Computer Science and

西南科技大学(SouthWest University of Science and Technology)

Experimental Instructor of Embedded System

嵌入式系统实验指导书

信息教研室　编写

二00七年

前言

FOREWORD

FOREWORD

FOREWORD

目　录

1 ADS 集成开发环境

1.1 ADS 集成开发环境简介

1.1.1 ARM 编译器的选择 —— ADS

目前,针对 ARM 处理器核的 C 语言编绎器有很多,如 SDT、ADS、IAR、TASKING 和 GCC 等。据了解，目前在国内最流行的是 SDT、ADS 和 GCC。SDT 和 ADS 均为 ARM 公司自己开发，ADS 为 SDT 的升级版，以后 ARM 公司不再支持 SDT，所以不会选择 SDT。GCC 虽然支持广泛，很多开发套件使用它作为编译器，与 ADS 比较其编译效率较低，这对充分发挥芯片性能很不得，所以最终使用 ADS 编译程序和调试。

本实验设备采用 ADS 编译器，其全称为 ARM Developer Suite。ADS 由命令行开发工具、ARM 时实库、GUI 开发环境(Code Warrior 和 AXD)、实用程序和支持软件组成。有了这些部件,用户就可以为 ARM 系列的 RISC 处理器编写和调试自己的开发应用程序了。ADS 支持汇编语言和标准 C 语言和标准 C++语言。

1.1.2 ADS 库路径

ADS 库路径是在 ADS 软件安装路径的 lib 目录下的两个子目录。假设，ADS 软件安装在 e:\arm\adsv1_2 目录,则在 e:\arm\adsv1_2\lib 目录下的两个子目录 armlib 和 cpplib 是 ARM 的库所在路径。

armlib 这个子目录包含了 ARM C 库，浮点代数运算库，数学库存等各类库函数。与这些库相就的头文件在 e:\arm\adsv1_2\include 目录中。

Cpplib 这个子目录包含了 Rogue Wave C++库和 C++支持函数库。Rogue Wave C++库和 C++支持函数库合在一起被称为 ARM C++库。与这些库相应的头文件安装在 e:\arm\adsv1_2\include 目录下。

读者需特别注意的几点：

(1) ARM C 库函数是以二进制格式提供的；

(2) ARM 库函数禁止修改。如果读者想对库函数创建新的实现的话，可以把这个新的函数编译成目标文件，然后在链接的时候把它包含进来。这样在链接的时候，使用的是新的函数实现而不是原来的库函数。

(3) 通常情况下，为了创建依赖于目标的应用程序，在 ANSI C 库中只有很少的几个函数需要实现重建。

(4) Rogue Wave Standard C++函数库的源代码不是免费发布的，可以从 Rogue Wave Software Inc，或 ARM 公司通过支付许可证费用来获得源文件。

1.2 ADS 集成开发环境的使用

1.2.1 进入 ADS 集成开发环境

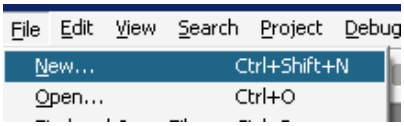
点击桌面 ADS 图标，如下图所示，进入 ADS 集成开发环境。

1.2.2 建立一个工程

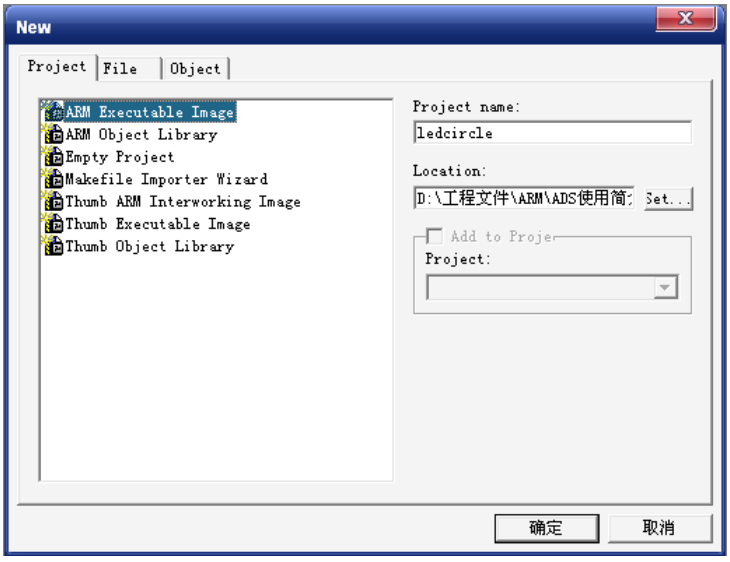
(1) 在 CodeWarrior 中新建一个工程有两种方法：

方法一：在工具栏中单击“New”按钮，如下图：

方法二：在“File”菜单中选择“New…”，如下图：



(2)执行上步骤后，则会弹出“新建工程对话框”，如下图：



在这个对话框中为用户提供了 7 种可选择的工程类型。

ARM Executabl Image：用于由 ARM 指令的代码生成一个 ELF 格式的可执行映像文件；

ARM Object Library：用于由 ARM 指令的代码生成一个 armar 格式的目标文件库；

Empty Prlject：用于创建一个不包含任何库或源文件的工程；

Makefile Importer Wizard：用于将 Visual C 的 nmake 或 GNU make 文件转入到 CodeWarrior IDE 工程文件；

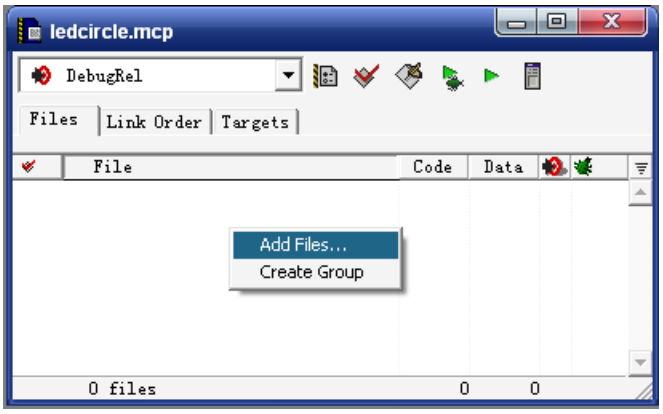
Thumb ARM Executable Image：用于由 ARM 指令和 Thumb 指令的混和代码生成一个可执行的 ELF 格式的映像文件；

Thumb Executable image：用于由 Thumb 指令创建一个可执行的 ELF 格式的映像文件；

Thumb Object Library：用于由 Thumb 指令的代码生成一个 armar 格式的目标文件库。

在这里选择 **ARM Executable Image**,在“Project name:”中输入工程文件名,本例为“ledcircle”,点击“Location:”文本框的“Set…”按钮，浏览选择想要将该工程保存的路径，将宽大些设置好后，点击“确定”，即可建立一个新的名为 ledcircle 的工程。

此时会出现 ledcircle.mcp 的窗口，如下图如示，其中有三个标签页，分别为 files,linkorder,target。



1.2.3 新建原文件

(1)在“File”菜单中选择“New”，在打开的对话框中，选择标签页 File，在 File name 中输入要创建的文件名，若是汇编语言则文件名格式为：***.s，若是 C 语言则文件名格式为：***.c，然后再点击“确定”关闭窗口。

在这里还有一个细节，希望注意。在建立好一个工程时，默认的 target 是 DebugRel ,还有另外两个可用的 target,分别为 Realse 和 Debug，这三个 target 的含义分别为：

DebugRel：使用该目标，在生成目标的时候，会为每一个源文件生成调试信息；

Debug：使用该目标为每一个源文件生成最完全的调试信息；

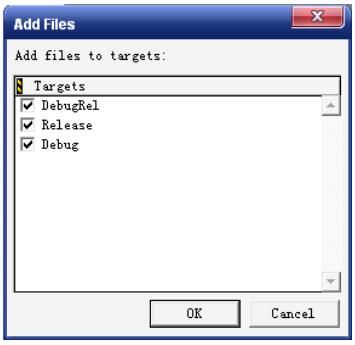
Release：使用该目标不会生成任何调试信息。

在本例中，使用默认的 **DebugRel** 目标。

1.2.4 向工程添加文件

(1)在 ledcircle.mcp 窗口的 file 标签页内右击鼠标右键，选中“Add Files…”可以把要用到的源程序添加到工程中。

(2)选中了要添加的文件后，会出现如下所示的一个对话框，询问用户把文件添加到何类目标中，在这里，我们选择 DebugRel 目标，再按 OK 即可把刚才创建的两个文件添加到工程中来。

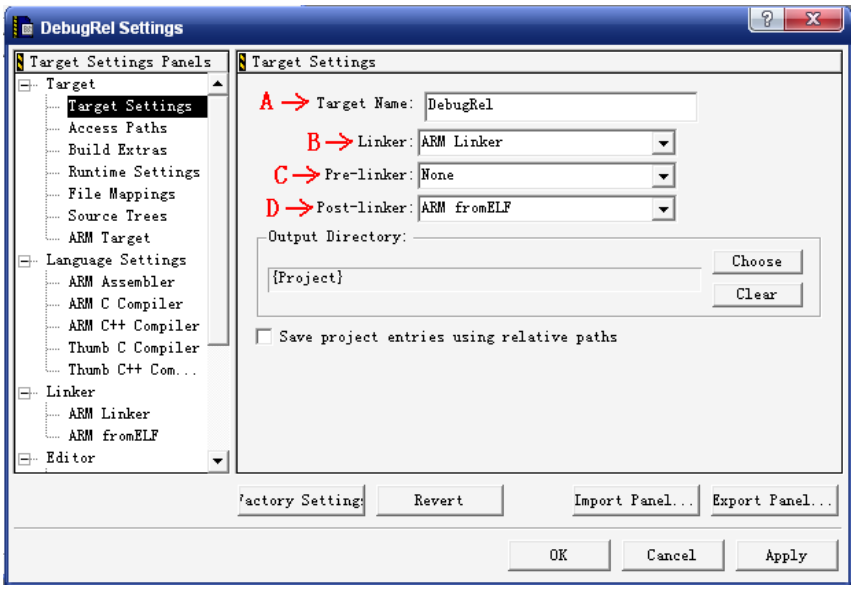


(3)对于本实验，在建立好每一个工程后都需将 stardcode 文件夹里的所有文件添加至工程中,且新建的每个文件里都需包含“config.h”头文件才行。原因不多说明，有兴趣可自行阅读文件代码。

1.2.5 编译和链接工程

在进行编译和链接前，首先讲述一下如何进行生成目标的配置。

点击 Edit 菜单，选择“DebugRel Settings…”，则出现如下图如示的对话框：



这个对话框中的设置很多，在这里只介绍一些最为常用的设置选项，读者若对其他未涉及到的选项感兴趣，可以查看相应的帮助文件。

(1)target 设置选项

A: TargetName 文本框显示了当前的目标设置。

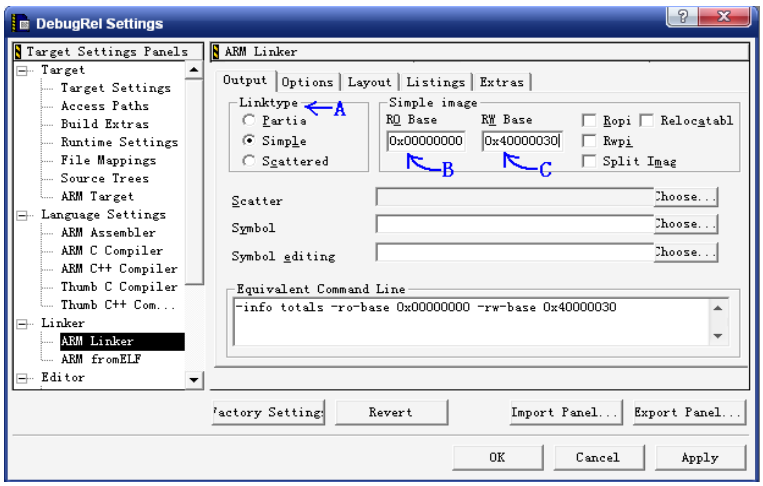
B: Linker 选项供用户选择要使用的链接器。在这里默认选择的是 ARM Linker，使用该链接器，将使用 armlink 链接编译器和汇编器生成的工程中的文件相应的目标文件。

C: Pre-Linker：目前 CodeWarrior IDE 不支持该选项。

D: Post-Linker：选择在链接完成后，还要对输出文件进行的操作。因为在本例中，希望生成一个可以烧写到 Flash 中去的二进制代码，所在在这里选择 **ARM fromELF**，表示在链接生成映像文件后，再调用 FromELF 命令将含有调试信息的 ELF 格式的映像文件转换成其化格式的文件。

(2)Linker 设置

鼠标选中 ARM Linker，出现如下图所示对话框。这里详细介绍该对话框的主要的标签页选项，因为这些选项对最终生成的文件有着直接的影响。

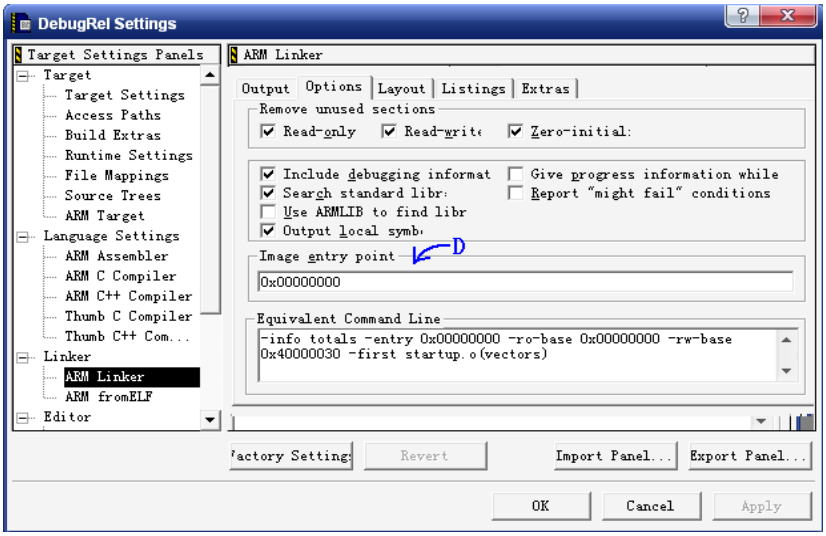


A: 在标签页 Output 中，Linktype 中提供了三种链接方式。Partial 方式表示链接器只进行部分链接，经过部分链接生成的目标文件，可以作为以后进一步链接时的输入文件。Simple 方式是默认的链接方式，也是最为频繁使用的链接方式，它链接生成简单的 ELF 格式的目标文件，使用的是链接器选项中指定的地址映射方式。Scattered 方式使得链接器要根据 scatter 格式文件中指定的地址映射，生成复杂的 ELF 格式的映像文件。这个选项一般情况下，使用不太多。因为我们所举的例子比较简单，选择 **Simple** 方式就可以了。在选择 Simple 方式后，就会出现 Simple image。

B: RO Base：这个文本框设置包含有 RO 段的加载域为同一个地址。默认是 0x8000。这里用户要根据自己硬件的实际 SDRAM 的地址空间来修改这个地址，保证在这里填写的地址，是程序运行时，SDRAM 地址空间所能覆盖的地址。针对本实验可以设置地址值为：**0x00000000** 。

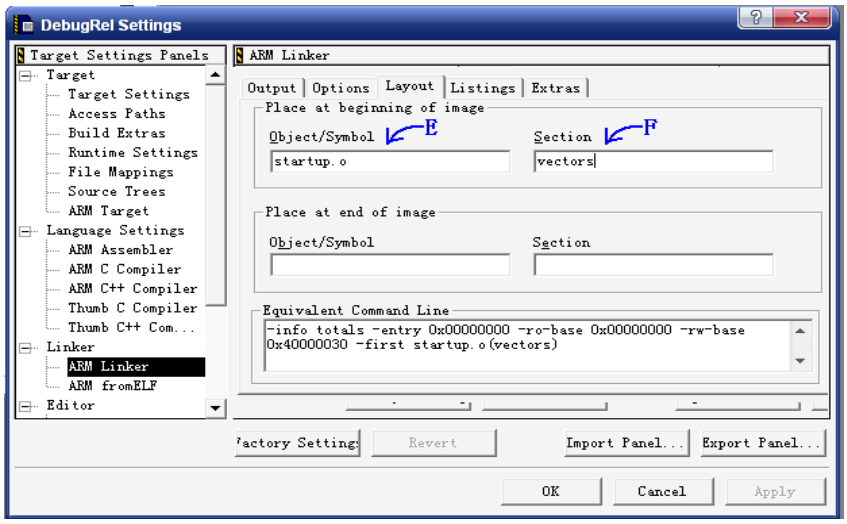
C: RW Base：这个文本框设置了包含 RW 和 ZI 输出段的运行域地址。如果选中 split 选项，链接器生成的映像文件将包含两个加载域和两个运行域，此时，在 RW Base 中所输入的地址为包含 RW 和 ZI 输出段的域设置了加载域和运行域地址。本实验可设置为：**0x40000030** 。

D: 在标签页 Options 中，将 Image entry point 文本框设置为：**0x00000000** 。如下图所示：

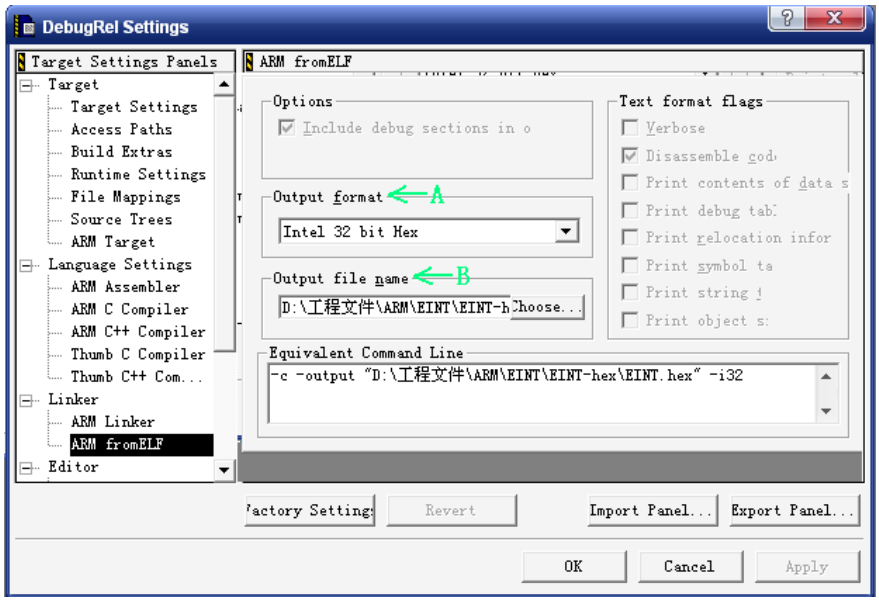


E: 在标签页 Layout 中，将 Object/Symbol 设为：**startup.o**

F: 将 Section 设为：**vectors**



鼠标选中 ARM fromELF，则会出现如下图所示的对话框：



A: 在 Output format 中选择 **Intel 32 bit Hex**

B: 在 Output file name 文本域输入期望生成的输出文件存放的路径，或通过点击 Choose…按钮从文件对话框中选择输出文件。如果在这个文本域不输入路径名，则生成的文件存放在工程所在的目录下。

(3)点击 CodeWarrior IDE 的菜单 Project 下的 make 菜单，或按 F7 键就可以对工程进行编译和链接了。

2 基础实验

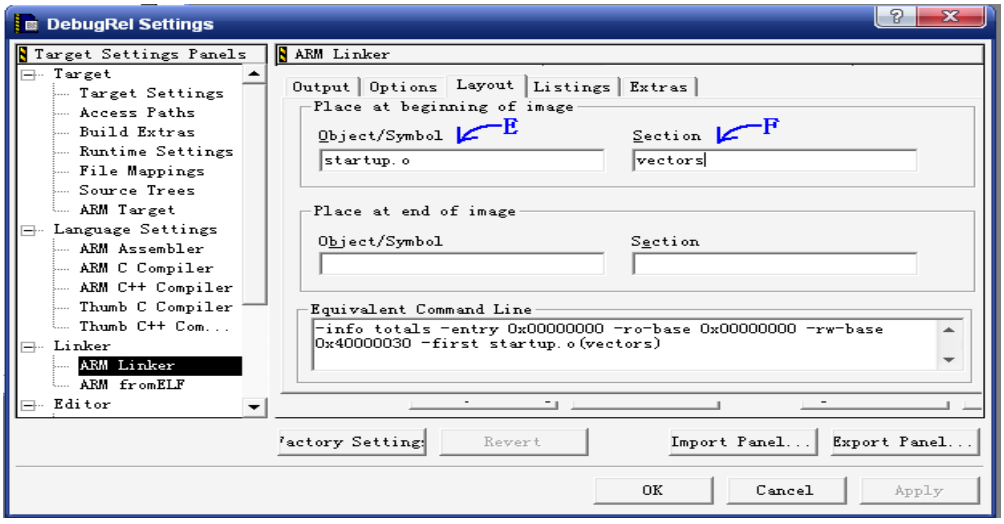
2.1 C 语言程序基础

2.1.1 启动代码

在 ARM 实验中，我们的程序都包含了一个名为“startcode” 文件夹。如果没有它我们的程序就无法运行。在 C51 中我们并没有包含一个类似的文件（夹），那是因为所用的编译器（keil uvision2）已经为我们做好了所有的初使化工作。我们把这种专门用来初使化系统硬件和软件的代码叫做启动代码（启动程序）。

嵌入式系统资源有限，程序通常都固化在 ROM 中运行。ROM 中程序执行前，需要对系统硬件和软件运行环境进行初始化，这些工作是用汇编语言（也有少量的 C 语言）编写的启动程序完成。启动代码是嵌入式程序的开头部分，应与应用程序一起固化在 ROM 中，应首先在系统上运行的启动代码应包含各模块中可能出现的所有段类，并合理安排他们的次序。启动代码要完成的工作主要分为：设置入口指针，设置中断向量，初始化堆栈和寄存器，初使化存储器系统，如果有必要改变处理器模式，状态等等。

启动代码可以由厂家或用户编写。在我们的 ARM 实验中，我们用已经编写好的启动代码（“startcode”）。来初使化系统。在进行编译器设置的时候，我们会注意到：



其中的 E 与 F 就是我们程序开始执行的地方，而它是启动代码。

2.1.2 编写一个简单的程序

想要在实验箱上运行一个程序，由先得创建一个工程。然后设置系统参数，包括启动代码的设置（详细请参考“ADS 集成开发环境的使用”）。最后我们编写应用程序实现相应的操作。简单的应用程序如下：

```
#include “config.h”

int main(void)
{
    PINSEL1 &= 0xfffcffff;          ①
    IO0DIR |= 0x01000000;           ②
```

```
    IO0CLR = 0x01000000;                ③
    While(1)                            ④
    {
        ;
    }
}
```

上面是一段很简单的程序，它的作用是使实验箱上的蜂鸣器发声（一直发声，不会停下来）。①是端口功能选择，这是选择其功能为 GPIO（普通输入输出端口）。②是端口方向选择，如果相应为 0 则是输入，相应位为 1 为输出。这里是设置蜂鸣器(P0.24)为输出。③设置 P0.24 输出为 0（低电平）。④是一个死循环，说明蜂鸣器会一直鸣叫。

（注：PINSEL1,IO0DIR,IO0CLR 都是端口寄存器，在“startcode”->“LPC2000.h”中有其定义。它们的地址分别为:0xE002C004,0xE0028008,0xE002800C。语句“PINSEL1 &= 0xfffcffff”的作用是把 PINSEL1 寄存器中的内容同 0xfffcffff 相与再赋给寄存器 PINSEL1。如果上面的程序不明白也没有什么，以后会学习。这里只是举例说明而已。）

2.1.3 文件（模块）管理

嵌入式编程一般采用的是面向过程的编程思想。我们将要实现的功能进行层层分解，然后在最底层分解一个一个模块，然后互相调用，实现系统功能。例如我们要实现串口 0 的通信功能：把实验箱上键盘输入的值传入到 PC 上，把从 PC 传来的值显示在实验箱上的数码管上。这时我们把要实现的功能分解，我们把分解后的一个具体功能叫做模块，故我们可以分解为：数码管模块（负责在数码管上显示输入数据），键盘模块（负责保存按下的键盘的数值），串口 0 模块（负责串口 0 的通信）。由于在实现的过程中我们要用到定时器我们也可以把定时器单独做为一个模块来管理。

一般来说一个模块包含一个头文件与实现文件。当然也可以包含几个头文件和实现文件，这时是把模块分组了。比如我们实现 LCD（液晶屏），它可以包含：接口层，驱动层，字库层。故我们可以写 6 个文件：LCD,h，LCD.c；LCDDriver.h，LCDDriver.c；LCDFont.h，LCDFont.c。特点要注意模块的化分与模块间的分层。我们应该尽量做到层次清晰，功能明确。这样实现起来才不易出错。

在程序实现过程中,我们为了便于模块的管理,我们可以为每一个模块建立一个文件夹。这样可以方便管理。我们就拿串口程序来举例：

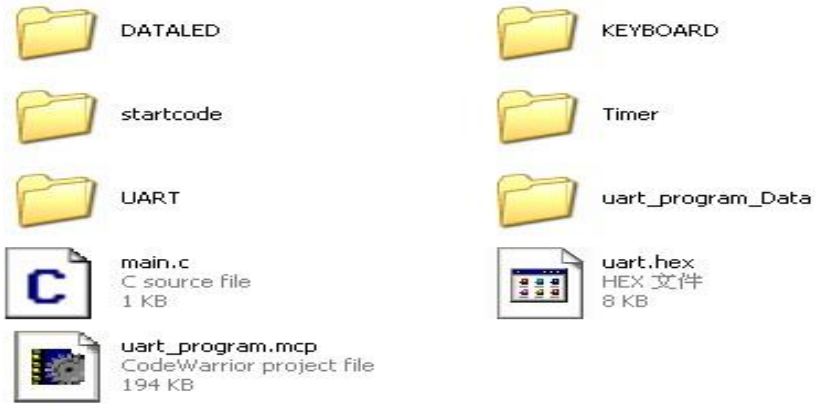


图 1　文件夹的管理

在程序中我们也可以把模块化分，这样使程序更加清晰。

可以在.mcp 文件中空白的地方点击右键，弹出“Add File…”与“Create Group”其中“Create Group”表示建立一个文件夹(在程序里面的如下图的“main”，“uart”…，并不是外部真实的文件夹）。“Add File…”表示在相应的文件夹里增加文件。如下图所示：

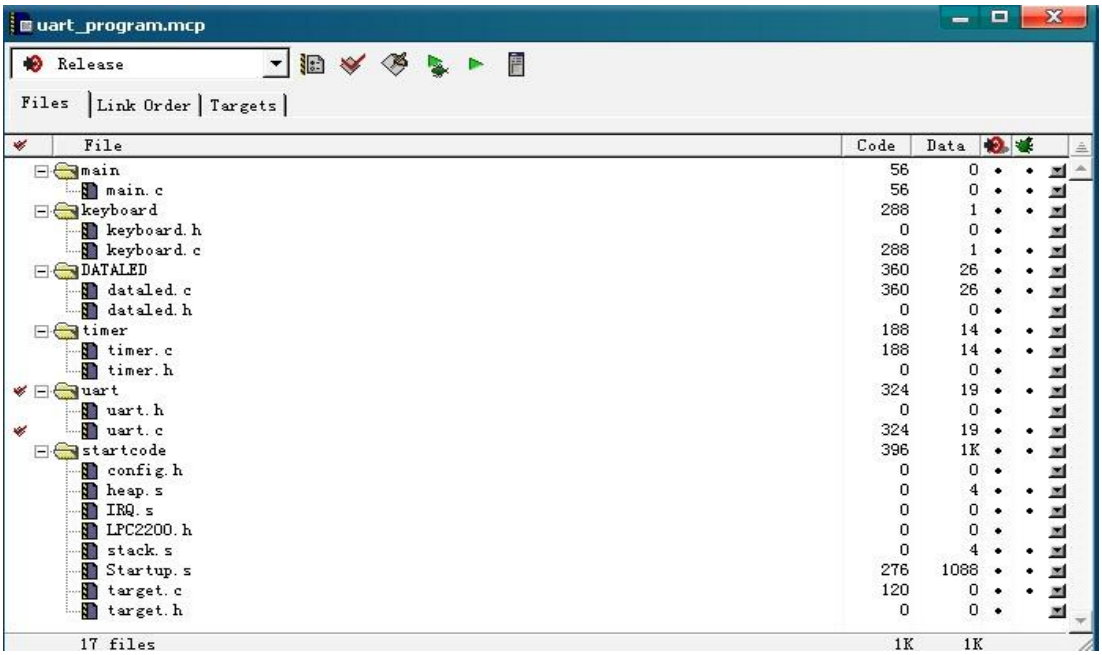


图 2　程序

2.2 流水灯程序

2.2.1 实验目的

1.1 理解并掌握 ARM 的端口操作

1.2 熟悉 HC595 工作原理

2.2.2 实验内容

2.1 掌握端口操作

2.2 让流水灯依次显示

2.2.3 实验预习要求

3.1 学习 ARM 端口操作方式

3.2 查询相关资料，掌握 HC595 原理

2.2.4 实验原理

4.1 端口操作原理

4.1.1 端口功能选择

由于 ARM 中端口基本上都有复用，所以对端口操作时首先要确定用该端口的什么功能，一般端口用作输入输出都是 GPIO 功能！确定为 GPIO 功能的实现方法一般为对对应端口 PINSEL 清 0，实现方法范例如下：PINSEL0 &= KEYBOARD_SMAT; 如果 KEYBOARD_SMAT 等于 0xffff00ff，那么 P0.4~P0.7 端口为 GPIO 功能！首先是对 PINSEL0 操作说明是对于 P0 端口中的 P0.0~P0.15 操作，每个端口对应两 bit，因为某些端口功能多于两种；所以例子中是对 P0.4~P0.7 操作。（端口功能介绍见 LPC2292 数据手册）

4.1.2 端口方向选择

对端口确认为 GPIO 功能后，则需要确定该端口是输入输出，ARM 中是通过 IOXDIR 寄存器实现！如：IO0DIR |= KEYBOARD_SCK;说明是对端口 P0 操作，在这里其操作的范围为整个端口 P0,每一个端口只有输出输入判断，估在 IOXDIR 一个端口只需要一个 bit 就可以做出判断。如果 KEYBOARD_SCK= 0X00000010，说明是对 P0.4 设置为输出。如 IO0DIR &= (KEYBOARD_KEY^0XFFFFFFFF);为相应的输入实现方式

4.1.3 清零与置位

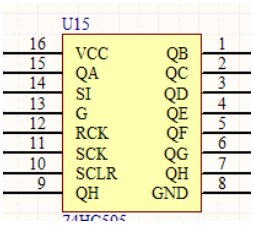
在端口为输出时,对于一个端口可以置位也可以清 0,在 ARM 中的实现方式如下！如：IO0CLR = KEYBOARD_SI;此语句是对 KEYBOARD_SI 为 1 的位全部清 0，如果 KEYBOARD_SI 等于 0x00000040，那么 P0.6 端口则清 0，如果 KEYBOARD_SI 等于 0xFFFFFFFF 则是对整个 P0 清 0；对应的 IOxSET 也是同样的方式，不同的它对相应端口置 1；

4.1.4 输入电平判断

在端口为输入时，判断以个端口是低电平还是高电平则是通过 IOXPIN 这类寄存器实现的。如：if((IO0PIN&KEYBOARD_KEY)!=0)，如果 KEYBOARD_KEY 等于

0x00000020 说明是对于 P0.5 做判断，如果(IO0PIN&KEYBOARD_KEY)等于 0，说明该端口为返回值为低电平！同理高电平也是通过此类判断！

4.2 HC595 工作原理



VCC:接电源；GND 接地

SCLR(10 脚)：低电平时将移位寄存器数据清零，通常接 VCC.

QH(9 脚)：级联输出端，接下以个 595 的 SI

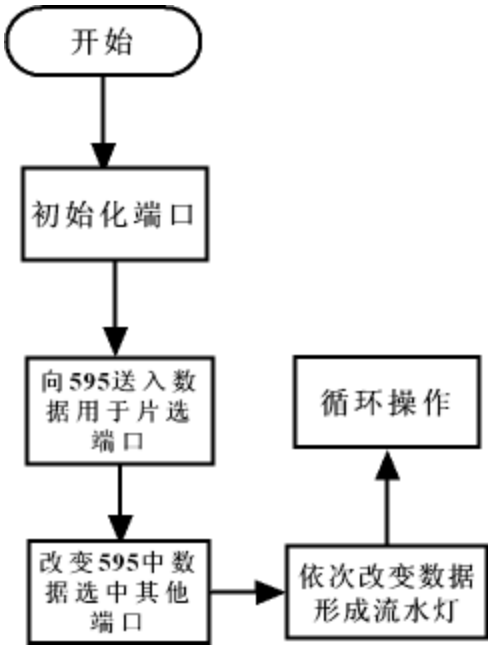
G(13 脚): 高电平时禁止输出（高阻态）。

QA~QH:为八位并行输出端

RCK,SCK，SI 如下：

实际中我们主要是对 595 控制 RCK,SCK，SI 来实现 595 的功能的。将一个八位数据转换到 QA~QH 上，以 0，1 的方式出现在 QA~QH 八个端口。通过 8 次 SCK 上升沿的移位信号用 SI 口接收每位数据，实现将一个字节的数据装入 595 中。SCK 的上升沿发出一个移位信号，每位数据都移动到下一个移位寄存器中；遇到同步发送脉冲 RCK（上升沿有效）时，将这个字节的数据输出，形成一个相当于 8 位或多位的并行数据；

2.2.5 实验流程图



2.2.6 参考源代码（见实验指导丛书源码版）

2.2.7 思考题

7.1 联想此实验与键盘实验的联系

2.3 码管扫描程序

2.3.1 实验目的

1.1 理解 ARM 数码管显示的一般原理，掌握使用数码显示的一般方法。

1.2 学会十六进制字型的显示方法

1.3 学会同时（实际上并不是同时，只是视觉）让多个数码管显示，显示任意字符

1.4 了解 74HC595 是如何工作的

2.3.2 实验内容

2.1 理解数码管显示的字符代码和数字之间的关系。

2.2 理解数码管是如何片选及显示

2.3 依次让多个数码管显示任意字符

2.4 学会使用 74HC595

2.3.3 实验预习要求

3.1 查找数码管扫描相关资料，了解循环扫描的基本原理，

3.2 复习 ARM 中对端口的操作步骤，复习数码管模块的是实现方法。

3.3 查找 74HC595 的芯片资料，了解该芯片的工作原理和在数码管扫描模块中的作用。

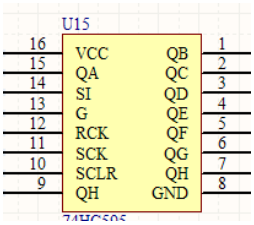
2.3.4 实验原理

4.1 端口寄存器介绍

PINSEL 是用于选择端口功能选择，IOPIN 用于读引脚当前状态，IOSET 控制寄存器引脚输出高电平，IOCLR 控制寄存器引脚输出低电平，IODIR 控制每个 IO 口的方向。例如：IODIR |= DLED_RCK;（DLED_RCK 为宏，是一个 32 位无符号整型）

4.2 595 原理

数码管扫描涉及到 4 个引脚，分别是 P0.17～P0.20。P0.17,P0.18,P0.20 分别作为 74HC595 的 SCK，SI，RCK。（结合 74HC595 手册理解）通过 SI 口接收每位数据；SCK 的高电平发出移位信号，每位数据都移动到下一个移位寄存器中；遇到同步发送脉冲 RCK 时，将多位数据一次发送到寄存器中，形成一个相当于 8 位或多位的并行数据；从而通过 595 给数码管给予片选和数据



图中通过 595 控制 RCK,SCK，SI，将一个八位数据转换到 QA~QH 上，以 0，1 的方式出现在 QA~QH 八个端口，从而实现输出高低电平

4.3 数码管原理

数码管内部为 8 个发光二极管，并排列为 8 字形，同时加一个位表示小数点，通过这 8 个发光二极管的合理组合，可以构成不同的数字字型和简单的字母字型，同时数码管还有一个位选信号。即 8 个数码管的公共端，用于电平选中。原理图中的 A~H 的 0，1 组合就可以组成不同的字符。

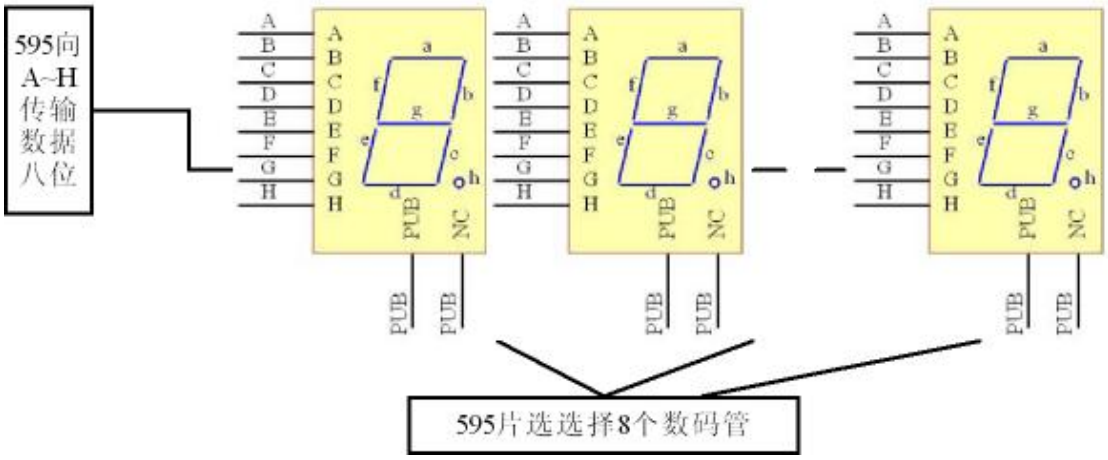


图 1-1 硬件示意图

2.3.5 实验流程图

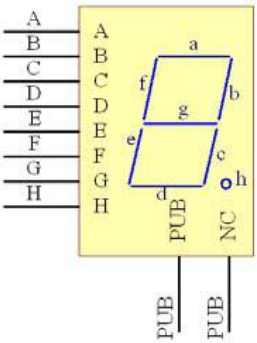


图 1-2 数码管原理图

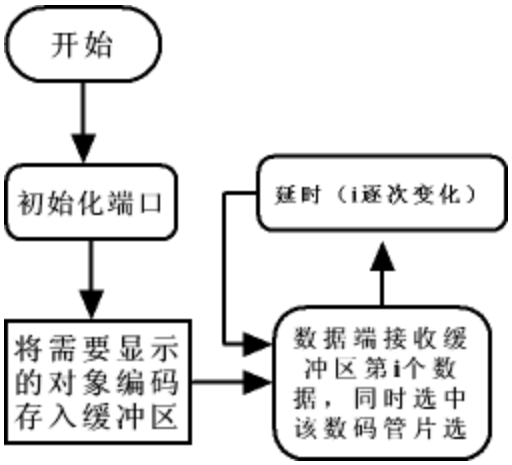


图 1-3 数码管总体流程图

2.3.6 参考源代码（见实验指导丛书源码版）

2.3.7 思考题

7.1 如何使你写的程序高效

7.2 如何使你写的程序可广泛移植到其他程序中

7.3 是否有其他的方法让数码管显示

2.4 键盘扫描程序

2.4.1 实验目的

1.1 了解键盘电路的布局，理解键盘扫描的基本原理

1.2 熟练掌握键盘轮转扫描的方法

1.3 熟悉 74HC595 芯片的原理，以及如何利用 74HC595 写程序实现串并转换。

2.4.2 实验内容

2.1 了解实验箱键盘模块的工作原理，引脚连接。

2.2 写程序实现轮转扫描算法，实现键盘的扫描。能够判断是否有键按下，结合数码管显示模块，实现按下某键显示相应内容。

2.4.3 实验预习要求

3.1 查找键盘相关资料，了解轮转扫描的基本原理，对比其与行扫描、全扫描的区别。

3.2 复习 ARM 中对端口的操作步骤，复习数码管模块的是实现方法。

3.3 查找 74HC595 的芯片资料，了解该芯片的工作原理和在键盘扫描模块中的作用。

2.4.4 实验原理

4.1 端口介绍

键盘扫描涉及到4个引脚，分别是P0.4～P0.7。P0.4,P0.6,P0.7 分别作为 74HC595 的 SCK，SI，RCK，（595 原理在前一实验已做介绍） p0.5 为返回值端口，也是按键是否按下的判断位。在硬件电路上，只要一个按键按下，并且对应的片选中该位为低电平（任意一位同时满足两条件），那么 rekey 端口将返回低电平，处理器可以根据该端口的电平来判断是否有键按下。具体是何键值则可以由程序设计判断，因为片选可以通过逐位清 0 （16 个位只能有一位为 0 ）来实现的（本实验才用轮转扫描算法）

4.2 轮转扫描是这样实现的

。通过 595 向 16 位键盘输入片选，其中只有一位为 0 ，为了在同一时刻只判断一位是否按下。通过对片选数据逐位右移，将对每一位判断是否按下，如果同时满足该位片选为 0 且有键按下，那么返回值为 0 ，说明有键按下，同时可在程序中用一记数变量记录判断是哪一位，同时就解决了是何键按下，因为一旦判断出返回值为 0 ，说明此刻记数变量的值即为按下键盘为的值（或者说相关，看你具体程序如何处理），此刻立即取出变量中的值，就做到了判断是何键位的目的。

4.3 硬件原理

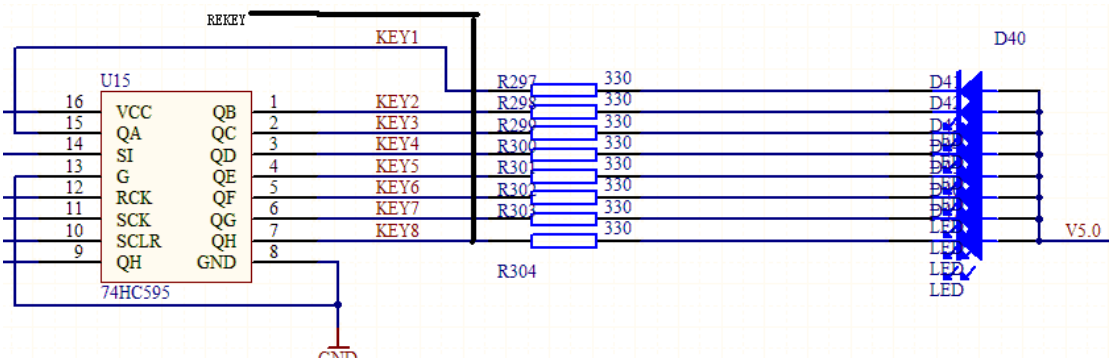


图 2-1 硬件示意图

图中通过 595 控制 RCK,SCK，SI，将一个八位数据转换到 QA~QH 上，以 0，1 的方式出现在 QA~QH 八个端口，从而实现输出高低电平

可以看出仅仅只有键盘（key）按下且 QA~QH 为低电平时，REKEY 端口才为低电平！从而可以用轮转算法实现

4.4 实验流程图

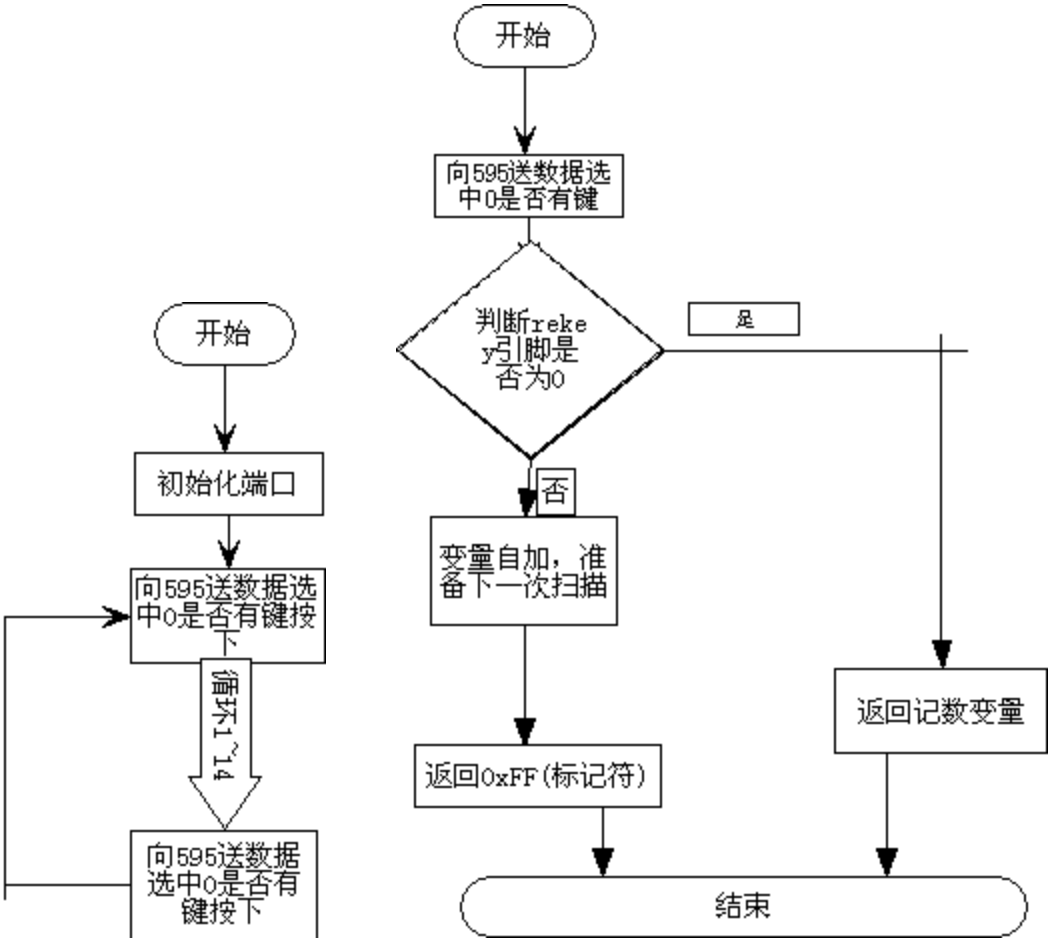


图 2-2 总体流程图

图 2-3 扫描函数流程图

2.4.5 参考源代码（见实验指导丛书源码版）

2.4.6 思考题

6.1 如果按键与需要的偶尔有差别，为什么？

6.2 当按下键后，是否需要继续扫描，两者如何实现？

6.3 同时按下两个键，什么结果，为什么？

2.5 阵屏汉字显示程序

2.5.1 实验目的

1.1 了解点阵屏显示基本原理

1.2 熟悉 74HC595 芯片的原理，以及如何利用 74HC595 写程序实现串并转换。

1.3 熟悉使用字模软件

1.4 进一步熟悉 ARM 的端口操作

2.5.2 实验内容

2.1 建立汉字字库

2.2 在点阵屏上显示汉字

2.5.3 实验预习要求

3.1 预习 GPIO（GERERAL PROGRAMABLE INPUT OUTPUT）通用可编程输入输出口的基本操作。

3.2 找相关 74HC595 芯片的资料，了解起基本原理和使用方法。

2.5.4 实验原理

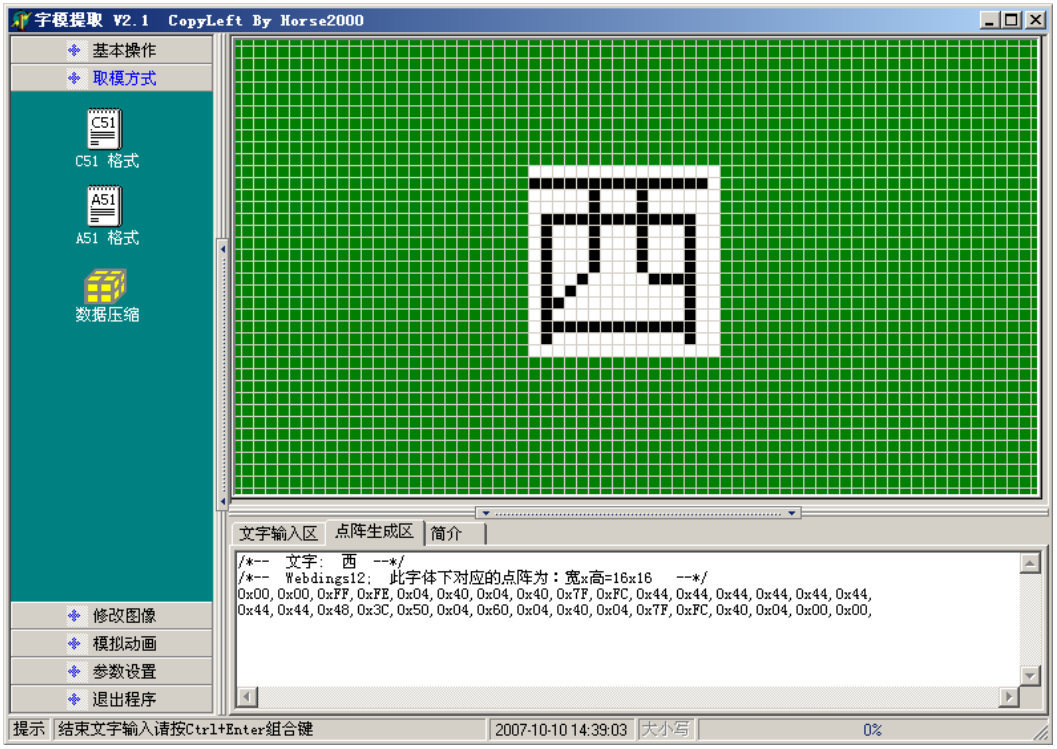
点阵屏涉及到 4 个引脚，分别是 P0.20～P0.17。P0.4,P0.17,P0.18 和 P0.19，P0.20 分别作为 74HC595 的 SCK，SI，RCK。（结合 74HC595 手册理解）通过 SI 口接收每位数据，P0.18 和 P0.19 分别对应 SI-X，SI-Y，用于控制行和列的显示；SCK 的高电平发出移位信号，每位数据都移动到下一个移位寄存器中；遇到同步发送脉冲 RCK 时，将多位数据一次发送到寄存器中，形成一个相当于 16 位或多位的并行数据。

点阵显示是这样实现的。利用串并转换将两个 16 位数据分别发送给 P0.18 和 P0.19 所对应的 SI-X 和 SI-Y。它们分别控制片选位和显示数据位。（至于行列哪个作片选位哪个做数据显示位可由自己定，一般便于观察和结合字模软件，将点阵屏相对于实验箱正向的行作为数据显示位，列作为片选位）通过控制片选和数据显示位可以实现固定行显示固定的亮灭信息。再通过扫描算法（与 LED 灯相似）实现点阵屏显示任意数据信息。

字模软件的使用：

通过字模软件，可以很容易的实现 16X16（或者其他大小）的点阵字模数据的提取，而不用手动的去计算。具体的实现方法可以结合软件自己尝试，比较容易。

字库的建立一个重要问题在于如何索引，即通过哪种方式可以方便的调用你的字库，一般建立字库采用二维数组。通过数字索引或者其他方式可以方便调用。如果是 ASC 码的字符，可以通过以 ASK 码为桥梁方便的建立索引。



2.5.5 实验流程图

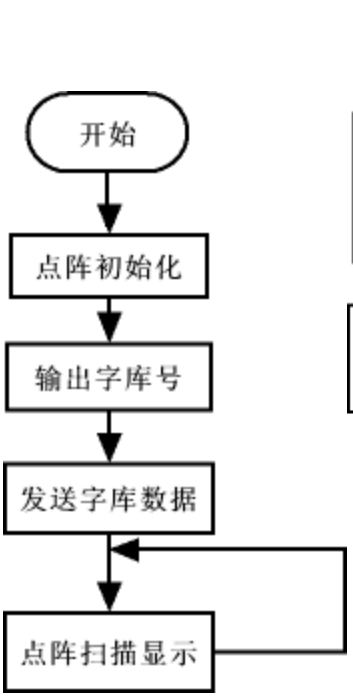


图 3-1 点阵总体流程图

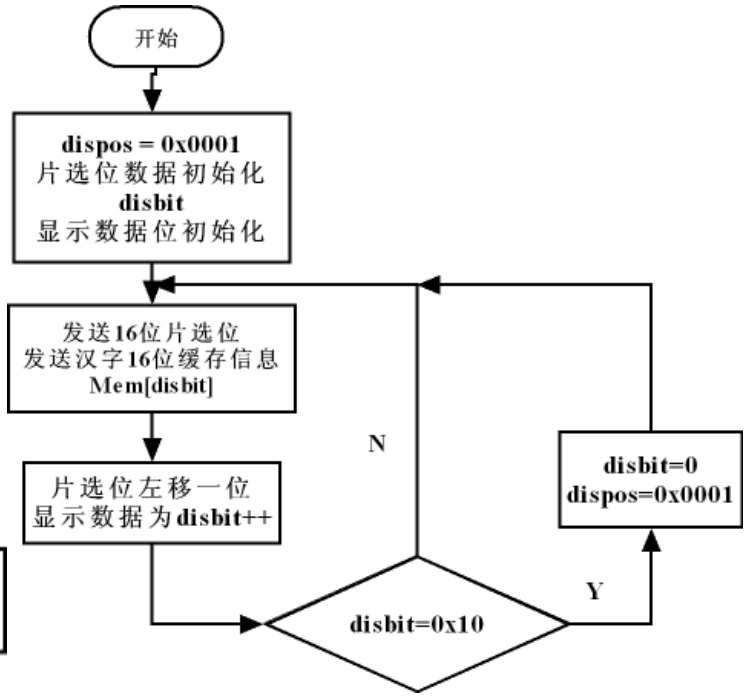


图 3-2 点阵扫描函数流程图

2.5.6 参考源代码

2.5.7 思考题

7.1 怎样使点阵屏隔一段时间显示另一个字？

7.2 与 LED 程序相比较，有什么异同？

2.6 外部中断实验

2.6.1 实验目的

1.1 了解外部中断的基本设置方法和原理。

1.2 通过外部中断实验学习 ARM 中中断的产生机制。

1.3 了解中断的应用。

2.6.2 实验内容

2.1 通过实验箱发声器和红外发送-接收对管实现通过外部中断令发生器产生声响。

2.6.3 实验预习要求

3.1 预习《ARM 嵌入式系统基础教程》中向量中断控制器和外部中断输入的章节，了解相关寄存器的内容和用法。

3.2 了解什么是外部中断，与其他中断有什么联系和区别。

2.6.4 实验原理

首先需要介绍向量中断控制器的相关知识：

4.1 向量中断控制器（VECTORED INTERRUPT CONTROLLER，简称 VIC）

向量中断寄存器：具有 32 个中断请求输入，可将其编程分为 3 类：FIQ，向量 IRQ，非向量 IRQ。可编程分配机制意味着不同外设的中断优先级可以动态分配及调整。

中断输入可以在 VIC 被设置成三类

1、FIQ 中断 ：具有最高优先级

2、向量 IRQ 中断 ：具有中等优先级 VIC 最多支持 16 个向量 IRQ 中断，这些中断被分为 16 个优先级，每个优先级有一个中断服务地址

3、非向量 IRQ 中断 ：具有最低优先级 所有的非向量 IRQ 中断共用一个中断服务地址

4.2 相关寄存器

与中断相关的常用和比较重要的寄存器主要有：

VICIntEnable 中断使能寄存器 32 位，每一位控制一个中断源，向某一位写入 1，允许该中断源产生中断

VICIntClr 中断使能清零寄存器，与 VICIntEnable 相反，写入 1，相应位中断源禁止中断

VICIntSelect 中断选择寄存器 向某位写 1，对应中断源产生 FIQ 中断，否则产生 IRQ 中断，其中断相应延时相应较长

VICVectCntl 中断向量控制寄存器 为中断源分配向量 IRQ 的优先级 ，VICVectCntl n，n 值越小优先级越高，0<=n<=15

VICVectCntl	位	7	6	5	4	3	2	1	0
功能		-	-	En	【	中断源序号	】		

VICVectAddr 向量地址寄存器 为该中断优先级设置服务程序入口地址

VICRawIntr 中断状态寄存器 32 寄存器，当某位为 1 时，对应位中断源产生中断请求

VICFIQStatus FIQ 状态寄存器 32 寄存器，当某位为 1 时，对应位中断源产生 FIQ 中断请求

VICIRQStatus IRQ 状态寄存器 32 寄存器，当某位为 1 时，对应位中断源产生 IRQ 中断请求

VICSoftInt 软件中断寄存器 32 寄存器，当某位为 1 时，对应位中断源产生相应中断请求

VICSoftIntClear 软件中断清零寄存器 当某位为 1 时，将清零 VICSoftInt 对应位

软件使能寄存器

VICSoftInt	位	31~1	0
	功能	-	该位为 1 时只有在特权模式下访问 VIC 寄存器

4.3 初始化基本步骤

以上寄存器并不是全部需要用到，当需要时再进行相应设置即可。这里给出中断初始化的基本步骤：

1、设置中断类型 VICIntSelect

2、设置中断优先级 VICVectCnt

3、将中断服务程序写入相应中断优先级的中断服务程序地址

如：VICVectAddrn = (int)IRQ_Timer0Interrupt

4、使能外部中断 VICIntEnable

4.4 中断源

下表列出了每一个外设功能的中断源。

中断类型号		中断类型号	
0	WDT(Watchdog Interrupt)	10	SPI0
1	-	11	SPI1
2	ARM Core	12	PLL
3	ARM Core	13	RTC
4	TIMER0	14	System Control(External Interrupt0(EINT0))
5	TIMER1	15	System Control(External Interrupt1(EINT1)
6	UART0	16	System Control(External Interrupt2(EINT2)
7	UART1	17	System Control(External Interrupt3(EINT3)
8	PWM0 (Match 0-6(MR0-6))	18	A/D

4.5 外部中断

LPC2114/2124/2210/2212/2214（包括我们使用的 ARM 芯片）含有 4 个外部中断输入（作为可选的引脚功能，即可以通过 PINSEL0/1 寄存器设置相应引脚为外部中断功能）。 外部中断输入可用于将处理器从掉电模式唤醒。

外部中断的相关寄存器有:

EXINT 外部中断标志寄存器 包含 EINT0、EINT1、EINT2 和 EINT3 的中断标志。
EXTWAKE 外部中断唤醒寄存器 包含 3 个用于控制外部中断中断是否将处理器从掉电模式唤醒的使能位。
EXTMODE 外部中断方式寄存器 控制由每个引脚的边沿或电平触发中断
EXTPOLAR 外部中断极性寄存器 控制由每个引脚的哪种电平或边沿来触发中断

在对外部中断进行初始化设置时要注意，用 EXINT 寄存器选择外部中断 0，然后利用 EXTMODE 寄存器选择边沿触发中断。电平触发中断和边沿触发中断的区别在于：电平触发中断是当输入电平为低或高时，将一直触发中断服务程序直到电平变为高或低；而边沿触发则是当电平发生跳变时才触发一次中断服务程序；根据不同需要选择不同触发方式。在我们的实验中要选取边沿触发。根据 EXTPOLAR 寄存器可以设置每个引脚的哪种电平或边沿来触发中断。

外部中断的初始化设置与一般的终端初始化设置基本一致，只是要多加属于外部中断自己的一些寄存器操作。

4.6 脉冲输入模块

脉冲输入模块中的红外线发送－接收对管用作脉冲输入

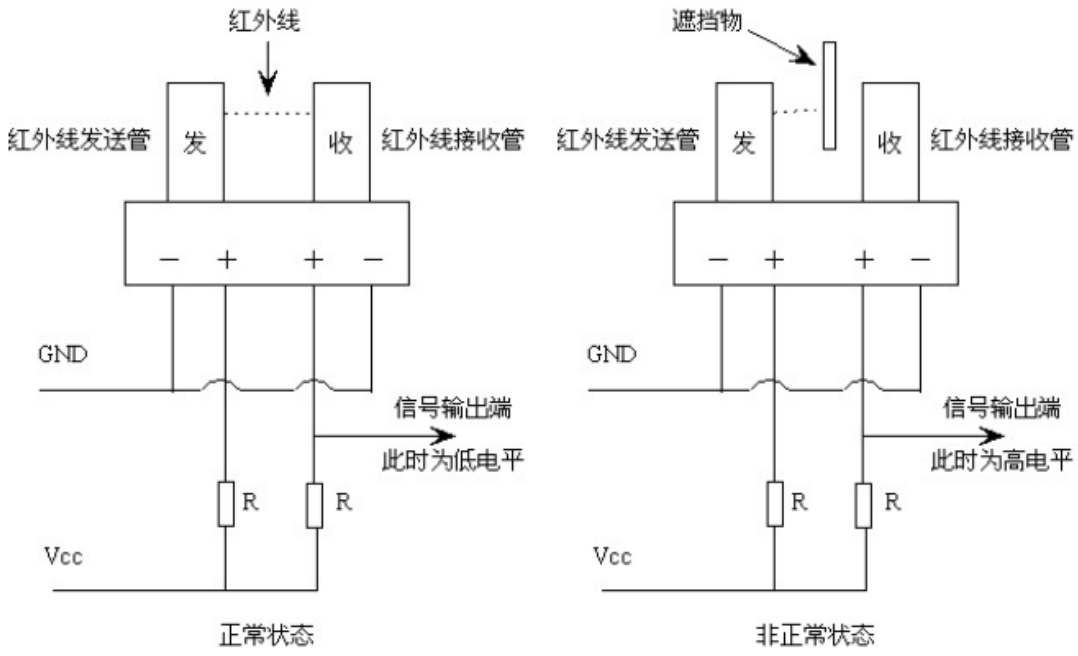


图 4-1 红外线发送-接收对管作无触点计数脉冲原理图

在正常状态下，发送管发送红外线，接收管接收，信号输出端输出为低电平；若发送管和接收管之间插入一件对红外线有遮挡的物体，接收管不能接收到红外线，此时信号输出端输出高电平。如果遮挡物是循环连续运动的话，该装置就是一个非常好的无触点信号源。

2.6.5 实验流程图

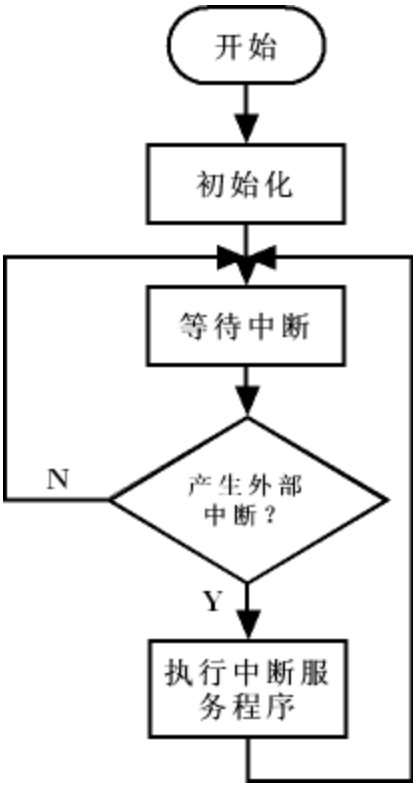


图 4-2 外部中断总体流程图

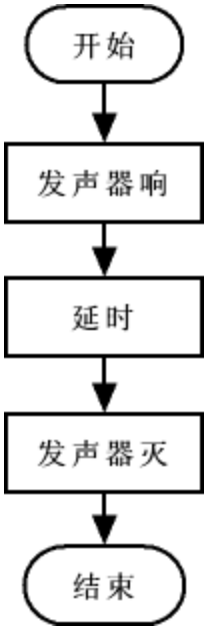


图 4-3 外部中断服务程序（结合代码）

2.6.6 参考源代码

2.6.7 思考题

7.1 如果改用外部中断 1 可不可以实现该外部中断实验？

7.2 能否实现一个利用外部中断进行计数的程序，相应显示在 LED 灯中？

2.7 定时器实验

2.7.1 实验目的

1.1 了解定时器工作的基本原理。

1.2 掌握定时器的各个寄存器的功能和设置方法。

1.3 了解 ARM 中定时器 0 和定时器 1 的基本用途。

2.7.2 实验内容

2.1 使用定时器 0 实现 1 秒定时，让蜂鸣器间隔一秒响一秒。

2.7.3 实验预习要求

3.1 预习定时器 0 和定时器 1 的章节，了解定时器特性，用途以及引脚。

3.2 对定时器的寄存器进行全面的了解。

2.7.4 实验流程图

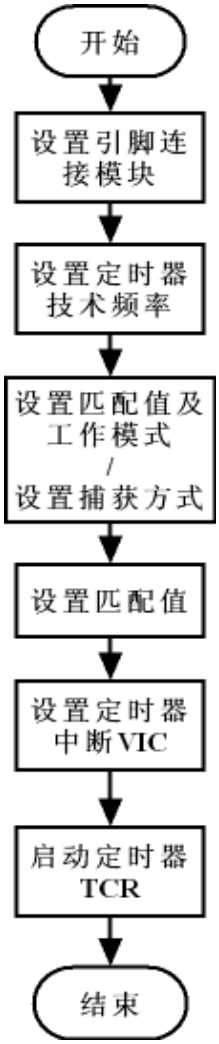


图 5-1 定时器初始化流程图

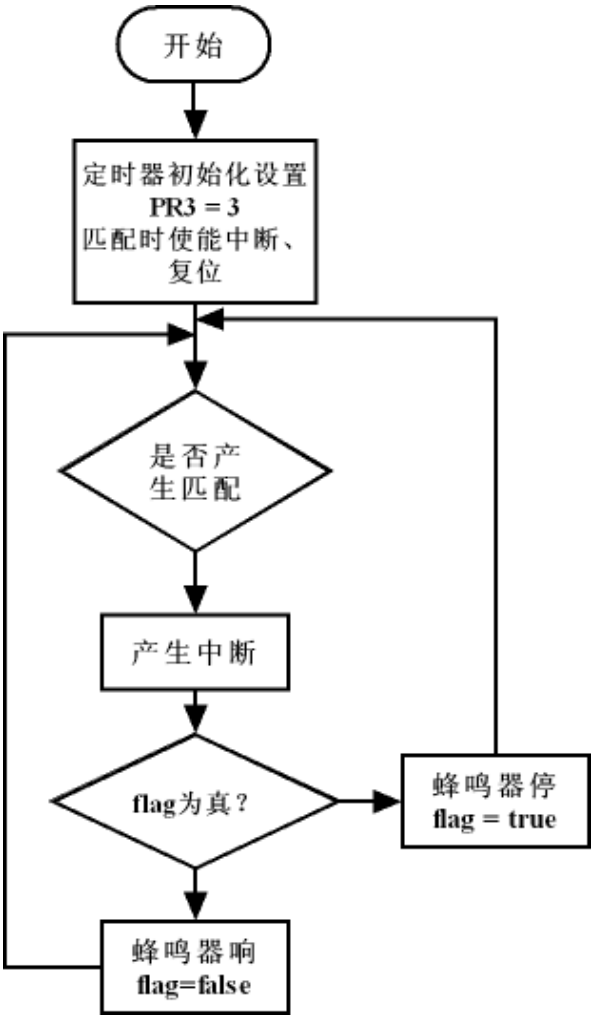


图 5-2 定时器整体流程图（结合代码）

2.7.5 实验原理

定时器对外设时钟（pclk）周期进行计数，根据 4 个匹配寄存器的设定，可设置位匹配（即到达匹配寄存器设定的定时值 0 时）产生中断或执行其他动作。他还包括 4 个捕获输入，用于在输入信号发生跳变时捕获定时器值，并可选择产生中断。

5.1 特性

（1）带有可编程 32 位预分频器的 32 位定时器/计数器；

（2）具有多达 4 路捕获通道。当输入信号发生跳变时可取得定时器的瞬间值。也可选择使捕获事件产生中断；

（3）4 个 32 位匹配寄存器，匹配时的动作有如下 3 种

---匹配时定时器继续工作，可选择产生中断

---匹配时停止定时器，可选择产生中断

---匹配时复位定时器，可选择产生中断

（4）4 个对应于匹配寄存器的外部输出，匹配时的输出有如下 4 种

---匹配时设置为低电平

---匹配时设置为高电平

---匹配时翻转

---匹配时无动作

5.2 相关寄存器描述 （更加详细的寄存器描述参考《ARM 嵌入式系统基础教程》P_{274}）

TCR 定时控制寄存器，控制定时计数器（禁止或复位，是否启用) 1 启用 0 禁止

TC 定时器计数器 32 位计数器，计数频率为 pclk 经过预分频计数器后频率值。（选择做定时器还是计数器，0 定时器，1 计数器）

PR 预分频控制寄存器 用于设定分频值，32 位寄存器,pclk/3 相当于 3 个脉冲时间产生一个上升沿 计数时钟频率=Fpclk/（N+1）

PC 预分频计数器 32 位计数器，计数频率为 pclk，当计数值等于预分频计数器的值时，TC 计数器加一

IR 中断标志寄存器 读该寄存器识别中断源，写该寄存器清除中断标志

MCR 匹配控制寄存器 用于控制在匹配时是否产生中断或复位 TC

位 功能

0 中断(MR0) 为 1 时，MR0 与 TC 值的匹配将产生中断

1 复位(MR0) 为 1 时，MR0 与 TC 值的匹配将使 TC 复位

2 停止(MR0) 为 1 时，MR0 与 TC 值的匹配将清零 TCR 的 bit0 位，使 TC 和 PC 停止

5:3 MR1

8:6 MR2 与上对应相同

9:11 MR3

MR n 匹配寄存器 n，通过 MCR 寄存器可以设置匹配发生时的动作

EMR 外部匹配寄存器 EMR 控制外部匹配管脚 MATx.0~MATx.3

CCR 捕获控制寄存器，用于设置捕获信号的触发特征（下降沿捕获还是上升沿），以及捕获发生时是否产生中断

CRn（1~3） 捕获寄存器 n，在捕获 n 引脚上产生捕获时间时，CRn 装载 TC 的值

5.3 定时器初始化步骤：

1、计数定时器计数频率 （设置 T0PR）
2、设置匹配值及工作模式 （设置 T0MCR）
3、设置捕获方式 （设置 T0CCR）
4、设置定时器中断 （设置 VIC）
5、启动定时器 （设置 TCR）

5.4 中断函数

ads1.2 规定，在定义中断服务函数时，必须加入关键字__irq，保证函数返回时会切换处理器模式。需要注意的是：注意在退出中断服务程序时，要清零相应外设的中断标志以及 VICVectAddr，为响应下一次中断做好准备 。

在中断函数中我们添加需要处理的事件程序。

2.7.6 参考源代码

2.7.7 思考题

7.1 如何设置定时器 1?

7.2 怎样将蜂鸣时间改为两秒一间隔？

7.3 结合数码管程序，把数码管扫描放到定时器中。

2.8 UART 串口通信模块

2.8.1 实验目的

1.1 了解计算机常用的接口。

1.2 掌握串口通信的原理，能实现实验箱与 PC 的通信。

2.8.2 实验内容

2.1 了解串口通信的数据帧格式及波特率设置。

2.2 实现实验箱与 PC 的通信，包括从在 PC 上显示实验箱的键盘按键，在实验箱的数据管显示 PC 发送的数据。

2.8.3 实验预习要求

3.1 查找相关串口资料，了解串口工作的基本流程。

2.8.4 实验原理

4.1 串口接口简介

计算机常见的接口包括：并行接口，串行接口，磁盘接口(IDE 接口，EIDE 接口)，SCSI 接口，USB 接口等。其中串口是计算机上一种常用的设备通信协议，大多数计算机包含两个基于 RS 232 的串口。串口通讯对嵌入式设备(单片机)而言意义重大，因为它不但可以实现单片机的数据传输到电脑端，而且也能实现电脑对单片机的控制，还能实现设备之间的通信。

4.2 串口数据帧格式及串口通信的控制

串口要实现数据的通信，必须要保证串口发送与接收数据的速率一致和相同的数据帧格式。我们可以通过设置波特率来实现传输数据速率的匹配，所谓波特率是指每秒传送的信息位的数量。如果接收端与发送端的波特率设置不同的话，就会造成数据传输错误。 串口的数据帧格式一般为：

注：1.数据帧起始位为0,数据帧结束为1（结束位可以是1个或2个1）
2.D0~D7是位有效数据（可以选择数据的位数为5~8位）
3.P为奇偶校验位

U0LCR寄存器：

位	7	6	[5:4]	6	2	[1:0]
功能	除数锁存	间隔	奇偶选择	奇偶设置	停止位	字长

图 6-1 数据帧格式及相应寄存器设置

我们可以通过串行接口实现数据的发送和接收，一般通过查询方式发送数据，通过中断方式接收数据。 其流程图如下：

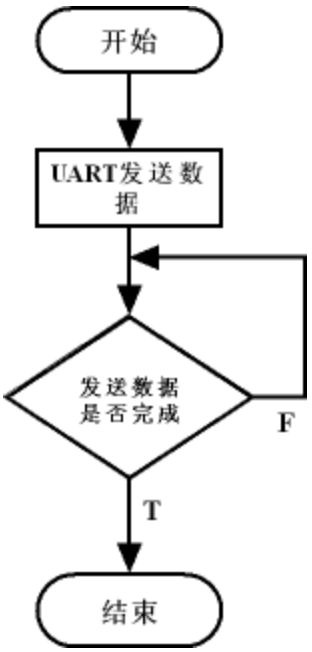


图 6-2 串口发送函数流程图图

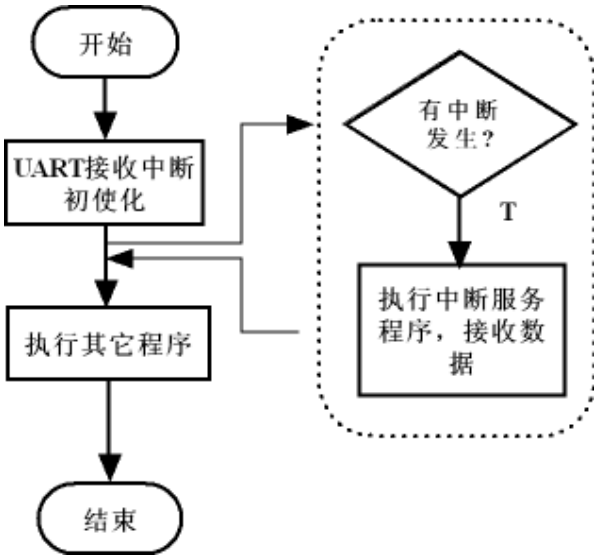


图 6-3 串口接收（中断）函数流程图

4.3 实现步骤

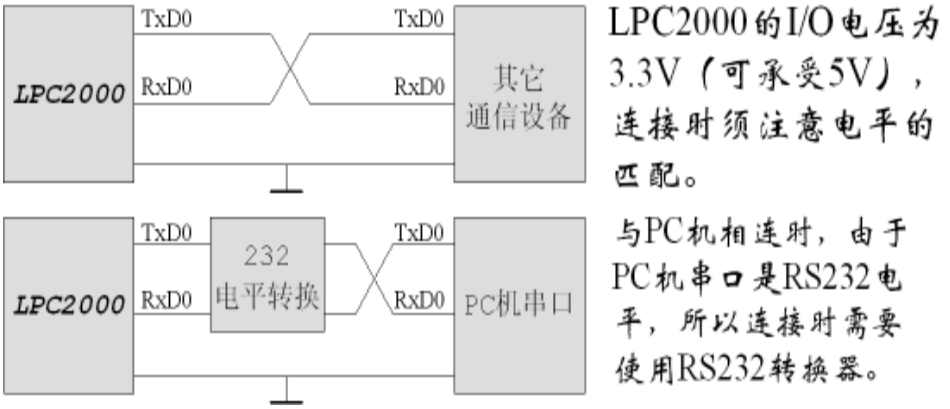


图 6-4 串口连接电路逻辑图

在 LPC2292 中与串口相关的寄存器(串口 0)：

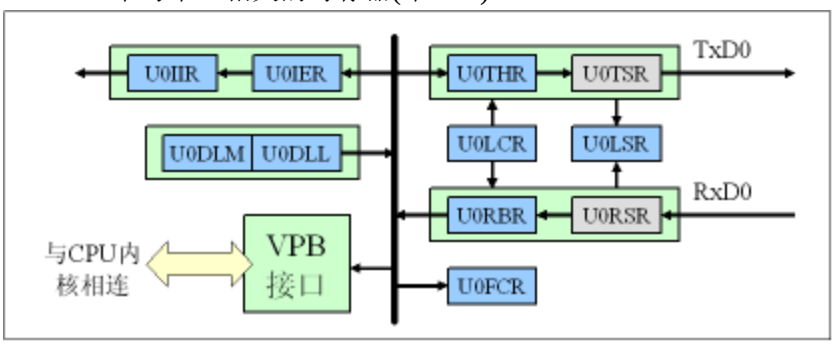


图 6-5 UART0 相关的寄存器

其中：(1)U0IER,U0IIR：中断使能与中断标志寄存器。

(2)U0DLM,U0DLL：波特率设置寄存器。

(3)U0THR,U0TSR：发送缓存与发送移位寄存器。

(4)U0RBR,U0RSR：接收缓存与接收移位寄存器。

(5)U0LCR：设置 UART0 格式(数据帧格式)。

(6)U0FCR：FIFO 控制。

(7)U0LSR：UART0 当前状态寄存器。

在使用 UART0 之前需要设置的寄存器：

U0IER：中断的设置(中断优先级，中断服务程序地址等)。

U0DLM,U0DLL：波特率 (每秒传送的信息位的数量，一般设置为 9600) 设置。

U0LCR：数据帧格式设置(包括有效数据位数，奇偶校验，结束位数等)。

U0FCR：使能 FIFO 与触发设置。 一般用默认值。

2.8.5 实验流程图

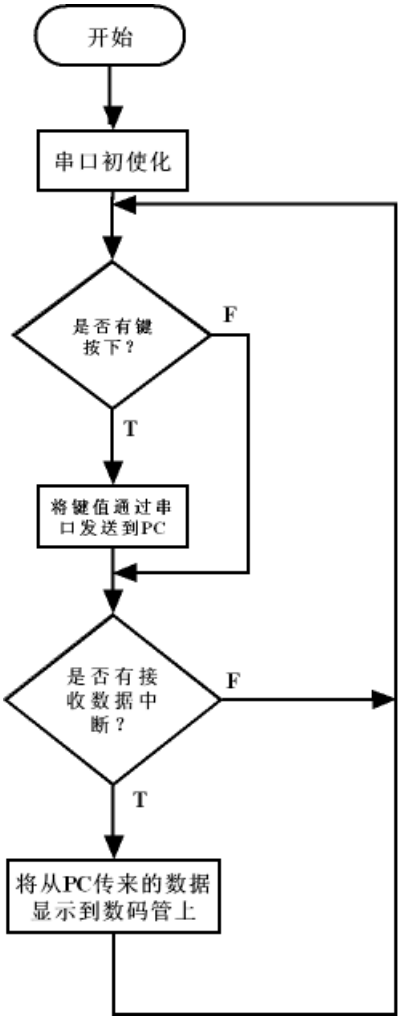


图 6-5 串口工作流程图

2.8.6 参考源代码

2.8.7 思考题

7.1 串口与并口有什么不同？

2.9 I2C 存储程序

2.9.1 实验目的

1.1 理解并掌握 I2C 工作原理！

1.2 理解并掌握 AT24C02 工作方式熟练掌握你存储方式以及读写各状态之间的转换和意义

2.9.2 实验内容

2.1 利用实验箱向 AT24C02 以字节形式写入并读出，用数码管显示

2.9.3 实验要求与预习

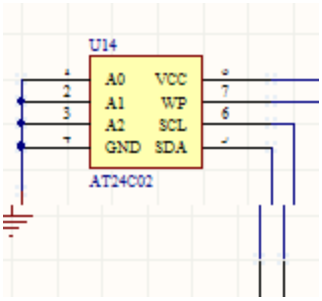
3.1 参考 ARM 嵌入式系统基础教程（PPT）了解 I2C 工作原理。

3.2 读 ATC2402 数据手册，理解你其操作原理

2.9.4 实验原理

4.1 总体介绍

I2C 接口是 Philips 推出的一种串行总线方式，用于 IC 器件之间的通信。它通过 SDA 和 SCL 两根线连到总线上的器件之间传递信息，并通过软件寻址识别到每个器件，而不是片选线。而对 SDA 和 SCL 则是通过器件之间发生应答和非应答信号。而关联应答和非应答信号，则会对器件产生不同状态，并通过状态的不同对器件操作。



4.2 写函数

写函数带选择地址字节写的状态（已经发送启示条件）0x08（装入器件地址和读写位）—（接受可器件地址和读写位并已经接受 ACK）0x18（将发送数据，即 at24c02 的存储地址）---（接收到 24C02 的写入地址，接受 ACK）0x28（将发送写入数据）---（接受到存储数据，接受到 ACK）0x28（发送停止条件，STO 复位），每一状态参考课本设置下一次操作的初始状态！

4.3 读函数

读函数带选择地址字节读的状态 0x08—0x18（发送的要读地址）---（前面跟写一样）0x28（将置位 STA 将从新发送启始条件）—（已经发送启始条件）0x10（将发送器件地址以及读状态）—（接受到状态信息）0x40（将接收数据，返回非 ACK）—（接收到数据）0x58（提取数据）

启动 i2c 后，循环等待中断，每到达一状态产生一次中断、直到完成，停止循环！

具体状态转换请仔细读 ARM 嵌入式系统基础教程（ppt）和 ATC2402 数据手册

2.9.5 实验流程图

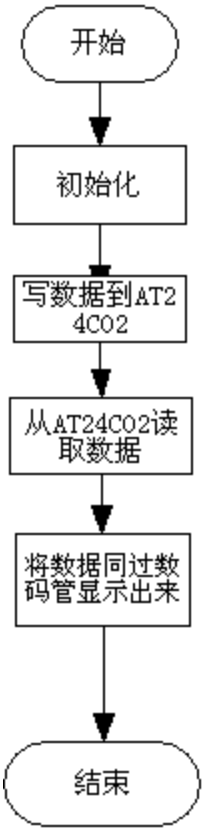


图 7-1 总的流程图

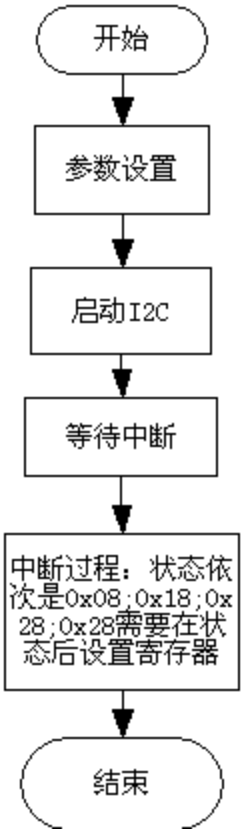


图 7-2 写函数流程图

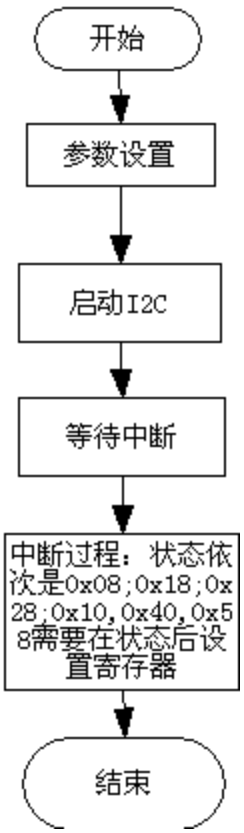


图 7-3 读函数流程图

2.9.6 参考源代码

2.9.7 思考题

7.1 如何完成字符串写入和带选择地址连续读

2.10 电机实验

2.10.1 实验目的

1.1 了解步进电机、直流电机工作的基本原理。

1.2 学习步进电机、直流电机的控制原理和控制方法。

2.10.2 实验内容

2.1 实现直流电机不同转速的转动。

2.2 利用定时器实现步进电机的不同转速的转动，并可以控制步进电机的正转和逆转。

2.10.3 实验预习要求

3.1 适当复习端口操作章节。

3.2 在网上查找步进、直流电机相关资料，了解基本原理并结合电机连接引脚熟悉控制原理。

2.10.4 实验原理

4.1 步进电机

步进电机是一种将电脉冲转化为角位移的执行机构。当步进驱动器接收到一个脉冲信号，它就驱动步进电机按设定的方向转动一个固定的角度(称为“步距角”)，它的旋转是以固定的角度一步一步运行的。可以通过控制脉冲个数来控制角位移量，从而达到准确定位的目的；同时可以通过控制脉冲频率来控制电机转动的速度和加速度，从而达到调速的目的。步进电机可以作为一种控制用的特种电机，利用其没有积累误差(精度为 100%)的特点，广泛应用于各种开环控制。

步进电机的一些基本参数：

4.1.1 电机固有步距角：

它表示控制系统每发一个步进脉冲信号，电机所转动的角度。电机出厂时给出了一个步距角的值，如 86BYG250A 型电机给出的值为 0.9°/1.8°（表示半步工作时为 0.9°、整步工作时为 1.8°），这个步距角可以称之为‘电机固有步距角’，它不一定是电机实际工作时的真正步距角，真正的步距角和驱动器有关。

4.1.2 步进电机的相数：

是指电机内部的线圈组数，目前常用的有二相、三相、四相、五相步进电机。电机相数不同，其步距角也不同，一般二相电机的步距角为 0.9°/1.8°、三相的为 0.75°/1.5°、五相的为 0.36°/0.72° 。在没有细分驱动器时，用户主要靠选择不同相数的步进电机来满足自己步距角的要求。如果使用细分驱动器，则‘相数’将变得没有意义，用户只需在驱动器上改变细分数，就可以改变步距角。

4.1.3 保持转矩（HOLDING TORQUE）：

是指步进电机通电但没有转动时，定子锁住转子的力矩。它是步进电机最重要的参数之一，通常步进电机在低速时的力矩接近保持转矩。由于步进电机的输出力矩随速度的增大而不断衰减，输出功率也随速度的增大而变化，所以保持转矩就成为了衡量步进电机最重要的

参数之一。比如，当人们说 2N.m 的步进电机，在没有特殊说明的情况下是指保持转矩为 2N.m 的步进电机。

4.1.4 DETENT TORQUE：

是指步进电机没有通电的情况下，定子锁住转子的力矩。DETENT TORQUE 在国内没有统一的翻译方式，容易使大家产生误解；由于反应式步进电机的转子不是永磁材料，所以它没有 DETENT TORQUE。

步进电机的一些特点：

（1）一般步进电机的精度为步进角的 3-5%，且不累积。

（2）步进电机外表允许的最高温度。

（3）步进电机的力矩会随转速的升高而下降。

（4）步进电机低速时可以正常运转,但若高于一定速度就无法启动,并伴有啸叫声。

在我们的实验箱中，与步进电机相关联的引脚有 P2.19～P2.22，分别对应相位 D、C、B、A。依次对 ABCD 相位的端口置 0，可以实现步进电机的正转；依次对相位 DCBA 的端口置 0，可以实现逆转。通过对定时器中断间隔的调整，可以控制转速。

4.2 直流电机

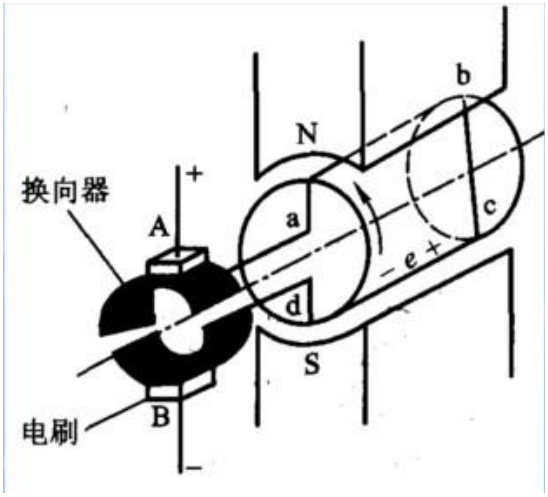


图 8-1 直流电机原理图

如图所示， N、S 为一对固定的磁，两磁极间装着一个可以转动的铁质圆柱体，圆柱体的表面上固定着一个线圈。N 极与 S 极的磁力线所通过圆柱体的途径如图中所表示。当线圈中通入直流电流时，线圈边上受到电磁力，根据左手定则确定力的方向，这一对电磁力形成了作用于电枢的一个电磁转矩，转矩的方向是逆时针方向。若电枢转动，线圈两边的位置互换，而线圈中通过的还是直流电流，则所产生的电磁转矩的方向却变为顺时针方向了，因此电枢受到一种方向交变的电磁转矩。这种交变的电磁转矩只能使电枢来回摇摆，而不能使电枢连续转动。显然，要使电枢受到一个方向不变的电磁转矩，关键在于，当线圈边在不同极性的磁极下，如何将流过线圈中的电流方向及时地加以变换，即进行所谓“换向”。为此必须增添一个叫做换向器的装置，换向器由互相绝缘的铜质换向片构成，装在轴上，也和电枢绝缘，且和电枢一起旋转。换向器又与两个固定不动的由石墨制成的电刷 A、B 相接触。装了这种换向器以后，若将直流电压加于电刷端，直流电流经电刷流过电枢上的线圈，则产生电磁转矩，电枢在电磁转矩的作用下就旋转起来。电枢一经转动，由于换向器配合电刷对电流的换向作用，直流电流交替地由线圈边 ab 和 cd 流入，使线圈边只要处于 N 极下，其中通过电流的方向总是由电刷 A 流入的方向，而在 S 极下时，总是从电刷 B 流出的方向。

这就保证了每个极下线圈边中的电流始终是一个方向。这样的结构，就可使电动机能连续地旋转。

直流电机调速：

实验箱通过端口 P2.23 控制直流电机。当 P2.23 为低电平，直流电机转动；当 P2.23 为高电平时，直流电机停止转动。要控制直流电机的转速，只有通过改变脉冲来实现。这里通过固定时间让电机转动再停止来实现转速的控制。在 P2.23 端口输出为方波的情况下，通过改变方波的占空比可以调节直流电机电流的大小，从而改变直流电机的转动速度。

不同占空比的方波波形：

图 8-2 占空比为 1：1 的方波

图 8-3 占空比为 5：1 的方波

2.10.5 实验流程图

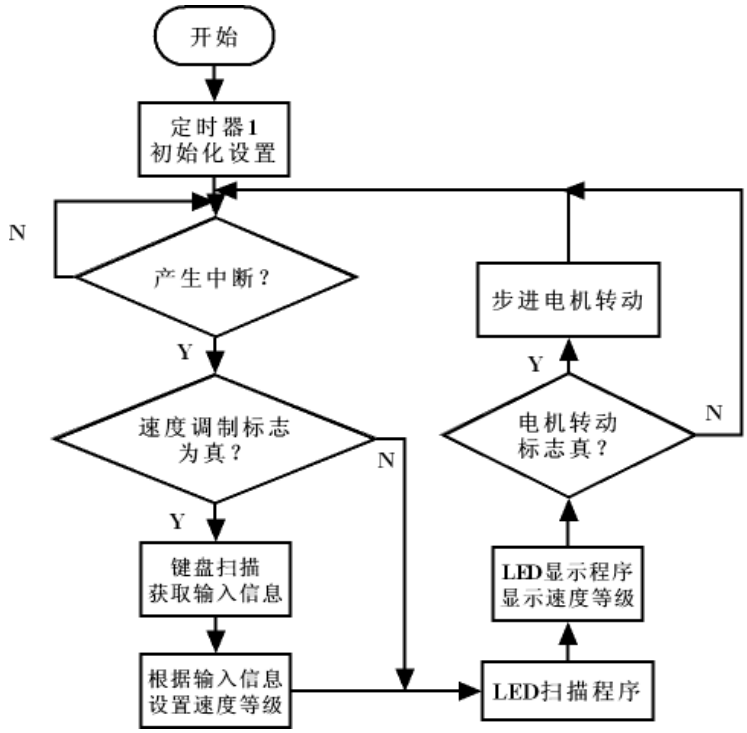


图 8-4 Timer1 程序流程图

2.10.6 参考源代码

2.10.7 思考题

7.1 怎样给直流电机添加键盘功能，实现实时改变速度等级？

7.2 怎样精确控制步进电机转动的角度？

7.3 3、只用一个定时器能实现功能吗？

2.11 LCD 液晶显示模块

2.11.1 实验目的

1.1 了解 LCD 液晶显示器的基本原理

1.2 学会使用 LCD 液晶显示模块显示信息

2.11.2 实验内容

2.1 掌握 LCD 液晶显示器的工作原理。

2.2 掌握 LCD 使用方法，能根据液晶显示模块指令实现字符及图形的显示。

2.11.3 实验预习要求

3.1 查找 LCD 的资料，了解 LCD 的基本原理及特点。

2.11.4 实验原理

4.1 液晶显示器的物理原理

液晶显示器(LCD)是现在非常普遍的显示器。它具有体积小、重量轻、省电、辐射低、易于携带等优点。液晶显示器（LCD）的原理与阴极射线管显示器（CRT）大不相同。LCD 是基于液晶电光效应的显示器件。液晶分子是一种规则性排列的有机化合物，它是一种介于固体和液体之间的物质，液晶本身并不能构发光。液晶的物理特性是：当通电时导通，排列变的有秩序，使光线容易通过；不通电时排列混乱，阻止光线通过。让液晶如闸门般地阻隔或让光线穿透。所以可以通过电压的更改产生电场而使液晶分子排列产生变化来显示图像。

4.2 液晶显示模块的控制原理

实验箱上 128 * 64 像素的液晶显示屏是由两个 64 * 64 像素的液晶显示屏组合的。其逻辑电路图为：

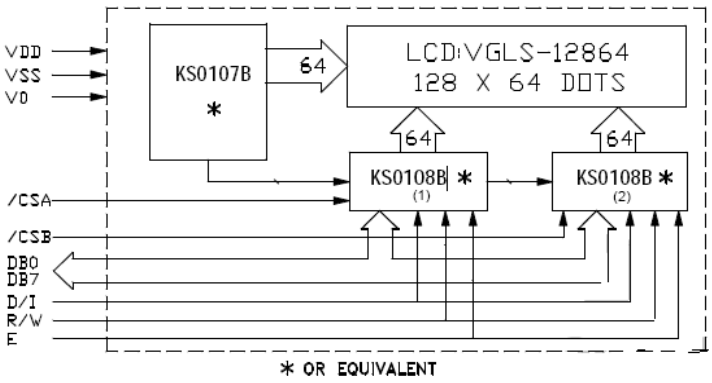


图 9-1 MGLS-12864 的逻辑电路图

(注：R/W,D/I 可以直接连上地址线 A1,A0 上，通过控制地址来实现对 LCD 内部 ARM 的操作)

液晶显示模块的指令系统由四部分组成：写入指令，写入数据，读取状态，读取数据

(1)写入指令：向 LCD 存储器中写入控制信息。控制信息可以分为：设置显示开关，设置页地址，设置列地址等(页地址和列地址唯一确定 LCD 内部 RAM 中的一个单元)。

(2)写入数据：向 LCD 内部 RAM 中写入将要显示的数据信息。如果我们想使 LCD 某位亮，那么我们要作的操作就是找到该位的地址（通过设置页和列地址实现），向该位数据位写入 1。

(3)读取指令：可以得到 LCD 的状态（空闲/忙，开/关，复位/正常）。

(4)读取数据：读取 LCD 内 RAM 中的数据信息(读写数据指令每执行完一次读写操作列地址就自动增一)。

每个 LCD 像素的亮或灭是由一位二进制数据来控制的。1 表示是该像素亮，而 0 表示该像素灭。如果我们要且此 LCD 显示一幅图片，我们可以用 128 * 64 Bit 的数据来表示该图片的信息。然后我们将这些数据写入到 LCD 内部的存储器中，LCD 即可显示我们想要的图片。LCD 内部存储器的逻辑图为：

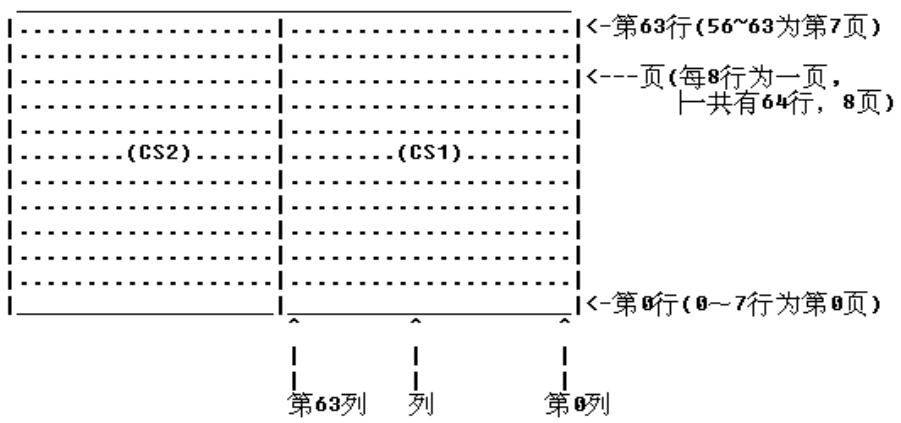


图 9-2　MGLS-12864 的逻辑电路图

4.3　实现步骤

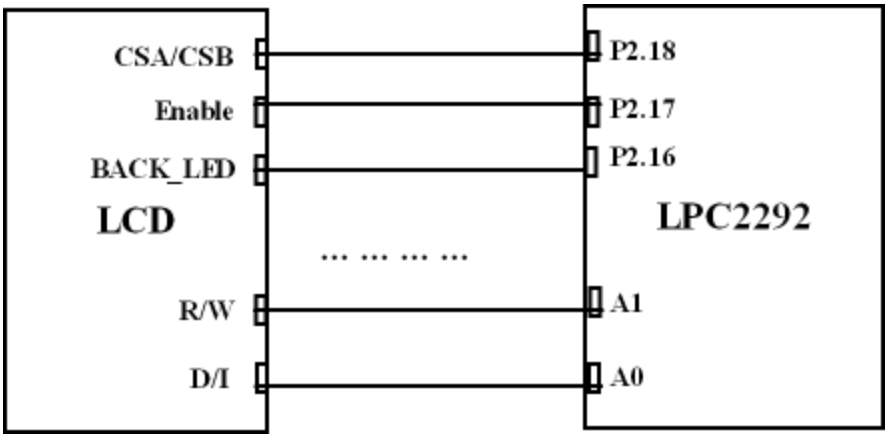


图 9-3　　LCD 与 LPC2292 的连接图

(1)　连接跳线(P2.16，P2.17，P2.18)。

(2)　初使化 LCD(设置数据线为 8 位，设置端口方向)。

(3)　把要显示的内容转换，存放在 LCD 缓冲区中（查找字库,图库->转换成 128*64　位大小的数据放在 LCD 缓冲区中）。

(4)　将 LCD 缓冲区中的数据，逐行写入到 LCD 内部 RAM 中去。

2.11.5 实验流程图

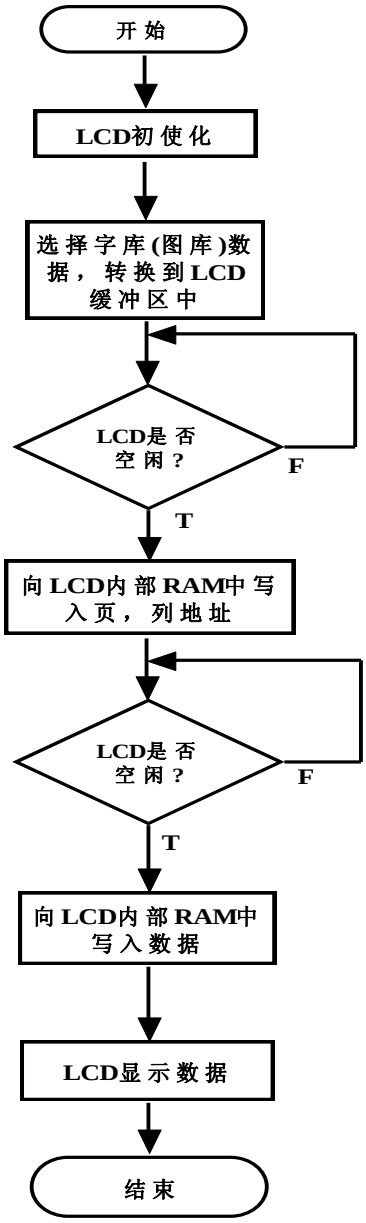


图 9-3 LCD 显示数据流程图

2.11.6 参考源代码

2.11.7 思考题

7.1 试比较控制 LCD 与控制点阵 LED 有什么不同？

7.2 怎样实现用 LCD 显示一幅自己的图片？

7.3 AD 转换程序

2.12 AD 转换

2.12.1 实验目的

1.1 掌握模数转换的基本原理，熟悉 ARM AD 转换功能特点，转换性能以及编程方法。

2.12.2 实验内容

2.1 利用旋转按纽改变输出数电的大小，并用数码管显示

2.12.3 实验要求与预习

3.1 参考 ARM 嵌入式系统基础教程（PPT）了解 AD 转换原理，并理解其寄存器的设置方式

2.12.4 实验原理

4.1 ADCR 设置

ad 转换相对于其他模块较为简单，主要是对 ADCR 的设置以及读取 ADDR。ADCR 主要包括一、0~7 位，SEL 通道设置如选通 0 通道则是 0x01.二、 8~15 位 CLKDIV AD 转换时钟，需要小于 4.5MHZ.三、 1 6 位.BURST 触发模式选择，两种方式： 1 burst，需对该位置，另外一种为软件方式。四.17~19 位 CLKS 转换位数以及转换时间五 24~26 位控制启动方式，软件模式有效。六，27 位，上升下降沿出发选择（START 010~111 情况下）。一般对 ADCR 模式选位 burst 模式而不是软件模式。ADDR 的数据读写，是通过查询最高位 DONE 为 1 表示转换成功。故读取转换好的 6 ～ 1 5 位的 10 位数据

4.2 具体设置方式参考参考 ARM 嵌入式系统基础教程（PPT 或者书本）

2.12.5 实验流程图

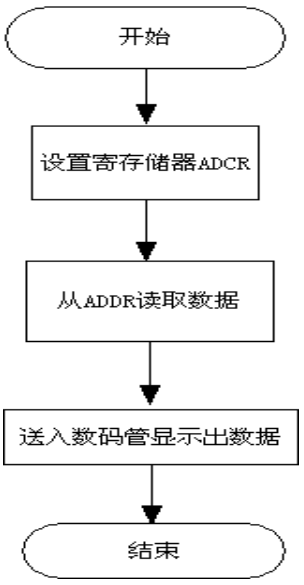


图 10-1 系统流程图

2.12.6 参考源代码

2.12.7 思考题

7.1 试采用两种触发方式，做出 AD 转换，

2.13 DA 转换

2.13.1 实验目的

1.1 掌握数/模转换的基本原理。

1.2 学会使用 TCL5620 实现数字信号向模拟信号的转换。

2.13.2 实验内容

2.1 实现 TCL5620 的数/模转换功能。

2.2 用示波器（万用表）验证其正确与否。

2.13.3 实验预习要求

3.1 查找相关串口资料，了解数模转换的原理。

2.13.4 实验原理

4.1 D/A 简介

D/A 转换器的内部电路构成无太大差异，大多数 D/A 转换器由阻陈列和 n 个电流开关（电压开关）构成。按输入的数据切换开关，产生比例输入的电流(电压)。D/A 转换器的原理图为：

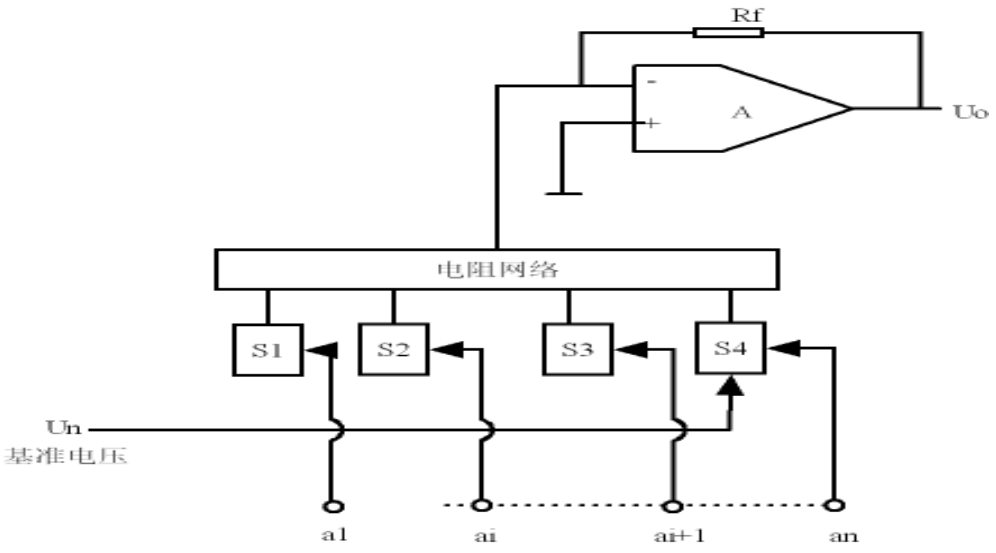


图 11-1 D/A 原理图

4.2 D/A 转换的实现

CS-II 实验箱采用的 D/A 芯片是 tlc5620,它有四个输出端口可以选择。它的一个数据帧格式为：

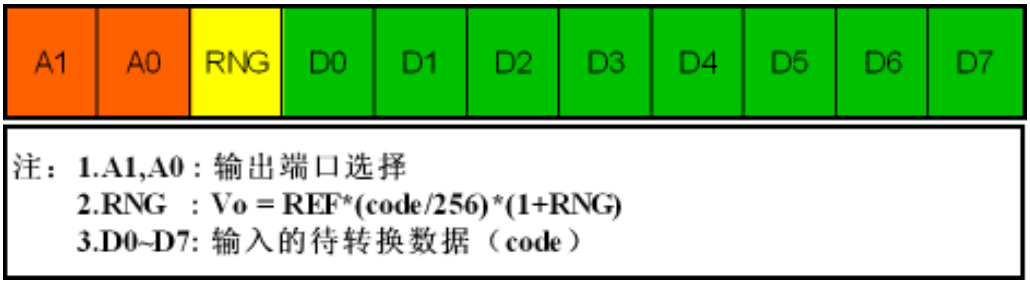


图 11-2 D/A 数据帧格式

实现 D/A 数据更新有几种模式，我们选择第三种模式，它的时序图为：

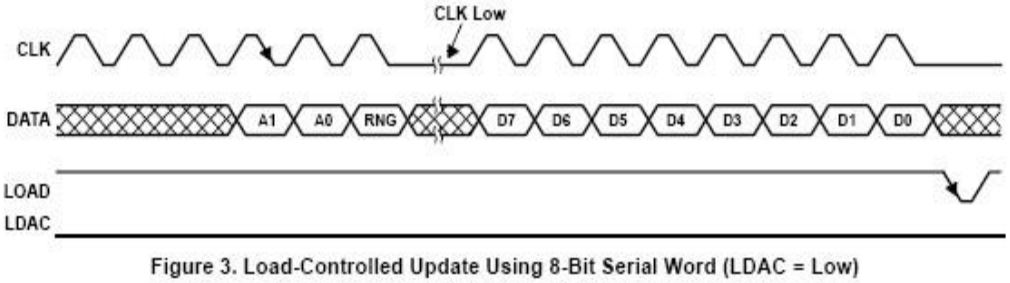


Figure 3. Load-Controlled Update Using 8-Bit Serial Word (LDAC = Low)

图 11-3 tlc5620 实现 DA 转换的时序图（注：LOAD 为 RCK）

Tlc5620 是串口输入的并行 DA 转换芯片。CLK 一次下降沿将一位数据写入到 D/A 内部寄存器，RCK 一次下降沿将更新 DA 的模拟输出。

4.3 实现步骤

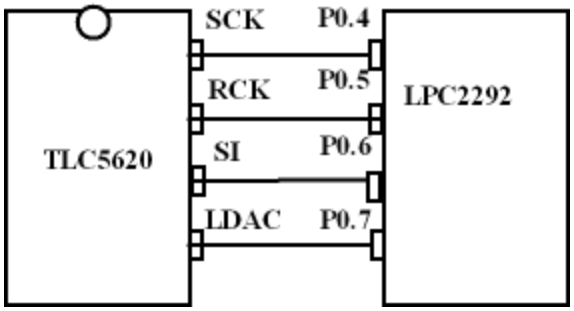


图 11-4 DA 与 LPC2292 连接图

(1) 连接跳线(RCK,SCK,SI,LDAC)。
(2) 设置端口方向(输出)。
(3) 输出端口选择，RNG 选择。
(3) 将数据帧写入到 DA 内部的寄存器中。

2.13.5 实验流程图

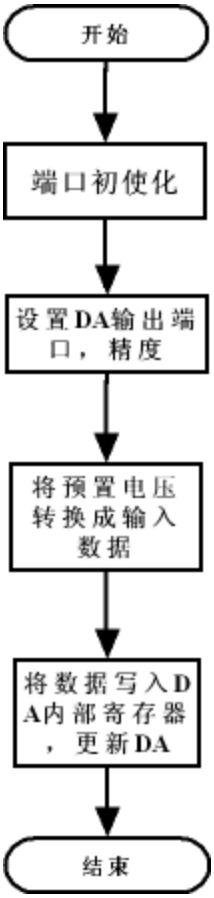


图 11-5 串口工作流程图

2.13.6 参考源代码

2.13.7 思考题

7.1 D/A 转换还有哪些芯片?怎样实现对其编程？

Experimental Instructor

计算机科学与技术学院

(School of Computer Science and Technology)

计算机科学与技术学院自编实验指导书

软件项目管理

计算机基本技能训练

计算机网络

计算机组成原理

C++程序语言设计

计算机图形学

编译原理

软件测试技术

汇编语言程序设计

网络程序设计

反病毒技术

Linux 系统及程序设计

Windows 程序设计

软件开发综合实验

Java 程序设计

网络攻防对抗

计算机制图

微机原理及应用

嵌入式系统实验

数据库应用设计

单片机与接口应用设计

信息电器分析与设计

数据结构应用设计

计算机操作系统综合实验

动态网页制作技术

数学实验

面向对象技术应用设计

信息安全综合实验