To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

应用笔记

在 78K0/Lx2 LCD 控制器上显示数据

文档编号：U18273CA1V0AN00(第一版)
发行日期：2008 年 3 月

日本印制

目录:

简介

本文档的目的是提供 NEC 微控制器设备中包含的外部硬件的简单示例。用户通过本文档可以更好的了解这些外部硬件的使用。

本文档包含以下内容：

- 外部硬件功能的描述
- 示例程序的描述和规格
- 软件流程
- Applilet 参考驱动
- 使用的示范平台
- 硬件模块图
- 软件模块

Applilet 是一个产生外部硬件驱动代码的软件工具。对于片上外部硬件的评估来说是一个方便的代码产生器。

用户应该参考设备的用户手册和其它相关文档以便了解更加详细的信息。

LCD 控制器的概述

NEC 微控制器提供不同段数驱动能力的，易于连接和使用的 LCD 控制器。当参考电压产生后，段和公共信号允许自动显示从存储器读出的数据，LCD 控制器支持多种标准特性，包含从静态到多分时显示等多种模式。

1 78K0/Lx2系列微控制器的LCD数据显示模式

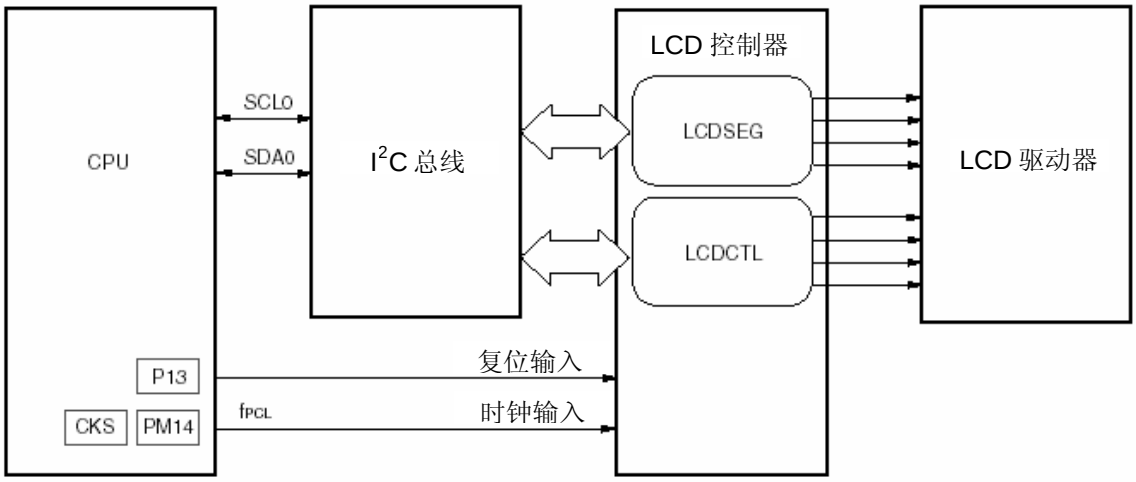
在 LCD 上显示数据就像在存储器中写入准备好的数据一样简单。一旦数据被写入存储器，就会驱动 LCD 面板上相应的段。在所有类型的偏置中，LCD 控制器负责参考电压和控制分频比周期。我们将会以 78K0/Lx2 系列微控制器中的 78K0/LG2 (uPD78F0397) 作为示例。

1.1 LCD 控制器特性

78K0/LG2 系列微控制器包含了片上 LCD 控制器，它具有如下特性：

- 产生内部 LCD 电源，可以驱动更高电压的 LCD 面板
- 通过分压电阻可以选择外部 LCD 参考电压
- 五种显示模式:
  - 静态
  - 1/2 分时，1/2 偏置
  - 1/3 分时，1/2 偏置
  - 1/3 分时，1/3 偏置
  - 1/4 分时，1/3 偏置
- 内部 LCD 帧时钟，可以进行多种帧频率的选择
- 4 条公共线路和 40 段线路，最多可以驱动 160 LCD 显示元素

下图为 78K0/LG2 CPU 和 LCD 控制器的模块图。用方框标出 LCDSEG 产生和控制段信号。用方框标出的 LCDCTL 提供控制设置设置和显示模式设置。



CPU 模块提供用于 LCD 控制器和使用 IIC 总线接口数据发送的时钟。这些都是内部信号。

条目		配置
LCD 控制器/驱动器	显示输出 (LCDSEG)	40 段信号 4 公共信号(COM0~COM3)
	控制寄存器 (LCDCTL)	LCD 模式设置寄存器(LCDMD) LCD 显示模式寄存器(LCDM) LCD 时钟控制寄存器(LCDC) LCD 电压推进控制寄存器 0(VLCG0)
CPU	控制寄存器	时钟输出选择寄存器(CKS) 端口寄存器 13(P13) 端口模式寄存器(PM14)

这些寄存器按如下方式控制 LCD 的操作：

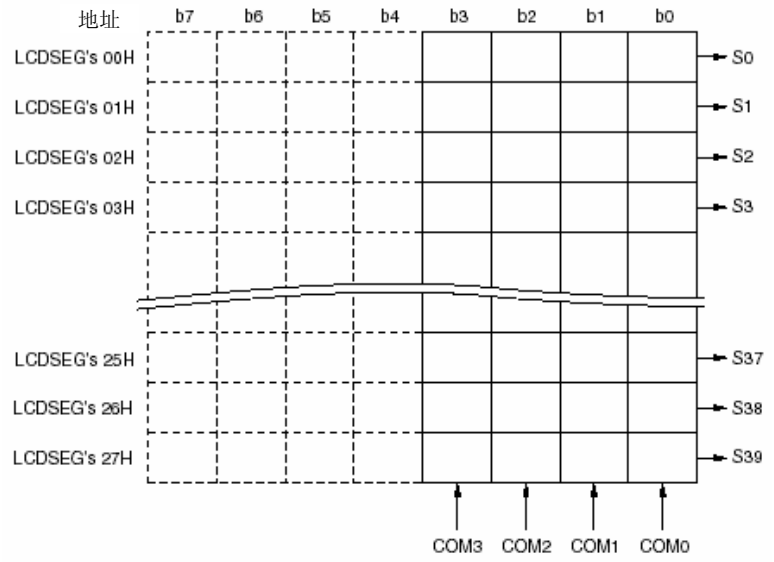
符号	***寄存器名称***	***寄存器功能描述***
LCDMD	模式设置寄存器	设置段的数目和 LCD 参考电压发生器
LCDM	显示模式寄存器	LCD 显示允许/禁止
		段引脚和公共引脚输出电平选择
		电压推进电路允许/禁止
		LCD 显示模式选择：分时和偏置选择
LCDC	时钟控制寄存器	LCD 时钟源和频率选择
VLCG0	电压推进控制寄存器 0	电压推进器操作期间，控制电压推进器电平
CKS	时钟输出选择寄存器	允许/禁止时钟输出(PCL)到 LCD 控制器
		选择 PLC 输出时钟频率
P13	端口寄存器 13	P13 为 LCD 控制器设置或释放复位
PM14	端口模式寄存器 14	允许/禁止时钟输出到 LCD 控制器

当设置 LCD 电压推进器控制寄存器(VLCG0)时，应该参考 LCD 面板电压规格书。

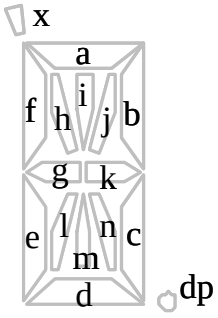
78K0/LG2 的 LCD 控制器在 1/2 分时，1/3 分时和 1/4 分时下支持 40 个段(S0 - S39)输出和 4 个公共(COM0 - COM3)输出。总体上讲，78K0/LG2 的具体特性如下所示。78K0 系列中的其它型号支持不同数量的段信号和公共信号。

LCD 驱动参考电压发生器	偏置模式	时隙数	使用的公共信号	段信号数	最大象素数
外部阻抗分压 内部阻抗分压	–	静态	COM0 (COM1~COM3)	40	40(40 段信号，1 公共信号)[注1]
	1/2	2	COM0, COM1		80(40 段信号，2 公共信号)[注2]
		3	COM0~COM2		120(40 段信号，3 公共信号)[注3]
内部电压推进 外部阻抗分压 内部阻抗分压	1/3	3	COM0~COM2		
		4	COM0~COM3		160(40 段信号，4 公共信号)[注4]

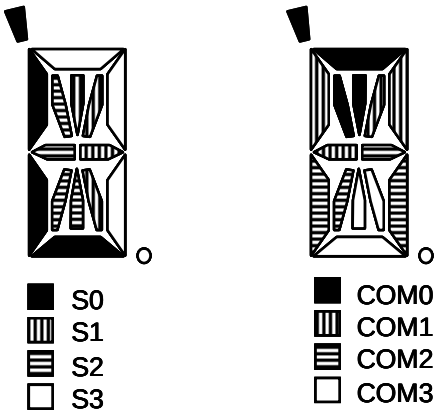
LCD 显示存储器数据映射到 LCDSEG 地址的 00h - 27h。显示数据存储器中的数据可以通过 LCD 控制器在 LCD 面板上显示。下图显示了 LCD 显示数据存储器内容和段/公共信号输出之间的关系。



例如，为了在 14 段数字 LCD 面板上显示数据，段、十进制点以及添加 'x' 段如下所示。



这种段的排列方式允许高质量的字母和数据显示。为了驱动LCD模块，78K0/LG2联合使用段信号线缆 (S0–S39) 和公共信号线(COM0–COM3)。下图显示了段信号线和公共信号线连接的示例。



78K0/LG2 LCD控制器在段和公共信号线上自动驱动波形来点亮或是熄灭特定的段。在内部存储器的相应位写入0或是1可以使相应的液晶段点亮 (该位为1)或是熄灭(该位为0)。下表中显示了每一段对应的内部LCD存储器的相应位，按照上图所示的的方式连接段信号线和公共信号线。

RAM地址	位							
	7	6	5	4	3	2	1	0
	LCD控制器未使用				COM3	COM2	COM1	COM0
SEG+0	0	0	0	0	D	E	F	X
SEG+1	0	0	0	0	N	K	J	I
SEG+2	0	0	0	0	M	L	G	H
SEG+3	0	0	0	0	DP	C	B	A

显示中的每一个数字需要4个RAM位置指定该段是点亮还是熄灭。只使用每个位置的低四位COM0–COM3；RAM位置的地址指定了其驱动的段。例如，为了显示数字“8”，a，b，c，d，e，f，g和k段将被点亮，而其它段将被熄灭。要被存储在RAM中的数据如下所示。

RAM 地址	位								
	7	6	5	4	3	2	1	0	
	LCD 控制器未使用				COM3	COM2	COM1	COM0	
SEG+0	0	0	0	0	1	1	1	0	= 0x0E
SEG+1	0	0	0	0	0	1	0	0	= 0x04
SEG+2	0	0	0	0	0	0	1	0	= 0x02
SEG+3	0	0	0	0	0	1	1	1	= 0x07

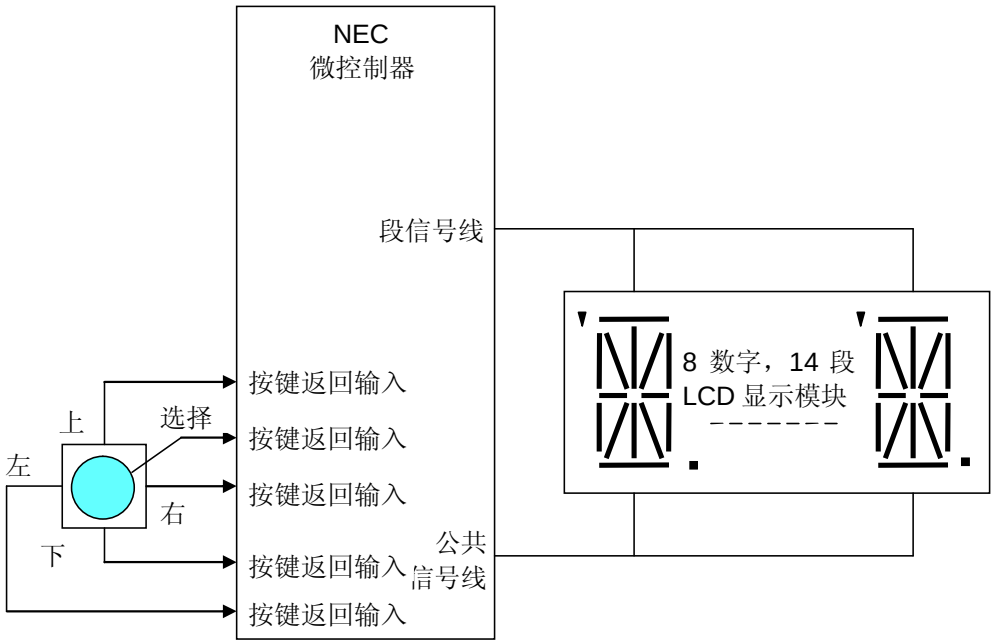
按照如下步骤配置 LCD 控制器：

- 释放 LCD 控制器复位，设置时钟源和打开时钟输出。这些操作需要使 P13.0，PM14.0 和 CKS 寄存器
- 如果需要，在 LCDMD 寄存器中设置内部电压推进
- 在 LCDSEG 存储器中设置初始数据
- 在 LCDM 寄存器中设置 LCD 显示模式
- 在 LCDC 寄存器中设置 LCD 电压推进频率和帧频率
- 使用 VLCG0 寄存器设置电压增益
- 通过 LCDM 寄存器中的 VLCON 位打开电压推进器
- 根据增益和设备 LVDD 电压，等待指定的时间
- 通过设置 LCDM 寄存器中的 SCOC 位为 1 ，将 LCD 显示的公共信号线和段信号线与 GND 断开
- 通过设置 LCDM 寄存器中的 LCDON 位为 1，允许 LCD 显示

至此，显示操作被允许并且开始显示数据。通过向 LCDSEG 存储器写入数据来改变显示的数据。

1.2 程序说明及规格书

本程序将示范使用负极按键作为输入在一个 8 数字，14 段 LCD 面板上进行数字显示。当负极按键被按下时，将显示按键的描述。



- 按下"左"　　在 14 段数字显示上显示"左"
- 按下"右"　　显示"右"
- 按下"上"　　显示"上"
- 按下"下"　　显示"下"
- 按下"选择"　显示"选择"

规格说明:

- 考虑到输入去抖动，按键按下后 5 毫秒应该看到显示相应的数据
- LCD 使用内部产生的电压，推进到 4.5V LCD 电压
- 到微控制器的 LVDD 电压的期望值为 5.0V
- LCD 被配置为 1/3 偏置，1/4 分时
- LCD 时钟频率为 32.768 KHz
- LCD 电压推进频率为 32.768 KHz
- LCD 帧频率为 128 Hz (在 4 个时隙的 32 Hz 完全重新刷新)
- LCD 控制器的 IIC 通信在 90,900 Hz 位时钟频率操作

1.3 软件流程图

示例程序包含下列主要章节：

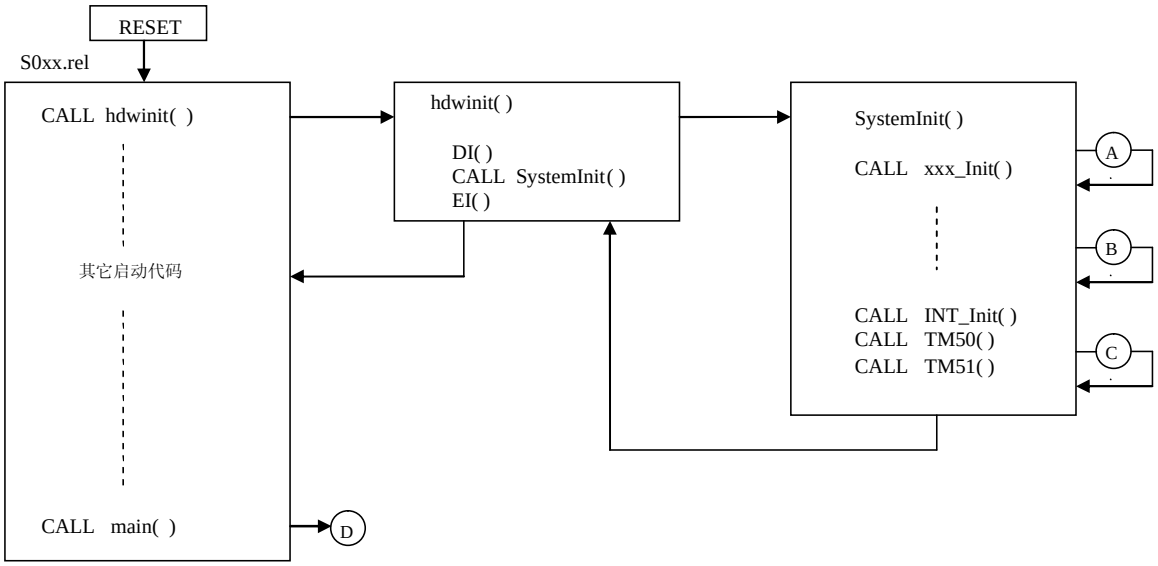
- 程序的初始化代码，在 main()程序之前调用
- IIC0 控制器初始化和 LCD 控制器初始化
- 主程序循环，它检测按键的状态和 LCD 上显示的字符串
- 用于显示数据的 LCD 子程序，它通过向 LCD 段存储器写入数据来实现
- 由 Applilet 产生的用于 IIC0 通信的子程序
- 由 Applilet 产生的用于处理定时器启动，停止和间隔设置的子程序
- 用户代码的子程序，它用于处理按键返回中断(由 Applilet 产生的中断服务程序加上用户代码形成)

这里的流程图将描述初始化，主程序， IIC0 和 LCD 初始化， IIC0 通信，LCD 数据发送和与定时器 00 相关的子程序以及中断服务程序。

1.3.1 程序启动和初始化

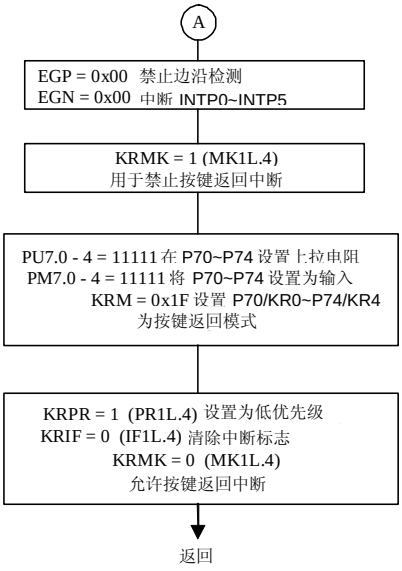
如果使用 C 语言编写 78K0 的程序，C 程序的启动代码由目标代码文件(例如 s01.rel)提供，它会链接到用户程序。这个启动代码被称为 hdwinit( )；用户可以在这里放置任何硬件的初始化代码。

当使用 Applilet 为 78K0 产生 C 语言代码时，hdwinit( )函数自动添加到用户程序，并且调用 SystemInit( )函数。SystemInit( )函数轮流调用每一个硬件的初始化程序。



在 hdwinit( )函数执行完成之后，启动程序代码调用用户程序的 main( )函数。所以在 main( )函数的起始，应该完成按键返回中断和定时器的硬件初始化 。main( )函数不需要调用单独的硬件初始化程序。然后进行 LCD 控制器的初始化和 IIC 通信通道的初始化。

1.3.2 INT_Init( ) – 按键返回中断初始化



由 SystemInit()程序调用的 INT_Init()程序用于设置按键返回功能。我们使用引脚 P70/KR0 到 P74/KR4 作为从负极按键的按键返回输入。由于 Applilet 未指定其它的外部中断，EGP 和 EGN 寄存器被设置未禁止其它外部中断。

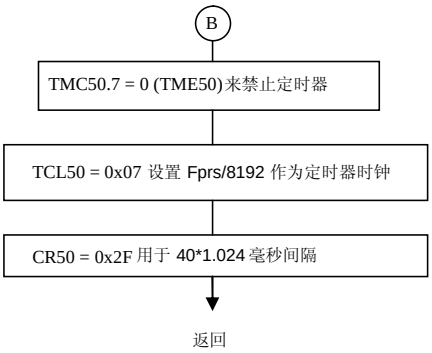
当修改其它寄存器时，按键返回中断屏蔽位 KRMK 被设置为 1 来屏蔽该中断。

为了给 P70-P74 端口引脚提供上拉电阻，PU7 寄存器被设置为 0x1F；为了设置这些引脚为输入状态，端口模式寄存器 PM7 要写入相同的值。按键返回寄存器 KRM 被设置为 0x1F 用于指定按键返回中断源 KR0 到 KR4 会产生按键返回中断。在这种设置下，P70 到 P74 中的任何引脚的负极电平都会产生中断。

按键返回中断的优先级被设置为低，中断标志(KRIF)被清除，清除屏蔽位从而允许按键返回中断 INTKR。

参考按键返回服务程序 MD_INTKR( )以便了解按键返回中断后的程序进成。

1.3.3 TM50_Init( ) – 定时器 50 初始化

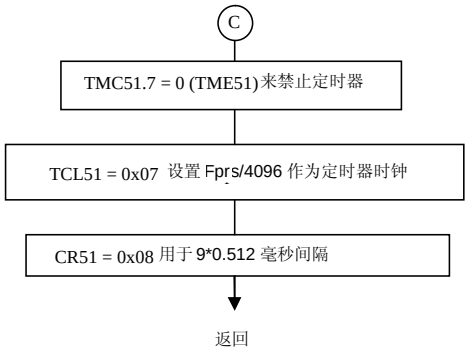


TM50_Init()程序同样被 SystemInit()程序调用，将定时器 50 初始化为 50 毫秒的间隔定时器。当改变寄存器设置时，首先要将定时器 50 的允许位(TME50)清零以禁止定时器运行。
设置 TCL50 寄存器为 0x07 以便将定时器 50 的时钟设置为 f_{PRS}/8192。示例程序中的 78K0/LG2 是在外部硬件时钟 f_{PRS} 下运行，该时钟来自 8 MHz 的内部高速振荡器。定时器时钟为 8,000,000/8192 或 976.5 Hz；每个时钟计数 1.024 毫秒。

将 CR50 寄存器设置为 0x2F (47)；此时 TM50 每 48 个计数与 CR50 比较一次，时间间隔为 48*1.024 = 49.15 毫秒，近似等于我们期望的 50 毫秒。

TM50 用于等待(50ms)程序中的 50 毫秒时间间隔。

1.3.4 TM51_Init( ) – 定时器 51 初始化



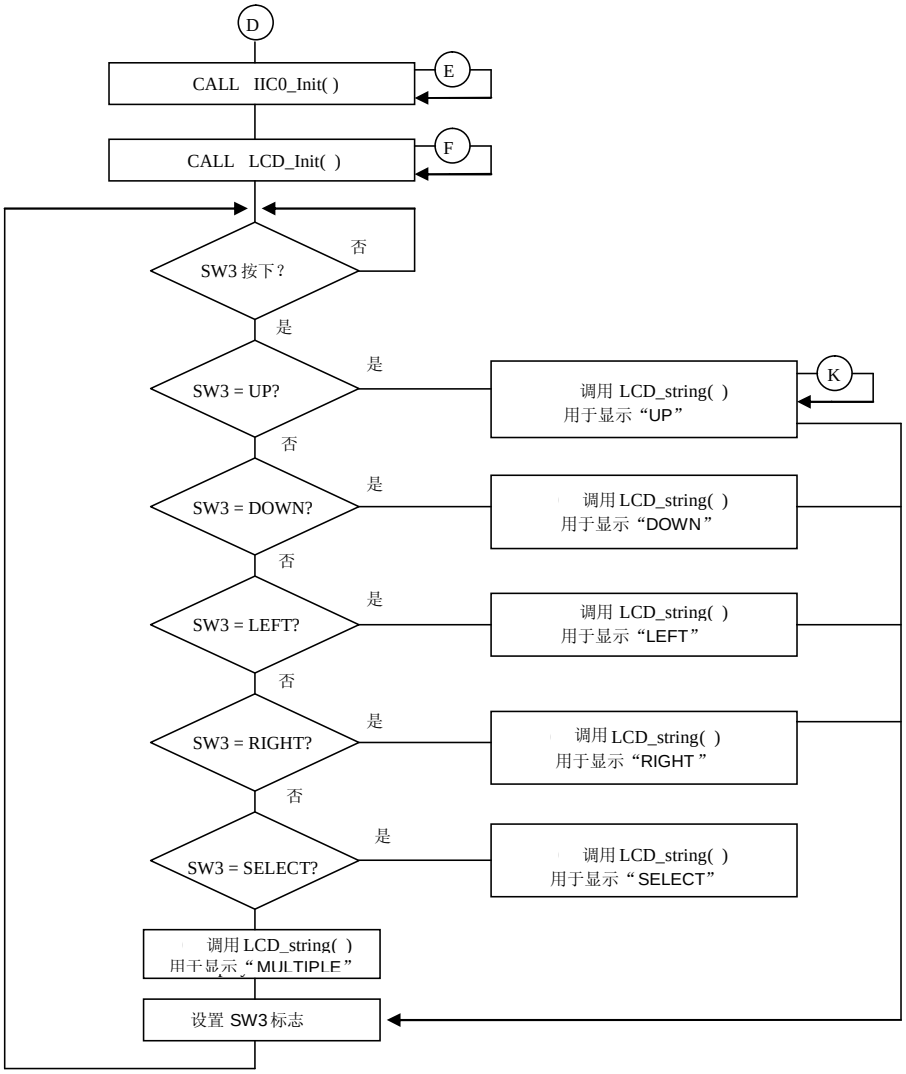
TM51_Init()程序同样被 SystemInit()程序调用，将定时器 51 初始化为 5 毫秒的间隔定时器。当改变寄存器设置时，首先要将定时器 51 的允许位(TME51)清零以禁止定时器运行。

设置 TCL51 寄存器为 0x07 以便将定时器 51 的时钟设置为 f_{PRS}/4096。注意定时器 50 和定时器 51 的时钟是不同的，虽然它们 TCL5x 寄存器的值是相同的。在 8 MHz 外部时钟下，定时器时钟为 8,000,000/4096 或 1953.125 Hz；每一个时钟为 512 微秒，或 0.512 毫秒。

CR51 比较寄存器设置为 8；此时，TM51 每 9 个计数与 CR51 比较一次，时间间隔为 9*0.512 = 4.608 毫秒，近似等于我们期望的 5 毫秒。

TM51 用于按键返回中断服务程序以便在初始中断后 5 毫秒检测按键数据。

1.3.5 Main( ) –主程序– LCD 显示

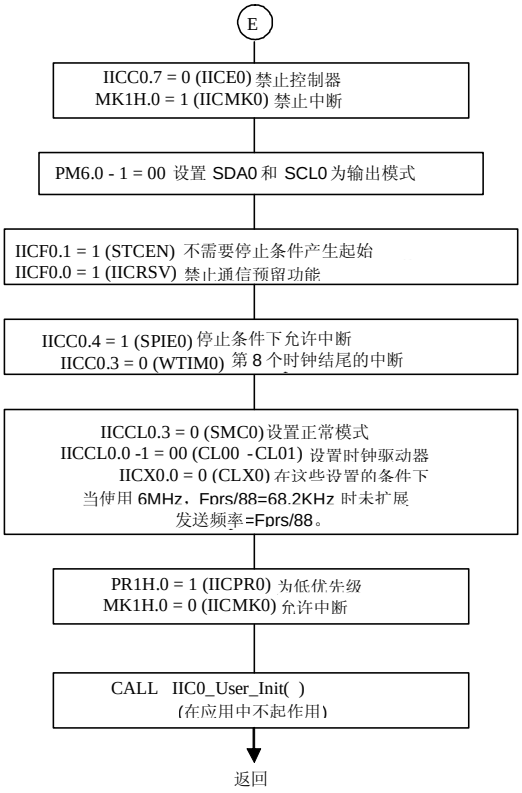


main()函数用于启动程序。Main()调用 IIC0_Init()来初始化 IIC 控制器，它用于与 LCD 控制器通信；然后再调用 LCD_Init()用于初始化 LCD 控制器。

在初始化之后，程序检测 SW3 是否已经被按下。在按键返回中断服务程序中设置变量 sw3_in。如果没有按键按下或未出现抖动，它的值为 0；如果有一个或多个按键按下，它的值为 1。根据负极按键的种类，主程序调用 LCD_string()函数在 LCD 上显示字符串。

一旦按键被释放，变量 sw3_in 将会再次被清 0；当产生下一个按键返回中断时，它才会被再次设置。

1.3.6 IIC0_Init( ) – IIC 控制器初始化



IIC0_Init()函数初始化 IIC0 外部硬件用于准备 IIC 通信。首先禁止 IIC0 控制器，并且屏蔽 IIC 中断 。

IIC0 控制器使用两个引脚，SCL0 (IIC 串行时钟)和 SDA0 (IIC 串行数据)以便和 LCD 控制器以及其它 IIC 设备通信。为了进行 IIC 操作，需要将端口模式寄存器 PM6 的位 0 和位 1 清 0，以便将引脚设置为输出模式。

将 IICF0 (IIC0 标志寄存器)位 STCEN 和 IICRSV 置 1。STCEN=1 允许控制器在不需要停止条件出现的情况下产生起始条件。IICRSV=1 禁止通信保留功能。

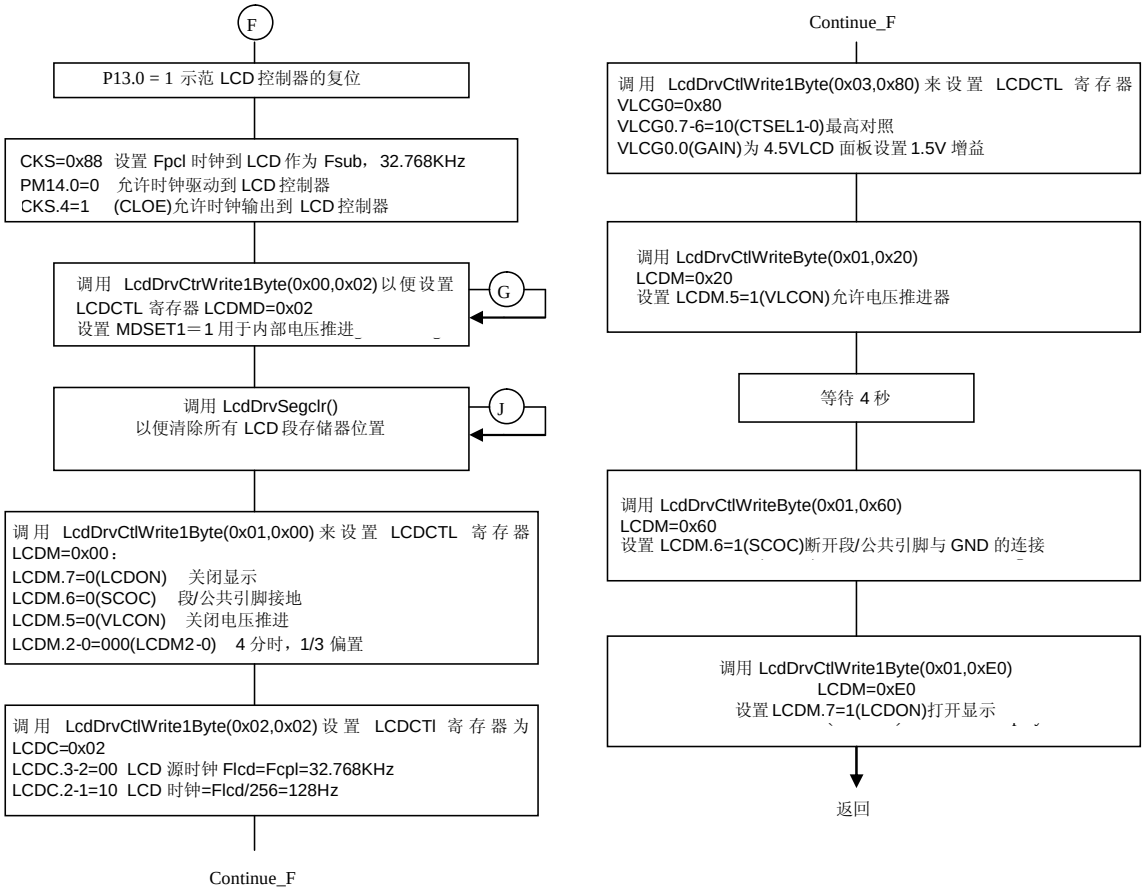
将 IICC0 (IIC0 控制寄存器)位 SPIE0 置 1 以便在停止条件下产生 INTIIC0 中断。将 WTIM0 位清 0，这将导致在数据的第 8 位时产生 IIC0 中断；通常情况下，这是从设备模式。初始化之后，IIC0 控制器作为从设备模式进行操作，当启动主设备模式功能时进入主设备模式；此时，WTIM0 位将会被置 1 用来在第 9 位允许中断，这是主设备模式的标准。

IICCL0 (IIC0 时钟寄存器)和 ICCX0 (IIC0 扩展寄存器)设置指定的 IIC 通信速度。这些设置为正常模式，时钟设置为 f_{PRS}/88；提供给外部设备 8 MHz 时钟，这些导致了 IIC 的时钟为 8,000,000/88 = 90.9 KHz，小于正常操作设备标准 IIC 最大 100 KHz 的频率。

IIC0 中断的优先级被设置为低，中断屏蔽设置为允许 IIC0 中断 INTIIC0。此时，IIC0 外部硬件准备操作，但是还未被允许。这是因为 IICC0 的位 7(IICE0)还未置 1。程序将在后面完成这一设置以便开始通信。

最后，调用 IIC0_User_Init( )函数执行用户指定的初始化。在本示例程序中，该程序为空并返回。

1.3.7 LCD_Init( ) – LCD 控制器初始化



LCD_Init()程序初始化 LCD 控制器，首先通过设置 uPD78F0397 的寄存器来控制允许和 LCD 控制器的时钟，然后通过使用 IIC0 外部硬件与 LCD 控制器通信，写入控制和段寄存器。

P13.0 控制 LCD 控制器的复位信号；复位释放后该位被置 1。然后设置 CKS 寄存器指定 LCD 的时钟为 32.768 KHz 的副时钟。PM14.0 控制 LCD 时钟驱动输出，并且设置允许 LCD 时钟输出到 LCD 控制器。CKS 寄存器位 CLOE 被设置为允许时钟输出到 LCD 控制器。

此时，控制器从复位状态中释放并且接收时钟，但是显示依然被禁止，这是因为 LCD 控制器寄存器处于处于缺省复位值。

LcdDrvCtlWrite1Byte()函数用于向 LCD 控制器 LCDCTL 寄存器写入值。首先 LCD 控制器模式寄存器 LCDMD 被设置为 0x02；这些设置控制器使用内部电压推进方式。我们指定的 LCD 面板工作在 4.5V，所以内部电压推进用于提高 LCD 偏置电压到该电平。这个值同样设置 LCDON 到 0，关闭显示；设置 SCOC 到 0，连接公共信号线和段信号线到 GND 来释放显示；并且设置 VLCON 为 0，关闭电压推进器。

LcdDrvSegClr()功能用于在 LCD 控制器段存储器中的所有段写入 0，以便在初始化后显示空白。

LCD 控制器显示模式寄存器 LCDM 被设置为 0x00，该设置禁止显示，并且将显示设置为 4 分时，1/3 偏置的模式。

LCD 控制器时钟寄存器 LCDC 设置为 0x02，以便设置 LCD 源时钟(用于电压推进器)使用与 LCD 控制器的 32.768 KHz 相同频率的时钟。该时钟至少应为 32 KHz。LCDC 的低两位设置 LCD 的帧频率为 fLCD/256 或 128 Hz。帧频率应为 128 Hz 或更低。在 4 分时条件下，意味着数字频率应为 128/4 = 32 Hz。

LCD 电压推进器控制寄存器 VLCG0 设置为 0x80 来设置最大对比，并且设置电压推进增益为 1.5(这将导致 4.5V 输出)。

然后将 LCDM 设置为 0x20，打开 VLCON 位，这可以允许电压推进器。

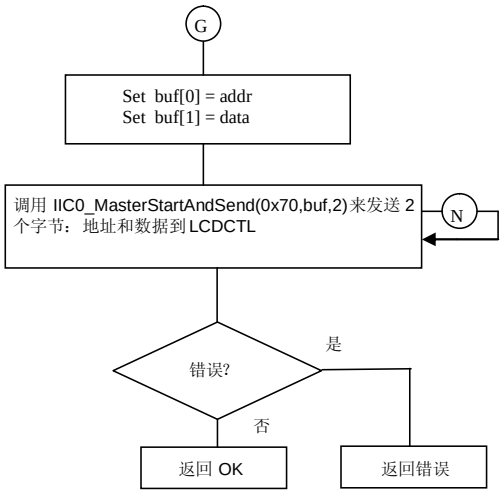
此时，执行等待用来允许电压推进器产生需要的 LCD 电压。如果 78F0397 的 LVDD 引脚处于 1.8V 到 4.5V 时，等待的时间应该为 0.5 秒；如果电压在 4.5 到 5.5V，等待的时间应该为 4 秒。由于我们指定了 LVDD 的电压为 5.0V，所以等待 4 秒。

等待通过调用 Wait(数字)子程序来完成，该子程序按照输入的参数乘 50 毫秒作为等待的时间。Wait(80)等待 80 * 50 = 4000 毫秒，或者 4 秒。

然后 LCDM 设置为 0x60，这将导致 SCOC 位为 1 来断开 LCD 公共信号线和段信号线到 GND 的连接。最后，LCDM 设置为 0xE0，这将导致 LCDON 位为 1 以便开始显示。

此时， LCD 控制器初始化完成并且显示空白数据。通过写入 LCDSEG 寄存器来改变 LCD 段显示存储器将导致指定的段被点亮。不需要向 LCDCTL 控制寄存器进行更进一步的写入操作。

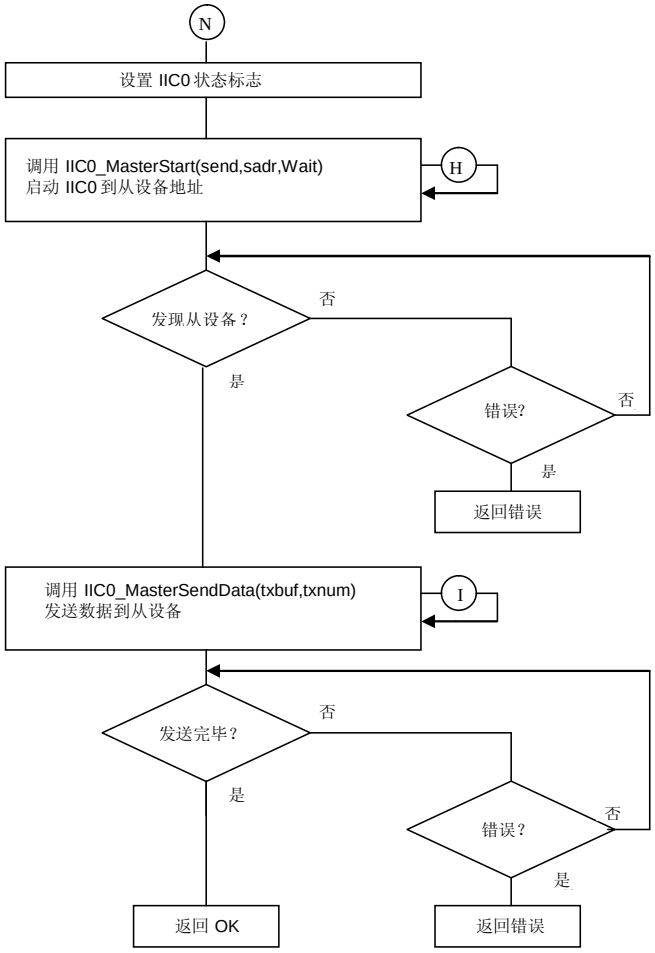
1.3.8 LcdDrvCtlWrite1Byte(addr, data) – 向 LCDCTL 寄存器写入一个字节的数据



LcdDrvCtrWrite1Byte(addr, data)函数用于向 LCDCTL 寄存器写入一个字节的数据，写入的地址由参数 addr 指定。LCDMD 被选择为 addr = 0，LCDM 为 addr = 1，LCDC 为 addr = 2 和 VLCG0 为 addr = 3。

该程序设置两个字节的数组 buf []，将寄存器的地址写入 buf[0]且数据写入 buf[1]。然后调用函数 IIC0_MasterStartAndSend(CSLV_ID_LCDCTL, &buf[0], 2)。然后将缓存中数据的两个字节发送到从地址 0x70，该地址为 LCDCTL 寄存器的地址。第一个发送的字节为控制寄存器的地址，第二个字节为向该寄存器写入的值。

1.3.9 IIC0_MasterStartAndSend(sadr, *txbuf, txnum) – 启动 IIC0，从缓存发送数据



IIC0_MasterStartAndSend(sadr, *txbuf, txnum)程序是 IIC0_MasterStart() 和 IIC0_MasterSendData()的联合函数。该函数不是通过 Applilet 产生的,但是却是一个主设备发送 IIC 数据所需的两个程序的简单联合。

程序首先设置 IIC 通信标志为缺省设置；在 IIC 通信处理中，这些标志会收到 IIC0 回调函数的影响。tUI_MasterError 被设置为 MD_OK (表示没有产生错误)，UI_MasterSendEnd 被设置为 MD_ERROR 来指示发送未完成；UI_MasterFindSlave 被设置为 MD_ERROR 来表示在该点上未找到从设备。

然后该程序调用 IIC0_MasterStart(Send, sadr, 10)函数来启动主设备的 IIC 通信以便发送数据，并且通过“sadr”指定从设备地址。返回时，任何一个 IIC 外部设备已经被允许，并且从设备地址已经被写入 IIC 寄存器用于发送，或者 IIC通信信道繁忙，这种情况在示例程序中不会发生。

然后程序等待 UI_MasterFindSlave 为 OK，或是 UI_MasterError 设置为错误条件。

IIC0 外部硬件在第 9 个时钟位时获得中断；该中断通过 MD_INTIIC0 中断服务程序来处理。如果 LCD 控制器 的地址为两个从设备地址之一 (LCDCTL 为 0x70，LCDSEG 为 0x72)，它将回应 ACK。然后设置标志 UI_MasterFindSlave。如果没有 ACK，中断服务程序将会设置 UI_MasterError 标志到 MD_NACK。

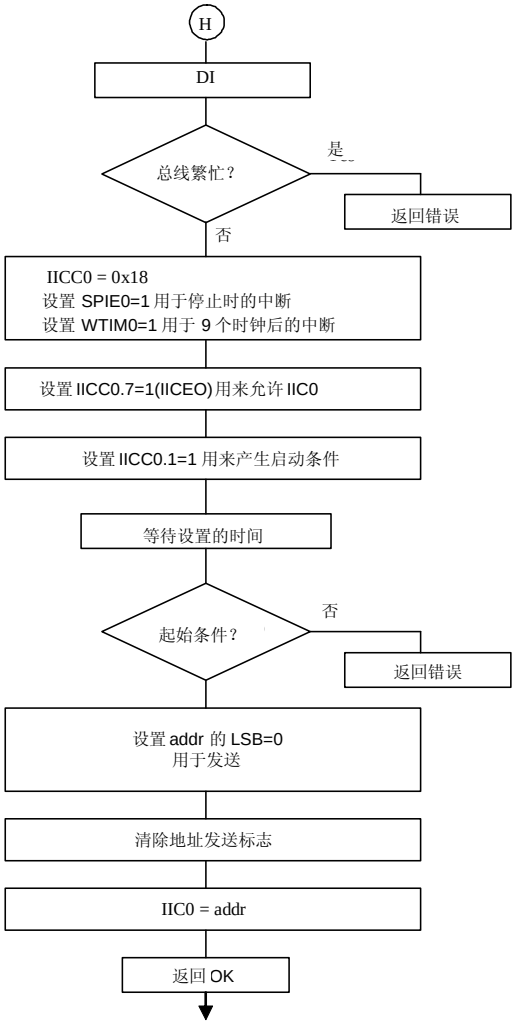
如果未找到从设备，IIC0_MasterStartAndSend() 返回错误代码。如果找到从设备，程序调用 IIC0_MasterSendData(txbuf, txnum)来将要发送的数据排成队列。

IIC0_MasterSendData()将写入和保存缓存地址和计数，并且将数据的第一个字节写 IIC0 寄存器，然后返回。

在这一点， 当所有数据都被发送之后，将会产生中断 INTIIC0。MD_INTIIC0 中断服务程序将会处理该中断 ，通过发送下一个字节的数据，或者在发送和接收所有字节时设置 UI_MasterSendDone，或者如果未从从设备返回 ACK 时设置错误。

IIC0_MasterStartAndSend()程序等待 UI_MasterSendDone 条件为真，或者 UI_MasterError 被设置为错误条件。然后返回调用者的操作状态。

1.3.10 IIC0_MasterStart(Send, addr, wait) – 启动 IIC0 发送到从设备地址



IIC0_MasterStart(Send, addr, wait)函数用于处理启动主设备模式 IIC 通信。

该程序检测 IIC 总线是否繁忙，如果是则返回错误。IICC0 寄存器位 SPIE0 和 WTIM0 被设置为允许在停止条件下的中断，并且在 9 个时钟后产生中断(主设备模式)。然后 IICC0 位 7(IICE0)被设置为 1 以允许 IIC0 外部硬件。

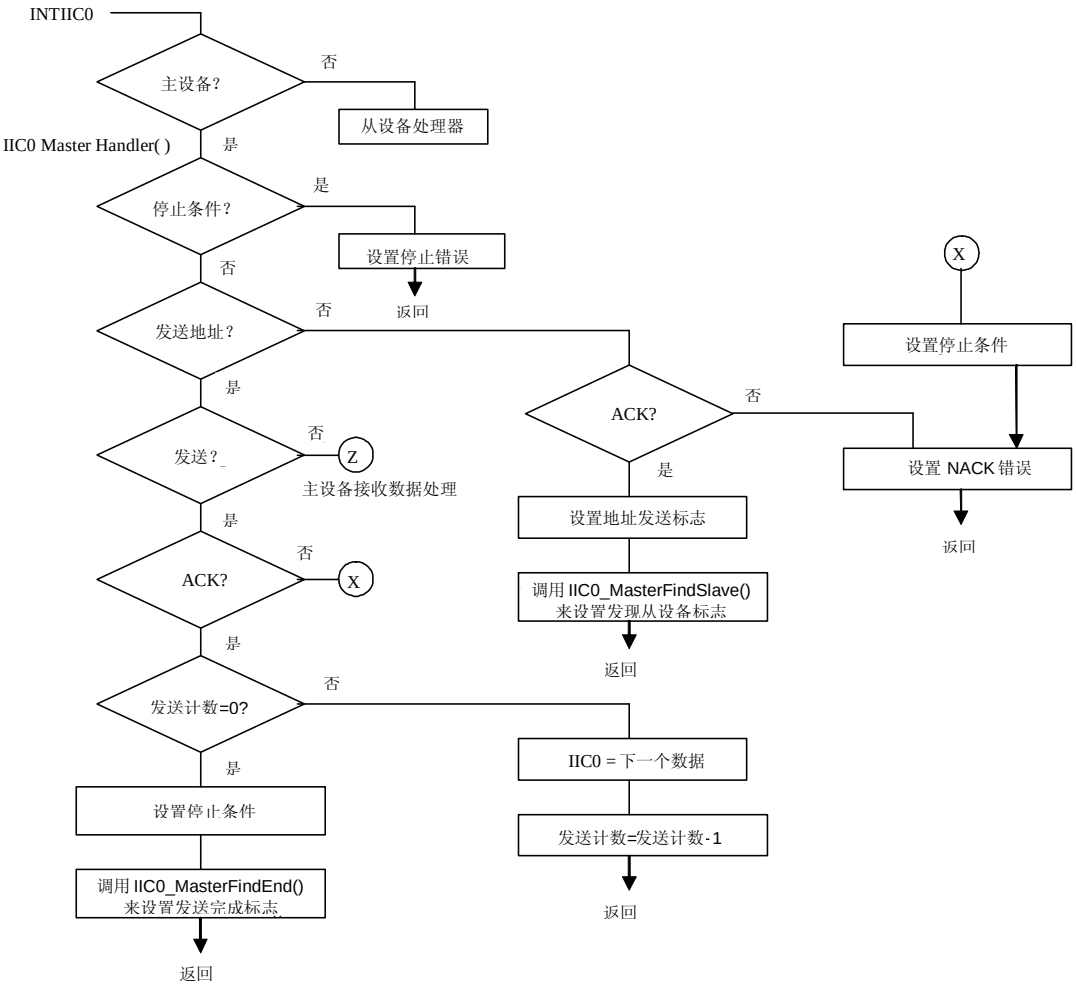
IICC0.1(STT0)被设置为 1 以便产生起始条件，并且执行等待允许记录起始条件。IICS0.1(STD0)指示起始条件的存在。如果没有检测到起始条件，则返回错误。

从设备地址的 LSB 被设置为 0，这指示了从主设备到从设备的数据发送，并且从设备地址被写入 IIC0 寄存器开始 IIC 发送。“address sent”标志被清除，它用于 MD_INTIIC0 中断服务程序以确定中段所需要的处理。

程序返回时，已经开始了和从设备的通信。

注意 IIC0_MasterStart()函数同样可以用于启动接收操作，进行从设备到主设备的数据发送。如果要进行这样的操作，只需将第一个参数从“发送”变为“接收”。

1.3.11 MD_INTIIC0( ) and IIC0_MasterHandler( ) - IIC0 中断服务程序



MD_INTIIC0()中断服务程序和它调用的函数，负责 IIC 发送的大部分工作。中断服务程序确定一个主设备或从设备的操作正在被处理，并且调用相应的处理程序，IIC0_MasterHandler()或是 IIC0_SlaveHandler()。与 LCD 控制器的通信，我们使用主设备通信，所以上面的流程图显示了 IIC0_MasterHandler()的流程图。

主设备发送的第一个中断将会在主设备发送从设备地址后，第九个时钟设置。

处理器首先检测是否检测到停止条件。这不会出现在通常的主设备模式通信，所以 UI_MasterError()被调用来指示一个停止错误。

然后处理器检测从设备的“地址发送”标志是否已经被设置。如果未被设置，该中断在发送从设备地址后产生。该程序检测是否接收到从设备的 ACK；如果没有，会设置 NACK 错误条件。如果接收到 ACK，从设备完全的回应地址；设置 UI_MasterFindSlave 标志，“发送的地址”标志为真，然后处理器返回。

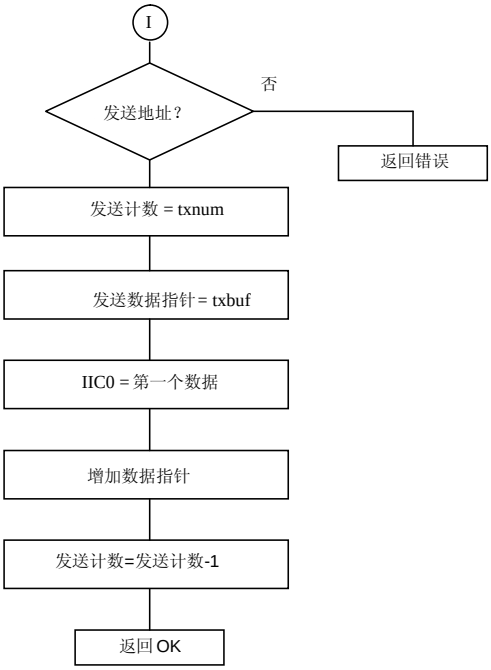
第一个中断之后，后面的中断将在数据字节被发送或是接收操作开始之后产生。处理器检测通信的方向是发送(主设备到从设备)或是接收(从设备到主设备)。由于这个应用笔记处理从主设备发送数据到从设备 LCD 控制器，接收数据的流程图未列出。

如果发送，处理器期望在发送每一个字节之后都会收到 ACK。如果未检测到 ACK ，会设置 NACK 错误条件，并且处理器返回。

如果接收到 ACK，处理器检测是否还有要被发送的数据字节。如果有，它会向 IIC0 写入下一个字节来启动发送，更新技术和缓存指针，然后返回。

如果没有要发送的字节，发送工作完成。处理器设置停止条件表示到从设备的发送工作完成，设置 UI_MasterSendDone 标志，然后返回。

1.3.12 IIC0_MasterSendData(*txbuf, txnum) –为 IIC0 发送排列数据

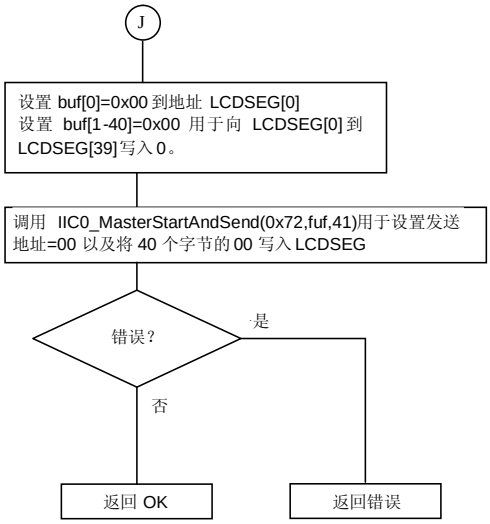


IIC0_MasterSendData(*txbuf, txnum)函数在 IIC0_MasterStart()执行完成后被调用，设置 UI_MasterFindSlave 标志来指示从设备存在并且准备接收数据。

程序检测“被发送的地址”标志是否被设置，并且在没有设置的情况下返回错误。如果该程序在产生从设备地址发送完成的 INTIIC0 中断之前被调用将会导致上面的结果。

然后程序通过 txbuf 设置要发送数据的缓存地址，设置要发送的字节技术到 txnum，将第一个字节写 IIC0 寄存器并且增加指针和减少计数，所以第二个以及以后的字节会被 MD_INTIIC0 中断服务程序发送。

1.3.13 LcdDrvSegClr( ) – 清楚所有的 LCD 段数据

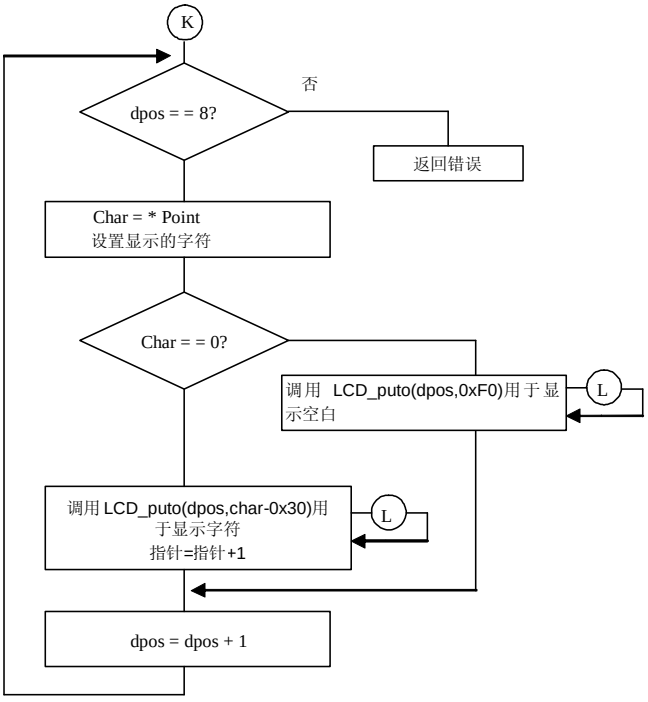


LcdDrvSegClr()程序清除所有定位于 LCD 控制器的 LCDSEG 存储器为 0，这将导致 LCD 显示编程空白。

该函数设置 41 个字节的缓存。Buf[0]包含 0，作为第一个 LCDSEG 定位的地址；buf[1]到 buf[40]同样被设置为 0，以指定被写入 LCDSEG 的数据定位于 0 到 39。

然后程序调用 IIC0_MasterStartAndSend(CSLV_ID_LCDSEG, &buf[0], 41)来发送 41 字节的数据到从设备地址 CSLV_ID_LCDSEG，实际地址为 0x72。第一个字节的发送(buf[0] == 0x00)通过 LCD 控制器执行作为 LCDSEG 地址开始写入数据；后面的 40 个字节的 0x00 将写入 LCDSEG 地址 0 到 39，在这些地址上设置段位。

1.3.14 LCD_string(*point, dpos) -在指定的数字位置向 LCD 写入字符串



LCD_string()函数用于写入一个字符串(字符字节的数组，以 0 结尾)以便在指定的位置进行 LCD 显示。Dpos = 0 意味着写入的字符串从左侧开始显示；dpos = 1 将脱离最左侧字符，并且在下一个字符的位置写入字符串。

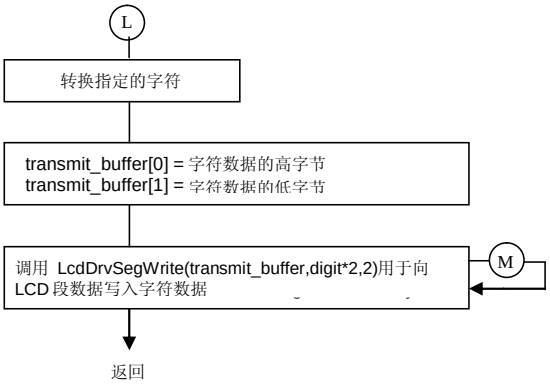
在程序循环的过程中，增加显示的位置直到 dpos = 8。这说明 LCD 面板支持 8 个字符，最左侧的位置是 dpos=0，最右侧的的位置是 dpos=7。

对于每一个 dpos 位置，显示的字符从*point array 中读取。如果字符为 0，代表字符串的结束，后面的位置都为空。

如果字符不为 0，调用函数 LCD_putc(dpos, char – 0x30)来写入相应字符段数据用于显示。数值 0x30 是 ASCII 字符数经过减法器得到的，改变科印刷的数值从 0x30 (‘0’)到 0x00，或从 0x41 (‘A’)到 0x11，这些将行程字符段数据数组的索引。

在当前的字符显示完成后，更新 dpos，并更新字符指针指向下一个字符(除非检测到 0)，并且继续运行程序直到 dpos 大于或等于 8。

1.3.15 LCD_putc(digit, data) - Write One Character to 在特定的数字位置向 LCD 写入一个字符



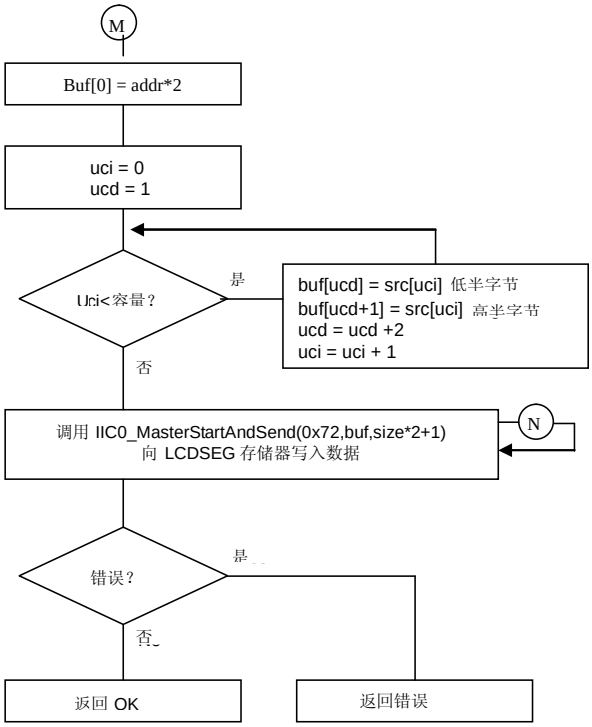
LCD_putc()程序装载一个字符数据 (使用 – 0x30 偏置)，并且在选择的位置进行显示。

检测字符数据值和字符段数据数组的索引是否正确。每一个字符需要 16 位数据，写入 4 个连续的 4 位 LCDSEG 位置。字符段数据数组是一个 16 位位置的数组，每个字符包含 16 位。

数据索引用于字符段数据数组，16 位数据写入 2 个字节数组 transmit_buffer[]。然后程序会调用函数 LcdDrvSegWrite(&transmit_buffer[0], digit * 2, 2)将两个字节写入 4 个由 digit * 2 指定的连续位置地址。

LCD_putc(digit, data)程序的输入为数字 0 到 7，指定字符位置；LcdDrvSegWrite()装载字节地址 0 到 14，如果以字节编址则指定段存储器的起始位置。这将通过在 LcdDrvSegWrite()中乘以 2 来获得进一步的修改 LCDSEG 的位置。

1.3.16 LcdDrvSegWrite(*src, addr, size) -在 LCD 段存储器中写入字符数据的字节



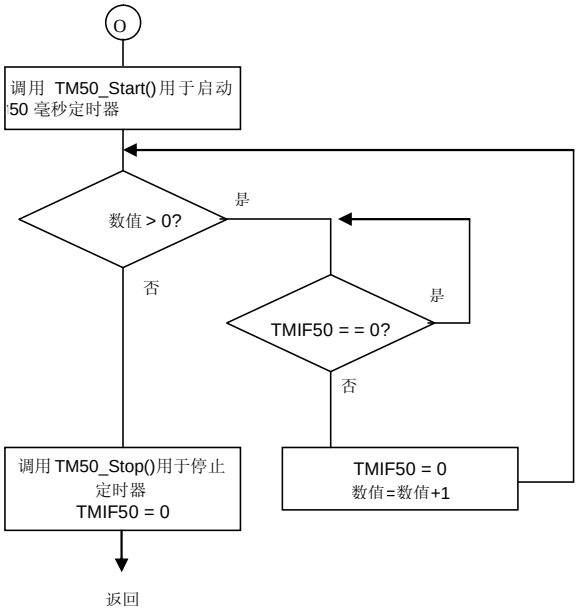
LcdDrvSegWrite()程序写入数据字节到 LCDSEG 存储器， *src 参数指定一个到字节数组的指针；addr 参数指定了 写入数据的 LCDSEG 存储器的字节地址；size 指定了要写入的字节数。

程序使用 IIC0_MasterStartAndSend()函数设置写入数据的缓存。数组中的第一个字节 buf[0]，包含了 LCDSEG 存储器中地址的地址字节。由于每一个字节的地址对应两个 LCDSEG4 位位置，字节地址乘以 2 来获得 LCDSEG 位置。

然后*src 数组中的字节数据将会转换成 buf[]数组中的 4 位值用于写入 LCDSEG 位置。每个字节中的低 4 位写入低 buf[]位置，每个字节中的高 4 位写入下一个位置。

在转换完成后，buf[]数组包含 LCDSEG 存储器的地址和数据。IIC0_MasterStartAndSend(CSLV_ID_LCDSEG, &buf[0], size * 2 + 1)被调用来发送数据到地址为 CLSV_ID_LCDSEG 的 IIC 从设备，它的真实地址为 0x72。 第一个字节(addr * 2)提供 LCDSEG 位置，数组中的其余段数据写入接下来的地址位置。

1.3.17 Wait(number) –等待指定的 50 毫秒间隔的个数



Wait(number)函数被 LCD_Init()调用来等待指定的 50 毫秒间隔的个数。

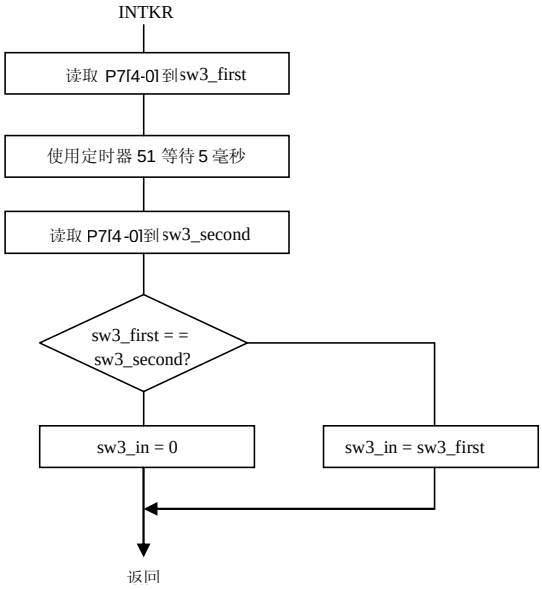
Wait( )调用 TM50_Start( )来启动 TM50 计数值。TM50 已经被 TM50_Init()初始化完毕，所以在大约 50 毫秒后定时器的 CR50 会与 TM50 相等。当 TM50 等于 CR50 时，设置 TMIF50 中断标志位。由于设置了中断屏蔽位(TMMK50)，所以不会产生中断。

Wait( )函数循环检测 TMIF50 标志的状态，直到发现该标志被设置。此时，正好经过了大约 50 毫秒。 Wait( )函数然后复位 TMIF50 中断标志。

Wait( )函数等待时间间隔的个数使用输入的参数指定的，在运行过程中不断减少该值直到该值为 0。在循环的结尾，经历了指定数目的 50 毫秒间隔。

Wait( )停止定时器，并且确保中断标志复位，以便下次调用时使用。

1.3.18 MD_INTKR( ) – 按键返回中断服务程序



MD_INTKR()程序是 INTKR 的中断服务程序。当负极按键 SW3 上的任意开关按下时，按键返回输入引脚连接到 GND。此时按键返回引脚的下降沿触发按键中断 INTKR。

MD_INTKR()程序读取相应按键的输入端口来获得当前引脚的状态。 定时器 TM51 用于 5 毫秒等待；当 5 毫秒到时，再一次读取端口状态。

如果两次的按键值相同，表明 5 毫秒间隔内的按键状态相同，即的确有按键按下。在这种状态下，全局变量 sw3_in 的值设置为代表被按下的按键的数值，然后程序返回。

如果两次的按键值不相同， sw3_in 被设置为 0 来表示未发现稳定的按键按下，然后程序返回。

如果按键继续振荡，按键返回输入的下一个下降沿会再次导致 INTKR 中断，并且再次检测按键的状态。

1.4 Applilet 的参考驱动

NEC 的 Applilet 程序生成器可以自动产生 C 或是汇编语言源代码来管理 NEC 微控制器设备的外部硬件。使用的 Applilet 的版本清参考附录。

Applilet 用于产生程序的主函数和基本的初始化代码，按键返回中断、8 位定时器 TM50 和 TM51 和用于与 LCD 控制器进行通信的 IIC0 外部设备的驱动代码。在 Applilet 生成基本代码之后，用户可以添加程序的特定功能。

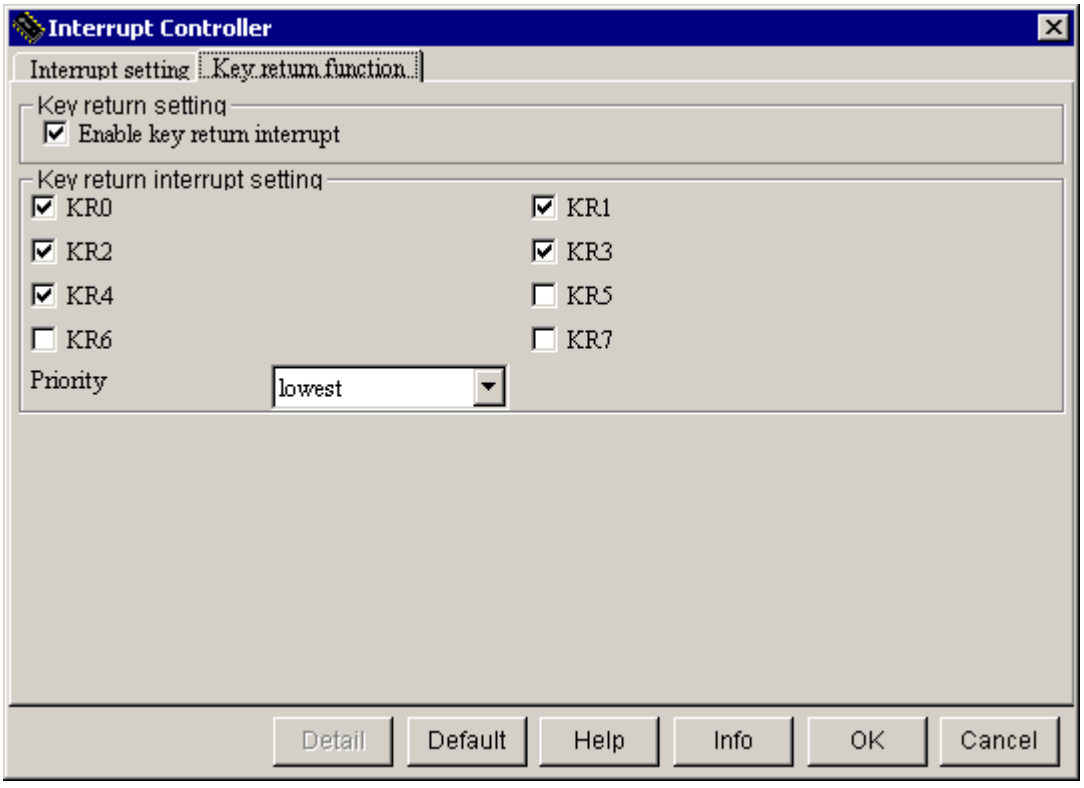
为了支持 LCD 控制器，NEC 提供示例程序代码，它适用于本示范程序。 该代码调用 Applilet 产生的 IIC0 程序执行与 LCD 控制器的通信。

本节描述 Applilet 如何设置生成按键返回中断、定时器 TM50 和 TM51 以及 IIC0 外部硬件的代码，并且列出生成的程序。添加的文件也会被列出。

启动 Applilet 会创建一个新的工程文件，该文件被保存为.prx 文件。Applilet 显示一个用户界面以允许选择和设置不同的外部硬件模块。

1.4.1 配置 Applilet 用于按键返回中断

选择 INT 模块下的按键返回功能标签即可设置按键返回中断的详细信息。

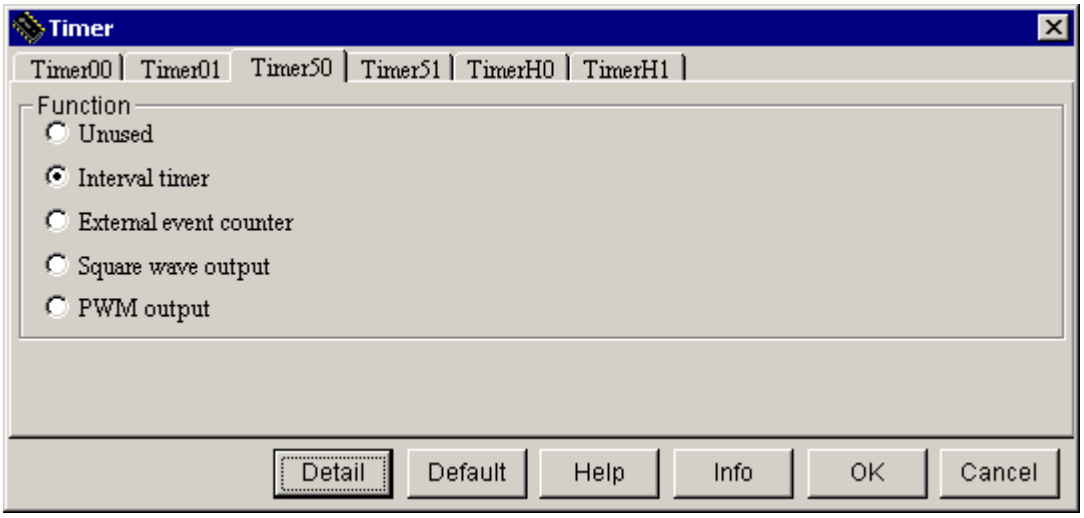


检验框允许检测按键返回中断，以便在下面的按键返回引脚产生中断。该中断的优先级被设置为低。

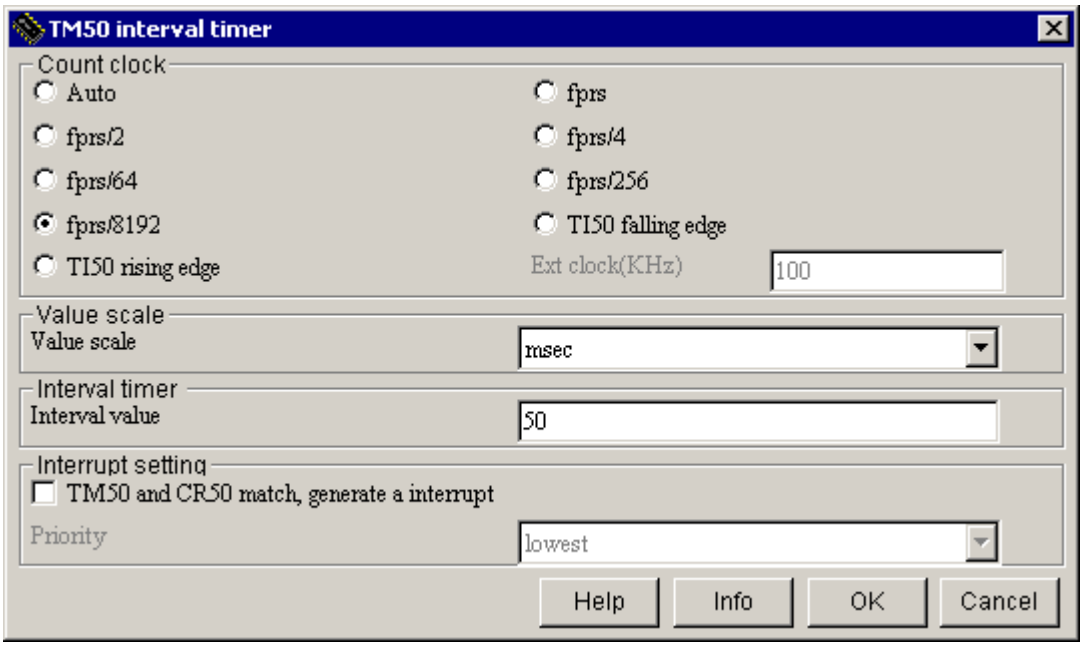
负极按键连接在引脚 P70/KR0 到 P74/KR4，所以我们为 KR0 到 KR4 选择检验框。KR5，KR6 和 KR7 未使用；相应的引脚 P75/KR5，P76/KR6 和 P77/KR7 可以用于通用目的 I/O 引脚而不影响按键返回中断功能。

1.4.2 配置 Applilet 用于定时器 TM50

选择“Timer”模块后会屏幕会出现显示各种定时器的模块。选择定时器 50 标签后设置 TM50 用于间隔定时器。



一旦选择使用 TM50，在选择的模式下，可以使用设置 TM50 的“Detail”按钮。点击“Detail”按钮之后会出现 TM50 作为间隔定时器的详细对话。

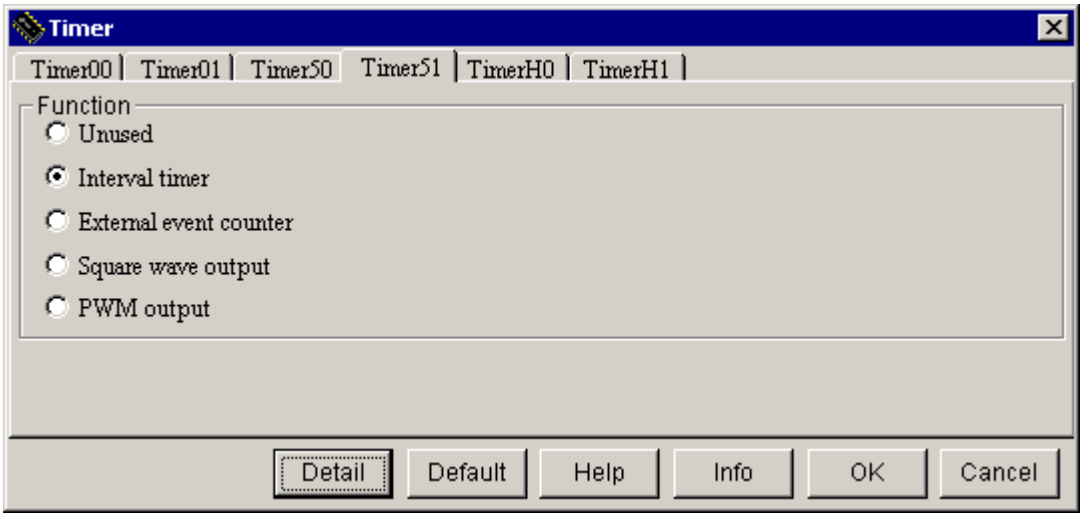


在这里可以选择 TM50 操作的详情。我们希望该定时器每 50 毫秒产生一次中断，所以数值精度选择为“msec”(毫秒)，并且间隔数值设置为 50。计数时钟设置为 fprs/8192，用于 TM50 的基本时钟。Applilet 将会计算相应的设置提供给定时器的比较寄存器，它们会产生 50 毫秒的间隔。

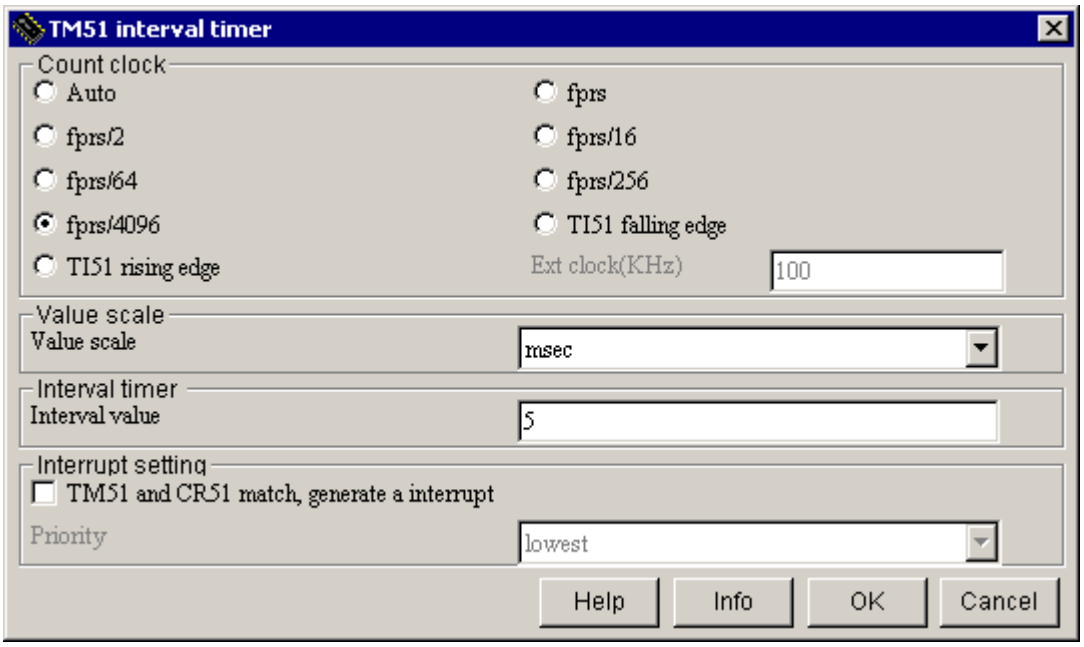
由于我们不想在定时间隔完成后产生中断，当 TM50(定时器计数寄存器)和 CR50(定时器比较寄存器)相等时，我们离开产生中断的检验框。

1.4.3 配置 Applilet 用于 TM51

在定时器模块中，选择定时器 50 标签并且 TM51 用于间隔定时器。



一旦选择了使用 TM51，可以使用设置 TM51 的“Detail”按钮。点击“Detail”后出现 TM51 作为间隔定时器的详细设置。

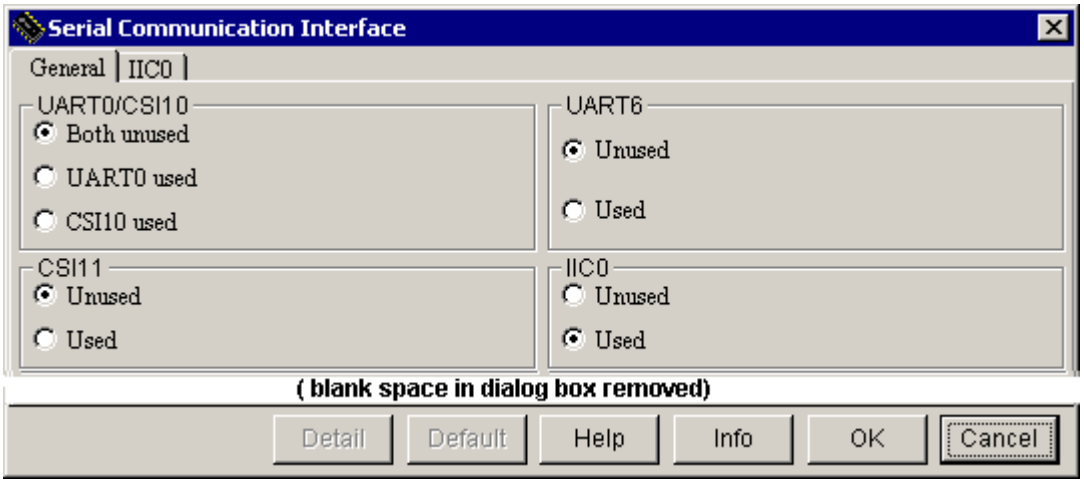


在这里可以选择 TM51 操作的详情。我们希望该定时器每 5 毫秒产生一次中断，所以数值精度选择为“msec”(毫秒)，并且间隔数值设置为 5。计数时钟设置为 fprs/4096，用于 TM51 的基本时钟。Applilet 将会计算相应的设置提供给定时器的比较寄存器，它们会产生 5 毫秒的间隔。

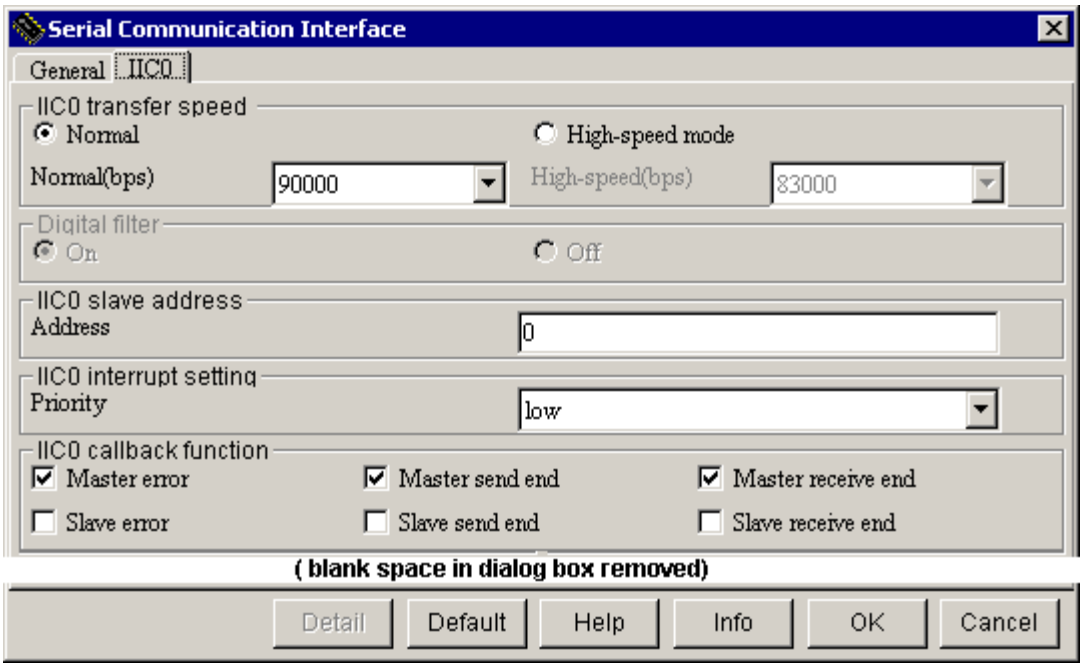
由于我们不想在定时间隔完成后产生中断，当 TM51(定时器计数寄存器)和 CR51(定时器比较寄存器)相等时，我们离开产生中断的检验框。

1.4.4 配置 Applilet 用于 IIC0 通信

在串行模块中，选择使用 IIC0 外部硬件。



一旦选择使用 IIC0，可以使用一个 IIC0 的标签。选择该标签后将会显示 IIC0 外部硬件的详细设置。



IIC0 外部硬件设置为正常模式操作；选择该模式后，使用下拉菜单选择一个通信的速率。这些速率基于外部硬件时钟的分频，所以可用的选择取决于外部硬件时钟。这里我们选择的 90000 bps 为最大的可用速率。

IIC0 外部硬件可以操作在从设备模式或是主设备模式下，并且可以在两种模式之间切换。除非处于一个主设备发起的发送中，否则一般情况下设备都会运行在从设备模式下，并且向从检测到本机为从设备地址外部主设备回应其 IIC 发送。在示例程序中，我们不使用从设备模式；从设备地址为缺省的 0。

IIC0 中断设置为低优先级。

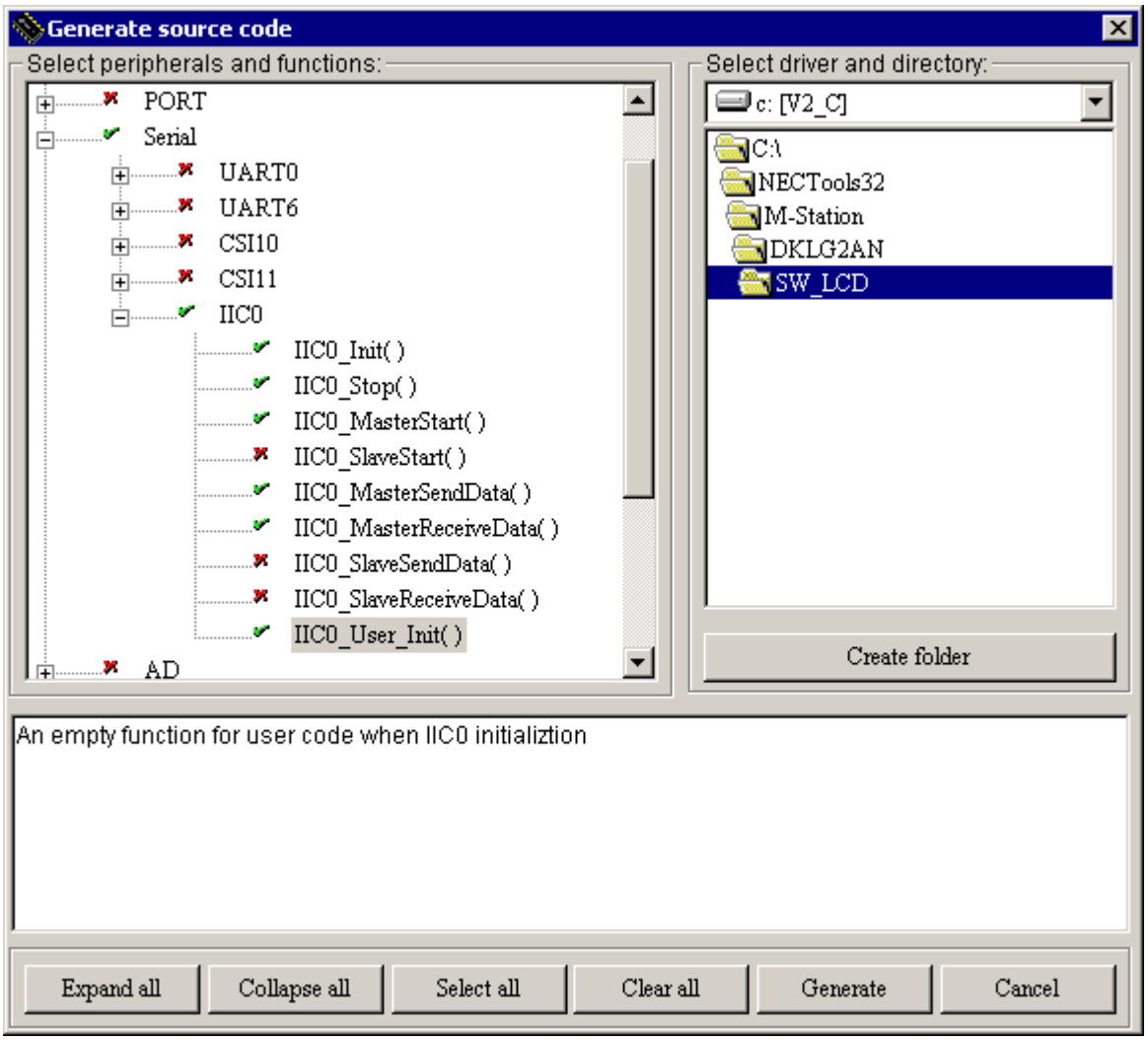
Applilet 会生成一些标准的程序来运行，并且拥有生成空白回调程序的选项，在这些回调程序中，用户可以自己添加代码来处理特定的事件。我们将使用这些回调程序来支持主设备通信，并且用于“主设备错误”的检测框也是如此， 检测“主设备发送完成”和“主设备接收完成”。不检测用于从设备回调的检测框。

对于 IIC0 外部硬件，同样有可以选择的 Applilet 功能，这些功能包含了串行模块的 IIC0 标签中未选择的。这些附加的功能可以在 Applilet 功能视图或是生成代码时选择。

1.4.5 使用 Applilet 生成代码，选择附加的 IIC0 程序

一旦 IIC0 对话框设置，选择 “Generate code”选项。Applilet 将会显示外部硬件和产生的函数，并且允许选择存储源代码的目录。

为了选择附加的 IIC0 程序，左侧方块中的外部硬件/函数树被扩展显示当前选择的用于支持 IIC0 的函数。



对于一个新建的工程，只显示 IIC0_Init()。我们选择附加的函数 IIC0_Stop()，IIC0_MasterStart()，IIC0_MasterSendData()，IIC0_MasterReceiveData()和 IIC0_User_Init()。Applilet 将会生成被选择的程序。

当按下“Generate”按钮时，Applilet 在一些 C 语言源文件(扩展名.c)和头文件中(扩展名.h)中创建代码，并且在对话框中显示了创建文件的列表。

为了支持按键返回中断，Applilet 生成 int.h，int.c 和 int_user.c。

为了支持 TM50 和 TM51，Applilet 生成 timer.h，timer.c 和 timer_user.c。

为了支持 IIC0，Applilet 生成 serial.h，serial.c 和 serial_user.c。

其它生成的文件包括带有空白主函数的 main.c 文件。

1.4.6 Applilet 为按键返回中断产生的文件和函数

为支持按键返回中断产生的代码位于文件 int.h，int.c 和 int_user.c，包含下列的条目。

1.4.6.1 Int.h

头文件 int.h 包含控制按键返回中断的声明和按键返回初始化的定义值。头文件 macrodriver.h，用于所有由 Applilet 生成的代码，同时也定义了一些数据类型和数值，比如返回到一些函数的 MD_STATUS 值。

在这里，我们需要在 Applilet 生成的代码中添加外部声明的 sw3_in 变量，用于报告负极按键 SW3 的状态，所以程序的其它部分可以通过包含 int.h 的方式检验该变量。

1.4.6.2 Int.c

源文件 Int.c 包含了由 Applilet 生成的函数 用于按键返回中断：

void INT_Init(void);
INT_Init()程序初始化在 Applilet 的按键返回对话框中指定的按键返回寄存器和中断。

1.4.6.3 Int_user.c

源文件 int_user.c 包含了用于用户代码的末节函数。这些函数在代码生成时是空的，用于允许用户添加特定应用的代码。

__interrupt void MD_INTKR(void);
这是按键返回中断 INTKR 的中断服务程序，当在任意一个按键返回引脚上出现下降沿时产生。

通过 Applilet 生成，该中断服务程序为空函数。为了使用按键返回中断来去抖动并且读取 SW3 负极按键，必须在该程序中添加用户代码。

同样需要添加变量 sw3_in 的声明。

1.4.7 Applilet 生成的用于 TM50 和 TM51 的文件和函数

用于支持 TM50 和 TM51 的生成代码位于文件 timer.h，timer.c 和 timer_user.c，包含下列条目。

1.4.7.1 Timer.h

头文件 timer.h 包含了控制 TM50 和 TM51 函数的声明，以及定时器初始化的定义值。头文件 macrodriver.h 用于所有 Applilet 生成的代码，同时定义了一些数据类型和数值，比如用于一些函数返回的 MD_STATUS 的值。

1.4.7.2 Timer.c

源文件 Timer.c 包含了下列由 Applilet 产生的用于 TM50 和 TM51 的函数：

void TM50_Init(void);
TM50_Init()程序初始化 TM50 详细对话框中指定的外部硬件。

void TM50_Start(void);
TM50_Start()程序通过允许定时器启动 TM50 操作。

void TM50_Stop(void);
TM50_Stop()程序通过禁止定时器停止 TM50 操作。

MD_STATUS TM50_ChangeTimerCondition(USHORT value);
TM50_ChangeTimerCondition()函数改变 CR50 比较寄存器中的值，因此改变 TM50 的间隔周期。

void TM51_Init(void);
void TM51_Start(void);
void TM51_Stop(void);
MD_STATUS TM51_ChangeTimerCondition(USHORT value);
这些函数在 TM51 上执行与上面函数在 TM50 上相同的功能。

1.4.7.3 Timer_user.c

源文件 timer_user.c 包含末节函数用于添加用户代码。这些函数在生成代码时为空，用户可以添加特定应用的代码。由于我们没有选择任何由 TM50 和 TM51 产生的中断，该文件不包含任何代码。

1.4.8 Applilet 生成的用于 IIC0 通信的文件和函数

为支持 IIC0 产生的代码位于文件 serial.h，serial.c 和 serial_user.c，包含下面的条目。

1.4.8.1 Serial.h

头文件 serial.h 包含控制 IIC0 函数的声明，以及初始化 IIC0 时的定义值。头文件 macrodriver.h 用于所有由 Applilet 生成的代码，同样定义了一些数据类型和数值，比如一些函数的返回值 MD_STATUS。

我们必须对表示 IIC0 状态的变量添加外部声明：

```
extern MD_STATUS UI_MasterError;
extern MD_STATUS UI_MasterSendEnd;
extern MD_STATUS UI_MasterReceiveEnd;
extern MD_STATUS UI_MasterFindSlave;
```

以及 IIC0_Init( )和 IIC0_MasterStartAndSend( )。

1.4.8.2 Serial.c

源文件 Serial.c 包含了下面由 Applilet 生成的用于 IIC0 的函数：

void IIC0_Init(void);
该程序初始化在 IIC0 详细对话框中指定的外部硬件。

MD_STATUS IIC0_MasterStart(TransferMode mode, UCHAR adr, UCHAR wait);

该程序启动主设备模式通信操作 **mode** 参数可以是 “Send” 或是“Receive”以便指定方向； **adr** 参数指定了 IIC 从设备的地址； **wait** 指定了等待起始条件产生的循环次数。该程序创建了一个起始条件并且发送从设备地址。

void IIC0_Stop(void);
该设备停止 IIC0 外部硬件；在示例程序中未使用到此函数。

MD_STATUS IIC0_MasterSendData(UCHAR* txbuf , UINT txnum);
该程序在 IIC0_MasterStart( )开始一个 “Send”发送后调用。**txbuf** 参数指向要发送字节的数组， **txnum** 参数指出总共需要发送的字节数。该程序设置缓存指针和计数，并且将第一个字节写入缓存。后续的字节在中断服务程序 MD_INTIIC0( )中发送。

MD_STATUS IIC0_MasterReceiveData(UCHAR* rxbuf, UINT rxnum);
该程序在 IIC0_MasterStart( )开始一个 “Receive” 发送后调用。**rxbuf** 参数用于保存接收到字符的缓存地址，**rxnum** 指定了接收到的字符数。在示例程序中未使用该程序。

__interrupt void MD_INTIIC0(void) ;
这是 IIC0 中断服务程序，由 INTIIC0 中断调用。根据 IIC0 主设备或是从设备，本程序调用 IIC0_MasterHandler( ) 或 IIC0_SlaveHandler( )。

MD_STATUS IIC0_SlaveHandler(void);
本程序被 MD_INTIIC0( )调用，处理主设备模式中的 IIC0 中断。

MD_STATUS IIC0_MasterHandler(void);
本程序被 MD_INTIIC0( )调用，处理从设备模式中的 IIC0 中断。

1.4.8.3 Serial_user.c

源文件 serial_user.c 包含了末节函数用于添加用户代码。在代码生成时，这些函数为空来允许用户添加特定应用的代码。

我们添加了下面的变量用于允许子程序检测 IIC0 通信的状态：

MD_STATUS UI_MasterError；	存储主设备错误
MD_STATUS UI_MasterSendEnd；	报告发送完成
MD_STATUS UI_MasterReceiveEnd；	报告接收完成
MD_STATUS UI_MasterFindSlave；	报告从设备回应，可以开始发送/接收

void IIC0_User_Init(void);
本程序被 IIC0_Init( )调用，未使用。

void CALL_IIC0_MasterError(MD_STATUS flag);
当检测到错误的时候，本程序被 IIC0_MasterHandler( )程序调用。Code was added to store the 在 UI_MasterError 中添加代码保存 **flag** 参数。

void CALL_IIC0_MasterReceiveEnd(void);
当接收计数为 0 时，本程序被 IIC0_MasterHandler( )调用。添加代码设置 UI_MasterReceiveEnd 标志为 MD_OK。在示例程序中未使用。

void CALL_IIC0_MasterSendEnd(void);
当最后一个字节发送后，本程序被 IIC0_MasterHandler( )调用。添加代码设置 UI_MasterSendEnd 标志为 MD_OK。

void CALL_IIC0_SlaveAddressMatch(void);
当设置的外部硬件 IIC0 从设备地址与接收到的地址匹配时，IIC0_SlaveHandler( )调用该程序。在示例程序中未使用。

void CALL_IIC0_MasterFindSlave(void);
当从设备地址设置完成且收到 ACK 时，IIC0_MasterHandler( )调用该程序。该程序设置 UI_MasterFindSlave 标志为 MD_OK。

下列的程序不是通过 Applilet 生成的，但也被写入示例程序中：

MD_STATUS IIC0_MasterStartAndSend(UCHAR sadr, UCHAR* txbuf, UINT txnum);
本程序联合 IIC0_MasterStart()和 IIC0_MasterSendData()函数，所以一起完成向特定的从设备发送数据。 首先调用 IIC0_MasterStart(Send, sadr, 10)设置发送的从设备地址；然后等待 UI_MasterFindSlave 被设置，来表明从设备已经响应。

然后调用 IIC0_MasterSendData(txbuf, txnum)用来发送指定的数据。该程序等待 UI_MasterSendDone 标志被设置来表明发送完成。

在等待的过程中，检测 UI_MasterError 标志，以确定发送的从设备地址或数据是否发送错误。

1.4.9 其它 Applilet 生成的文件

在示例程序中，Applilet 生成了一些其它源文件。这些文件和功能如下所示。

文件	功能
Macrodriver.h	Applilet 生成的程序的头文件
Systeminit.c	SystemInit()和 hdwinit()函数用于初始化
Main.c	主程序
System.h	时钟相关的定义
System.c	Clock_Init()函数
Option.asm	选项字节和安全字节的定义
Option.inc	选项字节和安全字节的定义设置

1.4.10 示范程序中非 Applilet 生成的文件

示范程序中包含下列文件，它们不是由 Applilet 生成的。

文件	功能
define.h	定义负极按键值
lcd.h	高电平 LCD 功能的头文件
lcd.c	用于 LCD 显示的写入字符和字符串的代码
LcdDrvApp.h	LCD 驱动函数的头文件
LcdDrvApp.c	初始化代码，写入和读取 LCD 显示控制器； 这些函数调用由 Applilet 生成的 IIC0 程序

1.5 示范平台

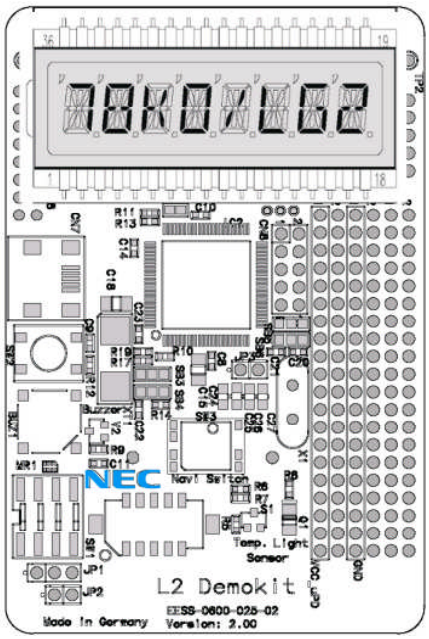
在本文档发行后从 NEC 可用的开发工具中选用一个示范平台。在一些情况下用户可以在感兴趣的 NEC 微控制器中使用标准现货元件复制相同的硬件。

1.5.1 资源

为了示范程序，使用到下面的资源：

- 带有 8 位微控制器 uPD78F0397 的 DemoKit-LG2 示范板
- DemoKit-LG2 资源:
  - 5 位置负极按键 SW3
  - 8 字符 LCD 显示

如果想要了解上面列出的硬件的详细情况，请参考相应的用户手册。

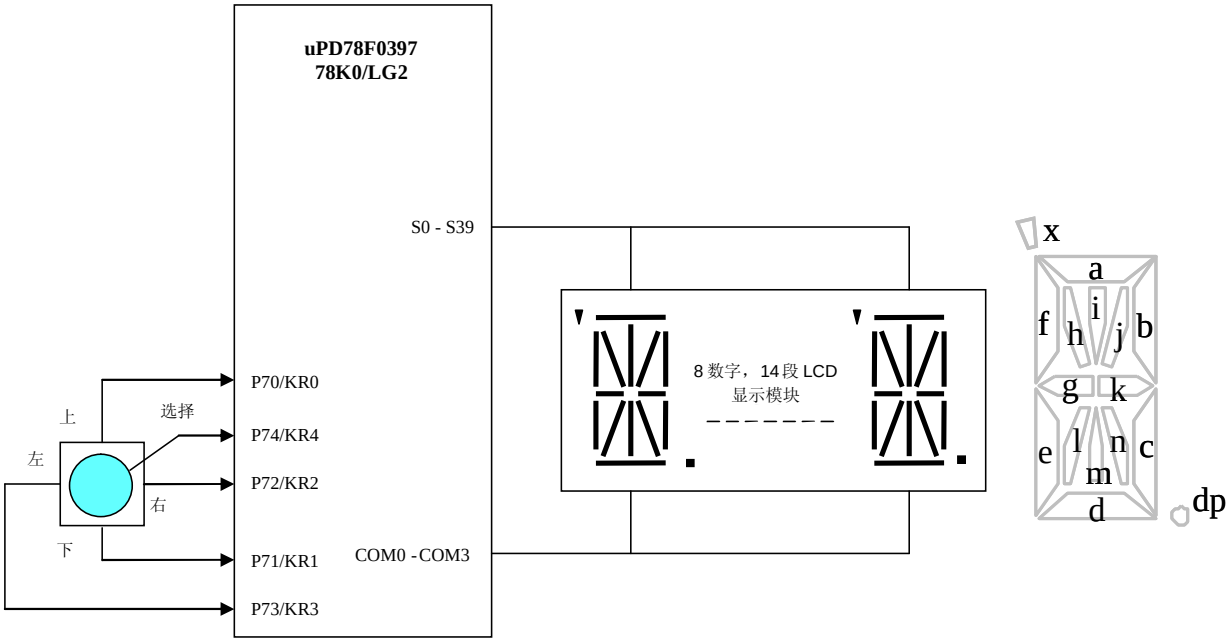


1.5.2 示例程序

假定硬件已经正确的配置并且示范程序的代码已经写入微控制器 uPD78F0397，按照如下方式进行示范：

- 按下右按键　　在显示屏上观察“ RIGHT ”
- 按下左按键　　在显示屏上观察“ LEFT ”
- 按下上按键　　在显示屏上观察“ UP ”
- 按下下按键　　在显示屏上观察“ DOWN ”
- 按下选择按键　在显示屏上观察“ SELECT ”

1.6 硬件模块图



LCD 引脚分配和连接

LCD 引脚	COM1	COM2	COM3	COM4	78K0/LG2 MCU	LCD 引脚	COM1	COM2	COM3	COM4	78K0/LG2 MCU
1	X1	1F	1E	1D	S0	36	1H	1G	1L	1M	S2
2	1I	1J	1K	1N	S1	35	1A	1B	1C	DP1	S3
3	X2	2F	2E	2D	S4	34	2H	2G	2L	2M	S6
4	2I	2J	2K	2N	S5	33	2A	2B	2C	DP2	S7
5	X3	3F	3E	3D	S8	32	3H	3G	3L	3M	S10
6	3I	3J	3K	3N	S9	31	3A	3B	3C	DP3	S11
7	X4	4F	4E	4D	S12	30	4H	4G	4L	4M	S14
8	4I	4J	4K	4N	S13	29	4A	4B	4C	DP4	S15
9	X5	5F	5E	5D	S16	28	5H	5G	5L	5M	S18
10	5I	5J	5K	5N	S17	27	5A	5B	5C	DP5	S19
11	X6	6F	6E	6D	S20	26	6H	6G	6L	6M	S22
12	6I	6J	6K	6N	S21	25	6A	6B	6C	DP6	S23
13	X7	7F	7E	7D	S24	24	7H	7G	7L	7M	S26
14	7I	7J	7K	7N	S25	23	7A	7B	7C	DP7	S27
15	X8	8F	8E	8D	S28	22	8H	8G	8L	8M	S30
16	8I	8J	8K	8N	S29	21	8A	8B	8C	DP8	S31
17	NC	NC	NC	COM4	COM4	20	COM0	NC	NC	NC	COM0
18	NC	NC	COM3	NC	COM3	19	NC	COM1	NC	NC	COM1

1.7 软件模块

文件	目的	由 Applilet 生成	用户修改
Main.c	主程序	是	是
Macrodriver.h	Applilet 使用的通用定义	是	否
System.h	与 Clock 相关的定义	是	否
Systeminit.c	SystemInit()和 hdwinit()函数	是	否
System.c	Clock_Init()函数	是	否
Int.h	与中断相关的定义	是	是[注1]
Int.c	与按键返回中断相关的函数	是	否
Int_user.h	按键返回中断的用户代码	是	是[注1]
Serial.h	与 IIC0 相关的定义	是	是[注2]
Serial.c	与 IIC0 相关的函数	是	是[注2]
Serial_user.c	用于 IIC0 回调程序的用户代码	是	是[注2]
Timer.h	与定时器相关的定义	是	否
Timer.c	定时器功能	是	否
TIMER_user.c	用于定时器中断处理的用户代码	是	是
Option.inc	选项字节，POC 和安全定义	是	否
Option.asm	选项字节，POC 和安全数据	是	否
define.h	负极按键值的定义	--	是
Lcd.h	LCD 高级函数定义	--	是
Lcd.c	LCD 高级函数	--	是
LcdDrvApp.h	LCD 控制器驱动定义	--	是
LcdDrvApp.c	LCD 控制器驱动函数	--	是

注 1: Int.h 添加了对 sw3_in 的声明，该变量用于存储负极按键去抖动输入。Int_user.c 添加了变量 sw3_in，以及处理按键返回中断的代码。

注 2: Serial.h 添加了与 IIC0 相关的全局变量的声明，在 Serial_user.c 中定义，并且声明了函数 IIC0_Init( )(缺省未声明)和 IIC0_MasterStartAndSend( )。Serial.c 去除了 78K0/LG2 不支持的 P6 的设置，并且使用 NEC C 编译器中的等价指令取代了"asm"版本中的 DI 和 EI 指令。Serial_user.c 将 IIC0 回调函数设置与 IIC0 相关的全局变量，并且添加了 IIC0_MasterStartAndSend( )函数。

2 附录A – 开发工具

在应用笔记中使用了下面的软件和硬件工具。

2.1 软件工具

工具	版本	内容
78K0KX2 的 Applilet	V1.51	78K0/KE2 设备的源代码产生工具
PM Plus	V5.20	用于程序编译和链接的工程管理器
CC78K0	V3.60	NEC 78K0 设备的 C 编译器
RA78K0	V3.70	NEC 78K0 设备的汇编器
DF0397.78K	V1.01	uPD78F0397 的设备文件

注:用于为 78K0/Kx2 系列微控制器产生代码的 Applilet 版本。78K0/KE2(uPD78F0537_64)是选择的用于输出的特定设备。该设备与 DemoKit-LG2 中的 78K0/LG2 设备(uPD78F0397)非常相似。

2.2 硬件工具

工具	版本	内容
DemoKit-LG2	V2.00	78K0397 (78K0/LG2)的示范工具

3 附录B – 软件列表

3.1 Main.c

```
/*  NEC LCD 应用笔记的 main.c */
/*
*********************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 : main.c
**  概要 : 该文件执行主函数
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器:  NEC/CC78K0
**
*********************************************************************************
*/
/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "int.h"
#include "timer.h"
#include "serial.h"
/*
*********************************************************************************
**  宏定义
*********************************************************************************
*/

/* 包含文件 for DemoKit-LG2 LCD */
#include "defines.h"
#include "lcd.h"
#include "LcdDrvApp.h"

//-------------------------------------------------------------------------------
// 全局字符串常量
//-------------------------------------------------------------------------------
const unsigned char *s_UP        = "   UP   ";
const unsigned char *s_DOWN      = "  DOWN  ";
const unsigned char *s_LEFT      = "  LEFT  ";
const unsigned char *s_RIGHT     = "  RIGHT ";
const unsigned char *s_SELECT    = " SELECT ";
const unsigned char *s_MULTIPLE  = "MULTIPLE";

/*
**-------------------------------------------------------------------------------
**
```

```
**  概要:
**     主函数
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-------------------------------------------------------------------------
*/
void main( void )
{
       IMS = MEMORY_IMS_SET;
       IXS = MEMORY_IXS_SET;
       /* TODO. 添加用户代码 */

       /* 初始化 IIC */
       IIC0_Init();

       /* 初始化 LCD */
       LCD_Init();

       while(!sw3_in) {
              LCD_string_shift((unsigned char *)"NEC DEMOKIT-LG2");
              LCD_string_shift((unsigned char *)"LCD APP NOTE");
              LCD_string_shift((unsigned char *)"PRESS SW3 TO START");
       }
       sw3_in = 0;

       LCD_string((unsigned char *)" -SW3-  ", 0);

       while(1){
              if (sw3_in != 0) {
                     switch(sw3_in) {
                     case UP:
                            LCD_string(&s_UP[0],0);
                            break;
                     case DOWN:
                            LCD_string(&s_DOWN[0],0);
                            break;
                     case LEFT:
                            LCD_string(&s_LEFT[0],0);
                            break;
                     case RIGHT:
                            LCD_string(&s_RIGHT[0],0);
                            break;
                     case SELECT:
                            LCD_string(&s_SELECT[0],0);
                            break;
                     default:
                            LCD_string(&s_MULTIPLE[0],0);
                            break;
                     }
                     sw3_in=0;
              }
       }
}
```

3.2 Macrodriver.h

```
/*
***************************************************************************
**
```

```
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 : macrodriver.h
**  概要 : 该文家为通用头文件
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器:  NEC/CC78K0
**
************************************************************************************
*/
#ifndef      _MDSTATUS_
#define _MDSTATUS_

#pragma sfr
#pragma di
#pragma ei
#pragma NOP
#pragma HALT
#pragma STOP

/* 数据类型定义 */
typedef      unsigned long ULONG;
typedef      unsigned int  UINT;
typedef      unsigned short      USHORT;
typedef      unsigned char UCHAR;
typedef      unsigned char BOOL;

#define      ON     1
#define      OFF    0

#define      TRUE   1
#define      FALSE  0

#define IDLE  0                  /* 空闲状态 */
#define READ  1                  /* 读取模式 */
#define WRITE 2                  /* 写入模式 */

#define SET   1
#define CLEAR 0

#define MD_STATUS          unsigned short
#define MD_STATUSBASE             0x0

/* 状态列表定义 */
#define MD_OK                     MD_STATUSBASE+0x0   /* 寄存器设置 OK */
#define MD_RESET                  MD_STATUSBASE+0x1   /* 复位输入 */
#define MD_SENDCOMPLETE                 MD_STATUSBASE+0x2   /* 发送数据完毕 */
#define MD_OVF                          MD_STATUSBASE+0x3   /* 定时器计数溢出 */

/* 错误列表定义 */
#define MD_ERRORBASE       0x80
#define MD_ERROR           MD_ERRORBASE+0x0    /* 错误 */
#define MD_RESOURCEERROR   MD_ERRORBASE+0x1    /* 无资源可用 */
#define MD_PARITYERROR            MD_ERRORBASE+0x2    /* UARTn 奇偶错误 */
```

```
#define MD_OVERRUNERROR           MD_ERRORBASE+0x3    /* UARTn 溢出错误 */
#define MD_FRAMEERROR             MD_ERRORBASE+0x4    /* UARTn 帧错误 */
#define MD_ARGERROR        MD_ERRORBASE+0x5    /* 参数输入错误 */
#define MD_TIMINGERROR            MD_ERRORBASE+0x6    /* 时序操作错误 */
#define MD_SETPROHIBITED   MD_ERRORBASE+0x7    /* 禁止设置 */
#define MD_DATAEXISTS             MD_ERRORBASE+0x8    /* TXBn 寄存器中下一个要被发送的数据 */
#define MD_SPT                    MD_STATUSBASE+0x8   /* IIC 停止 */
#define MD_NACK                   MD_STATUSBASE+0x9   /* IIC 无 ACK */
#define MD_SLAVE_SEND_END  MD_STATUSBASE+0x10  /* IIC 从设备发送完成 */
#define MD_SLAVE_RCV_END   MD_STATUSBASE+0x11   /* IIC 从设备接收完成 */
#define MD_MASTER_SEND_END MD_STATUSBASE+0x12   /* IIC 主设备发送完成 */
#define MD_MASTER_RCV_END  MD_STATUSBASE+0x13  /* IIC 主设备接收完成 */

/* 主时钟和副时钟作为时钟源 */
enum ClockMode { HiRingClock, SysClock };

/* IMS 和 IXS 的数值 */
#define  MEMORY_IMS_SET           0xCC
#define  MEMORY_IXS_SET           0x00
/* 清除 IO 寄存器位并且设置 IO 寄存器位 */
#define ClrIORBit(Reg, ClrBitMap) Reg &= ~ClrBitMap
#define SetIORBit(Reg, SetBitMap) Reg |= SetBitMap

enum INTLevel { Highest, Lowest };


#define       SYSTEMCLOCK  8000000
#define       SUBCLOCK     32768
#define       MAINCLOCK    8000000
#define       FRCLOCK      8000000
#define       FRCLOCKLOW   240000


#endif
```

3.3 System.h

```
/*
*********************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  system.h
**  概要 :    该文件为 SYSTEM 模块执行设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*********************************************************************************
*/
#ifndef       _MDSYSTEM_
#define       _MDSYSTEM_
```

```
/*
*****************************************************************************
**  宏定义
*****************************************************************************
*/
#define       CG_X1STAB_SEL 0x5
#define       CG_X1STAB_STA 0x1f
#define       CG_CPU_CLOCKSEL     0x0

enum  CPUClock  {  SystemClock,  Sys_Half,  Sys_Quarter,  Sys_OneEighth,  Sys_OneSixteen,
Sys_SubClock };
enum PSLevel { PS_STOP, PS_HALT };
enum StabTime { ST_Level0, ST_Level1, ST_Level2, ST_Level3, ST_Level4 };

void  Clock_Init( void );

#endif
```

3.4 Systeminit.c

```
/*
*****************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  systeminit.c
**  概要 :    该文件执行宏初始化。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*****************************************************************************
*/
/*
*****************************************************************************
**  包含文件
*****************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "int.h"
#include "timer.h"
#include "serial.h"
/*
*****************************************************************************
**  宏定义
*****************************************************************************
*/

/*
**---------------------------------------------------------------------------
**
**  概要:
**     初始化每一个宏
```

```
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-------------------------------------------------------------------------------
*/
void SystemInit( void )
{
       /* 时钟发生器初始化 */
       Clock_Init();
       /* INT 初始化 */
       INT_Init();
       /* TM50 初始化 */
       TM50_Init();
       /* TM51 初始化 */
       TM51_Init();
}

/*
**-------------------------------------------------------------------------------
**
**  概要:
**     初始化硬件设置
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-------------------------------------------------------------------------------
*/
void hdwinit( void )
{
       DI( );
       SystemInit( );
       EI( );
}
```

3.5 System.c

```
/*
*********************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  system.c
**  概要 :    该设备为 SYSTEM 模块执行设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
```

```
**
**  编译器 :  NEC/CC78K0
**
*******************************************************************************
*/
/*
*******************************************************************************
**  包含文件
*******************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
/*
*******************************************************************************
**  宏定义
*******************************************************************************
*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     初始化时钟发生器和振荡稳定时间。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void Clock_Init( void )
{
       ClrIORBit(MCM, 0x05);                    /* 高速振荡器运行 */

       SetIORBit(MCM, 0x01);                    /* 外部硬件时钟:frh */
       SetIORBit(PM12, 0x18);                   /* P123/124 输入模式 */
       ClrIORBit(OSCCTL, 0x20);          /* XT1 输入模式 */
       SetIORBit(OSCCTL, 0x10);
       SetIORBit(MOC, 0x80);                    /* 停止 X1 时钟 */
       PCC = CG_CPU_CLOCKSEL;
}
```

3.6 Int.h

```
/*
*******************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  int.h
**  概要 :    该文件为 INT 模块执行设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
```

```
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*******************************************************************************
*/

#ifndef      _MDINT_
#define      _MDINT_
/*
*******************************************************************************
**  宏定义
*******************************************************************************
*/
#define      EGP_INT      0x0
#define      EGN_INT      0x0
#define      PU7_KR 0x1f
#define      PM7_KR 0x1f
#define      KRM_KR 0x1f

enum ExternalINT {
       EX_INTP0, EX_INTP1, EX_INTP2, EX_INTP3,
       EX_INTP4, EX_INTP5, EX_INTP6, EX_INTP7
       };

enum INTInputEdge {
       None, RisingEdge, FallingEdge, BothEdge
       };

enum MaskableSource {
       INT_LVI, INT_INTP0, INT_INTP1, INT_INTP2,
       INT_INTP3, INT_INTP4, INT_INTP5, INT_SRE6,
       INT_SR6, INT_ST6, INT_CSI10_ST0, INT_TMH1,
       INT_TMH0, INT_TM50, INT_TM000, INT_TM010,
       INT_AD, INT_SR0, INT_WTI, INT_TM51,
       INT_KR, INT_WT, INT_INTP6, INT_INTP7,
       INT_IIC0_DMU, INT_CSI11, INT_TM001, INT_TM011
       };
void INT_Init( void );
__interrupt void MD_INTKR( void );

/* 全局值用于按键输入去抖动*/
extern __sreg volatile unsigned char sw3_in;

#endif
```

3.7 Int.c

```
/*
*******************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  int.c
**  概要 :    该文件执行 INT 模块的设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
```

```
**
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*************************************************************************
*/

/*
*************************************************************************
**  包含文件
*************************************************************************
*/
#include "macrodriver.h"
#include "int.h"
/*
*************************************************************************
**  宏定义
*************************************************************************
*/

/*
**-----------------------------------------------------------------------
**
**  概要:
**     该函数初始化外部中断，按键返回函数。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------
*/
void INT_Init( void )
{
       EGP = EGP_INT;
       EGN = EGN_INT;

       KRMK = 1;                        /* 禁止 INTKR */
       PU7 |= PU7_KR;
       PM7 |= PM7_KR;
       KRM = KRM_KR;                    /* 设置 KR 输入模式 */
       KRPR = 1;
       KRIF = 0;
       KRMK = 0;                        /* 允许 INTKR */
}
```

3.8 Int_user.c

```
/*
*************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
```

```
**
**  文件名 :  int_user.c
**  概要 :    该文家执行 INT 模块的设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*************************************************************************
*/

#pragma      interrupt    INTKR  MD_INTKR

/*
*************************************************************************
**  包含文件
*************************************************************************
*/
#include "macrodriver.h"
#include "int.h"
/*
*************************************************************************
**  宏定义
*************************************************************************
*/

#include "timer.h"  // TM51 的程序和值
/* 全局值用于按键输入去抖动 */
__sreg volatile unsigned char sw3_in;

/*
**-----------------------------------------------------------------------
**
**  概要:
**     INTKR 中断服务程序。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------
*/
__interrupt void MD_INTKR( void )
{
unsigned char sw3_first,sw3_second;
    sw3_first= (~P7) & 0x1f;     // 第一次读取 SW3
    TM51_ChangeTimerCondition(TM_TM51_INTERVALVALUE);
    TM51_Start();
    while(!TMIF51);             // 等待定时器 51 中断
    TM51_Stop();
    TMIF51=0;
    sw3_second= (~P7) & 0x1f;     // 第二次读取 SW3
    if(sw3_first==sw3_second)
        sw3_in=sw3_first; // SW3 去抖动
    else
        sw3_in=0;
}
```

3.9 Serial.h

```
/*
*******************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  serial.h
**  概要 :    该文件执行 SERIAL 模块的设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*******************************************************************************
*/
#ifndef _MDSERIAL_
#define _MDSERIAL_
/*
*******************************************************************************
**  全局变量
*******************************************************************************
*/
#define      IIC0_SLAVEADDRESS   0x0


enum TransferMode { Send, Receive };
MD_STATUS IIC0_MasterStart( enum TransferMode , UCHAR , UCHAR );
MD_STATUS IIC0_MasterSendData( UCHAR* , UINT );
MD_STATUS IIC0_MasterReceiveData( UCHAR* , UINT );
void IIC0_Stop( void );
__interrupt void MD_INTIIC0( void ) ;
void IIC0_User_Init( void );
MD_STATUS IIC0_SlaveHandler( void );
MD_STATUS IIC0_MasterHandler( void );
void CALL_IIC0_SlaveAddressMatch( void );
void CALL_IIC0_MasterFindSlave( void );
void CALL_IIC0_MasterSendEnd( void  );
void CALL_IIC0_MasterReceiveEnd( void  );
void CALL_IIC0_MasterError( MD_STATUS flag );

/* 用于上层程序的回调函数添加的标志 */
extern MD_STATUS UI_MasterError;
extern MD_STATUS UI_MasterSendEnd;
extern MD_STATUS UI_MasterReceiveEnd;
extern MD_STATUS UI_MasterFindSlave;

/* 为初始化程序添加的定义 */
void IIC0_Init( void );

/* 添加联合函数 */
MD_STATUS IIC0_MasterStartAndSend(UCHAR sadr, UCHAR* txbuf, UINT txnum);

#endif
```

3.10 Serial.c

```
/*
*********************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  serial.c
**  概要 :    该文件执行 SERIAL 模块的设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*********************************************************************************
*/


#pragma interrupt   INTIIC0             MD_INTIIC0

/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "serial.h"
UCHAR  iic0_m_sta_flag;           /* 在主设备模式下检测用于发送地址的开始标志 */
UCHAR  *iic0_m_send_pbuf;         /* 主设备模式下发送数据的指针 */
UINT   iic0_m_send_size;          /* 主设备模式下发送数据的大小 */
UCHAR  *iic0_m_rev_pbuf;          /* 主设备模式下接收数据的指针 */
UINT   iic0_m_rev_size;           /* 主设备模式下接收数据的大小 */
UCHAR  iic0_s_sta_flag;           /* 在从设备模式下检测用于发送地址的开始标志 */
UCHAR  *iic0_s_send_pbuf;         /* 从设备模式下发送数据的指针 */
UINT   iic0_s_send_size;          /* 从设备模式下发送数据的大小 */
UCHAR  *iic0_s_rev_pbuf;          /* 从设备模式下接收数据的指针 */
UINT   iic0_s_rev_size;           /* 从设备模式下接收数据的大小 */

/*
**-------------------------------------------------------------------------------
**
** 概要:
**     该函数初始化 IIC0 模块。
**
** 参数:
**     无
**
** 返回值:
**     无
**
**-------------------------------------------------------------------------------
*/
void IIC0_Init( void )
{
```

```
        ClrIORBit(IICC0, 0x80);                 /* 停止 IIC0 */

        SetIORBit(MK1H, 0x1);                   /* 禁止中断 */

        ClrIORBit(PM6, 0x03);                   /* 端口设置 */
// 为 78F0397(78K0/LG2)移除 P6 的设置
//      ClrIORBit(P6, 0x03);

        SetIORBit(IICF0, 0x02);                 /* 起始条件不需要停止条件 */
        SetIORBit(IICF0, 0x01);                 /* 通讯保留 – 禁止 */
        SetIORBit(IICC0, 0x10);                 /* 停止条件中断 – 允许 */
        ClrIORBit(IICC0, 0x08);                 /* 中断控制 - 8 个时钟的下降沿 */

        /* 发送时钟 */
        /* fprs/88 */
        ClrIORBit(IICCL0, 0x08);          /* 正常模式 */
        ClrIORBit(IICCL0, 0x3);                 /* CL00 = 0 CL01 = 0 */
        ClrIORBit(IICX0, 0x1);                  /* 禁止扩展 */

        /* 选择中断优先级 */
        SetIORBit(PR1H, 0x1);                   /* 中断优先级低 */

        ClrIORBit(MK1H, 0x1);                   /* 允许中断 */

        IIC0_User_Init( );

        return;
}

/*
**-----------------------------------------------------------------------------
**
** 概要:
**      该函数在主设备模式下负责起始 IIC0。
**
** 参数:
**      enum TransferMode mode :    选择发送模式
**              Send :              发送数据
**              Receive :           接收数据
**      UCHAR adr :                 为选择的从设备设置地址
**      UCHAR wait :                当获得起始条件时设置所需的等待
**
** 返回值:
**      MD_OK
**      MD_ERROR
**      MD_ARGERROR
**
**-----------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterStart( enum TransferMode mode, UCHAR adr, UCHAR wait )
{
        /* 总线检测 */
//      __asm("di"); // 使用预处理函数取代在线汇编
        DI();
        if( IICF0 & 0x40 ){
//              __asm("ei");        /* 总线繁忙 */
                EI();  // 使用预处理函数取代在线汇编
                return MD_ERROR;
        }

        /* 起始 IIC0 */
        SetIORBit(IICC0, 0x18);           /* SPIE0 = WTIM0 = 1 */
```

```
        SetIORBit(IICC0, 0x80);           /* IICE0 = 1 */

        SetIORBit(IICC0, 0x02);           /* 产生起始条件 */
//      __asm("ei");
        EI();  // 使用预处理功能取代在线汇编

        /* 等待 */
        while( wait-- );

        if( !(IICS0 & 0x2) ){      /* 检测起始条件 */
                return MD_ERROR;
        }

        /* 设置发送模式到地址 */
        /* 在从设备的地址位 0 选择发送或是接收 */
        if( mode == Send ){
                ClrIORBit(adr, 0x01);      /* 如果主设备为发送模式，清除位 0 */
        }
        else if( mode == Receive ){
                SetIORBit(adr, 0x01);      /* 如果主设备位接收模式，设置位 0 */
        }
        else{
                return MD_ARGERROR;
        }

        iic0_m_sta_flag = 0;
        IIC0 = adr;                /* 设置地址 */

        return MD_OK;
}

/*
**-----------------------------------------------------------------------------
**
** 概要:
**     该函数负责停止 IIC0。
**
** 参数:
**     无
**
** 返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void IIC0_Stop( void )
{
        ClrIORBit(IICC0, 0x80);           /* 停止发送 */
        return;
}

/*
**-----------------------------------------------------------------------------
**
** 概要:
**     该函数负责在主设备模式下负责 IIC0 数据发送。
**
** 参数:
**     UCHAR* txbuf :       发送缓存指针
**     UINT txnum :         缓存容量
**
** 返回值:
**     MD_OK
```

```
**     MD_ERROR :           不能发送地址
**
**-------------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterSendData(UCHAR* txbuf, UINT txnum)
{
       if( iic0_m_sta_flag == 0 ){
              return MD_ERROR;           /* 不能发送地址 */
       }

       /* 设置参数 */
       iic0_m_send_size = txnum;
       iic0_m_send_pbuf = txbuf;

       IIC0 = *iic0_m_send_pbuf ++ ;           /* 开始条件 */
       iic0_m_send_size--;

       return MD_OK;
}

/*
**-------------------------------------------------------------------------------
**
** 概要:
**     该函数负责在主设备模式下的 IIC 数据的接收。
**
** 参数:
**     UCHAR* rxbuf :       接收缓存指针
**     USHORT rxnum :       缓存容量
**
** 返回值:
**     MD_OK
**     MD_ERROR :           不能发送地址
**
**-------------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterReceiveData(UCHAR* rxbuf, UINT rxnum)
{
       if( iic0_m_sta_flag == 0 ){
              return MD_ERROR;           /* 不能发送地址 */
       }

       /* 设置参数 */
       iic0_m_rev_size = rxnum;
       iic0_m_rev_pbuf = rxbuf;

       ClrIORBit(IICC0, 0x08);                 /* 清除 WTIM0 */
       SetIORBit(IICC0, 0x04);                 /* 设置 ACKE0 */

       SetIORBit(IICC0, 0x20);                 /* 开始接收 */

       return MD_OK;
}

/*
**-------------------------------------------------------------------------------
**
** 概要:
**     IIC0 中断服务程序
**
** 参数:
**     无
**
```

```
** 返回值:
**     无
**
**-------------------------------------------------------------------------------
*/
__interrupt void MD_INTIIC0( void )
{
       MD_STATUS sta;
       if( IICS0 & 0x80 ) {
              sta = IIC0_MasterHandler();
       }
       else {
              sta = IIC0_SlaveHandler();
       }
}

/*
**-------------------------------------------------------------------------------
**
** 概要:
**     该函数在产生 IIC0 中断请求时被调用
**
** 参数:
**     无.
**
** 返回值:
**     MD_OK
**     MD_ERROR :    不能获取地址
**                   非从设备模式
**     MD_SLAVE_RCV_END : 接收到所有的数据
**     MD_SLAVE_SEND_END : 发送完所有数据
**     MD_SPT : 获得停止条件
**
**-------------------------------------------------------------------------------
*/
MD_STATUS IIC0_SlaveHandler( void )
{
       /* 控制停止条件 */
       if( IICS0 & 0x01 ){                              /* 获得停止条件 */
              /* 从设备发送完成并且获得停止条件 */
              if( iic0_s_sta_flag &&( iic0_s_send_size == 0 ) ){
                     return MD_SLAVE_SEND_END;
              } else {
                     return MD_SPT;
              }
       }

       /* 控制获取地址 */
       if( iic0_s_sta_flag == 0 ){
              if( !(IICS0 & 0x20) ){                   /* 检测 EXC0 -> 外部代码 */
                     if( IICS0 & 0x10 ){               /* 检测 COI0 -> 地址 */
                            iic0_s_sta_flag = 1;
                            CALL_IIC0_SlaveAddressMatch();   /* 从设备地址匹配 */
                     }
                     else{
                            return MD_ERROR;
                     }
              }
              else{
                     return MD_ERROR;
              }
       }
```

```
	/* 从设备发送控制 */
	else if( IICS0 & 0x08 ){
		if(!( IICS0 & 0x04 )){			/* 检测 ACKD0 -> 确认 */
			return MD_NACK;
		}
		IIC0 = *iic0_s_send_pbuf ++ ;
		iic0_s_send_size--;
	}
	/* 从设备接收控制 */
	else{
		*iic0_s_rev_pbuf ++ = IIC0;
		iic0_s_rev_size--;
		SetIORBit(IICC0, 0x20);				/* WREL1 = 1 开始接收 */
		if( iic0_s_rev_size == 0 ){			/* 检测所有的接收数据 */
			ClrIORBit(IICC0, 0x04);			/* 清除 ACKE0 */
			return MD_SLAVE_RCV_END;
		}
	}
	return MD_OK;
}

/*
**-----------------------------------------------------------------------------
**
** 概要:
**     该函数在发生 IIC0 中断请求时被调用。
**
** 参数:
**     无.
**
** 返回值:
**     MD_OK
**     MD_ERROR :          在发送地址后未收到 ack
**                         非主设备模式
**                         从设备未发送 ack
**     MD_MASTER_RCV_END : 所有数据接收完成
**     MD_MASTER_SEND_END : 所有数据发送完毕
**
**-----------------------------------------------------------------------------
*/
MD_STATUS IIC0_MasterHandler( void )
{
	/* 控制停止条件 */
	if( !( IICF0 & 0x40 ) ){			/* 获取停止条件 */
		CALL_IIC0_MasterError(MD_SPT);
		return MD_SPT;
	}

	/* 控制发送条件 */
	if( !iic0_m_sta_flag ){
		if( IICS0 & 0x4 ){			/* 检测 ACK */
			iic0_m_sta_flag = 1;		/* 地址完成 */
			CALL_IIC0_MasterFindSlave();
		}
		else{
			CALL_IIC0_MasterError(MD_NACK);
			return MD_NACK;
		}
	}
	/* 主设备发送控制 */
	else if( IICS0 & 0x8 ){
		if( !(IICS0 & 0x4) ){			/* 检测 ACK */
			SetIORBit(IICC0, 0x01);			/* 产生停止条件 */
```

```
                CALL_IIC0_MasterError(MD_NACK);
                return MD_NACK;
            }

            if( !iic0_m_send_size ){                /* 发送完成 */
                SetIORBit(IICC0, 0x01);                 /* 产生停止条件 */
                CALL_IIC0_MasterSendEnd( );
                return MD_MASTER_SEND_END;
            }
            IIC0 = *iic0_m_send_pbuf ++ ;               /* 发送数据 */
            iic0_m_send_size--;
        }
        /* 主设备接收控制 */
        else {
            *iic0_m_rev_pbuf ++ = IIC0;                 /* 接收数据 */
            iic0_m_rev_size--;
            if( iic0_m_rev_size == 0 ){                 /* 接收完成 */
                ClrIORBit(IICC0, 0x04);                 /* ACK 停止 */
                SetIORBit(IICC0, 0x01);                 /* 产生停止条件 */
                CALL_IIC0_MasterReceiveEnd( );
                return MD_MASTER_RCV_END;
            }
            SetIORBit(IICC0, 0x20);                     /* 开始接收 */
        }
        return MD_OK;
}
```

3.11 Serial_user.c

```
/*
*********************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  serial_user.c
**  概要 :    该文件执行 SERIAL 模块的设备驱动。
**  APIlib :  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器 :  NEC/CC78K0
**
*********************************************************************************
*/
/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "serial.h"

/* 通过回调程序添加标志用于上层程序 */
MD_STATUS UI_MasterError;
```

```
MD_STATUS UI_MasterSendEnd;
MD_STATUS UI_MasterReceiveEnd;
MD_STATUS UI_MasterFindSlave;

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     当 IIC0 初始化时，该函数为空函数以便添加用户代码
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/

void IIC0_User_Init( void )
{

}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     主设备错误,
**     回调程序对于用户开放
**
**  参数:
**     MD_STATUS flag
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void CALL_IIC0_MasterError( MD_STATUS flag )
{
       /* 用户操作 */
       UI_MasterError = flag;
       return;
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     主设备接收完毕,
**     回调程序对于用户开放
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void CALL_IIC0_MasterReceiveEnd( void )
{
```

```
        /* 用户操作 */
        UI_MasterReceiveEnd = MD_OK;
        return;
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     主设备发送完成,
**     回调程序对于用户开放
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void CALL_IIC0_MasterSendEnd( void )
{
        /* 用户操作 */
        UI_MasterSendEnd = MD_OK;
        return;
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     IIC0 从设备地址匹配
**     回调程序对于用户开放
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void CALL_IIC0_SlaveAddressMatch( void )
{
        /* 用户操作 */
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     主设备发现从设备地址
**     回调程序对于用户开放
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void CALL_IIC0_MasterFindSlave( void )
```

```
{
	/* 用户操作 */
	UI_MasterFindSlave = MD_OK;
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**     联合 IIC0_MasterStart(Send, (sadr), 10)
**        和 IIC0_MasterSendData(UCHAR* txbuf, UINT txnum)
**
**  参数:
**     UCHAR sadr :  为选择从设备设置地址
**     UCHAR* txbuf :       发送缓存指针
**     UINT txnum :  缓存容量
**
**  返回值:
**     MD_OK
**     MD_ERROR
**     MD_ARGERROR
**  MD_NACK – 从设备地址超时
**
**---------------------------------------------------------------------------
*/

MD_STATUS IIC0_MasterStartAndSend(UCHAR sadr, UCHAR* txbuf, UINT txnum)
{
MD_STATUS status;
int i,j;

	// 在需要时初始化 IIC
//     IIC0_Init();

	// 设置第一次操作
	UI_MasterError = MD_OK;
	UI_MasterSendEnd = MD_ERROR;
	UI_MasterFindSlave = MD_ERROR;

	status = IIC0_MasterStart(Send, sadr, 10);
	if (status != MD_OK) {
		return status;
	}

	i = 10000;
	do {
		i--;
	} while ( (UI_MasterFindSlave == MD_ERROR) && (UI_MasterError == MD_OK) && ( i >
0 ) );

	if (i == 0) {
		return MD_NACK;
	}

	if (UI_MasterError != MD_OK) {
		return UI_MasterError;
	}

	// got slave address ok here
	status = IIC0_MasterSendData(txbuf, txnum);    // 发送数据字节
	if (status != MD_OK) {
		return status;
	}
	i = 10000;
	do {
```

```
            i--;
      } while ( (UI_MasterError == MD_OK) && (UI_MasterSendEnd == MD_ERROR) && (i > 0) );

      if (i == 0) {
            return MD_NACK;
      }
      if (UI_MasterError != MD_OK) {
            return UI_MasterError;
      }
      if (UI_MasterSendEnd != MD_OK) {
            return UI_MasterSendEnd;
      }

      // 修改相关的其它 LCD 代码
//    SetIORBit(MK1H, 0x1);                  /* 禁止中断 */
//    ClrIORBit(IICC0, 0x10);                /* 清除 SPIE0 位 */
//    ClrIORBit(IF1H, 0x01);                 /* 清除 IICIF0 位 */
      return MD_OK;/* 无错误 */
}
```

3.12 Timer.h

```
/*
*********************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 : timer.h
**  概要 : 该文件执行定时器模块的设备驱动
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器:  NEC/CC78K0
**
*********************************************************************************
*/

#ifndef _MDTIMER_
#define _MDTIMER_
/*
*********************************************************************************
**  宏定义
*********************************************************************************
*/
#define      REGVALUE_MAX  0xff
#define      TM_TM00_CLOCK 0x0
#define      TM_TM00_INTERVALVALUE      0x00
#define      TM_TM00_SQUAREWIDTH 0x00
#define      TM_TM00_PPGCYCLE    0x00
#define      TM_TM00_PPGWIDTH    0x00
#define      TM_TM00_ONESHOTCYCLE0x00
#define      TM_TM00_ONEPULSEDELAY      0x00
#define      TM_TM01_CLOCK 0x0
#define      TM_TM01_INTERVALVALUE      0x00
#define      TM_TM01_SQUAREWIDTH 0x00
```

```
#define       TM_TM01_PPGCYCLE    0x00
#define       TM_TM01_PPGWIDTH    0x00
#define       TM_TM01_ONESHOTCYCLE0x00
#define       TM_TM01_ONEPULSEDELAY      0x00
#define       TM_TM50_CLOCK 0x7
#define       TM_TM50_INTERVALVALUE      0x2f
#define       TM_TM50_SQUAREWIDTH 0x2f
#define       TM_TM50_PWMACTIVEVALUE     0x2f
#define       TM_TM51_CLOCK 0x7
#define       TM_TM51_INTERVALVALUE      0x8
#define       TM_TM51_SQUAREWIDTH 0x8
#define       TM_TM51_PWMACTIVEVALUE     0x8
#define       TM_TMH0_CLOCK 0x0
#define       TM_TMH0_INTERVALVALUE      0x00
#define       TM_TMH0_SQUAREWIDTH 0x00
#define       TM_TMH0_PWMCYCLE    0x00
#define       TM_TMH0_PWMDELAY    0x00
#define       TM_TMH1_CLOCK 0x0
#define       TM_TMH1_INTERVALVALUE      0x00
#define       TM_TMH1_SQUAREWIDTH 0x00
#define       TM_TMH1_PWMCYCLE    0x00
#define       TM_TMH1_PWMDELAY    0x00
#define       TM_TMH1_CARRIERDELAY0x00
#define       TM_TMH1_CARRIERWIDTH0x00


/* 定时器 00 到 01，50，51，H0，H1 配置器初始化 */
void TM50_Init(void);
void TM51_Init(void);

/* 定时器启动 */
void TM50_Start(void);
void TM51_Start(void);

/* 定时器停止 */
void TM50_Stop(void);
void TM51_Stop(void);
MD_STATUS TM50_ChangeTimerCondition(UCHAR value);
MD_STATUS TM51_ChangeTimerCondition(UCHAR value);

#endif       /* _MDTIMER_*/
```

3.13 Timer.c

```
/*
************************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
**  该设备的驱动由 Applilet 创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2005
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 : timer.c
**  概要 : This file implements a 设备 driver for the timer module
**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**  设备 :    uPD78F0537
**
**  编译器:  NEC/CC78K0
```

```
**
*****************************************************************************
*/

/*
*****************************************************************************
**  包含文件
*****************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"


/*
*****************************************************************************
**  宏定义
*****************************************************************************
*/
/* TM00 脉宽测量 */

/* TM01 脉宽测量 */

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     该函数初始化 TM50 模块。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void TM50_Init( void )
{
       ClrIORBit(TMC50, 0x80);
       TCL50 = TM_TM50_CLOCK;                  /* 计数时钟=fx/8192 */
       /* TM50 间隔 */
       CR50 = TM_TM50_INTERVALVALUE;
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**     该函数可以启动 TM50 计数器。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-----------------------------------------------------------------------------
*/
void TM50_Start( void )
{
       /* TM50 间隔 */
       SetIORBit(TMC50, 0x80);
}
```

```
/*
**---------------------------------------------------------------------------
**
**  概要:
**     该函数可以停止 TM50 计数器并且清除计数寄存器 。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**---------------------------------------------------------------------------
*/
void TM50_Stop( void )
{
       ClrIORBit(TMC50, 0x80);
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**     该函数可以改变 TM50 条件。
**
**  参数:
**     UCHAR :         值
**  返回值:
**     MD_OK
**     MD_ERROR
**
**---------------------------------------------------------------------------
*/
MD_STATUS TM50_ChangeTimerCondition(UCHAR value)
{
       CR50 =value;
       return MD_OK;
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**     该函数初始化 TM51 模块。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**---------------------------------------------------------------------------
*/
void TM51_Init( void )
{
       ClrIORBit(TMC51, 0x80);
       TCL51 = TM_TM51_CLOCK;                  /* 计数时钟=fx/4096 */
       /* TM51 间隔 */
       CR51 = TM_TM51_INTERVALVALUE;
}

/*
**---------------------------------------------------------------------------
**
```

```
**  概要:
**     该功能启动 TM51 计数器。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-------------------------------------------------------------------------------
*/
void TM51_Start( void )
{
       /* TM51 间隔 */
       SetIORBit(TMC51, 0x80);
}

/*
**-------------------------------------------------------------------------------
**
**  概要:
**     该函数停止 TM51 计数器和清除计数寄存器。
**
**  参数:
**     无
**
**  返回值:
**     无
**
**-------------------------------------------------------------------------------
*/
void TM51_Stop( void )
{
       ClrIORBit(TMC51, 0x80);
}

/*
**-------------------------------------------------------------------------------
**
**  概要:
**     该函数改变 TM51 条件。
**
**  参数:
**     UCHAR :       值
**  返回值:
**     MD_OK
**     MD_ERROR
**
**-------------------------------------------------------------------------------
*/
MD_STATUS TM51_ChangeTimerCondition(UCHAR value)
{
       CR51 =value;
       return MD_OK;
}
```

3.14 TIMER_user.c

```
/*
*******************************************************************************
**
**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
```

```
**   该设备的驱动由 Applilet 创建。
**
**   版权(C)属于 NEC 电子公司 2002 - 2005
**   NEC 电子公司保留所有的权利。
**
**   使用本程序的一切后果由本人负责。
**   NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**   文件名 : timer_user.c
**   概要 : 该函数执行定时器模块的设备驱动
**   APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
**
**   设备 :    uPD78F0537
**
**   编译器:  NEC/CC78K0
**
************************************************************************************
*/

#pragma sfr
/*
************************************************************************************
**   包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"


/*
************************************************************************************
**   宏定义
************************************************************************************
*/

/* 定时器 00，定时器 01 脉宽测量 */
```

3.15 Option.inc

```
;************************************************************************************
;**
;**   在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
;**   该设备的驱动由 Applilet 创建。
;**
;**   版权(C)属于 NEC 电子公司 2002 - 2005
;**   NEC 电子公司保留所有的权利。
;**
;**   使用本程序的一切后果由本人负责。
;**   NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
;**
;**   文件名 : option.asm
;**   概要 :   该文件执行选项字节/安全 ID 设置。
;**   APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
;**
;**   设备 :   uPD78F0537
;**
;**   编译器 : NEC/CC78K0
;**
;************************************************************************************

;
;************************************************************************************
```

```
;**  宏定义
;*************************************************************************
;
OPTION_BYTE   EQU    00H
POC81  EQU    00H
POC82  EQU    00H
POC83  EQU    00H
CG_ONCHIP     EQU    02H
CG_SECURITY0  EQU    0ffH
CG_SECURITY1  EQU    0ffH
CG_SECURITY2  EQU    0ffH
CG_SECURITY3  EQU    0ffH
CG_SECURITY4  EQU    0ffH
CG_SECURITY5  EQU    0ffH
CG_SECURITY6  EQU    0ffH
CG_SECURITY7  EQU    0ffH
CG_SECURITY8  EQU    0ffH
CG_SECURITY9  EQU    0ffH
```

3.16 Option.asm

```
;*************************************************************************
;**
;**  在 8 位单片微控制器 78K0/KB2, 78K0/KC2,78K0/KD2, 78K0/KE2 和 78K0/KF2 中
;**  该设备的驱动由 Applilet 创建。
;**
;**  版权(C)属于 NEC 电子公司 2002 - 2005
;**  NEC 电子公司保留所有的权利。
;**
;**  使用本程序的一切后果由本人负责。
;**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
;**
;**  文件名 : option.asm
;**  概要 : 该文件执行选项字节/安全 ID 设置。
;**  APIlib:  NEC78K0KX2.lib V1.01 [09 Aug. 2005]
;**
;**  设备 :   uPD78F0537
;**
;**  编译器 : NEC/CC78K0
;**
;*************************************************************************

;
;*************************************************************************
;**  包含文件
;*************************************************************************
$ INCLUDE (option.inc)
       OPT_SET CSEG AT 80H
OPTION:       DB     OPTION_BYTE
       DB     POC81
       DB     POC82
       DB     POC83
       ONC_SET CSEG AT 84H
ONCHIP:       DB     CG_ONCHIP

       CSEG SECUR_ID
SECURITY0:    DB     CG_SECURITY0
SECURITY1:    DB     CG_SECURITY1
SECURITY2:    DB     CG_SECURITY2
SECURITY3:    DB     CG_SECURITY3
SECURITY4:    DB     CG_SECURITY4
SECURITY5:    DB     CG_SECURITY5
```

```
SECURITY6:    DB     CG_SECURITY6
SECURITY7:    DB     CG_SECURITY7
SECURITY8:    DB     CG_SECURITY8
SECURITY9:    DB     CG_SECURITY9
END
```

3.17 define.h

```
#define UP     0x01
#define DOWN   0x02
#define RIGHT  0x04
#define LEFT   0x08
#define SELECT 0x10

#define BAUDRATE 115200
```

3.18 Lcd.h

```
/*
*****************************************************************************
**
**  该文件为 NEC 应用笔记创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2006
**  NEC 电子公司保留所有的权利。
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  lcd.h
**  概要 :    该文件为 DemoKit-LG2 的 LCD 功能执行头文件
**
**  设备 :    uPD78F0397
**
**  编译器 :  NEC/CC78K0
**
*****************************************************************************
*/
#ifndef      _LCD_H_
#define      _LCD_H_

void Wait(unsigned char Number);
void LCD_Init(void);

__callt extern void LCD_putc(unsigned  char digit, unsigned  char data);
__callt extern void LCD_string(unsigned char const *point, unsigned char dpos);
__callt extern void LCD_string_shift(unsigned char const *point);

#endif /* _LCD_H_ */
```

3.19 Lcd.c

```
/*
*****************************************************************************
**
**  该文件为 NEC 应用笔记创建。
**
**  版权(C)属于 NEC 电子公司 2002 - 2006
**  NEC 电子公司保留所有的权利。
```

```
**
**  使用本程序的一切后果由本人负责。
**  NEC 电子公司假定不承担关于使用本文件造成的用户和第三方的损失。
**
**  文件名 :  lcd.c
**  概要 :    该文件执行 DemoKit-LG2 上的 LCD 功能
**            从 DemoKit-LG2 示例代码修改来使用 LcdDrvApp.h
**
**  设备 :    uPD78F0397
**
**  编译器 :  NEC/CC78K0
**
*******************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"  /* 位定时器 50 程序 */
#include "int.h"    /* 为 sw3_in 定义 */
#include "defines.h"
#include "lcd.h"
#include "LcdDrvApp.h"

//-----------------------------------------------------------------------------
// 全局常量: 最小的 ASCII 表用于 14 段 LCD 面板
//-----------------------------------------------------------------------------
unsigned const short characters[43]   = {

                                    0x0e70, // '0'
                                     0x2060, // '1'
                                     0x4c32, // '2'
                                     0x4872, // '3'
                                     0x4262, // '4'
                                     0x4a52, // '5'
                                     0x4e52, // '6'
                                     0x0070, // '7'
                                     0x4e72, // '8'
                                     0x4a72, // '9'
                                     0x500a, // '+'
                                     0x4002, // '-'
                                     0x4232, // '°'
                                     0x0800, // '_'
                                     0x2004, // '/'
                                     0x4c42, // 'o'
                                     0x0000, // ' '  => 空
                                     0x4672, // 'A'
                                     0x4e42, // 'B'
                                     0x0e10, // 'C'
                                     0x4c62, // 'D'
                                     0x0e12, // 'E'
                                     0x0612, // 'F'
                                     0x4e50, // 'G'
                                     0x4662, // 'H'
                                     0x1008, // 'I'
                                     0x0c70, // 'J'
                                     0xa602, // 'K'
                                     0x0e00, // 'L'
                                     0x2661, // 'M'
                                     0x8661, // 'N'
                                     0x0e70, // 'O'
                                     0x4632, // 'P'
                                     0x8e70, // 'Q'
                                     0xc632, // 'R'
                                     0x4a52, // 'S'
                                     0x1018, // 'T'
                                     0x0e60, // 'U'
                                     0x8061, // 'V'
```

```
                                0x9664, // 'W'
                                0x2009, // 'Y'
                                0x2814  // 'Z'
                               };

// 清除显示的空白串
const unsigned char *s_clear   ="        ";

//-------------------------------------------------------------------------------
// 全局变量
//-------------------------------------------------------------------------------
__sreg unsigned char transmit_buffer[2];
__sreg char message_byte_count;

//-------------------------------------------------------------------------------
// 模块名称: 等待
// 描述: 该模块产生(number * 50ms)的延迟。
//-------------------------------------------------------------------------------
void Wait(unsigned char number)
{
       TM50_Start();// 启动 50ms 定时器
       while (number > 0) {
              while (TMIF50 == 0)
                     ;             //等待时间完成的标志
              TMIF50 = 0;   // 清除标志
              number--;     // 计数减少
       }
       TM50_Stop(); // 停止定时器
       TMIF50 = 0;          // 确保标志为 0
}

//-------------------------------------------------------------------------------
// 模块名称: LCD_Init
// 描述: 调用 LcdDrv 函数初始化 LCD
//-------------------------------------------------------------------------------
void LCD_Init(void)
{
       LcdDrvInit();
       LcdDrvOnWait();

       Wait(80);                    // 电压推荐等待时间 = 4s
       LcdDrvOn();
}

//-------------------------------------------------------------------------------
// 模块名称:   LCD_putc
// 描述: 设置字符到 LCD 控制器
//-------------------------------------------------------------------------------
__callt void LCD_putc (unsigned char digit, unsigned char data)
{
    //转换特定字符
    switch(data)
    {
      case 0x2f:  data = 0x0d;  // '_'
                  break;
      case 0xf0:  data = 0x10;  // ' ' => 空
                  break;
      case 0x80:  data = 0x0c;  // '°'
                  break;
      case 0xfb:  data = 0x0a;  // '+'
                  break;
      case 0xfd:  data = 0x0b;  // '-'
                  break;
```

```
      case 0xff:  data = 0x0e;  // '/'
                  break;
      case 0x3f:  data = 0x0f;  // 'o'
                  break;
      default:    break;
    }

    // 装载发送缓存
    transmit_buffer [0] = ((characters[data])& 0xff00)>>8;
    transmit_buffer [1] = ((characters[data])& 0x00ff);
    message_byte_count = 2;

    digit<<=1;

    LcdDrvSegWrite(&transmit_buffer[0],digit,message_byte_count);
}

//-------------------------------------------------------------------------------
// 模块:   LCD_string
// 描述:   向 LCD 模块发送字符串
//-------------------------------------------------------------------------------
__callt void LCD_string(unsigned char const *point, unsigned char dpos)
{
  while(dpos<=7)
  {
      if(point[0])
      {
          LCD_putc(dpos,*point-0x30);
          *point++;
      }
      else
      {
          LCD_putc(dpos,0xf0);
      }
      dpos++;
  }
}

//-------------------------------------------------------------------------------
// 模块:   LCD_string_shift
// 描述:   向 LCD 模块发送字符串
//-------------------------------------------------------------------------------
__callt void LCD_string_shift(unsigned char const *point)
{
  unsigned char dpos=7;
  while(dpos!=0xff)
  {
      if(sw3_in)
      {
        LCD_string(&s_clear[0],0);
        return;
      }
      LCD_string(point,dpos);
      dpos--;
      Wait(4);
  }
  *point++;
  dpos=0;
  while(point[0])
  {
      if(sw3_in)
      {
         LCD_string(&s_clear[0],0);
         return;
      }
      *point++;
```

```
      LCD_string(point,dpos);
      Wait(4);
  }
}
```

3.20 LcdDrvApp.h

```
/*****************************************************************************
;
;    NNNNNN          NN  EEEEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;    NNNNNNNN        NN  EEEEEE                 CCCCCC
;    NNNNNNNNNN      NN  EEEEEE                CCCCCC
;    NN  NNNNNNNN    NN  EEEEEEEEEEEEEEEEE    CCCCCC
;    NN    NNNNNNNN  NN  EEEEEE                CCCCCC
;    NN      NNNNNNNNNN  EEEEEE                 CCCCCC
;    NN          NNNNNN    EEEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;
;    NEC Electronics    78K0/Lx2
;
;*****************************************************************************
;   78K0/Lx2    LCD 控制示范程序
;*****************************************************************************
;   LCD 控制器/驱动器控制   -- 函数和常量定义文件
;*****************************************************************************
;[History]
;   2005.06.--  创建
;   2006.01.18 – 使用 Applilet 生成的 IIC 程序代替 iic.c
;              - 在 DemoKit-LG2 中的值 LCDC 由 0x0C 修改为 0x02
;*****************************************************************************/
/*===========================================================================
;   外部参考
;===========================================================================*/
#ifndef _LCDDRVAPP_H_
#define _LCDDRVAPP_H_

/*== 各种接口函数 ==*/
/* LCD 驱动器初始化处理  */
extern  unsigned char LcdDrvInit( void );
/* LCD 驱动器显示打开等待开始预处理
   (仅在内部步进模式选择时使用 */
extern  unsigned char LcdDrvOnWait( void );
/* LCD 驱动显示打开处理 */
extern  unsigned char LcdDrvOn( void );
/* LCD 驱动显示关闭处理 */
extern  unsigned char LcdDrvOff( void );
/* LCD 驱动控制寄存器写入处理 */
extern  unsigned char LcdDrvCtrWrite( unsigned char *src,
                                      unsigned char addr,
                                      unsigned char size );
/* LCD 驱动段数据写入处理 */
extern  unsigned char LcdDrvSegWrite( unsigned char *src,
                                      unsigned char addr,
                                      unsigned char size );
/* LCD 驱动段数据清楚处理 */
extern  unsigned char LcdDrvSegClr( void );
/* 在 LCD 驱动控制寄存器中单字节写入处理 */
extern  unsigned char LcdDrvCtrWrite1Byte( unsigned char addr,
                                           unsigned char data );
/* LCD 驱动段数据单字节写入处理 */
extern  unsigned char LcdDrvSegWrite1Byte( unsigned char addr,
                                           unsigned char data );
/*== 控制寄存器地址值 ==*/
```

```
#define CLDR_ADDR_LCDMD 0x00    /* 0x00: LCD 模式选择寄存器      */
#define CLDR_ADDR_LCDM  0x01    /* 0x01: LCD 显示模式寄存器     */
#define CLDR_ADDR_LCDC  0x02    /* 0x02: LCD 时钟控制寄存器    */
#define CLDR_ADDR_VLCG0 0x03    /* 0x03: LCD 步进控制寄存器 */

/*== 错误类型数据 ==*/
enum
{
    CLDR_ERR_None       /* 0: 没有错误                            */
,   CLDR_ERR_NACK       /* 1: 接收到 NACK                        */
,   CLDR_ERR_BUSY       /* 2: 繁忙(禁止通信)     */
,   CLDR_ERR_PARA       /* 3: 参数错误                    */
};

/*============================================================================
;   常量定义
;
;   在寄存器中设置用于控制 LCD 控制器/驱动器
;   以及主单元控制寄存器
;============================================================================*/
/*--------------------------------------------------------------------------
;   控制寄存器中的设置用于 LCD 片上的 LCD 控制单元
;--------------------------------------------------------------------------*/
/*
;[通信格式]
;
;   地址                    位
;              7       6       5       4       3       2       1       0
;              +-------+-------+-------+-------+-------+-------+-------+-------+
; 00H LCDMD    |SEGSET2|SEGSET1|SEGSET0|   0   |   0   |   0   |MDSET1 |MDSET0 |
;              +-------+-------+-------+-------+-------+-------+-------+-------+
;           MDSET1/0   : 选择 LCD 参考电压发生器
;           SEGSET2-0  : 设置段数(固定为 40)
;
;              +-------+-------+-------+-------+-------+-------+-------+-------+
; 01H LCDM     | LCDON | SCOC  | VLCON |   0   |   0   | LCDM2 | LCDM1 | LCDM0 |
;              +-------+-------+-------+-------+-------+-------+-------+-------+
;            LCDM2-0   : 选择 LCD 控制器/驱动器显示模式
;            VLCON     : 允许/禁止步进电路的操作
;            SCOC      : 控制段引脚/公共引脚的输出
;            LCDON     : 允许/禁止 LCD 显示
;
;              +-------+-------+-------+-------+-------+-------+-------+-------+
; 02H LCDC     |   0   |   0   |   0   |   0   | LCDC3 | LCDC2 | LCDC1 | LCDC0 |
;              +-------+-------+-------+-------+-------+-------+-------+-------+
;           LCDC1/0    : 设置 LCD 时钟
;           LCDC3/2    : 设置 LCD 源时钟
;
;              +-------+-------+-------+-------+-------+-------+-------+-------+
; 03H VLCG0    |CTSEL1 |CTSEL0 |   0   |   0   |   0   |   0   |   0   | GAIN  |
;              +-------+-------+-------+-------+-------+-------+-------+-------+
;           GAIN       : 设置参考电压等级
;           CTSEL1/0   : 选择对照调节器
;
;
; **从下列的试样中选择每一个寄存器的设置。
;
;============================================================================*/
/*--------------------------------------------------------------------------
;   LCD 模式选择寄存器(寄存器地址: 00H)
;--------------------------------------------------------------------------*/
/*--- 选择 LCD 参考电压发生器 ---*/
```

```
//  #define CLDR_LCDMD  0b00000000  /* <1> 外部分压电阻模式 */
//  #define CLDR_LCDMD  0b00000001  /* <2> 内部分压电阻模式 */
    #define CLDR_LCDMD  0b00000010  /* <3> 内部步进模式    */
/*            76543210                                    */
/*            -----000 ....... 0: 改为必须设置       */
/*            ||||||||                                    */
/*            ||||||||  **按照上面试样的<1>到<3>设置        */
/*            ||||||||                                                */
/*            ||||||++--    MDSET1/0    选择 LCD 参考电压发生器 */
/*            |||+++----    <固定为 000 >                             */
/*            +++-------    <固定为 000 > 选择段的数量 (SEGSET2/1/0) */

/*---------------------------------------------------------------------
;   LCD 显示模式寄存器(寄存器地址:  01H)
;---------------------------------------------------------------------*/
/*-- 选择 LCD 控制器/驱动器显示模式 --*/
/*                                                  */
/*                                        |       分压      |      步进      | */
/*                                        |     电阻模式    |       模式     | */
/*                                        |   分时  | 偏置 |   分时  |  偏置 | */
/*                                        |         |      |         |       | */
    #define CLDR_LCDM   0b11100000  /* <1> |   4     | 1/3  |   4     | 1/3   | */
//  #define CLDR_LCDM   0b11100001  /* <2> |   3     | 1/3  |   3     | 1/3   | */
//  #define CLDR_LCDM   0b11100010  /* <3> |   2     | 1/2  |   4     | 1/3   | */
//  #define CLDR_LCDM   0b11100011  /* <4> |   3     | 1/2  |   3     | 1/3   | */
//  #define CLDR_LCDM   0b11100100  /* <5> |     静态      |      禁止设置    | */
/*            76543210                                 */
/*            XXX--000...... 0: 改为必须设置, */
/*            ||||||||       X: 该位是否需要设置, 由软件控制*/
/*            ||||||||                                                */
/*            ||||||||  **按照上面试样的<1>到<5>设置          */
/*            ||||||||  **位 7 到 5 由软件控制               */
/*            ||||||||                                                */
/*            |||||+++--    LCDM2/1/0   选择 LCD 控制器/驱动器显示模式 */
/*            |||++-----    <固定为 00>                               */
/*            ||+-------    VLCON (固定为 1) 允许/禁止步进电路     */
/*            |+--------    SCOC  (固定为 1) 控制段/公共引脚输出 */
/*            +---------    LCDON (固定为 1) 允许/禁止 LCD 显示        */

/*---------------------------------------------------------------------
;   LCD 时钟控制寄存器(寄存器地址:  02H)
;---------------------------------------------------------------------*/
/*-- 选择 LCD 源时钟(fLCD) & LCD 时钟  --*/
/*                                                  */
/*                                        | LCD 源时钟  | LCD 时钟     | */
/*                                        |   (fLCD)    | LCD 时钟     | */
//  #define CLDR_LCDC   0b00000000  /* <1> | fPCL        | fLCD/2^6     | */
//  #define CLDR_LCDC   0b00000001  /* <2> | fPCL        | fLCD/2^7     | */
    #define CLDR_LCDC   0b00000010  /* <3> | fPCL        | fLCD/2^8     | */
//  #define CLDR_LCDC   0b00000011  /* <4> | fPCL        | fLCD/2^9     | */
//  #define CLDR_LCDC   0b00001000  /* <5> | fPCL/2      | fLCD/2^6     | */
//  #define CLDR_LCDC   0b00001001  /* <6> | fPCL/2      | fLCD/2^7     | */
//  #define CLDR_LCDC   0b00001010  /* <7> | fPCL/2      | fLCD/2^8     | */
//  #define CLDR_LCDC   0b00001011  /* <8> | fPCL/2      | fLCD/2^9     | */
//  #define CLDR_LCDC   0b00001100  /* <9> | fPCL/2^2    | fLCD/2^6     | */
//  #define CLDR_LCDC   0b00001101  /* <10>| fPCL/2^2    | fLCD/2^7     | */
//  #define CLDR_LCDC   0b00001110  /* <11>| fPCL/2^2    | fLCD/2^8     | */
//  #define CLDR_LCDC   0b00001111  /* <12>| fPCL/2^2    | fLCD/2^9     | */
/*            76543210                                                */
/*            ----0000 ...... 0: 该位必须被设置                      */
/*            ||||||||                                                */
```

```
/*            ||||||||  **按照上述的试样<1>到<12>设置                          */
/*            ||||||||                                                       */
/*            ||||||++--    LCDC1/0      设置 LCD 时钟                       */
/*            ||||++----    LCDC3/2      设置 LCD 源时钟                     */
/*            ++++------    <固定为 0000 >                                    */
/*-------------------------------------------------------------------------
;   LCD 步进控制寄存器(寄存器地址:03H)
;-------------------------------------------------------------------------*/
/*-- 选择参考电压(VLC2)等级&对照调节(TYP. value) -*/
/*                                                  */
/*                                        |   参考   |        对照            | */
/*                                        |   电压   |        调节            | */
/*                                        | (VLC2)  |     (TYP. value)       | */
/*                                        |level *1 |  VLC0 |  VLC1 |  VLC2  | */
    #define CLDR_VLCG0  0b10000000  /* <1> |1.5V     | 4.89V | 3.27V | 1.633V | */
//  #define CLDR_VLCG0  0b11000000  /* <2> |1.5V     | 4.71V | 3.13V | 1.567V | */
//  #define CLDR_VLCG0  0b00000000  /* <3> |1.5V     | 4.50V | 3.00V | 1.500V | */
//  #define CLDR_VLCG0  0b01000000  /* <4> |1.5V     | 4.29V | 2.87V | 1.433V | */
//  #define CLDR_VLCG0  0b10000001  /* <5> |1.0V     | 3.29V | 2.27V | 1.133V | */
//  #define CLDR_VLCG0  0b11000001  /* <6> |1.0V     | 3.21V | 2.13V | 1.067V | */
//  #define CLDR_VLCG0  0b00000001  /* <7> |1.0V     | 3.00V | 2.00V | 1.000V | */
//  #define CLDR_VLCG0  0b01000001  /* <8> |1.0V     | 2.79V | 1.87V | 0.933V | */
/*            76543210                                                       */
/*            00-----0 ...... 0: 该位必须设置                                */
/*            ||||||||                                                       */
/*            ||||||||  **按照上面的试样<1>到<8>设置                         */
/*            ||||||||                                                       */
/*            |||||||+--    GAIN          选择参考电压等级                   */
/*            ||+++++---    <固定为 00000 >                                  */
/*            ++--------    CTSEL1/0     选择对照调整                        */
/*                                                                           */
/*  *1: 当目标 LCD 面板为 4.5V 时，选择参考电压等级为 1.5 V。                */
/*      当目标 LCD 面板为 3.0V 时，选择参考电压等级为 1.0 V。                */
/*=========================================================================
;   主控制寄存器设置的定义
;=========================================================================*/
/*-------------------------------------------------------------------------
;    选择时钟输出
;-------------------------------------------------------------------------*/
/*-- 选择 PCL 的输出时钟 --*/
/*   fSUB=32.768kHz               */
/*   fPRS=外部时钟         */
/*                                                  | fPRS=10MHz  | fPRS=20MHz | */
//  #define CLDR_CKS    0b00000110  /* <1> | fPRS/2^6 | 156.25kHz   | 312.5 kHz  | */
//  #define CLDR_CKS    0b00000111  /* <2> | fPRS/2^7 |  78.125kHz  | 156.25kHz  | */
    #define CLDR_CKS    0b00001000  /* <3> | fSUB     |  32.768kHz  |  32.768KHz | */
/*            76543210                                                       */
/*            ---X0000 ...... 0: 改为必须设置                                */
/*            ||||||||        X: 是否需要设置该位由软件控制                  */
/*            ||||||||                                                       */
/*            ||||||||  **按照上述式样<1>到<3>设置                           */
/*            ||||||||  **位 4 通过软件控制                                  */
/*            ||||||||                                                       */
/*            ||||++++--    CCS3/2/1/0  选择 PCL 的时钟输出                  */
/*            |||+------    CLOE (固定为 0)  允许/禁止时钟输出到 LCD         */
/*            +++-------    <固定为 000 >                                    */
/*-----------------------------------<文件完毕>----------*/
#endif /* _LCDDRVAPP_H */
```

3.21 LcdDrvApp.c

```
/*****************************************************************************
;
;    NNNNNN          NN  EEEEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;    NNNNNNNN        NN  EEEEEE                 CCCCCC
;    NNNNNNNNNN      NN  EEEEEE                CCCCCC
;    NN  NNNNNNNN    NN  EEEEEEEEEEEEEEEEE    CCCCCC
;    NN    NNNNNNNN  NN  EEEEEE                CCCCCC
;    NN      NNNNNNNNNN  EEEEEE                 CCCCCC
;    NN          NNNNNN    EEEEEEEEEEEEEEEE     CCCCCCCCCCCCCCC
;
;    NEC 电子 78K0/Lx2
;
;*****************************************************************************
;   78K0/Lx2    LCD 控制示样程序
;*****************************************************************************
;   LCD 控制器/驱动器控制    --处理文件--
;*****************************************************************************
;[历史]
;   2005.06.--  创建文件
;   2005.07.01  [050701] 临时添加电源到 VLC0
;   2006.01.11  使用 Applilet 生成的 IIC 程序-rdh
;   2006.01.27  在 LcdDrvOff 中修改关闭 VLCON - rdh
;   2006.03.30  在应用笔记中使用程序仅用于写入
*****************************************************************************/
#pragma sfr

/*============================================================================
;   包含
;============================================================================*/
#include    "macrodriver.h"
#include    "serial.h"
#include    "LcdDrvApp.h"

static void LcdDrvClkOut( void );
static void LcdDrvClkStop( void );

/*============================================================================
;   LCD 驱动和变量设置的控制区域
;============================================================================*/
/*===========
;   从设备 ID
;==========*/
#define     CSLV_ID_LCDCTL  0b01110000      /* 控制寄存器(LCDCTL) */
#define     CSLV_ID_LCDSEG  0b01110010      /* 段数据(LCDSEG)  */

/*===========
;   在主要方定义控制寄存器设置
;==========*/
/*-- 时钟输出选择寄存器 --*/
#define     LDR_CKS     CKS
#define    LDR_CKS_CLOE    LDR_CKS.4       /* 时钟输出允许/禁止 */

/*-- 端口/端口模式寄存器 (与微控制器直接相连) --*/
#define     PO_LDR_RST  P13.0           /* 复位 LCD            */
#define     PM_LDR_OUT  PM14.0          /* 时钟输出到 LCD     */

/*-- 端口/端口模式寄存器 --*/                          /*[050701]>>*/
#define     PO_VLC0_HL  P7.7            /* 到 VLC0 的电压 */
#define     PM_VLC0_HL  PM7.7           /* 到 LLC0 的电压 [050701] */

/*===========
```

```
   控制寄存器设置
==========*/
/* 选择 LCD 参考电压发生器: 内部步进模式 */
#define     CLDR_LCDMD_VOL  0b00000010

/*****************************************************************
;   LCD 驱动器初始化
;-----------------------------------------------------------------
;   [I N]   -
;   [OUT]   0= 设置 OK, 1 = 接收到 NACK, 2 = 繁忙
;*****************************************************************/
unsigned char LcdDrvInit( void )
{
    register unsigned char result = CLDR_ERR_None;

    PO_LDR_RST = 1;                 /* 取消 LCD 复位状态  */
    LDR_CKS = ( CLDR_CKS & 0b00001111);  /* 输出时钟设置(CCS3-0) */

    /*-- 允许时钟输出到 LCD  --*/
    LcdDrvClkOut();

    /*--  选择参考电压发生器 --*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDMD, CLDR_LCDMD );

    /*-- 清除段数据  --*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvSegClr();
    }

    /*-- 选择显示模式 --*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00000111 ));
    }

    /*-- LCD 时钟设置 --*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDC, CLDR_LCDC );
    }
    return  ( result );
}

/*****************************************************************
; LCD 驱动显示打开等待启动预处理 (当选择步进模式时)
;-----------------------------------------------------------------
;   <<注>>
;
; 当参考电压发生器选择不仅模式时仅调用 N。在该处理调用后需要等待 500ms，然后调用"LcdDrvOn"。
;
;-----------------------------------------------------------------
;   [输入]   -
;   [输出]   0= 设置 OK, 1 = 接收到 NACK, 2 = 繁忙
;*****************************************************************/
unsigned char LcdDrvOnWait( void )
{
#if (CLDR_LCDMD==CLDR_LCDMD_VOL)
    /*  参考电压发射器:  内部步进模式 */
    register unsigned char result = CLDR_ERR_None;

    /*-- 允许时钟输出到 LCD  --*/
    LcdDrvClkOut();

    /*-- 设置 LCD 步进级别和参照 --*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_VLCG0, CLDR_VLCG0 );
```

```
    /*-- 允许 LCD 步进  --*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00100111 ));
    }
    /* -- 在 500 毫秒后调用 "LcdDrvOnWait"函数 -- */
    return  ( result );
#else/*!(CLDR_LCDMD==CLDR_LCDMD_VOL)*/
    /*  参考电压发射器:  分压电阻模式 */
    return  ( 0 );
#endif/*(CLDR_LCDMD)*/
}

/*****************************************************************
;   LCD 驱动打开处理
;-----------------------------------------------------------------
;   <<注>>
;
; 当参考电压发生器选择内部步进模式时，在调用"LcdDrvOnWait"后，必须等待 500 毫秒
; 之后才能调用下一个函数。
;
;-----------------------------------------------------------------
;   [输入]   -
;   [输出]   0= 设置 OK, 1 = 接收到 NACK, 2 = 繁忙
;*****************************************************************/
unsigned char LcdDrvOn( void )
{
    register unsigned char result = CLDR_ERR_None;

#if (CLDR_LCDMD==CLDR_LCDMD_VOL)
    /*  参考电压发生器: 内部步进模式 */
    /*-- 设置取消电势输出 --*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b01100111 ));

    /*-- 显示打开设置 --*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b11100111 ));
    }

#else/*!(CLDR_LCDMD==CLDR_LCDMD_VOL)*/
    /*  参考电压发生器: 分压电阻模式 */
    /*-- 允许时钟输出到 LCD  --*/
    LcdDrvClkOut();

    /*-- 设置取消电势输出 --*/
    result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b01000111 ));

    /*-- 显示打开设置 --*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b11000111 ));
    }

#endif/*(CLDR_LCDMD)*/
    return  ( result );
}

/*****************************************************************
;   LCD 驱动显示关闭处理
;-----------------------------------------------------------------
;   [输入]   -
;   [输出]   0= 设置 OK, 1 = 接收到 NACK, 2 = 繁忙
;
;   *清除 AX 寄存器
;*****************************************************************/
```

```
unsigned char LcdDrvOff( void )
{
    register unsigned char result = CLDR_ERR_None;

    /*-- 清除段数据  --*/
    result = LcdDrvSegClr();

    /*-- 显示关闭设置 --*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b01100111 ));
    }

    /*-- 段/公共缓存输出禁止设置--*/
    if( result == CLDR_ERR_None ){
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00100111 ));
    }

#if (CLDR_LCDMD==CLDR_LCDMD_VOL)

    /*-- LCD 步进禁止设置 --*/
    if( result == CLDR_ERR_None ){
#if 0   /* 改正 060127 –位 5 为 1, 不关闭 VLCON */
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00100111 ));
#else   /* 通过将位 5 清 0 的方式改正关闭 VLCON */
        result = LcdDrvCtrWrite1Byte( CLDR_ADDR_LCDM, ( CLDR_LCDM & 0b00000111 ));
#endif
    }

#endif/*(CLDR_LCDMD)*/

    if( result == CLDR_ERR_None ){
        /*-- 禁止时钟输出到 LCD --*/
        LcdDrvClkStop();
    }
    return  ( result );
}

/*****************************************************************
;   写入 LCD 驱动控制数据
;----------------------------------------------------------------
; [输入]   src : 控制寄存器数据的存储地址
;     addr    : 控制寄存器地址数值
;     size    : 要被发送字节的数目 (最大 4 字节)
; [输出]   0= 设置 OK, 1 = 接收到 NACK, 2 = 繁忙, 3 = 参数错误
;*****************************************************************/
unsigned char SLDRCTLW( unsigned char *src,
                        unsigned char addr, unsigned char size )
{
    register unsigned char result = CLDR_ERR_None;
    register unsigned char cnt;
    MD_STATUS status;
    unsigned char buf[5];
    unsigned char uc;

    /*-- 允许时钟输出到 LCD --*/
    LcdDrvClkOut();

    /*-- 检测参数   --*/
    if(( size == 0 )||( addr > 0x03 )||(( 0x03+1 - addr ) < size )){
        result = CLDR_ERR_PARA;
    }
    buf[0] = addr;
    for (uc = 0; uc < size; uc++) {
        buf[uc+1] = src[uc];
```

```
    }

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDCTL, buf, size + 1 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*****************************************************************
;   写入 LCD 驱动段数据
;-----------------------------------------------------------------
; [输入]   src : 要被写入的段数据地址
;      addr    : 开始写入操作的数据地址
;      size    : 要被发送的字节数(最大 20 字节)
; [输出]   0= 设置 OK, 1 = 接收到 NACK, 2 = 繁忙, 3 = 参数错误
;-----------------------------------------------------------------
;   ** 在发送段数据之前要进行平滑处理
;
;   <接收数据>
;                     位 7  位 6  位 5  位 4  位 3  位 2  位 1  位 0
;                    +----+
;      第一个字节 (00H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;                    |<-     段:S1      ->|<-    段:S0       ->|
;
;                    +----+----+----+----+----+----+----+----+
;     第二个字节 (01H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;         :            :
;                    +----+----+----+----+----+----+----+----+
;    第十九个字节 (12H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;                    +----+----+----+----+----+----+----+----+
;    第二十个字节 (13H)|COM3|COM2|COM1|COM0|COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;                    |<-     段:S39     ->|<-    段:S38     ->|
;
;   <发送数据>
;                     位 7  位 6  位 5  位 4  位 3  位 2  位 1  位 0
;                    +----+----+----+----+----+----+----+----+
;       第一个字节(00H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;                                        |<-     段:S0      ->|
;                                               地址:00H     |
;
;                    +----+----+----+----+----+----+----+----+
;      第二个字节 (01H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;                      :
;                    +----+----+----+----+----+----+----+----+
;   第三十九个字节(26H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;                    +----+----+----+----+----+----+----+----+
;    第四十个字节 (27H)|  0 |  0 |  0 |  0 |COM3|COM2|COM1|COM0|
;                    +----+----+----+----+----+----+----+----+
;                                        |<-  段:S39   ->|
;                                        |    地址:27H    |
;
;*****************************************************************/
unsigned char LcdDrvSegWrite( unsigned char *src,
                              unsigned char addr, unsigned char size )
{
    register unsigned char result = CLDR_ERR_None;
```

```
    register unsigned char cnt;
    MD_STATUS status;
    unsigned char buf[41];
    unsigned char uci,ucd;

    /*-- 允许时钟输出到LCD --*/
    LcdDrvClkOut();

    /*-- 检测参数   --*/
    if(( size == 0 )||( addr > 0x13 )||(( 0x13+1 - addr ) < size )){
        result = CLDR_ERR_PARA;
    }

    buf[0] = addr*2;
    for (uci = 0, ucd = 1; uci < size; uci++, ucd = ucd + 2) {
        buf[ucd]   = src[uci] & 0x0f ;
        buf[ucd+1] = (src[uci] >> 4) & 0x0f;
    }

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDSEG, buf, (size * 2) + 1 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*********************************************************************
;   清除LCD驱动段数据
;---------------------------------------------------------------------
; [输入]   -
; [输出]   0= 设置OK, 1 = 接收到NACK, 2 = 繁忙
;*********************************************************************/
unsigned char LcdDrvSegClr( void )
{
    register unsigned char result = CLDR_ERR_None;
    register unsigned char cnt;
    MD_STATUS status;
    unsigned char buf[41];
    unsigned char uc;

    /*-- 允许时钟输出到LCD --*/
    LcdDrvClkOut();

    buf[0] = 0; // 在0位置开始
    for (uc = 1; uc < 41; uc++) {
        buf[uc] = 0;
    }

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDSEG, buf, 41 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*********************************************************************
;   写入LCD驱动控制寄存器(1字节设置)
;---------------------------------------------------------------------
; [输入]   addr    : 控制寄存器地址值
;          data    : 控制寄存器数据
; [输出]   0= 设置OK, 1 = 接收到NACK, 2 = 繁忙, 3 = 参数错误
;*********************************************************************/
unsigned char LcdDrvCtrWrite1Byte( unsigned char addr, unsigned char data )
{
    register unsigned char result = CLDR_ERR_None;
    MD_STATUS status;
    unsigned char buf[2];
```

```
    unsigned char uc;

    /*-- 允许时钟输出到LCD --*/
    LcdDrvClkOut();

    buf[0] = addr;
    buf[1] = data;

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDCTL, buf, 2 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*********************************************************************
;   写入LCD驱动段数据 (1字节设置)
;---------------------------------------------------------------------
; [输入]   addr    : 控制寄存器地址数据
;          data    : 控制寄存器数据
; [输出]   0= 设置OK, 1 = 接收到NACK, 2 = 繁忙, 3 = 参数错误
;*********************************************************************/
unsigned char LcdDrvSegWrite1Byte( unsigned char addr, unsigned char data )
{
    register unsigned char result = CLDR_ERR_None;
    register unsigned char work;
    MD_STATUS status;
    unsigned char buf[5];
    unsigned char uc;

    /*-- 允许时钟输出到LCD --*/
    LcdDrvClkOut();

    buf[0] = addr;
    buf[1] = data & 0x0f;
    buf[2] = ( data >>4 ) & 0x0f;

    status = IIC0_MasterStartAndSend( CSLV_ID_LCDSEG, buf, 3 );
    if (status != MD_OK)
        result = CLDR_ERR_NACK;
    return  ( result );
}

/*********************************************************************
;   允许时钟和电源输出到LCD
;---------------------------------------------------------------------
; [I N]   -
; [OUT]   -
;*********************************************************************/
static void LcdDrvClkOut( void )
{
    PM_LDR_OUT   = 0;
    LDR_CKS_CLOE = 1;
#if (CLDR_LCDMD==CLDR_LCDMD_VOL)                        /*[050701]>>*/
    PO_VLC0_HL   = 0;   /* 低输出(电源提供到VLC0)*/
#else /*!(CLDR_LCDMD==CLDR_LCDMD_VOL)*/
    PO_VLC0_HL   = 1;   /* 高输出(电源未提供到VLC0)*/
#endif/*(CLDR_LCDMD)*/                                  /*[050701]<<*/
}

/*********************************************************************
;   禁止时钟和电源输出到LCD
;---------------------------------------------------------------------
; [输入]   -
; [输出]   -
;*********************************************************************/
```

```
static void LcdDrvClkStop( void )
{
    LDR_CKS_CLOE = 0;
    PM_LDR_OUT   = 1;
    PO_VLC0_HL   = 0;                                   /*[050701]*/
}

/*-------------------------------------------------<文件结束>---*/
```