 显示出程序运行的结果，以和预期的结果进行比较，从而顺利地调试程序。单击菜单 View→Variables，可以查看到变量的值，如图 2.24 所示。

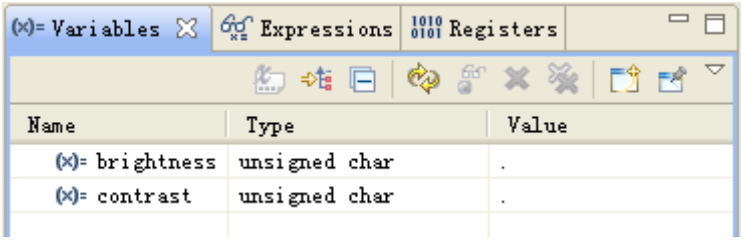


图 2.24 变量查看窗口

（5）点击菜单 View→Registers，可以查看到寄存器的值，如图 2.25 所示。

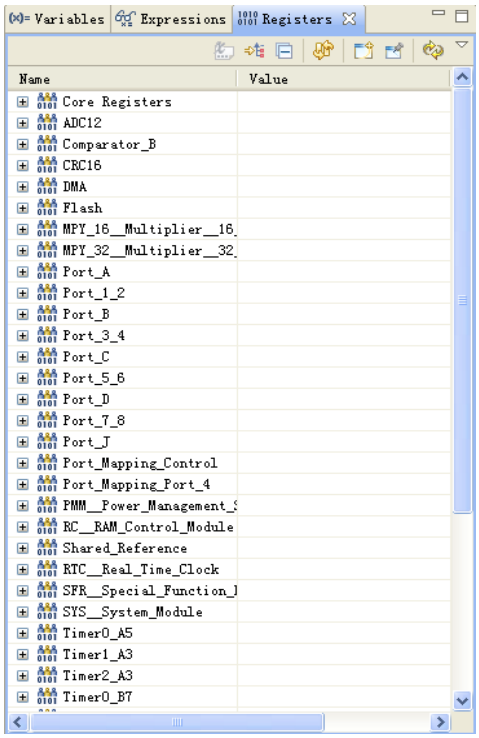


图 2.25 寄存器查看窗口

（6）点击菜单 View→Expressions，可以得到观察窗口，如图 2.26 所示。可以通过 Add new 添加观察变量，或者在所需观察的变量上右击，选择 Add Watch Expression 添加到观察窗口。

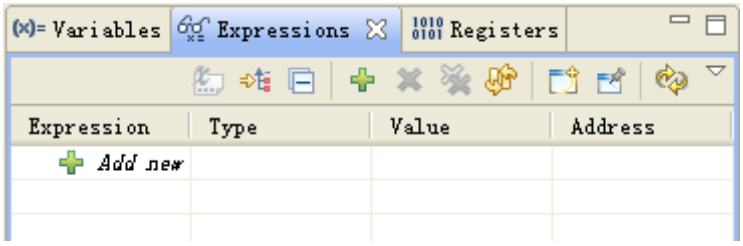


图 2.26 观察窗口

（7）点击菜单 View→Disassembly，可以得到汇编程序观察窗口，如图 2.27 所示。

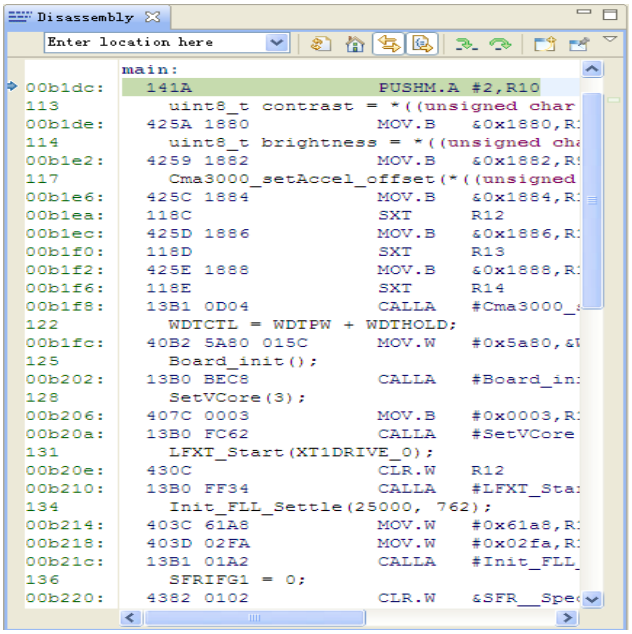


图 2.27 汇编程序观察窗口

（8）点击菜单 View→Memory Browser，可以得到内存查看窗口，如图 2.28 所示。

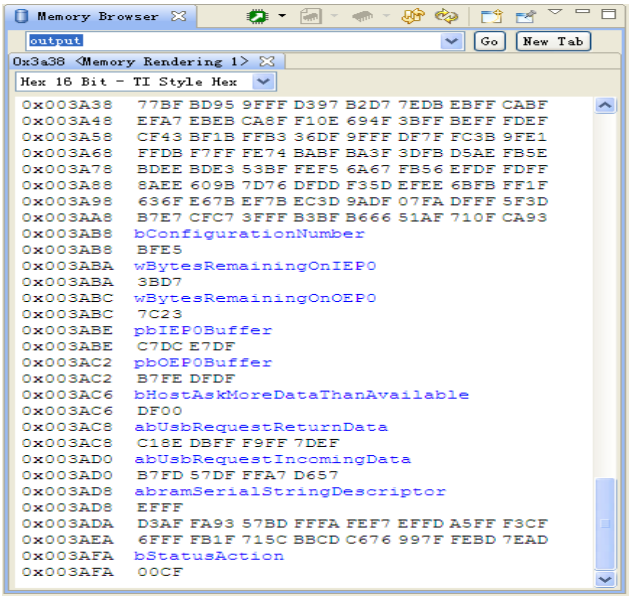


图 2.28 内存查看窗口

（9）点击菜单 View→Break points，可以得到断点查看窗口，如图 2.29 所示。

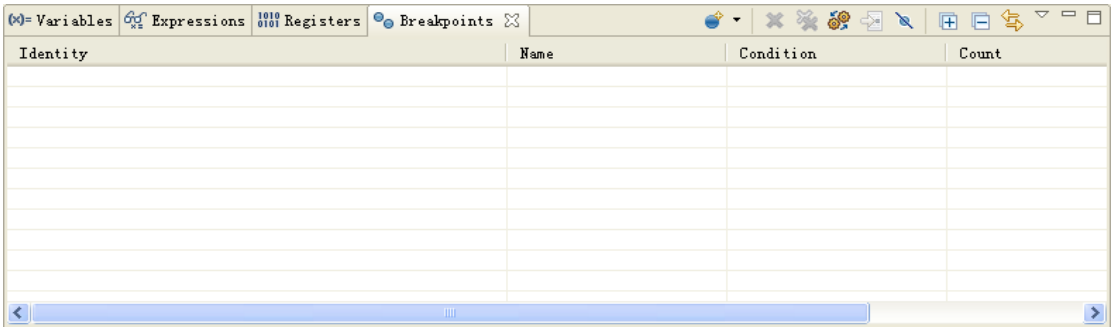


图 2.29 断点查看窗口

2.5 CCSv5.1 资源管理器介绍及应用

（1）CCSv5.1 具有很强大的功能，并且其内部的资源也非常丰富，利用其内部资源进行 MSP430 单片机开发，将会非常方便。现在演示 CCSv5.1 资源管理器的应用。如图 2.30 所示，通过 Help-->Welcome to CCS 打开 CCSv5.1 的欢迎界面。

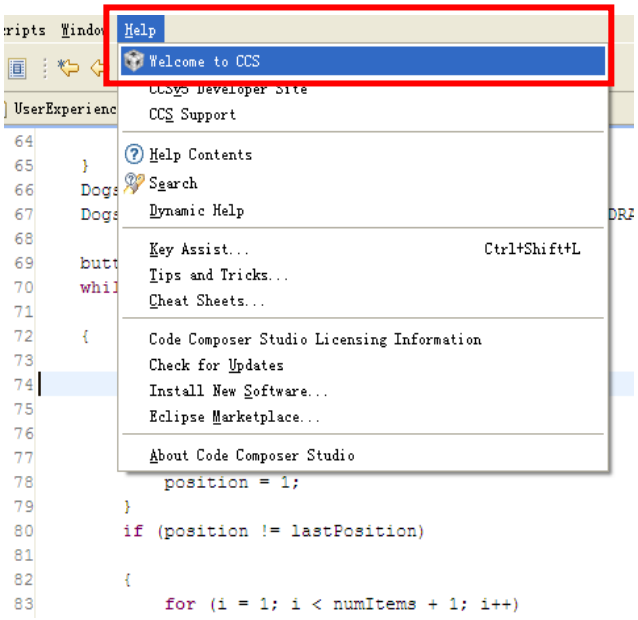


图 2.30 欢迎界面打开途径

（2）具体 TI 欢迎界面如图 2.31 所示，利用 New Project 链接可以新建 CCS 工程，具体新建步骤可以参考 2.3 节；利用 CCSv5.1 新建工程；利用 Examples 链接可以搜索到示例程序资源；利用 Import Project 链接可以导入已有 CCS 工程文件，具体导入步骤可以参考 2.2 节；利用 CCSv5.1 导入已有工程；利用 Support 链接可以在线获得技术支持；利用 Web Resources 链接可以进入 CCSv5.1 网络教程，学习 CCSv5.1 有关知识。

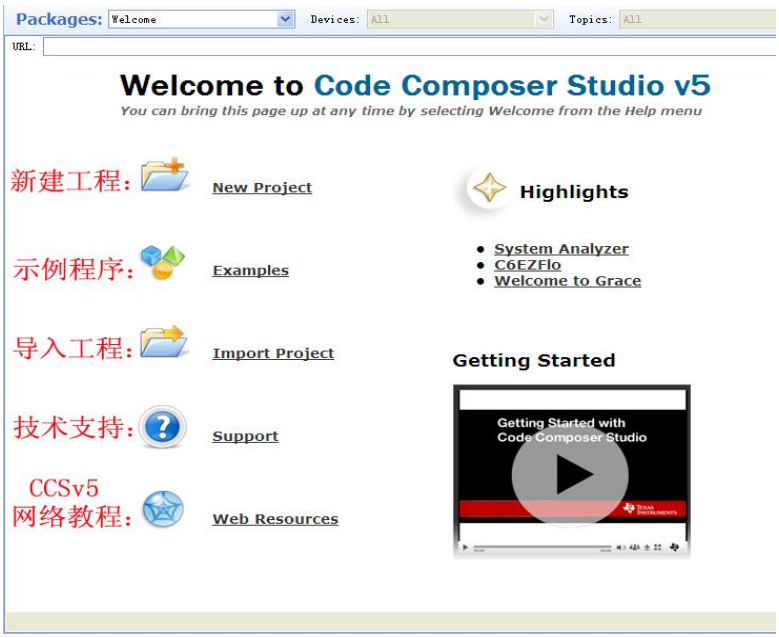


图 2.31 TI 欢迎窗口界面

（3）在“Packages”下拉菜单下选择 ALL，进入 CCSv5.1 资源管理器，如图 2.32 所示。在左列资源浏览器中，包含 MSP430Ware。MSP430Ware 将所有的 MSP430 MCU 器件的代码范例、数据表与其他设计资源整合成一个便于使用的程序包；基本上包含了成为一名 MSP430 MCU 专家所需要的一切。

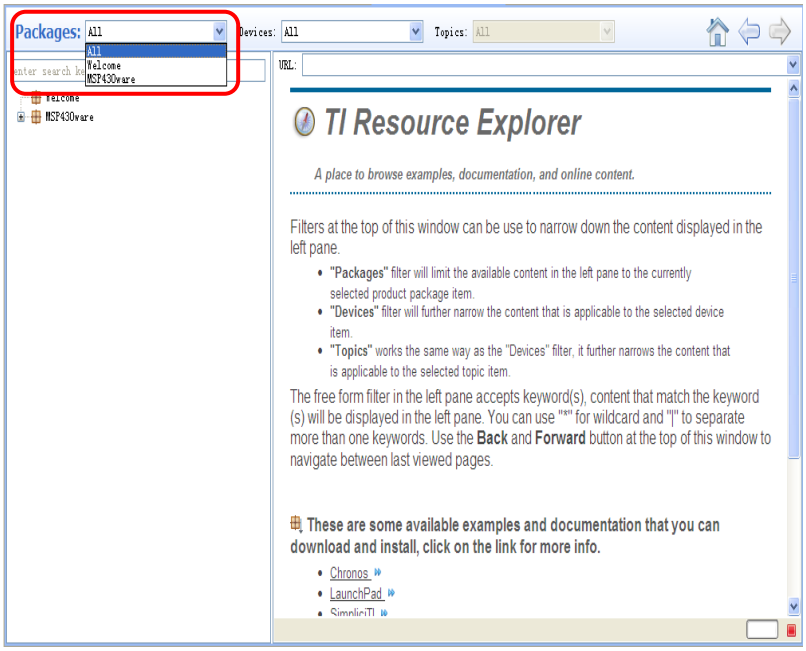


图 2.32 CCSv5.1 资源管理器窗口

（4）如图 2.33 所示，展开 MSP430ware，其包含三个方面内容：MSP430 单片机资源、开发装置资源以及 MSP430 资源库。

图 2.33 MSP430ware 界面

（5）展开 MSP430 单片机资源，得到如图 2.34 所示的界面，展开 MSP430F5XX/6XX，其中包含 F5xx/6xx 系列的用户指导、数据手册、单片机选型表以及示例代码。

图 2.34 单片机资源管理图

（6）展开 Code Examples，在下拉选项上选择 MSP430F552x，在右面窗口中，将得到 MSP430F552x 有关各内部外设的应用程序资源，如图 2.35 所示。若您打算在 ADC 模块的基础上，开发 MSP430，首先可以选择一个有关 ADC 的工程，作为讲解，在此选择第二个工程：MSP430F55xx_adc_01.c。单击该工程名称，将会弹出一个对话框，选择单片机型号，在此选择 MSP430F5529，单击 OK。之后您将在工程浏览器中，看到导入的工程：MSP430F55xx_adc_01，您可以在此基础上进行单片机的开发。

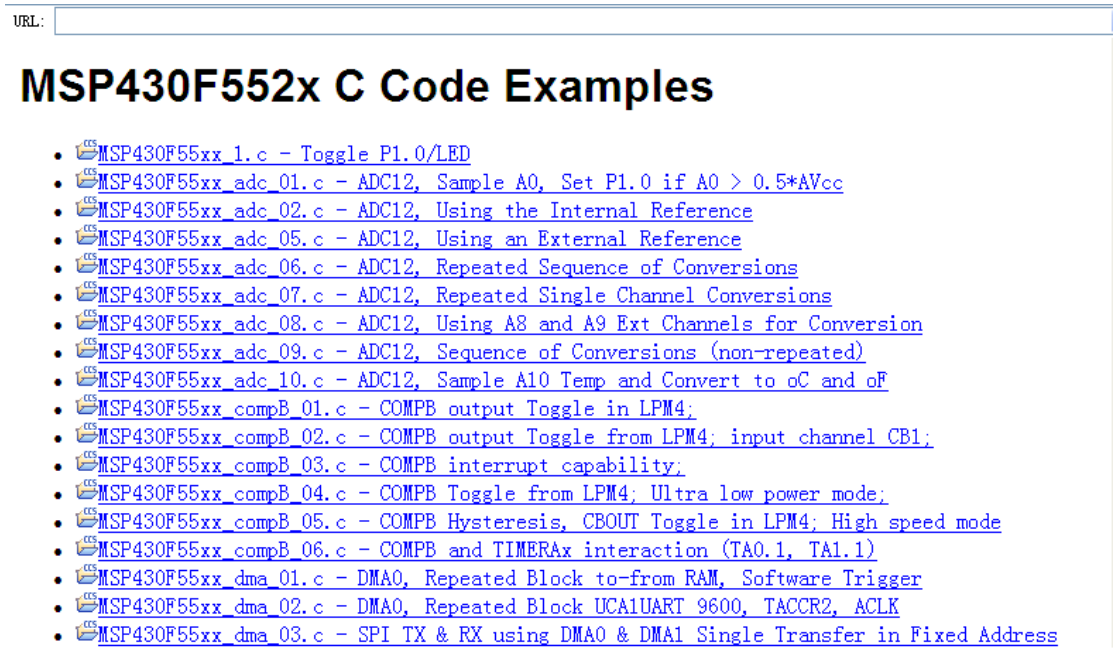


图 2.35 MSP430F552x 应用程序资源

（7）展开 Development Tools 开发装置资源，得到如图 2.36 所示的界面，其中包含 MSP-EXP430F5529 开发板资源。

图 2.36 开发装置资源管理图

（8）单击 User Experience Project (Code Limited)，在右面窗口中将得到如图 2.37 所示窗口。示例程序导入步骤分为四步，在保证开发板仿真器连接正确的前提下（在此利用开发板内置仿真器），单击第一步，将示例工程导入 CCS，您将在资源浏览器中，看到导入的工程：MSP-EXP430F5529 User Experience_16KB，并且第一步和第三步后面蓝色的对号变亮。单击第二步，对示例工程进行编译，编译完成后，将发现第二步后面蓝色的对号变亮。单击第四步，将示例工程下载到开发板。

图 2.37 MSP-EXP430F5529 原板载程序资源

（9）展开 Libraries 资源库，得到如图 2.38 所示的界面，其中包含 MSP430 驱动程序库以及 USB 的开发资源包。“MSP430 驱动程序库”为全新高级 API，这种新型驱动程序库能够使用户更容易地对 MSP430 硬件进行开发。就目前而言，MSP430 驱动程序库可支持 MSP430F5xx 和 6xx 器件。MSP430USB 开发资源包包含了开发一个基于 USB 的 MSP430 项目所需的所有源代码和示例应用程序，该开发资源包只支持 MSP430USB 设备。

图 2.38 资源库管理图

第二部分：实验例程介绍

第三章 液晶显示及时钟实验

3.1 实验目的

（1）学习实验所需硬件电路原理；
（2）学习实验相关程序资源；
（3）学习按键外部中断的原理；
（4）学习定时器 TB0 的原理及操作；
（5）学习利用定时器产生 PWM 波形的方法；
（6）学习 RTC（实时时钟）的原理及操作；
（7）学习 ADC12 的原理及操作；
（8）学习点阵 LCD 液晶显示的原理及操作。

3.2 实验所需硬件电路模块介绍

本实验中需用到以下电路模块：点阵 LCD 液晶显示模块、按键输入模块、齿轮电位计采样模块，现将其电路介绍如下：

3.2.1 点阵 LCD 液晶显示模块电路

液晶显示电路如图 3.1 所示，在该电路中，液晶为 102×64 点阵 LCD，采用 SPI 模式实现数据的传输，在该电路中数据传输是单向的，数据只允许写入，其中 LCD_CS（P7.4）为片选信号、LCD_D/C（P5.6）为命令数据切换信号、SCLK（P4.3/PM_UCB1CLK）为数据传输时钟信号、SIMO（P4.1/PM_UCB1SIMO）为从设备输入主设备输出信号、LCD_RST（P5.7）为液晶复位信号、VCC 为显示电源提供信号，液晶引脚说明如图 3.2 所示，液晶显示模块电路引脚设置如下（在 Board_init()函数中）：

```
P5OUT &= ~(BIT6 + BIT7);          // LCD_C/D, LCD_RST输出拉低
P5DIR |= BIT6 + BIT7;             //P5.6、P5.7引脚切换为输出模式
P7OUT &= ~(BIT4 + BIT6);          // LCD_CS, LCD_BL_EN输出拉低
P7DIR |= BIT4 + BIT6;            //P7.4、P7.6引脚切换为输出模式
P4OUT &= ~(BIT1 + BIT3);         // SIMO, SCK输出拉低
P4DIR &= ~BIT2;                  // SOMI引脚作为输入
P4DIR |= BIT1 + BIT3;            //P4.1、P4.3 引脚切换为输出模式
```

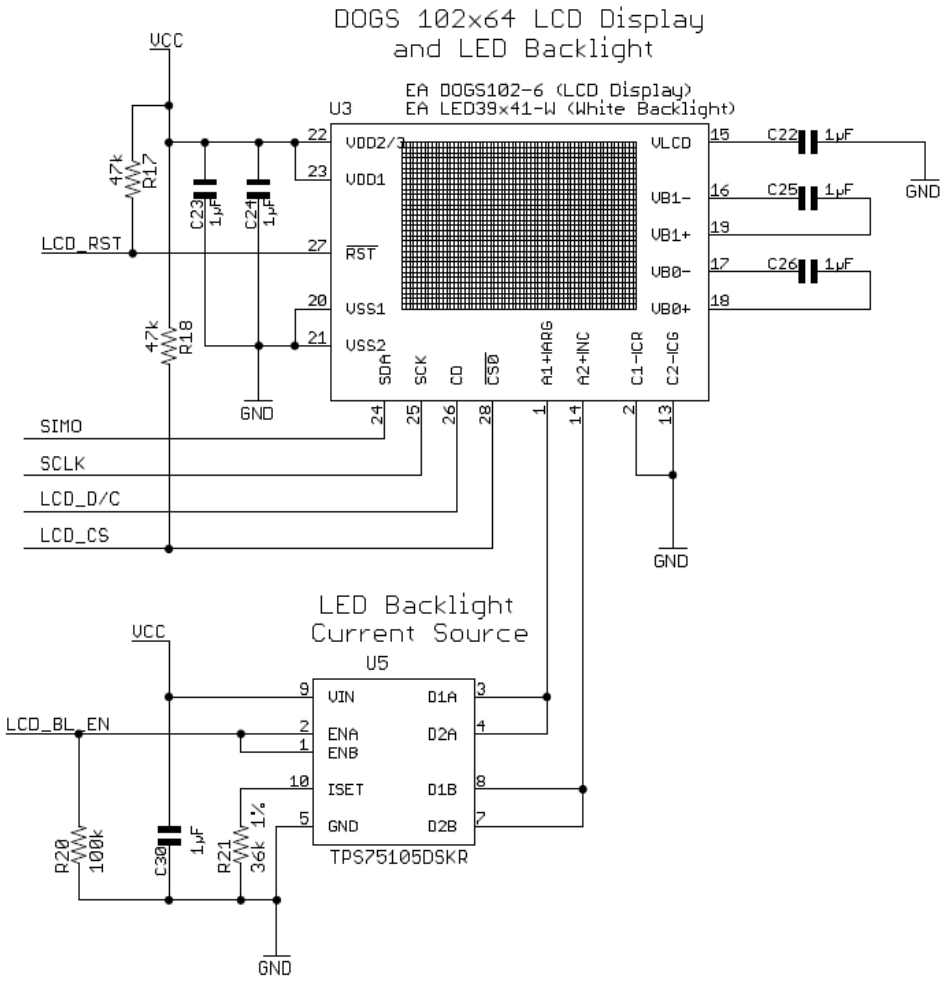


图 3.1 LCD 液晶显示电路

Pin	Symbol	Level	Function
1	NC		(A1+: LED backlight)
2	NC		(C1-: LED backlight)
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13	NC		(C2-: LED backlight)
14	NC		(A2+: LED backlight)

Pin	Symbol	Level	Function
15	VLCD	-	Power LC Drive
16	VB1-	-	Voltage Converter
17	VB0-	-	Voltage Converter
18	VB0+	-	Voltage Converter
19	VB1+	-	Voltage Converter
20	VSS	L	Power Supply 0V (GND)
21	VSS	L	Power Supply 0V (GND)
22	VDD2/3	H	Power Supply +2,5..3,3V
23	VDD1	H	Power Supply +2,5..3,3V
24	SDA	H / L	Data in (SPI: MOSI)
25	SCK	H / L	Clock (SPI: CLK)
26	CD	H / L	L= Command, H= Data
27	RST	L	Reset (active low)
28	CS0	L	Chip Select (active low)

图 3.2 LCD 液晶芯片 EA DOGS102-6 引脚说明

该液晶对比度可以通过命令进行更改，如图 3.3 所示，调节命令中 PM（0～63）的数值就可调节液晶显示对比度；该液晶背光为 LED 背光，通过在 LCD_BL_EN（P7.6）上输出 PWM 信号进行调节背光亮度。具体资料可以参考 http://www.lcd-module.de/eng/pdf/grafik/dogs102-6e.pdf。

Command		Command Code									Function
		CD	D7	D6	D5	D4	D3	D2	D1	D0	
(1)	Write Data Byte	1	data bit D[7..0]								Write one byte to memory
(4)	Set Column Address LSB	0	0	0	0	0	CA[3..0]				Set the SRAM column address CA=0..131
	Set Column Address MSB		0	0	0	1	CA[7..4]				
(5)	Set Power Control	0	0	0	1	0	1	PC[2..0]			PC0: 0=Booster OFF; 1=Booster ON PC1: 0=Regulator OFF; 1=Regulator ON PC2: 0=Follower OFF; 1=Follower ON
(6)	Set Scroll Line	0	0	1	SL[5..0]						Set the display startline number SL=0..63
(7)	Set Page Address	0	1	0	1	1	PA[3..0]				Set the SRAM page address PA=0..7
(8)	Set VLCD Resistor Ratio	0	0	0	1	0	0	PC[5..3]			Configure internal resistor ratio PC=0..7
(9)	Set Electronic Volume	0	1	0	0	0	0	0	0	1	Adjust contrast of LCD panel PM=0..63
			0	0	PM[5..0]						
(10)	Set All Pixel On	0	1	0	1	0	0	1	0	C1	C1=0: show SRAM content C1=1: Set all SEG-Drivers to ON

图 3.3 部分液晶显示命令

3.2.2 按键输入模块电路

按键输入电路如图 3.4 所示，在该电路中有两个按键 S1（P1.7）和 S2（P2.2）。注意，在该电路中按键无上拉电阻，应在程序中，利用 PxREN 使上拉电阻使能。另外还有两个具有特殊功能的按键 S3 和 S4，按键 S3 使系统复位，按键 S4 通过 USB 端口触发 BSL 过程。按键输入模块电路引脚定义如下（在 Board_init()函数中）：

```
PADIR &= ~0x0480;                    // P1.7、P2.2 作为输入
```

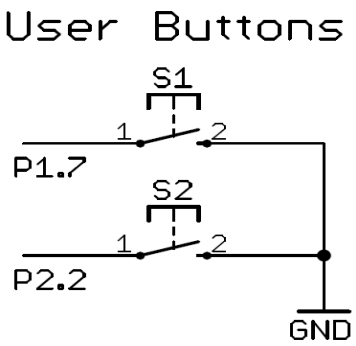


图 3.4 按键输入电路

3.2.3 齿轮电位计采样模块电路

齿轮电位计采样电路如图 3.5 所示，在该电路中，齿轮电位计的中间引脚与单片机 ADC 的通道 A5 相连，通过对其进行采样确定齿轮电位计的位置。通过短路块 JP2 可以断开与 P8.0 口的连接。齿轮电位计采样模块电路引脚定义如下（在 Board_init()函数中）：

```
P6DIR &= ~BIT5;                      // A5 ADC 作为输入
P8OUT &= ~BIT0;                      // P8.0输出拉低
P8DIR |= BIT0;                       //P8.0作为输出
```

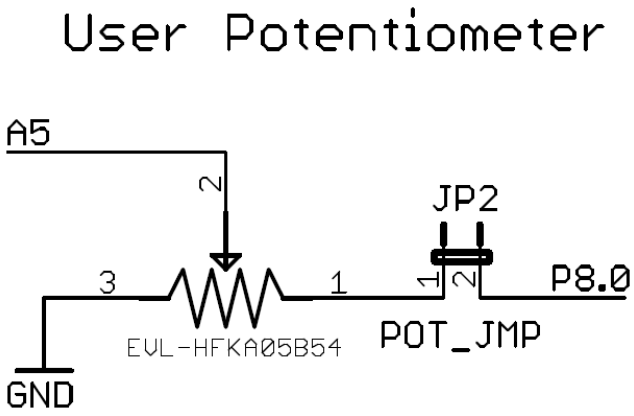


图 3.5 齿轮电位计采样电路

3.3 程序资源介绍

在 MSP-EXP430F5529 开发板实验一程序代码文件夹（MSP-EXP430F5529 LAB CODE\LAB1）中包含一个名为 MSP-EXP430F5529_HAL 的硬件模块程序资源库，其中包含本实验所需的程序资源，现将其功能简单介绍如下：

- HAL_Board.h/.c ： MSP-EXP430F5529开发板基本配置程序；
- HAL_Menu.h/.c ： 用户菜单界面功能管理程序；
- HAL_Buttons.h/.c ： 按键S1和S2的功能管理程序；
- HAL_Dogs102x6.h/.c ： 102x64液晶功能管理程序；
- HAL_Wheel.h/.c ： 齿轮电位计功能管理程序。

3.3.1 HAL_Board.h/.c 程序介绍

HAL_Board.h/.c 模块程序的功能为对 MSP-EXP430F5529 开发板端口进行初始化和控制 LED 指示灯的状态。

（1）LED的定义程序段：

```
#define LED1        0x01
#define LED2        0x02
#define LED3        0x04
#define LED4        0x08
#define LED5        0x10
#define LED6        0x20
#define LED7        0x40
#define LED8        0x80
#define LED ALL     0xFF
```

（2）下面函数的功能为 MSP-EXP430F5529 开发板初始化：

```
extern void Board_init(void);
```

（3）下面函数的功能为实现 LED 的控制。其中参数：LedMask--LED 的定义值。

```
extern void Board_ledOn(uint8_t ledMask);
extern void Board_ledOff(uint8_t ledMask);
extern void Board_ledToggle(uint8_t ledMask);
```

例子：若点亮 LED1 和 LED2，其程序可写为：

```
extern void Board_ledOn(LED1+LED2);
```

3.3.2 HAL_Menu.h/.c 程序介绍

下面函数的功能为显示菜单，并返回利用滚动齿轮电位计选择的菜单项目数。其中参数：**menuText--指向菜单选项文本的指针；numItems--总的菜单项目数（1～7）。

```
extern uint8_t Menu_active(char **menuText, uint8_t numItems);
```

例子：为了显示菜单，应首先创建菜单选项的文本结构，如下面程序所示（lab1.c 中）：

```
static const char *const clockMenuText[] = {
    "===LAB1:Clock====",
    "1. Contrast ",
    "2. Backlight ",
    "3. Digital Clock ",
    "4. Analog Clock ",
    "5. Set Time "
};
```

然后利用下面的程序进行菜单的显示和执行代码的选择，Menu_active((char **) clockMenuText, 5)返回菜单选择项，之后利用 switch/case 语句进行选择。

```
        selection = Menu_active((char **)clockMenuText, 5);
        if (buttonsPressed & BUTTON_S2);
        else
          switch (selection)
           {
              case 1: ContrastSetting(); break;
              case 2: BacklightSetting(); break;
              case 3: DisplayDigitalClock(); break;
              case 4: DisplayAnalogClock(); break;
              case 5: SetTime(); break;
              default: break;
           }
```

3.3.3 HAL_Buttons.h/.c 程序介绍

该段程序的功能是用来对按键进行初始化和功能管理。在该程序模块中，按键 S1 和 S2 利用 P1.7 口和 P2.2 口的外部中断功能。此外，看门狗定时器在此模块中用作按键消抖。

（1）按键定义程序段：

```
#define BUTTON_S2       0x0400
#define BUTTON_S1       0x0080
#define BUTTON_ALL      0x0480
```

（2）按键键值定义变量和消抖标志位：

```
volatile extern uint16_t buttonsPressed;
volatile extern uint8_t buttonDebounce;
```

（3）该程序模块定义了三个函数，分别为：按键初始化函数、使能按键中断函数和禁用按键中断函数。其中参数：buttonsMask--按键键值。

```
extern void Buttons_init(uint16_t buttonsMask);
extern void Buttons_interruptEnable(uint16_t buttonsMask);
extern void Buttons_interruptDisable(uint16_t buttonsMask);
```

（4）以下为按键 S1 和按键 S2 的中断服务程序：

```
#pragma vector=PORT1_VECTOR
__interrupt void Port1_ISR(void)
```

```
#pragma vector=PORT2_VECTOR
__interrupt void Port2_ISR(void)
```

例子：① 若在程序设计中，需等待按键被按下，则可使用以下程序段：按键未被按下，则执行段内程序；若按键被按下则跳出该段循环。

```
while (!buttonsPressed)
  {
   ……
  }
```

② 若在程序设计中，需等待 S2 按键被按下，则可使用以下程序段：若 S2 按键未被按下，则执行段内程序；若 S2 按键被按下则跳出该段循环。

```
while (!(buttonsPressed & BUTTON_S2))
  {
   ……
  }
```

3.3.4 HAL_Dogs102x6.h/.c 程序介绍

（1）下面函数的功能为 LCD 初始化。

```
extern void Dogs102x6_init(void);
```

（2）下面函数的功能为背光初始化。

```
extern void Dogs102x6_backlightInit(void);
```

（3）下面函数的功能为禁用 LCD。

```
extern void Dogs102x6_disable(void);
```

（4）下面函数的功能为通过三线 SPI 方式，将命令字发送给 LCD。其中参数：* sCmd--指向命令字的指针；i--命令字的位数。

```
extern void Dogs102x6_writeCommand(uint8_t* sCmd, uint8_t i);
```

（5）下面函数的功能为通过三线 SPI 方式，将数据写入 LCD。其中参数* sData--指向数据的指针；i--数据的位数。

```
extern void Dogs102x6_writeData(uint8_t* sData, uint8_t i);
```

（6）下面函数的功能为设置 LCD 内存的地址。其中参数 pa（0～7）--要写入 LCD 内存中的行地址；ca（0～101）--要写入 LCD 内存中的列地址，其中（0,0）表示屏幕的左上角地址。

```
extern void Dogs102x6_setAddress(uint8_t pa, uint8_t ca);
```

（7）下面函数的功能为返回对比度的值，可以供以后的程序中调用，得到当前的对比度值。

```
extern uint8_t Dogs102x6_getContrast(void);
```

（8）下面函数的功能为返回背光度的值，可以供以后的程序中调用得到当前的背光度值。

```
extern uint8_t Dogs102x6_getBacklight(void);
```

（9）下面函数的功能为设置 LCD 对比度，其中参数：newContrast--所需设置对比度的级别，其取值范围为 0～31，31 为对比度最大的设置。

```
extern void Dogs102x6_setContrast(uint8_t newContrast);
```

（10）下面函数的功能为设置 LCD 背光亮度，其中参数：brightness--所需设置背光的级别，其取值范围为 0～11，11 为背光亮度最大的设置。

```
extern void Dogs102x6_setBacklight(uint8_t brightness);
```

（11）下面函数的功能为翻转液晶显示的内容。

```
extern void Dogs102x6_setInverseDisplay(void);
```

（12）下面函数的功能为将翻转后的液晶显示的内容再翻转为正的。

```
extern void Dogs102x6_clearInverseDisplay(void);
```

（13）下面函数的功能为将当前显示的图像下移固定的行数，其中参数：lines--需下移的行数，其取值范围为 0～63。

```
extern void Dogs102x6_scrollLine(uint8_t lines);
```

（14）下面函数的功能为启用 LCD 液晶上的所有像素，进行测试。

```
extern void Dogs102x6_setAllPixelsOn(void);
```

（15）下面函数的功能为中断（14）函数的功能，使液晶返回正常工作。

```
extern void Dogs102x6_clearAllPixelsOn(void);
```

（16）下面函数的功能为清除屏幕上的像素，即清屏程序。

```
extern void Dogs102x6_clearScreen(void);
```

（17）下面函数的功能为将一个字符写入确定行列的 LCD 液晶显示屏上，其中参数：row--写入字符的行地址（其取值范围为 0～7）；col--写入字符的列地址（其取值范围为 0～102）；f--指向需写入字符的指针；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_charDraw(uint8_t row, uint8_t col, uint16_t f,
uint8_t style);
```

（18）下面函数的功能为将一个字符写入确定坐标的 LCD 液晶显示屏上，其中参数：x--水平坐标（其取值范围为 0～102），y--垂直坐标（其取值范围为 0～63）；f--指向需写入字符的指针；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_charDrawXY(uint8_t x, uint8_t y, uint16_t f,
uint8_t style);
```

（19）下面函数的功能为将一串字符串写入确定行列的 LCD 液晶显示屏上，其中参数：row--写入字符串的行地址（其取值范围为 0～7）；col--写入字符串的列地址（其取值范围为 0～102）；*word--指向需写入字符串 word[]的指针；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_stringDraw(uint8_t row, uint8_t col, char *word,
uint8_t style);
```

（20）下面函数的功能为将一串字符串写入确定坐标的 LCD 液晶显示屏上，其中参数：x--水平坐标（其取值范围为 0～102），y--垂直坐标（其取值范围为 0～63）；*word--指向需写入字符串 word[]的指针；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_stringDrawXY(uint8_t x, uint8_t y, char *word,
uint8_t style);
```

（21）下面函数的功能为清除 LCD 上某行的显示内容，其中参数：row--需清除内容的行数，其取值范围为 0～7。

```
extern void Dogs102x6_clearRow(uint8_t row);
```

（22）下面函数的功能为绘制一个像素的内容，其中参数：x--绘制点的 x 坐标（其取值范围为 0～63）；y--绘制点的 y 坐标（其取值范围为 0～101）；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_pixelDraw(uint8_t x, uint8_t y, uint8_t style);
```

（23）下面函数的功能为在 LCD 显示屏上绘制一个水平线，其中参数：x1--水平线起始像素 x 坐标（0～101）；x2--水平线终止像素 x 坐标（0～101）；y--水平线的 y 坐标（0～63）；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_horizontalLineDraw(uint8_t x1, uint8_t x2,
uint8_t y, uint8_t style);
```

（24）下面函数的功能为在 LCD 显示屏上绘制一个垂直线，其中参数：y1--垂直线起始像素 y 坐标（0～63）；y2—垂直线终止像素 y 坐标（0～63）；x--垂直线的 x 坐标（0～101）；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_verticalLineDraw(uint8_t y1, uint8_t y2, uint8_t
x, uint8_t style);
```

（25）下面函数的功能为绘制一条从（x1,y1）到（x2,y2）的直线，其中参数：（x1,y1）为所绘直线的起始坐标；（x2,y2）为所绘直线的终止坐标；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_lineDraw(uint8_t x1, uint8_t y1, uint8_t x2,
uint8_t y2, uint8_t style);
```

（26）下面函数的功能为绘制一个圆，其中参数：x--所绘圆的圆心 x 坐标；y--所绘圆的圆心 y 坐标；radius--所绘圆的半径；style--文本的样式选择（0 表示白底黑字，1 表示黑底白字）。

```
extern void Dogs102x6_circleDraw(uint8_t x, uint8_t y, uint8_t radius,
uint8_t style);
```

（27）下面函数的功能为绘制一个图案，其中参数：IMAGE[]--存储所绘图案的字模；row--图案开始绘制的行数；col--图案开始绘制的列数。

```
extern void Dogs102x6_imageDraw(const uint8_t IMAGE[], uint8_t row,
uint8_t col);
```

（28）下面函数的功能为清除一定面积的图案，其中参数：height--需清除图案的行数；width--需清除图案的列数（清除图案的面积为 height*width）；row--清除图案的起始行数；col--清除图案的起始列数。

```
extern void Dogs102x6_clearImage(uint8_t height, uint8_t width,
uint8_t row, uint8_t col);
```

3.3.5 HAL_Wheel.h/.c 程序介绍

（1）齿轮电位计的初始化函数。

```
extern void Wheel_init(void);
```

（2）下面函数的功能为确定齿轮电位计的位置，其返回值为齿轮电位计的位置，取值范围为 0～7。

```
extern uint8_t Wheel_getPosition(void);
```

（3）下面函数的功能为得到并返回齿轮电位计的采样值，注意，该函数内部包含采样软件消抖。

```
extern uint16_t Wheel_getValue(void);
```

（4）下面函数的功能为禁用齿轮电位计。

```
extern void Wheel_disable(void);
```

（5）下面函数的功能为启用齿轮电位计。

```
extern void Wheel_enable(void);
```

3.4 实验内容

本章实验包括以下五个小实验：（1）对比度调节实验；（2）背光调节实验；（3）数字时钟实验；（4）模拟时钟实验；（5）时钟设置实验。

实验 1 主函数 lab1()的整体程序流程图如图 3.6 所示。

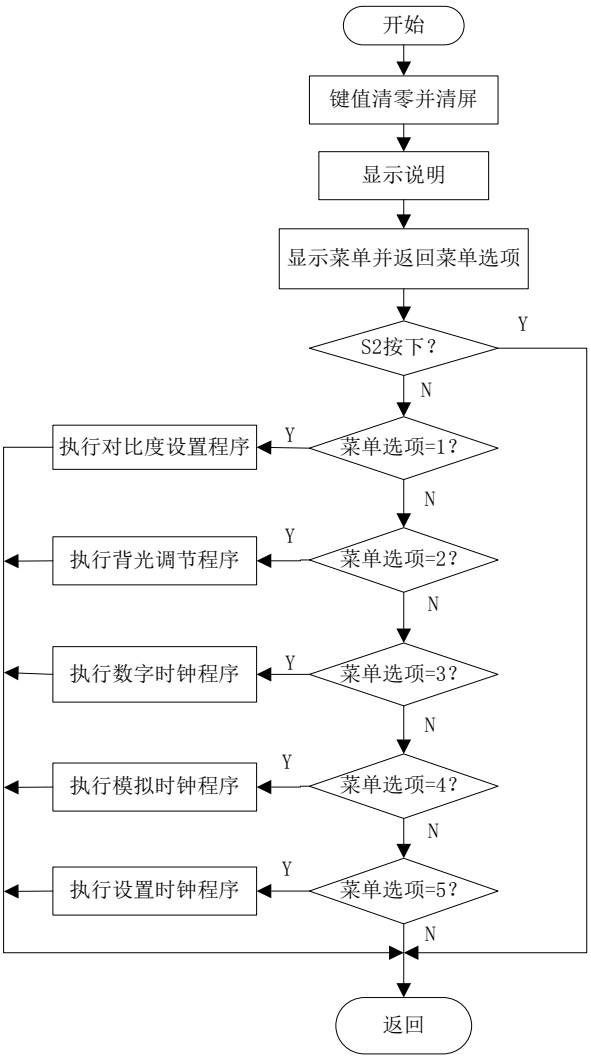


图 3.6 液晶显示及时钟实验整体程序流程图

3.5 实验原理

（1）按键外部中断

在该实验中，按键S1和S2与单片机具有外部中断功能的P1.7和P2.2引脚相连，在主函数初始化程序中利用以下函数进行按键的初始化和外部中断的使能。

```
void Buttons_init(uint16_t buttonsMask)
{
    BUTTON_PORT_OUT |= buttonsMask;      //P1.7和P2.2引脚输出高电平
    BUTTON_PORT_REN |= buttonsMask;      //上拉电阻使能
    BUTTON_PORT_SEL &= ~buttonsMask;    //选择功能为GPIO
}
```

```
void Buttons_interruptEnable(uint16_t buttonsMask)
{
    BUTTON_PORT_IES &= ~buttonsMask;     //上升沿触发中断
    BUTTON_PORT_IFG &= ~buttonsMask;     //清中断标志位
    BUTTON_PORT_IE |= buttonsMask;       //使能中断
}
```

（2）定时器TB0

MSP430F5529Timer_B为16位定时计数器，具有捕获/比较、PWM输出、时间间隔定时等功能，同时具有丰富的中断能力，其具有以下特性：

◆四种工作模式和四种计数长度的异步16位定时/计数器；

◆可选择配置的时钟源；

◆多达七个可配置的捕获/比较寄存器；

◆可配置的PWM输出；

◆同步双缓冲比较锁存器；

◆对所有 TB 中断快速响应的中断向量寄存器。

在本实验中利用 TB0.4 输出 PWM 信号，调节背光亮度。利用以下函数对 TB0 进行初始化：

```
void Dogs102x6_backlightInit(void)
{
    CS_BACKLT_DIR |= BACKLT;    //P7.6设为输出
    CS_BACKLT_OUT |= BACKLT;    //P7.6输出拉高
    CS_BACKLT_SEL |= BACKLT;    //P7.6设为定时器功能
    TB0CCTL4 = OUTMOD_7;        //输出模式7：PWM复位/置位
    TB0CCR4 = TB0CCR0 >> 1;
    TB0CCR0 = 50;               //PWM周期
    TB0CTL = TBSSEL_1 + MC_1; //  ACLK，增计数模式
}
```

由初始化函数可知，P7.6引脚作为定时器输出，TB0工作在增计数下输出模式7：PWM复位/置位，PWM的周期为50个ACLK周期时间。由图3.7可知，在定时器输出模式7下，输出PWM的占空比为：TB0CCR4/TBOCCR0。因此，在实验中，只需调节TB0CCR4的数值就可调节PWM输出的占空比，进而调节LCD背光的亮度。

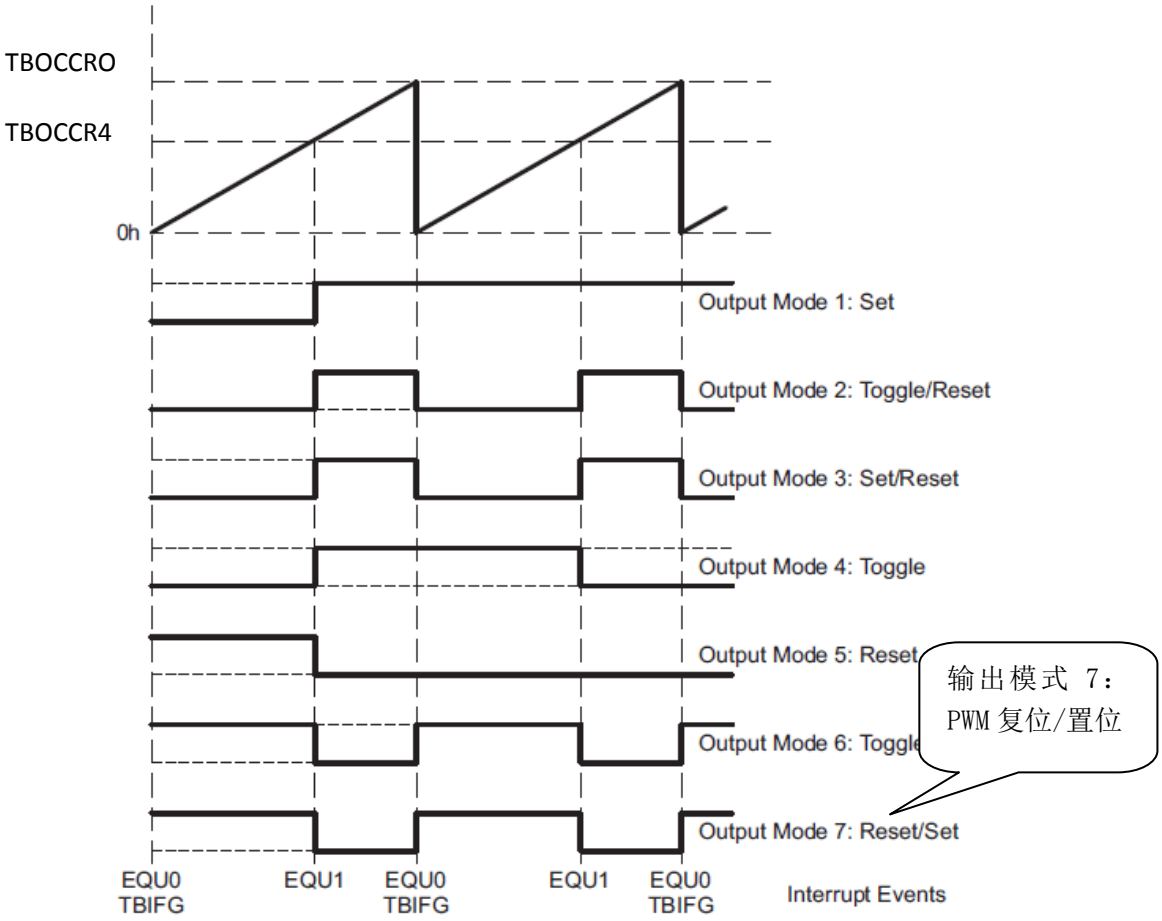


图3.7 增计数模式时输出实例

在实验中，利用Dogs102x6_setBacklight()函数对背光值进行设置：在该函数中brightness的取值范围为0～11，能够获得的占空比如下所示。

brightness取值	0	1	2	3	4	5
占空比	12/50	15/50	18/50	21/50	24/50	27/50
brightness取值	6	7	8	9	10	11
占空比	30/50	33/50	36/50	39/50	42/50	45/50

```
void Dogs102x6_setBacklight(uint8_t brightness)
{
    unsigned int dutyCycle = 0, i, dummy;
    if (brightness > 0)
    {
        TB0CCTL4 = OUTMOD_7;  //PWM复位/置位
        dummy = (TB0CCR0 >> 4);  //dummy取值为3
        dutyCycle = 12;
        for (i = 0; i < brightness; i++)
            dutyCycle += dummy;
        TB0CCR4 = dutyCycle;
        if (!backlight)
            TB0CTL |= MC0;
    }
    else
    {
        TB0CCTL4 = 0;
        TB0CTL &= ~MC0;
    }
    backlight = brightness;
}
```

（3）RTC（实时时钟）

MSP430F5529单片机实时时钟的框图如图3.8所示，该实时时钟模块提供了具有日历模式、灵活可编程闹钟和校准的时钟计数器功能，其具有以下特征：

◆可配置成实时时钟模式或一般目的的计数器；

◆在日历模式下提供了秒、分钟、小时、星期、日期、月份和年份；

◆具有中断功能；

◆实时时钟模式里可选择BCD码或二进制格式；

◆实时时钟模式里具有可编程闹钟；

◆实时时钟模式里具有时间偏差的逻辑校正。

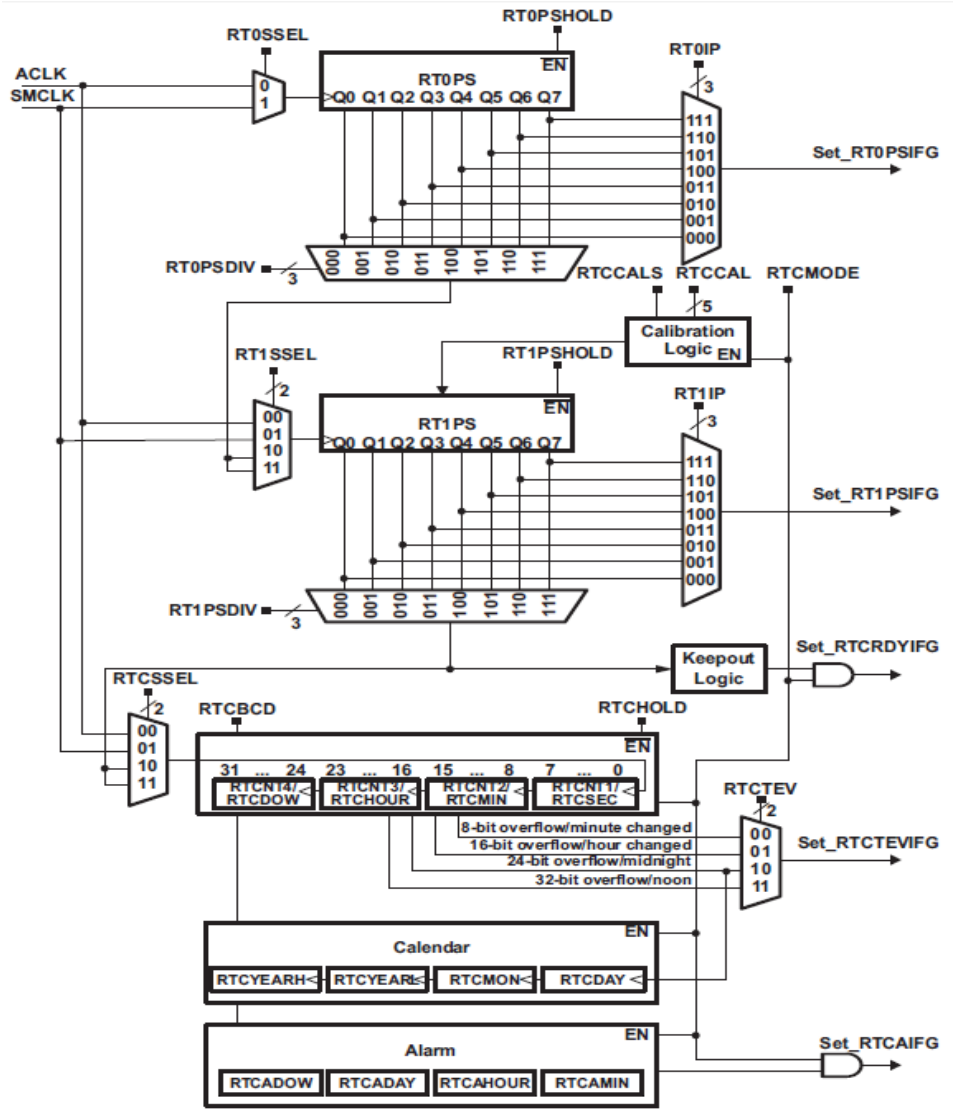


图3.8 实时时钟框图

在本实验中，利用以下SetupRTC()函数进行实时时钟的设置。在该函数中，RTC的日历模式被选中，实时时钟模块选择以BCD码形式提供秒、分、时、日、月、年的值。在日历模式下，分频计、RT0PS和RT1PS自动配置成为实时时钟提供一秒间隔的时钟，实时时钟时间寄存器的值每秒钟更新一次，只需正确读取实时时钟时间寄存器的值就可获取当前的时间。

```
void SetupRTC(void)
{
    RTCCTL01 = RTCMODE + RTCBCD + RTCHOLD + RTCTEV_1;//日历模式 BCD码格式 实时时钟停止  时钟变换（调整小时）
    RTCHOUR = 0x04;
    RTCMIN = 0x30;
    RTCSEC = 0x00;
    RTCDAY = 0x01;
    RTCMON = 0x01;
    RTCYEAR = 0x2011;//初始时间
    RTCCTL01 &= ~RTCHOLD;//日历正在运作
    RTCPS1CTL = RT1IP_5;            // 中断频率2hz
    RTCPS0CTL = RT0IP_7;          // 中断频率128hz
    RTCCTL0 |= RTCRDYIE + RTCTEVIE;         //实时时钟中断使能
}
```

（4）ADC12（12位模数转换器）

ADC12的结构图如图3.9所示，该模块为一个高效的12位模数转换器，具有以下特点：

◆最高可达200-ksps的采样速度；

◆无数据丢失的12位转换器；

◆采样-保持由采样周期控制，采样周期可通过软件设置或定时器确定；

◆利用软件对Timer_A和Timer_B进行初始化；

◆芯片内部的基准电压发生器（可选电压MSP430F5529：1.5V、2.0V、2.5V）；

◆软件选择内部或外部基准；

◆12路独立可配置的外部输入通道；

◆内部温度传感器转化通道，参考电压为AVCC和外部基准；

◆独立的选择通道基准源，包括正基准和负基准；

◆可选择的转化时钟源；

◆四种转化模式：单通道单次转换模式、序列通道单次转换模式、单通道多次转换模式、序列通道多次转换模式；

◆具有快速响应的18位中断向量寄存器；

◆16位的转换结果存储寄存器。

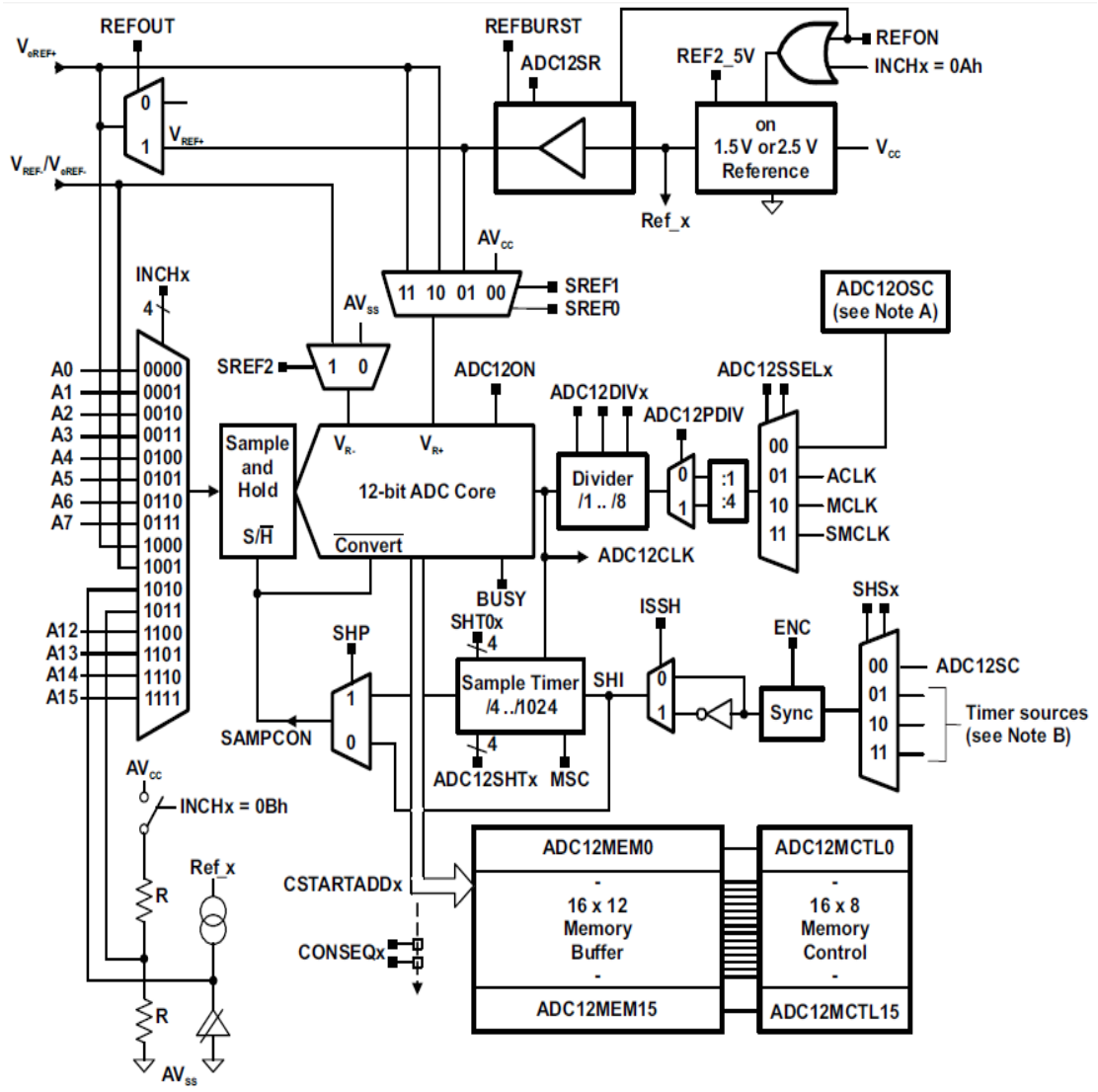


图3.9 ADC12结构图

在本实验中利用ADC12进行齿轮电位计的采样，以下函数为ADC12的初始化设置：

```
void Wheel_init(void)
{
    WHEEL_PORT_DIR |= WHEEL_ENABLE;
    WHEEL_PORT_OUT |= WHEEL_ENABLE;          // P8.0引脚输出高电平
    ADC12CTL0 = ADC12SHT02 + ADC12ON;      // 采样转换时间，打开ADC12内核
    ADC12CTL1 = ADC12SHP;                    // 利用采样定时器
    ADC12MCTL0 = ADC12INCH_5;                // 利用A5通道作为输入
    ADC12CTL0 |= ADC12ENC;                   // 使能转换
    ADC_PORT_SEL |= ADC_INPUT_A5;          // P6.5选择ADC功能
}
```

在实验中利用Wheel_getValue()函数，进行ADC采样。在该函数中，给出开始采样信号后，程序会进入低功耗模式0，CPU会关断，等待采样完成后置位采样中断标志位，唤醒CPU，之后进入采样中断服务程序，读取采样转换值。

```
uint16_t Wheel_getValue(void)
{
    ADC12IE = 0x01;                          // 使能ADC中断
    ADC12CTL0 |= ADC12SC;                   // 开始采样转换
    __bis_SR_register(LPM0_bits + GIE); // 进低功耗模式0
    ADC12IE = 0x00;                      // 禁用ADC中断
    //以下采样消抖
    if (positionData > positionDataOld)
        if ((positionData - positionDataOld) > 10)
            positionDataOld = positionData;
        else
            positionData = positionDataOld;
    else
    if ((positionDataOld - positionData) > 10)
        positionDataOld = positionData;
    else
        positionData = positionDataOld;
    return positionData;
}
```

当采样完成，ADC12硬件会自动将转换结果存储到相应的ADC12MEMx中，每个转换存储器ADC12MEMx都有自己对应的控制寄存器ADC12MCTLx，以控制各个转换存储器选择基本的转换条件。齿轮电位计采样，使用了ADC12MCTL0控制寄存器，因此转换结果存储在ADC12MEM0中，实验中利用以下ADC12中断服务程序读取采样转换结果：

```
#pragma vector = ADC12_VECTOR
__interrupt void ADC12_ISR(void)
{
    switch (__even_in_range(ADC12IV, ADC12IV_ADC12IFG15))
    {
        ……
        case  ADC12IV_ADC12IFG0:
            positionData = ADC12MEM0;                    // 读取转换结果
            __bic_SR_register_on_exit(LPM0_bits);
            break;
        ……
    }
}
```

3.6 对比度调节实验

3.6.1 程序代码

该实验的程序代码包含在 lab1.c 文件内：

```
void ContrastSetting(void)
{
    ..........
}
```

3.6.2 程序流程

对比度调节实验程序流程图如图 3.10 所示。

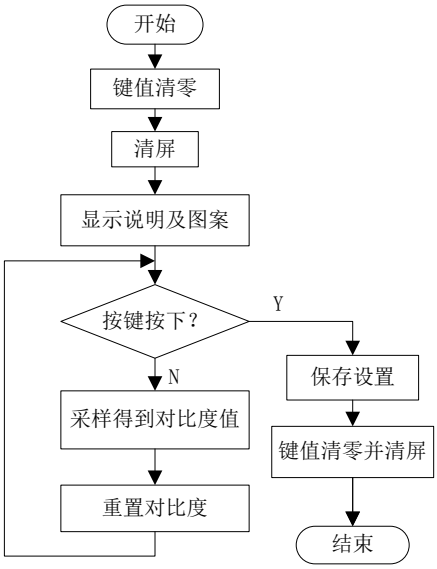


图 3.10 对比度调节实验程序流程图

3.6.3 实验步骤

（若 LAB1 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB1工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab1.c文件，在244行找到该对比度调节实验程序代码ContrastSetting ()，并在其中设置断点，断点位置如图3.11阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择1.Contrast，然后按下S1键，由于在

该程序中设置了断点，程序开始的界面，如图3.11所示。

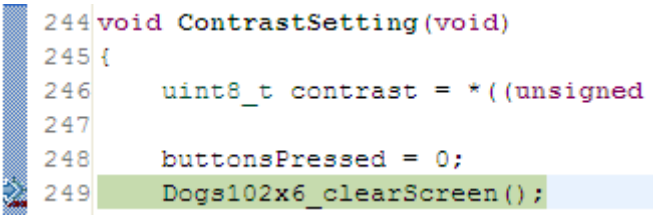


图3.11 对比度调节实验程序开始界面

（7）点击运行按钮，通过滚动齿轮电位计将当前液晶对比度调至合适值，任意按下 S1 或 S2 键将当前对比度设置进行保存并退出。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

3.6.4 实验结果

通过本实验，可以利用齿轮电位计调节液晶 LCD 的显示对比度。

3.7 背光调节实验

3.7.1 程序代码

该实验的程序代码包含在 lab1.c 文件内：

```
void BacklightSetting(void)
{
    ..........
}
```

3.7.2 程序流程

背光调节实验程序流程图如图 3.12 所示。

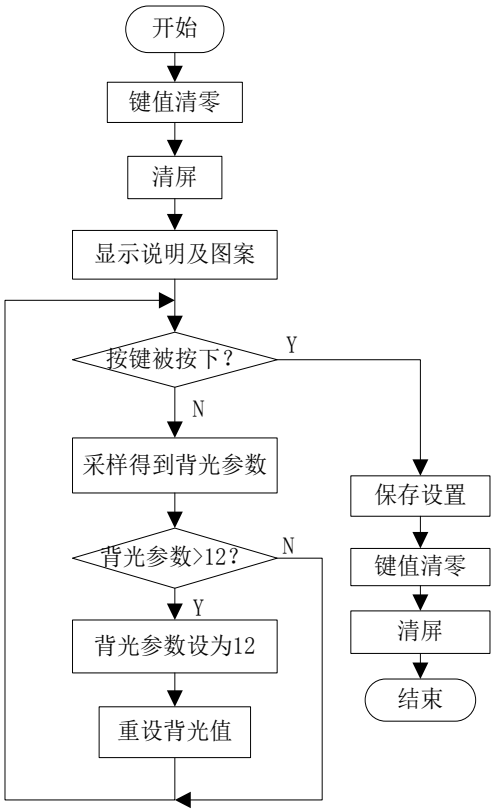


图 3.12 背光调节实验程序流程图

3.7.3 实验步骤

（若 LAB1 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB1工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab1.c文件，在274行找到该背光调节实验程序代码BacklightSetting ()，并在其中设置断点，断点位置如图3.13阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：2.Backlight，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图3.13所示。

```
274 void BacklightSetting(void)
275 {
276     uint8_t brightness = *((unsig
277
278     buttonsPressed = 0;
279     Dogs102x6_clearScreen();
```

图3.13 背光调节实验程序开始界面

（7）点击运行按钮，通过滚动齿轮电位计将当前液晶背光调至合适值，任意按下 S1 或 S2 键将当前背光设置进行保存并退出。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键  ，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

3.7.4 实验结果

通过本实验，可以利用齿轮电位计调节液晶 LCD 的背光。

3.8 数字时钟实验

3.8.1 程序代码

该实验的程序代码包含在 lab1.c 文件内：

```
void DisplayDigitalClock(void)
{
    ..........
}
```

3.8.2 程序流程

数字时钟程序流程图如图 3.14 所示。

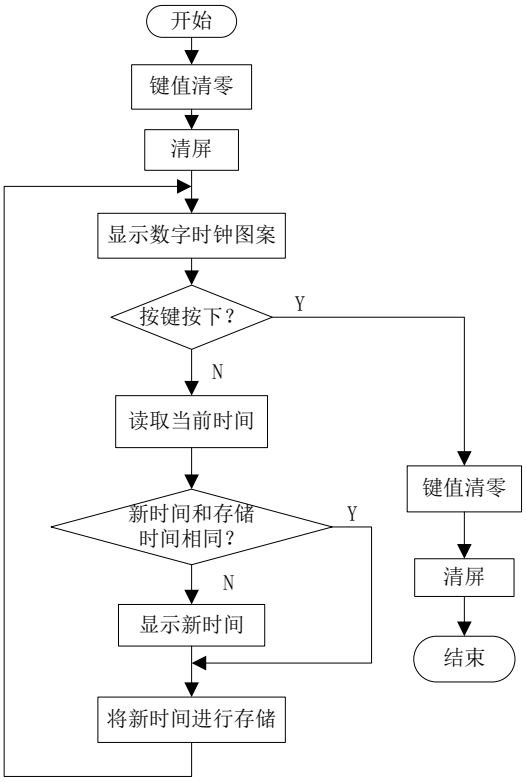


图3.14 数字时钟程序流程图

3.8.3 实验步骤

（若 LAB1 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB1工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab1.c文件，在339行找到该数字时钟实验程序代码DisplayDigitalClock ()，并在其中设置断点，断点位置如图3.15中阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：3.Digital Clock，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图3.15所示。

```
339 void DisplayDigitalClock(void)
340 {
341     buttonsPressed = 0;
342     Dogs102x6_clearScreen();
```

图3.15 数字时钟实验程序开始界面

（7）点击运行按钮，在 LCD 上将显示一个数字时钟的表盘，并且实时更新当前时间。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

3.8.4 实验结果

通过本实验，在液晶 LCD 上显示一个数字表盘，并实时更新显示当前时间。

3.9 模拟时钟实验

3.9.1 程序代码

该实验的程序代码包含在 lab1.c 文件内：

```
void DisplayAnalogClock(void)
{
    ..........
}
```

3.9.2 程序流程

模拟时钟试验程序流程图如图 3.16 所示。

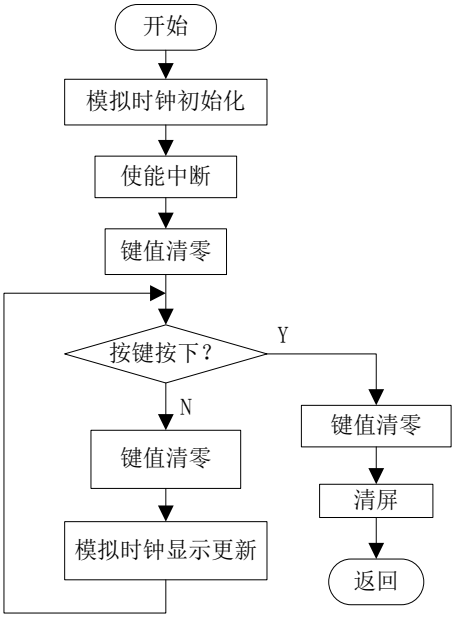


图3.16 模拟时钟实验程序流程图

3.9.3 实验步骤

（若 LAB1 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB1工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab1.c文件，在531行找到该数字时钟实验程序代码DisplayAnalogClock ()，并在其中设置断点，断点位置如图3.17中阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：4.Analog Clock，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图3.17所示。

```
531 void DisplayAnalogClock(void)
532 {
533     AnalogClockInit();
```

图3.17 模拟时钟程序开始界面

（7）点击运行按钮，在液晶 LCD 上将显示一个模拟时钟的表盘，并且实时更新当前时间。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键 ，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

3.9.4 实验结果

通过本实验，在液晶LCD上显示一个模拟表盘，并实时更新显示当前时间。

3.10 时钟设置实验

3.10.1 程序代码

该实验的程序代码包含在 lab1.c 文件内：

```
void SetTime(void)
{
    ..........
}
```

3.10.2 软件流程

时钟设置主程序流程图如图 3.18 所示，小时设置程序流程图如图 3.19 所示，分钟设置程序流程图如图 3.20 所示。

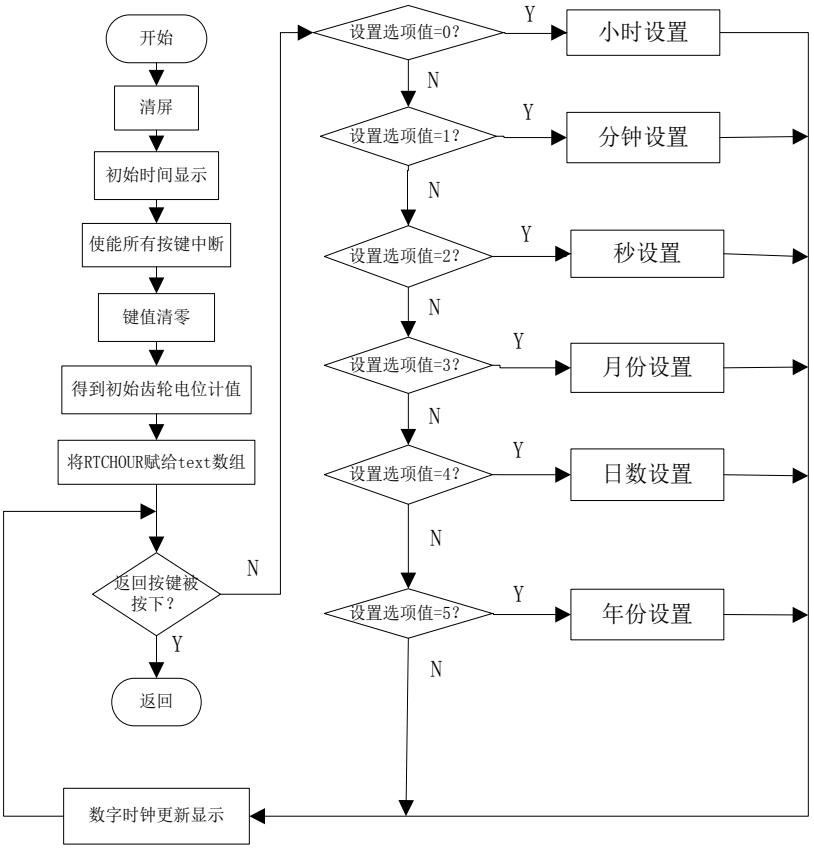


图 3.18 时钟设置主程序流程图

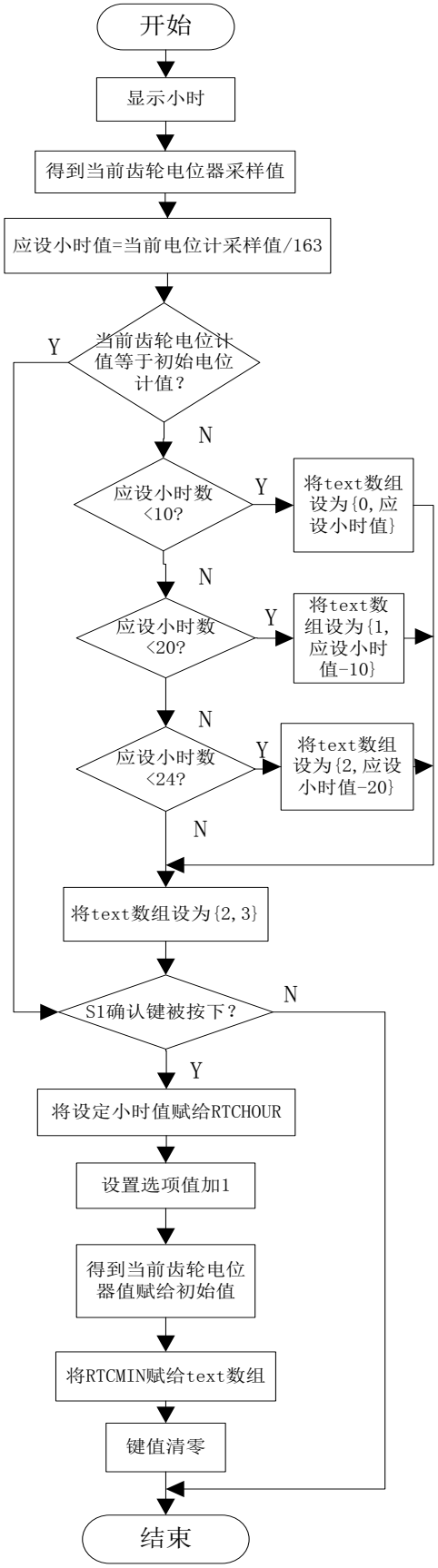


图 3.19 小时设置程序流程图

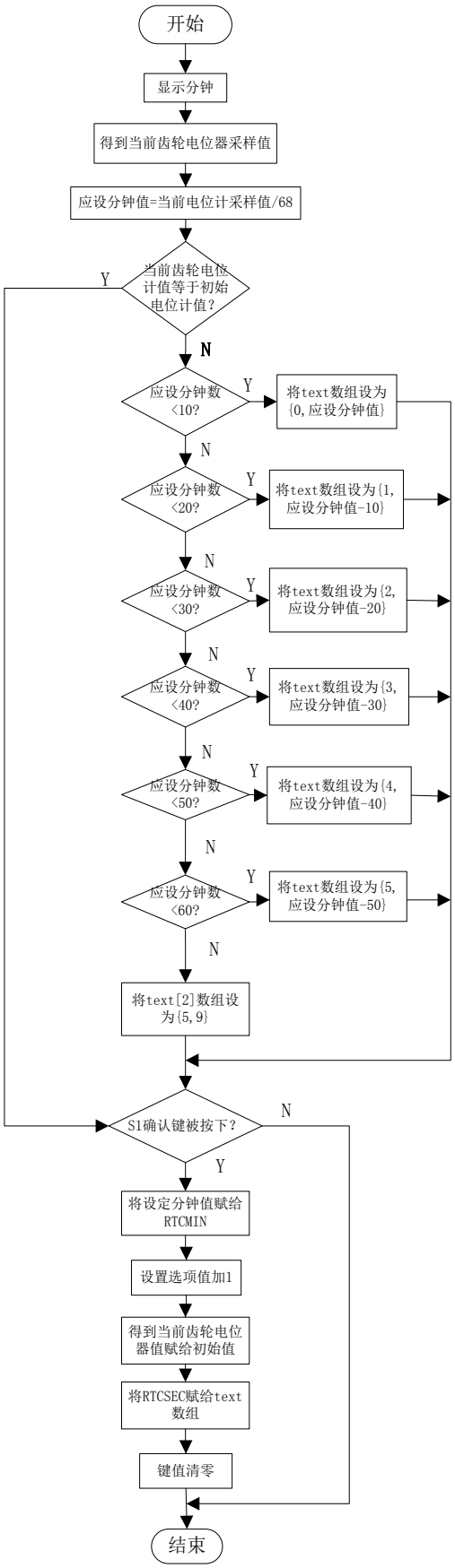


图 3.20 分钟设置程序流程图

其余设置与以上设置程序类似，您可类比读取程序。

3.10.3 实验步骤

（若 LAB1 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB1工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab1.c文件，在559行找到该时钟设置实验程序代码SetTime ()，并在其中设置断点，断点位置如图3.21中阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：5.Set Time，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图3.21所示。

```
559 void SetTime(void)
560 {
561     uint8_t horizontalPosition = 0;
562     uint8_t verticalPosition = 0;
563     char text[3] = {'0', '0', '\0'};
564     uint16_t wheelValue = 0;
565     uint16_t intialWheelValue = 0;
566
567     Dogs102x6_clearScreen();
```

图3.21 时钟设置实验程序开始界面

（7）点击运行按钮，在液晶 LCD 上将显示当前的数字时间，并进行实时更新。首先进行小时的设置，通过滚动齿轮电位计更改当前小时数，按下 S1 键进行确认，并进入分钟的设置。按照同样的方法，依次对分钟、秒、日、月、年进行设置，当对年设置完成后，按下 S1 键表示设置完成，退出设置程序。注意：在设置的任意过程中，按下 S2 键，都会将当前设置进行保存，并退出设置程序。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键 ，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

3.10.4 实验结果

当程序运行时，在液晶LCD上显示一个数字表盘，通过齿轮电位计和按键，可对当前时间进行设置并保存。

第四章 触摸按键应用实验

4.1 实验目的

（1）学习 MSP430F5529 单片机比较器的原理及操作；
（2）学习 LED 的控制原理及操作；
（3）学习利用定时器实现频率计的方法；
（4）学习电容触摸按键硬件电路原理；
（5）学习电容触摸程序资源；
（6）学习触摸按键应用实验操作及编程思想。

4.2 实验所需硬件电路模块介绍

本实验中需用到以下电路模块：点阵 LCD 液晶显示模块、按键输入模块、齿轮电位计采样模块、LED 指示模块、电容触摸按键模块。现将电容触摸按键模块电路和 LED 指示模块电路介绍如下：

4.2.1 电容触摸按键模块电路

如图 4.1 所示，该电路包括 5 个电容触摸按键，在每个触摸按键中包含一个 LED 指示灯，连接到端口 P1.1～P1.5，该 LED 可以用来指示按键触摸的状态。每个输入 CB0～CB4 连接到比较器 COMPB 的输入端，同时，CBOUT 连接到比较器的输出端。电容触摸资料可以参考以下链接 http://www.ti.com/tool/capsenselibrary。

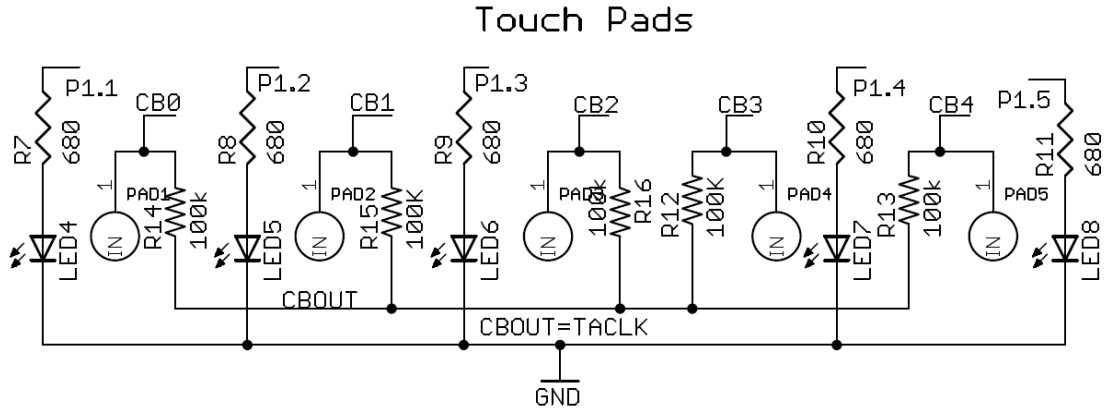


图 4.1 电容触摸按键模块电路

电容触摸按键模块电路引脚定义如下（在 Board_init()函数中）：

```
P1OUT &= ~0x7E;
P1DIR |= 0x7E;
P6OUT = 0x00;
P6DIR = 0x00;
```

4.2.2 LED 指示模块电路

除电容触摸按键电路中的 5 个 LED 外，MSP-EXP430F5529 开发板还有 1 个 LED 用于内置仿真器连接的指示和 3 个 LED 用于一般用途，其连接电路如图 4.2 所示，注意：通过短路块 JP3 可以断开 LED1 与 P1.0 口的连接。

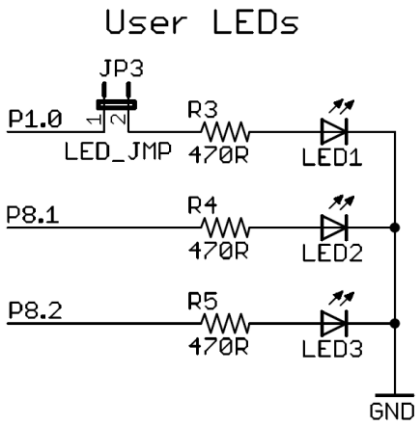


图 4.2 LED 连接电路

LED 引脚定义如下（在 Board_init()函数中）：

```
LED145678_PORT_OUT &= ~(BIT0 + BIT1 + BIT2 + BIT3 + BIT4 + BIT5);
LED145678_PORT_DIR |= BIT0 + BIT1 + BIT2 + BIT3 + BIT4 + BIT5;
LED23_PORT_OUT &= ~(BIT1 + BIT2);
LED23_PORT_DIR |= BIT1 + BIT2;
```

4.3 程序资源介绍

4.3.1 LED 控制程序介绍

LED 控制程序在第三章中已有介绍，请参考：3.3.1 HAL_Board.h/.c 程序介绍。

4.3.2 电容触摸程序介绍

在 MSP-EXP430F5529 开发板实验二程序代码文件夹（MSP-EXP430F5529 LAB CODE\LAB2）中包含一个名 CTS 的文件夹，该文件夹为电容触摸程序资源库。TI 电容式触摸产品系列可实现世界上最低功耗的触摸感测性能。MSP430 电容触摸程序资源库为开发人员提供了给任意 MSP430 微控制器增添触摸感应功能的选项，并且仅仅占用 1KB 的程序存储空间。该开源程序资源库免除了开发复杂触摸感测算法的需要，并支持各种不同的电容式触摸感应器，包括按钮、滑块、滚轮和近距离感应器。

以下介绍电容触摸程序资源库中的一些重要程序的功能。

（1）下面函数的功能为电容触摸初始化，检测基准电容，其中参数：groupOfElements--被测电容传感器结构体指针。

```
void TI_CAPT_Init_Baseline(const struct Sensor* groupOfElements);
```

（2）下面函数的功能为平均多次测量结果，动态更新基准电容值。其中参数：groupOfElements --被测电容传感器结构体指针；numberOfAverages--平均测量的次数。

```
void TI_CAPT_Update_Baseline(const struct Sensor* groupOfElements,
uint8_t numberOfAverages);
```

（3）下面函数的功能为复位基准电容检测算法。

```
void TI_CAPT_Reset_Tracking(void);
```

（4）下面函数的功能为检测电容传感器的电容值。其中参数：groupOfElements --被测电容传感器结构体指针；counts--测量值被写入的地址。

```
void TI_CAPT_Raw(const struct Sensor* groupOfElements, uint16_t *
counts)
```

（5）下面函数的功能为测量传感器电容值的变化。其中参数：groupOfElements --被测电容传感器结构体指针；deltaCnt--测量值被写入的地址。

```
void TI_CAPT_Custom(const struct Sensor* groupOfElements, uint16_t *
deltaCnt)
```

（6）下面函数的功能为确定单个电容触摸按键的按下状态。其中参数：groupOfElements--被扫描按键结构体指针。若按键被按下则返回 1；若按键未被按下则返回 0。

```
uint8_t TI_CAPT_Button(const struct Sensor * groupOfElements)
```

（7）下面函数的功能为确定一列按键中哪一个被按下。其中参数：groupOfElements--被扫描按键结构体指针。返回按键按下（1）或未被按下（0）的常量结构体。

```
const struct Element *TI_CAPT_Buttons(const struct Sensor
*groupOfElements)
```

（8）下面函数的功能为确定所按滑块的位置。其中参数：groupOfElements--指向滑块的指针。若被按下，则返回滑块的位置；若未被按下，则返回错误的值。

```
uint16_t TI_CAPT_Slider(const struct Sensor* groupOfElements)
```

（9）下面函数的功能为确定具有最大响应的按键。其中参数：groupOfElements--被扫描按键结构体指针；deltaCnt--测量值被写入的地址。返回主导按键的标志。

```
uint8_t Dominant_Element(const struct Sensor* groupOfElements,
uint16_t* deltaCnt)
```

您还可以通过以下链接下载电容触摸软件库：http://www.ti.com/tool/capsenselibrary#1。

4.4 实验内容

本章实验包括以下三个小实验：（1）触摸滑块演示实验；（2）触摸按键柱形图演示实验；（3）simon 游戏实验。

实验 2 主函数 lab2()的整体程序流程图如图 4.3 所示：

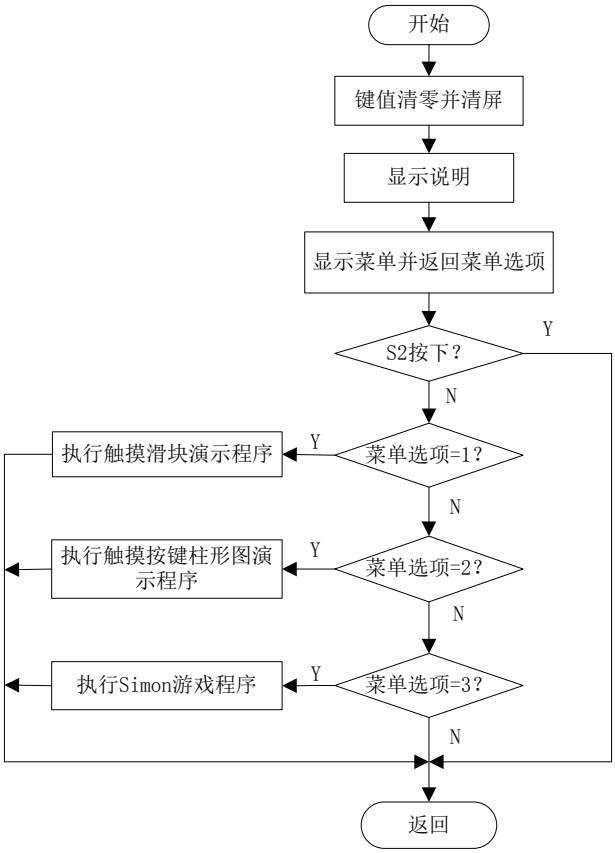


图 4.3 触摸按键应用实验整体程序流程图

4.5 实验原理

（1）比较器B

比较器 B 是一个实现模拟电压比较的外围模块，广泛应用于工业仪表、手持式仪表等产品中，可以实现多种测量功能，如测量电流、电压、电阻、电容、电池检测以及产生外部模拟信号，也可结合其他模块实现精确的 A/D 转换功能，MSP430F5529 单片机的比较器 B 包含多达 16 个通道的比较功能，其具有以下特性：

◆反向和同相端输入多路复用器；
◆比较器输出可编程 RC 滤波器；
◆输出提供给定时器 A 捕获输入；
◆端口输入缓冲区程序控制；
◆中断能力；
◆可选参考电压发生器、电压滞后发生器；
◆外部参考电压输入；
◆超低功耗比较器模式；

◆中断驱动测量系统--支持低功耗运行。

比较器 B 框图如图 4.4 所示，更多应用及操作请参考 MSP430F5529 用户指导。

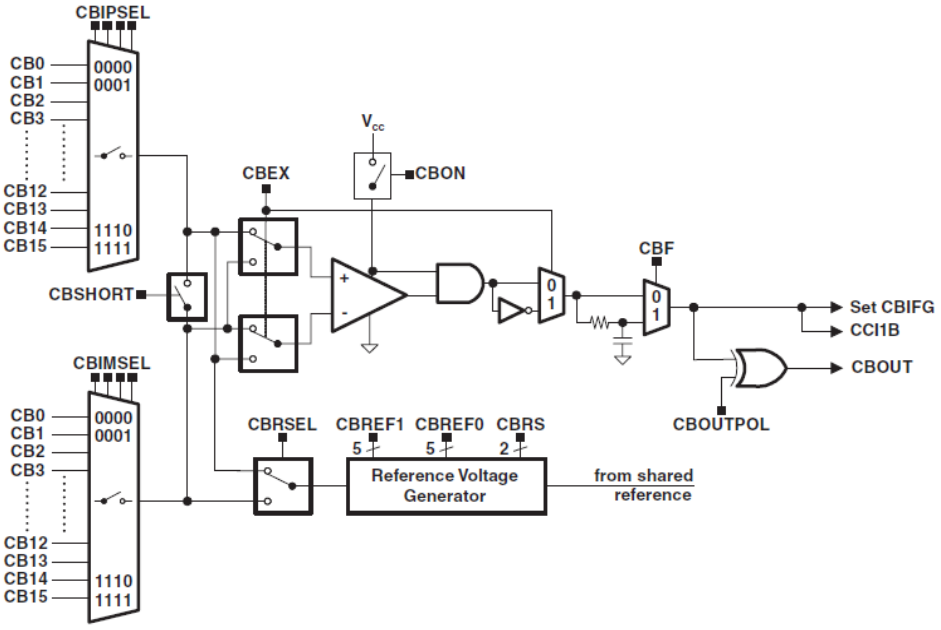


图 4.4 比较器 B 框图

在本实验中，利用 TI_CTS_fRO_COMPB_TA1_SW_HAL()函数（在 CTS\CTS_HAL.c 文件内 1469 行，或者在调试时，对电容触摸初始化函数单步调试进入该段程序）对比较器进行设置。

```
void TI_CTS_fRO_COMPB_TA1_SW_HAL(const struct Sensor *group,uint16_t
*counts)
{
   ……
    CBCTL2 = CBREF14+CBREF13 + CBREF02;
    CBCTL1 = CBON + CBF;              // 开启比较器B，比较器输出经过滤波器
    CBCTL3 |= (group->cbpdBits);              // 禁用CBPD位
    CBCTL2 |= CBRS_1;                        //打开参考电压
    ……
   CBCTL0 = CBIMEN + (group->arrayPtr[i])->inputBits;//启用模拟输入通道
    ……
}
```

（2）电容触摸按键原理

首先，人体是具有一定电容的。当我们把 PCB 上的铜画成如图 4.5 形式的时候，就完成了一个最基本的触摸感应按键。

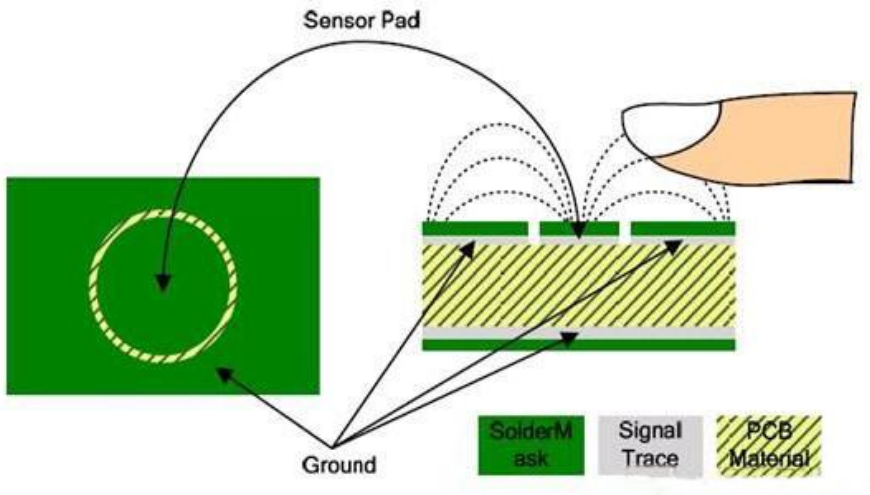


图 4.5 触摸按键结构图

图 4.5 中左边的图，是一个基本的触摸按键，中间圆形绿色的为铜（我们可以称之为“按键”），在这些按键中会引出一根导线与 MCU 相连，MCU 通过这些导线来检测是否有按键“按下”（检测的方法将在后面介绍）；外围的绿色也是铜，不过外围的这些铜是与 GND 大地相连的。在“按键”和外围的铜之间是空隙（我们可以称为空隙 d）。图 4.5 中右边的图是左图的截面图，当没有手指接触时，只有一个电容 Cp ，当有手指接触时，“按键”通过手指就形成了电容 Cf 。由于两个电容是并联的，所以手指接触“按键”前后，总电容的变化率为 C% = ((Cp+Cf)-Cp)/Cp = Cf/Cp。

图 4.6 更简单地说明了上述原理。

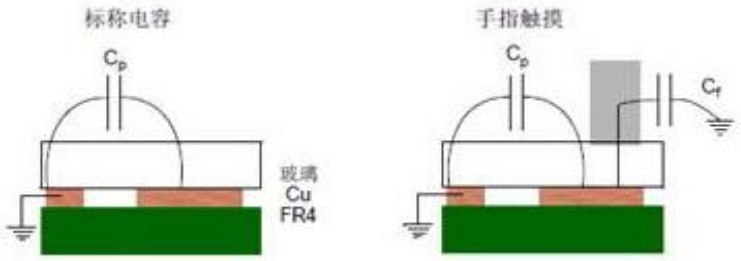


图 4.6 触摸按键等效图

如图 4.7 所示，MSP-EXP430F5529 开发板利用基于张弛振荡器的检测方案实现电容触摸按键的感应。利用比较器 B 实现一个张弛振荡触摸按键的电路如图 4.7 所示，在输入端，比较器正接比较器内部参考电压，比较器负接电阻 Rc 与感应电容之间，CBOUT 与 TACLK 相连。

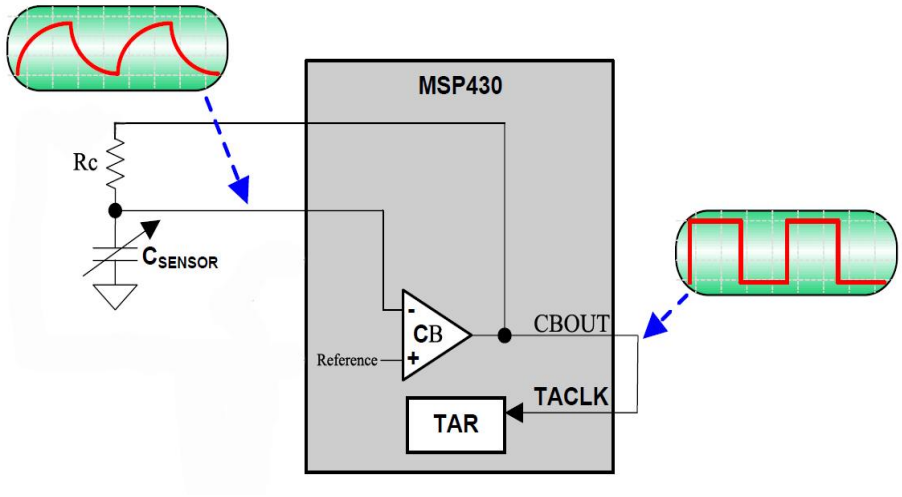


图 4.7 张弛振荡器按键电路

如图 4.8 所示，当手指触摸到电容触摸按键以后，电容会由 C1 变化至 C2，张弛震荡器的输出频率会发生变化，因此只需在固定时间内，利用定时器 A 作为频率计计算张弛振荡器的输出频率，那么如果在某一时刻输出频率有较大的变化的话，那就说明电容值已经被改变，即按键被“按下”了。电容触摸按键的资料可以通过以下链接获得：http://www.ti.com/tool/capsenselibrary。

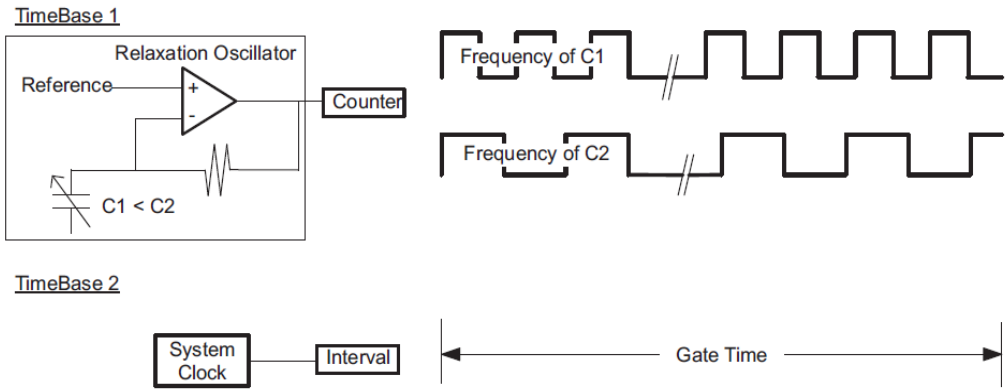


图 4.8 基于张弛振荡器的电容触摸按键检测方法

（3）利用定时器 A 实现频率计的方法

MSP430F5529Timer_A是具有7个捕获/比较寄存器的16位定时/计数器，具有捕获/比较、PWM输出、时间间隔定时等功能，同时具有丰富的中断能力，其具有以下特性：

◆四种工作模式的异步16位定时/计数器；

◆可选择配置的时钟源；

◆多达七个可配置的捕获/比较寄存器；

◆可配置的PWM输出；

◆异步输入和输出锁存；

◆对所有 TA 中断快速响应的中断向量寄存器。

在本实验中，利用 TI_CTS_fRO_COMPB_TA1_SW_HAL 函数（在 CTS\CTS_HAL.c 文件内 1469 行），实现定时器 A 作为频率计，计算张弛振荡器的输出频率，进而测量电容传感器变化的功能：

```
void TI_CTS_fRO_COMPB_TA1_SW_HAL(const struct Sensor *group,uint16_t
*counts)
{
    ……
    for (i = 0; i<(group->numElements); i++)//(group->numElements)=5
    {
        j=0;
        CBCTL0 = CBIMEN + (group->arrayPtr[i])->inputBits;
        TA1CTL = TASSEL_0+TACLR+MC_1; // TA1CLK,计数内容清零, 增计数模式
        TA1CTL &= ~TAIFG;        // 清除中断标志位
        while(!(TA1CTL & TAIFG))
        {
            j++;
        }
        counts[i] = j;        // 将j值进行保存
    }
   ……
}
```

在该函数中，利用 for 语句作了五次循环，分别测量 5 个触摸按键的电容变化。在每次的测量中，将 TA1CLK 作为 TA 的时钟输入，清除计数内容和中断标志位，在计算时钟脉冲个数的过程中，利用 j 值进行累加；当计数完成，TA1 中断标志置位，累加完成，并将 j 值进行存储。j 值可体现 TA1CLK 的频率，也可体现出电容充放电的时间，最终可反映出触摸按键的电容值的变化。

4.6 触摸滑块演示实验

4.6.1 程序代码

该实验的程序代码包含在 lab2.c 文件内：

```
void CapLED(void)
{
    ……
}
```

4.6.2 程序流程

触摸滑块演示实验程序流程图如图 4.9 所示。

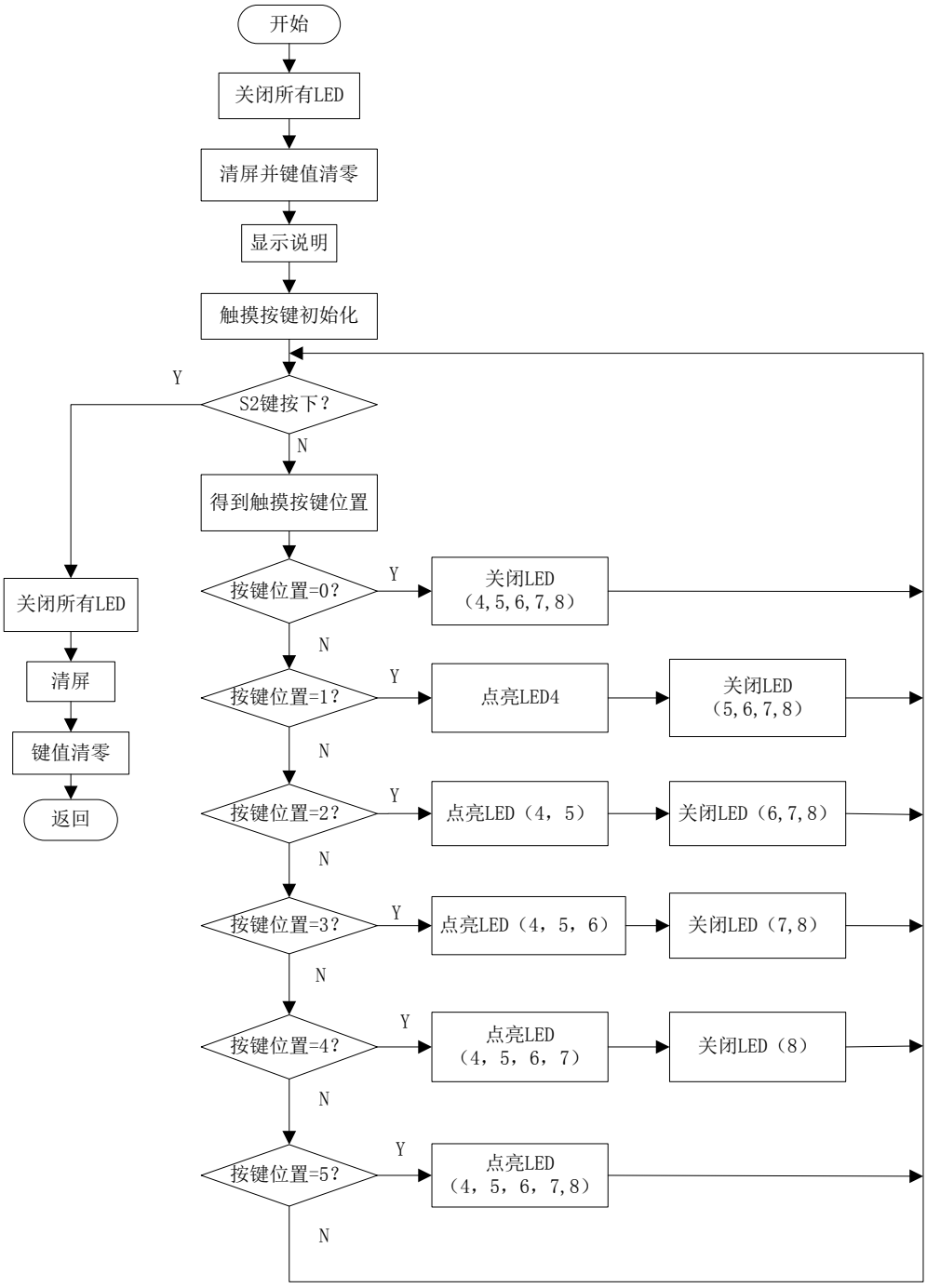


图 4.9 触摸滑块演示实验程序流程图

4.6.3 实验步骤

（若 LAB2 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB2工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab2.c文件，在105行找到该触摸滑块演示实验程序代码CapLED ()，并在其中设置断点，断点位置如图4.10阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：1.CapLED，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图4.10所示。

```
105 void CapLED(void)
106 {
107     uint8_t sliderPosition = 0;
108
109     Board_ledOff(LED_ALL);
110     Dogs102x6_clearScreen();
```

图4.10 触摸滑块演示实验程序开始界面

（7）点击运行按钮，会在 LCD 上观察到：Slide Finger on Touch Pads 字符，将手指在触摸滑块上由左向右滑动，蓝色 LED 会由左向右依次点亮；再将手指在触摸滑块上由右向左滑动，蓝色 LED 会由右向左依次熄灭。将手指触摸任意触摸按键，将会点亮其内部及其左侧的蓝色 LED，并将其右侧蓝色 LED 熄灭。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键 ，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

4.6.4 实验结果

通过本实验，可以实现触摸滑块的功能，并通过相应 LED 进行指示。

4.7 触摸按键柱形图演示实验

4.7.1 程序代码

该实验的程序代码包含在 lab2.c 文件内：

```
void CapDemo(void)
{
    ..........
}
```

4.7.2 程序流程

触摸按键柱形图演示实验程序流程图如图 4.11 所示。

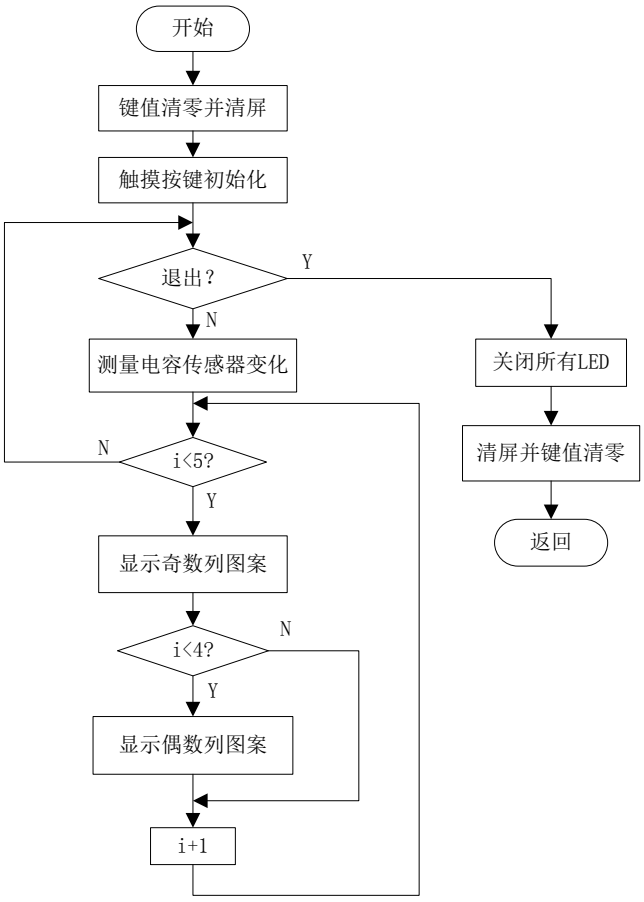


图 4.11 触摸按键柱形图演示实验程序流程图

4.7.3 实验步骤

（若 LAB2 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB2工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab2.c文件，在第158行找到触摸按键柱形图演示实验程序代码CapDemo ()，并在其中设置断点，断点位置如图4.12阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：2.CapDemo，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图4.12所示。

```
158 void CapDemo(void)
159 {
160     uint8_t quit = 0, spacing = 0,
161     uint16_t deltaCount[5];
162
163     Dogs102x6_clearScreen();
```

图4.12 触摸按键柱形图演示实验程序开始界面

（7）点击运行按钮，会在 LCD 上观察到：Touch Demo 字符，将手指在触摸滑块上由左向右滑动，会观察到柱形图会由左向右依次变高；由右向左，类似现象。每个按键与最高柱形图列数依次对应，将手指触摸左边第一个触摸按键，第一列柱形图为最高，由左向右依次触摸，相应最高柱形图的列数为 1、3、5、7、9 列。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

4.7.4 实验结果

通过该实验，能够实现触摸按键的功能，并通过相应柱形图进行显示。

4.8 Simon 游戏实验

4.8.1 程序代码

该实验的程序代码包含在 lab2.c 文件内：

```
void Simon(void)
{
    ..........
}
```

4.8.2 程序流程

Simon 游戏实验程序流程图如图 4.13 所示。

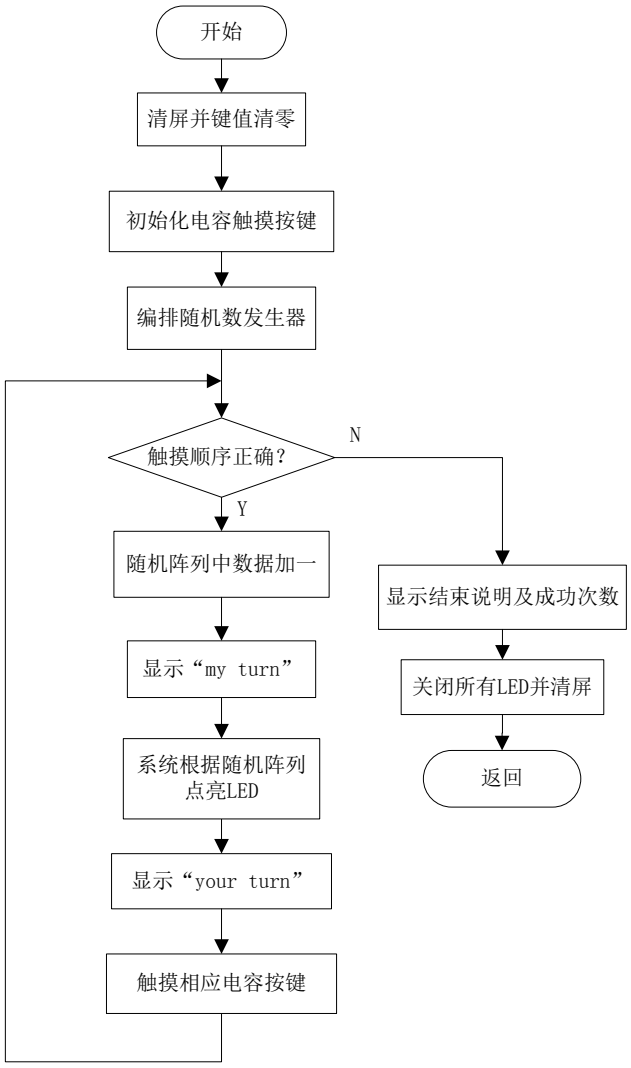


图 4.13 Simon 游戏实验程序流程图

4.8.3 实验步骤

（若 LAB2 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB2工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab2.c文件，在第243行找到该simon游戏实验程序代码simon ()，并在其中设置断点，断点位置如图4.14阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：3.simon，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图4.14所示。

```
243 void simon(void)
244 {
245     uint8_t turn = 0, quit = 0, i;
246     struct Element *user;
247
248     char turndisp[2];
249     //array of Simon moves
250     uint8_t lights[20];
251
252     Dogs102x6_clearScreen();
253     Dogs102x6_stringDraw(0, 0, "
```

图4.14 simon游戏实验程序开始界面

（7）点击运行按钮，首先触摸板上的 LED 会按照一定序列显示，之后实验者须按照同一序列按下正确的触摸按键。游戏开始为一个单一数字的序列，游戏每成功一次，会得到开发板的响应并使序列中数字个数加一，直至实验者输入错误的序列，游戏结束，最后系统显示成功的次数。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

4.8.4 实验结果

通过该实验，实现触摸按键的功能。

第五章 加速度计应用实验

5.1 实验目的

（1）学习 MSP430F5529 单片机 SPI 通讯模式的原理及操作；
（2）学习加速度计模块硬件电路原理；
（3）学习加速度计应用程序资源；
（4）学习加速度计应用实验的操作及编程思想。

5.2 实验所需硬件电路模块介绍

本实验中需用到以下电路模块：点阵 LCD 液晶显示模块、按键输入模块、齿轮电位计采样模块、加速度计模块。现将加速度计模块电路介绍如下：
该三坐标轴加速度计尺寸较小：1.0x2.0x0.95mm^3（宽×长×高），其供电在 1.7V 到 3.6V 之间。在正常测量模式下，它有 400/100/40HZ 的采样频率，分别对应 70/50/11 微安的电流消耗。在运动检测模式下，可以实现采样频率为 10HZ 的 7 微安的电流消耗。该加速度计可以工作在两个不同的测量范围下：±2G 或±8G，并具有 8 位分辨率。加速度计数据手册 https://dl-web.dropbox.com/get/Public/cma3000_d01_datasheet_8277800a.03%5B1%5D.pdf?w=0095bd50。

该加速度计与单片机的连接如图 5.1 所示，它们之间的通讯采用 SPI 模式，利用以下引脚进行实现：ACCEL_SOMI (P3.4/UCA0SOMI)，ACCEL_SIMO (P3.3/UCA0SIMO)，ACCEL_SCK (P2.3/UCA0CLK) 和 ACCEL_CS (P3.5)。由该电路图中可知，该加速度计由 ACCEL_PWR (P3.6)进行供电，所以单片机能够控制该加速度计的活动状态。根据 ACCEL_INT (P2.5)引脚能够获得以下事件中断：

•正常测量模式：当有数据更新时，ACCEL_INT 引脚提供一个中断；
•自由落体检测模式：当检测到有自由落体情况发生时，ACCEL_INT 引脚提供一个中断；
•运动检测模式：当检测到有运动发生时，ACCEL_INT 引脚提供一个中断。

加速度计详细资料可以参考：https://dl-web.dropbox.com/get/Public/cma3000-d0x_product_family_specification_8281000a.04%5B1%5D.pdf?w=d1ad9c1b 。

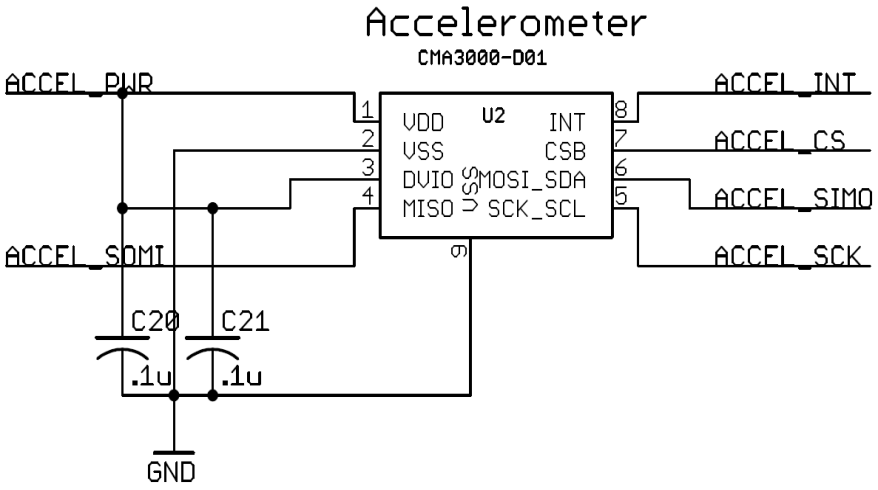


图 5.1 三坐标轴加速度计连接电路

加速度计引脚定义如下（在 Board_init()函数中）：

```
    P2DIR &= ~BIT5;                 // ACCEL_INT 引脚作为输入
    P2OUT &= ~BIT7;                 // ACCEL_SCK引脚输出拉低
    P2DIR |= BIT7;
    P3OUT &= ~(BIT3 + BIT5 + BIT6); // ACCEL_SIMO, ACCEL_CS, ACCEL_PWR
输出拉低
    P3DIR &= ~BIT4;                 // ACCEL_SOMI 引脚作为输入
    P3DIR |= BIT3 + BIT5 + BIT6;    // ACCEL_SIMO, ACCEL_CS, ACCEL_PWR
作为输出
```

5.3 程序资源介绍

5.3.1 HAL_Cma3000.h/.c 程序介绍

在 MSP-EXP430F5529 开发板实验三程序代码文件夹（MSP-EXP430F5529 LAB CODE\LAB3）中包含一个名为 MSP-EXP430F5529_HAL 的硬件模块程序资源库，其中包含本实验所需的程序资源：

HAL_Cma3000.h/.c --加速度计功能管理程序。现将其介绍如下：

（1）X 轴、Y 轴、Z 轴加速度值变量定义程序段。

```
extern int8_t Cma3000_xAccel;
extern int8_t Cma3000_yAccel;
extern int8_t Cma3000_zAccel;
```

（2）下面函数的功能为三坐标轴加速度计初始化。

```
extern void Cma3000_init(void);
```

（3）下面函数的功能为禁用三坐标轴加速度计。

```
extern void Cma3000_disable(void);
```

（4）下面函数的功能为从加速度计中读取各轴加速度值。

```
extern void Cma3000_readAccel(void);
```

（5）下面函数的功能为获取加速度值偏移量，并将各轴偏移量进行存储，用于加速度计的校正。其中参数：xAccel_offset--X 轴偏移量；yAccel_offset--Y 轴偏移量；zAccel_offset--Z 轴偏移量。

```
extern void Cma3000_setAccel_offset(int8_t xAccel_offset, int8_t
yAccel_offset, int8_t zAccel_offset);
```

（6）下面函数的功能为校正各轴加速度值，在该程序中，将从加速度计中读取的数据减去各轴偏移量，进行替换存储。

```
extern void Cma3000_readAccel_offset(void);
```

（7）下面函数的功能为从加速度计中读取数据。其中参数：Address--要读取的寄存器地址。

```
extern int8_t Cma3000_readRegister(uint8_t Address);
```

（8）下面函数的功能为将数据写入加速度计。其中参数：Address--要写入寄存器的地址；Data--要写入的数据。

```
extern int8_t Cma3000_writeRegister(uint8_t Address, int8_t Data);
```

5.3.2 液晶手动刷新程序介绍

在动态立方体演示实验中，将用到液晶手动刷新的程序函数，本章实验将其放置在HAL_Dogs102x6.c 文件内。

与其他章 HAL_Dogs102x6.c 文件相比，本章该文件改动了两个函数：（1）增加了手动刷新函数 Dogs102x6_refresh()（HAL_Dogs102x6.c 文件 1438 行），其中参数：mode--1：允许液晶即时刷新；0：手动刷新，该程序在首次调用时，液晶禁止刷新，在第二次调用时，液晶进行刷新。

```
void Dogs102x6_refresh(uint8_t mode)
{
  drawmode = DOGS102x6_DRAW_IMMEDIATE;
  Dogs102x6_imageDraw(dogs102x6Memory, 0, 0);
  drawmode = mode;
}
```

（2）改动了通过 SPI 向 LCD 发送数据的函数 Dogs102x6_writeData()（HAL_Dogs102x6.c 文件 429 行），具有两种发送模式：手动发送模式和即时发送模式。其他章该函数只具有即时发送模式。

```
void Dogs102x6_writeData(uint8_t *sData, uint8_t i)
{
    ……
    if (drawmode == DOGS102x6_DRAW_ON_REFRESH)
    {
      ……
    }
    else
    {
      ……
    }
}
```

5.4 实验内容

本章实验包括以下三个小实验：（1）加速度计校准实验；（2）动态立方体演示实验；（3）数字拼图游戏实验。

实验三主函数 lab3()的整体程序流程图如图 5.2 所示：

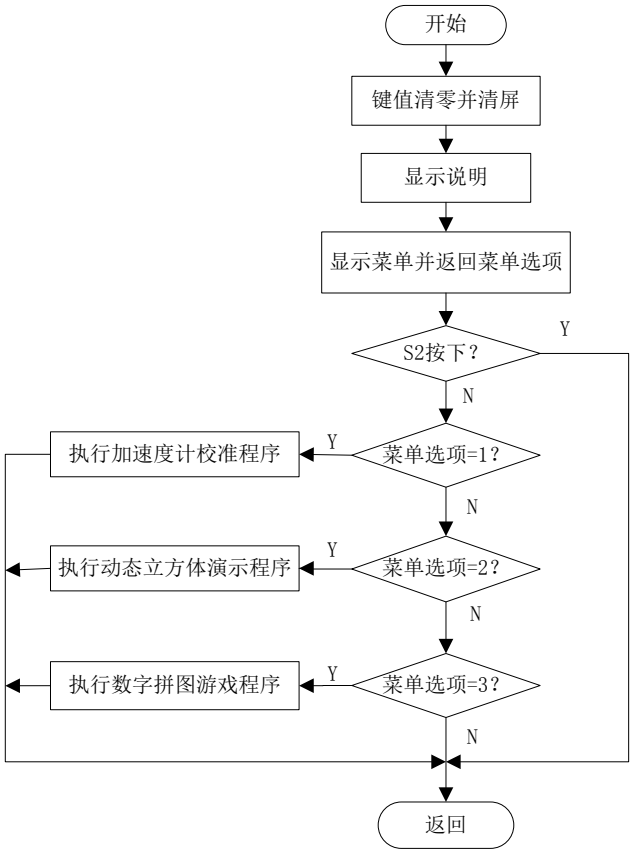


图 5.2 加速度计应用实验整体程序流程图

5.5 实验原理

（1）SPI 模式

串行外围设备接口 SPI 总线技术是一种高速的、全双工、同步的通讯总线。当 MSP430F5529 的 USCI 控制寄存器 UCSYNC 置位且 UCMODEx 位选择 SPI 模式时，串行模块工作在 SPI 模式。它以主从方式工作，这种模式通常有一个主设备和一个或多个从设备，需要至少 4 根线，事实上 3 根也可以（单向传输时），以下是所有基于 SPI 的设备共有的，SIMO、SOMI、SCLK、CS：

●SIMO—主设备数据输出，从设备数据输入；

●SOMI—主设备数据输入，从设备数据输出；

●SCLK—时钟信号，由主设备产生；

●CS—从设备使能信号，由主设备控制。

其中，CS 是控制芯片是否被选中的，也就是说只有片选信号为预先规定的使能信号时（高电位或低电位），对此芯片的操作才有效。这就允许在同一总线上连接多个 SPI 设备成为可能。

下面介绍负责通信的 3 根线：通信是通过数据交换完成的，SPI 是串行通信协议，也就是说数据是一位一位地传输的。这就是 SCLK 时钟线存在的原因，由 SCLK 提供时钟脉冲，SIMO 和 SOMI 则基于此脉冲完成数据传输。数据输出通过 SIMO 线，数据在时钟上升沿或下降沿时改变，在紧接着的下降沿或上升沿被读取。完成一位数据传输，输入也使用同样原理。这样，在至少 8 次时钟信号的改变（上升沿和下降沿为一次），就可以完成 8 位数据的传输。要注意的是，SCLK 信号线只由主设备控制，从设备不能控制信号线。

MSP430F5529 的同步通讯模块有如下特性：

◆7～8 位的数据长度；
◆最高有效位在前或最低有效位在前的数据发送和接收；
◆支持 3 线或 4 线 SPI 操作；
◆支持主机与从机模式；
◆接收和发送有独立的移位寄存器；
◆接收和发送有独立的缓冲寄存器；
◆连续发送和接收；
◆时钟的极性和相位可编程；
◆主模式的时钟频率可编程；
◆接收和发送有独立的中断能力；
◆LPM4 下从模式工作。

图 5.3 为 SPI 模式下的 USCI 框图。

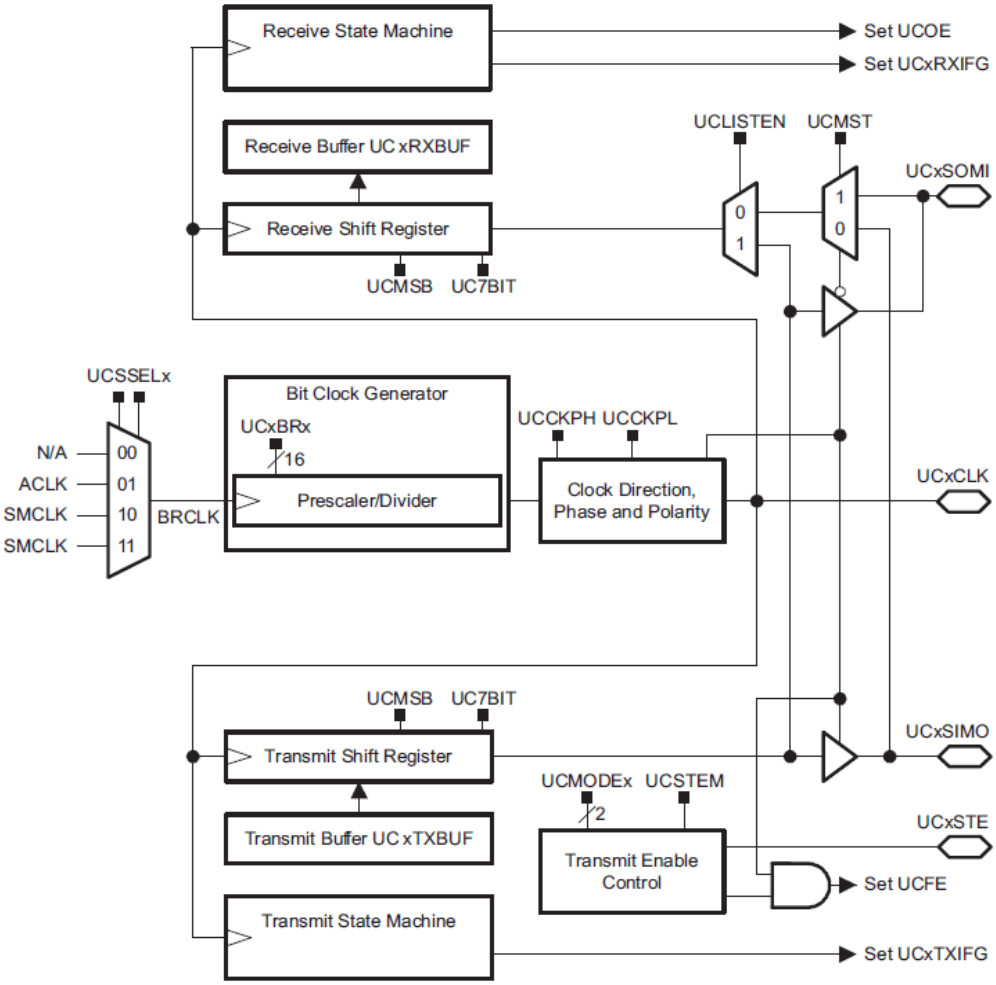


图 5.3 USCI 框图--SPI 模式

在本实验中利用 SPI 模式实现 MSP430F5529 单片机与三坐标轴加速计的通信，使用以下函数进行 SPI 模式的设置：

```
void Cma3000_init(void)
{
        ……
        UCA0CTL1 |= UCSWRST;     //使UCSWRST控制位置位设置串口
        UCA0CTL0 = UCMST + UCSYNC + UCCKPH + UCMSB;//主机模式、同步模式、
时钟极性为高、MSB先发
        UCA0CTL1 = UCSWRST + UCSSEL_2;//使用SMCLK
        UCA0BR0 = 0x30;
        UCA0BR1 = 0;
        UCA0MCTL = 0;//波特率调整控制寄存器清零
        UCA0CTL1 &= ~UCSWRST;//设置结束，拉低UCSWRST
        ……
}
```

在该函数中将 USCI 模块配置为 SPI 主机模式，与三坐标轴加速度计从设备连接如图 5.4 所示。在主机发送端，当数据被传送到发送缓冲寄存器 UCxTXBUF 中后，USCI 立即开始数据发送。如果发送移位寄存器为空，UCxTXBUF 中的数据被传送到发送移位寄存器中，其中以最高有效位或最低有效位先发送，这取决于 UCMSB 的设置。在相反的时钟边沿，UCxSOMI 传输线上的数据被移入到接收移位寄存器中。若收到数据，接收到的数据会从接收移位寄存器移到接收缓冲寄存器 UCxRXBUF 中，并且置位接收终端标志位 UCRxIFG，表明接收/发送操作完成。

发送标志位置位表明数据已经从 UCxTXBUF 中移到发送移位寄存器中，并且 UCxTXBUF 准备发送新的数据，并不表示接收/发送操作完成。用户程序可以使用接收中断标志和发送中断标志完成协议的控制。在主机模式下，USCI 在接收数据的同时，发送的数据必须写入 UCxTXBUF，因为接收和发送操作是同时进行的。

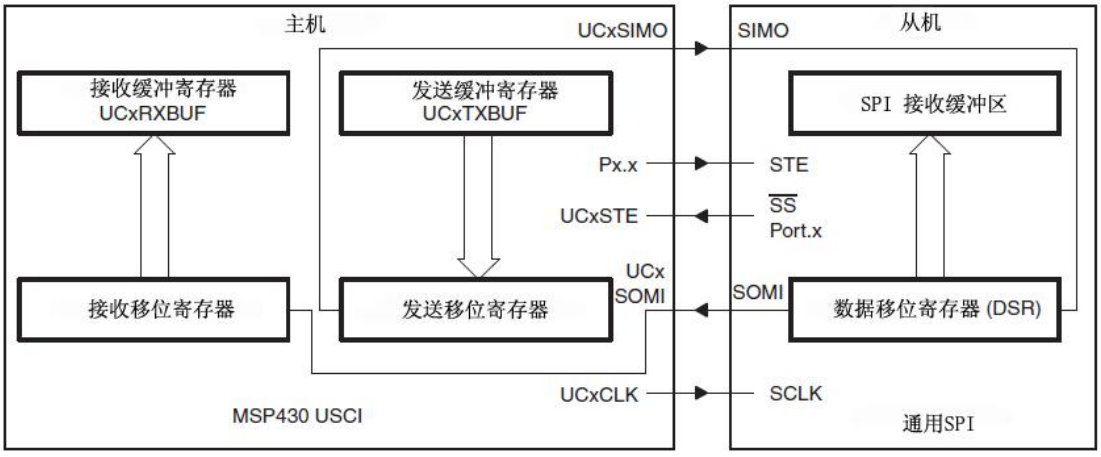


图 5.4 SPI 主机模式

USCI 模块串行时钟 UCxCLK 的极性和相位由 UCCKPL 和 UCCKPH 位控制，最高有效位先发送时的时序如图 5.5 所示。

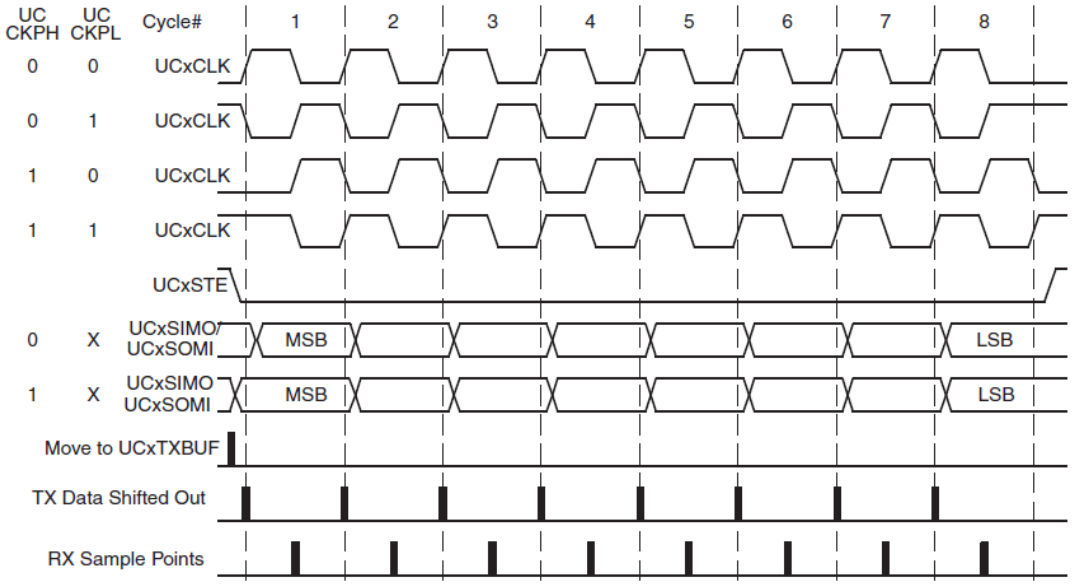


图 5.5 SPI 时序（UCMSB=1）

5.6 加速度计校准实验

5.6.1 程序代码

该实验的程序代码包含在 lab3.c 文件内：

```
void CalibrateAccel(void)
{
    ..........
}
```

5.6.2 程序流程

加速度计校准实验程序流程图如图 5.6 所示。

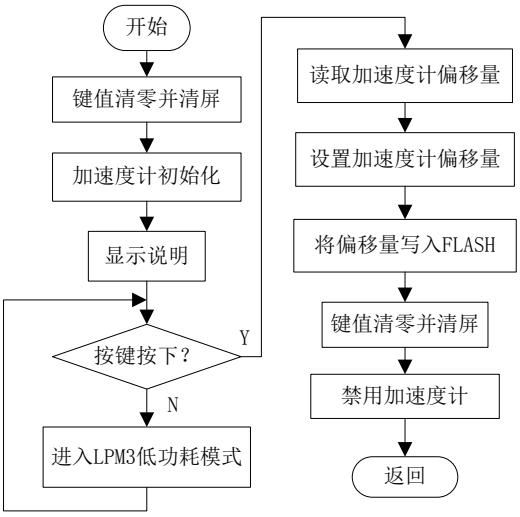


图 5.6 加速度计校准实验程序流程图

5.6.3 实验步骤

（若 LAB3 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB3工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab3.c文件，在第107行找到该加速度计校准实验程序代码CalibrateAccel ()，并在其中设置断点，断点位置如图5.7阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：1.Calibrte Accel，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图5.7所示。

```
107 void CalibrateAccel(void)
108 {
109     uint8_t i;
110
111     uint16_t Cma3000_xAccel_offset;
112     uint16_t Cma3000_yAccel_offset;
113     uint16_t Cma3000_zAccel_offset;
114
115     static const char *const AccelCalibrateText[] = {
116         "**    Accel    **",
117         "*  Calibration  *",
118         "",
119         " Place Board on  ",
120         "  Flat Surface   ",
121         "",
122         "",
123         "    Press S1     "
124     };
125
126     buttonsPressed = 0;
127     Dogs102x6_clearScreen();
```

图5.7 加速度计校准实验程序开始界面

（7）点击运行按钮，会在 LCD 上观察到加速度计校准的说明：Place Board on Flat Surface（将开发板放到平坦的表面），若开发板放置完成，按下 S1 完成校准，并退出校准程序。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

5.6.4 实验结果

通过本实验，可以完成加速度计的校准，并将校准后的参数存入内存，供其他实验所用。

5.7 动态立方体演示实验

5.7.1 程序代码

该实验的程序代码包含在 UserExperienceDemo→ Cube.c 文件内：

```
void Cube(void)
{
    ..........
}
```

5.7.2 程序流程

动态立方体演示实验程序流程图如图 5.8 所示。

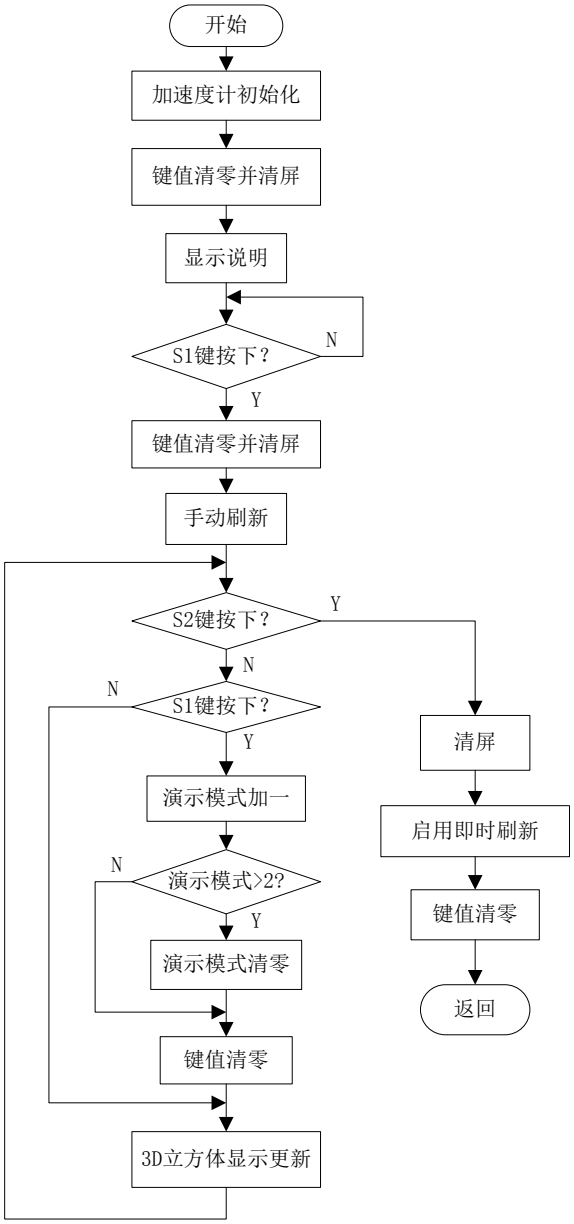


图 5.8 动态立方体演示实验程序流程图

5.7.3 实验步骤

（若 LAB3 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB3工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）展开UserExperienceDemo文件夹，双击打开Cube.c文件，在第161行找到该动态立

方体演示实验程序代码Cube ()，并在其中设置断点，断点位置如图5.9阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：2.Demo Cube，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图5.9所示。

```
161 void Demo_Cube(void)
162 {
163     unsigned int mode = 0;
164
165     memset(cube_data_last, 0, sizeof(cube_data_last));
166
167     Cma3000_init();                          // Initialize
```

图5.9 动态立方体演示实验程序开始界面

（7）点击运行按钮，程序等待 S1 键按下，若按下 S1 键，则开始动态立方体演示。该动态立方体演示实验，有三种模式，每次按下 S1 键，将使演示模式加一，首先进入的是演示模式 0：在液晶 LCD 上将会显示一个旋转的透明动态立方体，该动态立方体的旋转速度不受开发板加速度计的控制，即无论如何倾斜开发板，该透明动态立方体的旋转规律相同；若按下 S1 键，将进入演示模式 1：在液晶 LCD 上将会显示一个旋转的实心动态立方体，其旋转规律与模式 0 相同；若再次按下 S1 键，将进入演示模式 2：在液晶 LCD 上将会显示一个旋转的透明动态立方体，该动态立方体的旋转速度受开发板加速度计的控制，若将开发板放置在实验 5.6 所校准的平面上的话，动态立方体的旋转速度会很慢，缓慢倾斜开发板，会观察到，动态立方体的旋转速度随着倾斜角度的增大而迅速变快，再次按下 S1 键，将会进入模式 0。在实验的过程中，按下 S2 键，将会退出本实验。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键 ，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

5.7.4 实验结果

通过本实验，利用动态立方体的旋转速度，来反映加速度计的倾斜状态。

5.8 数字拼图游戏实验

5.8.1 程序代码

该实验的程序代码包含在 UserExperienceDemo→Puzzle→ puzzle.c 文件内：

```
void StartPuzzle(void)
{
    ..........
}
```

5.8.2 程序流程

数字拼图游戏实验程序流程图如图 5.10 所示。

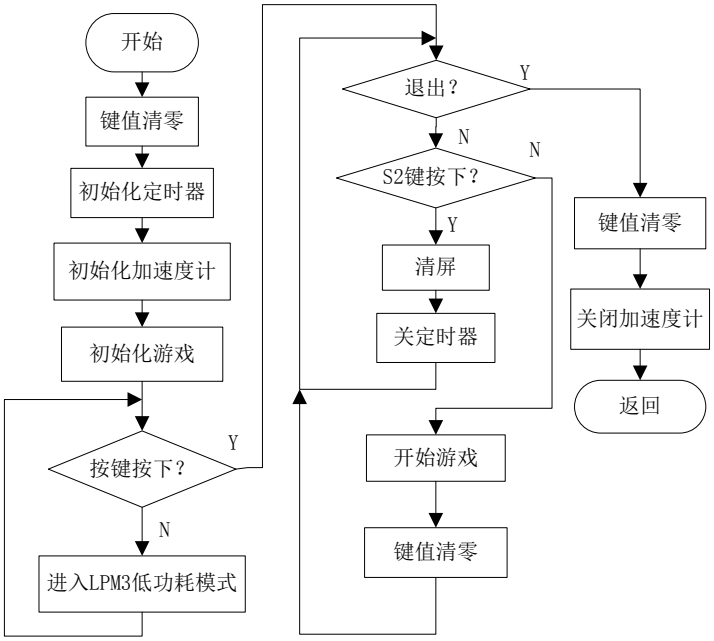


图 5.10 数字拼图游戏实验程序流程图

5.8.3 实验步骤

（若 LAB3 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB3工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）展开UserExperienceDemo文件夹，再展开Puzzle文件夹，双击打开puzzle.c文件，在第39行找到该数字拼图游戏实验程序代码StartPuzzle ()，并在其中设置断点，断点位置如图5.11阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：3.Tilt Puzzle，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图5.11所示。

```
39 void StartPuzzle(void) {
40
41     unsigned int quit = 0;
42
43     buttonsPressed = 0;
44
45     TimerA1Init();
```

图5.11 数字拼图游戏程序开始界面

（7）点击运行按钮，程序等待 S1 键按下，若按下 S1 键，则开始数字拼图游戏。首先

在 LCD 液晶上会看到一个 3×3 的表格，其中有八个数字和一个空的空间。实验者通过倾斜开发板，利用加速度计使数字上下左右移动，若最终使每一行和每一列的数字之和都等于 12，则表示游戏成功。

为了使数字移动更加方便快捷，需要一个比较好的移动方法，现介绍如下：首先利用 5.6 节实验，在标准平面上对加速度计进行校准；然后将开发板放置在该标准平面上，进行游戏；在游戏过程中，每倾斜一次，移动一个数字，之后返回标准平面，准备下次数字移动。

注意：该游戏无解的概率为 50%。在实验的过程中，按下 S2 键，将会退出本实验。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键  ，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

5.8.4 实验结果

通过本实验，利用加速度计实现数字的移动。

第六章 USB 通信实验

6.1 实验目的

（1）学习 MSP430F5529 USB 模块原理；
（2）学习 USB 接口硬件电路原理；
（3）学习 MSP430F5529 USB 程序资源；
（4）学习 USB 通信实验操作及编程思想。

6.2 实验所需硬件电路模块介绍

在本实验中需用到以下电路模块：点阵 LCD 液晶模块、按键输入模块、Mini-USB 接口模块。现将 Mini-USB 接口模块电路介绍如下：

该实验利用 Mini-USB 接口实现 MSP430F5529 单片机与 PC 机的通信，如图 6.1 所示，其引脚连接为：5529_VBUS（VBUS）；PU.1/DM（PU.1/DM）；PU.0/DP（PU.0/DP）；PUR（PUR）；5529_LDO（VUSB）。在①部分电路中，利用 PUR 完成 D+信号的上拉，使主机能够识别当前设备为全速 USB 设备；在②部分电路中，利用 TPD2E001DRLR 芯片提供电流过载保护。

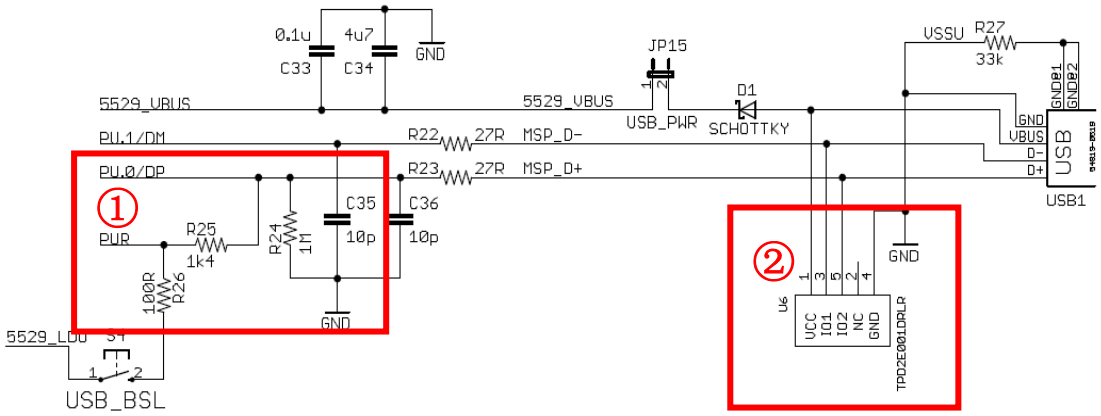


图 6.1 Mini-USB 接口模块电路

6.3 程序资源介绍

在 MSP-EXP430F5529 开发板实验四程序代码文件夹（MSP-EXP430F5529 LAB CODE\LAB4）中包含一个名为 USB 的 USB 开发资源库，其中包含本实验所需的程序资源，USB 程序开发资源库可以通过 MSP430ware 获得，具体的获得途径在 2.5 节中已有介绍；或者通过以下链接进行获得：http://www.ti.com/tool/msp430usbdevpack?DCMP=53xx663x&HQS=msp430usbdevpack-pr-tf。该 USB 开发资源库提供了一套完整的 MSP430 系列 API 库，该 API 库支持三种最常见的设备类型：

（1）通信设备类（CDC）；
（2）人机接口设备类（HID）；
（3）大容量存储类（MSC）；

所有的通信协议都是由 API 自动处理。用户应用程序与 API 之间的接口是非常简单的。应用程序开发之前，用户必须通过 MSP430 USB 描述符工具配置堆栈和 USB 描述符。此外在这个过程中，用户没有必要修改 API 原代码。

在编程环境中，API 被设计成自动适应选定设备。代码保持不变，但需选择正确的设备。三类 USB 设备（CDC/HID/MSC）共用一个 USB 分层，协议栈空间分为 API 空间和应用程序空间。协议栈组织结构图如图 6.2 所示：

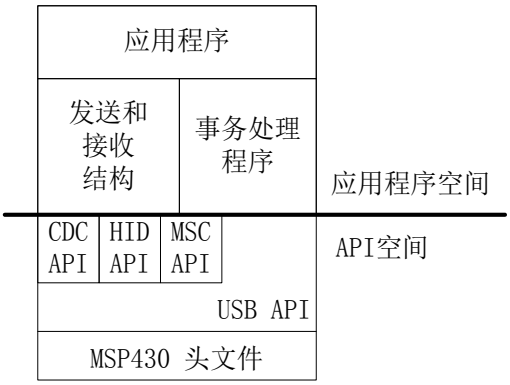


图 6.2 MSP430 USB API 协议栈组织

利用该 USB 资源库，实验者即使不具有很深刻的 USB 知识，也可以容易地完成 USB 程序的开发。USB 文件夹内资源的介绍如表 6.1 所示：

表 6.1 USB 文件描述

文件夹名称	文件名称	描述
USB_API/ USB_CDC_API	UsbCdc.c/.h	CDC 相关功能实现
USB_API/ USB_Common	defMSP430USB.h	定义相关 MSP430 USB 模块
	device.h	控制设备衍生
	types.h	数据类型定义
	usb.h/.c Usblsr.h	所有 USB 应用共有函数
	dma.c	DMA 传输函数
USB_API/ USB_HID_API	UsbHid.h/.c UsbHidReportHandler.h/.c UsbHidReq.h/.c	HID 相关功能实现
USB_API/ USB_MSC_API	UsbMsc.h UsbMscReq.h/.c UsbMscScsi.h/,c UsbMscStateMachine.h/.c	MSC 相关功能实现
USB_API/ USB_PHDC_API	UsbPHDC.h/.c	PHDC 相关功能实现
USB_config	descriptors.h/.c	Descriptors.c 包含了定义 USB 描述符的数据结构，一般情况下，描述符可以通过 MSP430 USB 描述符工具进行自定义设置。 descriptors.h 包含了设置常数和附加描述符信息。
	UsbIsr.c	USB 中断服务处理程序以及相关的函数

USB_User	usbConstructs.h/.c	包含发送/接收操作函数
	UsbMscUser.h/.c	包含 MSC 用户应用程序
	usb_eventHandling.c	事务处理函数

在 API 的使用过程中需要用到 MSP430 的两个功能：USB 模块（及其相关引脚）和一个 DMA 通道。当 USB 模块启用时，应用程序必须避免这些资源被访问。不同的 API 配置所需的内存要求如表 6.2 所示：

表 6.2 不同 API 配置所需的内存要求

接口配置	Code（FLASH）	Data(RAM)
CDC	5.1K	268 字节
HID	5.2K	260 字节
MSC	8.3K	698 字节
CDC+CDC	5.4K	294 字节
HID+HID	5.4K	284 字节
CDC+HID	6.9K	293 字节

以下介绍 USB 资源库中的一些重要程序的功能：

（1）下面函数的功能为确定 USB 连接的状态，返回 USB 的连接状态值。

```
BYTE USB_connectionState ();
```

（2）下面为 USB 连接状态值的定义。

```
#define ST_USB_DISCONNECTED         0x80
#define ST_USB_CONNECTED_NO_ENUM    0x81
#define ST_ENUM_IN_PROGRESS         0x82
#define ST_ENUM_ACTIVE             0x83
#define ST_ENUM_SUSPENDED           0x84
#define ST_ERROR                   0x86
#define ST_NOENUM_SUSPENDED         0x87
```

（3）下面函数的功能为 USB 时钟的初始化。

```
void ClockUSB(void);
```

（4）下面函数的功能为 USB 端口的初始化。

```
BYTE USB_init(VOID);
```

（5）下面函数的功能为使能用户应用程序。

```
BYTE USB_setEnabledEvents (WORD events);
```

（6）下面函数的功能为启动 PLL，使能 USB 模块。

```
BYTE USB_enable ();
```

（7）下面函数的功能为禁用 PLL 和 USB 模块。

```
BYTE USB_disable(VOID);
```

（8）下面函数的功能为重置 USB。

```
BYTE USB_reset ();
```

（9）下面函数的功能为通过拉高 PUR 位，使 USB 设备与主机连接。

```
BYTE USB_connect ();
```

（10）下面函数的功能为通过拉低 PUR 位，使 USB 设备与主机断开连接。

```
BYTE USB_disconnect ()
```

（11）下面函数的功能为使能特定的事务处理程序。

```
BYTE USB_setEnabledEvents (WORD events)
```

（12）下面函数的功能为返回事务启用和禁用的状态。

```
WORD USB_getEnabledEvents ()
```

（13）下面函数的功能为手动进行 USB 的连接或断开。

```
BYTE USB_handleVbusOnEvent()
```

```
BYTE USB_handleVbusOffEvent()
```

注意下面函数中的 xxx 表示 CDC 或 HID。

（14）下面函数的功能为发送或接收数据。

```
BYTE USBxxx_sendData (const BYTE* data, WORD size, BYTE intfNum);
```

```
BYTE USBxxx_receiveData (BYTE* data, WORD size, BYTE intfNum);
```

（15）下面函数的功能为返回接口状态。

```
BYTE USBxxx_intfStatus (BYTE intfNum, WORD* bytesSent, WORD*
bytesReceived);
```

（16）下面函数的功能为返回在 USB 缓冲区中的数据字节数。

```
BYTE USBxxx_bytesInUSBBuffer (BYTE intfNum);
```

（17）下面函数的功能为拒绝接收在 USB 缓冲区中的数据。

```
BYTE USBxxx_rejectData (BYTE intfNum);
```

（18）下面函数的功能为返回 MCU 已经接收到的数据的字节数。

```
WORD xxxReceiveDataInBuffer(BYTE*,WORD,BYTE);
```

（19）下面函数的功能为发送数据，直到所有数据发送完成或总线不可用时停止。

```
BYTE xxxSendDataWaitTilDone(BYTE* dataBuf, WORD size, BYTE intfNum,
ULONG ulTimeout);
```

（20）下面函数的功能为使数据在后台发送。

```
BYTE xxxSendDataInBackground(BYTE* dataBuf, WORD size, BYTE intfNum,
ULONG ulTimeout);
```

（21）下面函数的功能为接收在 USB 缓冲区中的数据。

```
WORD xxxReceiveDataInBuffer(BYTE*,WORD,BYTE);
```

（22）下面函数的功能为提示 API 处理从主机收到的 SCSI 命令。

```
BYTE USBMSC_poll(VOID);
```

（23）下面函数的功能为通知 API 缓冲区请求已经完成。

```
BYTE USBMSC_bufferProcessed(VOID);
```

6.4 实验内容

本章实验包含一个实验：终端显示实验。

6.5 实验原理

MSP430F5529 单片机的 USB 模块具有以下特性：

◆完全符合 USB2.0 规范；

—集成 12Mbps 全速 USB 收发器

—多达 8 个输入和 8 个输出端点

—支持控制、中断和批量传输模式

◆拥有独立于 PMM 模块的电源系统；

—集成了 3.3V 输出的低功耗线性稳压器，该稳压器从 5V 的 VBUS 取电， 输出足以驱动整个 MSP430 工作

—集成 1.8V 低功耗线性稳压器为 PHY 和 PLL 模块供电

—3.3V 输出线性稳压器电流限制功能

◆内部 48MHZ 的 USB 时钟；

—集成可编程锁相环（PLL）

—高度自由化的输入时钟频率，可使用低成本晶振

◆当 USB 模块禁止时；

—缓冲空间被映射到通用 RAM 空间，为系统提供额外的 2KB 的 RAM

—USB 功能引脚变为具有强电流驱动能力的通用 I/O 口

USB 模块的结构框图如图 6.3 所示：

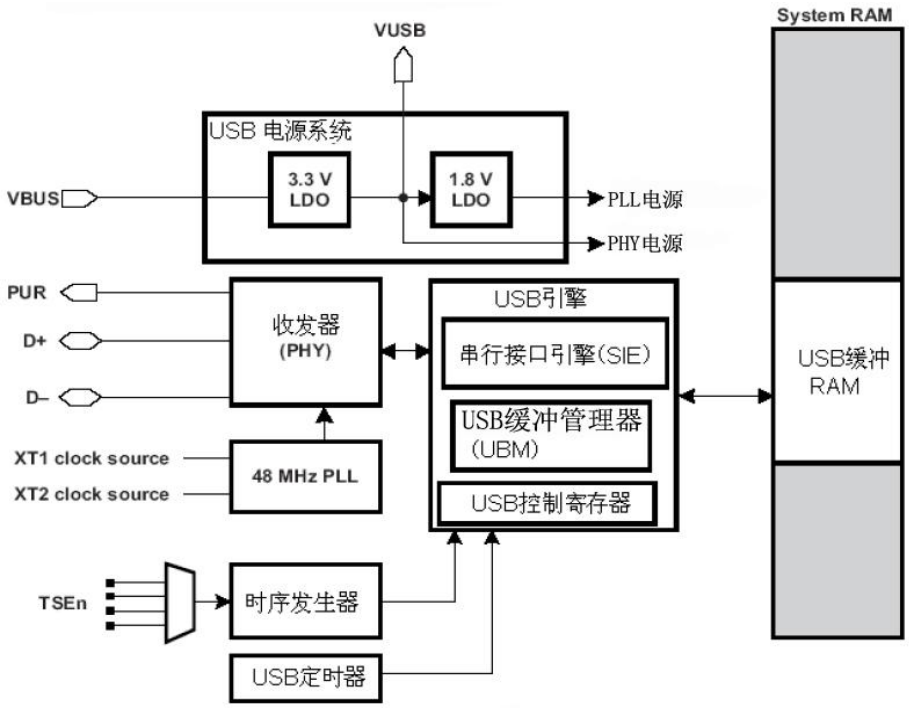


图 6.3 USB 模块框图

PLL 锁相环模块为 USB 操作提供高精度低抖动的 48MHZ 的时钟。PLL 结构框图如图 6.4 所示，如果设备上存在高频晶振 XT2，那么 PLL 的参考时钟频率就为 XT2CLK，无论低频晶振 XT1 是否可用；如果不存在 XT2，那么 PLL 的参考时钟频率就为 XT1CLK。MSP-EXP430F5529 开发板存在可用高频晶振 XT2，因此本实验的 PLL 参考时钟频率为 XT2CLK（4MHZ）。

PLL 锁相环模块输出频率的计算公式为：

$$f_{out} = CLK_{sel} \times \frac{DIVM}{DIVQ}$$

其中：CLK_{sel} 是 PLL 的参考时钟频率，为 4MHZ；

DIVQ 取决于 UPQB 的设置，本实验中，UPQB 设置为 001，DIVQ 值为 2（具体可参考 MSP430F552x 数据手册 P897-- Table 38-1）；

DIVM 取决于 UPMB 的设置，本实验中，UPMB 设置为 010111，DIVM 值为 24（具体可参考 MSP430F552x 数据手册 P897-- Table 38-2）。

因此，本实验中 PLL 锁相环模块输出频率为$f_{out} = 4 \times \frac{24}{2} MHZ = 48MHZ$ 。

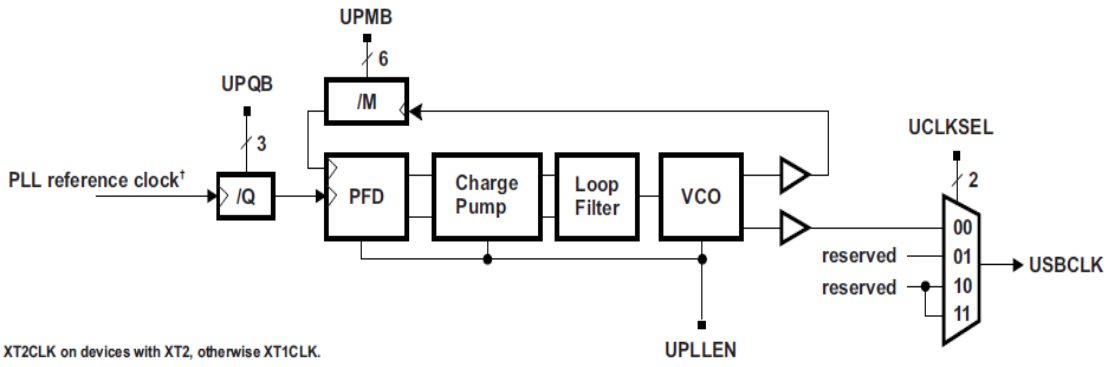


图 6.4 PLL 结构框图

USB 模块的电源系统内含双稳压器（3.3V 和 1.8V），当 5V 的 VBUS 可用时，允许整个 MSP430 从 VBUS 供电。作为可选，USB 模块电源系统可以只为 USB 模块供电，可以为整个系统供电，也可以在一个自供电设备中完全不被使用。USB 模块为整个系统供电时的结构框图，如图 6.5 所示：

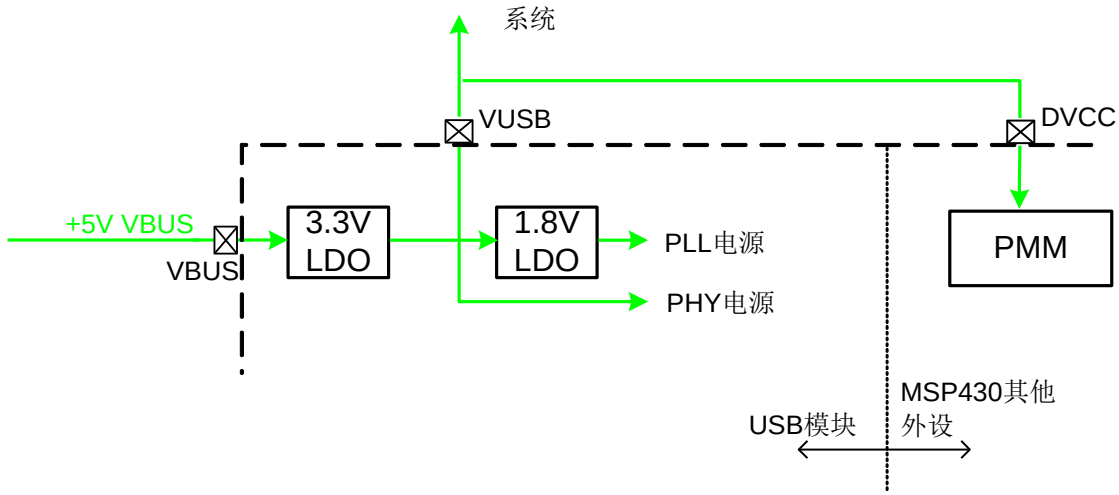


图 6.5 USB 模块为整个系统供电时的结构框图

总体而言，USB 模块与 CPU 及各外设之间的关系可如图 6.6 所示：

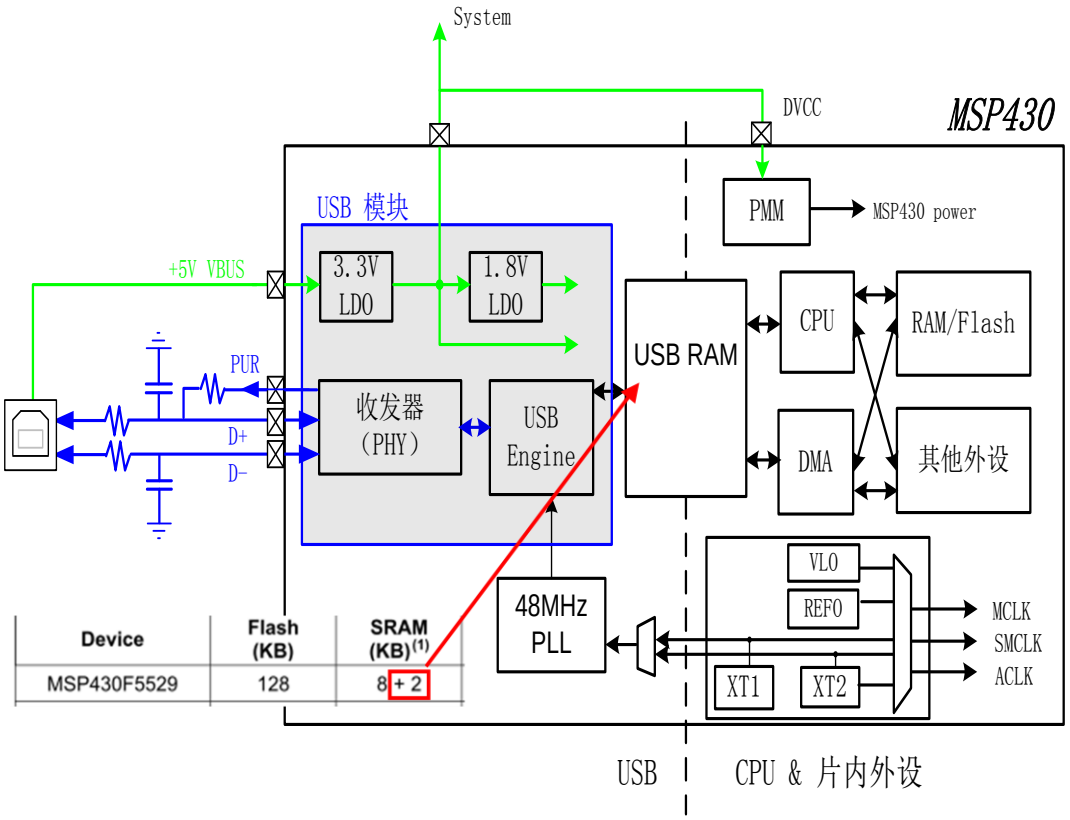


图 6.6 USB 模块与 CPU 及各外设之间的关系框图

MSP430F5529 的 USB 模块支持控制、批量和中断数据传输。按照 USB 传输规范，端点 0 预留为控制端点，该端点为双向传输。除了控制端点以外，USB 模块还能够支持多达 7 个输入端点和 7 个输出端点的数据传输。这些额外的端点可以配置成批量或中断端点。

控制传输：控制传输被用来实现 USB 设备和主机之间配置、命令和状态的通信。控制传输使用输入端点 0 和输出端点 0。控制传输的三种类型是：控制写入、无数据控制写入和控制读取。注意控制端点必须在 USB 设备连接到 USB 主机之前进行初始化。主机采用控制写入传输方式将数据写入 USB 设备。控制写入传输包含设置阶段事务、数据输出阶段事务和状态输入阶段事务。

中断传输/批量传输：USB 模块支持数据以中断/批量传输的方式出入主机。输入端点 1 到 7 和输出端点 1 到 7 都能够被配置为中断/批量端点。

USB 通信最高的传输速率为 12Mbps，这仅仅是理论上的最大值。应用程序无法达到这个速度，因为在通信的过程中有很多因素限制了通信速度，如主机应用程序、总线负载及 USB 端口收到信息的处理过程等。在测试中，CDC API 协议栈在下列情况下可以达到 788KB/sec 的速度：

◆8MHZ 的 CPU MCLK；

◆主机发送数据到 USB 设备；

◆主机应用程序拒绝接收数据。

若主机应用程序不得不接收数据，传输速率可能会降至 200～500KB/sec。可以通过提高时钟频率或使用 DMA 移动数据，来增加传输速度。

本实验为研究 CDC 类型 USB 的通信，MSP430F5529 通过一个虚拟的 COM 端口与主机通信。在 PC 方面，利用超级终端作为上位机软件；在 MSP430 单片机方面，单片机将接收到的数据在 LCD 液晶上进行显示。

6.6 终端显示实验

6.6 .1 程序代码

该实验的程序代码包含在 lab4.c 文件内：

```
void lab4(void)
{
    ……
    ClockUSB();
    USB_init();
    USB_setEnabledEvents(kUSB_allUsbEvents);
    msc_Init();
    if (USB_connectionInfo() & kUSB_vbusPresent)
    {
        if (USB_enable() == kUSB_succeed)
        {
            USB_reset();
            USB_connect();
        }
    }
    while (!(buttonsPressed & BUTTON_S2))
    {
        switch (USB_connectionState())
        {
            case  ST_USB_DISCONNECTED:
                 ……
                break;
            case  ST_USB_CONNECTED_NO_ENUM:
                ……
                break;
            case  ST_ENUM_ACTIVE:
              ……
                break;
              ……
        }
    }
    ……
}
```

请注意该程序代码结构，编程时该程序结构可参考套用。

6.6.2 软件流程

终端显示实验程序流程图如图 6.7 所示。

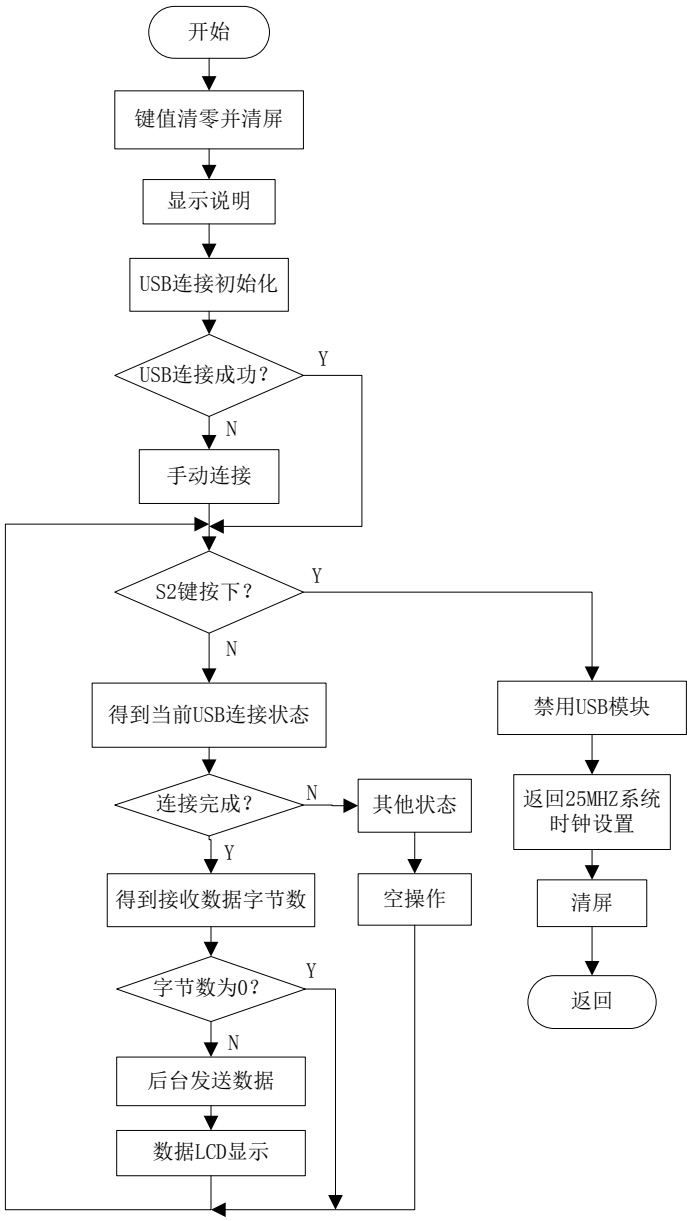


图 6.7 终端显示实验程序流程图

6.6.3 实验步骤

（若 LAB4 工程已导入，（1）（2）（3）步可省略，注意 USB 线连接方法）

（1）将电源选择拨码开关打至eZ档。

（2）利用两根Mini-USB线连接开发板和PC机，连接方法如图6.8所示。

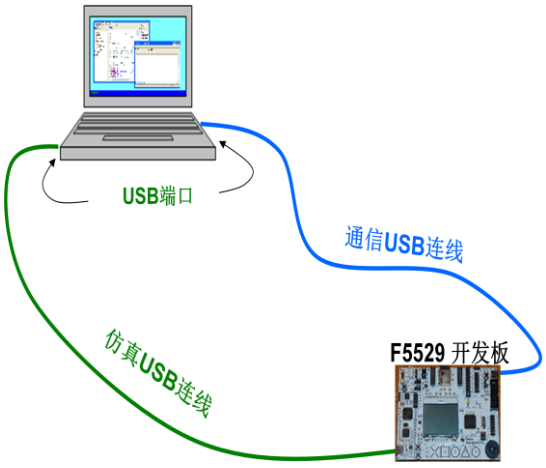


图6.8 终端显示实验USB连线

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB4工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab4.c文件，在第70行找到该终端显示实验程序代码lab4 ()，并在其中设置断点，断点位置如图6.9阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，由于在程序中设置了断点，程序开始的界面，如图6.9所示。

```
70 void lab4(void)
71 {
72     WORD bytesReceived;
73     uint8_t row = 2;
74     uint8_t col = 0;
75     uint8_t i = 0;
76
77     buttonsPressed = 0;
78     Dogs102x6_clearScreen();
```

图6.9 终端显示实验程序开始界面

（7）点击运行按钮，按下S1键进入该实验程序，在桌面右下角会显示“发现新硬件”，之后弹出图6.10窗口，选择“从列表或指定位置安装（高级）（S）”选项。

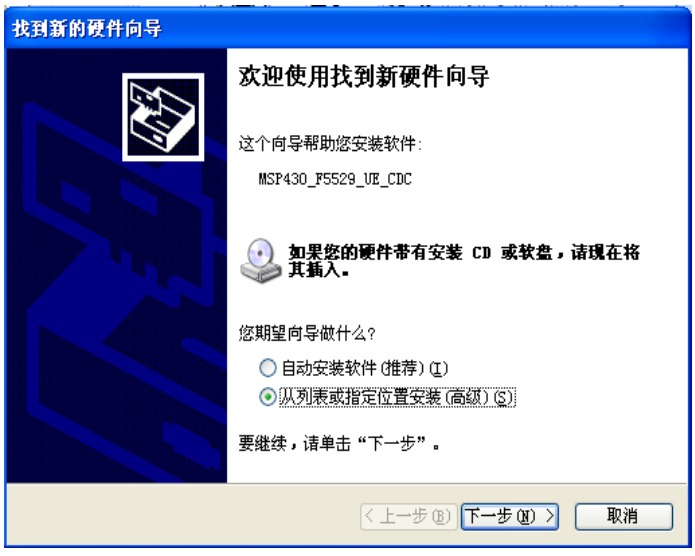


图6.10 硬件驱动安装向导一

（8）单击下一步，会得到图6.11窗口，从浏览中选择硬件驱动所在文件夹的路径：F\MSP-EXP430F5529 \Workspace\ MSP-EXP430F5529 LAB CODE\LAB4\Drivers。

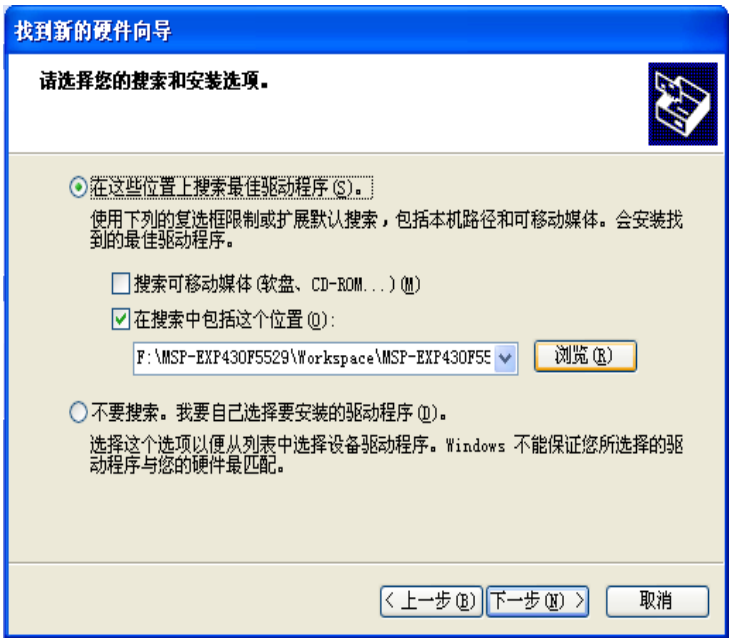


图6.11 硬件驱动安装向导二

（9）单击下一步，会得到图6.12窗口，单击完成，完成硬件驱动的安装。

图6.12 硬件驱动安装向导三

（10）打开设备管理器，查看虚拟的COM端口，如图6.13所示，在此虚拟出的为COM8端口，端口号会由于电脑的不同，而有所不同，但其名称不会改变，请实验者注意。

图6.13 设备管理器

（11）打开XP系统自带的超级终端软件，打开路径为：开始→程序→附件→通讯→超级终端。会弹出如图6.14所示窗口。任意命名都是可以的，在此命名为LAB。

图6.14 超级终端命名窗口

（12）单击确定，会弹出图6.15所示窗口，选择连接时所用端口，在此选择COM8端口（该端口为之前安装开发板驱动，虚拟出的端口）。

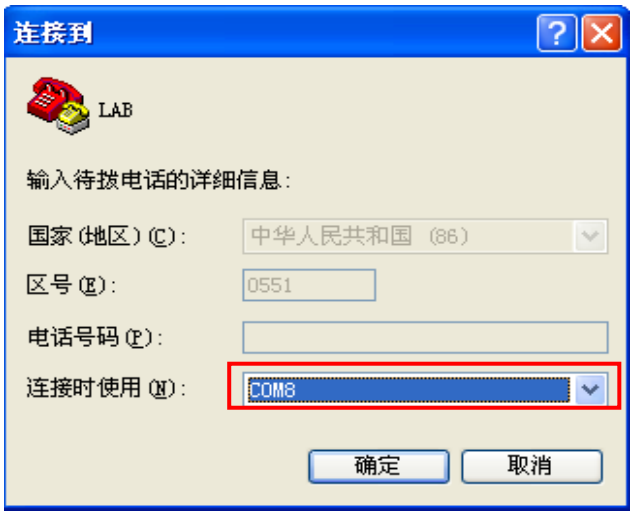


图6.15 连接端口选择窗口

（13）单击确定，会弹出图6.16所示窗口，该端口设置为UART通信时所用，本实验为USB通信，无需设置，仅点击确定按钮，打开COM端口。

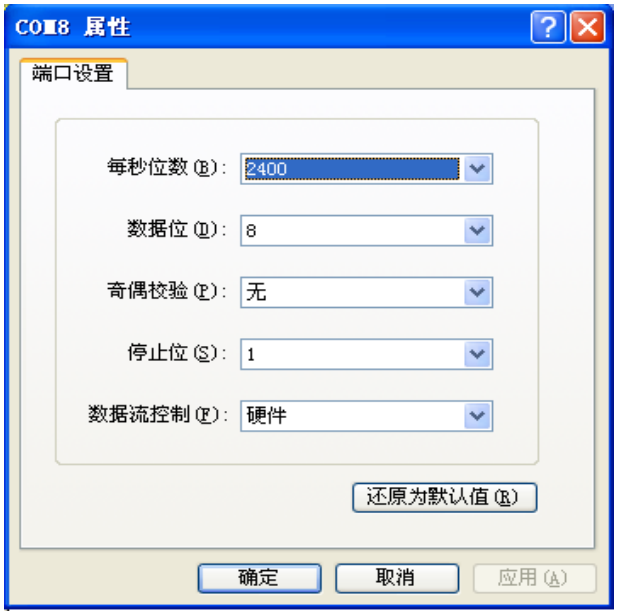


图6.16 端口设置窗口

（14）在超级终端中键入所需通信的字符，如图6.17所示，同时将观察到所键入的字符在液晶LCD上显示，如图6.18所示。

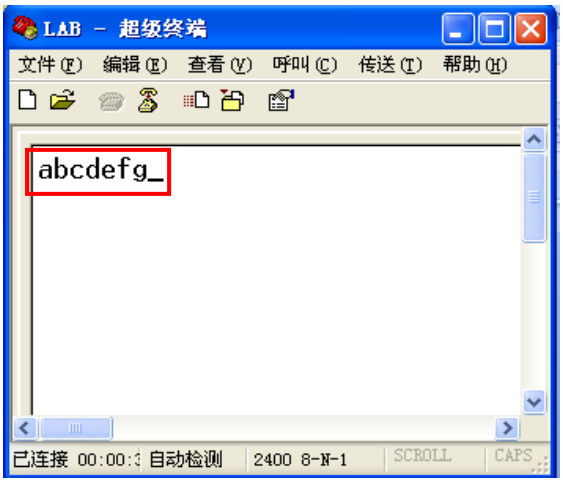


图6.17 超级终端窗口

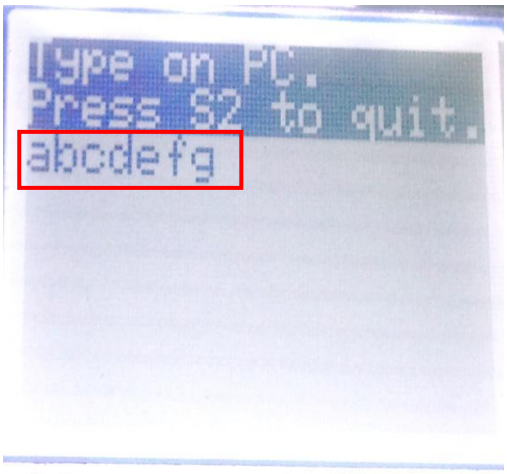


图6.18 液晶LCD显示收到的字符

（15）单击重新开始按钮，重复第（6）步。

（16）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

6.6.4 实验结果

通过该实验实现 PC 机与 MSP430F5529 单片机的通信，MSP430 单片机将收到的数据在 LCD 液晶上显示。

第七章 Micro SD 卡应用实验

7.1 实验目的

（1）学习 SD 卡接口的硬件电路原理；
（2）学习 SD 卡读写程序资源；
（3）学习 SD 卡与 PC 机的通信操作及编程思想；
（4）学习单片机读取 SD 卡信息的操作及编程思想。

7.2 实验所需硬件电路模块介绍

本实验中需用到以下电路模块：按键输入模块、齿轮电位计模块、LED 指示模块、点阵 LCD 液晶显示模块、Mini-USB 接口模块、SD 卡接口模块。现将 SD 卡插槽模块电路介绍如下：

图 7.1 为 SD 卡接口电路，该电路采用 SPI 通信模式实现 SD 卡与单片机之间的数据通信，其引脚连接如下：SD_CS（P3.7），SIMO（P4.1/PM_UCB1SIMO），SCLK（P4.3/PM_UCB1CLK），SOMI（P4.2/PM_UCB1SOMI）。图 7.2 为 SD 卡实物及引脚描述。

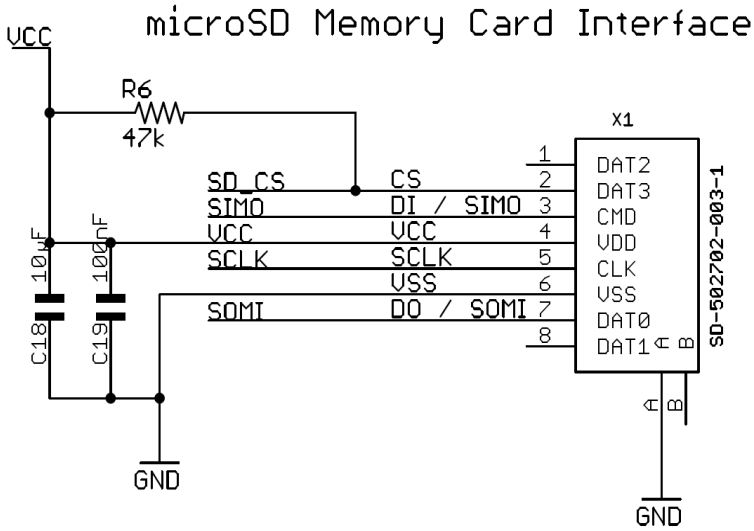


图 7.1 SD 卡接口电路

编号	名称	描述
8	-	保留
7	DO	数据输出
6	VSS	电源地
5	SCLK	时钟
4	VCC	电源
3	DI	数据输入
2	CS	片选
1	-	保留

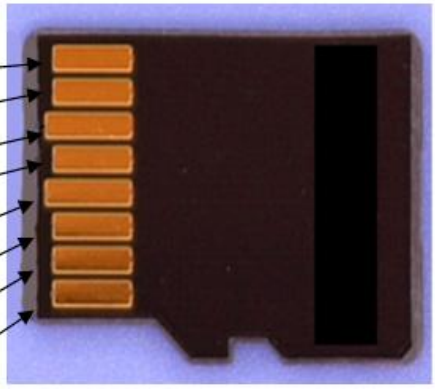


图 7.2 SD 卡实物及引脚描述

7.3 程序资源介绍

在 MSP-EXP430F5529 开发板实验五程序代码文件夹（MSP-EXP430F5529 LAB CODE\LAB5）中包含一个名为 MSP-EXP430F5529_HAL 的硬件模块程序资源库，其中包含本实验所需的程序资源：

HAL_SDCard.h/.c --SD 卡功能管理资源程序，现将其介绍如下。

（1）下面函数的功能为 SD 卡初始化：

```
extern void SDCard_init(void);
```

（2）下面函数的功能为启用快速 SD 卡的 SPI 传输，通常用在初始化函数之后，使数据以最大速度传输。

```
extern void SDCard_fastMode(void);
```

（3）下面函数的功能为通过 SPI 方式读取一个字节的数据，其中参数：*pBuffer--存储接收字节数组的指针；size--接收字节的数量。

```
extern void SDCard_readFrame(uint8_t *pBuffer, uint16_t size);
```

（4）下面函数的功能为通过 SPI 方式发送一个字节的数据，其中参数：*pBuffer--存储发送字节数组的指针；size--要发送字节的数量。

```
extern void SDCard_sendFrame(uint8_t *pBuffer, uint16_t size);
```

（5）下面函数的功能为设置 SD 卡的片选信号为高。

```
extern void SDCard_setCSHigh(void);
```

（6）下面函数的功能为设置 SD 卡的片选信号为低。

```
extern void SDCard_setCSLow(void);
```

7.4 实验内容

本章实验包括以下两个小实验：（1）USB 型 SD 卡读写实验；（2）SD 卡读取显示实验。实验 5 主函数 lab5()的整体程序流程图如图 7.3 所示：

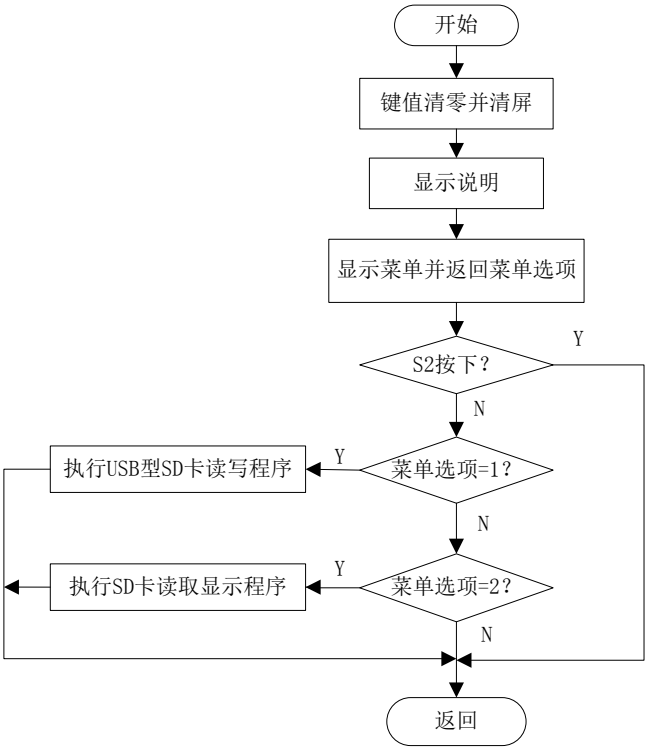


图 7.3 Micro SD 卡应用实验整体程序流程图

7.5 USB 型 SD 卡读写实验

7.5.1 程序代码

该实验的程序代码包含在 UserExperienceDemo→Massstorage.c 文件内：

```
void MassStorage(void)
{
  ..........
}
```

7.5.2 程序流程

USB 型 SD 卡读写实验程序流程图如图 7.4 所示。

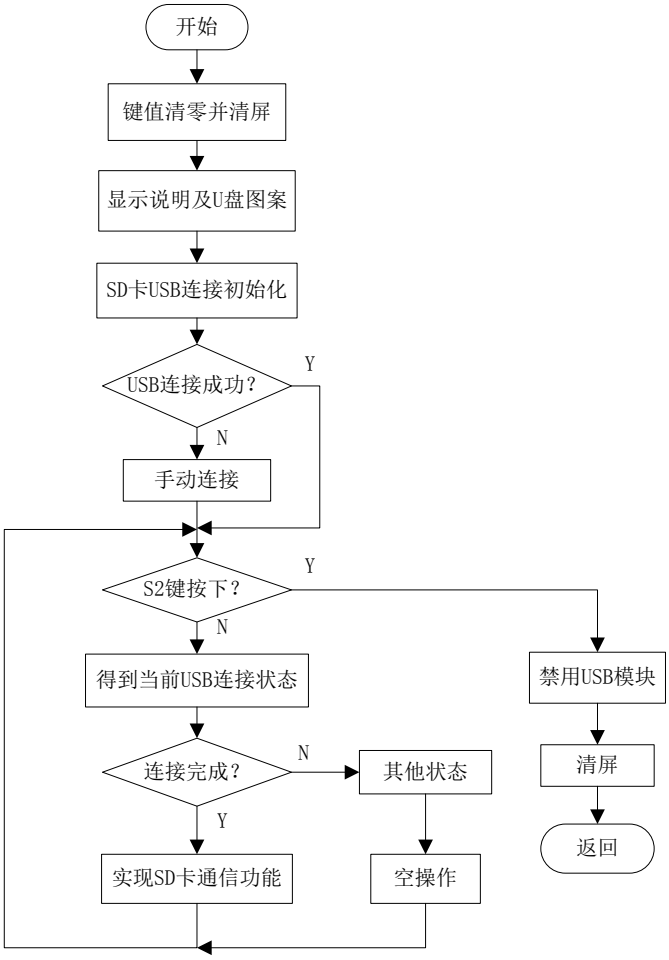


图 7.4 USB 型 SD 卡读写实验程序流程图

7.5.3 实验步骤

（若 LAB5 工程已导入，（1）（2）（3）步可省略，注意 USB 线连接方法）

（1）将电源选择拨码开关打至eZ档。

（2）利用两根Mini-USB线连接开发板和PC机，连接方法如图7.5所示。

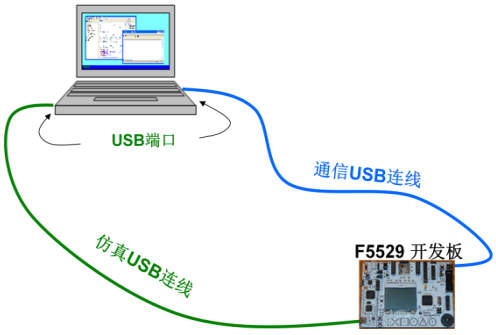


图7.5 USB型SD卡读写实验USB连线

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB5工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）展开UserExperienceDemo文件夹，双击打开Massstorage.c文件，在第125行找到该USB型SD卡读写实验程序代码MassStorage ()，并在其中设置断点，断点位置如图7.5阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：1.USB miscroSD，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图7.6所示。

```
125 void MassStorage(void)
126 {
127     buttonsPressed = 0;
128     Dogs102x6_clearScreen();
129     Dogs102x6_stringDraw(0, 0,
```

图7.6 USB型SD卡读写程序开始界面

（7）点击运行按钮，在桌面右下角将显示硬件连接图标。打开我的电脑，将出现一个可移动的存储设备，该存储设备即为SD卡，如图7.7所示。

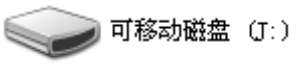


图7.7 可移动磁盘图标

（8）双击打开该可移动磁盘，在其中新建三个文件，如图 8.7 所示，在 1.txt 中任意键入英文字符，在 2.txt 中任意键入中文文字，将以上两个文件复制到 3 文件夹内，供 7.6 实验所用。

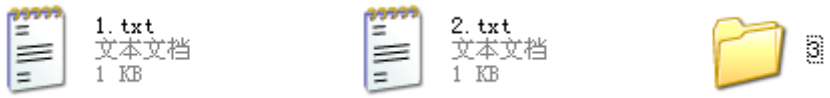


图 7.8 新建文件目录

（9）单击重新开始按钮，重复第（6）步。

（10）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

7.5.4 实验结果

通过本实验，可以利用 USB 通信方式实现 SD 卡的读写。

7.6 SD 卡内存读取显示实验

7.6.1 程序代码

该实验的程序代码包含在 UserExperienceDemo→SDcard.c 文件内：

```
void SDCard(void)
{
  ..........
}
```

7.6.2 程序流程

SD 卡内存读取显示实验程序流程图如图 7.9 所示。

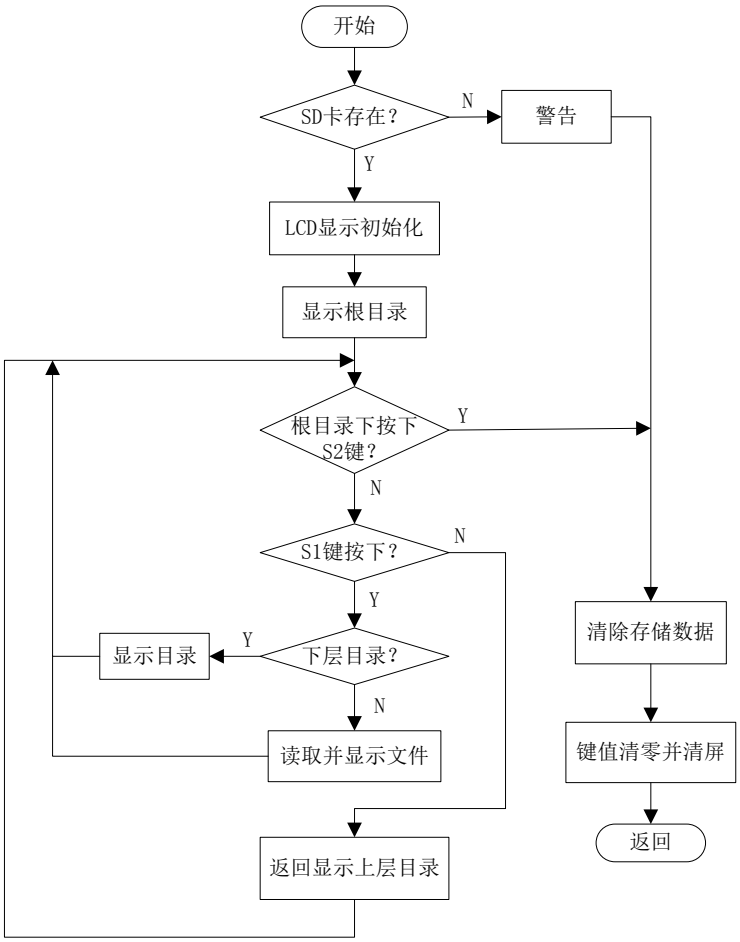


图 7.9 SD 卡内存读取显示实验程序流程图

7.6.3 实验步骤

（若 LAB5 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB5工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）展开UserExperienceDemo文件夹，双击打开SDcard.c文件，在第116行找到该SD卡内存读取显示实验程序代码SDCard ()，并在其中设置断点，断点位置如图7.10阴影部分所示。

（5）将工程编译通过，并点击调试按钮进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：2.SD Card Access，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图7.10所示。

```
116 void SDCard(void)
117 {
118     FRESULT rc;
119     uint8_t a = 0, b = 0;
120     uint8_t level = 0;
121
122     for (a = 0; a < PATH_SIZE; a++)
123         path[a] = 0;
```

图7.10 SD卡内存读取显示程序开始界面

（7）点击运行按钮，将会看到液晶上显示当前SD卡内存的根目录，如图8.10所示。

（8）通过齿轮电位计和按键S1进入1.txt和2.txt文件，可以观察到，液晶LCD可以显示英文字符，却无法显示中文文字。然后进入“3文件夹”，可以观察到“3文件夹”下的目录。

（9）单击重新开始按钮，重复第（6）步。

（10）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

7.6.4 实验结果

通过本实验，单片机可以读取 SD 卡内存，并在液晶 LCD 上显示。

第八章 功耗测试实验

8.1 实验目的

（1）学习 MSP430F5529 单片机低功耗模式的原理及操作；
（2）测试单片机在各个模式下的功耗；
（3）学习功耗测试实验操作及编程思想；

8.2 实验所需硬件电路模块介绍

在本实验中需用到以下电路模块：点阵 LCD 液晶模块、按键输入模块、齿轮电位计模块、电容触摸模块。

8.3 实验内容

本章实验包含一个实验：功耗测试实验。

8.4 实验原理

TI 的 MSP430 是一个特别强调低功耗的单片机系列，通过模块的智能化运行管理和 CPU 的状态组合以先进方式支持超低功耗各种要求。MSP430 系列单片机各个模块运行完全是独立的，定时器、输入/输出端口、A/D 转换、看门狗、液晶显示器等都可以在主 CPU 休眠的状态下独立运行。当需要主 CPU 工作时，任何一个模块都可以通过中断唤醒 CPU，从而使系统以最低功耗运行。这一点是 MSP430 系列单片机最突出的优点，也是与其他单片机的最大区别。

通过设置控制位 SCG1、SCG0、OscOff、CPUOff 可使 MSP430 从活动模式进入到相应的低功耗模式；而各种低功耗模式又可通过中断方式回到活动模式。图 6.1 示出了各种模式之间的关系。

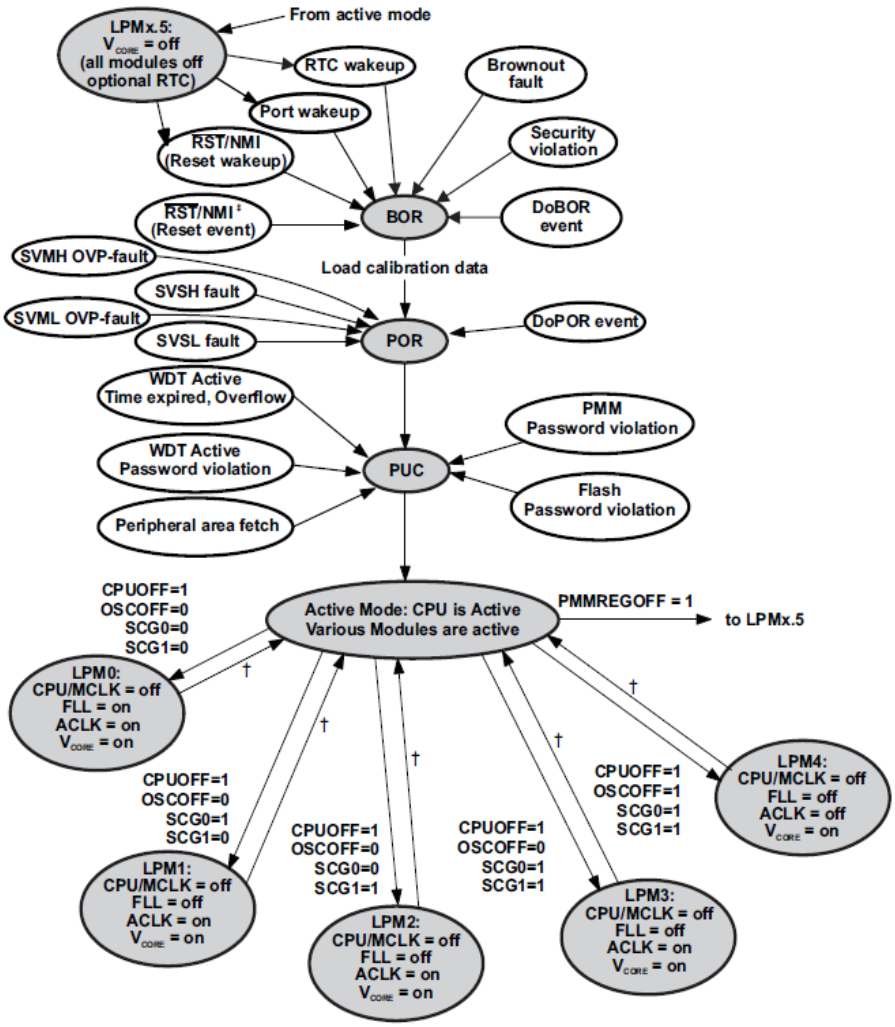


图 8.1 MSP430 工作模式状态图

表 8.1 反映了各种工作模式、各控制位、CPU、时钟活动状态及唤醒中断源之间的相互关系。

表 8.1 MSP430 单片机各工作模式介绍

工作模式	控制位	CPU 和时钟状态	唤醒中断源
活动模式（AM）	SCG1=0 SCG0=0 OscOff=0 CPUOff=0	CPU 活动 MCLK 活动 SMCLK 活动 ACLK 活动 DCO 可用 FLL 可用	定时器、ADC、DMA、UART、WDT、I/O、比较器、外部中断、RTC、串行通信、其他外设
低功耗模式 0（LPM0）	SCG1=0 SCG0=0 OscOff=0 CPUOff=1	CPU 禁止 MCLK 禁止 SMCLK 活动 ACLK 活动 DCO 可用 FLL 可用	定时器、ADC、DMA、UART、WDT、I/O、比较器、外部中断、RTC、串行通信、其他外设
低功耗模式 1（LPM1）	SCG1=0 SCG0=1 OscOff=0 CPUOff=1	CPU 禁止 MCLK 禁止 SMCLK 活动 ACLK 活动 DCO 可用 FLL 禁止	定时器、ADC、DMA、UART、WDT、I/O、比较器、外部中断、RTC、串行通信、其他外设

低功耗模式 2（LPM2）	SCG1=1 SCG0=0 OscOff=0 CPUOff=1	CPU 禁止 MCLK 禁止 SMCLK 禁止 ACLK 活动 DCO 可用 FLL 禁止	定时器、ADC、DMA、UART、WDT、I/O、比较器、外部中断、RTC、串行通信、其他外设
低功耗模式 3（LPM3）	SCG1=1 SCG0=1 OscOff=0 CPUOff=1	CPU 禁止 MCLK 禁止 SMCLK 禁止 ACLK 活动 DCO 禁用 FLL 禁止	定时器、ADC、DMA、UART、WDT、I/O、比较器、外部中断、RTC、串行通信、其他外设
低功耗模式 3.5（LPM3.5）	SCG1=1 SCG0=1 OscOff=1 CPUOff=1	当 PMMREGOFF = 1，无 RAM 保持，RTC 可以启用（仅限 MSP430F5xx）	外部中断、RTC
低功耗模式 4（LPM4）	SCG1=1 SCG0=1 OscOff=1 CPUOff=1	CPU 禁止 所有时钟禁止	外部中断
低功耗模式 4.5（LPM4.5）	SCG1=1 SCG0=1 OscOff=1 CPUOff=1	当 PMMREGOFF = 1，无 RAM 保持，RTC 禁止（仅限 MSP430F5xx）	外部中断

注：MSP430F5529 不具有 LPM3.5 工作模式

表 8.2 反映了活动模式下的 FLASH 执行存储器 MSP430 电流消耗列表，具体请参考 MSP430F552x 数据手册第 50 页。

表 8.2 活动模式下流入 Vcc 的电流（不包含外部电流）

参数	Vcc	VCORE	频率（$f_{DCO}=f_{MCLK}=f_{SMCLK}$）										单位
			1MHZ		8MHZ		12MHZ		20MHZ		25MHZ		
			典型	最大	典型	最大	典型	最大	典型	最大	典型	最大	
I_{AM}	3.0V	0	0.36	0.47	2.32	2.60							mA
		1	0.40		2.65		4.0	4.4					mA
		2	0.44		2.90		4.3		7.1	7.7			mA
		3	0.46		3.10		4.6		7.6		10.1	11.0	mA

表 8.3 反映了各低功耗模式模式下的 MSP430 电流消耗列表，具体请参考 MSP430F552x 数据手册第 51 页。

表 8.3 低功耗模式下流入 Vcc 的电流（不包含外部电流）

参数	Vcc	Vcore	-40℃[1]		25℃[1]		60℃[1]		85℃[1]		单位
			典型	最大	典型	最大	典型	最大	典型	最大	
$I_{LPM0,1MHZ}$	2.2v	0	73		77	85	80		85	97	μA
	3.0V	3	79		83	92	88		95	105	
I_{LPM2}	2.2V	0	6.5		6.5	12	10		11	17	μA
	3.0V	3	7.0		7.0	13	11		12	18	

$I_{LPM3,LFXT1}$ [2]	2.2V	0	1.60		1.90		2.6		5.6		μA
		1	1.65		2.00		2.7		5.9		
		2	1.75		2.15		2.9		6.1		
	3.0V	0	1.8		2.1	2.9	2.8		5.8	8.3	
		1	1.9		2.3		2.9		6.1		
		2	2.0		2.4		3.0		6.3		
		3	2.0		2.5	3.9	3.1		6.4	9.3	
$I_{LPM3,VLO}$ [3]	3.0V	0	1.1		1.4	2.7	1.9		4.9	7.4	μA
		1	1.1		1.4		2.0		5.2		
		2	1.2		1.5		2.1		5.3		
		3	1.3		1.6	3.0	2.2		5.4	8.5	
I_{LPM4}	3.0V	0	0.9		1.1	1.5	1.8		4.8	7.3	μA
		1	1.1		1.2		2.0		5.1		
		2	1.2		1.2		2.1		5.2		
		3	1.3		1.3	1.6	2.2		5.3	8.1	
$I_{LPM4.5}$	3.0V		0.15		0.18	0.35	0.26		0.5	1.0	μA

[1] 单片机运行时所处的典型环境温度；

[2] 在低功耗模式 3 下，选择 LFXT1 为 ACLK 时钟源时的 MSP430 消耗电流；

[3] 在低功耗模式 3 下，选择 VLO 为 ACLK 时钟源时的 MSP430 消耗电流。

与低功耗模式相关的内部函数：

```
__bis_SR_register(LPM0_bits); 或 LPM0; //进入低功耗模式 0（注意开头两个"_"）
__bis_SR_register(LPM1_bits); 或 LPM1;              //进入低功耗模式 1
__bis_SR_register(LPM2_bits); 或 LPM2;              //进入低功耗模式 2
__bis_SR_register(LPM3_bits); 或 LPM3;              //进入低功耗模式 3
__bis_SR_register(LPM4_bits); 或 LPM4;              //进入低功耗模式 4
__bic_SR_register_on_exit(LPM0_bits);或 LPM0_EXIT  //退出低功耗模式 0
__bic_SR_register_on_exit(LPM1_bits);或 LPM1_EXIT  //退出低功耗模式 1
__bic_SR_register_on_exit(LPM2_bits);或 LPM2_EXIT  //退出低功耗模式 2
__bic_SR_register_on_exit(LPM3_bits);或 LPM3_EXIT  //退出低功耗模式 3
__bic_SR_register_on_exit(LPM4_bits);或 LPM4_EXIT  //退出低功耗模式 4
__bis_SR_register(LPMx_bits + GIE);//常用,进低功耗模式 x,启用中断(x=0～4)
```

8.5 功耗测试实验

8.5.1 程序代码

该实验的程序代码包含在 lab6.c 文件内：

```
void lab6(void)
{
    ..........
}
```

8.5.2 软件流程

功耗测试实验程序流程图如图 8.2 所示。

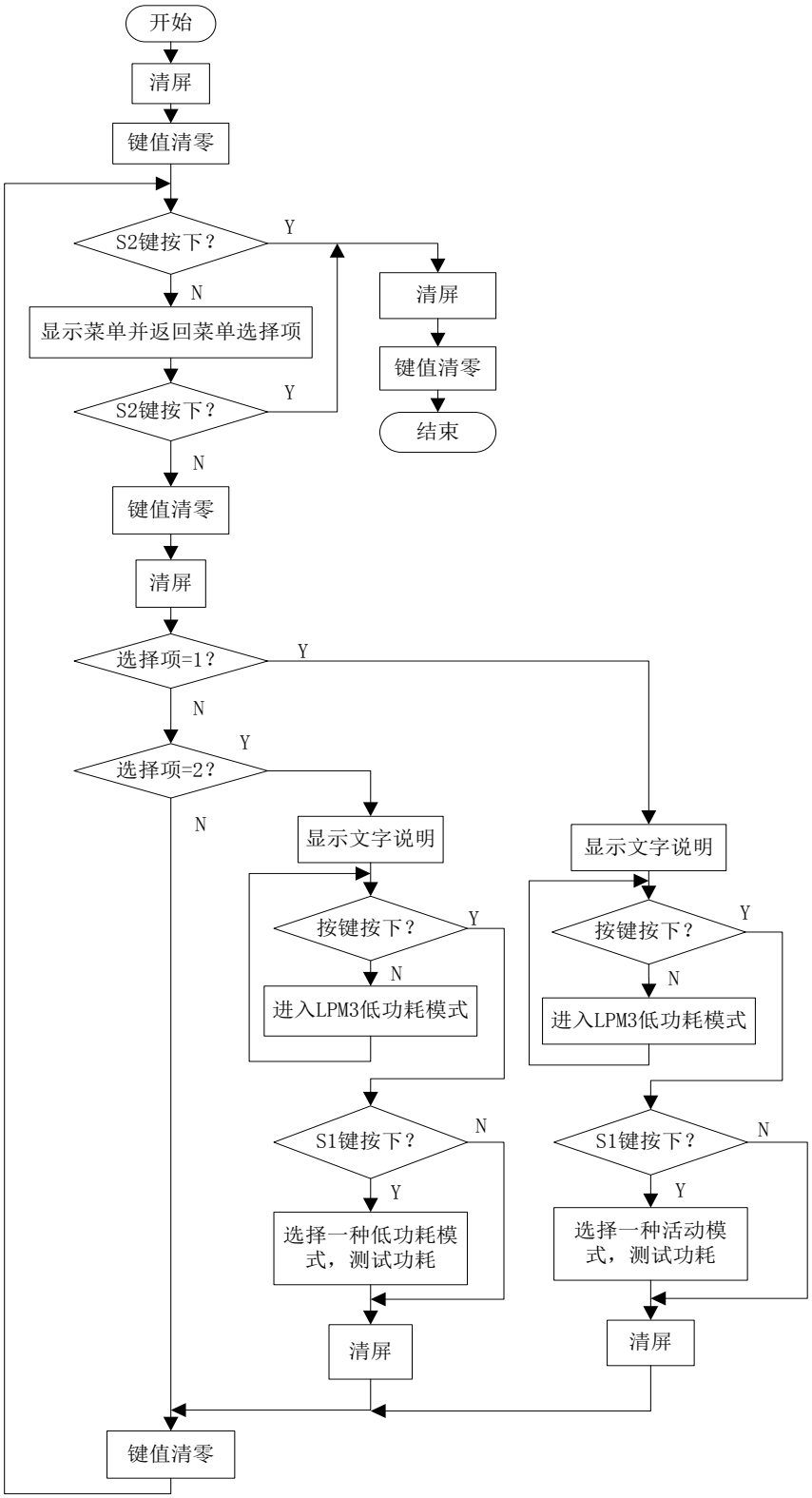


图 8.2 功耗测试实验程序流程图

8.5.3 实验步骤

（若 LAB6 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB6工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）将工程编译通过，并点击调试按钮，下载实验六程序，进入调试界面。

（5）如图8.3所示，由于在利用内置仿真器供电的情况下，DVCC不仅为MSP430F5529供电，也为内置仿真器供电，因此，若在此情况下测量MSP430F5529功耗，将造成测量错误。所以，需利用开发板右上角5529USB进行供电。单击停止按钮，退出调试界面，将Mini-USB线更改连接到右上角5529USB端口，并将电源拨码开关拨至LDO档位。

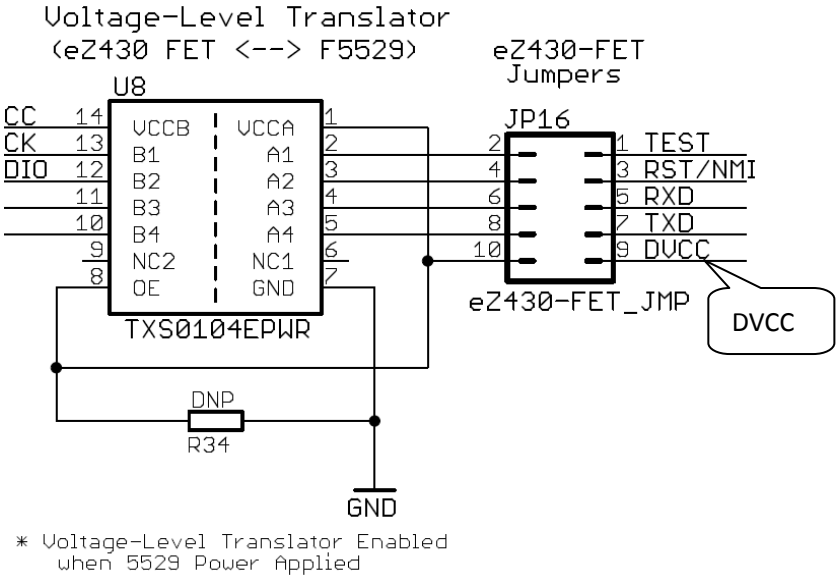


图8.3 内置仿真器与F5529之间电平转换电路

（6）程序开始运行，在 lab6:Power Tests 显示下，按下 S1 键进入本实验：在 LCD 上显示两个菜单，允许实验者测试活动模式和各低功耗模式下的 MSP430 单片机功耗。若进入活动模式(Active Mode)测量菜单，实验值可以看到两列菜单，左列控制单片机的核心电压，右列控制 MCLK，右边一列只显示在当前核心电压下有效的 MCLK，利用左面两个电容触摸按键选择左列或右列，利用齿轮电位计在所选列中选择行数，当选择完成后，按下 S1 开始测量，单片机功耗可以通过数字万用表进行测量，具体连接如图 8.4 所示。注意：需拿掉 JP6 短路块，将万用表打到 μA 档位并将万用表横跨在 430PWR 通孔中。当以上步骤完成，实验者即可获得在所选工作模式下的 MSP430 单片机功耗。当测试完成，按下 S1 或 S2 按键返回测试菜单。

若进入低功耗模式(Low Power Mode)测量菜单，实验者可以看到五种低功耗模式，通过齿轮电位计选择一种低功耗模式，按下 S1 开始测量，测量方法同活动模式下功耗测量。

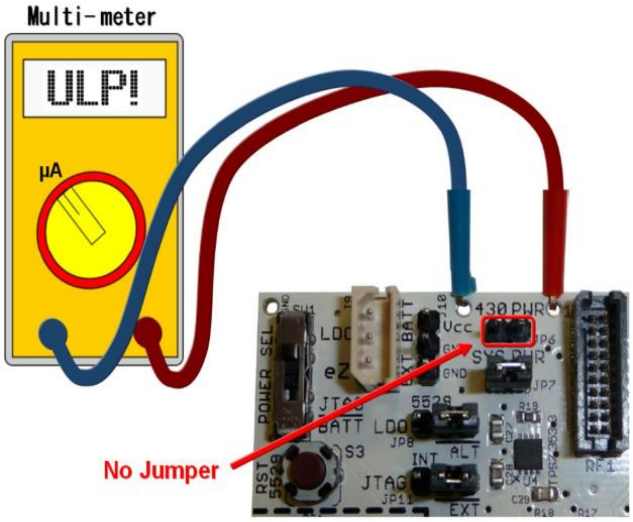


图 8.4 MSP430 单片机功耗测试连接图

（7）当测试完成，进入CCSv5.1编程环境，双击打开lab6.c文件，在第65行找到该功耗测试实验程序代码lab6 ()，并在其中设置断点，断点位置如图8.5阴影部分所示。

（8）将程序进行重新下载，进入调试界面，步骤参考（1）、（2）、（4）。运行程序，由于在程序中设置了断点，程序开始的界面，如图8.5所示。

```
65 void lab6(void)
66 {
67     uint8_t selection = 0;
68
69     buttonsPressed = 0;
70     Dogs102x6_clearScreen();
```

图8.5 功耗测试程序开始界面

（9）利用以下调试按键  ，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

8.5.4 实验结果

通过本实验，可以测得 MSP430 单片机在活动模式或低功耗模式下的功耗。现将实验所测 MSP430F5529 功耗列表如下（测试环境温度 30℃）：

表 8.4 活动模式下测试 MSP430F5529 功耗列表

参数	Vcore	MCLK							单位
		1MHZ	4MHZ	8MHZ	12MHZ	16MHZ	20MHZ	25MHZ	
I_{AM}	1.40V(0)	0.355	1.203	2.358					mA
	1.60V(1)	0.395	1.358	2.669	3.959	5.217			
	1.80V(2)	0.422	1.463	2.882	4.281	5.649	6.844		
	1.90V(3)	0.444	1.551	3.059	4.545	6.001	7.273	9.076	

表 8.5 低功耗模式下测试 MSP430F5529 功耗列表

参数	LPM0-1MHZ	LPM3-REF0	LPM3-LFXT1	LPM3-VLO	LPM4
I_{LPM}	96.2μA	5.0μA	2.6μA	1.5μA	1.3μA

提示：实验者可以通过 JP7 短路块，测试当前系统功耗，测试方法同上。

第九章 综合实验

9.1 实验目的

（1）综合学习按键、齿轮电位计、液晶、USB 通信及加速度计的应用；
（2）学习利用定时器周期性定时的方法；
（3）学习飞船避障游戏实验操作及编程思想；
（4）学习 USB 鼠标实验操作及编程思想。

9.2 实验所需硬件电路模块介绍

本实验中需用到以下电路模块：点阵 LCD 液晶显示模块、按键输入模块、齿轮电位计采样模块、Mini-USB 接口模块、加速度计模块、LED 指示模块。

9.3 实验内容

本章实验包括以下两个小实验：（1）飞船避障游戏实验；（2）USB 鼠标实验。实验（1）为点阵 LCD 液晶、按键输入、齿轮电位计的综合应用实验；实验（2）为按键输入、LED 控制、USB 通信和加速度计的综合应用实验。

实验七主函数 lab7()的整体程序流程图如图 9.1 所示：

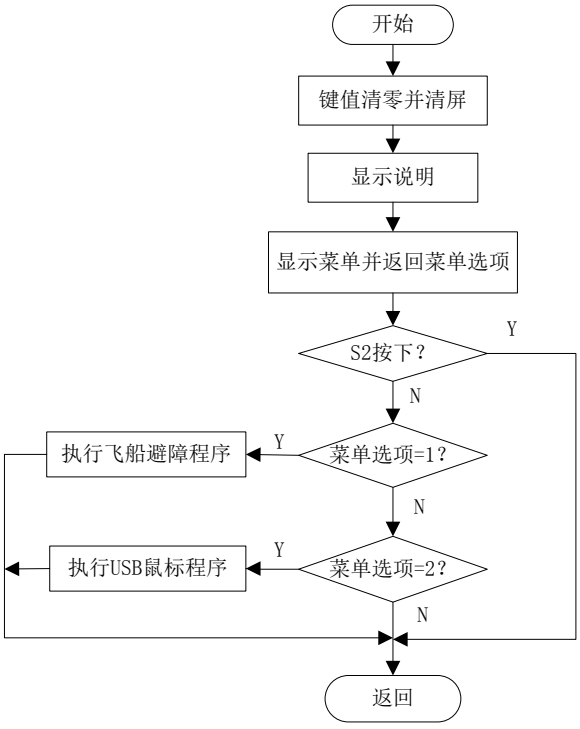


图 9.1 综合实验整体程序流程图

9.4 实验原理

（1）利用定时器 TA2 周期性定时

在 USB 鼠标实验中，利用 TA2 周期性定时发送鼠标位置报告，其实现程序代码如下：

```
void Mouse(void)
{
    ……
    TA2CCTL0 = CCIE;                    //CCR0中断使能
    TA2CCR0 = 547;              // 定时每16.7ms发送一次USB鼠标位置报告
    TA2CTL = TASSEL_1 + TACLR; // ACLK, 清除定时器计数
        ……
    TA2CTL |= MC_1;                 //开启定时器
    __bis_SR_register(LPM0_bits + GIE);//进低功耗模式0
   ……
}
```

在该程序中首先对 TA2 寄存器进行设置，将 CCR0 中断使能，并确定定时时间。在处理事件之前，需开启定时器，并进入低功耗模式 0，当定时完成，会进入 TA2 中断服务程序，如下所示。当一次事件执行完成，会循环进入低功耗模式 0，等待处理事件。

```
#pragma vector=TIMER2_A0_VECTOR
__interrupt void TIMER2_A0_ISR(void)
{
    Cma3000_init();                          //初始化加速度计
    Cma3000_readAccel_offset();             //读取移除偏移量后的加速度值
    sendNewMousePosition = TRUE;            //设置发送标志位
    __bic_SR_register_on_exit(LPM0_bits);//退出低功耗模式0
}
```

9.5 飞船避障游戏实验

9.5.1 程序代码

该实验的程序代码包含在 lab7.c 文件内：

```
void LaunchpadDef(void)
{
  ……
}
```

9.5.2 程序流程

飞船避障游戏实验程序流程图如图 9.2 所示。

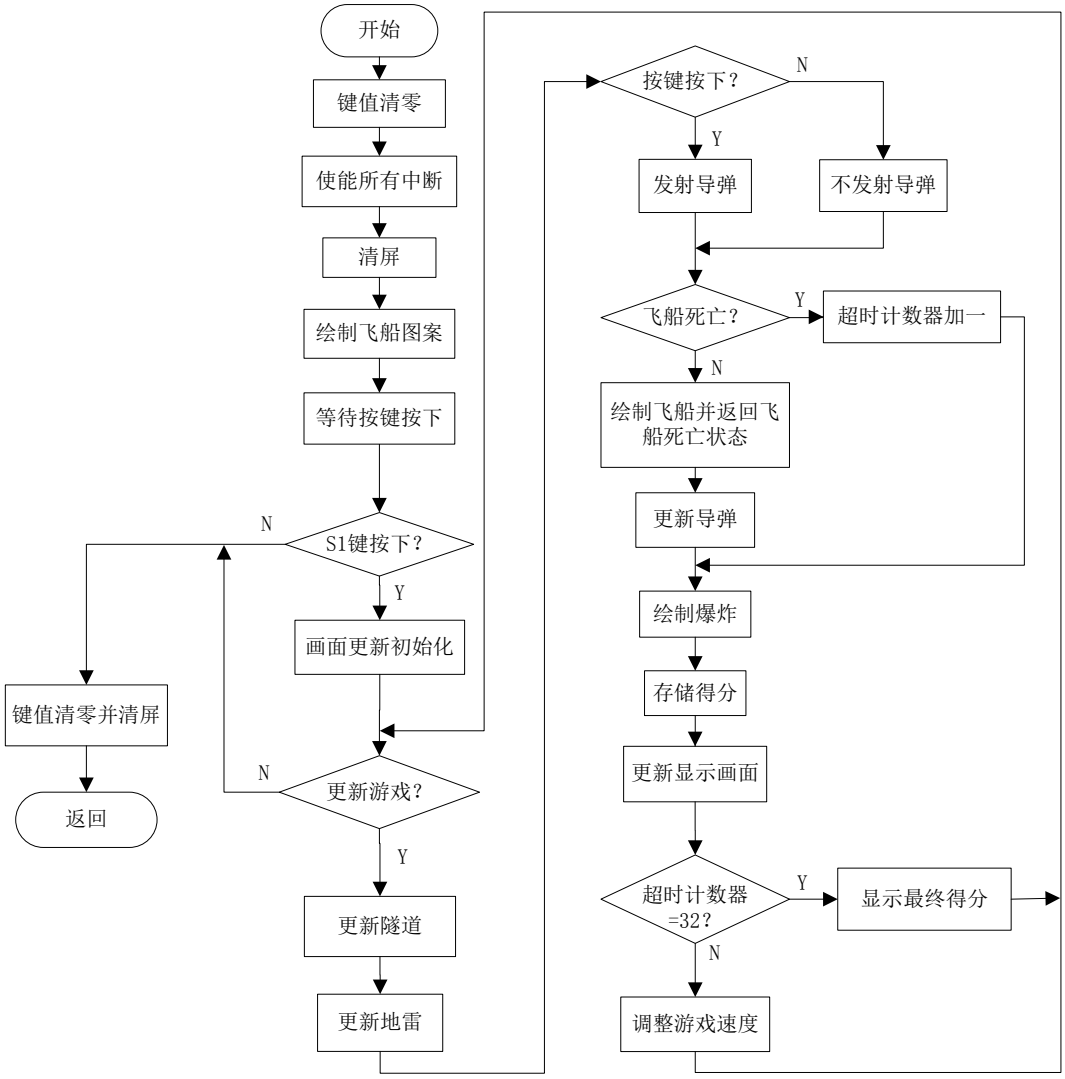


图 9.2 飞船避障游戏实验程序流程图

9.5.3 实验步骤

（若 LAB7 工程已导入，（1）（2）（3）步可省略）

（1）将电源选择拨码开关打至eZ档。

（2）利用Mini-USB线连接开发板仿真端口（左下角）和PC机。

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB7工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）双击打开lab7.c文件，在第774行找到该飞船避障游戏实验程序代码LaunchpadDef ()，并在其中设置断点，断点位置如图9.3阴影部分所示。

（5）将工程编译通过，并点击调试按钮 进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：1.Defender，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图9.3所示。

```
774 void LaunchpadDef(void)
775 {
776     uint32_t idx, a;
777     uint8_t dead, timeout;
778     uint8_t fire;
779     uint8_t restart = 0;
780     char scoreString[6];
781
782     buttonsPressed = 0;
783     __enable_interrupt();
784     Dogs102x6_clearScreen();
```

图9.3 飞船避障游戏实验程序开始界面

（7）点击运行按钮，首先会在 LCD 上观察到一个飞船的图案，程序等待按键按下，若 S2 键按下，则退出程序；若 S1 键按下，则开始游戏。在游戏的过程中，利用按键和齿轮电位计使飞船躲避障碍，提高分数：按键能使飞船发射子弹，齿轮电位计能使飞船上下移动。当飞船撞到障碍或者隧道，游戏结束，液晶 LCD 会显示游戏得分，在此时若按下 S2，退出程序，若按下 S1，游戏会重新开始。注意：①随着得分的提高，游戏运行速度越快且隧道越窄；②在游戏的过程中，按下 S2 键不会退出程序，只会发射子弹。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

9.5.4 实验结果

通过本实验，能够利用液晶 LCD、齿轮电位计及按键输入，实现飞船避障游戏功能。

9.6 USB 鼠标实验

9.6.1 程序代码

该实验的程序代码包含在 UserExperienceDemo→ mouse.c 文件内：

```
void Mouse(void)
{
    ..........
}
```

9.6.2 程序流程

USB 鼠标实验程序流程图如图 9.4 所示，Timer_A 中断服务程序流程图如图 9.5 所示。

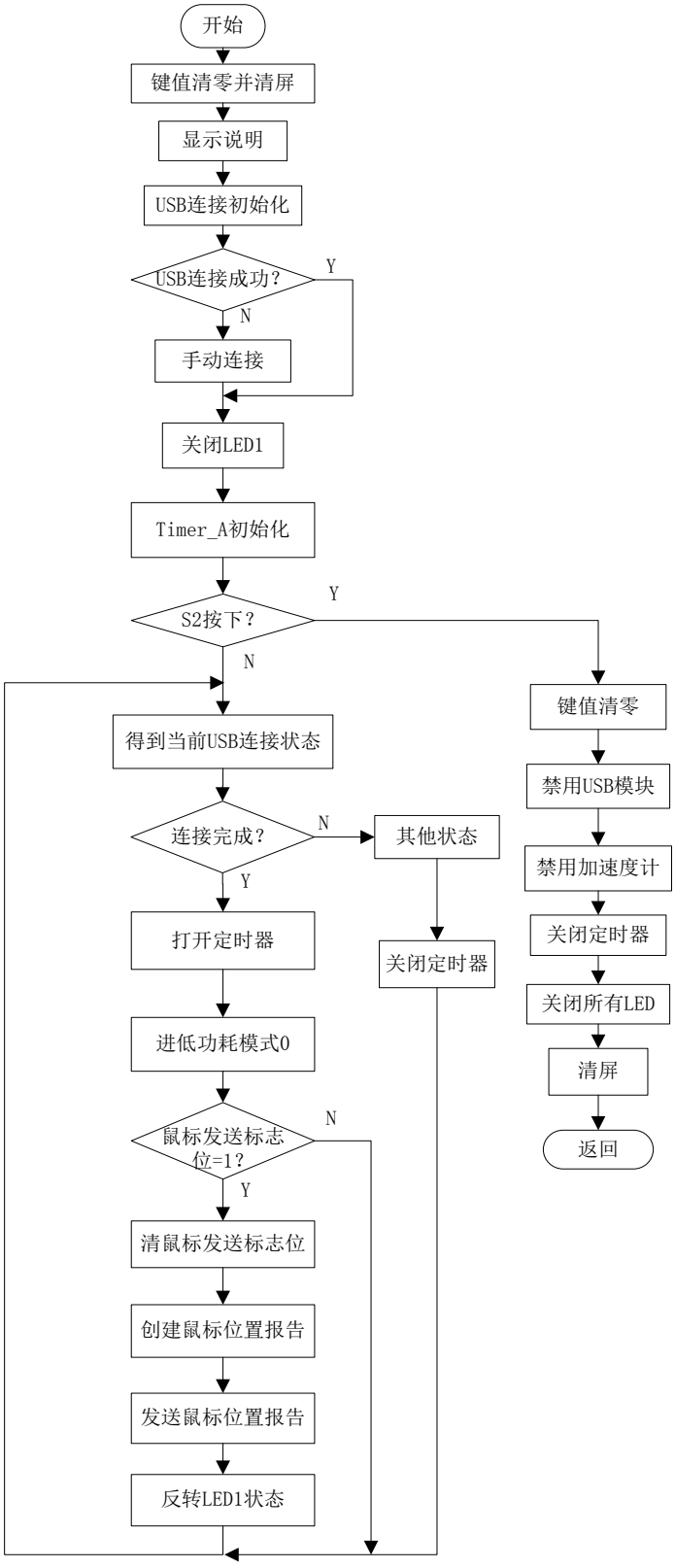


图 9.4 USB 鼠标实验程序流程图

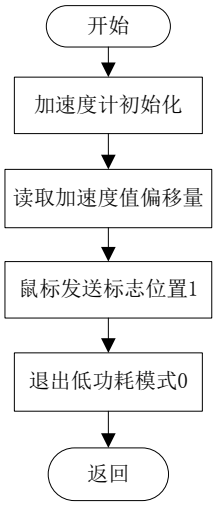


图 9.5 Timer_A 中断服务程序流程图

9.6.3 实验步骤

（若 LAB7 工程已导入，（1）（2）（3）步可省略，注意 USB 线连接方法）

（1）将电源选择拨码开关打至eZ档。

（2）利用两根Mini-USB线连接开发板和PC机，连接方法如图9.6所示。

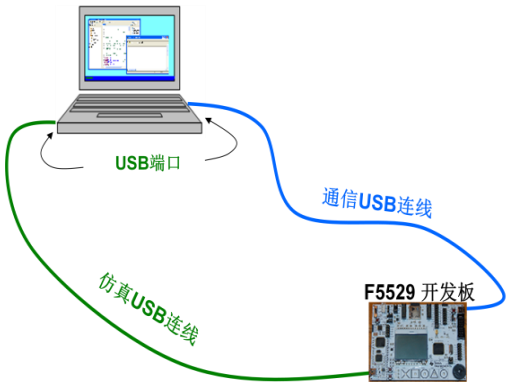


图9.6 USB鼠标实验USB连线

（3）打开CCSv5.1软件，确认工作区间“F\MSP-EXP430F5529\Workspace”，并导入LAB7工程，导入步骤请参考2.2节：利用CCSv5.1导入已有工程。

（4）展开UserExperienceDemo文件夹，双击打开mouse.c文件，在第100行找到该USB鼠标实验程序代码Mouse ()，并在其中设置断点，断点位置如图9.7阴影部分所示。

（5）将工程编译通过，并点击调试按钮进入调试界面。

（6）运行程序，在主菜单下，通过齿轮电位计选择：2. USB Mouse，然后按下S1键，由于在该程序中设置了断点，程序开始的界面，如图9.7所示。

```
100 void Mouse(void)
101 {
102     uint8_t i;
103
104     buttonsPressed = 0;
105
106     Dogs102x6_clearScreen();
107     for (i = 0; i < 4; i++)
```

图9.7 USB鼠标实验程序开始界面

（7）点击运行按钮，会观察到液晶LCD上显示系统说明，红色LED1闪烁，此时实验者可以通过倾斜开发板，来移动鼠标在PC桌面上的位置，按下S1键可以实现鼠标单击的功能。在实验的过程中，按下S2键退出本实验。

（8）单击重新开始按钮，重复第（6）步。

（9）利用以下调试按键，配合断点，进行代码的调试，理解各段代码含义，并观察各段实验现象。

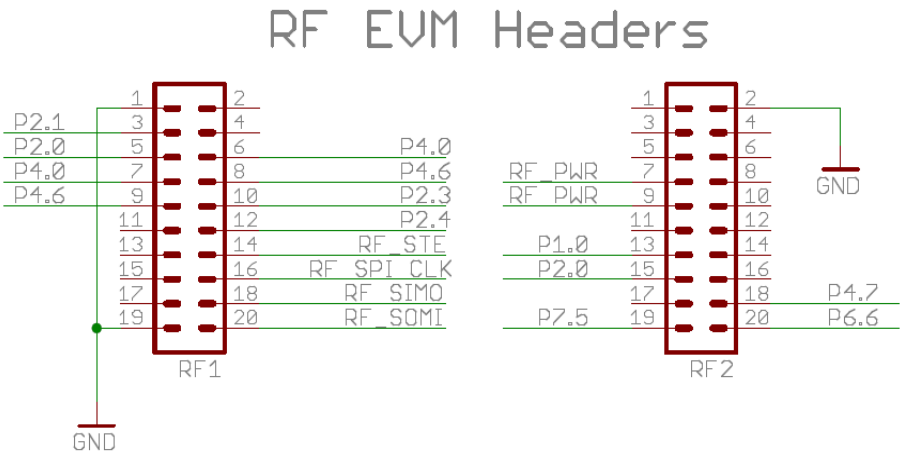
9.6.4 实验结果

通过本实验，利用加速度计、按键输入、LED 指示和 USB 通信，实现倾斜开发板移动 USB 鼠标的功能。

附录一 RF 无线接口模块电路介绍

MSP-EXP430F5529 开发板具有与射频 EVM 模块系列连接的接口，如附图 1 所示，无线收发模块在 SPI 模式下利用 UCB0 与 MSP430F5529 单片机连接，其中 RF_STE（P2.6）为从设备使能引脚、RF_SPI_CLK（P3.2/USB0CLK）为 SPI 模式传输时钟引脚、RF_SIMO(P3.0/USB0SIMO）为从设备输入主设备输出引脚、RF_SOMI（P3.1/USB0SOMI）为从设备输出主设备输入引脚。以下无线电子板能够与 MSP-EXP430F5529 开发板兼容：

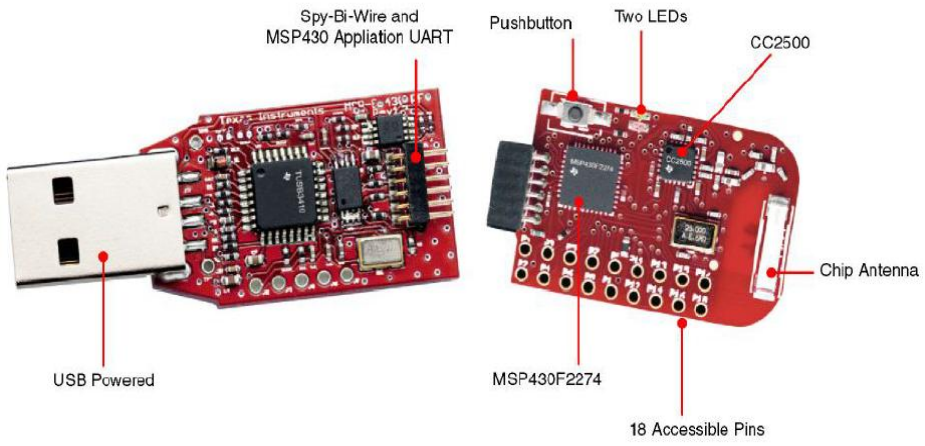
• CC1100EMK/CC1101EMK –低于 1GHZ 无线电
• CC2500EMK – 2.4GHZ 无线电
• CC2420EMK/CC2430EMK – 2.4GHz 802.15.4 无线电
• CC2520EMK/CC2530EMK – 2.4GHz 802.15.4 无线电
• CC2520 + CC2591 EM (在 R4 和 R8 通过 0Ω 电阻连接的情况下)



附图 1 RF 无线接口模块电路

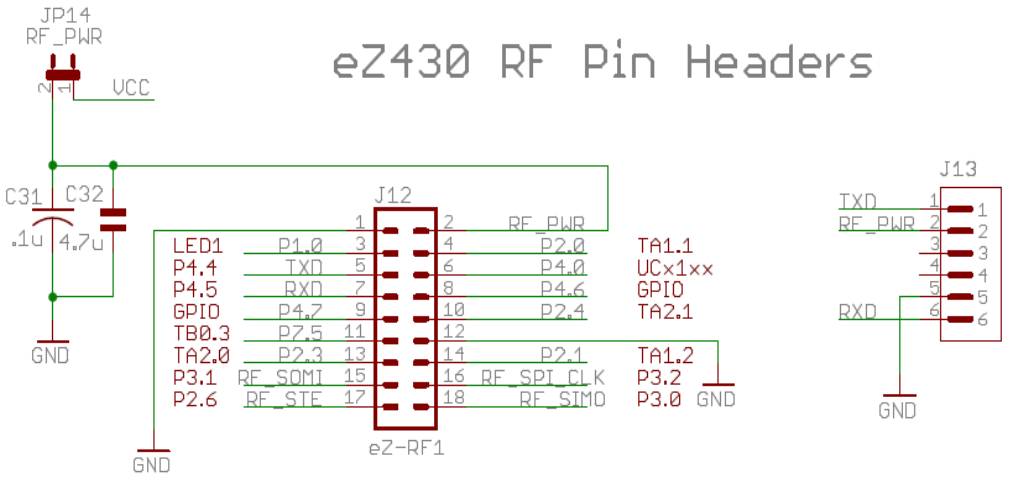
附录二 EZ430-RF 接口模块电路介绍

MSP-EXP430F5529 开发板能够与如附图 2 所示的 EZ430-RF2500T 模块相连，其连接方式有两种：通过 18 针连接器（J12-eZRF）和 6 针连接器（J13-EZRF Target）。



附图 2　EZ430-RF2500T 无线通讯模块

附图 3 为 EZ430 RF 的接口电路：附图 3（A）为 J12-EZRF 电路；附图 3（B）为 J13-EZRF Target 电路。建议：无论在哪种连接下，天线应尽量远离 MSP-EXP430F5529 开发板。



(A) J12-EZRF　　(B) J13-EZRF Target

附图 3 EZ430-RF 接口电路