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

使用 8 位和 16 位定时器

文档编号：U18269CA1V0AN00
发布日期： 2007 年 8 月

德国印制

CMOS设备的注释

① 输入引脚处的电压波形

输入噪音或一个反射波引起的波形失真可能导致错误发生。如果由于噪音等的影响使CMOS设备的输入电压范围保持在VIL(MAX)和VIH(MIN)之间，设备可能发生错误。在输入电平固定时以及输入电平从VIL (MAX)过渡到VIH (MIN)时的传输期间，要防止散射噪声影响设备。

② 未使用的输入引脚的处理

CMOS设备的输入端保持开路可能导致误操作。如果一个输入引脚未被连接，则由于噪音等原因可能会产生内部输入电平，从而导致误操作。CMOS设备的操作特性与Bipolar或NMOS设备不同。CMOS设备的输入电平必须借助上拉或下拉电路固定在高电平或低电平。每一个未使用引脚都应该通过附加电阻连接到VDD或GND。如果有可能尽量定义为输出引脚。对未使用引脚的处理因设备而异，必须遵循与设备相关的规定和说明。

③ ESD防护措施

如果MOS设备周围有强电场，将会击穿氧化栅极，从而影响设备的运行。因此必须采取措施，尽可能防止静电产生。一旦有静电，必须立即释放。对于环境必须有适当的控制。如果空气干燥，应当使用增湿器。建议避免使用容易产生静电的绝缘体。半导体设备的存放和运输必须使用抗静电容器、抗静电屏蔽袋或导电材料容器。所有的测试和测量工具包括工作台和工作面必须良好接地。操作员应当佩戴静电消除手带以保证良好接地。不能用手直接接触半导体设备。对于装配有半导体设备的PW板也应采取类似的静电防范措施。

④ 初始化之前的状态

在上电时MOS设备的初始状态是不确定的。在刚刚上电之后，具有复位功能的MOS设备并没有被初始化。因此上电不能保证输出引脚的电平，I/O设置和寄存器的内容。设备在收到复位信号后才进行初始化。具有复位功能的设备在上电后必须立即进行复位操作。

⑤ 电源开关顺序

在一个设备的内部操作和外部接口使用不同的电源的情况下，按照规定，应先在接通内部电源之后再接通外部电源。当关闭电源时，按照规定，先关闭外部电源再关闭内部电源。如果电源开关顺序颠倒，可能会导致设备的内部组件过电压，产生异常电流，从而引起内部组件的误操作和性能的退化。

对于每个设备电源的正确开关顺序必须依据设备的规范说明分别进行判断。

⑥ 电源关闭状态下的输入信号

不要向没有加电的设备输入信号或提供I/O上拉电源。因为输入信号或提供I/O上拉电源将引起电流注入，从而引起设备的误操作，并产生异常电流，从而使内部组件退化。

每个设备电源关闭时的信号输入必须依据设备的规范说明分别进行判断

详细信息请联系：

（中国区）

网址：
http://www.cn.necel.com/
http://www.necel.com/

[北京]

日电电子（中国）有限公司
中国北京市海淀区知春路 27 号
量子芯座 7，8，9，15 层
电话：（+86）10-8235-1155
传真：（+86）10-8235-7679

[上海]

日电电子（中国）有限公司上海分公司
中国上海市浦东新区银城中路 200 号
中银大厦 2409-2412 和 2509-2510 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

上海恩益禧电子国际贸易有限公司
中国上海市浦东新区银城中路 200 号
中银大厦 2511-2512 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

[深圳]

日电电子（中国）有限公司深圳分公司
深圳市福田区益田路卓越时代广场大厦 39 楼
3901，3902，3909 室
电话：（+86）755-8282-9800
传真：（+86）755-8282-9899

[香港]

香港日电电子有限公司
香港九龙旺角太子道西 193 号新世纪广场
第 2 座 16 楼 1601-1613 室
电话：（+852）2886-9318
传真：（+852）2886-9022
2886-9044

- 本文档信息于 2006 年 5 月开始使用。将来可能未经预先通知而更改。在实际进行生产设计时，请参阅各产品最新的数据表或数据手册等相关资料以获取本公司产品的最新规格。
- 并非所有的产品和/或型号都向每个国家供应。请向本公司销售代表查询产品供应及其他信息。
- 未经本公司事先书面许可，禁止复制或转载本文件中的内容。本文件所登载内容的错误，本公司概不负责。
- 本公司对于因使用本文件中列明的本公司产品而引起的，对第三者的专利、版权以及其它知识产权的侵权行为概不负责。本文件登载的内容不应视为本公司对本公司或其他人所有的专利、版权以及其它知识产权作出任何明示或默示的许可及授权。
- 本文件中的电路、软件以及相关信息仅用以说明半导体产品的运作和应用实例。用户如在设备设计中应用本文件中的电路、软件以及相关信息，应自行负责。对于用户或其他人因使用了上述电路、软件以及相关信息而引起的任何损失，本公司概不负责。
- 虽然本公司致力于提高半导体产品的质量及可靠性，但用户应同意并知晓，我们仍然无法完全消除出现产品缺陷的可能。为了最大限度地减少因本公司半导体产品故障而引起的对人身、财产造成损害（包括死亡）的危险，用户务必在其设计中采用必要的安全措施，如冗余度、防火和防故障等安全设计。
- 本公司产品质量分为：

  “标准等级”、“专业等级”以及“特殊等级”三种质量等级。

  “特殊等级”仅适用于为特定用途而根据用户指定的质量保证程序所开发的日电电子产品。另外，各种日电电子产品的推荐用途取决于其质量等级，详见如下。用户在选用本公司的产品时，请事先确认产品的质量等级。

  “标准等级”： 计算机，办公自动化设备，通信设备，测试和测量设备，音频・视频设备，家电，加工机械以及产业用机器人。

  “专业等级”： 运输设备（汽车、火车、船舶等），交通用信号控制设备，防灾装置，防止犯罪装置，各种安全装置以及医疗设备（不包括专门为维持生命而设计的设备）。

  “特殊等级： 航空器械，宇航设备，海底中继设备，原子能控制系统，为了维持生命的医疗设备、用于维持生命的装置或系统等。

  除在本公司半导体产品的数据表或数据手册等资料中另有特别规定以外，本公司半导体产品的质量等级均为“标准等级”。如果用户希望在本公司设计意图以外使用本公司半导体产品，务必事先与本公司销售代表联系以确认本公司是否同意为该项应用提供支持。

（注）

（1） 本声明中的“本公司”是指日本电气电子株式会社（NEC Electronics Corporation）及其控股公司。

（2） 本声明中的“本公司产品”是指所有由日本电气电子株式会社或为日本电气电子株式会社（定义如上）开发或制造的产品。

目录:

介绍

本文档的目的是为 NEC 微控制器的外设提供一些简单的例子。用户可以因此更好的理解对这些外设的使用。

文档的格式包括以下内容：

- 外设性能的描述
- 例子程序描述和说明
- 软件流程图
- Applilet 参考驱动程序
- 演示平台的使用
- 硬件框图
- 软件模块

Applilet 是外设驱动代码生成工具。它可以方便的为片上外设提供代码。

用户如需了解细节请参考外设的用户手册和其他相关文档。

8 位和 16 位定时器综述

NEC 系列微控制器大部分都包括 8 位和/或 16 位定时器。很多设备有多个定时器。定时器用作基本测量或计数。大多数的定时器有复用功能和为各种不同的需求配置使用。另外，所有的定时器都可以灵活地进行时钟选择，允许不同的时钟源和时钟分频。

NEC 微控制器的典型定时器，有以下功能。

- 间隔定时器
- 外部事件计数器
- 方波发生器
- 单脉冲发生器
- 可编程脉冲发生器(PPG)
- 可编程脉冲宽度调制器(PWM)
- 脉冲宽度测量

操作控制寄存器可用作设定定时器操作和输出模式。时钟选择寄存器选择定时器操作的基本时钟。一系列时钟选择可分为主时钟、主时钟分频、副时钟和外部时钟输入。定时器可操作在自由跑模式、重复改变输出或中断时、启动或停止间隔信号或脉冲定时。

间隔定时器在预置的时间间隔产生中断信号。当计数器的值和预置的比较寄存器的值相等时产生中断。这个功能可用于检查状态或按预置的间隔执行特定的任务。

外部事件计数器模式，外部事件输入到外部计数器。外部计数器比较预先设定的外部监视器的值。当外部计数值与外部监视器的值相等时产生中断。这个特点可用作外部事件监视器和当检测到预先设定的外部事件执行特定的任务。

自由跑模式可编程创建各种形状和持续时间的输出脉冲。由相等周期的高低输出产生方波。由设置特殊频率 PWM (脉冲宽度调制) 输出 on-off 周期, 并改变 on-time 相对 off-time 占空比周期。

PPG (可编程脉冲发生) 模式可用作脉冲产生，可用于创造自定义脉冲，改变脉冲频率和高/低占空比。

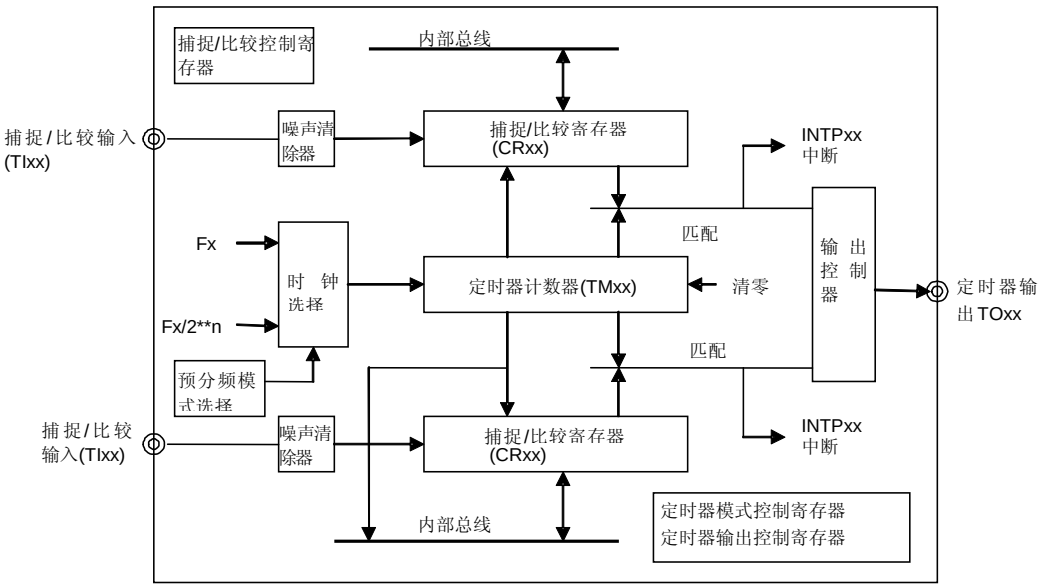
单脉冲发生用于创建自定义可变宽度和启动时间延时的信号脉冲，基于外部触发或软件触发。选择 NEC 微控制器，结合定时器可产生应用于遥控的载波脉冲波形。

由 8 位和 16 位定时器不同特征的示例，如下。

- 用 16 位定时器 00(TM00)的间隔定时器，按键噪声去抖动 (噪声消除)
- 用 16 位定时器 00(TM00)外部事件计数器，8 位定时器 50(TM50)作为方波发生器
- 用 16 位定时器 01(TM01)单脉冲功能产生脉冲，由定时器 00 作为外部事件计数器
- 用 8 位定时器 51(TM51) 的 PWM 输出产生模拟电压(低功耗 D/A 转换)
- 用 8 位定时器 51 (TM51) 和 8 位定时器 H1 (TMH1) 结合产生载波输出，连接到发光二极管作为遥控数据传输

- PPG 和方波输出 使用定时器 00 (TM00) 连接到扬声器产生变量音调
- 时间保持使用 32.768 KHz 晶振作为第二基础时间，使用钟表定时器 (WTM) 创建周期中断

NEC 78K0 系列的定时器包括以下功能模块。分类描述为典型定时器；特殊定时器，或增加专门的模块用于特殊功能，例如载波发生。



操作寄存器:

寄存器	名称	功能描述
定时器/计数器	TMxx	定时器/计数器与计数时钟同步增加
		定时器/计数器是只读寄存器
捕捉/比较寄存器	CRxx/CRyy	这些寄存器可用作捕捉或比较寄存器
	比较	寄存器的值常与定时器/计数器比较
		当匹配时，产生中断
	捕捉	定时器/计数器的值被捕捉/比较寄存器捕捉
		TIxx, TIyy 用作捕捉触发输入
捕捉触发沿输入	TIxx/TIyy	外部输入到捕捉定时器/计数器的值进入捕捉寄存器
		当捕捉时，它清零定时器/计数器(依靠模式设置)
		定时器的值在上升沿、下降沿或同时触发捕捉

控制寄存器:

寄存器	名称	功能描述
模式控制寄存器	TMCxx	允许/禁止定时器/计数器
		操作模式- 自由跑, 清零重启模式
		定时器/计数器溢出标志
捕捉/比较控制寄存器	CRCxx	选择比较操作或捕捉操作
		捕捉触发沿选择
输出控制寄存器	TOCxx	允许/禁止输出，倒置输出，初值输出
预分频模式控制寄存器	PRMxx	时钟选择。捕捉触发有效沿选择
端口模式寄存器	PMx	Tixx 输入和 Toxx 输出的输入或输出模式选择

1 间隔定时器使用16位定时器00 (TM00)

大多数 8 位或 16 位 NEC 78K0 系列微控制器都支持间隔定时器功能。它用作测量时间间隔并且根据时间进行动作。

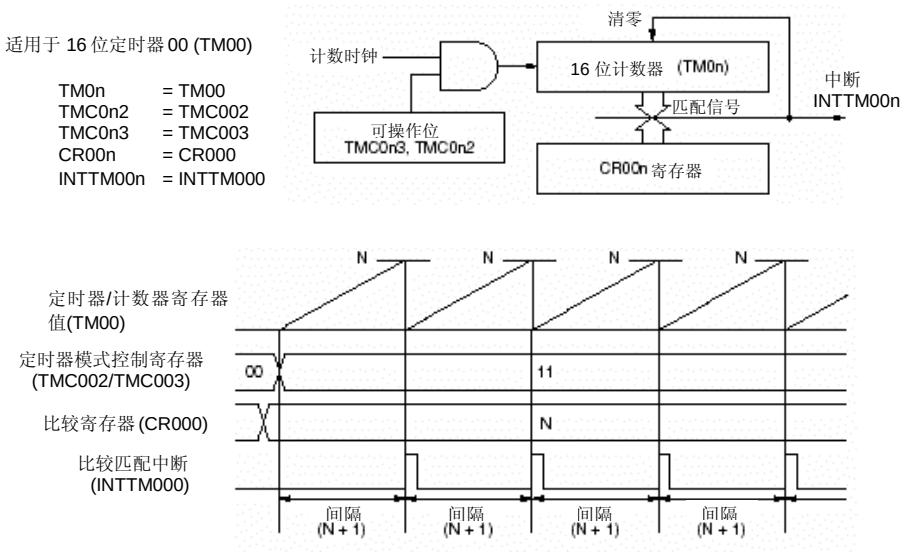
1.1 16 位间隔定时器特征

在间隔定时器模式，16 位定时器提供以下特征：

- 基于外围设备时钟分频的可变时间，使用 PRM00 寄存器
- 可变间隔时间，使用 CR000 计数基础时间时钟
- 比较 TM00 和 CR000 可选择中断

如果定时器/计数器模式选择寄存器设置为"清零&启动模式"，定时器/计数器将在定时器/计数器(TM00)寄存器和比较寄存器(CR000)相等时，清零并重启计数时钟的同步计数。当定时器/计数器寄存器(TM00)和比较寄存器(CR000)相等时，定时器/计数器清零并产生匹配中断。

注：以下框图被用作定时器 00 和定时器 01，并显示了寄存器名称。



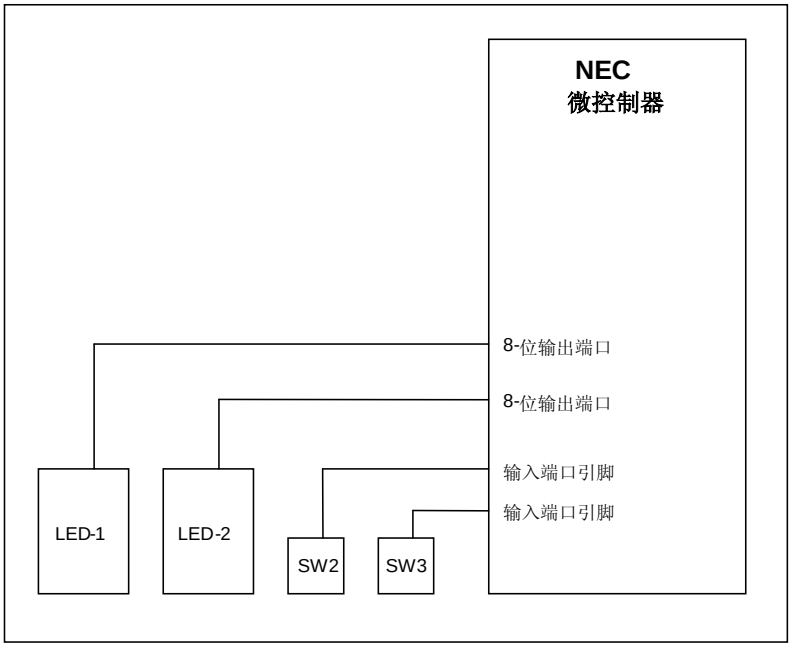
- 定时器/计数器模式寄存器 = 11　　清零&启动模式
- 比较寄存器值 = N　　当计数为 N，发生匹配
- 比较匹配中断　　当发生匹配，产生中断

定时器用作间隔定时器的操作配置，按以下步骤：

- 设置预分频模式寄存器 PRM00，在 CR000 设置间隔时间
- 如要使用中断，要设置中断的优先级，清中断标志，非屏蔽中断
- 由设置定时器模式寄存器 TMC00 清零&启动模式允许定时器操作，比较 TM00 和 CR000

1.2 程序描述和说明

演示程序使用间隔定时器来按键输入消除噪声。当第一个按键按下，计数器的值递减，按键去抖动，并且在 LED 上显示按下的按键次数。当第二个按键按下，计数值递增，按键去抖动，并在 LED 上显示。



按键去抖动由按键输入取样完成，在预置的时间间隔。取样周期是程序使用间隔定时器。如果定时器/计数器配置为"清零-启动模式"，并在比较寄存器设置了期望值，则每次定时器寄存器和比较寄存器匹配时，将产生中断。然后，定时器寄存器清零并以选择的时钟同步启动计数。

在预置的时间间隔，当定时器寄存器和比较寄存器匹配时，按键状态取样。如果按键按下在预置的次数并在间隔内不改变状态，就认为是"按键按下"。取样周期由用户编程。如果，在取样周期内按键改变状态，将认为是按键噪声，并不认为是按键输入。

按键连接到 78K0 微控制器，输入状态可被取样。LED 连接到 8 位端口用于显示按下的按键。

- 初始化　　设置定时器/计数器模式，比较寄存器值，并设置端口模式
- 按下 SW2　　在 LED 上显示按下 SW2 次数 (减少)
- 按下 SW3　　在 LED 上显示按下 SW3 次数 (增加)

说明：

- 设置定时器 00 每 1 ms 提供周期中断
- 按键必须稳定连续输入(10 ms)以去抖动

1.3 软件流程图

演示程序由以下主要几部分组成：

- 用于程序的初始化代码，在 main() 程序启动前调用
- 主程序循环，检查按键状态和在 LED 显示计数
- 由 Applilet 产生的子程序处理启动定时器 00，停止和改变间隔
- 用户代码子程序用于处理定时器 00 中断(Applilet 产生框架中断服务程序，增加用户代码)
  注 TM50 不产生中断
- 用于按键输入，LED 显示输出的子程序

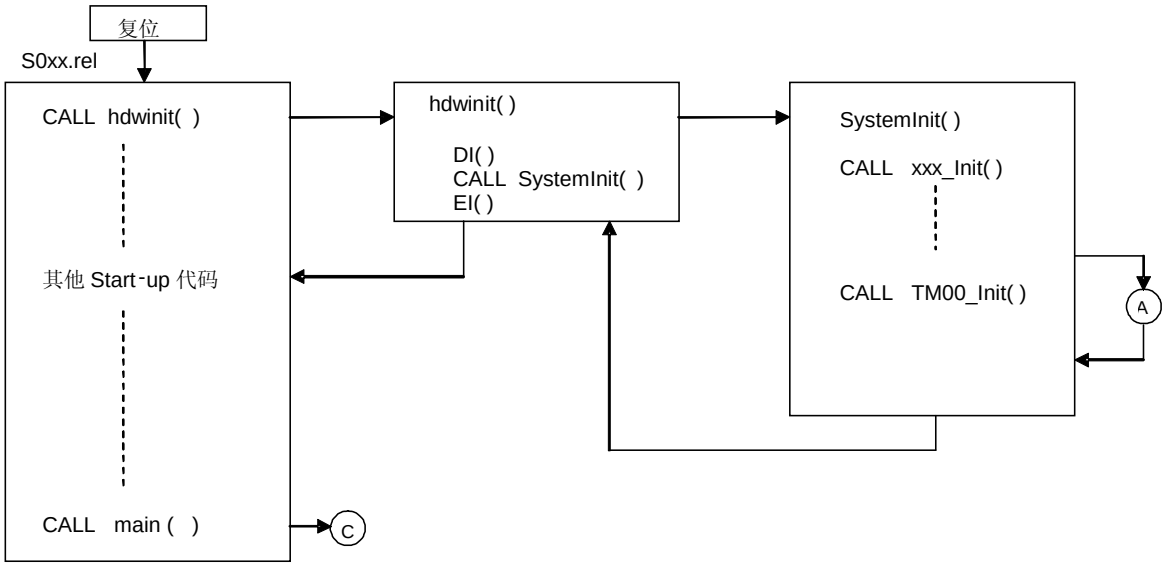
这里的流程图描述初始化、主程序、定时器 00 相关子程序 和中断服务程序。

1.3.1 程序启动和初始化

适用于 78K0 的 C 语言程序，C 语言启动代码由目标代码文件提供，例如 s01.rel，联接到用户程序。启动代码调用 hdwinit( )函数，用户可以在此放置硬件初始化代码。

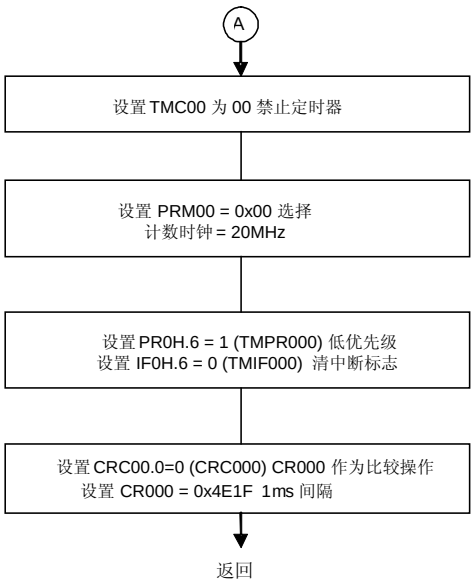
当使用 Applilet 产生 78K0 C 语言，hdwinit( ) 函数自动增加到用户程序，并调用 SystemInit( )函数。SystemInit( ) 函数轮流调用各外设的初始化程序。

流程图 – 程序启动



hdwinit( ) 函数完成后，启动代码调用用户程序 main( ) 函数。启动 main( ) 函数，执行所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

1.3.2 TM00_Init( ) – 定时器 00 初始化



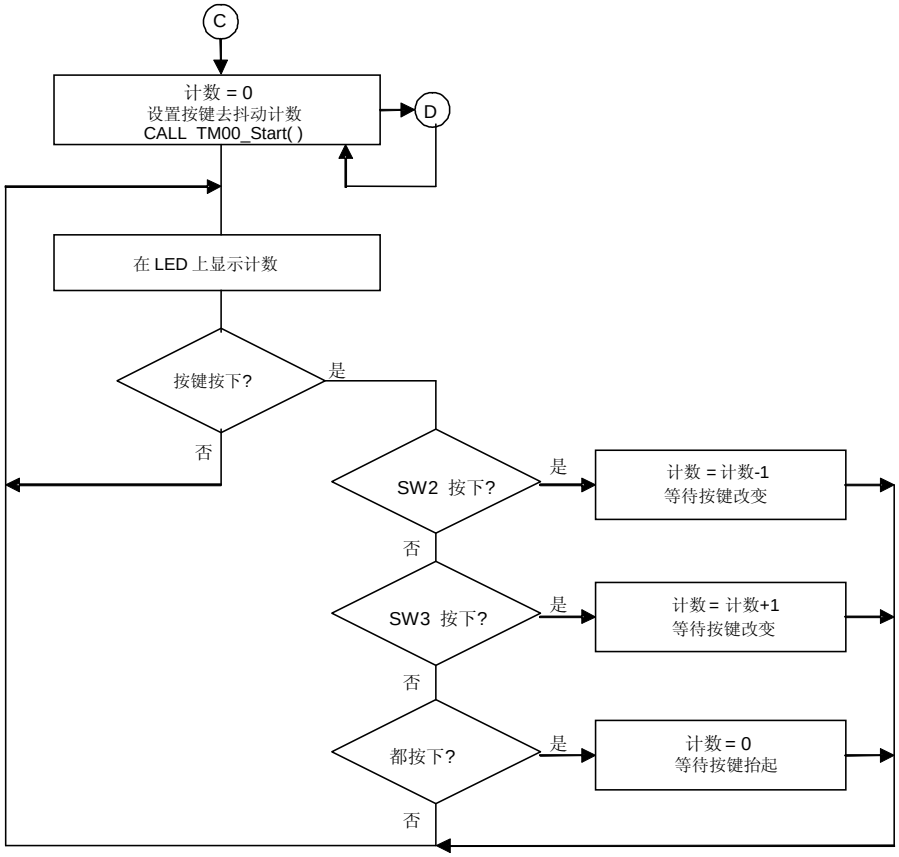
TM00_Init() 程序，由 SystemInit() 调用，用于间隔定时器操作预置定时器 00 寄存器。改变寄存器设置时推荐首先禁止定时器操作。

预置模式寄存器 PRM00，控制定时器时钟，被设置为 0x00。以这个设置，定时器 00 将使用 f_{PRS}，主系统时钟 20 MHz 作为计数时钟。这意味着计数寄存器 TM00 将每 1/20,000,000 s 或每 50 ns 计数一次。

将涉及 INTTM000 中断的寄存器设置为低优先级并且清中断标志。 屏蔽寄存器控制允许定时器中断。

CRC00 寄存器设置 CR000 为比较寄存器，并设置比较值 0x4E1F。CR000 与 TM00 每 (0x4E1F + 1)或 20,000 计数值匹配。每 20,000 * 50 ns=1,000,000 ns= 1ms。0x4E1F 由 Applilet 计算在对话框中提供指定的 1ms 间隔。

1.3.3 Main( ) – 主程序 – 按键去抖动的间隔定时器

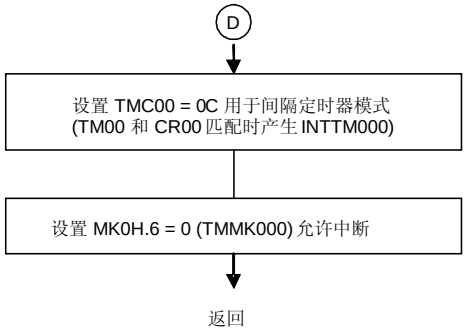


主程序设置可变计数初值，调用按键去抖动初始化程序，并调用 TM00_Start() 启动定时器 00 计数，进入循环。

主程序在 LED 上显示当前计数值。检查是否有按键按下，调用程序获取当前按键去抖动状态。如果按键抬起，将返回循环的顶部，显示计数，直到去抖动值完成后。如果按键按下，计数值改变，显示更新为新值。

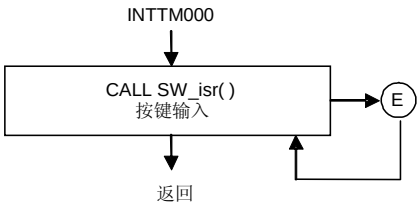
在程序循环中，每 1ms 产生 INTTM000 中断，并调用 MD_INTTM000() 中断服务程序。

1.3.4 TM00_Start( ) – 启动定时器 00 操作



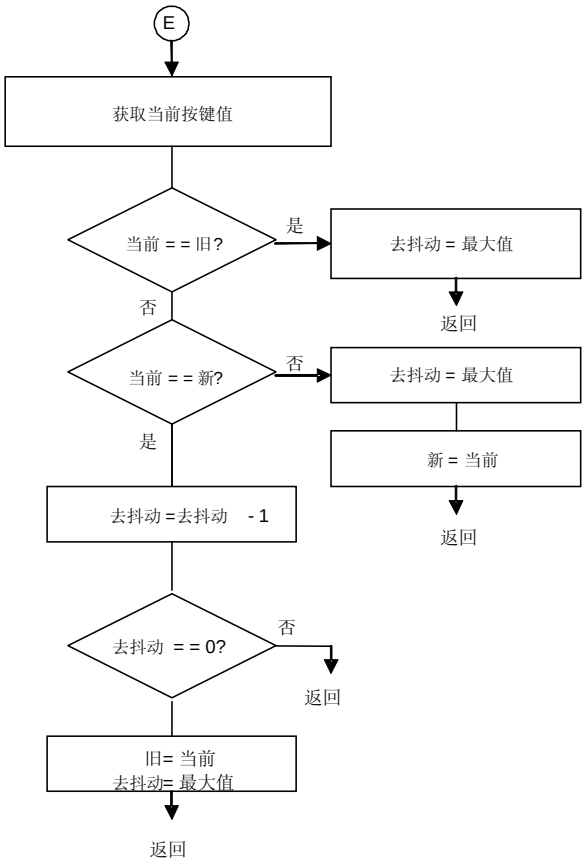
TM00_Start() 程序设置 TMC00 寄存器启动间隔定时器操作，并在 MK0H 中断屏蔽寄存器清 TMMK000 位，允许 INTTM000 中断。

1.3.5 MD_INTTM000( ) –INTTM000 中断服务程序



当产生 INTTM000 中断时，调用 MD_INTTM000() 中断服务程序，并在 sw_0537.c 程序中调用 sw_isr() 按键去抖动。

1.3.6 SW_isr( ) – 按键去抖动中断服务程序



此程序 (不由 Applilet 产生)，每 1ms 调用 1 次。比较当前按键值和预置值，并且一旦值是稳定的去抖动指定的计数值，更新 sw_new 变量。去抖动值由主程序调用 sw_get() 检查。

1.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言或汇编语言代码来管理 NEC 微控制器的外设。请参见使用 Applilet 版本的附录。

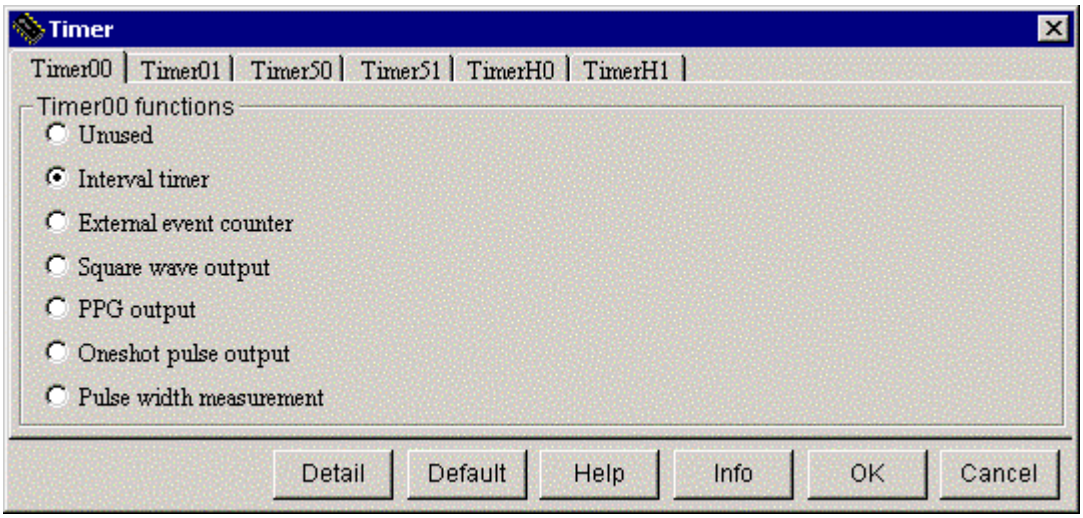
Applilet 用于创建基础的初始化代码和定时器 00 的驱动代码，并且可以管理用于按键输入和 LED 显示输出的 I/O 端口。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 00 的代码和产品程序列表描述如下。

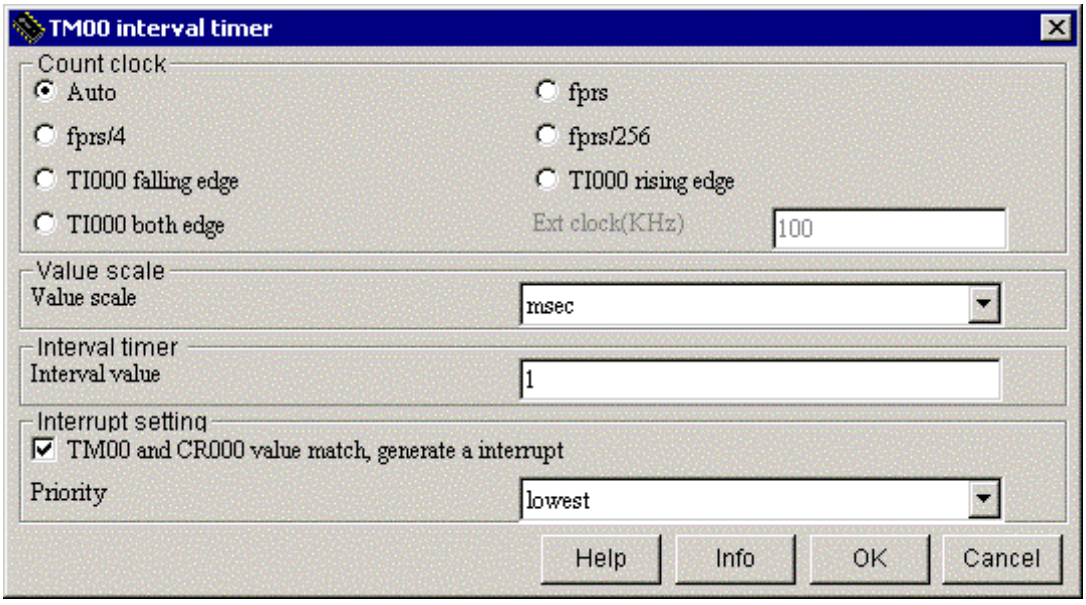
启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同外设的界面。

1.4.1 定时器 00 的 Applilet 配置

选择 “Timer”，选择定时器 00 列表，显示了可用的定时器 00 设置。将定时器设置为间隔定时器。



一旦选择使用定时器 00 ，“Detail” 按钮可用于设置定时器 00 选择模式。 点击“Detail” 按钮出现对话框设置间隔定时器。



这里是选择定时器 00 的操作细节。我们希望定时器每 1ms 产生中断，刻度设置为 “msec” (毫秒)，并且间隔值设置为 1。左侧设置计数时钟选择“Auto”；Applilet 将适当地选择定时器时钟允许 1ms 间隔。

因此我们希望当定时器间隔到时产生中断，勾选检查 TM00(定时器计数寄存器)和 CR000(定时器比较寄存器)匹配时产生中断。中断优先权设置为低，因为我们并不需要十分精确的间隔。

1.4.2 Applilet 产生代码

一旦定时器 00 对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其功能，允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码、中断服务程序和定时器 00 的驱动子程序。

当“Generate”按钮按下，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持定时器 00，Applilet 产生 timer.h, timer.c 和 timer_user.c，并且其他几个和定时器 00 无关。

1.4.3 Applilet 为定时器 00 产生的文件和函数

支持定时器 00 所产生的代码在文件 timer.h, timer.c 和 timer_user.c 中。它们包括以下内容。

1.4.3.1 Timer.h

头文件 timer.h 包含了控制定时器 00 的函数声明和定时器 00 初始化值的定义。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。

1.4.3.2 Timer.c

源文件 Timer.c 包含了 Applilet 为定时器 00 产生的下列函数：

void TM00_Init(void);
TM00_Init() 程序初始化定时器 00 外设，在 Applilet 定时器 00 对话框中设定。

void TM00_Start(void);
TM00_Start() 程序启动定时器 00 操作，并允许 INTTM000。

void TM00_Stop(void);
TM00_Stop() 程序停止定时器 00 操作，并禁止定时器中断。此程序不能用于演示程序。

MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
TM00_ChangeTimerCondition() 函数改变定时器 00 比较寄存器 CR000 和 CR010 的值，并改变定时器 00 的间隔。

array_reg 参数指向列值，放置一个或其他的比较值；
array_num 参数为 1 (选择 CR000) 或 2 (选择 CR010 和 CR000)。我们不需要改变定时器 00 的间隔时间，演示程序可不使用此程序。

1.4.3.3 Timer_user.c

源程序 timer_user.c 给用户提供了一些框架函数。这些函数是空函数，允许用户增加自己的应用代码。

__interrupt void MD_INTTM000(void);
这个中断服务程序用于，当 TM00 和 CR000 值匹配时产生定时器 00 中断 INTTM000。一旦定时器启动，中断每毫秒产生一次。

由 Applilet 产生的中断服务程序为空。使用间隔定时器按键去抖动，调用放置在 MD_INTTM000() 中的函数 sw_isr() 。

在 sw_0537.c 中的 sw_isr() 程序，检查每毫秒的按键值，并且每个值要稳定数毫秒，支持去抖动按键新值。 一旦程序执行，sw_isr() 返回到 MD_INTTM000()，然后返回主程序。

sw_isr() 代码放置在 MD_INTTM000()中，保存调用次数并返回 sw_isr()。sw_0537.c 中是按键相关的所有函数。

1.4.4 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
System.h	时钟关联定义
System.c	Clock_Init()函数
Port.h	定义默认 I/O 端口状态
Port.c	PORT_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

1.4.5 不由 Applilet 产生的演示程序

演示程序包括以下不由 Applilet 产生的文件。

文件	功能
Sw_0537.h	按键按下输入头文件
Sw_0537.c	按键读取和去抖动代码
Led_0537.h	7 段 LED 头文件
Led_0537.c	7 段 LED 上显示数据头文件

1.5 演示平台

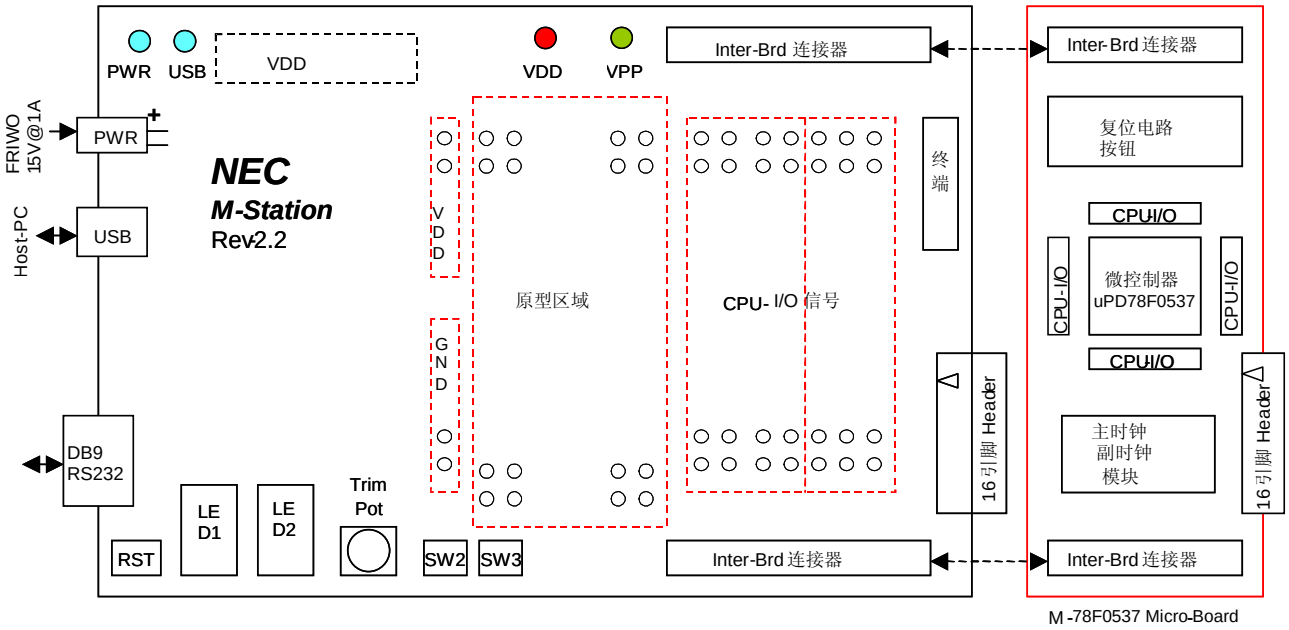
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

1.5.1 资源

为了演示该程序，需要使用以下资源：

- 焊有 uPD78F0537 8 位微控制器的 M-78F0537 Micro-Board
- M-Station II 仿真系统，M-Station II 上的资源：
  - 7 段 LED LED1 和 LED2
  - 按钮按键 SW2 和 SW3

以上硬件列表的详细说明，请参见用户手册，该手册可以向 NEC 索取。

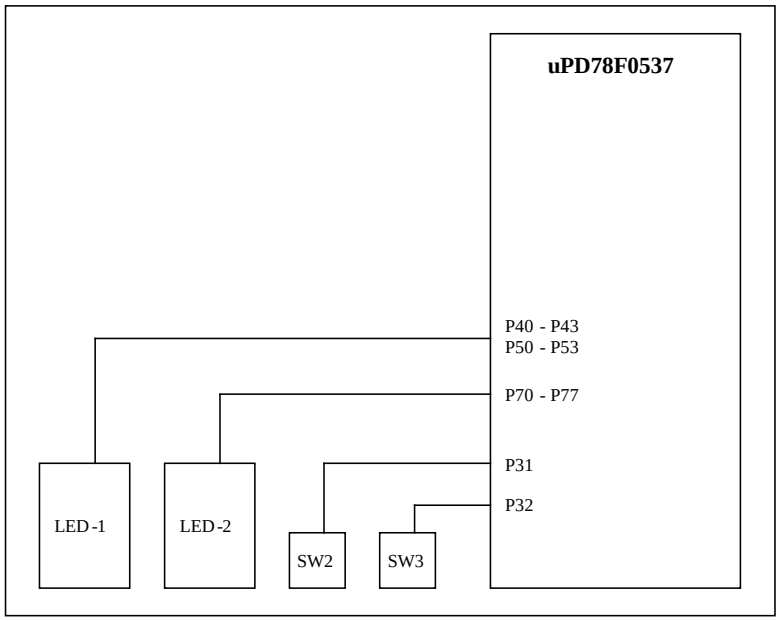


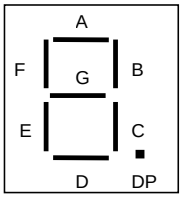
1.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 按下 SW2　　观察减少数
- 按下 SW3　　观察增加数

1.6 硬件框图





LED-1 和 LED-2

段	LED-1	LED-2
A	P40	P70
B	P41	P71
C	P42	P72
D	P43	P73
E	P50	P74
F	P51	P75
G	P52	P76
DP	P53	P77

1.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

列表中的文件在附录中。

文件	目的	由 Applilet 产生	由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟关联定义	是	否
Systeminit.c	SystemInit() 和 hdwinit() 函数	是	否
System.c	Clock_Init()函数	是	否
Timer.h	定时器关联定义	是	否
Timer.c	定时器函数	是	否
TIMER_user.c	定时器中断的用户代码	是	是
Port.h	端口关联定义	是	否
Port.c	Port_Init()函数	是	否
Led_0537.h	LED 显示定义	--	是
Led_0537.c	LED 显示函数	--	是
Sw_0537.h	按键输入定义	--	是
Sw_0537.c	按键输入函数	--	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

2 外部事件计数器使用16位定时器00 (TM00)

定时器/计数器可用于外部事件检测，计数或测量事件间的时间。外部事件有效沿信号为微控制器的一个输入引脚。负沿 (1 -> 0)，正沿 (0 -> 1)，或同时用作事件。NEC 大多数 78K0-系列微控制器定时器/计数器支持外部事件计数器。

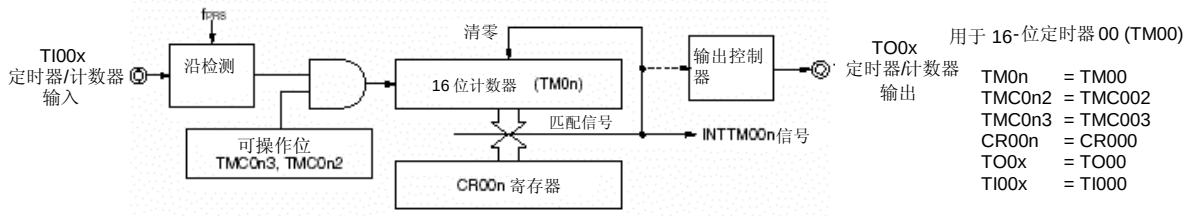
本演示程序，16 位定时器/计数器 00 (TM00) 用作计数外部事件。8 位定时器/计数器 TM50 用作方波发生器，输出连接到 TM00 定时器/计数器输入。方波沿被当作事件处理。TM00 检测并计数，当达到指定计数值时产生中断。程序将显示中断发生次数，提供外设对事件频率直观表示。

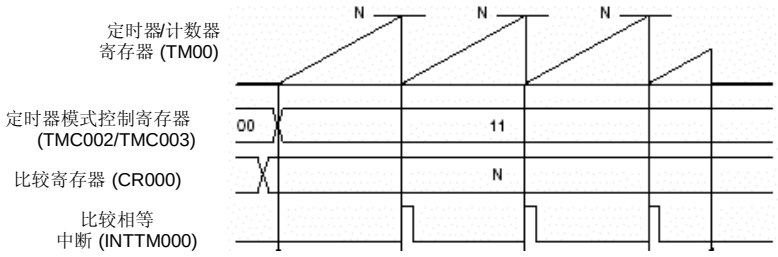
2.1 外部事件计数器特征

外部事件计数器，16 位定时器提供以下特征：

- o 可在外部事件的下降沿、上升沿或双沿计数
- o 事件计数从 1 到 65535
- o 指定事件值产生时可选项中断

如果选择 TI000 输入为定时器/计数器的时钟输入，并且定时器/计数器模式寄存器设置为"清零&启动模式"。定时器/计数器将与外部事件时钟(TI000)输入同步清零并启动计数。TI000 作为外部事件。定时器/计数器寄存器与捕捉/比较寄存器(CR000)比较，并且当两个寄存器值匹配时产生中断。定时器/计数器将清零并重启计数，重复计数处理。捕捉/比较寄存器包含预置事件检测数。





- 预置模式寄存器 = 11　　选择 TI000 作为计数时钟
- 定时器/计数器模式寄存器 = 11　　清零&启动模式
- 比较寄存器值　　预置外部事件检测值
- 比较匹配中断　　当匹配时产生中断

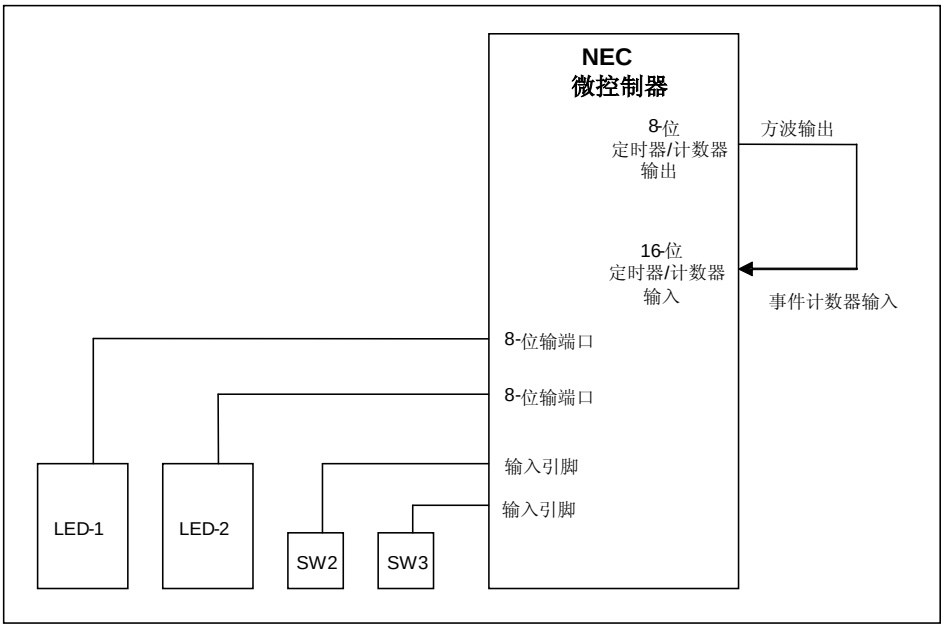
配置定时器作为外部事件计数器操作，按以下步骤：

- o 设置 PRM00 寄存器 TI000 沿输入和计数沿
- o 设置 CR000 值，设置 CRC00 将 CR000 用作比较寄存器
- o 设置中断优先级，清中断标志并取消中断屏蔽
- o 设置定时器模式寄存器 TMC00 用于清零&启动模式，当 TM00 和 CR000 匹配时允许定时器操作

2.2 程序描述和说明

演示程序使用 8 位定时器 50 产生方波输出，连接到 16 位定时器/计数器 TM00 输入。TM00 在外部事件计数器模式下处理方波的有效沿，并且计数到预置的值时产生中断，事件计数复位重新计数。

中断发生次数在 2 位 7 段 LED 上显示。可读取 2 个输入按键。按下一个按键设置方波输出到低频率，结果是低速计数。按下其他按键，设置为高频率，结果是快速计数。



8 位定时器/计数器 TM50：

- 用于产生外部事件
- 当定时器/计数器溢出改变输出：作为外部事件处理
  - 选择快速时钟将产生快速外部事件
  - 低速时钟将产生慢速外部事件
- 8 位定时器/计数器输出连接到 16 位定时器/计数器 TI000 输入

按键输入：

- 按下 SW2 设置 8 位定时器高速频率(100 Hz)
- 按下 SW3 设置 8 位定时器低速频率(50 Hz)

16 位定时器/计数器 TM00：

- 由设置预置模式寄存器选择 TI000 作为外部事件输入时钟
- 设置定时器/计数器模式寄存器执行"清零&启动模式"
- 设置比较寄存器预置值- 侦察外部事件数

2 位显示：

- 当每次 16 位定时器/计数器匹配时发生中断
- 按升序 1, 2, ．．．99 显示外部事件设置值

说明：

- TM00 设置计数 TI000 下降沿，并在 50 个计数值后产生中断
- TM50 设置 TO50 输出 50 Hz 或 100 Hz 方波

2.3 软件流程图

演示程序由以下主要几部分组成：

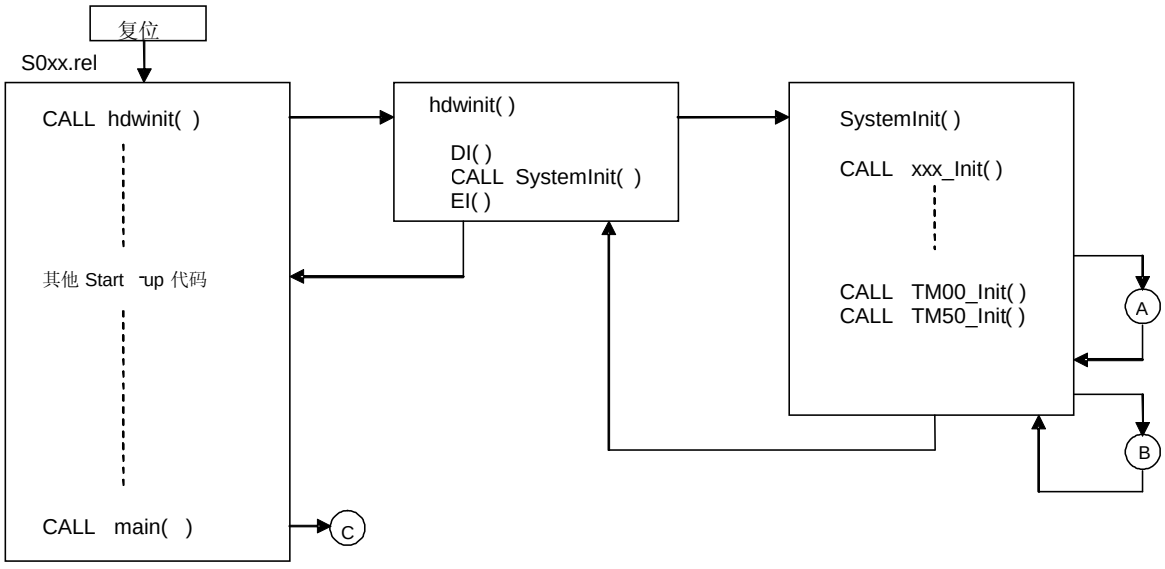
- o 用于程序的初始化代码，在 main() 程序启动前调用
- o 主程序循环，检查按键状态和改变 TM50 时间
- o 由 Applilet 产生的子程序处理启动定时器 00 和定时器 50，停止和改变间隔
- o 用户代码子程序用于处理定时器 00 中断(Applilet 产生框架中断服务程序，增加用户代码)
  注 TM50 不产生中断
- o 用于按键输入，LED 显示输出的子程序

这里的流程图描述初始化、主程序、定时器 00 和定时器 50 的相关子程序 和 TM00 中断服务程序。

2.3.1 程序启动和初始化

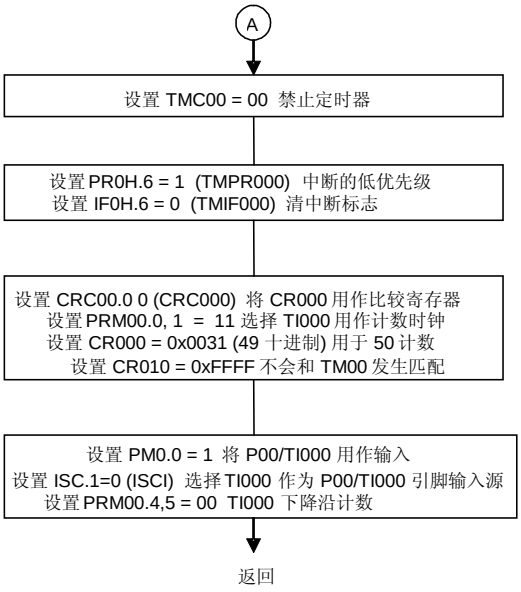
适用于 78K0 的 C 语言程序，C 语言启动代码由目标代码文件提供，例如 s01.rel，联接到用户程序。启动代码调用 hdwinit( )函数，用户可以在此放置硬件初始化代码。

当使用 Applilet 产生 78K0 C 语言，hdwinit( ) 函数自动增加到用户程序，并调用 SystemInit( )函数。SystemInit( ) 函数轮流调用各外设的初始化程序。



hdwinit( ) 函数完成后，启动代码调用用户程序 main( ) 函数。启动 main( ) 函数，执行所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

2.3.2 TM00_Init( ) -定时器 00 初始化



TM00_Init() 程序，由 SystemInit() 调用，用于间隔定时器操作预置定时器 00 寄存器。改变寄存器设置时推荐首先禁止定时器操作。

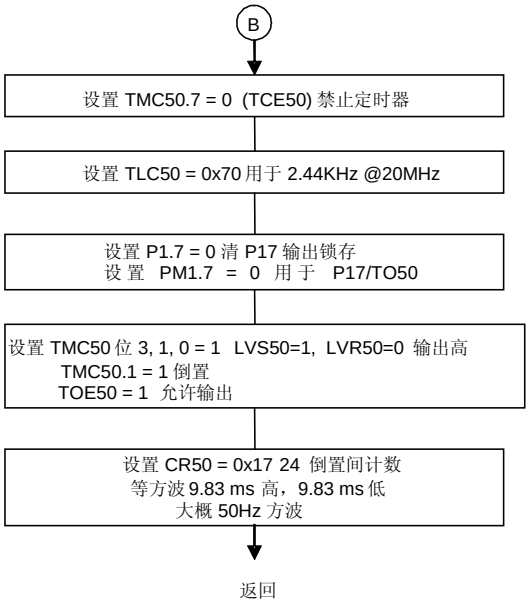
将涉及 INTTM000 中断的寄存器设置为低优先级并且清中断标志。 屏蔽寄存器控制允许定时器中断。

CRC00 寄存器设置 CR000 为比较寄存器，预置模式寄存器 PRM00 设置 TI000 为定时器 00 的计数时钟，并且是是下降沿。比较寄存器 CR000 设为 49，每 50 计数值产生中断。CR010 设为最大值，CR010 和 TM00 不会匹配产生中断。

端口模式寄存器 PM0 第 0 位设为 P00 为输入引脚，提供 P00/TI000 引脚输入。端口引脚设置来自 Port_Init()。ISC 寄存器设置选择 P00/TI000 引脚作为 TI000 输入源。

Applilet 计算值和提供在定时器 00 对话框进行了以上设置的代码。

2.3.3 TM50_Init( ) – 定时器 50 初始化



TM50_Init() 程序，初始化定时器 50 用作 50 Hz 的方波发生器。写控制寄存器时先禁止定时器操作。

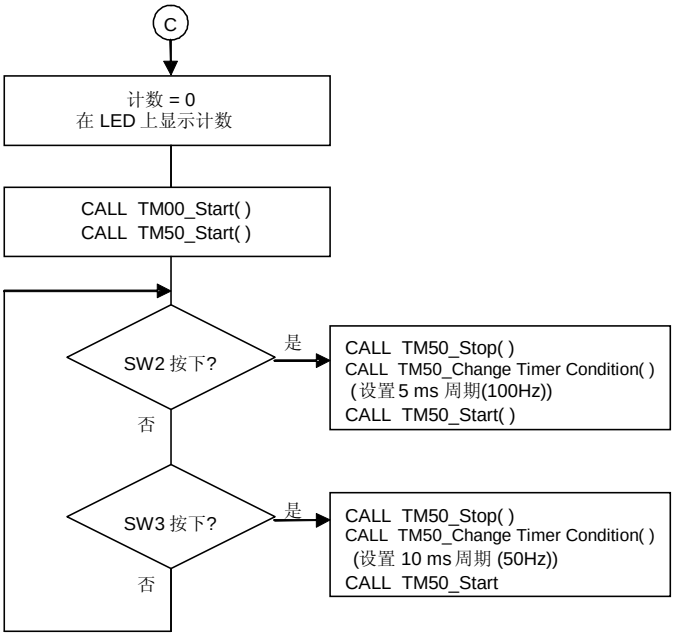
TLC50 寄存器设为 0x07，选择定时器 50 的时钟为 f_{PRS}/2**13。主时钟为 20 MHz，计数时钟为 20,000,000/8192 或 2.44 KHz。TM50 定时器每 409.6 ms 累加。

TM50 在 P17/TO50 输出方波。P1.7 设为 0，清端口锁存，并且 PM1.7 设为 0，引脚为输出。端口设置来自 Port_Init()。

当 TM50 和 CR50 匹配时，TMC50 输出高电平，然后倒置输出。产生的方波，高低电平宽度由 TM50/CR50 匹配间隔决定。CR50 设置为 0x17 或 23，结果是每 24 个计数时钟 TM50 与 CR50 匹配。每计数时钟 409.6 us，每 409.6 * 24 us 或 9.83 ms 匹配。提供大概 10ms 低和高宽度，产生 50 Hz 方波。

Applilet 计算值并且提供在定时器 50 对话框中进行以上设置的代码。

2.3.4 Main( ) – 主程序 – 外部事件计数器



主程序初始化计数变量和 LED 显示。调用 TM00_Start() 和 TM50_Start() 函数启动定时器初始化模式。

建立后，主程序循环调用在 sw_0537.c 中的 sw_get()函数检查按键。如果没有按键按下，程序什么也不执行。

定时器 50 产生 50 Hz 的方波到 P17/T050 输出，连接到 P00/TI000 输入。TI000 输入的下降沿，定时器 00 在寄存器 TM00 中计数输入沿。每 50 个沿（大概 1 秒），TM50 和 CR000 匹配，产生 INTTM000 中断。调用中断服务程序 MD_INTTM000()，累加计数并在 LED 上显示。

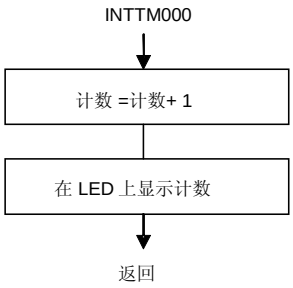
用户无动作，大概每秒显示计数。

如果 SW2 按下，调用 TM50_Stop() 函数停止方波发生器，并调用 TM50_ChangeTimerCondition 设置 CR50 为原来值的一半。结果是 100 Hz 方波改为 50 Hz 。调用 TM50_Start() 重启定时器。

每秒显示两次。

如果 SW3 按下，定时器 50 设置原值。 每秒显示一次。

2.3.5 MD_INTTM000( ) - INTTM000 中断服务程序



当发生 INTTM000 ， 调用 MD_INTTM000() 中断服务程序。程序累加全程变量，LED 显示计数。

2.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言 或汇编语言代码来管理 NEC 微控制器的外设。请参见使用 Applilet 版本的附录。

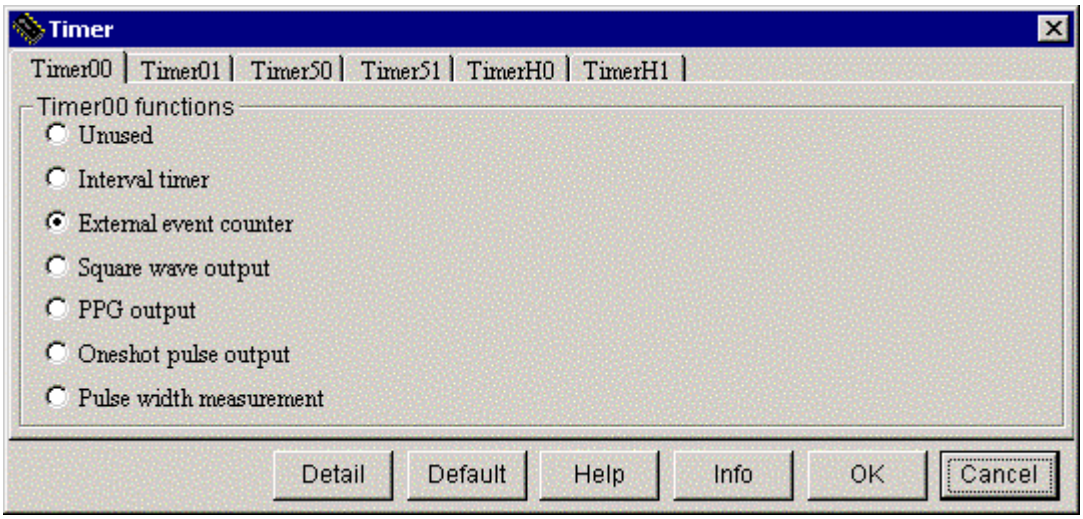
Applilet 用于创建基础的初始化代码和定时器 00、定时器 50 的驱动代码，并且可以管理用于按键输入和 LED 显示输出的 I/O 端口。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 00 和定时器 50 的代码和产品程序列表描述如下。

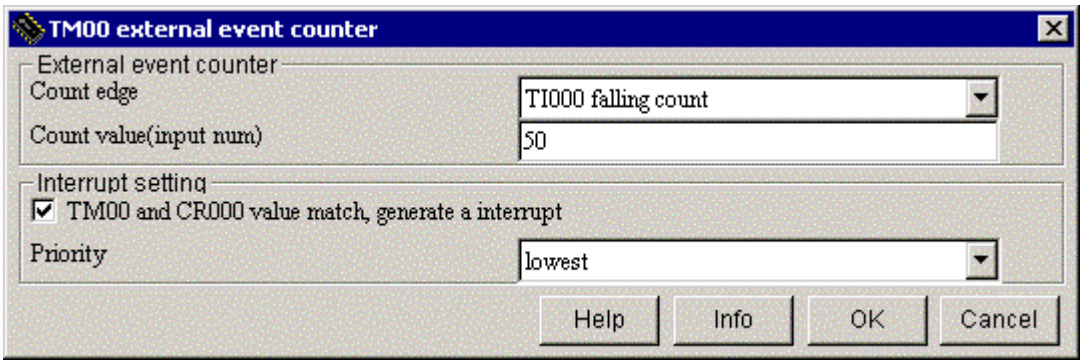
启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同外设的界面。

2.4.1 定时器 00 的 Applilet 配置

选择 “Timer”，选择定时器 00 列表显示可用的定时器 00 设置。将定时器 00 设置为外部事件计数器。



一旦选择使用定时器 00 ，“Detail” 按钮可用于设置定时器 00 选择模式。点击“Detail” 按钮出现定时器 00 对话框设置外部事件计数器。

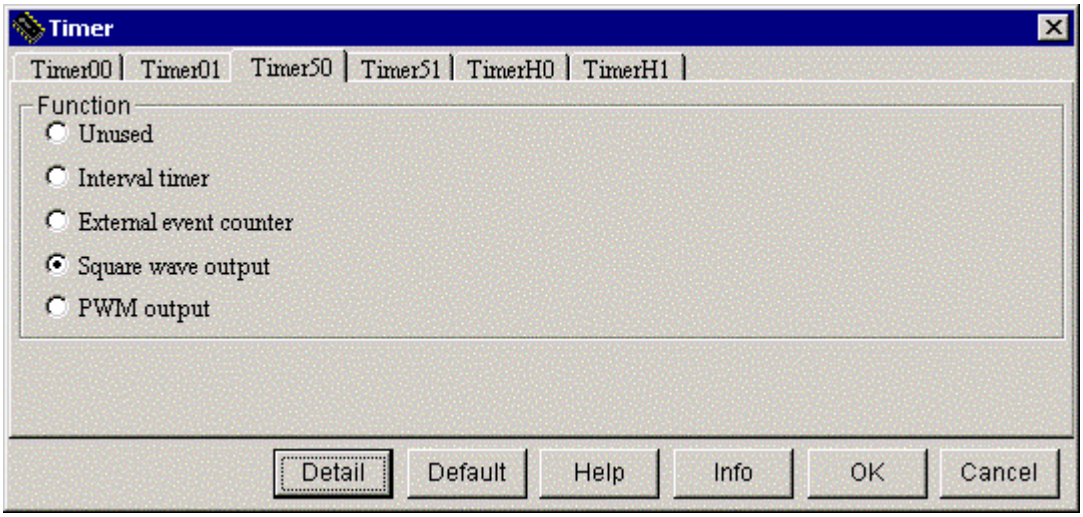


这里是选择定时器 00 的操作细节。我们希望定时器产生每 50 个 TI000 输入的计数值时中断。计数值设为 50，并且计数有效沿选择为 TI000 的下降沿。

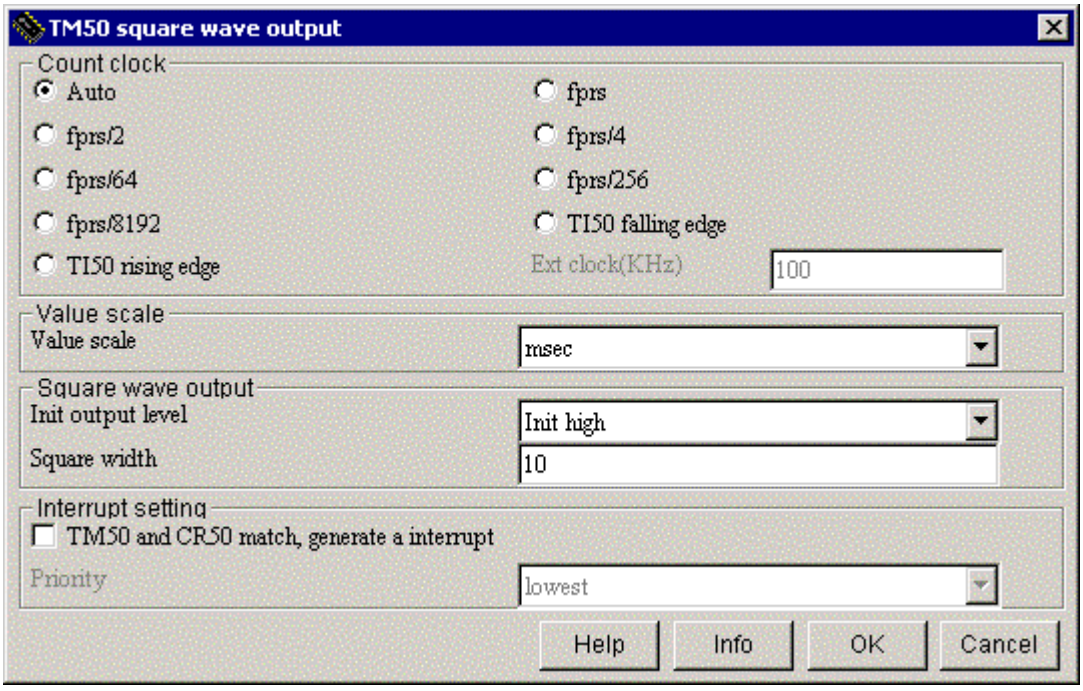
我们希望产生中断，勾选当 TM00（定时器 00 定时器计数寄存器）和 CR000（定时器 00 比较寄存器）匹配时产生中断。中断优先级设置为低。

2.4.2 定时器 50 的 Applilet 配置

选择 “Timer”，选择定时器 50 列表显示可用的定时器 50 设置。将定时器 50 设置为方波产生。



一旦选择使用定时器 50 ，“Detail” 按钮可用于设置定时器 50 选择模式。点击“Detail” 按钮出现定时器 50 对话框设置方波输出。



这里是选择定时器 50 的操作细节。我们希望定时器产生一个 50 Hz 的方波。方波为 10 ms 高，10 ms 低。

刻度值设置为 ms，并在方波宽度对话框中设置 10。输出设为高为先，计数时钟设为“Auto”。Applilet 会为时钟驱动器寄存器产生适当的值，提供设置时间。

我们不希望产生中断，不勾选当 TM50（定时器 50 定时器计数寄存器）和 CR50（定时器 50 比较寄存器）匹配时产生中断。因此，不产生中断，中断优先级设置处为灰色。

2.4.3 Applilet 产生代码

一旦定时器 00 对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其功能，允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码、中断服务程序、定时器 00 和定时器 50 的驱动子程序。

当“Generate”按钮按下，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持定时器 00 和定时器 50，Applilet 产生 timer.h、timer.c 和 timer_user.c，并且其他几个和定时器 00 无关。

2.4.4 Applilet 为定时器 00 和定时器 50 产生的文件和函数

支持定时器 00 和定时器 50 所产生的代码在文件 timer.h、timer.c 和 timer_user.c 中。它们包括以下内容。

2.4.4.1 Timer.h

头文件 timer.h 包含了控制定时器 00 和定时器 50 的函数声明，和定时器 00 和定时器 50 初始化值的定义。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。

2.4.4.2 Timer.c

源文件 Timer.c 包含了 Applilet 为定时器 00 和定时器 50 产生的下列函数：

void TM00_Init(void);
The TM00_Init() 程序初始化定时器 00 外设，在 Applilet 定时器 00 对话框中设定的。

void TM00_Start(void);
TM00_Start()程序启动定时器 00 操作 并允许 INTTM000。

void TM00_Stop(void);
TM00_Stop()程序停止定时器 00 操作并禁止定时器中断。此程序不能用于演示程序。

MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
TM00_ChangeTimerCondition() 函数改变定时器 00 比较寄存器 CR000 和 CR010 的值，并改变定时器 00 的间隔。

array_reg 参数指向列值，放置一个或其他的比较值；the array_num 参数为 1 (选择 CR000) 或 2 (选择 CR010 和 CR000)。我们不需要改变定时器 00 的间隔时间，演示程序可不使用此程序。

void TM50_Init(void);
TM50_Init() 程序初始化定时器 50 外设，在 Applilet 定时器 50 对话框中设定的。

void TM50_Start(void);
TM50_Start()程序由允许定时器来启动定时器 50 操作。

void TM50_Stop(void);
TM50_Stop() 程序由禁止定时器来停止定时器 50 操作。

MD_STATUS TM50_ChangeTimerCondition(UCHAR value);
TM50_ChangeTimerCondition() 函数改变定时器 50 比较寄存器 CR50 的值，并且改变定时器 50 方波高/低宽度。

2.4.4.3 Timer_user.c

源程序 timer_user.c 给用户提供了一些框架函数。这些函数是空函数，允许用户增加自己的应用代码。

__interrupt void MD_INTTM000(void);
这个中断服务程序，用于当 TM00 和 CR000 值匹配时产生定时器 00 中断 INTTM000。一旦定时器启动，中断每 50 计数值产生一次。

由 Applilet 产生的中断服务程序为空。使用外部事件计数器更新显示，增加计数变量和 LED 显示代码。

2.4.5 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
System.h	时钟关联定义
System.c	Clock_Init()函数
Port.h	定义默认 I/O 端口状态
Port.c	PORT_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

2.4.6 不由 Applilet 产生的演示程序

演示程序包括以下不由 Applilet 产生的文件。

文件	功能
Sw_0537.h	按键按下输入头文件
Sw_0537.c	按键读取和去抖动代码
Led_0537.h	7 段 LED 头文件
Led_0537.c	7 段 LED 上显示数据头文件

2.5 演示平台

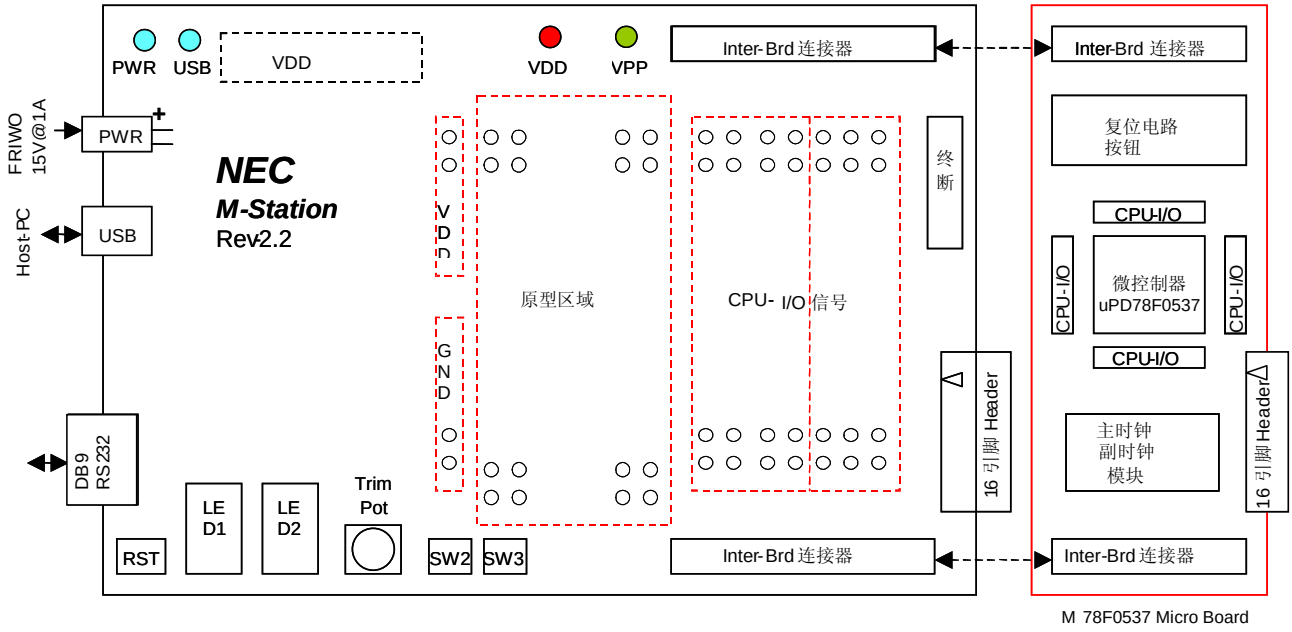
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

2.5.1 资源

为了演示该程序，需要使用以下资源：

- 焊有 uPD78F0537 8 位微控制器的 M-78F0537 Micro-Board
- M-Station II 仿真系统，M-Station II 上的资源：
  - 7 段 LED LED1 和 LED2
  - 按钮按键 SW2 和 SW3

以上硬件列表的详细说明，请参见用户手册，该手册可以向 NEC 索取。

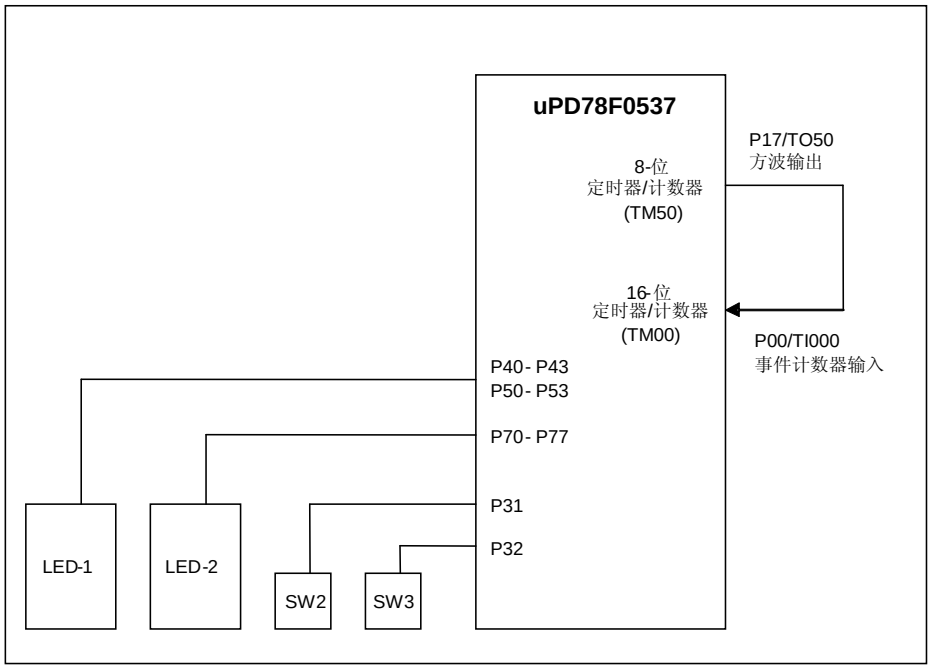


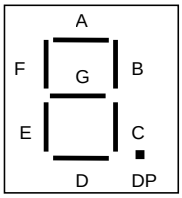
2.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 按下 SW2　　观察快速显示
- 按下 SW3　　观察慢速显示

2.6 硬件框图





LED-1 和 LED -2

段	LED-1	LED-2
A	P40	P70
B	P41	P71
C	P42	P72
D	P43	P73
E	P50	P74
F	P51	P75
G	P52	P76
DP	P53	P77

2.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

列表中的文件在附录中。

文件	目的	由 Applilet 产生	由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟关联定义	是	否
Systeminit.c	SystemInit() 和 hdwinit() 函数	是	否
System.c	Clock_Init() 函数	是	否
Timer.h	定时器关联定义	是	否
Timer.c	定时器函数	是	否
TIMER_user.c	定时器中断的用户代码	是	是
Port.h	端口关联定义	是	否
Port.c	Port_Init()函数	是	否
Led_0537.h	LED 显示定义	--	是
Led_0537.c	LED 显示函数	--	是
Sw_0537.h	按键输入定义	--	是
Sw_0537.c	按键输入函数	--	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

3 产生单脉冲使用16位定时器01 (TM01)

此演示程序，16 位定时器 01 (TM01) 被用作产生单脉冲。16 位定时器/计数器 TM01 配置为软件触发单脉冲发生；外部按键按下软件触发。脉冲产生在 TM01 输出连接到 16 位定时器/计数器 00(TM00)的输入，配置为外部事件计数器。TM00 计数 TM01 产生的脉冲。并且检测到脉冲设置数后增加显示计数。

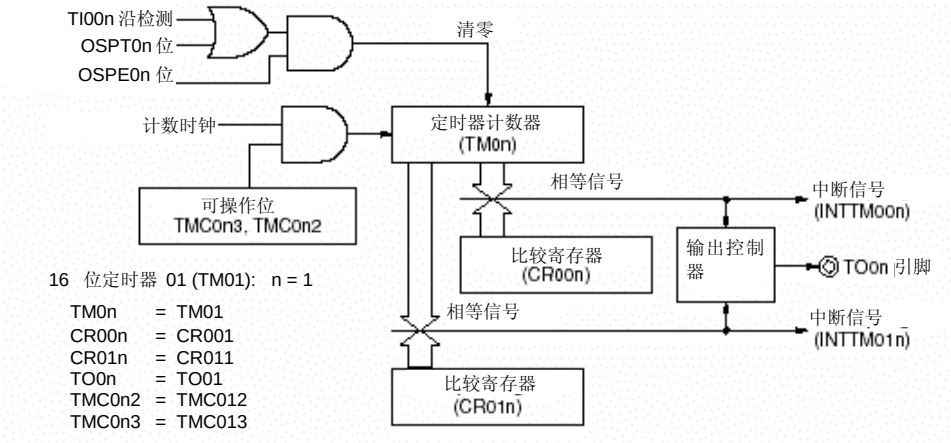
3.1 16-位间隔定时器单脉冲配置

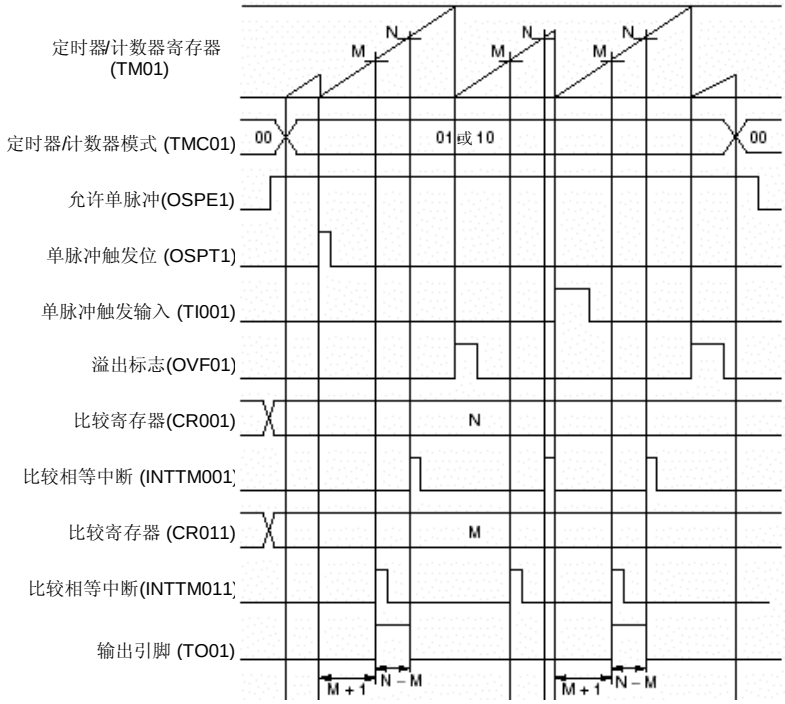
单脉冲从定时器/计数器输出，当：

- 定时器模式控制寄存器位设置为产生单脉冲
- 输出控制寄存器位设置为允许单脉冲

单脉冲模式，16 位定时器提供以下特征：

- 使用 PRM00 寄存器，基于外设时钟分频可变时间
- 软件触发或外部输入
- 使用 CR010 计数时间基准时钟，从触发到启动脉冲的可变延时
- CR000 设置时间从触发到结束脉冲的可变宽度脉冲
- 在脉冲开始或结束选择中断





单脉冲输出有效电平宽度= (N-M) * 计数时钟周期

例如

- 如果定时器/计数器模式寄存器设置为自由跑模式
- 输出控制寄存器设置为允许单脉冲和软件触发

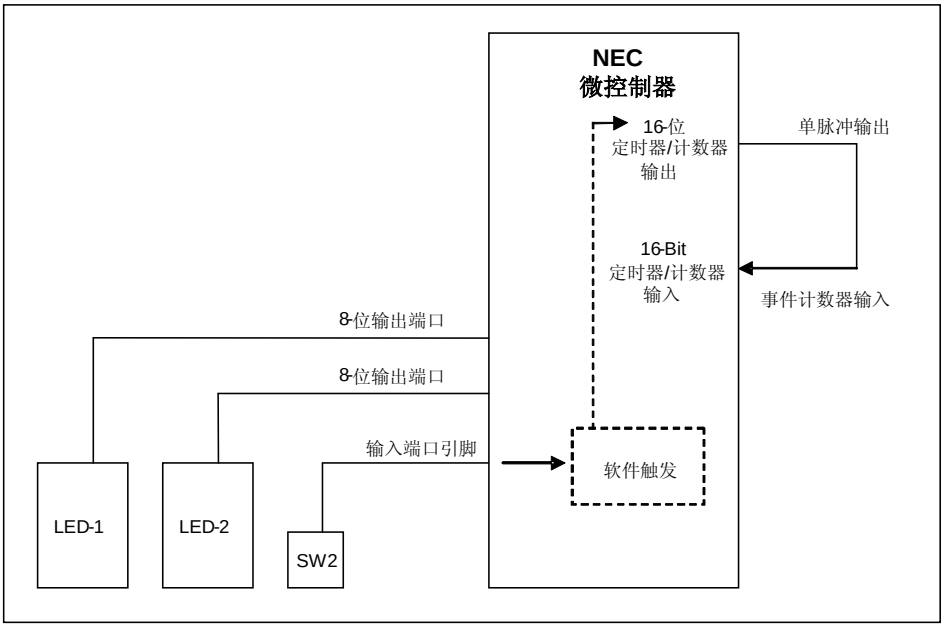
    - 定时器操作期间软件触发设置控制寄存器
    - 定时器/计数器(TM00)清零&启动触发
    - 输出不同于 2 比较寄存器(CR000 和 CR010) 的脉冲

配置定时器 01 用于软件触发单脉冲输出，按以下步骤：

- 使用 PRM01 设置基准时间；如果使用外部触发，设置输入有效沿
- 设置 CRC01 使用 CR001 和 CR011 作为比较寄存器
- 使用 CR010 设置脉冲起始时间，并且使用 CR001 设置脉冲结束时间
- 设置适当的端口模式寄存器和输出引脚锁存
- 设置 TOC01 寄存器允许输出，设置初始状态，设置 CR001 和 CR011 匹配时倒置输出，并设置单脉冲允许位
- 设置中断优先级，清中断标志，并且非屏蔽中断
- 启动定时器用于软件触发，设置 TMC01 寄存器用于自由跑模式(启动定时器用于外部触发，设置 TMC01 寄存器用于清零&启动模式)

3.2 程序描述和说明

演示程序使用外部按键引起 TM01 软件触发用于单脉冲。每次按键按下，产生一个脉冲。单脉冲输出是 TM00 外部事件计数器模式，TM01 计数脉冲输入 (外部事件)，并且输入的脉冲数达到设置数后中断。LED 显示中断数。



说明：

- 按下 SW2 引起 TM01 软件触发
- TM01 产生单脉冲，触发后高 50us 低 100us
- TM00 设置 TI000 下降沿计数，并 5 次计数后中断

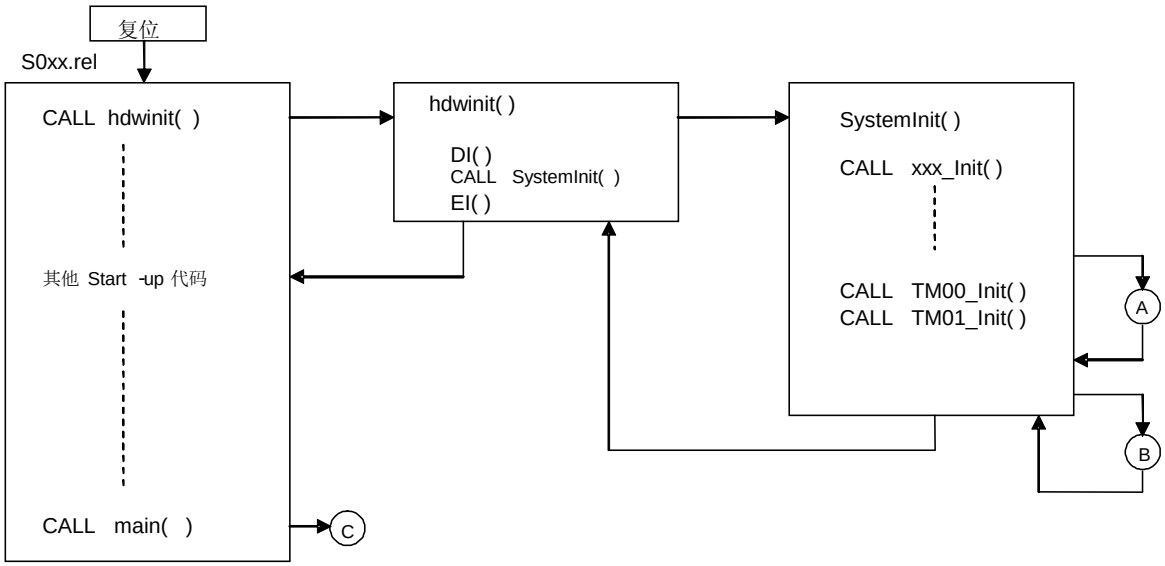
3.3 软件流程图

演示程序由以下主要几部分组成：

- 程序的初始化代码，在 main() 程序启动前调用
- 主程序循环，检查按键状态和 TM01 触发
- 由 Applilet 产生的子程序启动 TM00 和 TM01，停止和改变间隔
- 用户代码子程序用于处理 TM00 中断(Applilet 产生框架中断服务程序，增加用户代码)
  注 TM01 不产生中断
- 用于按键输入，LED 显示输出的子程序

这里的流程图描述初始化、主程序、TM00 和 TM01 相关子程序 和 TM00 的中断服务程序。

3.3.1 启动程序和初始化

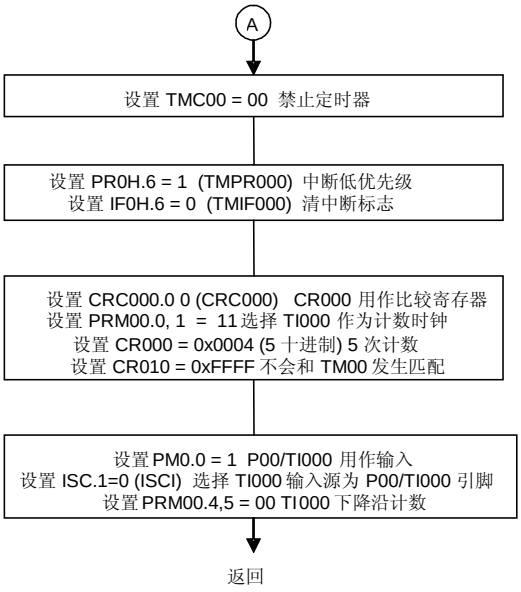


C 语言写的 78K0 程序启动代码由结果代码文件提供，例如 s0l.rel,，连接到用户程序。调用启动代码的函数名称是 hdwinit( )，用户可在这里放置任意的硬件初始化代码。

当 Applilet 产生 78 K0 的 C 语言，hdwinit( ) 函数自动运行增加到用户程序，并调用 SystemInit( ) 函数。这个 SystemInit( ) 函数用于每个外设调用初始化程序。

hdwinit( ) 函数结束后，启动代码调用用户程序的 main( ) 函数。启动 main( ) 函数，所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

3.3.2 TM00_Init( ) – 定时器 00 初始化



TM00_Init() 程序，由 SystemInit() 调用，用于外部事件计数器操作预置定时器 00 寄存器。改变寄存器设置时推荐先禁止定时器操作。

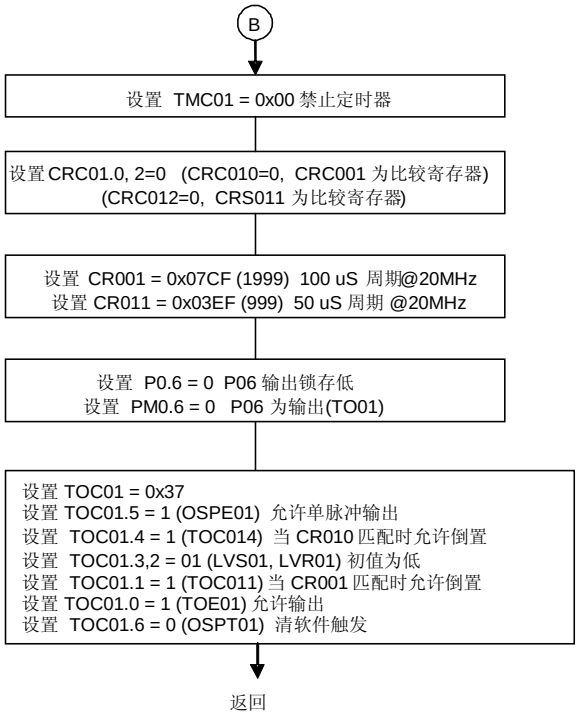
将涉及 INTTM000 中断的寄存器设置为低优先级并且清中断标志。 屏蔽寄存器控制允许定时器中断。

CRC00 寄存器设置 CR000 作为比较寄存器。预分频模式寄存器 PRM00 设置 TI000 作为定时器 00 的计数时钟，并且 TI000 下降沿计数。比较寄存器 CR000 设为 4，每 5 次计数产生一次中断。CR010 设为最大值，因此当 TM00 与 CR010 匹配时不会发生中断。

端口模式寄存器 PM0 位 0 设为 P00 作为输入引脚，用于提供 P00/TI000 输入。端口引脚设置同样位于 Port_Init() 程序。ISC 寄存器也设置为选择 P00/TI000 引脚作为 TI000 输入源。

Applilet 计算值和提供根据定时器 00 对话框进行了上述设置的代码。

3.3.3 TM01_Init( ) -定时器 01 初始化



TM01_Init() 程序，由 SystemInit() 调用，用于产生单脉冲操作预置定时器 01 寄存器。写控制寄存器时首先禁止定时器操作。

因为没有中断，不修改中断控制寄存器。定时器 01 的中断 INTTM001 禁止。

预分频寄存器 PRM01，控制时钟输入到定时器 01，设为 00 指定计数时钟为 f_{PRS}，20 MHz 为系统时钟。TM01 每 0.05us 计数。

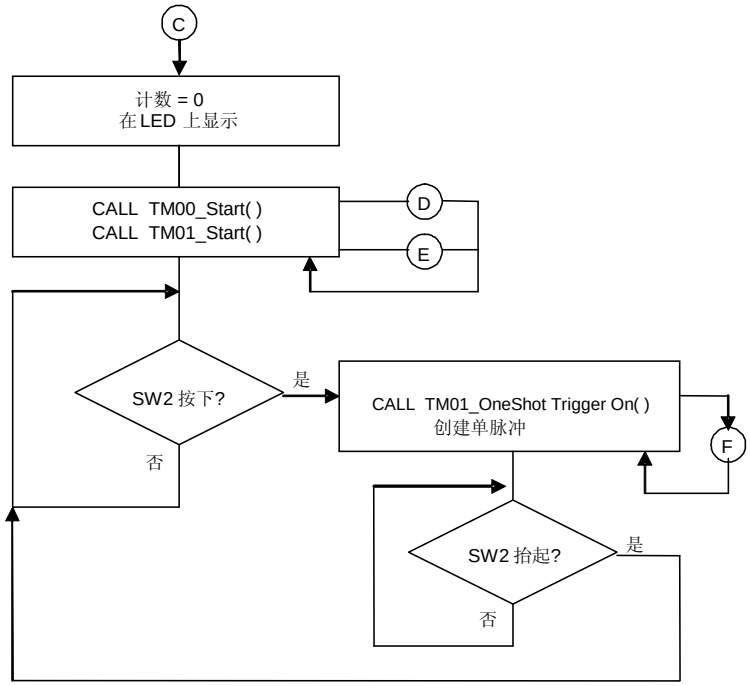
CR001 设为 0x7CF (1999)。与 TM01 2000 计数值或 2000 * 0.05 = 100us 比较。脉冲再次为低设置单脉冲周期。CR011 设为 0x3EF (999)；与 TM01 1000 计数值或 1000 * 0.05 = 50us 比较。控制单脉冲延时，高 50us 后触发。

CRC01 设置 CR001 和 CR011 用作比较寄存器。

因为要在 P06/TO01 输出单脉冲，P0.6 输出锁存设为零，并且端口模式寄存器 PM0 的位 6 设为零将 P06/TO01 引脚作为输出引脚。

TOC01 输出控制寄存器用于定时器 01 ，设置为允许单脉冲输出，当 CR001 和 CR011 都和 TM01 匹配时倒置。输出最初为低，清软件触发。

3.3.4 Main( ) – 主程序 – 产生单脉冲



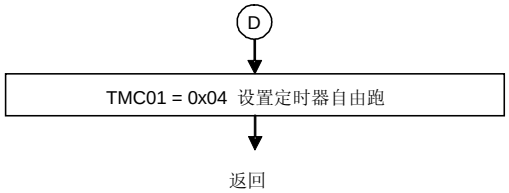
主程序设置计数变量初值为零，并在 LED 上显示计数。调用 TM00_Start() 和 TM01_Start() 启动定时器。

程序检查 SW2 状态，如果抬起，无动作。

如果 SW2 按下，程序调用 TM01_OneShotTriggerOn() 触发单脉冲。发生单脉冲在 P06/TO01 引脚输出。因为该引脚连接到 P00/TI000，将在定时器 00 外部事件计数器中计数。程序等待 SW2 抬起。

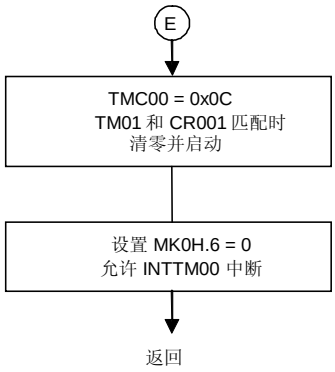
程序的结果是 SW2 每按下 5 次，定时器 00 计数 5 次，当事件计数器与设置值匹配时产生 INTTM000 。INTTM000 执行中断服务程序 MD_INTTM000 ，计数变量累加并显示。

3.3.5 TM01_Start( ) – 启动定时器 01



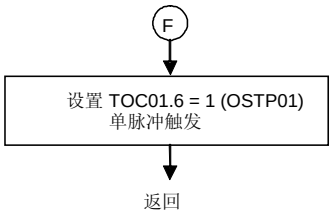
TM01_Start() 函数设置 TMC01 寄存器，允许定时器 01 操作在单脉冲产生模式下。单脉冲可由软件触发。

3.3.6 TM00_Start( ) – 启动定时器 00



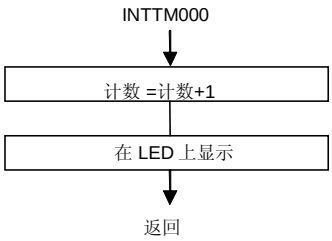
TM00_Start() 设置 TMC00 清零并启动外部事件计数器，允许 INTTM000 中断。

3.3.7 TM01_OneShotTriggerOn( ) – 定时器 01 单脉冲触发



TM01_OneShotTriggerOn() 函数设置 TOC01 寄存器 中的 OSTP01(单脉冲软件触发位) ，清零并启动单脉冲操作。定时器 01 产生单脉冲，并不创建其他脉冲直到再次调用此程序。

3.3.8 MD_INTTM000( ) – 中断服务程序



产生 INTTM000 ， 调用 MD_INTTM000() 中断服务程序。程序累加全局计数变量，并在 LED 上显示计数值。

3.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言 或汇编语言代码来管理 NEC 微控制器的外设。请参见使用 Applilet 版本的附录。

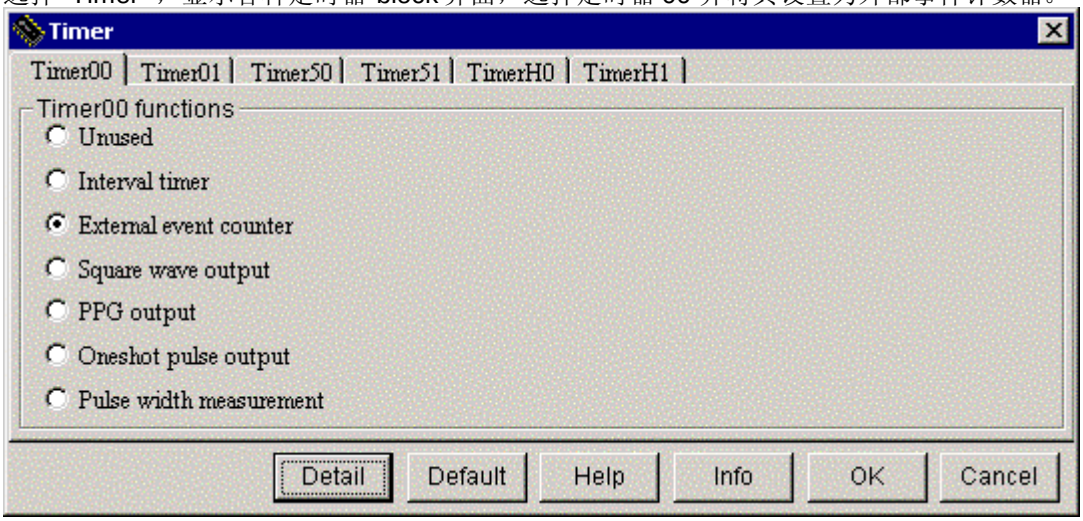
Applilet 用于创建基础的初始化代码和定时器 00、定时器 01 的驱动代码，并且可以管理 LED 显示输出的 I/O 端口。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 00 和定时器 0 的代码和产品程序列表描述如下。

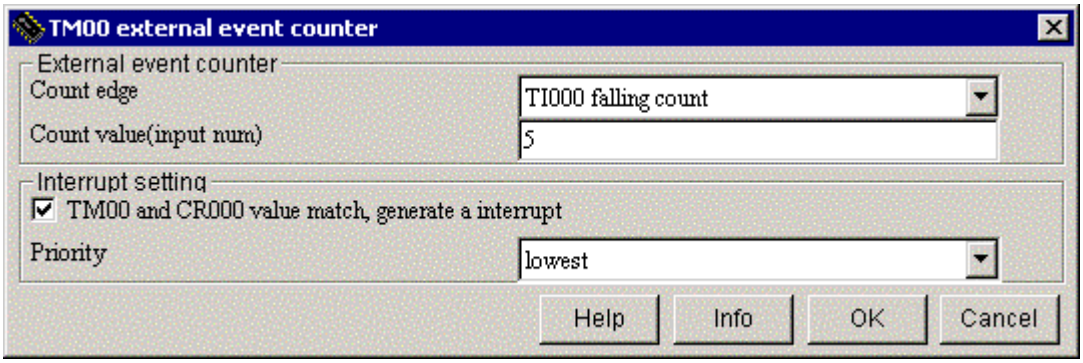
启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同的外设模块的界面。

3.4.1 定时器 00 的 Applilet 配置

选择 “Timer”，显示各种定时器 block 界面，选择定时器 00 并将其设置为外部事件计数器。



一旦选择使用定时器 00， “Detail” 按钮可用于设置定时器 00 选择模式。点击“Detail” 按钮出现对话框设置外部事件计数器。

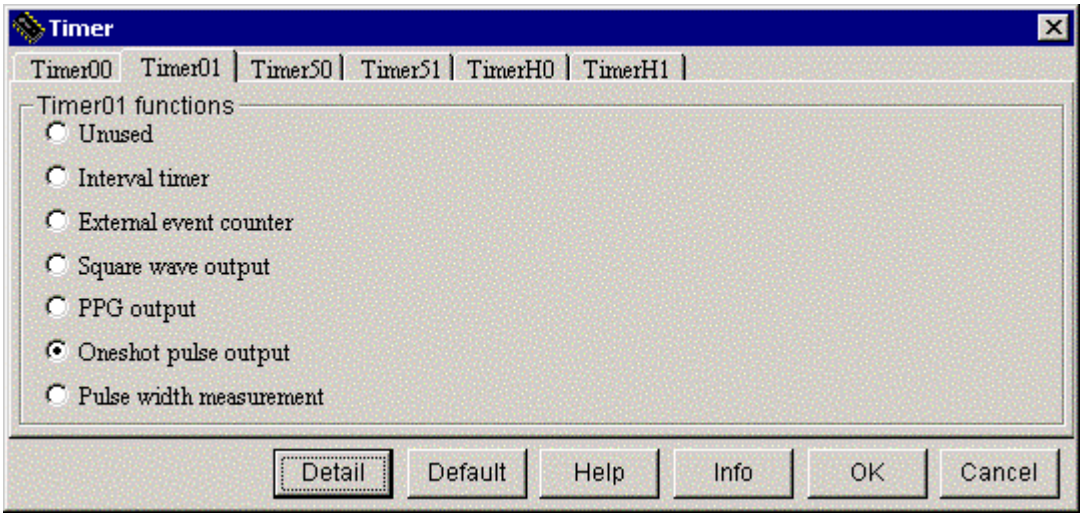


这里是选择定时器 00 的操作细节。我们希望定时器输入到 TI000 每 5 个计数值产生中断。计数值设为 5，并选择计数沿为 TI000 的下降沿。

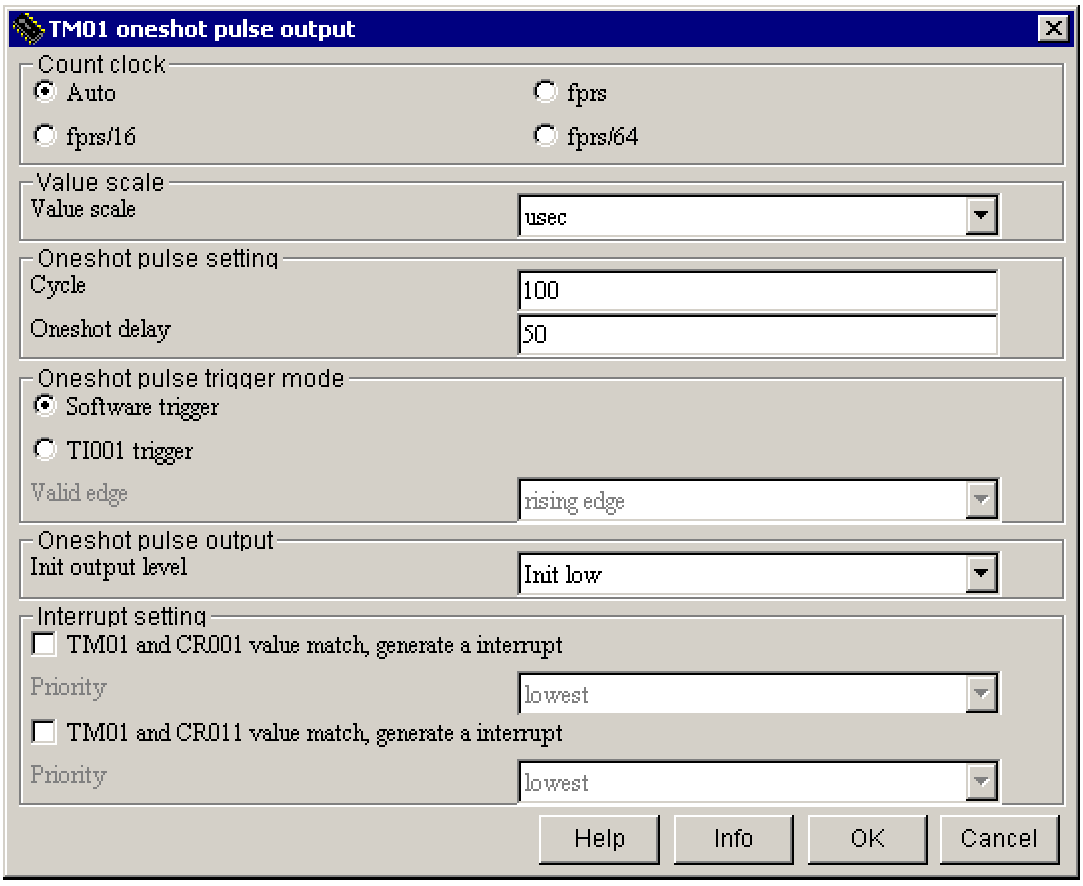
因此我们希望当计数值达到 5 时产生中断，检查 TM00(定时器计数寄存器)和 CR000(定时器比较寄存器)匹配时产生中断。中断优先权设置为最低。

3.4.2 定时器 01 的 Applilet 配置

选择 “Timer”，显示各种定时器 block 界面，选择定时器 01 并将其设置为单脉冲输出。



一旦选择使用定时器 01， “Detail” 按钮可用于设置定时器 01 选择模式。点击“Detail” 按钮出现对话框设置单脉冲产生。



这里是选择定时器 01 的操作细节。我们希望软件触发定时器产生单脉冲。脉冲延时 50us，并且宽度是 50us。

值刻度选择为 us。周期 (脉冲延时时间) 设置为 100，周期 100us，延时设为 50us。计数时钟设为“Auto”，Applilet 将产生时钟分频寄存器适当地值达到设置时间。

单脉冲设置为软件触发，胜于外部输入。程序将在 SW2 按键按下时触发单脉冲。单脉冲输出设为初值为低。

因为我们不希望产生中断，不勾选 TM01 (定时器 01 定时器计数寄存器) 与 CR001 或 CR010 匹配时产生中断。因为没有中断产生，不勾选中断优先级选择。

3.4.3 Applilet 产生代码

一旦定时器 对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其功能，允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码、中断服务程序、定时器 00 和定时器 01 驱动子程序。

当“Generate”按钮按下，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持定时器 00 和定时器 01，Applilet 产生 timer.h, timer.c 和 timer_user.c，并且其他几个和定时器无关。

3.4.4 Applilet 为定时器 00 和定时器 01 产生的文件和函数

支持定时器 00 和定时器 01 所产生的代码在文件 timer.h, timer.c 和 timer_user.c 中。它们包括以下内容。

3.4.4.1 Timer.h

头文件 timer.h 包含了控制定时器 00 和定时器 01 的函数声明，和定时器 00 和定时器 01 初始化值的定义。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。

3.4.4.2 Timer.c

源文件 Timer.c 包含了 Applilet 为定时器 00 和定时器 01 产生的下列函数：

void TM00_Init(void);
TM00_Init()程序初始化定时器 00 外设，在 Applilet 定时器 00 对话框中设定的。

void TM00_Start(void);
TM00_Start()程序启动定时器 00 操作 并允许 INTTM000。

void TM00_Stop(void);
TM00_Stop()程序停止定时器 00 操作并禁止定时器中断。此程序不能用于演示程序。

MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
TM00_ChangeTimerCondition()函数改变定时器 00 比较寄存器 CR000 和 CR010 的值，并改变定时器 00 的计数器比较值。

array_reg 参数指向列值，放置一个或其他的比较值；array_num 参数为 1 (选择 CR000) 或 2 (选择 CR010 和 CR000)。我们不需要改变定时器 00 的计数器比较值，演示程序可不使用此程序。

void TM01_Init(void);
TM01_Init()程序初始化定时器 01 外设，在 Applilet 定时器 01 对话框中设定的。

void TM01_Start(void);
TM01_Start()程序启动定时器 01 操作。

void TM01_Stop(void);
TM01_Stop()程序停止定时器 01 操作。

MD_STATUS TM01_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
TM01_ChangeTimerCondition()函数改变定时器 01 比较寄存器 CR001 和 CR011 的值，并改变定时器 01 的单脉冲周期和显示值。

array_reg 参数指向列值，放置一个或其他的比较值；array_num 参数为 1 (选择 CR001) 或 2 (选择 CR011 和 CR001)。我们不需要改变定时器 01 的单脉冲值，演示程序可不使用此程序。

void TM01_OneshotTriggerOn();
TM01_OneshotTriggerOn() 程序设置定时器 01 单脉冲软件触发。

3.4.4.3 Timer_user.c

源程序 timer_user.c 给用户提供了一些框架函数。这些函数是产生的空函数，允许用户增加自己的应用代码。

__interrupt void MD_INTTM000(void);
这个中断服务程序用于，当 TM00 和 CR000 值匹配时产生定时器 00 中断 INTTM000。一旦定时器启动，中断每 5 次计数产生一次。

由 Applilet 产生的中断服务程序为空。使用外部事件计数器更新显示，增加代码用于累加变量计数和 LED 显示。

3.4.5 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
System.h	时钟关联定义
System.c	Clock_Init()函数
Port.h	定义默认 I/O 端口状态
Port.c	PORT_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

3.4.6 不由 Applilet 产生的演示程序

演示程序包括以下不由 Applilet 产生的文件。

文件	功能
Sw_0537.h	按键按下输入头文件
Sw_0537.c	按键读取和去抖动代码
Led_0537.h	7 段 LED 头文件
Led_0537.c	7 段 LED 上显示数据头文件

3.5 演示平台

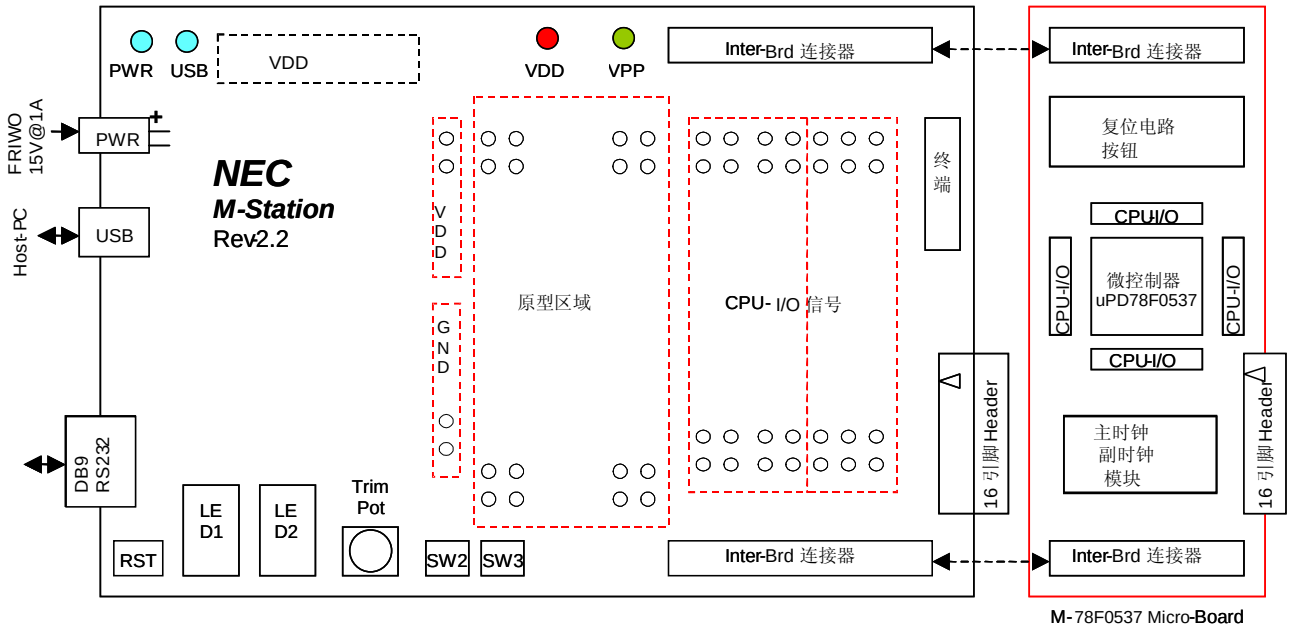
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

3.5.1 资源

为了演示该程序，需要使用以下资源：

- 焊有 uPD78F0537 8 位微控制器的 M-78F0537 Micro-Board
- M-Station II 仿真系统，M-Station II 上的资源：
  - 7 段 LED LED1 和 LED2
  - 按钮按键 SW2

以上硬件列表的详细说明，请参见用户手册，该手册可以向 NEC 索取。

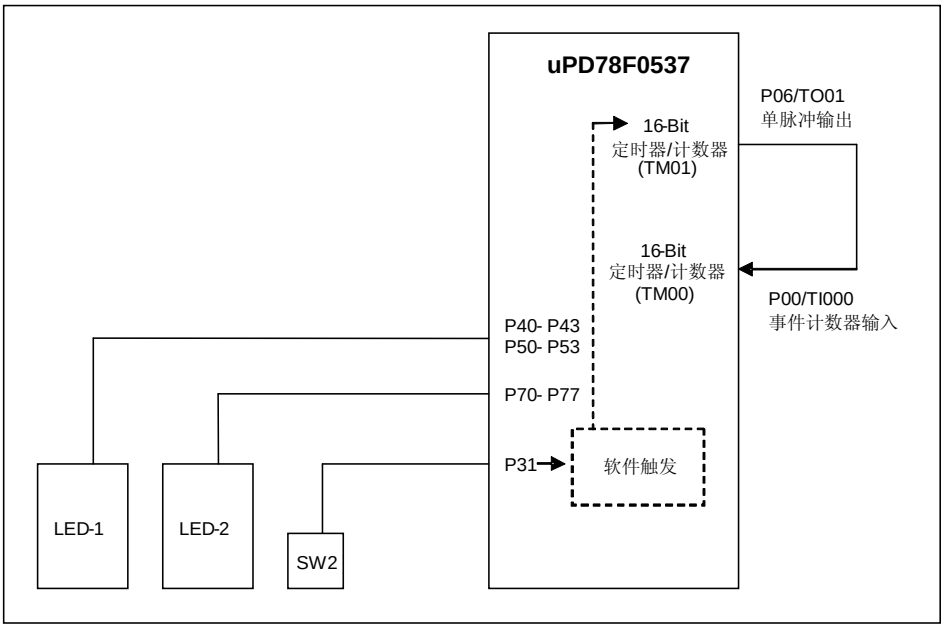


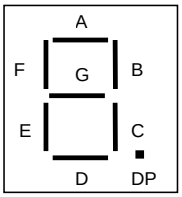
3.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 按下 SW2　　观察在 LED 上显示的中断次数
  每 5 次按下递增事件计数

3.6 硬件框图





LED-1 和 LED -2

段	LED-1	LED-2
A	P40	P70
B	P41	P71
C	P42	P72
D	P43	P73
E	P50	P74
F	P51	P75
G	P52	P76
DP	P53	P77

3.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

列表中的文件在附录中。

文件	目的	由 Applilet 产生	由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟选择定义	是	否
Systeminit.c	SystemInit() 和 hdwinit() 函数	是	否
System.c	Clock_Init()函数	是	否
Timer.h	定时器关联定义	是	否
Timer.c	定时器函数	是	否
TIMER_user.c	定时器中断的用户代码	是	是
Port.h	端口关联定义	是	否
Port.c	Port_Init()函数	是	否
Led_0537.h	LED 显示定义	--	是
Led_0537.c	LED 显示函数	--	是
Sw_0537.h	按键输入定义	--	是
Sw_0537.c	按键输入函数	--	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

4 适应于D/A 转换的PWM输出使用8位定时器51 (TM51)

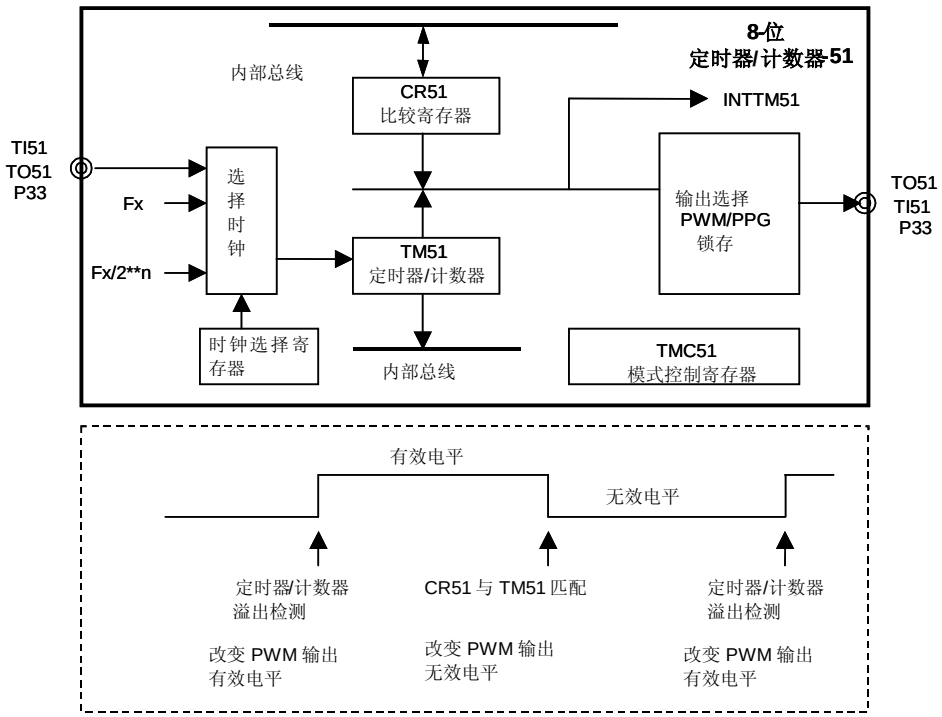
典型的，一个固定频率，可变占空比脉冲宽度调制 (PWM) 输出由 8 位定时器/计数器产生。PWM 信号应用于各种情况。例如， D/A 转换使用 PWM 输出，低功耗，低分辨率数字模拟的转换。

4.1 PWM 特征

在 PWM 模式中，8 位定时器提供以下特征：

- o 由主外设时钟分频选择 PWM 计数时钟
- o 输出脉冲从 0/256 时钟到 255/256 时钟
- o 输出有效高或有效低
- o 以起始 PWM 周期操作中断

当定时器/计数器模式寄存器设置为产生 PWM，一个固定频率，可变占空比 PWM 信号从 8 位定时器/计数器输出。

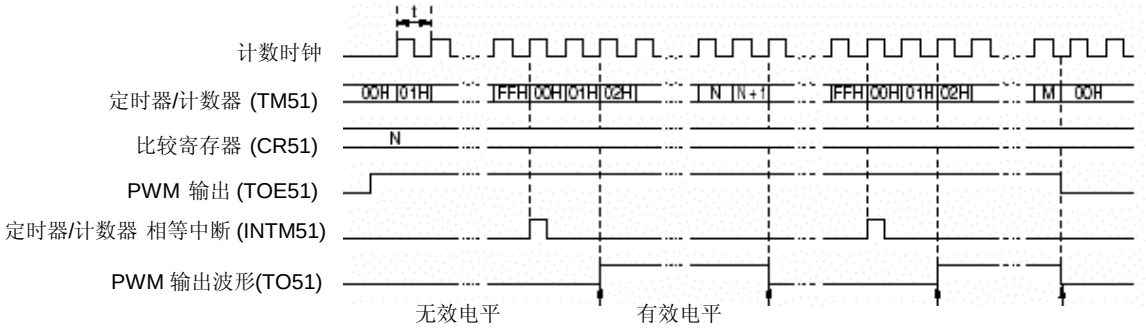


- 定时器/计数器控制模式寄存器　　设置 PWM 输出
- 定时器计数器计数时钟　　计数时钟由时钟选择寄存器选择
- 比较寄存器　　包含占空比周期值

当定时器/计数器溢出，PWM 输出设置为有效电平。当发生定时器/计数器和比较寄存器匹配(CR51 = TM51)，PWM 输出设置为非有效电平，直到下一溢出发生。在下一溢出，PWM 输出设置为有效电平并继续。使用 8 位定时器/计数器，PWM 输出为：

- 周期　　2**8 x t
- 有效时间　　N x t　　N = 00h ~ FFh
- 占空比　　N/2**8

PWM 时序如下所示。



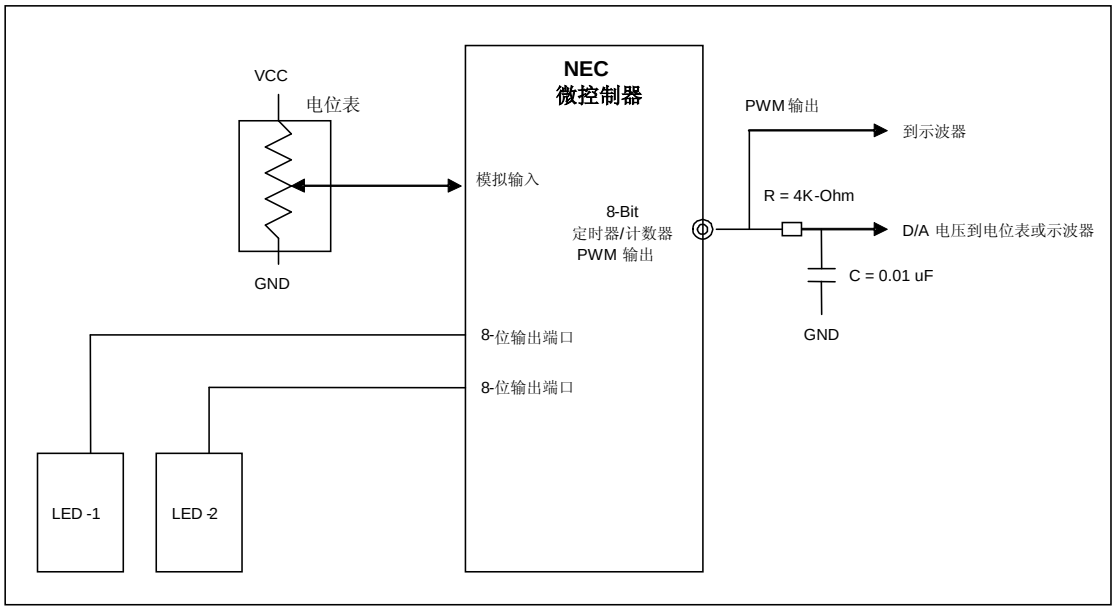
配置定时器 TM51 用于 PWM 操作，按以下步骤：

- 使用 TCL51 寄存器选择计数时钟
- 设置输出端口引脚为输出模式
- 如果使用中断，设置中断优先级，清中断标志，非屏蔽中断
- 使用 CR51 寄存器设置 PWM 有效时序
- 设置 PWM 模式，有效高或低，初始化输出状态，并使用 TMC51 允许输出
- TMC51.7(TCE51 位)允许定时器

4.2 程序描述和说明

使用 PWM 输出数字到模拟转换的程序描述。PWM 输出经过低通滤波器产生平均时间模拟信号。模拟信号振幅是与 PWM 脉冲宽度成比例的。

演示程序通过电位表将模拟信号输入到 NEC 微控制器。电位表电压读到微控制器使用 A/D 转换数字电压值。A/D 转换值在 LED 上显示。



A/D 转换值用作产生 PWM 脉冲信号宽度，由 A/D 转换值到比较寄存器产生 PWM。在经低通滤波器 PWM 输出相同的模拟信号电平。用户可使用示波器经低通滤波器调制模拟信号，并使用电压表产生经低通滤波器的模拟信号电压。

一般设计指南建议，PWM 频率应该在低通滤波器截止频率使波纹减到最小 5 到 10 的时间。当 PWM 频率最大大概为 78KHz，选择用作低通滤波器的电阻和电容大约为 4KHz， uPD78F0537 使用 20MHz 主时钟振荡。

说明：

- TM51 设置为 PWM 模式，有效高，初始输出低，无中断
- TM51 PWM 时钟设置为 20 MHz；PWM 周期为 20,000,000/256 = 78.125 KHz
- AD0/P20 设置作为 A/D 输入连续转换模式

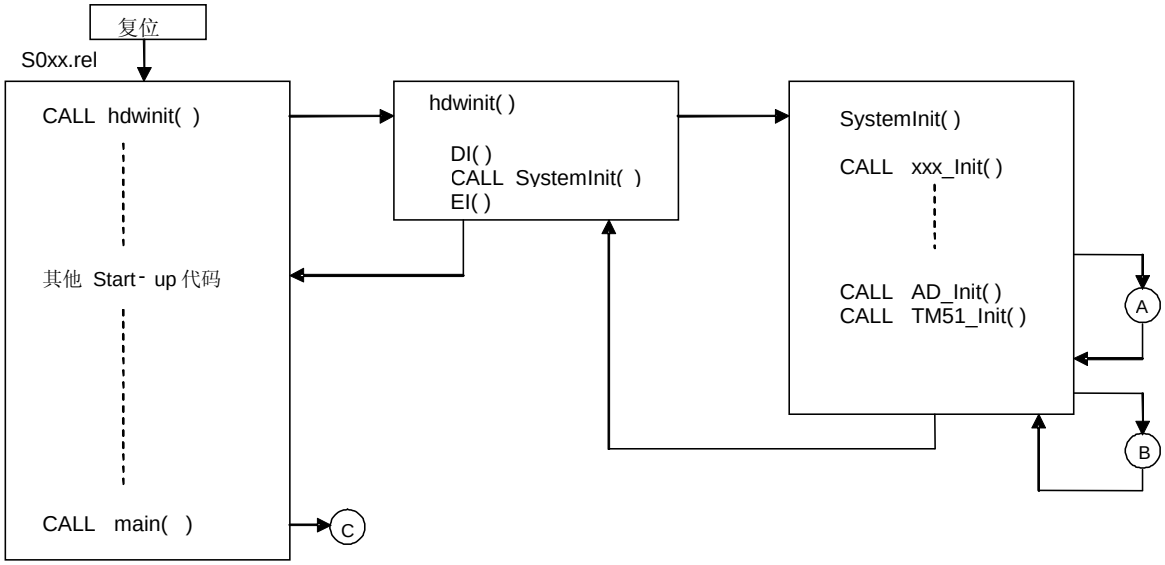
4.3 软件流程图

演示程序由以下主要几部分组成：

- o 程序初始化代码，main()程序开始前调用；
- o 主程序循环，检查 A/D 输入和改变 TM51 PWM 有效时序
- o 由 Applilet 产生 A/D，TM51 启动、停止和改变间隔子程序
- o LED 显示输出子程序

这里的流程图描述初始化、主程序和 TM51 相关子程序。

4.3.1 启动程序和初始化

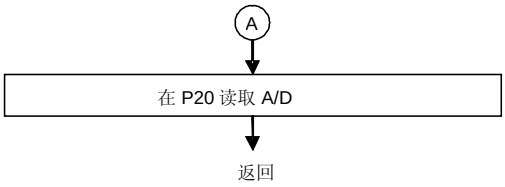


C 语言写的 78K0 程序启动代码由结果代码文件提供，例如 s0l.rel，连接到用户程序。调用启动代码的函数名称是 hdwinit( )，用户可在这里放置任意的硬件初始化代码。

当 Applilet 产生 78 K0 的 C 语言，hdwinit( ) 函数自动运行增加到用户程序，并调用 SystemInit( ) 函数。这个 SystemInit( ) 函数用于每个外设调用初始化程序。

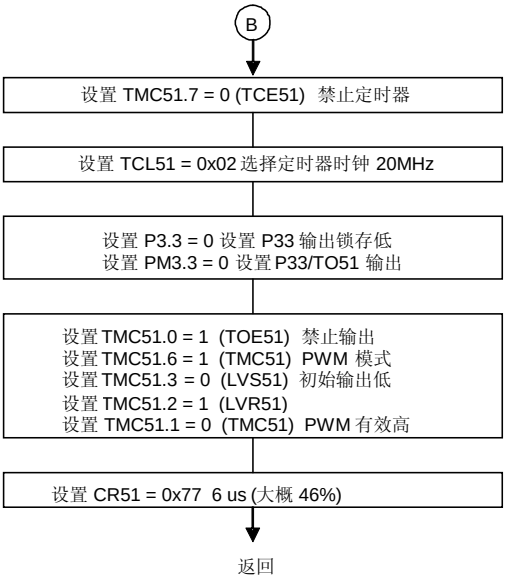
hdwinit( ) 函数结束后，启动代码调用用户程序的 main( ) 函数。启动 main( ) 函数，所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

4.3.2 AD_Init( ) – 初始化 A/D 转换器



AD_Init() 函数由 SystemInit()函数调用初始化 A/D 转换器。A/D 转换器建立从 ANI0/P20 引脚读取模拟输入。

4.3.3 TM51_Init( ) – 初始化 TM51 定时器/计数器用于 PWM 产生



TM51_Init() 函数，同样由 SystemInit()调用，建立定时器 51 用作 PWM 输出。当写控制寄存器时，最初定时器允许位设置为零，禁止定时器。

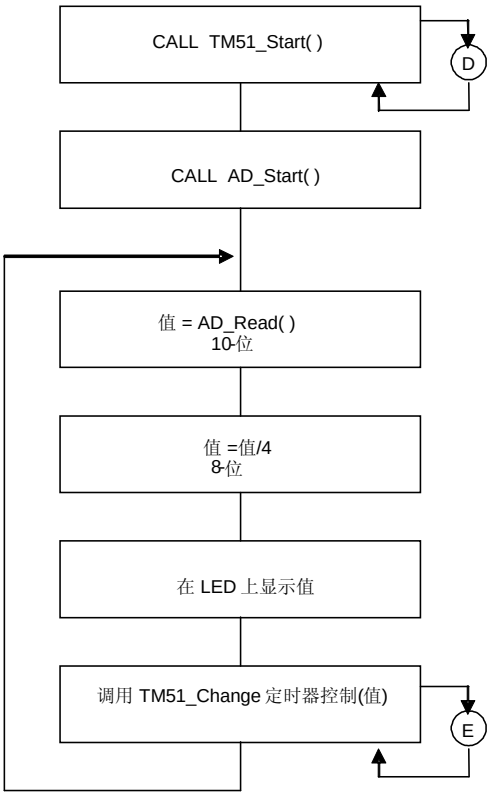
TCL51 设置为 0x02 选择 fPRS，主时钟 20 MHz ，作为定时器 51 的时钟输入。TM51 每 0.05 us 计数。TM51 每 256 计数值计数，结果每 256* 0.05us 或每 12.8us 一周期。PWM 周期频率为 78.125 KHz。

使用 P33/TO51 引脚输出 PWM 波形，所以 P3.3 输出锁存设为零，并且端口模式寄存器 PM3 的位 3 设为零将 P33/TO51 作为输出引脚。

TMC51 设置允许 TO51 输出，设置定时器 51 为 PWM 模式，默认输出低，并且 PWM 操作为有效高电平模式。

CR51 设为 0x77 或 119，结果 256 输出中 118 脉冲为高，或大概为 46%。

4.3.4 Main( ) – 主程序 - D/A 转换使用 PWM



程序启动，调用 TM51_Start() 程序启动 PWM 输出到 D/A 转换器， CR51 初值设为 0x77，结果是 118/256 * 5 V 或大概 2.30 V 的初始电压。调用 AD_Start()启动 A/D 转换器。

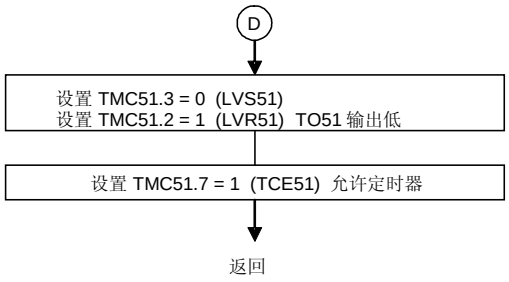
程序重复读取 A/D 转换器输入获取 10 位 A/D 值。值缩短为 8 位，并在 LED 上显示。

调用 TM51_ChangeTimerCondition(value) 程序，设置 CR51 寄存器值符合从 A/D 转换器读取的模拟输入。如果和之前设置的值不同，PWM 输出将是一个新占空比，高部分从 A/D 转换器读取的值。

A/D 转换器提供 00 到 FF 的输入。设置 CR51 寄存器，结果 PWM 为 0/256 到 255/256 占空比，或 D/A 转换器输出从 0 V 到 4.98 V。

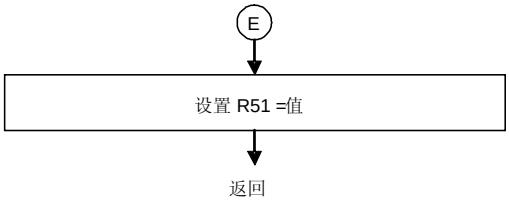
D/A 转换器电压将跟随 A/D 输入电压。

4.3.5 TM51_Start( ) – 启动定时器 51 操作



TM51_Start() 程序设置 TMC51 寄存器，控制定时器 51 输出低，并且允许定时器。

4.3.6 TM51_ChangeTimerCondition(value) – 设置 CR51



TM51_ChangeTimerCondition(value) 程序写入 CR51 比较寄存器值，改变 PWM 占空比。这个不需要停止 TM51 定时器。

4.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言 或汇编语言代码来管理 NEC 微控制器外设。请参见使用 Applilet 版本的附录。

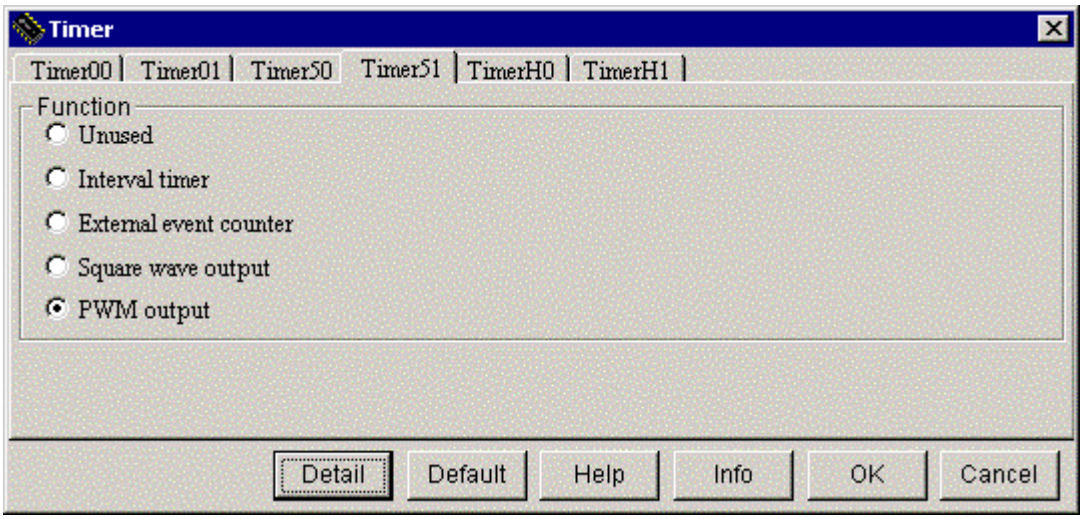
Applilet 用于创建基础的初始化代码和定时器 51 的驱动代码，读取 A/D 交换器代码，并且可以管理 LED 显示输出的 I/O 端口。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 51 的代码和产品程序列表描述如下。

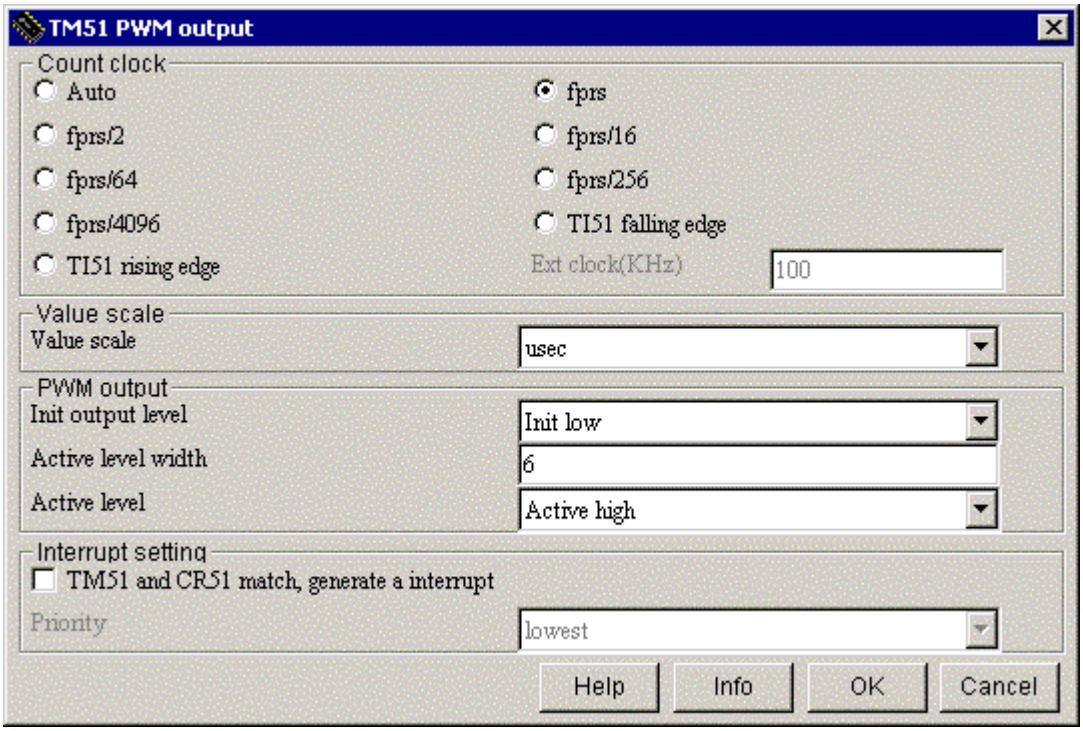
启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同的外设模块的界面。

4.4.1 定时器 51 的 Applilet 配置

选择 “Timer”，显示各种定时器界面，选择定时器 51 并将其设置为 PWM 模式。



一旦选择使用定时器 51 ，“Detail” 按钮可用于设置定时器 51 选择模式。点击“Detail” 按钮出现定时器 51 对话框设置 PWM。



这里是选择定时器 51 的操作细节。我们选择 fprs 时钟输入，提供最快的 PWM 频率。20 MHz 的 fprs ，PWM 频率为 78.125 KHz，PWM 周期为 12.8us。

我们设置 PWM 输出起始为低，高宽度为 6us。提供 6/12.8 或大概 46% 高电平。因为 PWM 宽度将改变与模拟输入相同，初值不是真实内容。

我们不希望当 TM51 和 CR51 匹配时定时器产生中断，所以不勾选左侧对话框。

4.4.2 Applilet 产生代码

一旦定时器对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其功能，允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码、中断服务程序和定时器 51 驱动子程序。

当“Generate”按钮按下，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持定时器 51，Applilet 产生 timer.h, timer.c 和 timer_user.c，并且其他几个和定时器无关。

4.4.3 Applilet 为定时器 51 产生的文件和函数

支持定时器 51 所产生的代码在文件 timer.h, timer.c 和 timer_user.c 中。它们包括以下内容。

4.4.3.1 Timer.h

头文件 timer.h 包含了控制定时器 51 的函数声明，和定时器 51 初始化值的定义。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。

4.4.3.2 Timer.c

源文件 Timer.c 包含了 Applilet 为定时器 51 产生的下列函数：

void TM51_Init(void);
TM51_Init()程序初始化定时器 51 外设，在 Applilet 定时器 51 对话框中设定。

void TM51_Start(void);
TM51_Start()程序启动定时器 51 操作。

void TM51_Stop(void);
TM51_Stop()程序停止定时器 00 操作。此程序不能用于演示程序 。

MD_STATUS TM51_ChangeTimerCondition(UCHAR value);
TM51_ChangeTimerCondition() 函数改变定时器 51 比较寄存器 CR51 的值，并且因此改变 PWM 高电平时间和占空比。

4.4.3.3 Timer_user.c

源程序 timer_user.c 给用户提供了一些框架函数。这些函数是产生的空函数，允许用户增加自己的应用代码。

4.4.4 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
Ad.h	A/D 转换器定义
Ad.c	A/D 转换器程序
Ad_user.c	用户 A/D 转换器程序
System.h	时钟关联定义
System.c	Clock_Init()函数
Port.h	定义默认 I/O 端口状态
Port.c	PORT_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

4.4.5 不由 Applilet 产生的演示程序

演示程序包括以下不由 Applilet 产生的文件。

文件	功能
Led_0537.h	7 段 LED 头文件
Led_0537.c	7 段 LED 上显示数据头文件

4.5 演示平台

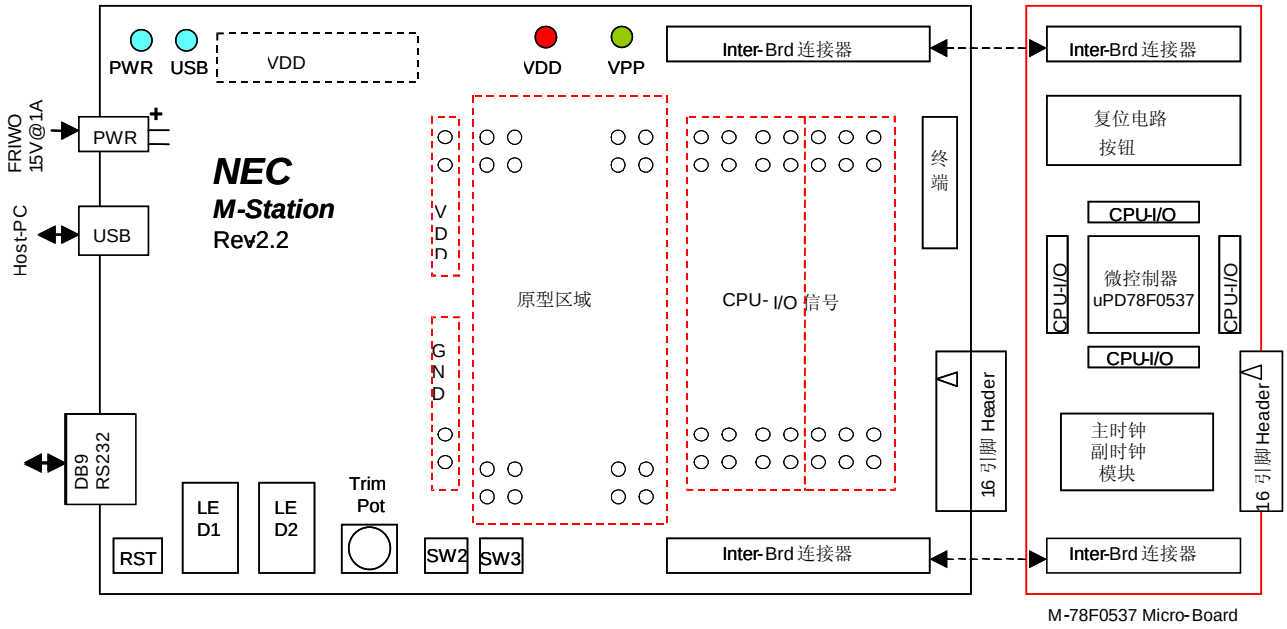
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

4.5.1 资源

为了演示该程序，需要使用以下资源：

- 焊有 uPD78F0537 8 位微控制器的 M-78F0537 Micro-Board
- M-Station II 仿真系统，M-Station II 上的资源：
  - 7-段 LED LED1 和 LED2
  - 电位表给 P20/ANI 提供可变模拟输入电压
  - 硬件增加低通滤波器

以上硬件列表的详细说明，请参见用户手册，该手册可以向 NEC 索取。

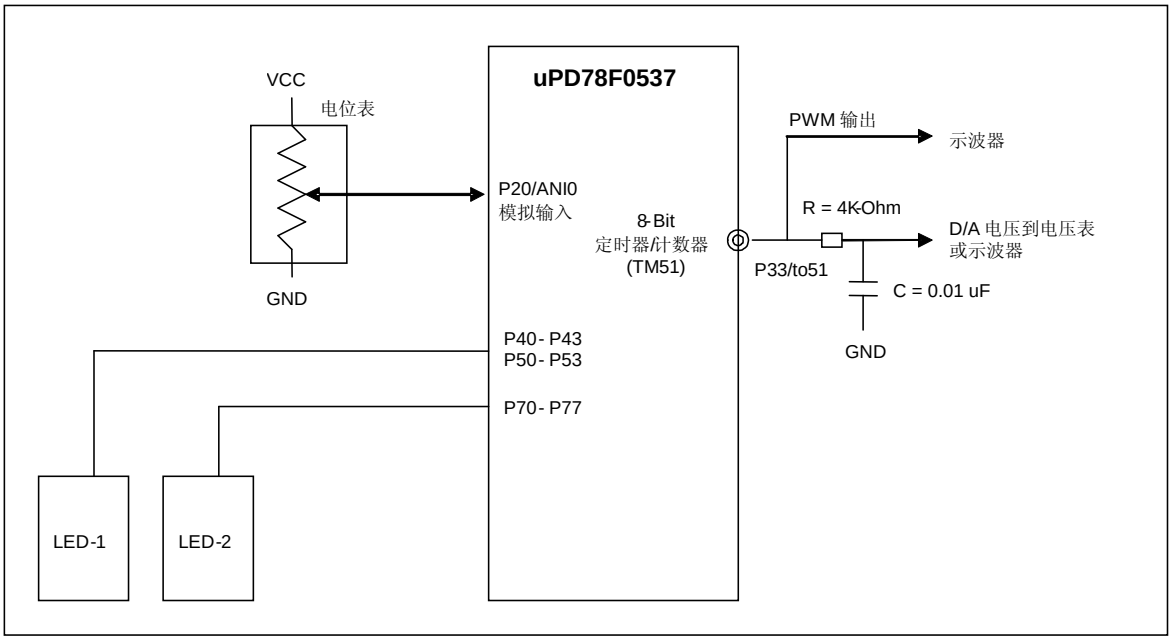


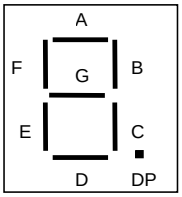
4.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 打开电位表，在 LED 上观察输入电平
- 在示波器上观察 PWM 波形输出
- 使用电压表或示波器观察 D/A 电压

4.6 硬件框图





LED-1 和 LED -2

段	LED-1	LED-2
A	P40	P70
B	P41	P71
C	P42	P72
D	P43	P73
E	P50	P74
F	P51	P75
G	P52	P76
DP	P53	P77

由定时器/计数器选择时钟产生固定频率 PWM 信号。PWM 脉冲宽度依据比较寄存器值改变，从 0%到 100%。PWM 脉冲宽度与模拟信号振幅成比例。例如，PWM 宽度为 50% 占空比与低通滤波器输出 1/2VDD 成比例。

低通滤波器的响应时间根据 RC 时间常数。RC 时间常数= 1/(2*Pi*频率)。电阻和电容值大概为 4KHz 带宽。PWM 频率为 78KHz，4KHz 低通滤波器截止频率。

4.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

列表中的文件在附录中。

文件	目的	由 Applilet 产生	由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟关联定义	是	否
Systeminit.c	SystemInit() 和 hdwinit() 函数	是	否
System.c	Clock_Init()函数	是	否
Timer.h	定时器关联定义	是	否
Timer.c	定时器函数	是	否
TIMER_user.c	定时器中断的用户代码	是	否
Port.h	端口关联定义	是	否
Port.c	Port_Init()函数	是	否
Led_0537.h	LED 显示定义	--	是
Led_0537.c	LED 显示函数	--	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

5 遥控载波发生使用TM51 和TMH1

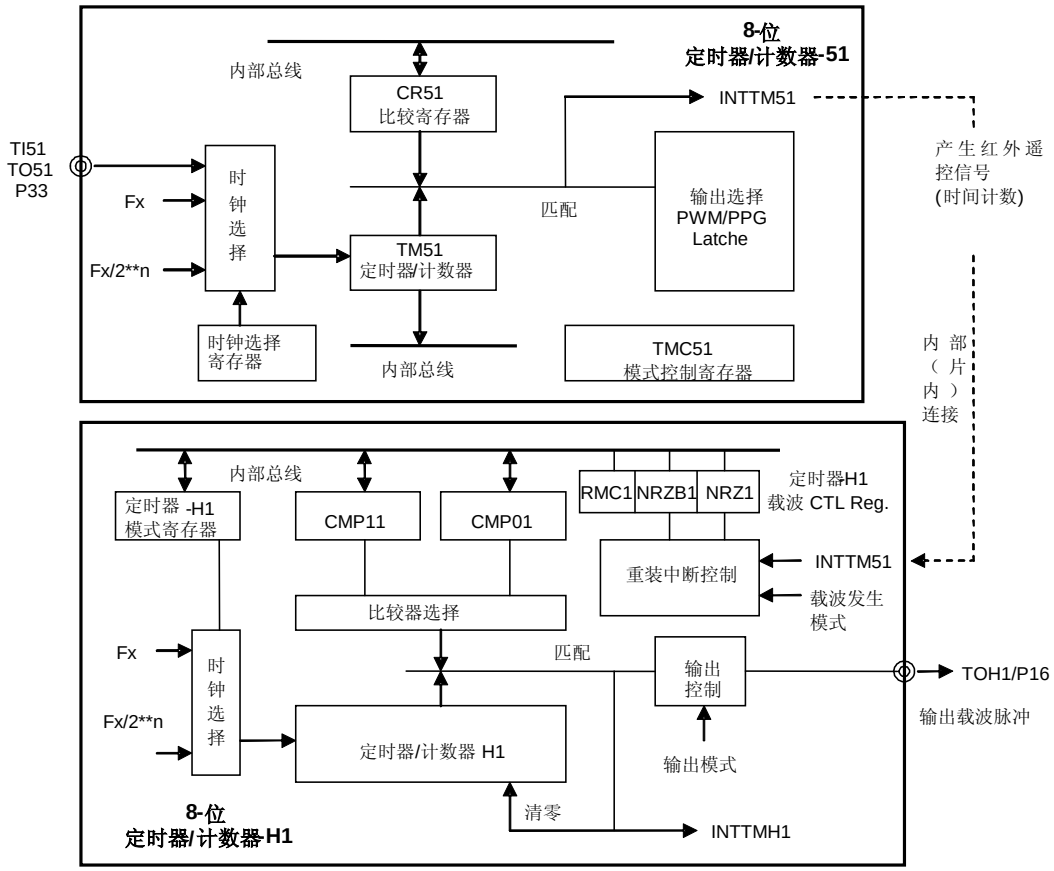
载波信号被用于红外遥控。典型的，一对 8 位定时器/计数器被用作产生红外时间计数信号。8 位定时器/计数器 H1(TMH1) 产生载波时钟，输出周期由另一 8 位定时器/计数器 51 (TM51)设置。“载波时钟发生器 TMH1”定时器/计数器值与 2 比较寄存器值比较，设置高电平和低电平载波脉冲。

几种 NEC 78K0 微控制器提供一对 8 位定时器/计数器支持载波输出发生器功能。除 TV 或 VCR 遥控外，载波发生器也可用于遥控的控制系统。

5.1 载波输出发生器特征

8 位定时器/计数器 51 和 8 位定时器/计数器 H1 用作发生载波脉冲输出。当定时器/计数器 H1 产生载波脉冲输出时，定时器/计数器 51 产生遥控信号(时间计数)。在载波发生器模式下，定时器 H1 和 51 提供以下特征：

- 可变载波频率
- 可变载波占空比
- 自变量载波开和载波关时间
- 时间周期到时自动改变载波开/载波关状态
- 自动改变载波开/载波关状态，用于规则载波脉冲产生
- 选项载波脉冲沿中断
- 载波开/关时间中断



- 比较寄存器：

    - CMP01　　设置载波脉冲低电平宽度
    - CMP11　　设置载波脉冲高电平宽度

- 载波脉冲输出控制

    - 由定时器/计数器 51 中断请求信号 INTTM51 控制
    - 也可由载波控制寄存器的 RMC1 位、 NRZB1 位和 NRZ1 位控制

        - NRZ1　　= 0　　禁止载波输出
          = 1　　当 RMC1 = 1 时，允许载波输出

        - RMC1 = 0　　NRZB1 = 0　　低电平输出
        - RMC1 = 0　　NRZB1 = 1　　高电平输出
        - RMC1 = 1　　NRZB1 = 0　　低电平输出
        - RMC1 = 1　　NRZB1 = 1　　载波输出

- 载波发生步骤

    - 设置操作寄存器

        - 设 CMP01 比较值
        - 设 CMP11 比较值

    - 设置控制寄存器

        - RMC1 = 1　　允许遥控输出
        - NRZB1 = 0/1　　允许载波输出

    - 设置定时器/计数器 H1 和定时器/计数器 51 模式寄存器：定时器开始计数

    - CMP01 = 定时器/计数器 H1　　相等中断 INTTMH1

        - 清定时器/计数器 H1
        - 转换比较寄存器到 CMP11 用于下一次比较

    - CMP11 = 定时器/计数器 H1　　相等中断 INTTMH1

        - 清定时器/计数器 H1
        - 转换比较寄存器到 CMP01 用于下一次比较

    - 重复比较清零和转换比较寄存器步骤实现产生载波时钟

- 在 TOH1 输出载波时钟

    - INTTM51 中断信号与 INTTMH1 中断信号同步

        - 当 INTTMH1　　转换 NRZB1 到 NRZ1
          往 NRZB1 写入下一值

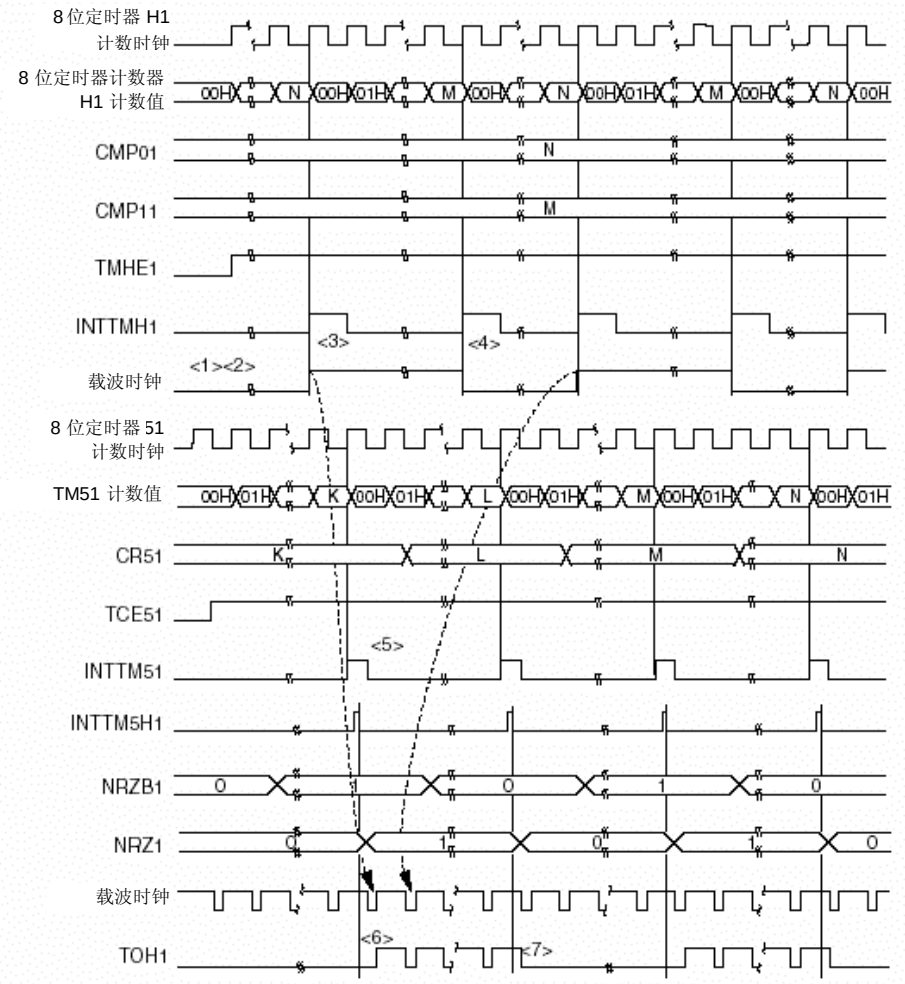
    - 如果 NRZ1 = 1,　　TOH1 输出 = 载波时钟
      NRZ1 = 0,　　TOH1 输出= 0

- 载波时钟输出周期

  - 如果 CMP01 = N
    CMP11 = M
    载波时钟频率= Fx,

  - 载波时钟输出周期 = (N + M + 2)/Fx
  - 载波时钟占空比 = (高电平宽度)/(载波时钟宽度) = (M + 1)/(N + M + 2)

- 载波时钟输出 CMP01 = N
  CMP11 = M



配置定时器 H1 和定时器 51 用于载波发生，按以下步骤。

- 设置载波时钟，载波模式和允许 TMHMD1 输出
- 如果使用中断，设置载波中断优先级，清中断标志，并且非屏蔽中断
- 设置输出引脚和锁存为低用于载波输出
- 设置开/关状态初值并使用 TMCYC1 允许载波
- 设置 CMP01 和 CMP11 寄存器用于载波占空比
- 设置 TM51 用作间隔定时器，使用 TCL51 设置 TM51 时钟
- 设置 TM51 中断优先级，清中断标志，并且非屏蔽中断
- 设 CR51 起始位时间

这时载波发生器已做好准备。启动载波发生：

- 启动定时器 H1 产生内部载波
- 启动定时器 51 计数

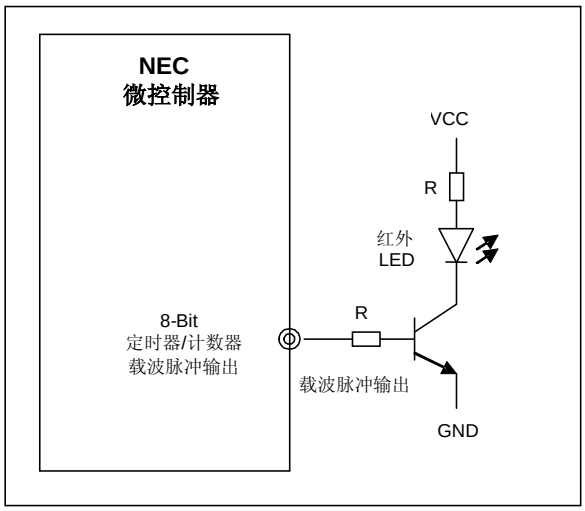
由 TM51 产生中断，信号下一半位周期的开始，并更新调整载波开/关状态。中断服务程序，下一半位周期的时序和开/关状态。

5.2 程序描述和说明

演示程序将产生由比较寄存器 CMP01 和 CMP11 设置的载波。定时器 51 被用作载波脉冲打开和关闭的时间间隔。载波输出连接到一个用于红外信号传输的发射二极管。TOH1 输出载波时钟，可用示波器观察。

在实际遥控应用中，它通常有按键置于 78K0 系列微控制器。遥控单元依据按键输入产生特殊载波时钟。例如，如果按键"5" 按下，它将产生载波时钟等同于选择 TV 的 5 频道。载波时钟与选择 TV 设置的载波时钟匹配。

本演示程序非实现真实的遥控单元。那个程序，要设置所有必要的计数器和载波时钟发生操作，以便用户能改变和修改演示程序来创建真实的遥控单元用于选择的 TV 或 VCR。



说明：

- TMH1 建立 8.5 us 高，18 us 低载波脉冲
- TM51 用于产生 0/1 位，有以下特征：
- 0 位 = 载波打开 565us，关闭 565us
- 1 位 = 载波打开 565us，关闭 1.685 ms

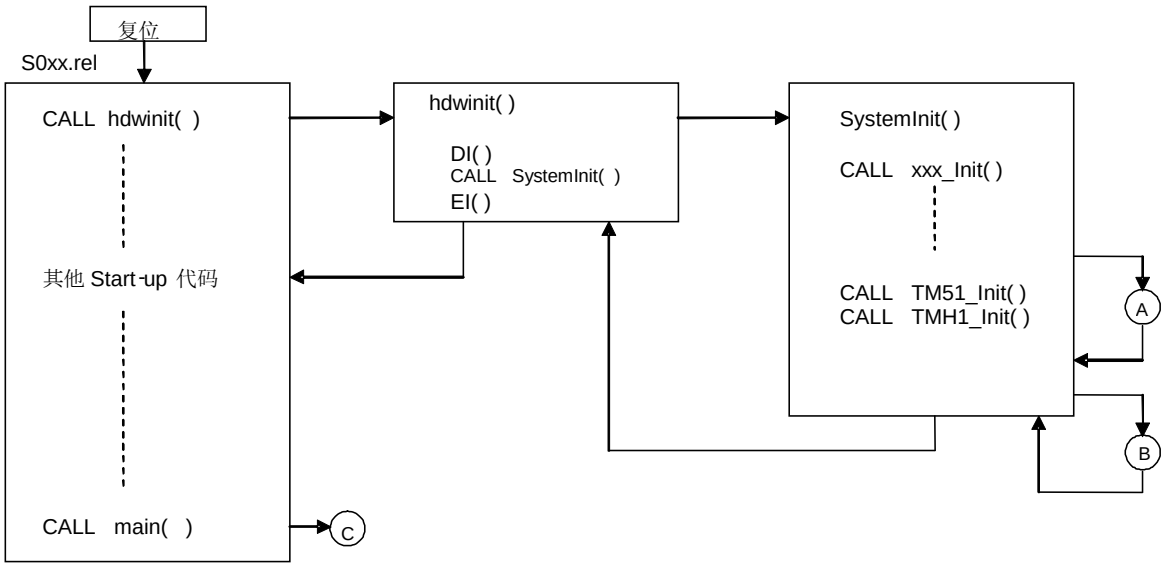
5.3 软件流程图

演示程序由以下主要几部分组成：

- o 程序的初始化代码，在 main() 程序启动前调用
- o 主程序循环，建立数据传输和启动发送
- o 由 Applilet 产生的子程序启动 TMH1 和 TM51，停止和改变间隔
- o 用户代码子程序用于处理 TM51 中断(Applilet 产生框架中断服务程序，增加用户代码)

这里的流程图描述初始化、主程序、TMH1 和 TM51 相关子程序 和 TM51 的中断服务程序。

5.3.1 启动程序和初始化

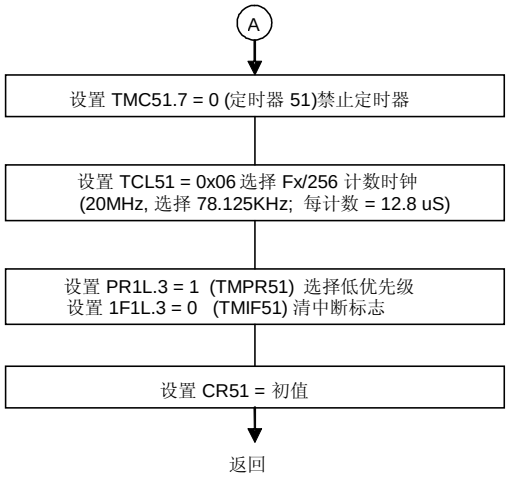


C 语言写的 78K0 程序启动代码由结果代码文件提供，例如 s0l.rel，连接到用户程序。调用启动代码的函数名称是 hdwinit( )，用户可在这里放置任意的硬件初始化代码。

当 Applilet 产生 78 K0 的 C 语言，hdwinit( ) 函数自动运行增加到用户程序，并调用 SystemInit( ) 函数。这个 SystemInit( ) 函数用于每个外设调用初始化程序。

hdwinit( ) 函数结束后，启动代码调用用户程序的 main( ) 函数。启动 main( ) 函数，所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

5.3.2 TM51_Init( ) – 定时器 51 初始化



TM51_Init() 函数由 SystemInit()调用，初始化定时器 51 用于间隔定时器操作。定时器 51 用作高电平时间位（载波脉冲开）和低电平时间位（载波脉冲关），决定位按 1 或 0 传输。

TMC51 位 7，TME51 在写控制寄存器时设置为 0 禁止定时器。

TCL51 设为 0x06，选择 f_{PRS}/256 作为 TM51 计数时钟。主系统时钟为 20 MHz，则 TM51 计数时钟为 20,000,000/256 或 78.126 KHz 。TM51 每次计数为 12.8 us。

用于定时器 51 的中断控制寄存器设为低优先级，并清中断标志。中断屏蔽定义禁止状态。

CR51 设间隔值，这里并不重要，使用定时器 51 前将重写 CR51。

5.3.3 TMH1_Init( ) – 定时器 H1 初始化

TMH1_Init()函数由 SystemInit()调用，初始化定时器 H1 用于载波发生操作。写控制寄存器时禁止定时器。

TMHMD1 设选择 f_{PRS}/4 作为定时器时钟。20 MHz 时，则 TMH1 计数时钟为 5 MHz 。TMH1 每次计数为 0.2 us。

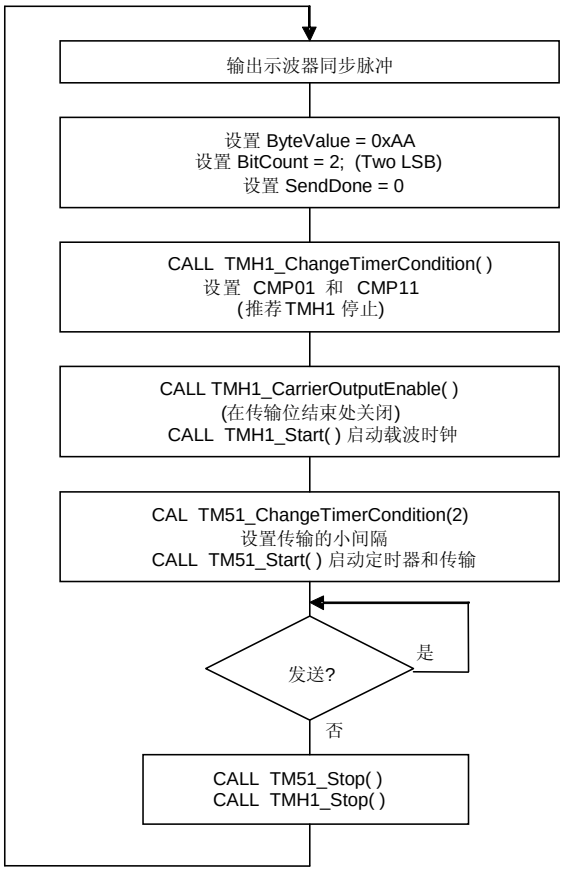
用于定时器 H1 的中断控制寄存器设为低优先级，并清中断标志。我们不需要定时器 H1 产生中断，只需创建载波脉冲。

P16/TOH1 为载波输出引脚， P1.6 输出锁存设为 0，并且设端口模式寄存器 PM1 的位 6 为 0，配置 P16/TOH1 作为输出。TMHMD1 位 0 (TOEN1) 设置为 1 允许定时器输出。

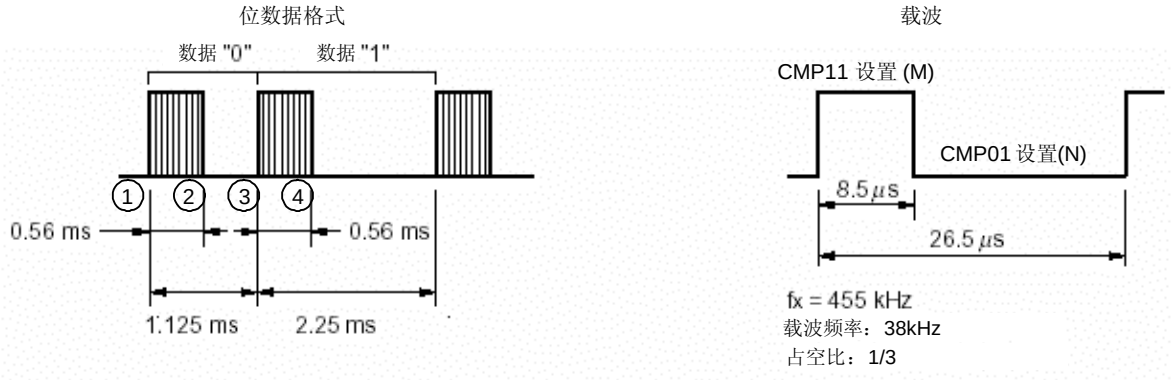
TMCYC1 位 2 (RMC1) 设置为 1 允许载波产生，并且 NRZB1 位设为 1 设置载波脉冲打开。

CMP01 寄存器设为 0x59 ， 90 * 0.2 us 或 18us 载波低电平时间；CMP11 寄存器设为 0x29 ， 42 * 0.2 或 8.4 us 载波高电平时间。载波大概是 1/3 高和 2/3 低占空比。

5.3.4 Main( ) – 主程序 – 载波脉冲产生



演示程序发生载波，如下所示。



① CMP51 设为 0.56 ms
② CMP51 设为 0.56 ms
③ CMP51 设为 0.56 ms
④ CMP51 设为 1.685 ms (2.25 - 0.56 = 1.685)

主程序产生符合 2 位数据 0xAA (10101010)的波形，以 LSB-first 发送输出。用示波器观察，程序在端口 P77 产生一个同步位。程序设置发送数据 0xAA，并且设置位计数为 2，设置 SendDone 标志为 0。

调用 TMH1_ChangeTimerCondition(*array, 2) 程序写入 CMP01 和 CMP11 值 ， 其决定载波时序。这些值不能修改，但推荐定时器 H1 停止时重写它们。

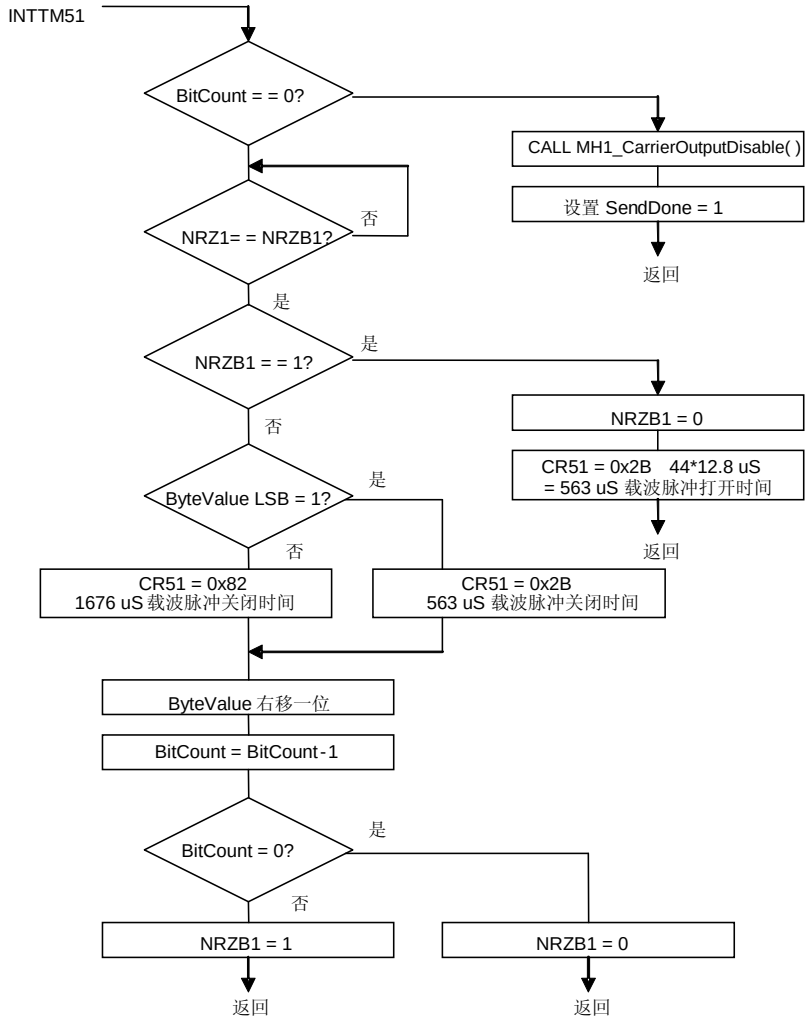
调用 TMH1_CarrierOutputEnable() 设置载波输出打开，和调用 TMH1_Start() 启动定时器 H1 。定时器 H1 将内部产生波形，但因为 NRZ1 位为 0，没有载波输出。

调用 TM51_ChangeTimerCondition(2) 建立定时器 51 ，CR51 值为 2；在最初的定时器 51 中断 INTTM51 前产生一个短时间(3 * 12.8 或 38.4 us) 。调用 TM51_Start() 启动定时器 51。

主程序在此一直等待，直到 SendDone 标志不再为 0，并且调用 TM51_Stop() 和 TMH1_Stop()停止定时器。此时发送 2 位。

所有的位发送完，产生中断 INTTM51，调用中断服务程序 MD_INTTM51。

5.3.5 MD_INTTM51( ) - INTTM51 中断服务程序



当 TM51 和 CR51 匹配时，产生 INTTM51 。因为定时器 H1 设为载波发生模式，INTTM51 实际改变 INTTM5H1 中断，直到定时器 H1 计数时钟的下一个下降沿，同步中断和切除载波。

一旦发生 INTTM5H1，NRZB1 位的值(最初为 1) 传送给 NRZ1 位，并在 P16/TOH1 发送载波。

位低部分发送完后调用中断服务程序，BitCount 为 0。调用 TMH1_CarrierOutputDisable() 函数，并且 SendDone 变量设为 1。

位发送完后，中断服务程序等待 NRZ1 变为和 NRZB1 相等；它根据载波状态决定下一发送如果设置 NRZB1 位。

如果 NRZB1 为高，我们启动位高部分发送，563us 载波。NRZB1 设为 0 改变 NRZ1 下一发送为 0，并且 CR51 设为 0x2B， 563 us 延时直到中断。此时，中断服务程序返回主程序。

如果 NRZB1 为低，我们启动位低部分发送。如果当前 LSB 为 1，低时间短(563 us)；当前 LSB 为 0，低时间长(1676 us)。CR51 设为适当值。

当前字节移到下一位，准备发送下一 LSB，并且 BitCount 变量递减。如果不是最后一位 (BitCount 不为零)，NRZB1 设为 1，NRZ1 将在下一中断为 1，并开始下一位高部分载波。

如果是最后一位，NRZB1 (left) 设为 0。下一中断，NRZ1 不为 1，并且不输出载波。下一中断 BitCount 为 0，并且禁止载波发送，设 SendData 为 1。

5.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言 或汇编语言代码来管理 NEC 微控制器外设。请参见使用 Applilet 版本的附录。

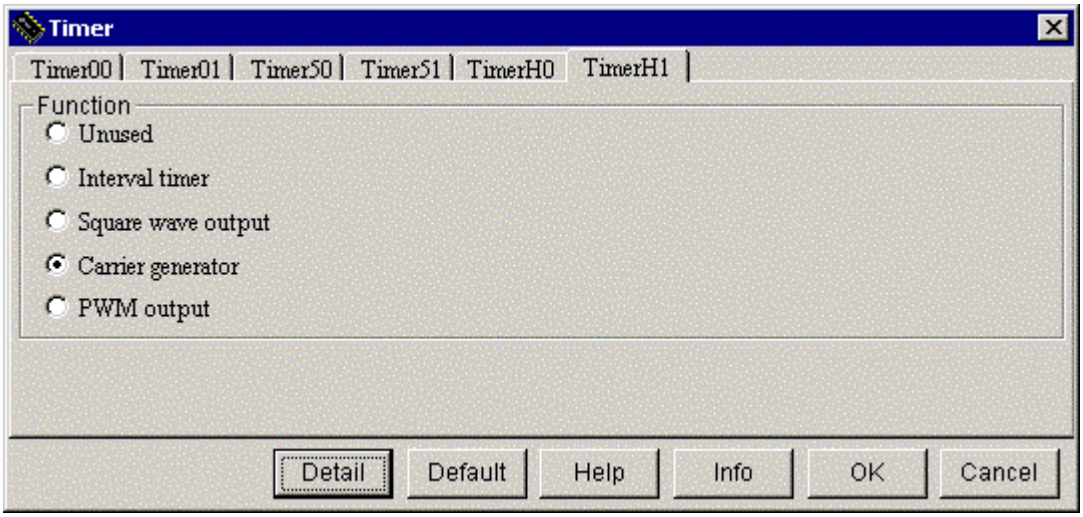
Applilet 用于创建基础的初始化代码和定时器 51 和定时器 H1 的驱动代码。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 51 和定时器 H1 载波发生的代码和产品程序列表描述如下。

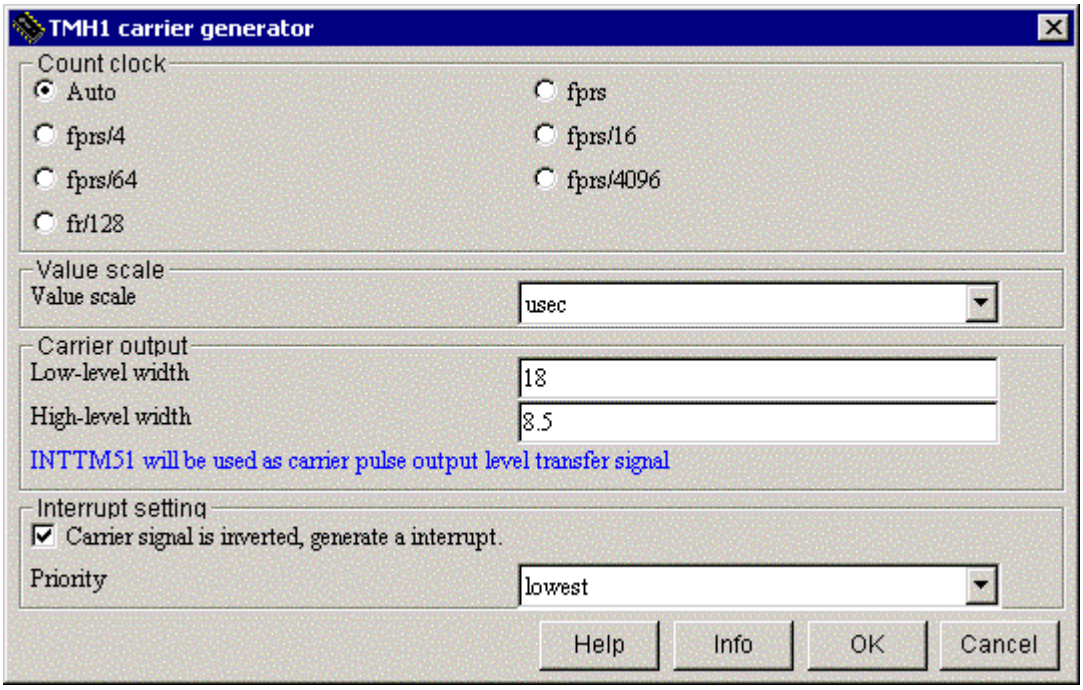
启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同的外设模块的界面。

5.4.1 定时器 H1 载波发生的 Applilet 配置

选择 “Timer”，选择定时器 H1 并设置为载波发生模式。



一旦选择使用定时器 H1，“Detail” 按钮可用于设置定时器 H1 选择模式。点击“Detail” 按钮出现对话框设置载波发生。

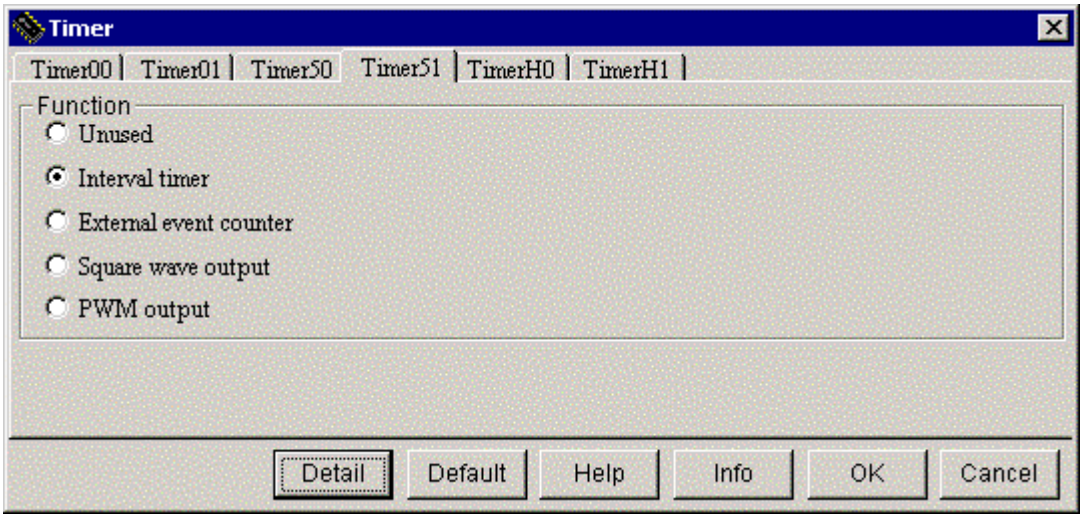


这里是选择定时器 H1 的操作细节。我们希望定时器产生 8.5us 高，18us 低载波。值刻度选择为 us，对话框中的高低电平输入适当的值。计数时钟设为“Auto”， Applilet 将产生时钟分频寄存器适当的值达到想要的宽度。

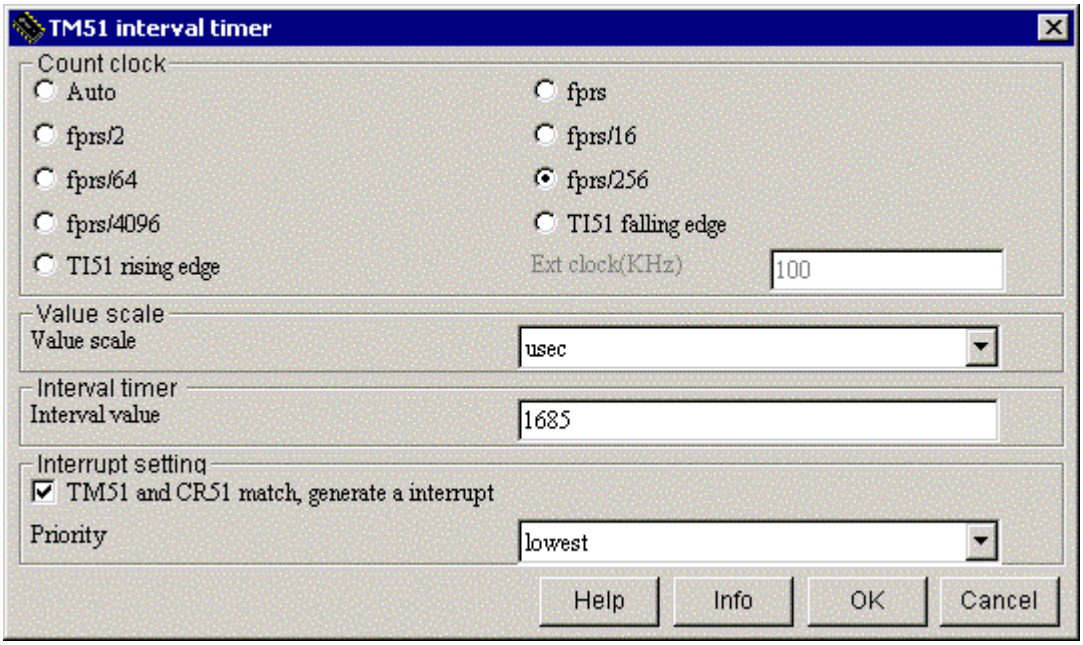
载波倒置时可以不需要产生中断，但我们需要选择中断产生。中断服务程序用于输出位倒置状态，可观察内部载波。

5.4.2 定时器 51 的 Applilet 配置

选择 “Timer”，显示各种定时器 51 可用设置界面，定时器 51 设为间隔定时器操作。



一旦选择使用定时器 51 ，“Detail” 按钮可用于设置定时器 51 选择模式。点击“Detail” 按钮出现定时器 51 对话框设为间隔定时器。



这里是选择定时器 51 的操作细节。我们希望定时器产生间隔时间从 563us 到 1685us。设置间隔值，计数时钟设为 fprs/256。结果为 20,000,000/256 或 78125 Hz 计数时钟，或每一时钟为 12.8 us。

因为我们希望定时器比较时产生中断，勾选 TM51（定时器 51 计数寄存器）和 CR51（定时器 51 比较寄存器）匹配时产生中断。中断设为低优先级。

5.4.3 Applilet 产生的代码

一旦定时器 对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其功能，允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码、中断服务程序和定时器 H1 和定时器 51 驱动子程序。

当“Generate”按钮按下，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持定时器 H1 和定时器 51，Applilet 产生 timer.h, timer.c 和 timer_user.c，并且其他几个和定时器无关。

5.4.4 Applilet 为定时器 H1 和定时器 51 产生的文件和函数

支持定时器 H1 和定时器 51 所产生的代码在文件 timer.h, timer.c 和 timer_user.c 中。它们包括以下内容。

5.4.4.1 Timer.h

头文件 timer.h 包含了控制定时器 H1 和定时器 51 的函数声明，和定时器 H1 和定时器 51 初始化值的定义。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。

5.4.4.2 Timer.c

源文件 Timer.c 包含了 Applilet 为定时器 H1 和定时器 51 产生的下列函数：

void TMH1_Init(void);
TMH1_Init()程序初始化定时器 H1 外设，在 Applilet 定时器 H1 对话框中设定。

void TMH1_Start(void);
TMH1_Start()程序启动定时器 H1 操作 并允许 INTTMH1。

void TMH1_Stop(void);
TMH1_Stop()程序停止定时器 H1 操作，并禁止定时器中断。

MD_STATUS TMH1_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
TMH1_ChangeTimerCondition()函数改变定时器 H1 比较寄存器 CMP01 和 CMP11 的值，并因此改变定时器 H1 的载波值。

array_reg 参数指向列值，放置一个或其他的比较值；array_num 参数为 1 (选择 CMP01) 或 2 (选择 CMP01 和 CMP11)。

void TMH1_CarrierOutputEnable();
TMH1_CarrierOutputEnable() 程序设置 NRZB1 位高，依据 NRZ1 和 RMC1 位状态，允许载波输出。

void TMH1_CarrierOutputDisable();
TMH1_CarrierOutputEnable() 程序设置 NRZB1 位低，载波输出低。

void TM51_Init(void);
TM51_Init() 程序初始化定时器 51 外设，在 Applilet 定时器 51 对话框中设定。

void TM51_Start(void);
TM51_Start() 程序允许定时器启动定时器 51 操作，并且允许定时器中断。

void TM51_Stop(void);
TM50_Stop() 程序禁止定时器停止定时器 51 操作，并且禁止中断。

MD_STATUS TM51_ChangeTimerCondition(UCHAR value);
TM51_ChangeTimerCondition() 函数改变定时器 51 比较寄存器 CR51 的值，并改变定时器间隔时间。

5.4.4.3 Timer_user.c

源程序 timer_user.c 给用户提供了一些框架函数。这些函数是产生的空函数，允许用户增加自己的应用代码。

__interrupt void MD_INTTM51( );
这个中断服务程序用于当 TM51 和 CR51 值匹配时产生定时器 51 中断 INTTM51。一旦定时器启动，由 CR51 指定的间隔时间到时产生中断。

由 Applilet 产生的中断服务程序为空。增加载波产生代码，按以上流程图描述。

__interrupt void MD_INTTMH1( );
这个中断服务程序用于当载波倒置时产生定时器 H1 中断 INTTMH1。Applilet 产生的程序为空，不需要增加任何载波发生代码。

5.4.5 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
System.h	时钟关联定义
System.c	Clock_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

5.4.6 不由 Applilet 产生的演示程序文件

演示程序不使用任何非 Applilet 产生的文件。

5.5 演示平台

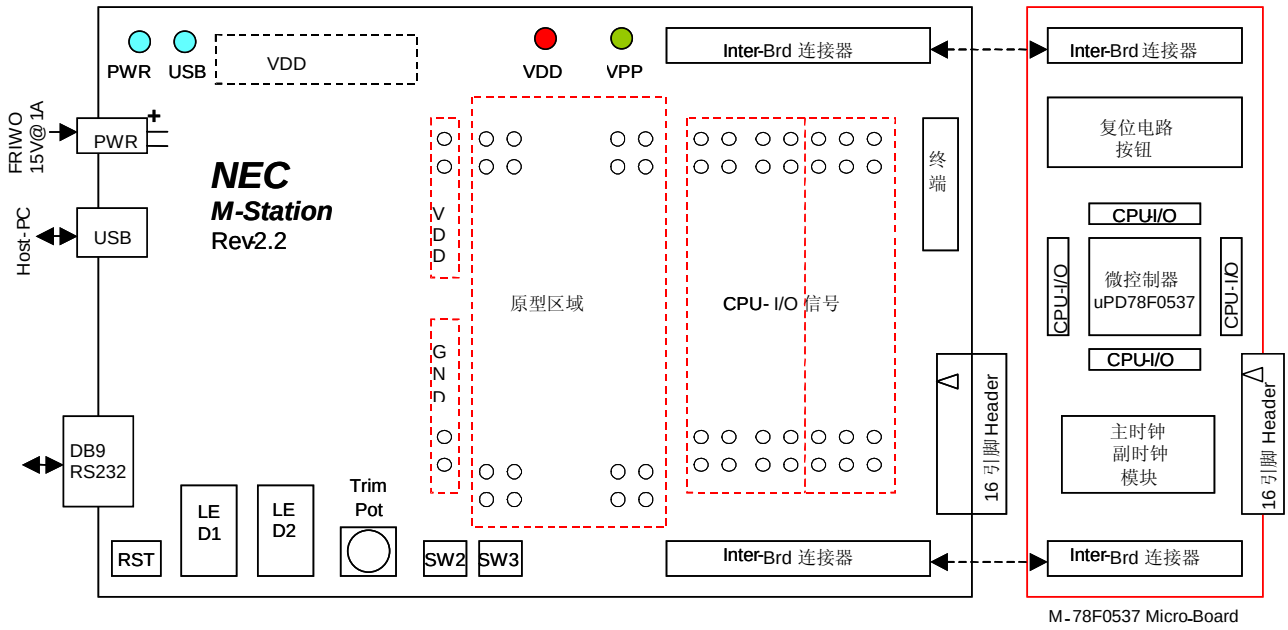
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

5.5.1 资源

为了演示该程序，需要使用以下资源：

- 焊有 uPD78F0537 8 位微控制器的 M-78F0537 Micro-Board
- M-Station II 仿真系统，M-Station II 上的资源：
  - 硬件增加红外发射二极管

以上硬件列表的详细说明，请参见用户手册，该手册可以向 NEC 索取。

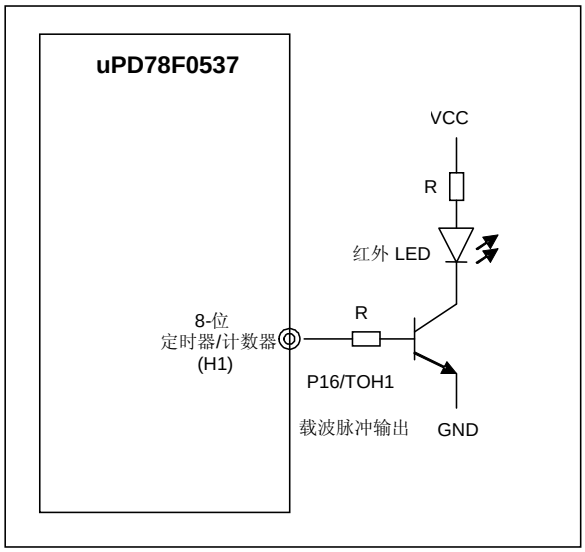


5.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 用示波器观察载波
  红外发射二极管可见

5.6 硬件框图



大多数红外遥控设备使用红外 LED 连接到 78K0 系列微控制器的载波脉冲输出发送数据。

红外 LED 显现载波脉冲发送。更多精确的波形通过示波器观察。发送红外控制 TV、VCRs、CD Player 等。不同的器件要求不同的编码格式和数据。演示程序使用专用波形发生用于载波脉冲。

5.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

这些文件列表在附录中。

文件	目的	由 Applilet 产生	可由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟关联定义	是	否
Systeminit.c	SystemInit() 和 hdwinit()函数	是	否
System.c	Clock_Init()函数	是	否
Timer.h	定时器关联定义	是	否
Timer.c	定时器函数	是	否
TIMER_user.c	定时器中断用户代码	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

6 PPG 输出音调发生使用16位定时器00 (TM00)

可编程脉冲发生 (PPG) 输出，78K0 系列微控制器大多数 8 位和 16 位定时器/计数器都支持。PPG 输出可为外设提供精准时钟。PPG 输出是定时器/计数器输出，可与定时器其他功能交替使用，例如 PWM、单脉冲输出和方波输出。

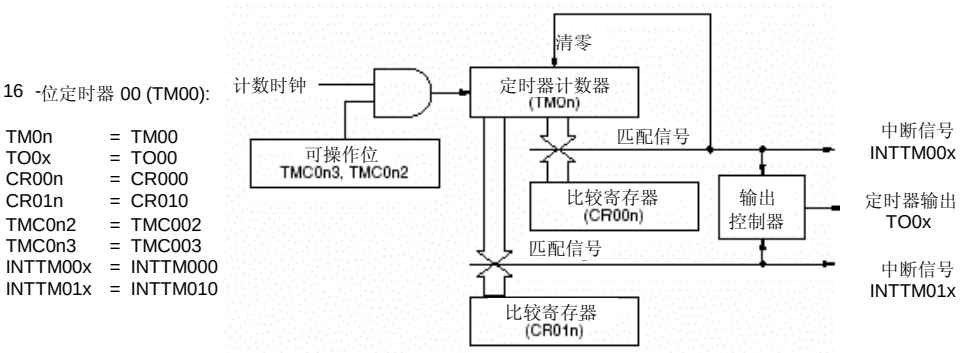
程序将使用 PPG 输出演示音调发生，由 16 位定时器/计数器输出驱动一个扬声器。但 PPG 功能与 PWM 输出、单脉冲输出和方波分享同样的产品，相同的电路使用定时器/计数器的各种输出用于演示不同的音调。

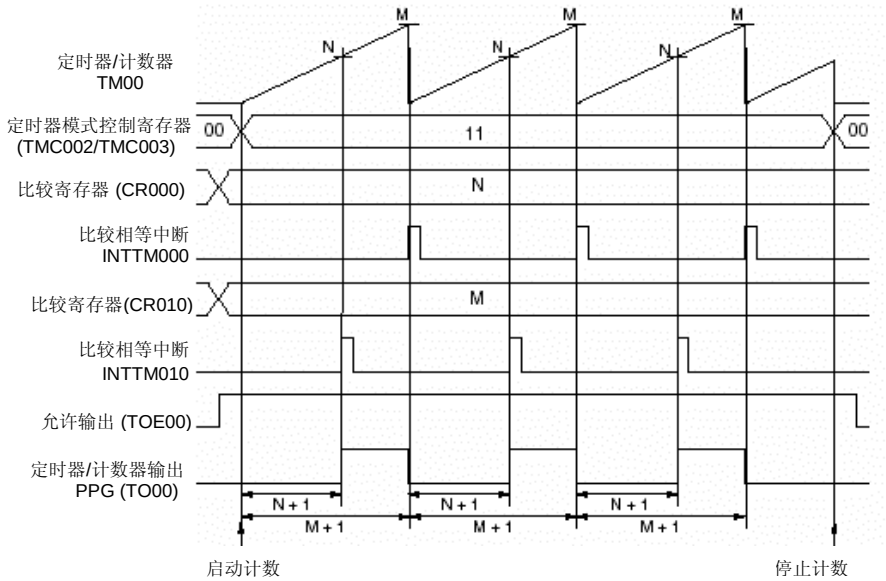
6.1 PPG 输出的特征

PPG 模式，16 位定时器提供以下特征：

- 使用 PPM00 寄存器，基于外部时钟分频可变时间
- 使用 CR010 寄存器，可变 PPG 周期
- 使用 CR000 寄存器，可变 PPG 占空比
- 设置初始输出电平、有效高或有效低输出
- 可在周期结束或有效时间结束选项中断

当定时器模式控制寄存器设为清零启动模式时，从定时器引脚(TO00)输出作为 PPG 信号，输出波形的脉冲宽度由比较寄存器(CR010)设置，周期由另一比较寄存器(CR000)设置。





- PPG 脉冲周期 = (M + 1) * 计数时钟周期
- PPG 占空比 = (N + 1)/(M = 1)

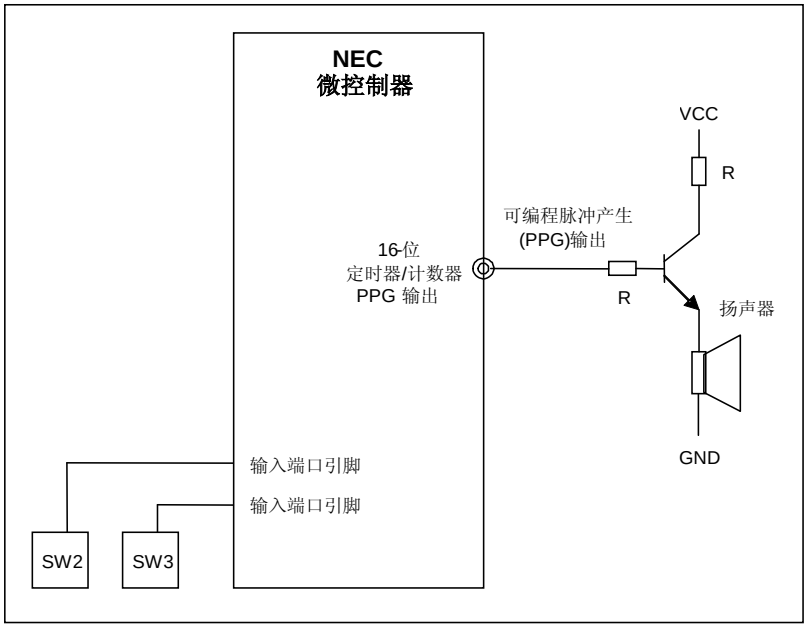
配置定时器用于 PPG 操作，按以下步骤：

- 使用 PRM00 设置基准时间
- 使用 CRC00 设置 CR000 和 CR010 作为比较寄存器
- 在 CR010 中设置周期
- 在 CR000 中设置有效时间
- 设置输出端口锁存和模式寄存器位为低，用于 PPG 输出
- 如果使用中断，设置中断优先级、清中断标志和非屏蔽中断
- 设置 TOC00 寄存器用于初始输出电平、允许输出和 CR000/CR010 匹配时倒置
- 允许定时器由定时器模式寄存器 TMC00 设置为清零启动模式，比较 TM00 和 CR000

6.2 程序描述和说明

演示程序将从 PPG 输出产生声音。16 位定时器/计数器配置为产生 PPG 输出。扬声器附在 PPG 输出处。

最初， 586 Hz 音调，产生 440 Hz 和 880 Hz 泛音。当按下按键 (SW2)，528 Hz 音调以 440 Hz 和 660 Hz 泛音输出。当按下另一按键 (SW3)，754 Hz 音调以 660 Hz 和 880 Hz 泛音输出到 PPG 输出。扬声器将产生低音和高音多音调声音。



说明：

- TM00 设为 PPG 模式，有效高，最初输出低，无中断
- TM00 时间基准时钟设为 5 MHz
- 脉冲产生：
  - 1704 us (586 Hz)，66.7%/33.3% 占空比 (440 和 880 Hz)
  - 1893.4 us (528 Hz)，60%/40%占空比(440 和 660 Hz)
  - 1325.4 us (754 Hz)，42.8%/57.2%占空比(880 和 660 Hz)

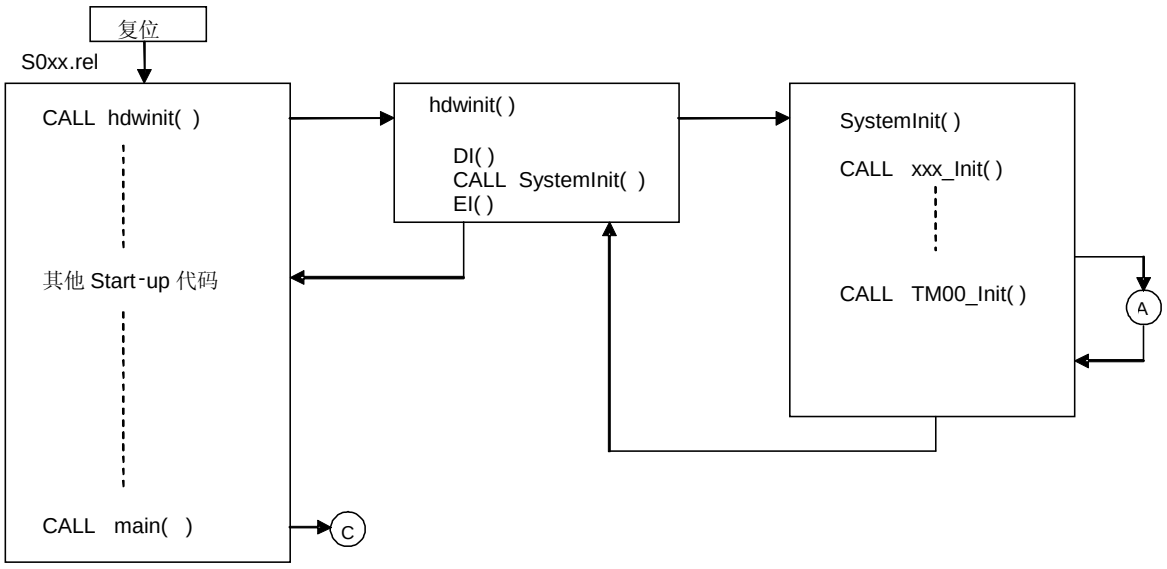
6.3 软件流程图

演示程序由以下主要几部分组成：

- 程序的初始化代码，在 main() 程序启动前调用
- 主程序循环，检查按键状态和改变 TM00 的时间
- 由 Applilet 产生的子程序启动 TM00，停止和改变间隔
- 按键输入、LED 显示输出子程序

这里的流程图描述初始化、主程序和 TM00 相关子程序。

6.3.1 启动程序和初始化

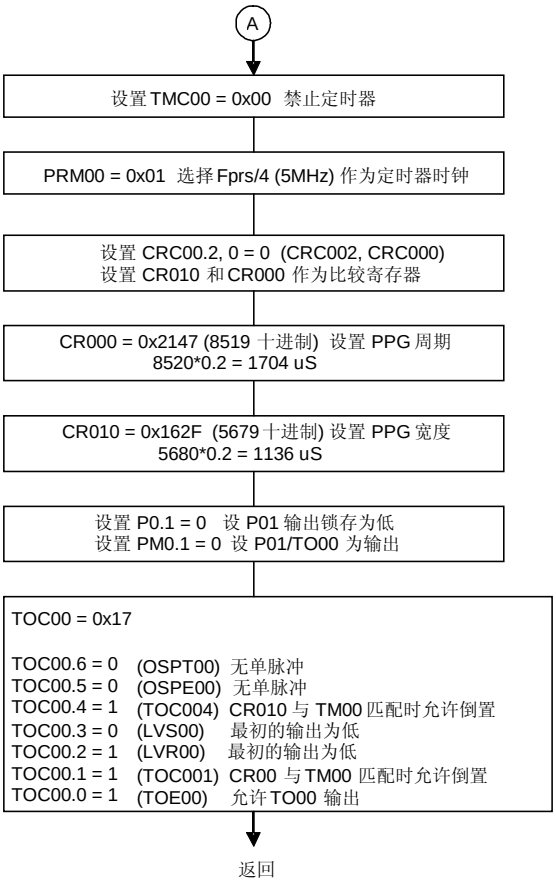


C 语言写的 78K0 程序启动代码由结果代码文件提供，例如 s0l.rel，连接到用户程序。调用启动代码的函数名称是 hdwinit( )，用户可在这里放置任意的硬件初始化代码。

当 Applilet 产生 78 K0 的 C 语言，hdwinit( ) 函数自动运行增加到用户程序，并调用 SystemInit( ) 函数。这个 SystemInit( ) 函数用于每个外设调用初始化程序。

hdwinit( ) 函数结束后，启动代码调用用户程序的 main( ) 函数。启动 main( ) 函数，所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

6.3.2 TM00_Init( ) –初始化定时器 00 用于 PPG 输出



TM00_Init() 程序，由 SystemInit()调用，初始化定时器 00 用于 PPG 输出。控制寄存器写入时禁止定时器。

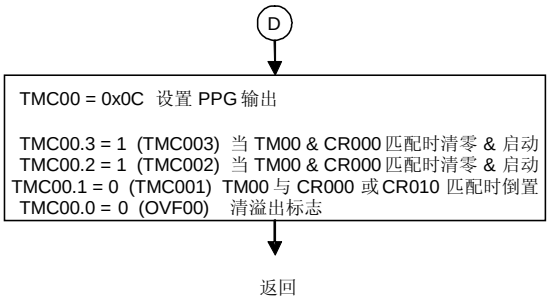
预分频模式寄存器设为 0x01 选择 fPRS/4 作为定时器 00 的时钟。主时钟为 20 MHz，定时器 00 的时钟为 5 MHz；每 0.2us 计数一次。

CRC00 设 CR010 和 CR000 作为比较寄存器。CR000 设为 0x2147 (8519)， PPG 周期为 8520 * 0x2 = 1704 us 适用于 586 Hz。CR010 设为 0x162F (5679) ，PPG 宽度为 5680 * 0.2 = 1136 us。PPG 波形为低 1136 us，高 (1704 – 1136) 568 us。结果是音调为 586 Hz，泛音频率与 568 us 和 1136 us 2，或 880 Hz 和 440 Hz 周期有关。

P01/TO00 引脚用于定时器 00 的 TO00 输出 ，所以 P0.1 输出锁存设为 0，端口模式寄存器 PM0 的位 1 设为 0，将此引脚作为输出模式。

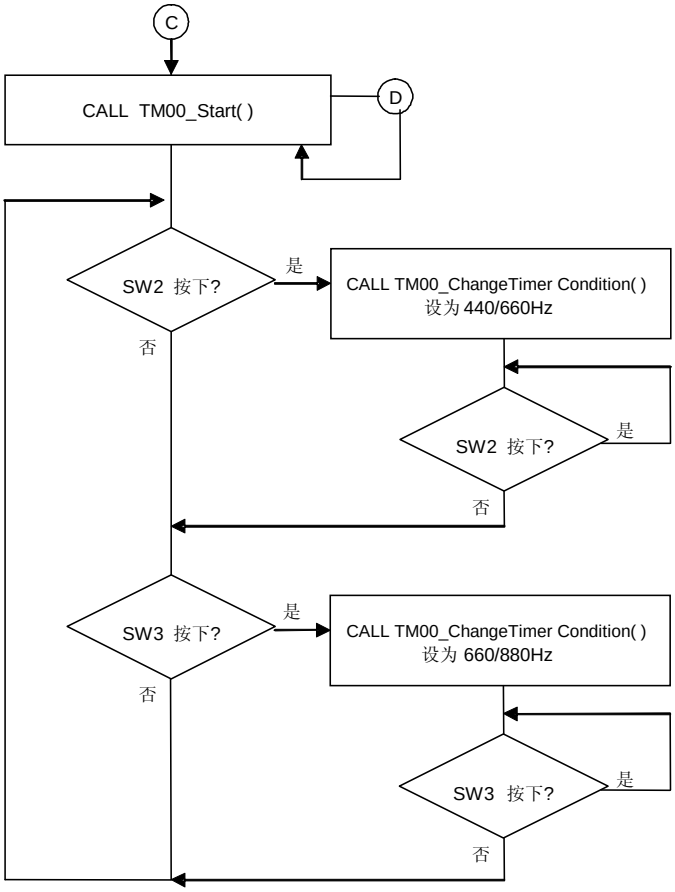
TOC00 寄存器设初始定时器输出为低，CR000 和 CR010 与 TM00 匹配时倒置，并允许 TO00 输出。

6.3.3 TM00_Start( ) – 启动定时器 00 操作



TM00_Start() 程序设置 TMC00 用于 PPG 输出，清零启动定时器。

6.3.4 Main( ) – 主程序 – 使用 PPG 产生 2 音调



主程序调用 TM00_Start() 允许定时器操作，以 CR000 和 CR010 缺省值启动 PPG 音调产生。

检查按键 SW2 或 SW3 是否按下。当 SW2 按下，调用 TM00_ChangeTimerCondition() ，改变 CR000 和 CR010 寄存器值来改变 PPG 时序，产生 528 Hz 输出，440 Hz 和 660 Hz 泛音。处理前等待 SW2 抬起。

当 SW3 按下，调用 TM00_ChangeTimerCondition() ，设置 CR000 和 CR010 用于产生 754 Hz 音调，以 660 Hz 和 880 Hz 泛音。并等待 SW3 抬起。

这样程序转换 2 音调的设置。程序代码可改变 CR000 和 CR010 值以用于组合成提前的音调。

6.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言 或汇编语言代码来管理 NEC 微控制器外设。请参见使用 Applilet 版本的附录。

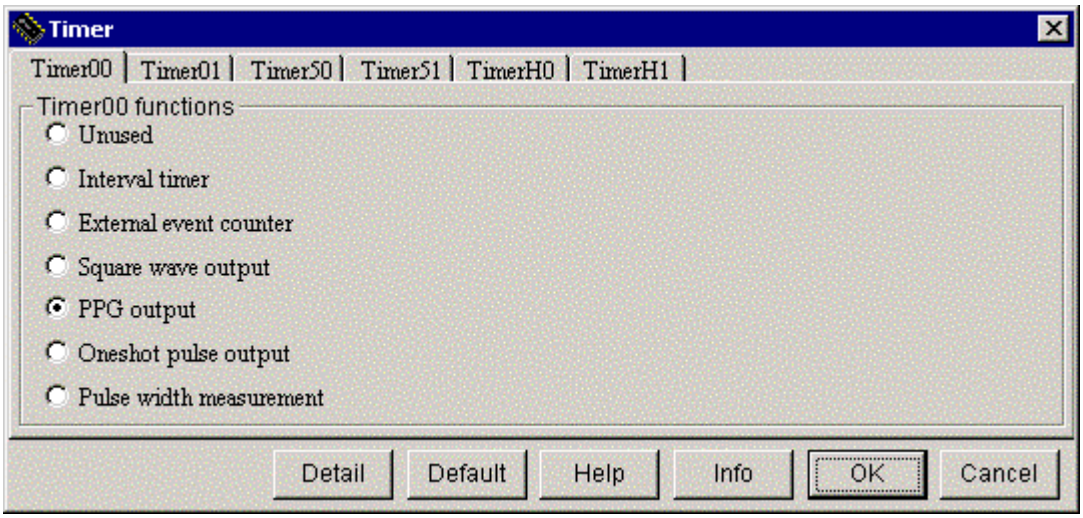
Applilet 用于创建基础的初始化代码和定时器 50 和定时器 00 的驱动代码，并且可以管理 LED 显示输出的 I/O 端口。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 50 和定时器 00 的代码和产品程序列表描述如下。

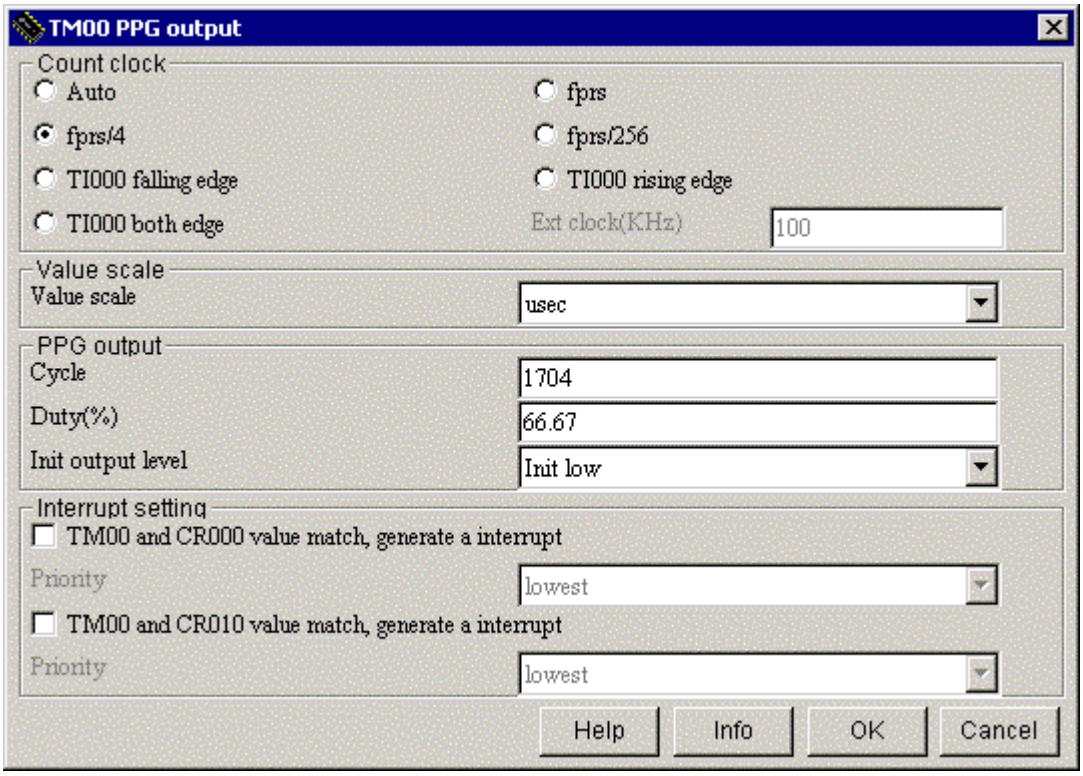
启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同的外设模块的界面。

6.4.1 定时器 00 的 Applilet 配置

选择 “Timer”，选择定时器 00 列表显示可用的定时器 00 设置。将定时器设置为 PPG 输出。



一旦选择使用定时器 00 ，“Detail” 按钮可用于设置定时器 00 选择模式。点击“Detail” 按钮出现对话框设置 PPG 。



这里是选择定时器 00 的操作细节。 我们希望定时器最初产生 1704us 的 PPG 波形，分成 1/3 和 2/3 占空比。

计数时钟选择 fprs/4，5 MHz ；这将允许计数器的最高分辨率。刻度值选择为 us，并 PPG 周期设置为 1704us。占空比设为 66.67%，产生想得到的 1/3 和 2/3 占空比。输出设为最初为低输出。

因为我们不希望当 TM00 与比较寄存器匹配时产生中断，不勾选产生中断对话框。

6.4.2 Applilet 产生的代码

一旦定时器 对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其功能，允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码、中断服务程序和定时器 00 驱动子程序。

当“Generate”按钮按下，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持定时器 00，Applilet 产生 timer.h, timer.c 和 timer_user.c，并且其他几个和定时器无关。

6.4.3 Applilet 为定时器 00 产生的文件和函数

支持定时器 00 所产生的代码在文件 timer.h, timer.c 和 timer_user.c 中。它们包括以下内容。

6.4.3.1 Timer.h

头文件 timer.h 包含了控制定时器 00 的函数声明，和定时器 00 初始化值的定义。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。

6.4.3.2 Timer.c

源文件 Timer.c 包含了 Applilet 为定时器 00 产生的下列函数：

void TM00_Init(void);
TM00_Init()程序初始化定时器 00 外设，在 Applilet 定时器 00 对话框中设定。

void TM00_Start(void);
TM00_Start()程序启动定时器 00 操作 并允许定时器工作。

void TM00_Stop(void);
TM00_Stop()程序停止定时器 00 操作。此程序不用在演示程序。

MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
TM00_ChangeTimerCondition()函数改变定时器 00 比较寄存器 CR000 和 CR010 的值，并因此改变定时器 00 的 PPG 波形值。

array_reg 参数指向列值，放置一个或其他的比较值；array_num 参数为 1 (选择 CR000) 或 2 (选择 CR010 和 CR000)。

6.4.3.3 Timer_user.c

源程序 timer_user.c 给用户提供了一些框架函数。这些函数是产生的空函数，允许用户增加自己的应用代码。因为我们不选择定时器 00 产生中断，这儿没有此文件的代码。

6.4.4 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
System.h	时钟关联定义
System.c	Clock_Init()函数
Port.h	定义默认 I/O 端口状态
Port.c	PORT_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

6.4.5 不由 Applilet 产生的演示程序

演示程序包括以下不由 Applilet 产生的文件。

文件	功能
Sw_0537.h	按键按下输入头文件
Sw_0537.c	按键读取和去抖动代码

6.5 演示平台

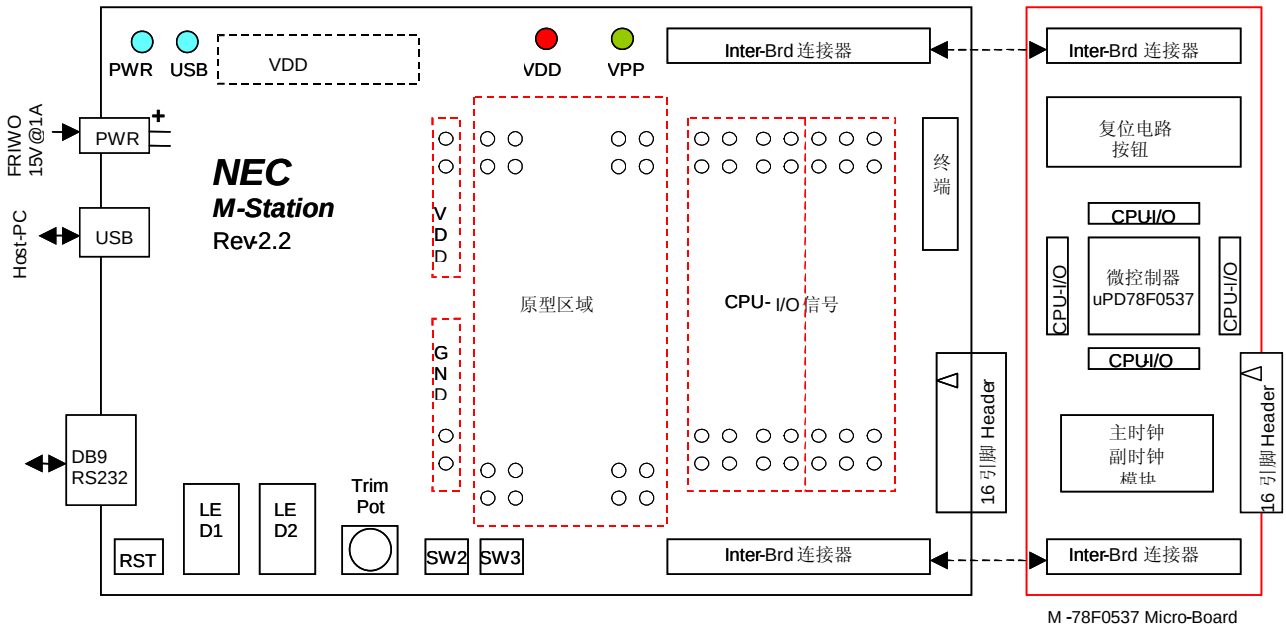
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

6.5.1 资源

为了演示该程序，需要使用以下资源：

- 焊有 uPD78F0537 8 位微控制器的 M-78F0537 Micro-Board
- M-Station II 仿真系统，M-Station II 上的资源：
  - 按键 SW2 和 SW3
  - 硬件增加连接扬声器到定时器输出

以上硬件列表的详细说明，请参见用户手册，该手册可以向 NEC 索取。

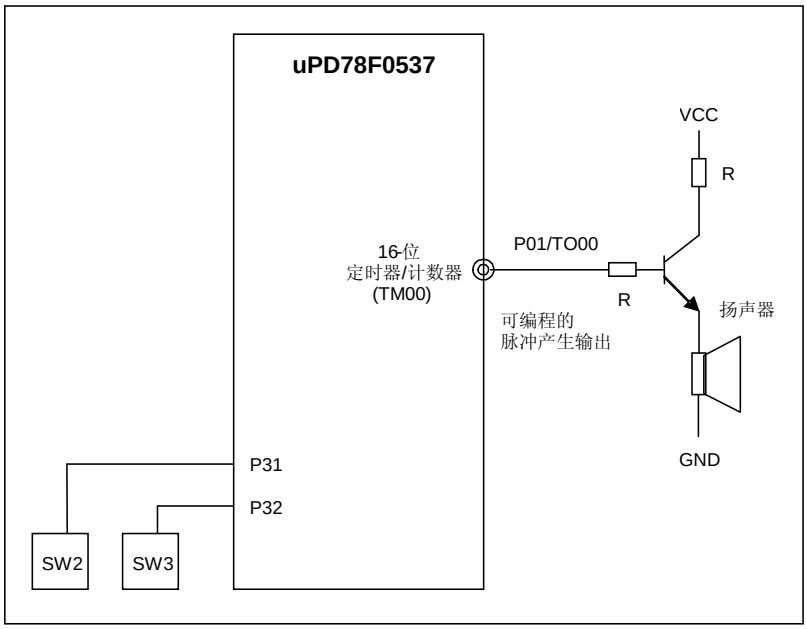


6.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 按下 SW2　　440Hz/660Hz　2 音调声音
- 按下 SW3　　660Hz/880Hz　2 音调声音

6.6 硬件框图



6.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

这些文件列表在附录中。

文件	目的	由 Applilet 产生	可由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟关联定义	是	否
Systeminit.c	SystemInit() 和 hdwinit()函数	是	否
System.c	Clock_Init()函数	是	否
Timer.h	定时器关联定义	是	否
Timer.c	定时器函数	是	否
TIMER_user.c	定时器中断用户代码	是	否
Port.h	端口关联定义	是	否
Port.c	Port_Init() 函数	是	否
Sw_0537.h	按键输入定义	--	是
Sw_0537.c	按键输入函数	--	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

7 方波音调产生使用16位定时器00 (TM00)

78K0 系列微控制器的大多数 8 位和 16 位定时器/计数器提供方波输出。

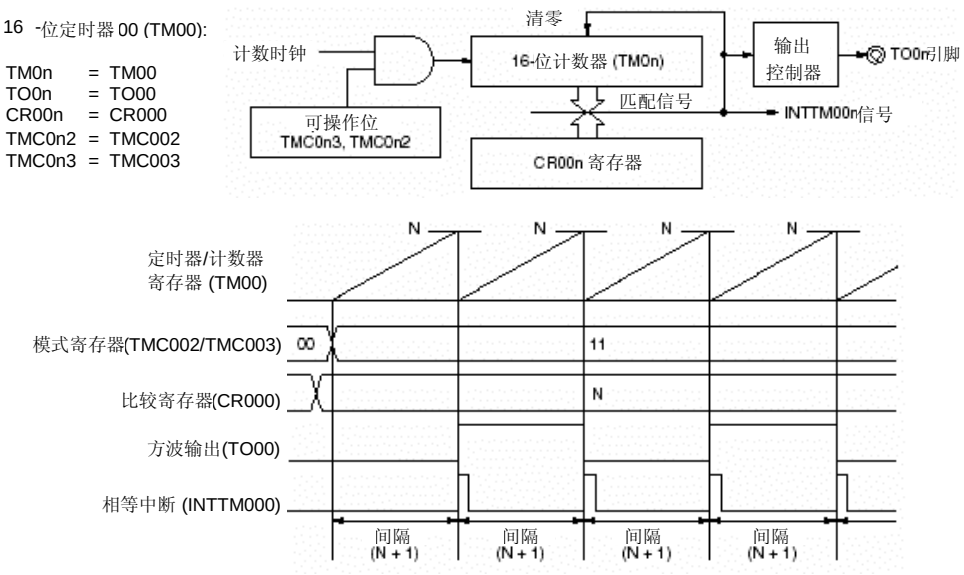
程序使用方波输出演示产生音调，由 16 位定时器/计数器输出驱动扬声器。

7.1 方波发生特征

在方波发生模式，16 位定时器/计数器 00 提供以下特征：

- 使用 PPM00 寄存器，基于外部时钟分频可变时间
- 使用 CR000 寄存器，可变方波 1 到 65536 时钟半周期时间
- 使用 CR000 计数高低电平周期，精准 50%占空比
- 可选每半周期结束产生中断

当定时器模式控制寄存器设为清零启动模式时，从定时器引脚(TO00)输出方波，输出波形的脉冲宽度由比较寄存器(CR000)设置。



方波频率 = 1/[2x(N+1)x(计数时钟周期)]

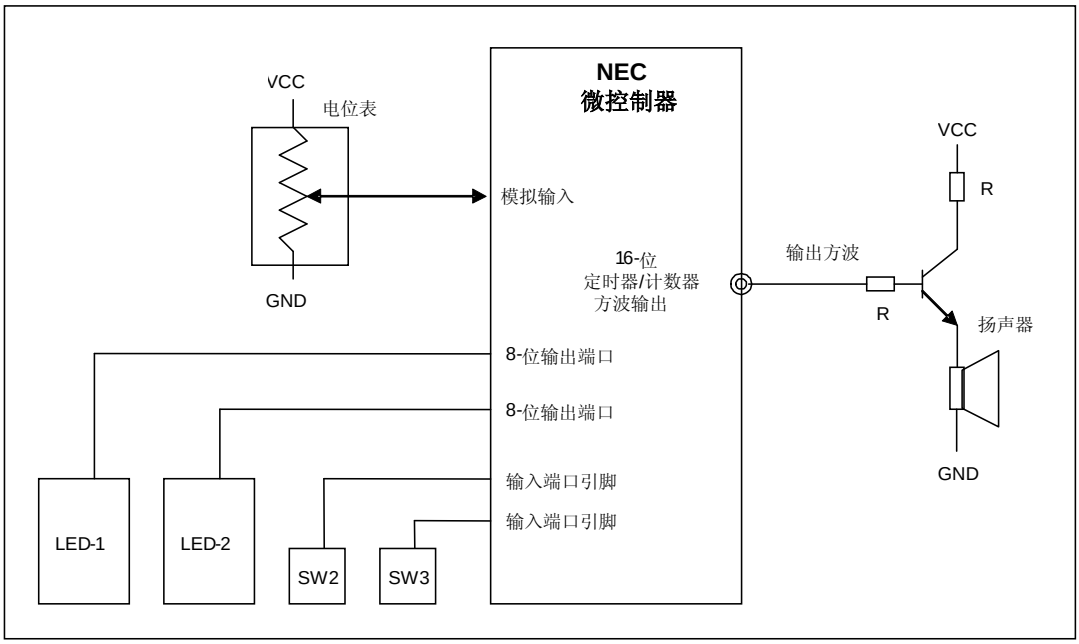
配置定时器用于方波输出操作，按以下步骤：

- 使用 PRM00 设置基准时间
- 使用 CRC00 设置 CR000 作为比较寄存器
- 在 CR000 设置半周期时间
- 设置输出端口锁存和模式寄存器位为低用于方波输出
- 如果使用中断，设置中断优先级、清中断标志和非屏蔽中断
- 设置 TOC00 寄存器用于初始输出和 CR000 匹配时倒置
- 允许定时器由定时器模式寄存器 TMC00 设置为清零启动模式，比较 TM00 和 CR000

7.2 程序描述和说明

演示程序将使用 2 路转换和电位表。最初，方波设置为大约 440Hz 波形，在连接到定时器/计数器输出的扬声器产生音调信号。

当按键 (SW2) 按下保持并打开电位表，声音音调将改变为较宽的范围(A/D 转换值从电位表被装载在比较寄存器顶部字节位置)。当另一按键 (SW3) 按下保持并打开电位表，声音音调将改变为较窄的范围(A/D 转换值从电位表被装载在比较寄存器底部字节位置)。



说明：

- TM00 设为方波模式，无中断
- TM00 时间基准时钟设为 5 MHz
- 模拟输入 ANI0/P20 用作从 0V 到 5.0V 的输入电压
- 音调产生可用 SW2 从 00xx 到 FFxx 调整
- 音调产生可用 SW3 从 00xx 到 FFxx 计数调整

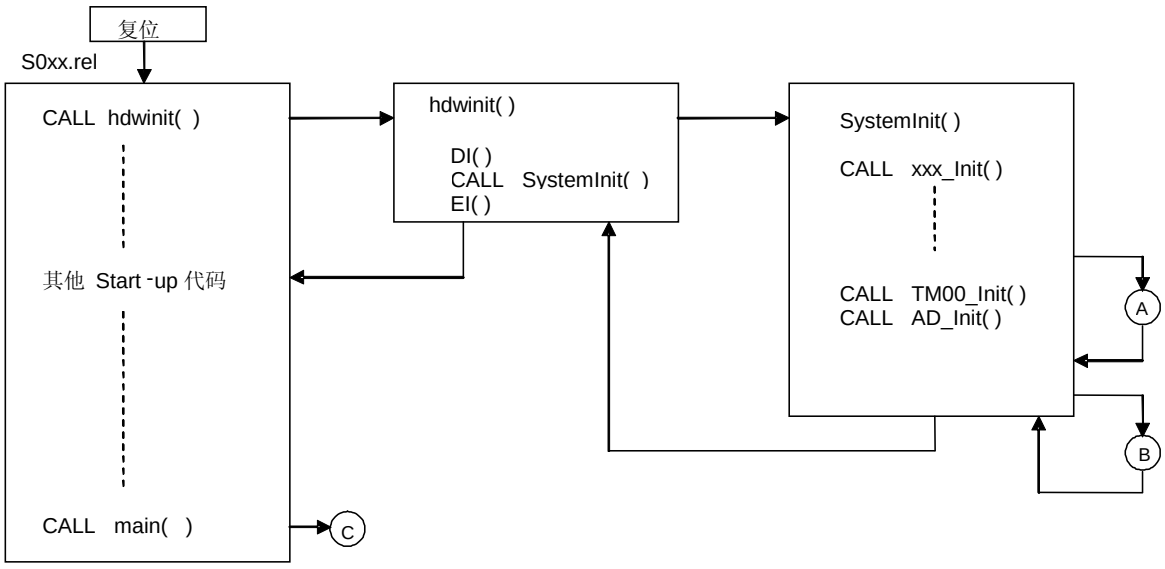
7.3 软件流程图

演示程序由以下主要几部分组成：

- 程序的初始化代码，在 main() 程序启动前调用
- 主程序循环，检查按键状态、读取 A/D 转换值和改变 TM00 的时间
- 由 Applilet 产生的子程序启动 TM00 和停止、改变间隔和把握 A/D 输入
- 按键输入、LED 显示输出子程序

这里的流程图描述初始化，主程序，TM00 相关子程序。

7.3.1 启动程序和初始化



C 语言写的 78K0 程序启动代码由结果代码文件提供，例如 s0l.rel，连接到用户程序。调用启动代码的函数名称是 hdwinit( )，用户可在这里放置任意的硬件初始化代码。

当 Applilet 产生 78 K0 的 C 语言，hdwinit( ) 函数自动运行增加到用户程序，并调用 SystemInit( ) 函数。这个 SystemInit( ) 函数用于每个外设调用初始化程序。

hdwinit( ) 函数结束后，启动代码调用用户程序的 main( ) 函数。启动 main( ) 函数，所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

7.3.2 TM00_Init( ) -初始化定时器 00 用于 方波输出

TM00_Init()程序，由 SystemInit()调用，初始化定时器 00 用于方波输出。控制寄存器写入时禁止定时器操作。

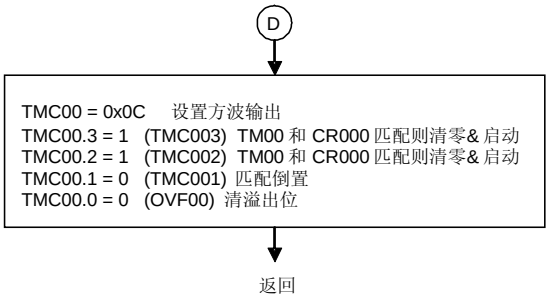
预分频模式寄存器设为 0x01 选择 fPRS/4 作为定时器 00 的时钟。主时钟为 20 MHz，定时器 00 的时钟为 5 MHz；每 0.2us 计数一次。

CRC00 设 CR010 和 CR000 作为比较寄存器。CR000 设为 0x162F (5679)，方波周期为 5680 * 0.2 = 1136 us 为 586 Hz。方波为高 1136 us，低 1136 us，总频率为 1/(1136 + 1136) = 大约 440 Hz。

P01/TO00 引脚用于定时器 00 的 TO00 输出 ，所以 P0.1 输出锁存设为 0，端口模式寄存器 PM0 的位 1 设为 0，将此引脚作为输出模式。

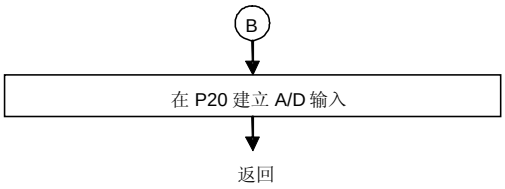
TOC00 寄存器设初始定时器输出为低，CR000 和 CR010 与 TM00 匹配时倒置，并允许 TO00 输出。

7.3.3 TM00_Start( ) – 启动定时器 00 操作



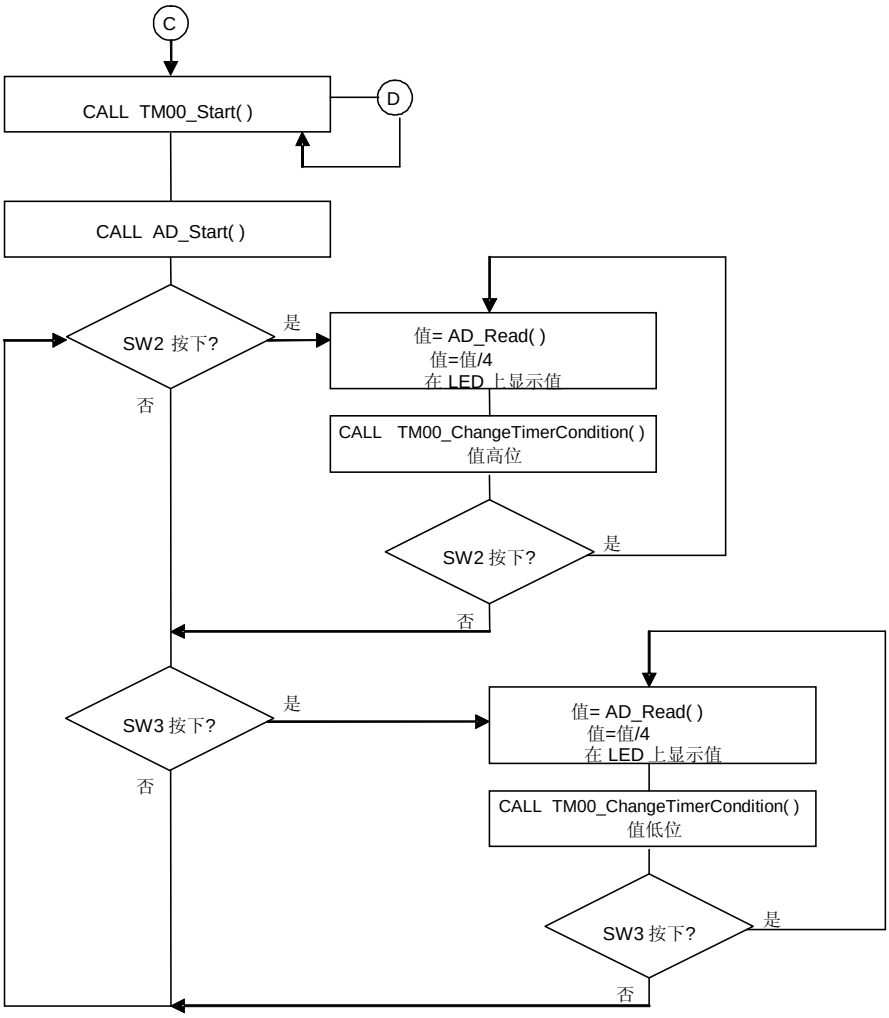
程序设置 TMC00 用于方波 输出，清零启动定时器。

7.3.4 AD_Init( ) – 初始化 A/D 转换器



AD_Init() 函数初始化 A/D 转换器来读取 ANI0/P20 引脚的模拟输入 ANI0 。此模拟输入由电位表设置可变电压。

7.3.5 Main( ) –主程序 – 使用 方波产生音调



主程序调用 TM00_Start()启动定时器 00 输出操作，调用 AD_Start()建立 A/D 输入。这里，定时器 00 将产生大约 440 Hz 的方波。

主程序检测 SW2 或 SW3 是否按下。如果没有按键按下，没有动作并且音调输出稳定。
如果 SW2 按下，程序从电位表读取模拟输入，并设置 CR000 高字节，在 LED 上显示值。其结果依据电位表的值产生一新的方波频率。只要 SW2 按下，重复此动作；由保持 SW2 按下并打开电位表，产生变化范围大的频率。

如果 SW3 按下，程序从电位表读取模拟输入，并设置 CR000 低字节，在 LED 上显示值。其结果紧挨着先前的产生一个新的方波频率。只要 SW3 按下，重复此动作；由保持 SW3 按下并打开电位表，对频率作调整。

7.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言 或汇编语言代码来管理 NEC 微控制器外设。请参见使用 Applilet 版本的附录。

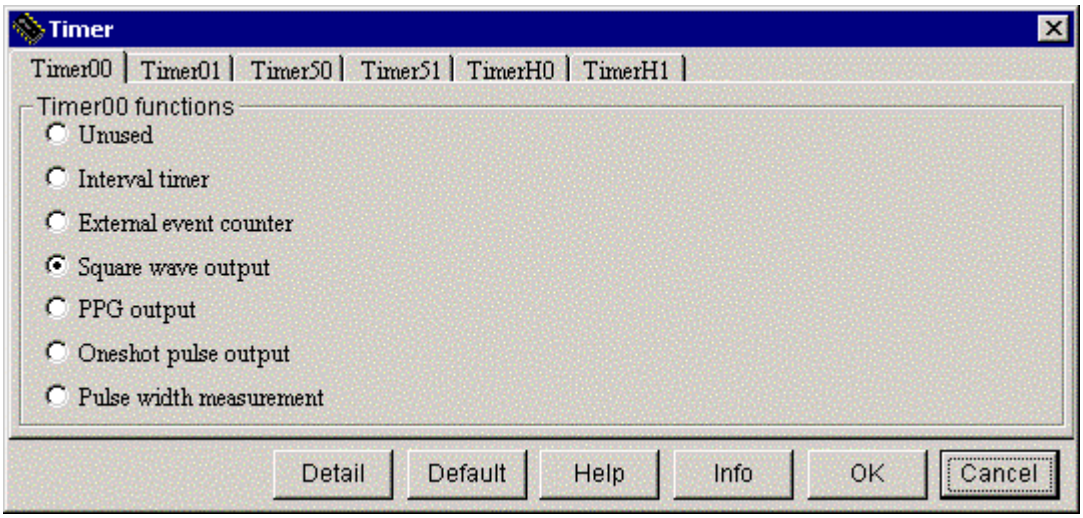
Applilet 用于创建基础的初始化代码和定时器 50 和定时器 00 的驱动代码，并且可以管理 LED 显示输出的 I/O 端口。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 50 和定时器 00 的代码和产品程序列表描述如下。

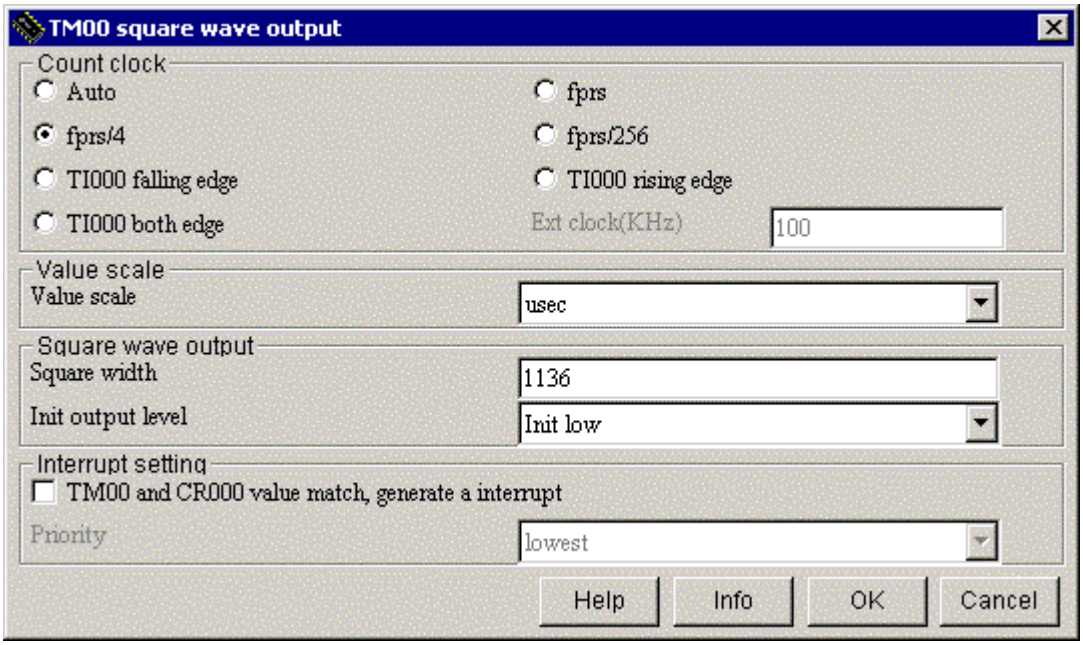
启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同的外设模块的界面。

7.4.1 定时器 00 的 Applilet 配置

选择 “Timer”，选择定时器 00 列表显示可用的定时器 00 设置。将定时器设置为方波输出。



一旦选择使用定时器 00 ，“Detail” 按钮可用于设置定时器 00 选择模式。 点击“Detail” 按钮出现对话框设置方波发生器。



这里是选择定时器 00 的操作细节。 我们希望定时器产生大概 440 Hz 的方波。需要高低 1136us 时间。

定时器 000 的计数时钟设为 fprs/4 ，5 MHz，每一时钟为 0.2us。刻度值选择为 us，并将方波宽度设为 1136 us。注意这里的宽度是方波的一半。初始输出设为低。

因为我们不希望当 TM00 与 CR000 匹配时产生中断，不勾选产生中断对话框。

7.4.2 Applilet 产生的代码

一旦定时器 对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其功能，允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码，中断服务程序，和定时器 00 驱动子程序。

当“Generate”按钮按下，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持定时器 00，Applilet 产生 timer.h, timer.c 和 timer_user.c，并且其他几个和定时器无关。

7.4.3 Applilet 为定时器 00 产生的文件和函数

支持定时器 00 所产生的代码在文件 timer.h, timer.c 和 timer_user.c 中。它们包括以下内容。

7.4.3.1 Timer.h

头文件 timer.h 包含了控制定时器 00 的函数声明，和定时器 00 初始化值的定义。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。

7.4.3.2 Timer.c

源文件 Timer.c 包含了 Applilet 为定时器 00 产生的下列函数：

void TM00_Init(void);
TM00_Init()程序初始化定时器 00 外设，在 Applilet 定时器 00 对话框中设定。

void TM00_Start(void);
TM00_Start()程序启动定时器 00 操作 并允许定时器工作。

void TM00_Stop(void);
TM00_Stop()程序停止定时器 00 操作。此程序不用在演示程序。

MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
TM00_ChangeTimerCondition()函数改变定时器 00 比较寄存器 CR000 和 CR010 的值，并因此改变定时器 00 的计数器比较值。

array_reg 参数指向列值，放置一个或其他的比较值；array_num 参数为 1 (选择 CR000) 或 2 (选择 CR010 和 CR000)。

7.4.3.3 Timer_user.c

源程序 timer_user.c 给用户提供了一些框架函数。这些函数是产生的空函数，允许用户增加自己的应用代码。因为我们不选择定时器 00 产生中断，这儿没有此文件的代码。

7.4.4 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
System.h	时钟关联定义
System.c	Clock_Init()函数
Ad.h	A/D 转换器定义
Ad.c, Ad_user.c	A/D 函数
Port.h	定义默认 I/O 端口状态
Port.c	PORT_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

7.4.5 不由 Applilet 产生的演示程序

演示程序包括以下不由 Applilet 产生的文件。

文件	功能
Sw_0537.h	按键按下输入头文件
Sw_0537.c	按键读取和去抖动代码
Led_0537.h	7 段 LED 头文件
Led_0537.c	7 段 LED 上显示数据头文件

7.5 演示平台

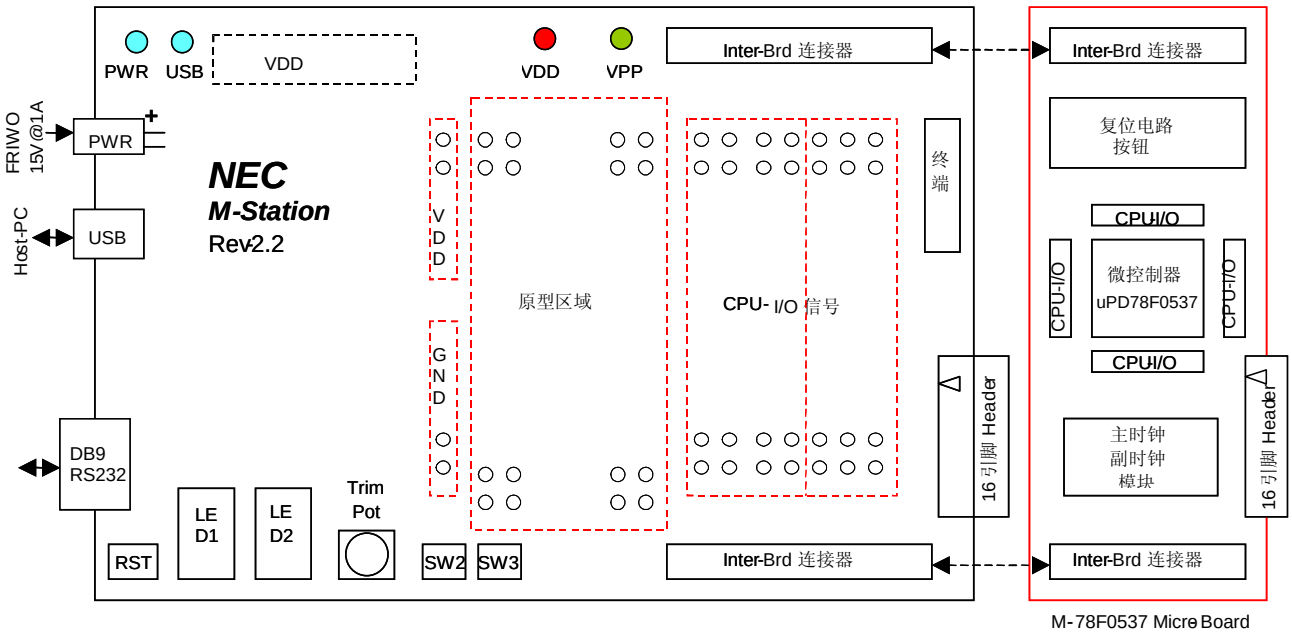
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

7.5.1 资源

为了演示该程序，需要使用以下资源：

- 焊有 uPD78F0537 8 位微控制器的 M-78F0537 Micro-Board
- M-Station II 仿真系统，M-Station II 上的资源：
  - 按键 SW2 和 SW3
  - 7 段 LED LED1 和 LED2
  - 电位表提供可变模拟输入电压
  - 硬件增加扬声器连接到定时器输出

以上硬件列表的详细说明，请参见用户手册，该手册可以向 NEC 索取。

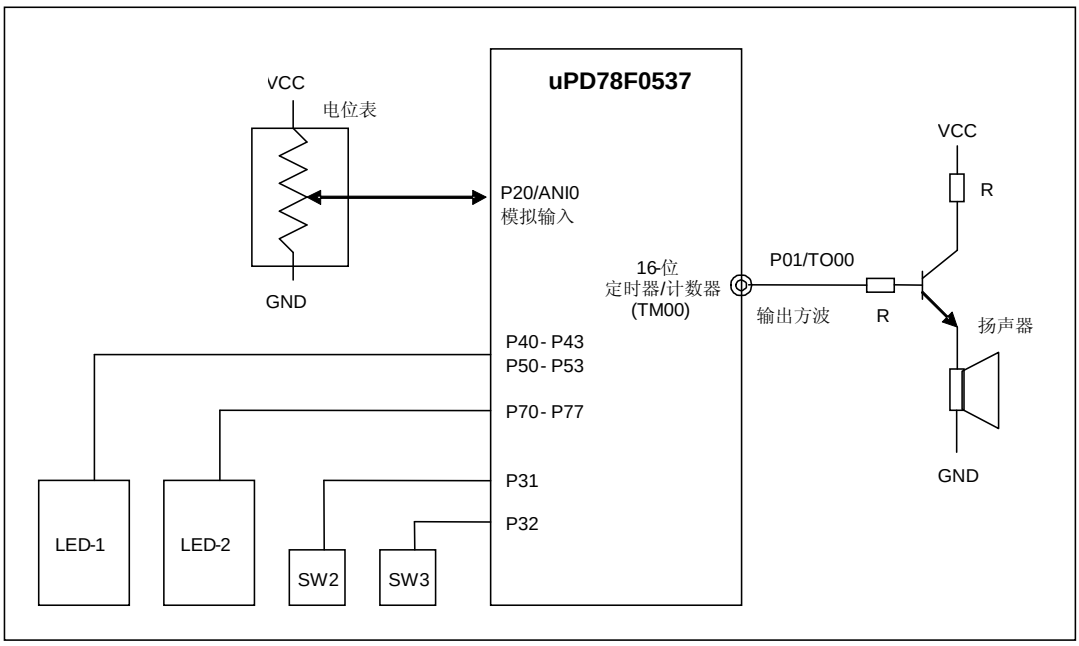


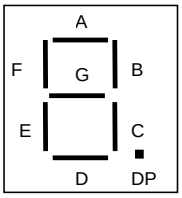
7.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 按下 SW2　按下并保持　改变电位表　改变宽范围音调
- 按下 SW3　按下并保持　改变电位表　改变窄范围音调

7.6 硬件框图





LED-1 和 LED -2

段	LED-1	LED-2
A	P40	P70
B	P41	P71
C	P42	P72
D	P43	P73
E	P50	P74
F	P51	P75
G	P52	P76
DP	P53	P77

7.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

这些文件列表在附录中。

文件	目的	由 Applilet 产生	可由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟关联定义	是	否
Systeminit.c	SystemInit() 和 hdwinit()函数	是	否
System.c	Clock_Init()函数	是	否
Timer.h	定时器关联定义	是	否
Timer.c	定时器函数	是	否
TIMER_user.c	定时器中断用户代码	是	否
Ad.h	A/D 转换器关联定义	是	否
Ad.c	A/D 转换器函数	是	否
Ad_user.c	A/D 转换器操作用户代码	是	否
Port.h	端口关联定义	是	否
Port.c	Port_Init() 函数	是	否
Led_0537.h	LED 显示定义	--	是
Led_0537.c	LED 显示函数	--	是
Sw_0537.h	按键输入定义	--	是
Sw_0537.c	按键输入函数	--	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

8 钟表定时器 (WTM)的时间保持

大多数 78K0 系统微控制器提供钟表定时器。 钟表定时器通常用于时间保持，基于精准时钟提供周期性中断。

程序将演示由使用 32.768 KHz 晶振的钟表定时器时间保持。使用钟表定时器用于钟表定时器和间隔定时器，由钟表定时器间隔模式来去抖动输入按键。

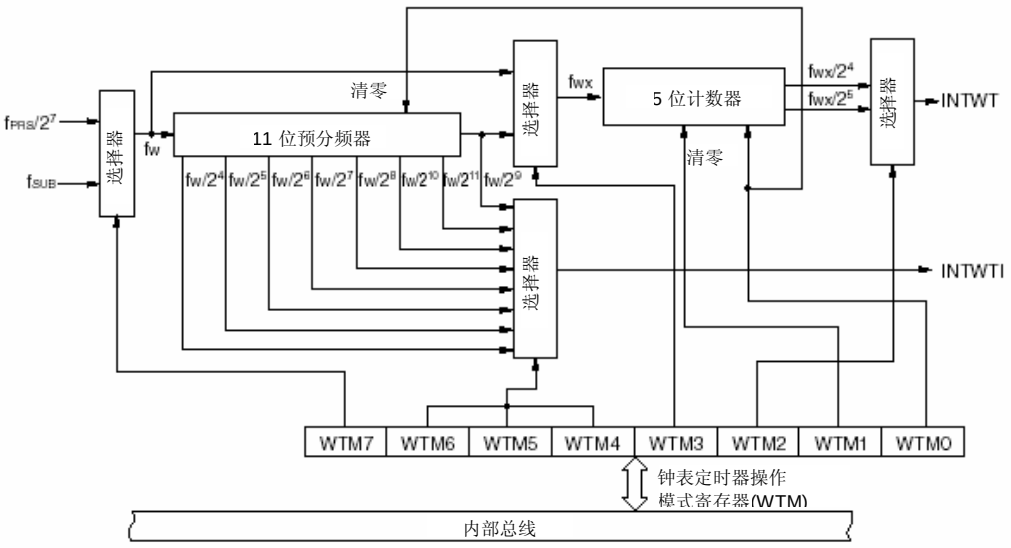
8.1 钟表定时器特征

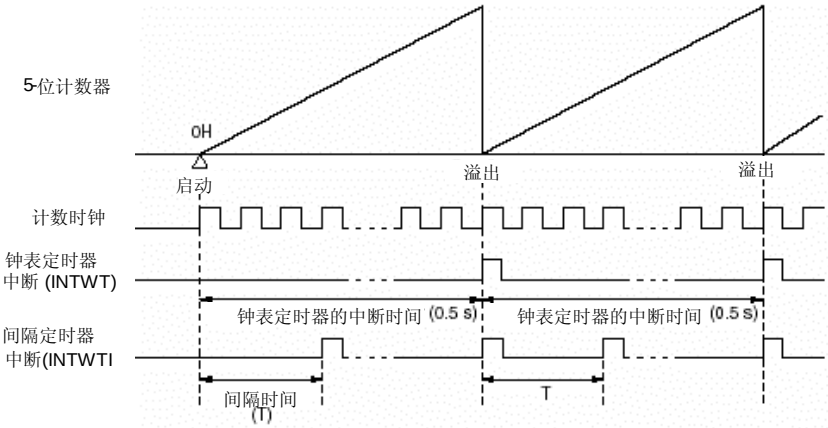
钟表定时器提供以下特征：

- 同时用于钟表定时器和间隔定时器
- 基于外设时钟分频或外部副时钟可变时间基准时钟
- 在四分之一预分频间隔可变钟表定时器中断输入
- 在八分之一预分频间隔可变间隔定时器中断

钟表定时器可用作钟表定时器或间隔定时器。钟表定时器和间隔定时器可同时使用。

钟表定时器框图





78K0/Kx2 系列微控制器的钟表定时器有以下硬件组成。

钟表定时器硬件配置

项目	配置
计数器	5位 ×1
预分频器	11位 ×1
控制寄存器	钟表定时器操作模式寄存器(WTM)

使用外设时钟或副时钟，在指定时间间隔，钟表定时器产生中断请求(INTWT) 。当钟表定时器操作在间隔定时器，它在预置的计数值产生中断请求(INTWTI)。

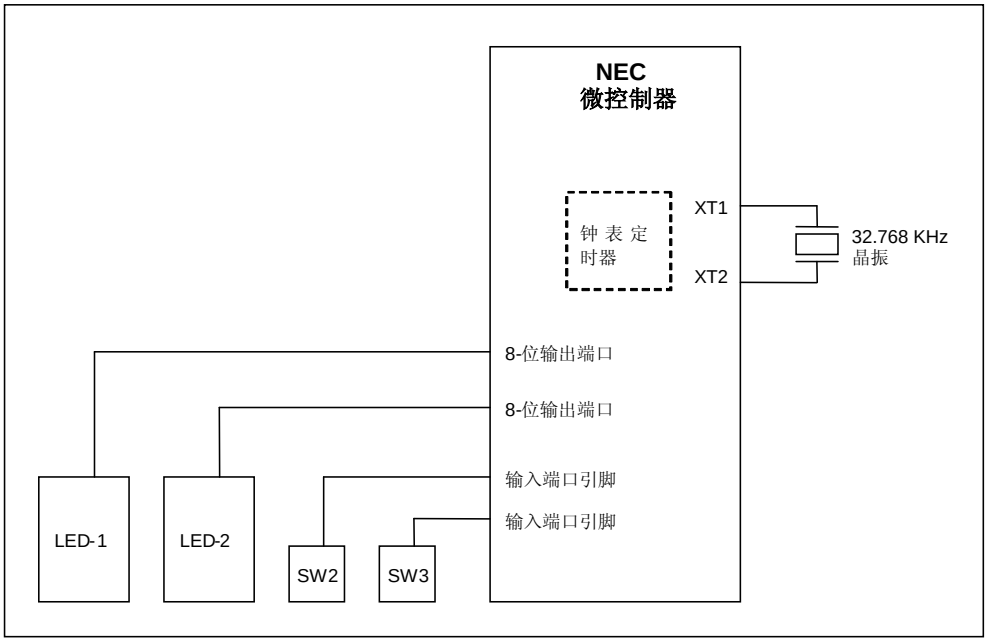
配置钟表定时器用于钟表定时器和/或间隔定时器操作，按以下步骤：

- 由 WTM.7 设置输入时钟为 f_{PRS}/128 或副时钟
- 使用 WTM3 - WTM2 设置钟表定时器分频
- 使用 WTM6 – WTM4 设置间隔定时器分频
- 如果使用中断，设置钟表定时器中断优先级，清中断标志并且非屏蔽中断
- 如果使用中断，设置间隔定时器中断优先级，清中断标志并且非屏蔽中断
- 设 WTM1 和 WTM0 为 1，允许定时器工作

8.2 程序描述和说明

演示程序使用钟表定时器在 2-数字 LED 上显示秒。秒时间由钟表定时器功能保持。钟表定时器间隔功能用于按键输入 SW2 和 SW3 去抖动。

当 SW2 按下，秒时间启动或停止。当 SW3 按下，秒时间复位为 00。



说明：

- 钟表定时器设为 32.768 KHz 副时钟输入
- 每 0.5s 发生周期钟表定时器中断
- 周期间隔定时器设为 1ms
- 当稳定 10ms 按键去抖动

8.3 软件流程图

演示程序由以下主要几部分组成：

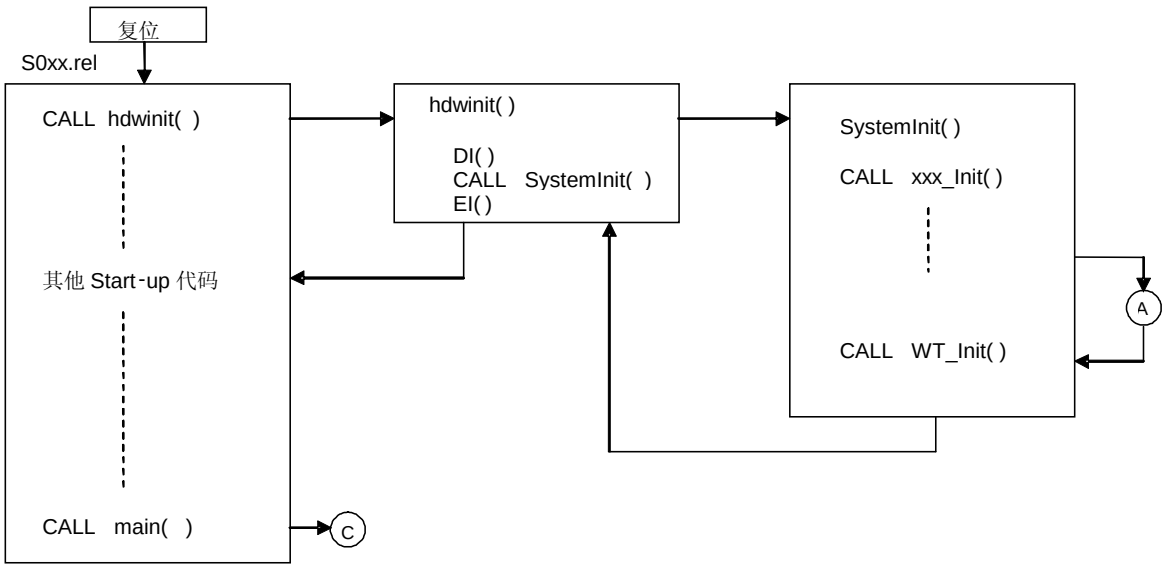
- 程序的初始化代码，在 main() 程序启动前调用
- 主程序循环，检查按键状态和启动、停止或清秒定时器
- 由 Applilet 产生的子程序启动 WTM，停止和改变间隔
- WTM 钟表定时器和间隔定时器中断用户代码子程序（Applilet 产生框架中断服务子程序，增加用户代码）
- 按键输入和 LED 显示输出子程序

这里的流程图描述初始化、主程序和钟表定时器相关子程序。

8.3.1 启动程序和初始化

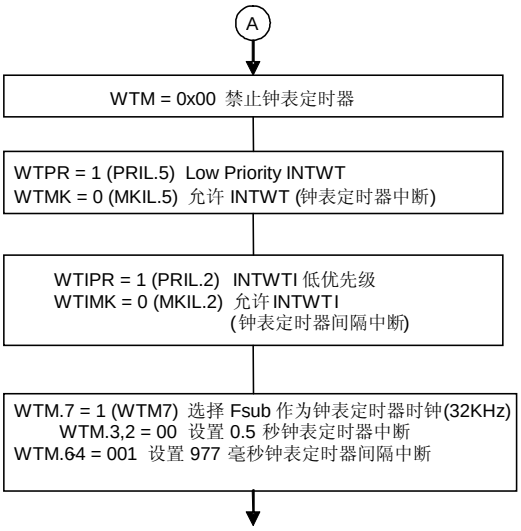
C 语言写的 78K0 程序启动代码由结果代码文件提供，例如 s0l.rel，连接到用户程序。调用启动代码的函数名称是 hdwinit( )，用户可在这里放置任意的硬件初始化代码。

当 Applilet 产生 78 K0 的 C 语言，hdwinit( ) 函数自动运行增加到用户程序，并调用 SystemInit( ) 函数。这个 SystemInit( ) 函数用于每个外设调用初始化程序。



hdwinit( ) 函数结束后，启动代码调用用户程序的 main( ) 函数。启动 main( ) 函数，所有外设初始化。main( ) 函数不需要调用个别外设初始化程序。

8.3.2 WT_Init( ) – 钟表定时器初始化



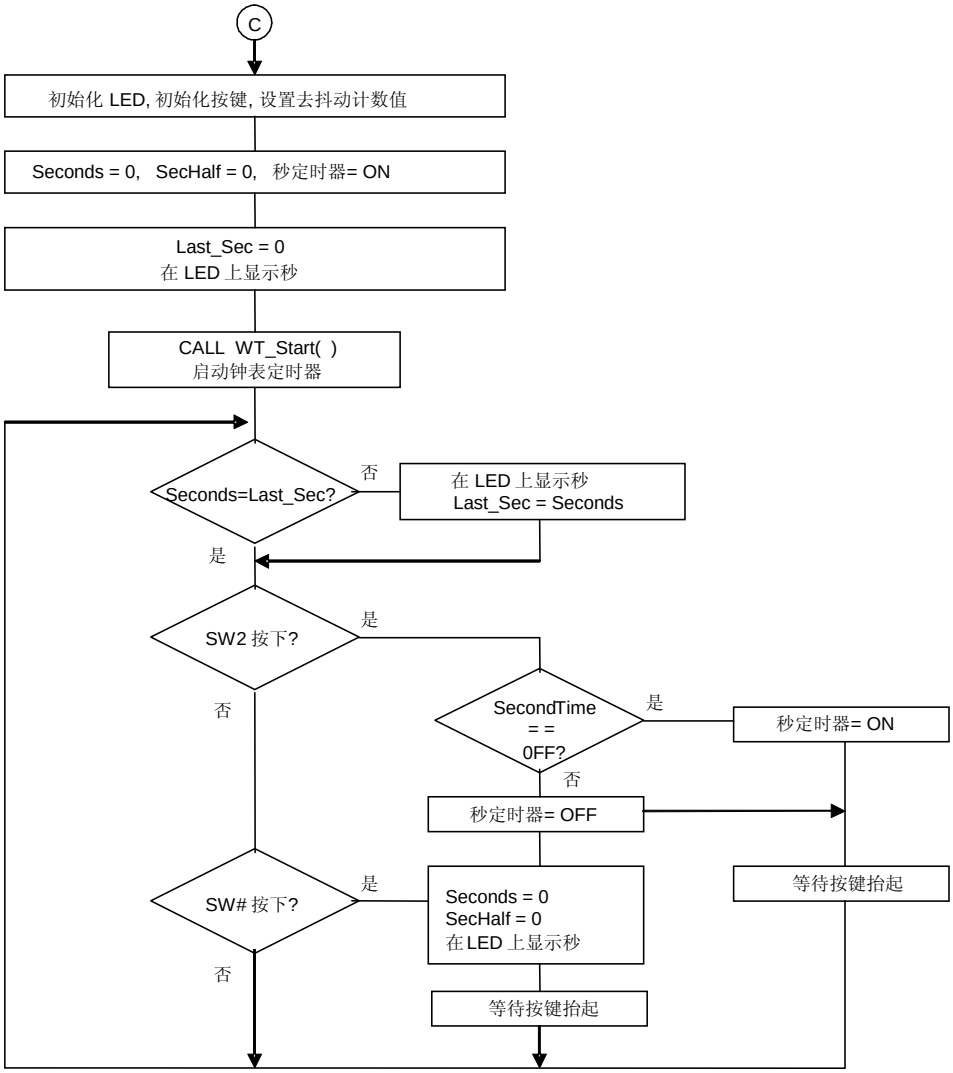
TM00_Init() 程序，由 SystemInit()调用，初始化钟表定时器和间隔定时器功能。当其他 WTM 寄存器设置改变时，最初设置 WTM.0 为 0 禁止钟表定时器。

WTPR (PR1L.5) 和 WTMK (MK1L.5) 位设 WTM 钟表定时器中断 INTWT 低优先级和允许中断。

WTIPR (PR1L.2) 和 WTIMK (MK1L.2) 位设 WTM 钟表定时器中断 INTWTI 低优先级和允许中断。

WTM.7 设为 1，设 32.768 KHz 副时钟为 fw，WTM 的基准时间。WTM 位 3 和位 2 都设为 0，设置钟表定时器中断时间为 16384/fw 或每 0.5 s。WTM 位 6 到 4 设为 001，间隔定时器中断时间为 32/fw 或每 977u s，大约 1ms 间隔时间。

8.3.3 Main( ) – 主程序 – 钟表定时器时间保持



main() 函数初始化 LED 和设置按键去抖动计数器。变量用于记录秒清零，并且 SecondTimer 变量设为打开。程序调用 WT_Start() 启动钟表定时器。

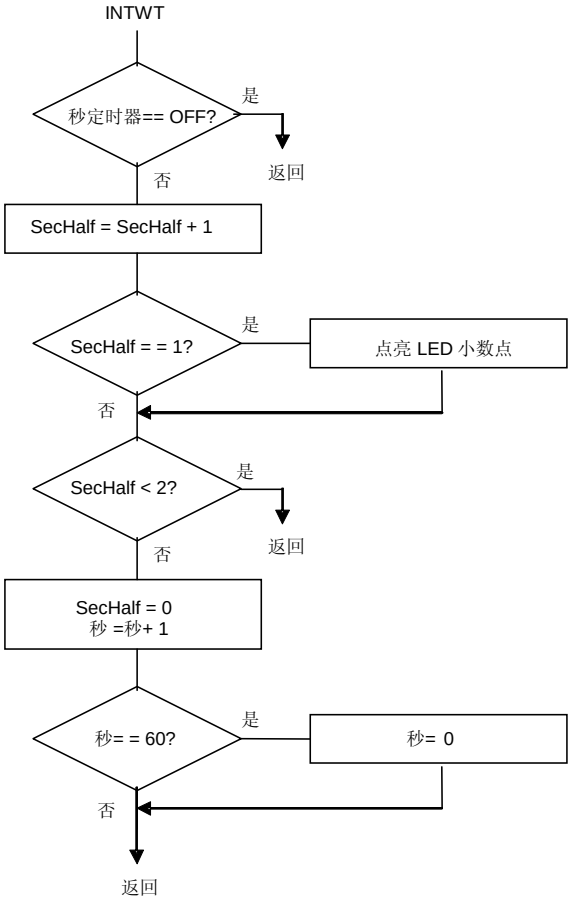
程序进入主循环。检查秒当前值和最后值是否匹配，Last_Sec。如果不相等，主程序显示秒变量和设置 Last_Sec 为秒变量。这样，主程序在显示上反射秒变量的任何变化。

程序检查如果按键按下，由检查按键状态的去抖动值实现。如果没有键按下，程序返回主程序顶部。

如果 SW2 按下，程序转换 SecondTimer 变量的状态，如果是 OFF 则变为 ON ，如果是 ON 则变为 OFF，并且等待按键抬起。

如果 SW3 按下，程序清变量秒为 0，并且等待按键抬起。

8.3.4 MD_Init( ) – 钟表定时器中断服务程序

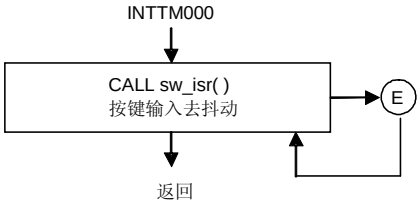


当每 0.5s 发生 INTWT 中断时，调用 MT_INTWT() 中断服务程序。如果秒定时器变量设为 OFF，程序返回，并且秒不累加。

如果秒定时器为 ON，则 SecHalf 半秒计数器累加。如果是从 0 到 1（第一个半秒结束），LED 上的小数点打开。这提供了一个可视的半秒指示器。

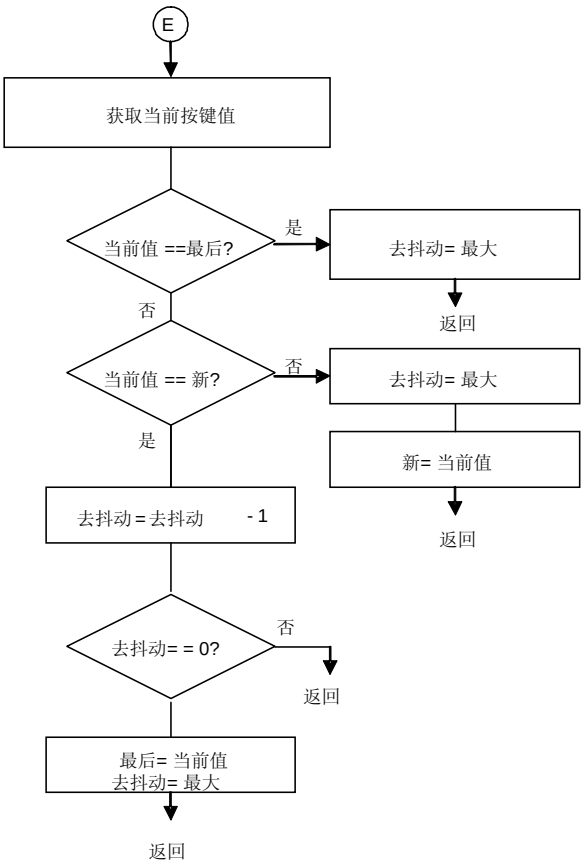
如果 SecHalf 小于 2，则程序返回。如果 SecHalf 为 2 (或包括) ，为 1 秒。秒变量累加，并且 SecHalf 设为 0。如果到了 60 秒，秒变量清 0。

8.3.5 MD_INTWTI( ) – 钟表定时器间隔中断服务程序



当每 ms 钟表定时器间隔中断 INTWTI 产生，调用 MD_INTWTI() 中断服务程序，并且在 sw_0537.c 中调用 sw_isr() 函数去抖动按键。

8.3.6 SW_isr( ) – 按键去抖动中断服务程序



此程序 (不由 Applilet 产生)每 1ms 由 MD_INTWTI( )调用。它比较按键的当前值和预置值，并且每次的值稳定到去抖动计数指定数值，更新 sw_new 变量提供按键去抖动设置。这个去抖动值由主程序调用 sw_get()程序检查。

8.4 Applilet 参考驱动程序

NEC 的 Applilet 程序生成器可自动产生 C 语言 或汇编语言代码来管理 NEC 微控制器外设。请参见使用 Applilet 版本的附录。

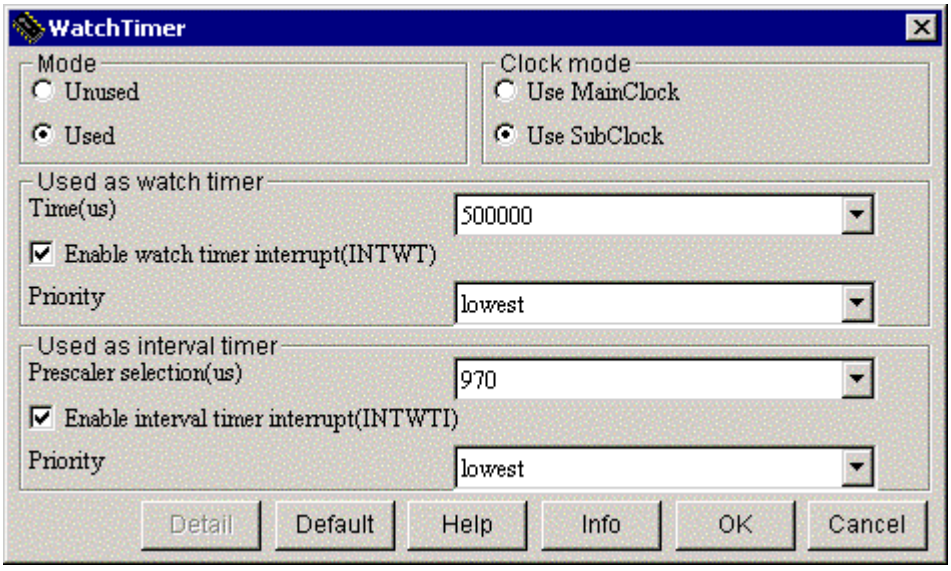
Applilet 用于创建基础的初始化代码和定时器 00 的驱动代码，并且可以管理 LED 显示输出的 I/O 端口。Applilet 创建基础代码后，用户加上自己的代码就可以实现特定的功能。

Applilet 如何产生定时器 00 的代码和产品程序列表描述如下。

启动 Applilet，创建 一个新工程文件，存为 a .prx 文件。Applilet 显示允许选择不同的外设模块的界面。

8.4.1 钟表定时器的 Applilet 配置

选择 “WatchTimer”，列表显示了钟表定时器的设置。



模式选择设为使用钟表定时器。“Use SubClock” 按钮选择时钟模式，使用 32.768 KHz 副时钟作为钟表定时器的时间基准。

在“Used as watch timer” 选项， 选择 0.5 s (500,000 us) 并且允许 INTWT 中断。

在 “Used as interval timer” 选项，选择 970 us 作为间隔时间周期，并且允许 INTWTI 中断。

8.4.2 Applilet 产生的代码

一旦钟表定时器 对话框设立，选择“Generate code” 选项。Applilet 将显示外围设备及其产生的功能， 允许选择一个目录保存源代码。当“Generate”按钮按下，Applilet 将产生必要的初始化代码，中断服务程序和钟表定时器驱动子程序。

当按下“Generate” 按钮，Applilet 创建 C 语言代码文件(extension .c) 和头文件 (extension .h)，并且在对话框中显示创建的文件列表。为了支持钟表定时器，Applilet 产生 timer.h, timer.c 和 timer_user.c，并且其他几个和定时器无关。

8.4.3 Applilet 为钟表定时器产生的文件和函数

支持定时器所产生的代码在文件 watchtimer.h, watchtimer.c 和 watchtimer_user.c 中。它们包括以下内容。

8.4.3.1 Watchtimer.h

头文件 watchtimer.h 包含了控制钟表定时器的函数控制声明。头文件 macrodriver.h，用于所有 Applilet 产生的代码，定义一些数据类型和值，例如 MD_STATUS 用于一些函数的返回值。增加变量说明和秒保持相关的值。

8.4.3.2 Watchtimer.c

源文件 watchtimer.c 包含了 Applilet 为钟表定时器产生的下列函数：

void WT_Init(void);
WT_Init() 程序依照在 Applilet 钟表定时器对话框内的指定，初始化钟表定时器。

void WT_Start(void);
WT_Start() 程序启动钟表定时器操作，并且允许 INTWT 和 INTWTI 中断。

void WT_Stop(void);
WT_Stop() 程序停止钟表定时器操作并禁止定时器中断。此程序不用于演示程序。

8.4.3.3 Watchtimer_user.c

源文件 watchtimer_user.c 包含用户代码框架函数。这些代码函数为空，允许用户增加特殊功能代码。

__interrupt void MD_INTWT(void);
这是钟表定时器中断 INTWT 的中断服务程序。由 Applilet 产生，此中断服务程序为空。使用钟表定时器来计数秒，增加代码。

__interrupt void MD_INTWTI(void);
这是间隔定时器中断 INTWTI 的中断服务程序。由 Applilet 产生，此中断服务程序为空。使用钟表定时器间隔定时器来按键去抖动，调用在 MD_INTTM000()中的 sw_isr()函数。

sw_0537.c 中的 sw_isr() 程序，有代码每 ms 检查按键值，并且一旦值稳定到必需的 ms 数，支持新值设置作为按键去抖动。一旦执行，sw_isr() 返回 MD_INTWTI()，然后当中断发生时返回到主程序。

sw_isr() 代码直接定位于 MD_INTWTI()；保存调用时间和返回 sw_isr()。允许所有按键相关函数在 sw_0537.c 源文件中。

8.4.4 其他由 Applile 产生的非关联定时器 00 的文件

在演示程序中，Applilet 产生了几个其他源文件，这些文件和它们的函数如下所示。

文件	功能
Macrodriver.h	Applilet 产生程序的通用头文件
Systeminit.c	用于初始化的 SystemInit() 和 hdwinit()函数
Main.c	主程序函数
System.h	时钟关联定义
System.c	Clock_Init()函数
Port.h	定义默认 I/O 端口状态
Port.c	PORT_Init()函数
Option.asm	定义选项字节和安全字节
Option.inc	定义设置选项字节和安全设置

8.4.5 不由 Applilet 产生的范例程序文件

演示程序包括以下文件，不由 Applilet 产生。

文件	功能
Sw_0537.h	按钮按键输入头文件
Sw_0537.c	按键读取和去抖动代码
Led_0537.h	7 段 LED 图形和功能头文件
Led_0537.c	在 7 段 LED 显示数据的代码

8.5 演示平台

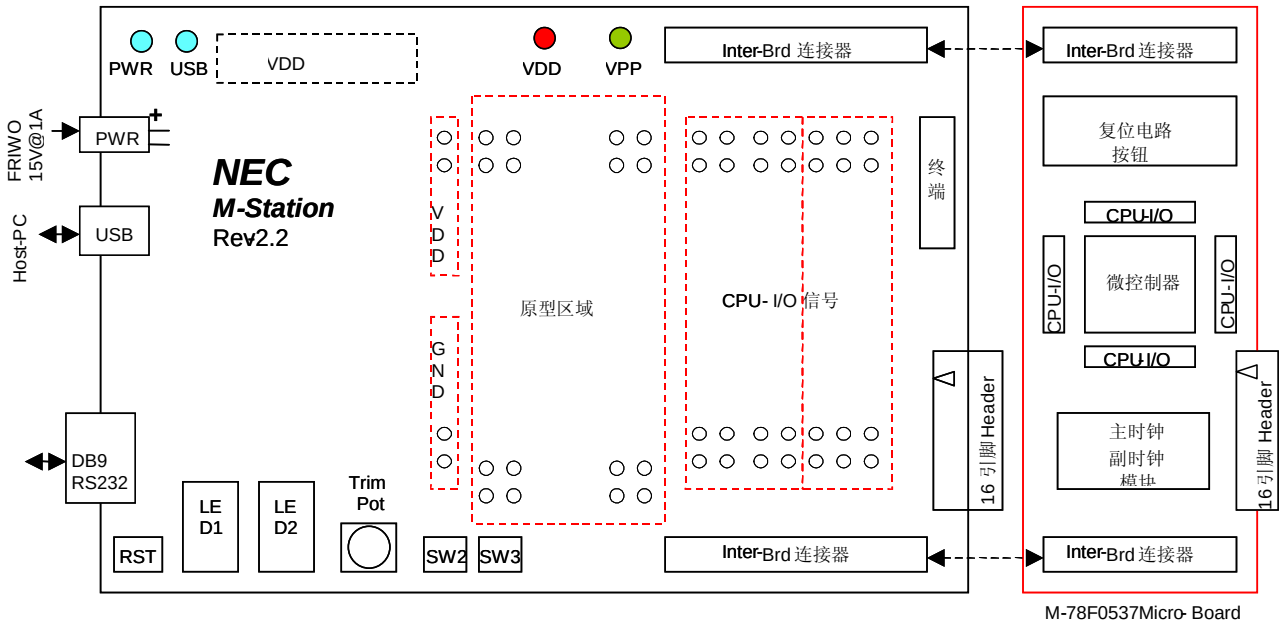
演示平台可以从本文档编写时 NEC 可用的开发工具中选择。在某些情况下，用户可以利用感兴趣的 NEC 提供的微控制器标准元器件复制相同的硬件。

8.5.1 资源

程序范例，要使用以下资源：

- M-78F0537 Micro-Board, 使用 uPD78F0537 8 位微控制器
- M-Station II 仿真系统，使用 M-Station II 资源：
  - 按钮按键 SW2 和 SW3
  - 7 段 LED LED1 和 LED2

以上硬件列表的详细描述，请参见用户手册。

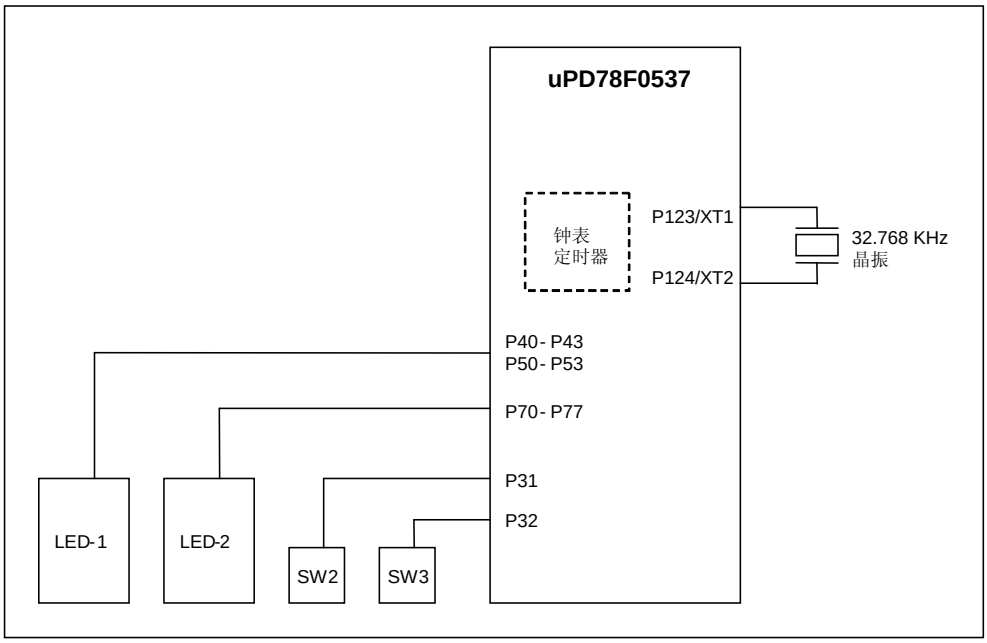


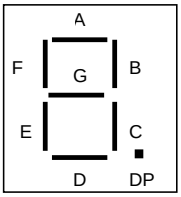
8.5.2 演示程序

假设硬件已经配置正确，并且 uPD78F0537 微控制器中已经下载了演示程序代码，演示如下：

- 观察第二次计数
- 按下 SW2　　观察第二次计数停止/启动
- 按下 SW3　　观察第二次计数复位为零

8.6 硬件框图





LED-1 和 LED -2

段	LED-1	LED-2
A	P40	P70
B	P41	P71
C	P42	P72
D	P43	P73
E	P50	P74
F	P51	P75
G	P52	P76
DP	P53	P77

8.7 软件模块

下列文件组成了演示程序的软件模块。下面的表格显示了这些文件哪些是由 Applilet 产生的，哪些需要修改才能创建演示程序。

这些文件列表在附录中。

文件	目的	由 Applilet 产生	可由用户修改
Main.c	主程序	是	是
Macrodriver.h	由 Applilet 产生的总定义	是	否
System.h	时钟关联定义	是	否
Systeminit.c	SystemInit() 和 hdwinit()函数	是	否
System.c	Clock_Init()函数	是	否
Watchtimer.h	钟表定时器关联定义	是	是[注 1]
Watchtimer.c	钟表定时器函数	是	否
Watchtimer_user.c	定时器中断用户代码	是	是
Port.h	端口关联定义	是	否
Port.c	Port_Init()函数	是	否
Led_0537.h	LED 显示定义	--	是
Led_0537.c	LED 显示函数	--	是
Sw_0537.h	按键输入定义	--	是
Sw_0537.c	按键输入函数	--	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

注 1： watchtimer.h 文件增加了时间保持的变量声明；这些变量定义在 watchtimer_user.c 中。

9 附录 A – 开发工具

以下是本应用笔记使用到的软件和硬件开发工具。

9.1 软件工具

工具	版本	注释
Applilet 适用于 78K0KE2	V1.51	适用于 78K0/KE2 的源代码产生工具
PM Plus	V5.21	程序编译连接的项目管理器
CC78K0	V3.70	NEC 78K0 的 C 编译器
RA78K0	V3.80	NEC 78K0 的汇编器
DF053764.78K	V2.00	uPD78F0537_64 设备文件

9.2 硬件工具

工具	版本	注释
M-Station 2	V2.1E	用于 NEC Micro-board 演示平台
M-78F0537	V1.0	NEC Micro-board 适用于 uPD78F0537；CPU 是 uPD78F0537DGB

10 附录 B – 软件列表

演示程序是为本应用笔记在同样的硬件开发的，使用 Applilet 程序创建工具。组成各个程序的许多文件是相同的。

为了避免重复，在以下八个部分列表是各个演示程序的独特文件。典型文件是：

Main.c	主程序
Systeminit.c	初始化外围设备代码，在不同的程序之间
Timer.h	定时器头文件，在不同的程序之间
Timer.c	
Timer_user.c	

钟表定时器范例程序， timer.h, timer.c 和 timer_user.c 文件由 watchtimer.h, watchtimer.c, 和 watchtimer_user.c 替换。

所有和大部分共同程序的文件列表在附录的第九部分。

10.1 间隔定时器使用 16 位定时器 00(TM00)

10.1.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记，间隔定时器部分*/
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "timer.h"

/* 增加包括非 Applilet 产生的函数 */
#include "led_0537.h"
#include "sw_0537.h"

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

```
*******************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void main( void )
{
unsigned char count;        /* 显示计数*/
unsigned char sw_val;       /* 按键值*/

        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;

        count = 0;                  /* 设置初值 */

        led_init();                 /* 初始化 LED */
        sw_init();          /* 初始化按键 */
        sw_set_debounce(10);    /* 设置去抖动计数器 10 用于 10 ms 稳定时间*/

        TM00_Start();   /* 启动定时器用于按键去抖动*/

        while(1){
                led_dig(count);             /* 在 LED 以十六进制显示计数*/

                sw_val = sw_get();          /* 获取去抖动按键值*/

                if (sw_val != SW_LU_RU) {
                        /* 非全抬起,一个或全按下*/
                        if (sw_val == SW_LD_RU) {
                                /* SW2 (左) 按下, 右 (SW3)抬起*/
                                count = count - 1; /* 计数递减*/
                                led_dig(count);             /* 显示 */
                                /* 等待按键改变*/
                                do {
                                        sw_val = sw_get();
                                } while (sw_val == SW_LD_RU);
                        }

                        if (sw_val == SW_LU_RD) {
                                /* SW2 抬起, SW3 按下*/
                                count = count + 1;          /* 累加计数*/
                                led_dig(count);             /* 显示 */
                                /* 用于按键改变等待*/
                                do {
                                        sw_val = sw_get();
                                } while (sw_val == SW_LU_RD);
                        }

                        if (sw_val == SW_LD_RD) {
                                /* SW2 和 SW3 都按下*/
                                count = 0;                              /* 计数复位*/
                                led_dig(count);             /* 显示 */
                                /* 等待按键全抬起*/
                                do {
                                        sw_val = sw_get();
                                } while (sw_val != SW_LU_RU);
```

```
                        }
                    } /* 结束如果(按键按下) */
            } /* 结束 while (1) 循环 */
} /* 结束 main() */
```

10.1.2 Systeminit.c

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        systeminit.c
**  概要 :          此文件执行宏初始化。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
*********************************************************************************
*/
/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "timer.h"
/*
*********************************************************************************
**  宏定义
*********************************************************************************
*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**      初始化每个宏
**
**  参数:
**      无
**
**  返回:
**      无
**
**-----------------------------------------------------------------------------
*/
void SystemInit( void )
{
    /* 时钟发生器初始化 */
    Clock_Init();
    /* 端口初始化 */
    PORT_Init();
    /* TM00 初始化 */
    TM00_Init();
}
```

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

10.1.3 Timer.h

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.h
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  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      0x4e1f
#define  TM_TM00_SQUAREWIDTH        0x4e1f
#define  TM_TM00_PPGCYCLE   0x4e1f
#define  TM_TM00_PPGWIDTH   0x00
#define  TM_TM00_ONESHOTCYCLE       0x4e1f
#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_ONESHOTCYCLE       0x00
#define  TM_TM01_ONEPULSEDELAY      0x00
```

```
#define  TM_TM50_CLOCK          0x2
#define  TM_TM50_INTERVALVALUE      0x00
#define  TM_TM50_SQUAREWIDTH        0x00
#define  TM_TM50_PWMACTIVEVALUE     0x00
#define  TM_TM51_CLOCK          0x2
#define  TM_TM51_INTERVALVALUE      0x00
#define  TM_TM51_SQUAREWIDTH        0x00
#define  TM_TM51_PWMACTIVEVALUE     0x00
#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_CARRIERDELAY       0x00
#define  TM_TMH1_CARRIERWIDTH       0x00

/* 定时器 00 到 01,50,51,H0,H1 配置初始化*/
void TM00_Init(void);

/*定时器启动*/
void TM00_Start(void);

/*定时器停止*/
void TM00_Stop(void);
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
__interrupt void MD_INTTM000(void);

#endif              /* _MDTIMER_*/
```

10.1.4 Timer.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"
/*
************************************************************************************
**  宏定义
```

```
*******************************************************************************
*/
/*TM00 脉冲宽度测量*/

/*TM01 脉冲宽度测量*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          初始化 TM00_模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void TM00_Init( )
{
        TMC00=0x00;
                                        /* 内部计数时钟*/
        PRM00 |= TM_TM00_CLOCK;
        SetIORBit(PR0H, 0x40);                  /* 低优先级*/
        ClrIORBit(IF0H, 0x40);
        /* TM00 间隔 */
        ClrIORBit(CRC00,0x01);
        CR000 = TM_TM00_INTERVALVALUE;
}
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          启动 TM00 计数器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void TM00_Start()
{
        TMC00 = 0x0c;                           /* 间隔定时器启动*/
        ClrIORBit(MK0H, 0x40);                  /* 允许 INTTM000 */
}
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          停止 TM00 计数器并清零计数寄存器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void TM00_Stop()
{
        TMC00=0x0;
        SetIORBit(MK0H, 0x40);                  /* INTTM000 停止 */
```

```
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          改变 TM00 状态。
**
**  参数:
**          USHORT* :       array_reg
**          USHORT :        array_num
**  返回:
**          MD_OK
**          MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
  switch (array_num){
       case 2:
             CR010=*(array_reg + 1);
       case 1:
             CR000=*(array_reg + 0);
             break;
       default:
             return MD_ERROR;
       }
            return MD_OK;
}
```

10.1.5 Timer_user.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer_user.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:          NEC/CC78K0
**
************************************************************************************
*/
#pragma sfr
#pragma interrupt  INTTM000 MD_INTTM000
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"

/* 增加包括用于按键的程序 */
#include "sw_0537.h"

/*
```

```
*****************************************************************************
**  宏定义
*****************************************************************************
*/
/* 定时器 00, 定时器 01 脉冲宽度测量*/
/*
**-------------------------------------------------------------------------
**
**  概要:
**          TM00 INTTM000 中断服务程序
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTTM000( )
{
        /* 调用用于去抖动按键的 ISR 程序*/
        sw_isr();
}
```

10.2 外部事件计数器使用 16 位定时器 00 (TM00)

10.2.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记, 方波外部事件计数器部分*/
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "timer.h"

/* 增加包括非 Applilet 产生的函数*/
#include "led_0537.h"
#include "sw_0537.h"

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

/*
************************************************************************************
**  全局变量
************************************************************************************
*/
unsigned char count;        /* 显示计数 */

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
```

```
**
**---------------------------------------------------------------------------
*/
void main( void )
{
unsigned char sw_val;       /* 按键值 */

        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;

        led_init();                     /* 初始化 LED */
        sw_init();          /* 初始化按键 */
        sw_set_debounce(10);    /* 设置去抖动计数器 10 用于 10 ms 稳定时间*/

        TMH0_Start();   /* 启动用于按键去抖动的定时器*/

        count = 0;                      /* 设置计数初值和显示*/
        led_dig(count);

        TM00_Start();   /* 启动外部事件计数器*/

        TM50_Start();   /* 启动方波发生器*/

        while(1){
                sw_val = sw_get();          /* 获取按键值*/

                if (sw_val == SW_LD_RU) {
                        /* SW2 (左) 抬起, 右 (SW3)按下*/
                        TM50_Stop();    /* 停止定时器 */
                        /* 设置用于半周期的定时器，周期频率 (100 Hz) */
                        TM50_ChangeTimerCondition(TM_TM50_SQUAREWIDTH / 2);
                        TM50_Start();   /*重启定时器*/
                        while (SW_LU_RU != sw_get())
                                ;                               /* 等待按键抬起*/
                }

                if (sw_val == SW_LU_RD) {
                        /* SW2 (左) 抬起, 右 (SW3)按下*/
                        TM50_Stop();    /*停止定时器*/
                        /* 设置用于最初周期的定时器, 周期频率 (50 Hz) */
                        TM50_ChangeTimerCondition(TM_TM50_SQUAREWIDTH);
                        TM50_Start();   /* 重启定时器*/
                        while (SW_LU_RU != sw_get())
                                ;                               /* 等待按键抬起*/
                }

        } /* 结束 while (1) 循环 */
} /* 结束 main() */
```

10.2.2 Systeminit.c

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        systeminit.c
**  概要 :          此文件执行宏初始化。
```

```
**  APIlib :          NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :            uPD78F0537
**
**  编译器 :          NEC/CC78K0
**
*******************************************************************************
*/
/*
*******************************************************************************
**  包含文件
*******************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "timer.h"
/*
*******************************************************************************
**  宏定义
*******************************************************************************
*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          初始化每个宏
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void SystemInit( void )
{
        /* 时钟发生器初始化 */
        Clock_Init();
        /* 端口初始化 */
        PORT_Init();
        /* TM00 初始化 */
        TM00_Init();
        /* TM50 初始化 */
        TM50_Init();
        /* TMH0 初始化 */
        TMH0_Init();
}

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

```
        SystemInit( );
        EI( );
}
```

10.2.3 Timer.h

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.h
**  概要 :          此文件执行定时器模块的设备驱动程序。
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  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_ONESHOTCYCLE       0x00
#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_ONESHOTCYCLE       0x00
#define  TM_TM01_ONEPULSEDELAY      0x00
#define  TM_TM50_CLOCK          0x7
#define  TM_TM50_INTERVALVALUE      0x17
#define  TM_TM50_SQUAREWIDTH        0x17
#define  TM_TM50_PWMACTIVEVALUE     0x17
#define  TM_TM51_CLOCK          0x2
#define  TM_TM51_INTERVALVALUE      0x00
#define  TM_TM51_SQUAREWIDTH        0x00
#define  TM_TM51_PWMACTIVEVALUE     0x00
#define  TM_TMH0_CLOCK          0x4
#define  TM_TMH0_INTERVALVALUE      0x12
#define  TM_TMH0_SQUAREWIDTH        0x12
#define  TM_TMH0_PWMCYCLE   0x12
#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_CARRIERDELAY        0x00
#define  TM_TMH1_CARRIERWIDTH        0x00
#define  TM_TM00_EXTERNALCOUNT       0x31


/* 定时器 00 到 01,50,51,H0,H1 配置初始化*/
void TM00_Init(void);
void TM50_Init(void);
void TMH0_Init(void);

/*定时器启动*/
void TM00_Start(void);
void TM50_Start(void);
void TMH0_Start(void);

/*定时器停止*/
void TM00_Stop(void);
void TM50_Stop(void);
void TMH0_Stop(void);
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
MD_STATUS TM50_ChangeTimerCondition(UCHAR value);
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num);
__interrupt void MD_INTTM000(void);
__interrupt void MD_INTTMH0(void);

#endif              /* _MDTIMER_*/
```

10.2.4 Timer.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.c
**  概要 :          此文件执行定时器模块的设备驱动程序。
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"
/*
************************************************************************************
**  宏定义
************************************************************************************
*/
/*TM00 脉冲宽度测量*/

/*TM01 脉冲宽度测量*/

/*
```

```
**-------------------------------------------------------------------------
**
**  概要:
**          初始化 TM00_模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TM00_Init( )
{
        TMC00=0x00;
        SetIORBit(PR0H, 0x40);                  /* 低优先级 */
        ClrIORBit(IF0H, 0x40);
        /* TM00 外部事件计数器*/
        ClrIORBit(CRC00,0x01);
        SetIORBit(PRM00,0x03);
        CR000 = TM_TM00_EXTERNALCOUNT;
        CR010 = 0xffff;

        SetIORBit(PM0,0x1);                     /* TI000(P00)输入 */
        ClrIORBit(ISC,0x2);
        ClrIORBit(PRM00,0x30);                  /* TI000 下降沿 */
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          启动 TM00 计数器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TM00_Start()
{
        TMC00 = 0x0c;                           /* 外部事件计数启动*/
        ClrIORBit(MK0H, 0x40);                  /* 允许 INTTM000 */
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          停止 TM00 计数器和清零计数寄存器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TM00_Stop()
{
        TMC00=0x0;
        SetIORBit(MK0H, 0x40);                  /* INTTM000 停止 */
}
/*
**-------------------------------------------------------------------------
```

```
**
**  概要:
**          改变 TM00 状态。
**
**  参数:
**          USHORT* :       array_reg
**          USHORT :        array_num
**  返回:
**          MD_OK
**          MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
  switch (array_num){
        case 2:
              CR010=*(array_reg + 1);
        case 1:
              CR000=*(array_reg + 0);
              break;
        default:
              return MD_ERROR;
        }
             return MD_OK;
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          初始化 TM50_模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TM50_Init( void )
{
          ClrIORBit(TMC50, 0x80);
          TCL50 = TM_TM50_CLOCK;                          /* 内部计数时钟*/
          /* TM50 方波输出*/
          ClrIORBit(P1, 0x80);
          ClrIORBit(PM1, 0x80);                     /* TO50 输出 */
          SetIORBit(TMC50, 0xb);
          CR50 = TM_TM50_SQUAREWIDTH;
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          启动 TM50 计数器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TM50_Start( void )
{
          /* TM50 方波输出*/
          SetIORBit(TMC50, 0x8b);
```

```
}
/*
**---------------------------------------------------------------------------
**
**  概要:
**          停止 TM50 计数器并清零计数寄存器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void TM50_Stop( void )
{
        ClrIORBit(TMC50, 0x80);
}
/*
**---------------------------------------------------------------------------
**
**  概要:
**          改变 TM50 状态。
**
**  参数:
**          UCHAR :             value
**  返回:
**          MD_OK
**          MD_ERROR
**
**---------------------------------------------------------------------------
*/
MD_STATUS TM50_ChangeTimerCondition(UCHAR value)
{
        CR50 =value;
        return MD_OK;
}
/*
**---------------------------------------------------------------------------
**
**  概要:
**          初始化 TMH0 模块。
**
**  参数:
**          无
**
**  返回:
**          无
**---------------------------------------------------------------------------
*/
void TMH0_Init( void )
{
        ClrIORBit(TMHMD0, 0x80);
                        /* 计数时钟=fx/1024 */
        TMHMD0 |= (TM_TMH0_CLOCK << 4);

        SetIORBit(PR0H, 0x10);                  /* 低优先级 */
        ClrIORBit(IF0H, 0x10);
        /* TMH0 间隔定时器 */
        ClrIORBit(TMHMD0, 0x8c);
        CMP00 = TM_TMH0_INTERVALVALUE;
}
/*
**---------------------------------------------------------------------------
**
**  概要:
**          启动 TMH0 计数器。
```

```
**
**  参数:
**          无
**
**  返回:
**          无
**---------------------------------------------------------------------------
*/
void TMH0_Start( void )
{
        SetIORBit(TMHMD0, 0x80);
        ClrIORBit(MK0H, 0x10);                  /* 允许 INTTMH0 */
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**          停止 TMH0 计数器操作。
**
**  参数:
**          无
**
**  返回:
**          无
**---------------------------------------------------------------------------
*/
void TMH0_Stop( void )
{
        ClrIORBit(TMHMD0, 0x80);
        SetIORBit(MK0H, 0x10);                  /* 禁止 INTTMH0 */
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**          改变 TMH0 状态。
**
**  参数:
**          UCHAR* :        array_reg
**          UCHAR :         array_num
**  返回:
**          MD_OK
**          MD_ERROR
**---------------------------------------------------------------------------
*/
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num)
{
  switch (array_num){
      case 2:
            CMP10=*(array_reg + 1);
      case 1:
            CMP00=*(array_reg + 0);
            break;
      default:
            return MD_ERROR;
      }
           return MD_OK;
}
```

10.2.5 Timer_user.c

```
/*
*******************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
```

```
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer_user.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
*****************************************************************************
*/
#pragma sfr
#pragma interrupt  INTTM000 MD_INTTM000
#pragma interrupt  INTTMH0 MD_INTTMH0
/*
*****************************************************************************
**  包含文件
*****************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"

/* 增加包括用于按键程序和 LED 程序 */
#include "sw_0537.h"
#include "led_0537.h"

/*
*****************************************************************************
**  宏定义
*****************************************************************************
*/
/* 定义外部全局变量计数所以 ISR 累加*/
extern unsigned char count;

/* 定时器 00, 定时器 01 脉冲宽度测量*/
/*
**-------------------------------------------------------------------------
**
**  概要:
**          TM00 INTTM000 中断服务程序
**
**  参数:
**          无
**
**  返回:
**          无
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTTM000( )
{
        count = count + 1;
        led_dig(count);
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          TMH0 INTTMH0 中断服务程序
**
**  参数:
**          无
**
```

```
**  返回:
**          无
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTTMH0( )
{
        /* 调用按键 ISR 程序用于去抖动按键 */
        sw_isr();
}
```

10.3 单脉冲发生使用 16 位定时器 01 (TM01)

10.3.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记, 单脉冲外部事件计数器部分*/
/*
************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************
*/
/*
************************************************************************
**  包含文件
************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "timer.h"

/* 增加包括非 Applilet 产生的函数 */
#include "led_0537.h"
#include "sw_0537.h"

/*
************************************************************************
**  宏定义
************************************************************************
*/
/*
************************************************************************
**  全局变量
************************************************************************
*/
unsigned char count;        /* 显示计数 */
/*
**-----------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
**-----------------------------------------------------------------------
*/
```

```
void main( void )
{
unsigned char sw_val;       /* 按键值 */
int i;

        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;

        led_init();                     /* 初始化 LEDs */
        sw_init();          /* 初始化按键 */
        sw_set_debounce(10);    /* 设置去抖动计数器 10 用于 10 ms 稳定时间*/

        TMH0_Start();   /* 启动按键去抖动定时器*/

        count = 0;                      /* 设置计数初值和显示*/
        led_dig(count);

        TM00_Start();   /* 启动外部事件计数器*/

        TM01_Start();   /* 获取单脉冲发生器*/

        while(1){
                sw_val = sw_get();          /* 获取按键值*/

                if (sw_val == SW_LD_RU) {
                        /* SW2 (左) 按下, 右 (SW3)抬起 */
                        /* 创建信号脉冲*/
                        TM01_OneshotTriggerOn();
                        while (SW_LU_RU != sw_get())
                                ;                               /* 等待按键抬起*/
                }

        } /* 结束 while (1) 循环 */
} /* 结束 main() */
```

10.3.2 Systeminit.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        systeminit.c
**  概要 :          此文件执行宏初始化。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
```

```
#include "port.h"
#include "timer.h"
/*
************************************************************************************
**  宏定义
************************************************************************************
*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          初始化每个模块
**
**  参数:
**          无
**
**  返回:
**          无
**-----------------------------------------------------------------------------
*/
void SystemInit( void )
{
        /* 时钟发生器初始化*/
        Clock_Init();
        /* 端口初始化*/
        PORT_Init();
        /* TM00 初始化*/
        TM00_Init();
        /* TM01 初始化*/
        TM01_Init();
        /* TMH0 初始化*/
        TMH0_Init();
}
/*
**-----------------------------------------------------------------------------
**  概要:
**          初始化硬件设置
**
**  参数:
**          无
**
**  返回:
**          无
**-----------------------------------------------------------------------------
*/
void hdwinit( void )
{
        DI( );
        SystemInit( );
        EI( );
}
```

10.3.3 Timer.h

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
```

```
** 文件名 :        timer.h
** 概要 :          此文件执行定时器模块的设备驱动程序
** APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
** 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_ONESHOTCYCLE       0x00
#define  TM_TM00_ONEPULSEDELAY      0x00
#define  TM_TM01_CLOCK          0x0
#define  TM_TM01_INTERVALVALUE      0x7cf
#define  TM_TM01_SQUAREWIDTH        0x7cf
#define  TM_TM01_PPGCYCLE   0x7cf
#define  TM_TM01_PPGWIDTH   0x3e7
#define  TM_TM01_ONESHOTCYCLE       0x7cf
#define  TM_TM01_ONEPULSEDELAY      0x3e7
#define  TM_TM50_CLOCK          0x2
#define  TM_TM50_INTERVALVALUE      0x00
#define  TM_TM50_SQUAREWIDTH        0x00
#define  TM_TM50_PWMACTIVEVALUE     0x00
#define  TM_TM51_CLOCK          0x2
#define  TM_TM51_INTERVALVALUE      0x00
#define  TM_TM51_SQUAREWIDTH        0x00
#define  TM_TM51_PWMACTIVEVALUE     0x00
#define  TM_TMH0_CLOCK          0x4
#define  TM_TMH0_INTERVALVALUE      0x12
#define  TM_TMH0_SQUAREWIDTH        0x12
#define  TM_TMH0_PWMCYCLE   0x12
#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_CARRIERDELAY       0x00
#define  TM_TMH1_CARRIERWIDTH       0x00
#define  TM_TM00_EXTERNALCOUNT      0x4


/* 定时器 00 到 01,50,51,H0,H1 配置初始化*/
void TM00_Init(void);
void TM01_Init(void);
void TMH0_Init(void);

/*启动定时器*/
void TM00_Start(void);
void TM01_Start(void);
void TMH0_Start(void);

/*定时器停止*/
void TM00_Stop(void);
void TM01_Stop(void);
void TMH0_Stop(void);
```

```
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
MD_STATUS TM01_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
void TM01_OneshotTriggerOn();
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num);
__interrupt void MD_INTTM000(void);
__interrupt void MD_INTTMH0(void);

#endif              /* _MDTIMER_*/
```

10.3.4 Timer.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/

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


/*
************************************************************************************
**  宏定义
************************************************************************************
*/
/*TM00 脉冲宽度测量*/

/*TM01 脉冲宽度测量*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**      初始化 TM00_模块。
**
**  参数:
**      无
**
**  返回:
**      无
**
**-----------------------------------------------------------------------------
*/
```

```
void TM00_Init( )
{
        TMC00=0x00;
        SetIORBit(PR0H, 0x40);                  /* 低优先级*/
        ClrIORBit(IF0H, 0x40);
        /* TM00 外部事件计数器*/
        ClrIORBit(CRC00,0x01);
        SetIORBit(PRM00,0x03);
        CR000 = TM_TM00_EXTERNALCOUNT;
        CR010 = 0xffff;

        SetIORBit(PM0,0x1);                     /* TI000(P00) 输入 */
        ClrIORBit(ISC,0x2);
        ClrIORBit(PRM00,0x30);                  /* TI000 下降沿 */
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**        启动 TM00 计数器。
**
**  参数:
**        无
**
**  返回:
**        无
**
**-------------------------------------------------------------------------
*/
void TM00_Start()
{
        TMC00 = 0x0c;                           /* 启动外部事件计数*/
        ClrIORBit(MK0H, 0x40);                  /* 允许 INTTM000 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**        停止 TM00 计数器并清零计数寄存器。
**
**  参数:
**        无
**
**  返回:
**        无
**
**-------------------------------------------------------------------------
*/
void TM00_Stop()
{
        TMC00=0x0;
        SetIORBit(MK0H, 0x40);                  /* 停止 INTTM000 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**        改变 TM00 状态。
**
**  参数:
**        USHORT* :       array_reg
**        USHORT :        array_num
**  返回:
**        MD_OK
**        MD_ERROR
**
```

```
**-------------------------------------------------------------------------
*/
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
  switch (array_num){
       case 2:
            CR010=*(array_reg + 1);
       case 1:
            CR000=*(array_reg + 0);
            break;
       default:
            return MD_ERROR;
       }
           return MD_OK;
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**       经软件触发产生单脉冲输出。
**
**  参数:
**       无
**
**  返回:
**       无
**
**-------------------------------------------------------------------------
*/
void TM01_OneshotTriggerOn()
{
       SetIORBit(TOC01, 0x40);
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**       初始化 TM01_模块。
**
**  参数:
**       无
**
**  返回:
**       无
**
**-------------------------------------------------------------------------
*/
void TM01_Init( )
{
       TMC01=0x00;
                                        /* 内部计数时钟 */
       PRM01 |= TM_TM01_CLOCK;
       /*TM01 单脉冲输出*/
       ClrIORBit(CRC01, 0x05);                  /* 用作比较寄存器 */

       CR001 = TM_TM01_ONESHOTCYCLE;
       CR011 = TM_TM01_ONEPULSEDELAY;

       ClrIORBit(P0,0x40);                      /* TO01(p06) 输出 */
       ClrIORBit(PM0,0x40);
       TOC01 = 0x37;                            /* 初值低 */
       ClrIORBit(TOC01, 0x40);                  /* 软件触发*/
}
/*
**-------------------------------------------------------------------------
**
**  概要:
```

```
**          启动 TM01 计数器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TM01_Start(void)
{
        TMC01 = 0x04;                   /* 启动单脉冲输出*/
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          停止 TM01 模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TM01_Stop(void)
{
        TMC01=0x0;
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          改变 TM01 状态。
**
**  参数:
**          USHORT* :       array_reg
**          USHORT :        array_num
**  返回:
**          MD_OK
**          MD_ERROR
**-------------------------------------------------------------------------
*/
MD_STATUS TM01_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
  switch (array_num){
        case 2:
                CR011=*(array_reg + 1);
        case 1:
                CR001=*(array_reg + 0);
                break;
        default:
                return MD_ERROR;
        }
              return MD_OK;
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          初始化 TMH0 模块。
**
**  参数:
```

```
**         无
**
**  返回:
**         无
**-------------------------------------------------------------------------
*/
void TMH0_Init( void )
{
        ClrIORBit(TMHMD0, 0x80);
                        /* 计数时钟=fx/1024 */
        TMHMD0 |= (TM_TMH0_CLOCK << 4);

        SetIORBit(PR0H, 0x10);                  /* 低优先级 */
        ClrIORBit(IF0H, 0x10);
        /* TMH0 间隔定时器*/
        ClrIORBit(TMHMD0, 0x8c);
        CMP00 = TM_TMH0_INTERVALVALUE;
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**         启动 TMH0 计数器。
**
**  参数:
**         无
**
**  返回:
**         无
**-------------------------------------------------------------------------
*/
void TMH0_Start( void )
{
        SetIORBit(TMHMD0, 0x80);
        ClrIORBit(MK0H, 0x10);                  /* 允许 INTTMH0 */
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**         停止 TMH0 计数器操作。
**
**  参数:
**         无
**
**  返回:
**         无
**-------------------------------------------------------------------------
*/
void TMH0_Stop( void )
{
        ClrIORBit(TMHMD0, 0x80);
        SetIORBit(MK0H, 0x10);                  /* 禁止 INTTMH0 */
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**         改变 TMH0 状态。
**
**  参数:
**         UCHAR* :        array_reg
**         UCHAR :         array_num
**  返回:
**         MD_OK
**         MD_ERROR
**-------------------------------------------------------------------------
*/
```

```
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num)
{
  switch (array_num){
      case 2:
            CMP10=*(array_reg + 1);
      case 1:
            CMP00=*(array_reg + 0);
            break;
      default:
            return MD_ERROR;
      }
          return MD_OK;
}
```

10.3.5 Timer_user.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer_user.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/

#pragma sfr
#pragma interrupt  INTTM000 MD_INTTM000
#pragma interrupt  INTTMH0 MD_INTTMH0
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"

/* 增加包括用于按键程序和 LED 程序 */
#include "sw_0537.h"
#include "led_0537.h"

/*
************************************************************************************
**  宏定义
************************************************************************************
*/
/* 定义外部变量计数所以 ISR 累加*/
extern unsigned char count;

/* 定时器 00, 定时器 01 脉冲宽度测量*/
/*
**-------------------------------------------------------------------------
**
**  概要:
**          TM00 INTTM000 中断服务程序
```

```
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------------
*/
__interrupt void MD_INTTM000( )
{
        count = count + 1;
        led_dig(count);
}
/*
**-------------------------------------------------------------------------------
**
**  概要:
**          TMH0 INTTMH0 中断服务程序
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------------
*/
__interrupt void MD_INTTMH0( )
{
        /* 调用用于去抖动的按键 ISR 程序*/
        sw_isr();
}
```

10.4 PWM 输出用于 D/A 转换使用 8 位定时器 51 (TM51)

10.4.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记, PWM D/A 转换器部分*/
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "ad.h"
#include "port.h"
#include "timer.h"

/* 增加包括非 Applilet 产生的函数*/
#include "led_0537.h"

/*
************************************************************************************
**  宏定义
************************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void main( void )
{
int i;
USHORT value;
```

```
        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;

        led_init();                     /* 初始化 LED */

        TM51_Start();       /* 启动定义值 PWM */

        AD_Start();                     /* 启动 A/D 转换器 */

        while(1){
                /* 获取 10 位 A/D 转换值*/
                AD_Read(&value);

                /* 转换 8 位值, 最高有效位并显示 */
                value = value >> 2;
                led_dig((UCHAR)(value & 0xFF));

                /* 设置 PWM 宽度 */
                /* 设置 CR51 的 PWM 延时 */
                /* 必须是至少 3 个计数时钟*/
                /* TM51 PWM 以 fPRS 运行, 没必要实际延时*/
                /* 但万一 PWM 时钟降低将有提示*/
                for (i=0; i < 10; i++)
                        ;               /* 延时 */
                TM51_ChangeTimerCondition((UCHAR)(value & 0xFF));

        } /* 结束 while (1) 循环 */
} /* 结束 main() */
```

10.4.2 Systeminit.c

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        systeminit.c
**  概要 :          此文件执行宏初始化。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
*********************************************************************************
*/
/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "ad.h"
#include "port.h"
#include "timer.h"
/*
*********************************************************************************
**  宏定义
```

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

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          初始化每个宏
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void SystemInit( void )
{
        /* 时钟发生器初始化*/
        Clock_Init();
        /* 端口初始化*/
        PORT_Init();
        /* AD 初始化*/
        AD_Init();
        /* TM51 初始化 */
        TM51_Init();
}

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

10.4.3 Timer.h

```
/*
*******************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.h
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
```

```
**
** 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_ONESHOTCYCLE       0x00
#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_ONESHOTCYCLE       0x00
#define  TM_TM01_ONEPULSEDELAY      0x00
#define  TM_TM50_CLOCK          0x2
#define  TM_TM50_INTERVALVALUE      0x00
#define  TM_TM50_SQUAREWIDTH        0x00
#define  TM_TM50_PWMACTIVEVALUE     0x00
#define  TM_TM51_CLOCK          0x2
#define  TM_TM51_INTERVALVALUE      0x77
#define  TM_TM51_SQUAREWIDTH        0x77
#define  TM_TM51_PWMACTIVEVALUE     0x77
#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_CARRIERDELAY       0x00
#define  TM_TMH1_CARRIERWIDTH       0x00


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

/*定时器启动*/
void TM51_Start(void);

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

#endif              /* _MDTIMER_*/
```

10.4.4 Timer.c

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

```
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
*********************************************************************************
*/

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


/*
*********************************************************************************
**  宏定义
*********************************************************************************
*/
/*TM00 脉冲宽度测量*/

/*TM01 脉冲宽度测量*/

/*
**-------------------------------------------------------------------------
**
**  概要:
**      初始化 TM51_模块。
**
**  参数:
**      无
**
**  返回:
**      无
**
**-------------------------------------------------------------------------
*/
void TM51_Init( void )
{
    ClrIORBit(TMC51, 0x80);
    TCL51 = TM_TM51_CLOCK;                  /* 计数时钟=fx */
    /* TM51 PWM 输出 */
    ClrIORBit(P3, 0x08);
    ClrIORBit(PM3, 0x08);                   /* TO51 输出 */
    SetIORBit(TMC51, 0x1);
    SetIORBit(TMC51, 0x40);
    ClrIORBit(TMC51, 0x8);
    SetIORBit(TMC51, 0x4);
    ClrIORBit(TMC51, 0x2);                  /* 有效高 */
    CR51 = TM_TM51_PWMACTIVEVALUE;
}
```

```
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          启动 TM51 计数器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void TM51_Start( void )
{
        /* TM51 PWM 输出 */
        ClrIORBit(TMC51, 0x8);
        SetIORBit(TMC51, 0x4);
        SetIORBit(TMC51, 0x80);
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          停止 TM51 计数器并清零计数寄存器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void TM51_Stop( void )
{
        ClrIORBit(TMC51, 0x80);
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          改变 TM51 状态。
**
**  参数:
**          UCHAR :             value
**  返回:
**          MD_OK
**          MD_ERROR
**
**-----------------------------------------------------------------------------
*/
MD_STATUS TM51_ChangeTimerCondition(UCHAR value)
{
        CR51 =value;
        return MD_OK;
}
```

10.4.5 Timer_user.c

```
/*
*******************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
```

```
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer_user.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  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 脉冲宽度测量*/
```

10.5 遥控载波发生使用 TM51 和 TMH1

10.5.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记, 载波发生部分*/
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
*********************************************************************************
*/
/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "timer.h"
/*
*********************************************************************************
**  宏定义
*********************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void main( void )
{
UCHAR bval[2];
int i;

        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;

        /* 调试, P75 - P77 输出 */
        PM7 &= 0x1F;
        P7 = 0x00;
```

```
        while(1){
                P7 = 0x00;                  /* 返回 P76 和 P75 为 0*/
                P7 ^= 0x80;                 /* P77 触发脉冲启动 */
                for (i=0; i<25; i++)
                        ;
                P7 ^= 0x80;

                ByteValue = 0xAA;           /* 位 0 = 0, 位 1 = 1 */
                BitCount  = 2;              /* 发送 2 位(0 然后 1) */
                SendDone  = 0;

                /* 允许前设定 TimerH1 值*/
                bval[0] = TM_TMH1_CARRIERDELAY;
                bval[1] = TM_TMH1_CARRIERWIDTH;
                TMH1_ChangeTimerCondition(bval, 1);
                TMH1_ChangeTimerCondition(bval, 2);

                TMH1_CarrierOutputEnable();
                TMH1_Start();

                TM51_ChangeTimerCondition(2);   /* 设置低值第一次中断*/
                TM51_Start();

                while (SendDone == 0)
                        ;
                TM51_Stop();
                TMH1_Stop();
        }
}
```

10.5.2 Systeminit.c

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        systeminit.c
**  概要 :          此文件执行宏初始化。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
*********************************************************************************
*/
/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "timer.h"
/*
*********************************************************************************
**  宏定义
*********************************************************************************
```

```
*/

/*
**---------------------------------------------------------------------------
**
**  概要:
**          初始化每个宏
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void SystemInit( void )
{
        /* 时钟发生器初始化*/
        Clock_Init();
        /* TM51 初始化 */
        TM51_Init();
        /* TMH1 初始化 */
        TMH1_Init();
}

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

10.5.3 Timer.h

```
/*
*******************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.h
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  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_ONESHOTCYCLE       0x00
#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_ONESHOTCYCLE       0x00
#define  TM_TM01_ONEPULSEDELAY      0x00
#define  TM_TM50_CLOCK          0x2
#define  TM_TM50_INTERVALVALUE      0x00
#define  TM_TM50_SQUAREWIDTH        0x00
#define  TM_TM50_PWMACTIVEVALUE     0x00
#define  TM_TM51_CLOCK          0x6
#define  TM_TM51_INTERVALVALUE      0x82
#define  TM_TM51_SQUAREWIDTH        0x82
#define  TM_TM51_PWMACTIVEVALUE     0x82
#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          0x1
#define  TM_TMH1_INTERVALVALUE      0x59
#define  TM_TMH1_SQUAREWIDTH        0x59
#define  TM_TMH1_PWMCYCLE   0x59
#define  TM_TMH1_PWMDELAY   0x29
#define  TM_TMH1_CARRIERDELAY       0x59
#define  TM_TMH1_CARRIERWIDTH       0x29


/* 定时器 00 到 01,50,51,H0,H1 配置初始化*/
void TM51_Init(void);
void TMH1_Init(void);

/*定时器启动*/
void TM51_Start(void);
void TMH1_Start(void);

/*定时器停止*/
void TM51_Stop(void);
void TMH1_Stop(void);
MD_STATUS TM51_ChangeTimerCondition(UCHAR value);
MD_STATUS TMH1_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num);
void TMH1_CarrierOutputEnable();
void TMH1_CarrierOutputDisable();
__interrupt void MD_INTTM51(void);
__interrupt void MD_INTTMH1(void);

/* 增加全局变量 */
extern UCHAR ByteValue;
```

```
extern UCHAR BitCount;
extern UCHAR SendDone;

/* 增加 CR51 值定义*/
#define TM_TM51_CR51_START   1
#define TM_TM51_CR51_HI                  43
#define  TM_TM51_CR51_ZLO     43
#define  TM_TM51_CR51_OLO     130

#endif              /* _MDTIMER_*/
```

10.5.4 Timer.c

```
/*
*****************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.c
**  概要 :          此文件执行定时器模块的设备驱动程序。
**  APIlib:         NEC78K0KX2.lib V1.01 [02005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
*****************************************************************************
*/

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


/*
*****************************************************************************
**  宏定义
*****************************************************************************
*/
/*TM00 脉冲宽度测量*/

/*TM01 脉冲宽度测量*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**      初始化 TM51_模块。
**
**  参数:
**      无
**
**  返回:
**      无
**
```

```
**-----------------------------------------------------------------------
*/
void TM51_Init( void )
{
        ClrIORBit(TMC51, 0x80);
        TCL51 = TM_TM51_CLOCK;                          /* 计数时钟=fx/256 */
        SetIORBit(PR1L, 0x08);                  /* 低优先级 */
        ClrIORBit(IF1L, 0x08);
        /* TM51 间隔 */
        CR51 = TM_TM51_INTERVALVALUE;
}

/*
**-----------------------------------------------------------------------
**
**  概要:
**      启动 TM51 计数器。
**
**  参数:
**      无
**
**  返回:
**      无
**
**-----------------------------------------------------------------------
*/
void TM51_Start( void )
{
        /* TM51 间隔 */
        SetIORBit(TMC51, 0x80);
        ClrIORBit(MK1L, 0x08);                  /* 允许 INTTM51 */
}

/*
**-----------------------------------------------------------------------
**
**  概要:
**      停止 TM51 计数器并清零计数寄存器。
**
**  参数:
**      无
**
**  返回:
**      无
**
**-----------------------------------------------------------------------
*/
void TM51_Stop( void )
{
        ClrIORBit(TMC51, 0x80);
        SetIORBit(MK1L, 0x08);                  /* 禁止 INTTM51 */
}

/*
**-----------------------------------------------------------------------
**
**  概要:
**      改变 TM51 状态。
**
**  参数:
**      UCHAR :         value
**  返回:
**      MD_OK
**      MD_ERROR
**
**-----------------------------------------------------------------------
*/
MD_STATUS TM51_ChangeTimerCondition(UCHAR value)
```

```
{
        CR51 =value;
        return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**        初始化 TMH1_模块。
**
**  参数:
**        无
**
**  返回:
**        无
**
**-------------------------------------------------------------------------
*/
void TMH1_Init( void )
{
        ClrIORBit(TMHMD1, 0x80);
                        /* 内部计数时钟 */
        TMHMD1 |= (TM_TMH1_CLOCK << 4);

        SetIORBit(PR0H, 0x08);                  /* 低优先级 */
        ClrIORBit(IF0H, 0x08);
        /* TMH1 载波发生输出*/
        SetIORBit(TMHMD1, 0x04);
        ClrIORBit(P1, 0x40);
        ClrIORBit(PM1, 0x40);                   /* P16/TOH1 */
        SetIORBit(TMHMD1, 0x1);
        SetIORBit(TMCYC1, 0x7);
        CMP01 = TM_TMH1_CARRIERDELAY;
        CMP11 = TM_TMH1_CARRIERWIDTH;
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**        启动 TMH1 计数器。
**
**  参数:
**        无
**
**  返回:
**        无
**
**-------------------------------------------------------------------------
*/
void TMH1_Start( void )
{
        /* TMH1 载波输出*/
        SetIORBit(TMHMD1, 0x80);
        ClrIORBit(MK0H, 0x08);                  /* 允许 INTTMH1 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**        停止 TMH1 计数器操作。
**
**  参数:
**        无
**
**  返回:
```

```
**         无
**
**-------------------------------------------------------------------------
*/
void TMH1_Stop( void )
{
        ClrIORBit(TMHMD1, 0x80);
        SetIORBit(MK0H, 0x08);                  /* 禁止 INTTMH1 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**         改变 TMH1 状态。
**
**  参数:
**         UCHAR* :        array_reg
**         UCHAR :         array_num
**  返回:
**         MD_OK
**         MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TMH1_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num)
{
  switch (array_num){
      case 2:
            CMP11=*(array_reg + 1);
      case 1:
            CMP01=*(array_reg + 0);
            break;
      default:
            return MD_ERROR;
      }
           return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**         允许载波输出。
**
**  参数:
**         无
**
**  返回:
**         无
**
**-------------------------------------------------------------------------
*/
void TMH1_CarrierOutputEnable( void )
{
        SetIORBit(TMCYC1, 0x02);
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**         禁止载波输出。
**
**  参数:
**         无
**
**  返回:
**         无
```

```
**
**-------------------------------------------------------------------------
*/
void TMH1_CarrierOutputDisable( void )
{
        ClrIORBit(TMCYC1, 0x02);
}
```

10.5.5 Timer_user.c

```
/*
*****************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer_user.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
*****************************************************************************
*/

#pragma sfr
#pragma interrupt  INTTM51 MD_INTTM51
#pragma interrupt  INTTMH1 MD_INTTMH1
/*
*****************************************************************************
**  包含文件
*****************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"

/* 增加全局变量 */
UCHAR ByteValue;
UCHAR BitCount;
UCHAR SendDone;

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

/* Timer00, Timer01 脉冲宽度持续时间*/
/*
**-------------------------------------------------------------------------
**
**  参数:
**        TM51 INTTM51 中断服务程序
**
**  参数:
**        无
**
**  返回:
```

```
**          无
**
**-------------------------------------------------------------------------
*/
#define NRZB1 TMCYC1.1
__interrupt void MD_INTTM51( )
{
        /* TODO */
        P7 ^= 0x20;         /* P75  INTTM51 */

        /* 如果位计数为零，将执行 */
        if (BitCount == 0) {
                TMH1_CarrierOutputDisable();
                SendDone = 1;
                return;
        }

        /* 等待直到 NRZ1 位改变为 NRZB1 */
        while (NRZB1 != NRZ1)
                ;
        if (NRZB1 == 1) {
                /* 启动位第一 (高) 部分 */
                NRZB1 = 0;
                CR51 = TM_TM51_CR51_HI;
        } else {
                /* NRZB1 == 0, 启动位第二(低) 部分 */
                if ((ByteValue & 0x01) == 0x01) {
                        /* LSB 为 1, 设置 1 的低宽度定时器 */
                        CR51 = TM_TM51_CR51_OLO;
                } else {
                        /* LSB 为 0, 设置 0 的低宽度定时器*/
                        CR51 = TM_TM51_CR51_ZLO;
                }
                ByteValue = ByteValue >> 1;          /* 转换下一位 */
                BitCount--;          /* 位计数值减*/
                if (BitCount != 0) {
                        /* 还有其他位，设置下一时间 NRZB1 为 1 */
                        NRZB1 = 1;
                } else {
                        /* 没有，设置 (离开) NRZB1 为 0 所以 NRZ1 下一时间不能为高*/
                        NRZB1 = 0;
                }
        }
}
/*
**-------------------------------------------------------------------------
**
**  概要:
**          TMH1 INTTMH1 中断服务程序
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTTMH1( )
{
        /* TODO */
        P7 ^= 0x40;                  /* P76 载波沿*/
}
```

10.6 PPG 输出音调发生使用 16 位定时器 00 (TM00)

10.6.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记, PPG 音调发生部分*/
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "timer.h"

/* 增加包括非 Applilet 产生的函数 */
#include "sw_0537.h"

/*
************************************************************************************
**  宏定义
************************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void main( void )
{
unsigned char sw_val;       /* 按键值 */
USHORT crval[2];                        /* 列用于设置定时器比较寄存器*/
USHORT cr000_value = TM_TM00_PPGCYCLE;      /* CR000 寄存器初值*/
USHORT cr010_value = TM_TM00_PPGWIDTH;      /* CR010 寄存器初值 */
```

```
USHORT temp;

        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;

        sw_init();              /* 初始化按键 */
        sw_set_debounce(10);    /* 设置去抖动计数器 10 用于 10 msec 稳定时间*/

        TMH0_Start();   /* 启动按键去抖动定时器*/

        TM00_Start();   /* 启动音调发生器*/
        /* PPG 设置 1704 us 周期, TM_TM00_PPGCYCLE (586 Hz) */
        /* PPG 设置 1136 us 脉冲宽度低电平, TM_TM00_PPGWIDTH ( 1/2 440 Hz) */
        /* 余数 (1704 - 1136) 568 us 高电平 ( 1/2 880 Hz) */
        /* 66.7/33.3 占空比 */

        while(1){
                sw_val = sw_get();              /* 获取去抖动按键值*/

                if (sw_val == SW_LD_RU) {
                        /* SW2 (左) 按下, 右 (SW3) 抬起 */
                        TM00_Stop();    /* 当改变 PPG 值停止定时器*/
                        temp = ((TM_TM00_PPGWIDTH * 2) / 3) + 1;
                        /* temp = 1/2 440 Hz * 2/3 脉冲宽度= 1/2 660 Hz 脉冲宽度*/
                        /* temp 3787 (0x0ECB), 时间是 3787 * 0.2 us = 757.4 us */
                        cr000_value = TM_TM00_PPGWIDTH + temp;
                        /* 1/2 440 (1136 us) + 1/2 660 (757.4 us) = 1893.4 us = 528 Hz */
                        cr010_value = TM_TM00_PPGWIDTH;                         /* 1/2 440 (1136 us) */
                        crval[0] = cr000_value;
                        crval[1] = cr010_value;
                        TM00_ChangeTimerCondition(crval, 2);
                        /* PPG 设置 1893.4 us 周期(528 Hz) */
                        /* PPG 设置 1136 us 脉冲宽度低电平, 1/2 440 Hz */
                        /* 余数(1893.4 - 1136) = 757.6 us 高电平, 1/2 660 Hz */
                        /* 60%/40% 占空比 */

                        TOC00 = 0x17;   /* 设置引脚初始化低*/
                        TM00_Start();   /* 重启定时器 */
                        while (SW_LU_RU != sw_get())
                                ;               /* 等待按键抬起 */
                }

                if (sw_val == SW_LU_RD) {
                        /* SW2 抬起, SW3 按下 */
                        /* SW2 (左) 按下, 右 (SW3) 抬起*/
                        TM00_Stop();    /* 当改变 PPG 值停止定时器*/
                        temp = ((TM_TM00_PPGWIDTH * 2) / 3) + 1;
                        /* temp = 1/2 440 Hz * 2/3 脉冲宽度= 1/2 660 Hz 脉冲宽度*/
                        /* temp 3787 (0x0ECB), 时间是 3787 * 0.2 us = 757.4 us */
                        cr000_value = (((TM_TM00_PPGWIDTH + 1 ) / 2) - 1) + temp;
                        /* 1/2 880 (568 us) + 1/2 660 (757.4 us) = 1325.4 us = 754 Hz */
                        cr010_value = ((TM_TM00_PPGWIDTH + 1) / 2) - 1;                 /* 1/2 880 */
                        crval[0] = cr000_value;
                        crval[1] = cr010_value;
                        TM00_ChangeTimerCondition(crval, 2);
                        /* PPG 设置 1325.4us 周期(754 Hz) */
                        /* PPG 设置 568us 脉冲宽度高电平, 1/2 880 Hz */
                        /* 余数 (1325.4 - 568) = 757.4 us 高电平, 1/2 660 Hz */
                        /* 42.8%/67.2% 占空比 */

                        TOC00 = 0x17;   /* 设置引脚初始化低*/
                        TM00_Start();   /* 重启定时器 */
                        while (SW_LU_RU != sw_get())
                                ;               /* 等待按键抬起 */
                }

        } /* 结束 while (1)循环 */
```

```
} /* 结束 main() */
```

10.6.2 Systeminit.c

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        systeminit.c
**  概要 :          此文件执行宏初始化
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
*********************************************************************************
*/
/*
*********************************************************************************
**  包含文件
*********************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "timer.h"
/*
*********************************************************************************
**  宏定义
*********************************************************************************
*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**      初始化每个模块
**
**  参数:
**      无
**
**  返回:
**      无
**-----------------------------------------------------------------------------
*/
void SystemInit( void )
{
    /* 时钟发生器初始化*/
    Clock_Init();
    /* 端口初始化*/
    PORT_Init();
    /* TM00 初始化*/
    TM00_Init();
    /* TMH0 初始化 */
    TMH0_Init();
}
/*
```

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

10.6.3 Timer.h

```
/*
************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.h
**  概要 :          此文件执行定时器模块设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************
*/

#ifndef _MDTIMER_
#define _MDTIMER_
/*
************************************************************************
**  宏定义
************************************************************************
*/
#define  REGVALUE_MAX           0xff
#define  TM_TM00_CLOCK          0x1
#define  TM_TM00_INTERVALVALUE      0x2147
#define  TM_TM00_SQUAREWIDTH        0x2147
#define  TM_TM00_PPGCYCLE   0x2147
#define  TM_TM00_PPGWIDTH   0x162f
#define  TM_TM00_ONESHOTCYCLE       0x2147
#define  TM_TM00_ONEPULSEDELAY      0x162f
#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_ONESHOTCYCLE       0x00
#define  TM_TM01_ONEPULSEDELAY      0x00
#define  TM_TM50_CLOCK          0x2
#define  TM_TM50_INTERVALVALUE      0x00
```

```
#define  TM_TM50_SQUAREWIDTH          0x00
#define  TM_TM50_PWMACTIVEVALUE       0x00
#define  TM_TM51_CLOCK          0x2
#define  TM_TM51_INTERVALVALUE        0x00
#define  TM_TM51_SQUAREWIDTH          0x00
#define  TM_TM51_PWMACTIVEVALUE       0x00
#define  TM_TMH0_CLOCK          0x4
#define  TM_TMH0_INTERVALVALUE        0x12
#define  TM_TMH0_SQUAREWIDTH          0x12
#define  TM_TMH0_PWMCYCLE    0x12
#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_CARRIERDELAY         0x00
#define  TM_TMH1_CARRIERWIDTH         0x00

/* 定时器 00 到 01,50,51,H0,H1 配置初始化*/
void TM00_Init(void);
void TMH0_Init(void);

/*定时器启动*/
void TM00_Start(void);
void TMH0_Start(void);

/*定时器停止*/
void TM00_Stop(void);
void TMH0_Stop(void);
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num);
__interrupt void MD_INTTMH0(void);

#endif              /* _MDTIMER_*/
```

10.6.4 Timer.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.c
**  概要 :          此文件执行定时器模块设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/

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

```
/*
*********************************************************************************
**  宏定义
*********************************************************************************
*/
/*TM00 脉冲宽度测量*/

/*TM01 脉冲宽度测量*/

/*
**---------------------------------------------------------------------------
**
**  概要:
**          初始化 TM00_模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void TM00_Init( )
{
        TMC00=0x00;
                                        /* 内部计数时钟*/
        PRM00 |= TM_TM00_CLOCK;
        /* TM00 PPG 输出功能*/
        ClrIORBit(CRC00,0x05);
        CR000 = TM_TM00_PPGCYCLE;
        CR010 = TM_TM00_PPGWIDTH;

        ClrIORBit(P0,0x2);                          /* TO00(P01) 输出 */
        ClrIORBit(PM0,0x2);
        TOC00 = 0x17;                               /* 初始化低 */
}
/*
**---------------------------------------------------------------------------
**
**  概要:
**          启动 TM00 计数器
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void TM00_Start()
{
        TMC00 = 0x0c;                               /* 启动 PPG 输出*/
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**          停止 TM00 计数器和清零计数寄存器。
**
**  参数:
**          无
**
**  返回:
```

```
**          无
**
**-------------------------------------------------------------------------
*/
void TM00_Stop()
{
        TMC00=0x0;
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          改变 TM00 状态。
**
**  参数:
**          USHORT* :       array_reg
**          USHORT :        array_num
**  返回:
**          MD_OK
**          MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
  switch (array_num){
      case 2:
          CR010=*(array_reg + 1);
      case 1:
          CR000=*(array_reg + 0);
          break;
      default:
          return MD_ERROR;
      }
        return MD_OK;
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          初始化 TMH0 模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TMH0_Init( void )
{
        ClrIORBit(TMHMD0, 0x80);
                        /* 计数时钟=fx/1024 */
        TMHMD0 |= (TM_TMH0_CLOCK << 4);

        SetIORBit(PR0H, 0x10);                  /* 低优先级*/
        ClrIORBit(IF0H, 0x10);
        /* TMH0 间隔定时器*/
        ClrIORBit(TMHMD0, 0x8c);
        CMP00 = TM_TMH0_INTERVALVALUE;
}

/*
**-------------------------------------------------------------------------
**
**  概要:
```

```
**          启动 TMH0 计数器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TMH0_Start( void )
{
        SetIORBit(TMHMD0, 0x80);
        ClrIORBit(MK0H, 0x10);                  /* 允许 INTTMH0 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          停止 TMH0 计数器操作。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TMH0_Stop( void )
{
        ClrIORBit(TMHMD0, 0x80);
        SetIORBit(MK0H, 0x10);                  /* 禁止 INTTMH0 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          改变 TMH0 状态。
**
**  参数:
**          UCHAR* :        array_reg
**          UCHAR :         array_num
**  返回:
**          MD_OK
**          MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num)
{
  switch (array_num){
      case 2:
            CMP10=*(array_reg + 1);
      case 1:
            CMP00=*(array_reg + 0);
            break;
      default:
            return MD_ERROR;
      }
           return MD_OK;
}
```

10.6.5 Timer_user.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer_user.c
**  概要 :          此文件执行定时器模块设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/

#pragma sfr
#pragma interrupt  INTTMH0 MD_INTTMH0
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"

/* 增加包括用于按键程序 */
#include "sw_0537.h"

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

/* 定时器 00, 定时器 01 脉冲宽度测量*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          TMH0 INTTMH0 中断服务程序
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTTMH0( )
{
        /* 调用按键 ISR 程序用于去抖动按键 */
        sw_isr();
}
```

10.7 方波音调发生使用 16 位定时器 00 (TM00)

10.7.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记, 方波音调发生部分*/
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "ad.h"
#include "port.h"
#include "timer.h"

/* 增加包括非 Applilet 产生的函数 */
#include "led_0537.h"
#include "sw_0537.h"

/*
************************************************************************************
**  宏定义
************************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void main( void )
{
unsigned char sw_val;       /* 按键值 */
USHORT value;                       /* 输入 A/D 值 */
```

```
USHORT crval[2];                    /* 列用于设置定时器比较寄存器*/
USHORT cr000_value = TM_TM00_SQUAREWIDTH;/* CR000 寄存器初值*/

        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;

        led_init();                 /* 初始化 LED */
        sw_init();          /* 初始化按键*/
        sw_set_debounce(10);    /* 设置去抖动计数器 10 用于 10 ms 稳定时间 */

        TMH0_Start();   /* 启动按键去抖动定时器*/

        TM00_Start();   /* 启动最初的发生 */
        AD_Start();                 /* 启动 A/D 转换器*/
        led_dig((UCHAR)(cr000_value >> 8));         /* 显示最初的计数器值高字节*/

        while(1){
                sw_val = sw_get();          /* 获取去抖动按键值*/

                if (sw_val == SW_LD_RU) {
                        /* SW2 (左) 按下, 右 (SW3) 抬起 */
                        while (SW_LD_RU == (sw_val = sw_get())) {
                                /* 只要 SW2 按下就执行 */
                                AD_Read(&value);                    /*获取 10 位 A/D 值*/
                                value = value >> 2;                 /*转换 8 位*/
                                led_dig((UCHAR)(value & 0xFF));     /* 显示 */
                                cr000_value = cr000_value & 0x00FF;         /*掩膜高字节*/
                                cr000_value |= (value << 8);                /* 新高字节*/
                                crval[0] = cr000_value;                             /* 投入列设置*/
                                TM00_ChangeTimerCondition(crval, 1);        /* 设置 CR000 */
                        }
                }

                if (sw_val == SW_LU_RD) {
                        /* SW2 抬起, SW3 按下 */
                        while (SW_LU_RD == (sw_val = sw_get())) {
                                /* 只要 SW3 按下 */
                                AD_Read(&value);                    /* 获取 10 位 A/D 值 */
                                value = value >> 2;                 /* 转换 8 位 */
                                led_dig((UCHAR)(value & 0xFF));     /* 显示 */
                                cr000_value = cr000_value & 0xFF00;         /* 掩膜低字节 */
                                cr000_value |= value & 0xFF;                /* 新低字节*/
                                crval[0] = cr000_value;                             /* 投入列设置*/
                                TM00_ChangeTimerCondition(crval, 1);        /* 设置 CR000 */
                        }
                }

                if (sw_val == SW_LD_RD) {
                        /* SW2 和 SW3 同时按下- 无动作 */
                }
        } /* 结束 while (1) 循环 */
} /* 结束 main() */
```

10.7.2 Systeminit.c

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
```

```
** 若使用此程序用户必须自己承担责任。
** NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
** 文件名 :        systeminit.c
** 概要 :          此文件执行宏初始化。
** APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
** 设备 :          uPD78F0537
**
** 编译器 :        NEC/CC78K0
**
*****************************************************************************
*/
/*
*****************************************************************************
** 包含文件
*****************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "ad.h"
#include "port.h"
#include "timer.h"
/*
*****************************************************************************
** 宏定义
*****************************************************************************
*/

/*
**-------------------------------------------------------------------------
**
** 概要:
**         启动每个模块
**
** 参数:
**         无
**
** 返回:
**         无
**
**-------------------------------------------------------------------------
*/
void SystemInit( void )
{
        /* 时钟发生器初始化*/
        Clock_Init();
        /* 端口初始化*/
        PORT_Init();
        /* AD 初始化*/
        AD_Init();
        /* TM00 初始化*/
        TM00_Init();
        /* TMH0 初始化*/
        TMH0_Init();
}

/*
**-------------------------------------------------------------------------
**
** 概要:
**         初始化硬件设置
**
** 参数:
**         无
**
** 返回:
**         无
```

```
**
**---------------------------------------------------------------------------
*/
void hdwinit( void )
{
        DI( );
        SystemInit( );
        EI( );
}
```

10.7.3 Timer.h

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.h
**  概要 :          此文件执行定时器模块设备驱动程序。
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/

#ifndef _MDTIMER_
#define _MDTIMER_
/*
************************************************************************************
**  宏定义
************************************************************************************
*/
#define  REGVALUE_MAX           0xff
#define  TM_TM00_CLOCK          0x1
#define  TM_TM00_INTERVALVALUE      0x162f
#define  TM_TM00_SQUAREWIDTH        0x162f
#define  TM_TM00_PPGCYCLE   0x162f
#define  TM_TM00_PPGWIDTH   0x00
#define  TM_TM00_ONESHOTCYCLE       0x162f
#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_ONESHOTCYCLE       0x00
#define  TM_TM01_ONEPULSEDELAY      0x00
#define  TM_TM50_CLOCK          0x2
#define  TM_TM50_INTERVALVALUE      0x00
#define  TM_TM50_SQUAREWIDTH        0x00
#define  TM_TM50_PWMACTIVEVALUE     0x00
#define  TM_TM51_CLOCK          0x2
#define  TM_TM51_INTERVALVALUE      0x00
#define  TM_TM51_SQUAREWIDTH        0x00
#define  TM_TM51_PWMACTIVEVALUE     0x00
#define  TM_TMH0_CLOCK          0x4
#define  TM_TMH0_INTERVALVALUE      0x7a0
#define  TM_TMH0_SQUAREWIDTH        0x7a0
```

```
#define  TM_TMH0_PWMCYCLE   0x7a0
#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_CARRIERDELAY       0x00
#define  TM_TMH1_CARRIERWIDTH       0x00


/* 定时器 00 到 01,50,51,H0,H1 配置初始化 */
void TM00_Init(void);
void TMH0_Init(void);

/*定时器启动*/
void TM00_Start(void);
void TMH0_Start(void);

/*定时器停止*/
void TM00_Stop(void);
void TMH0_Stop(void);
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num);
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num);
__interrupt void MD_INTTMH0(void);

#endif              /* _MDTIMER_*/
```

10.7.4 Timer.c

```
/*
************************************************************************************
**
**  此设备驱动文件适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  由 NEC Electronics Corporation 版权所有.
**
**  若使用此程序必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer.c
**  概要 :          此文件执行定时器模块设备驱动程序。
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/

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


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

```
/*TM00 脉冲宽度测量*/

/*TM01 脉冲宽度测量*/

/*
**---------------------------------------------------------------------------
**
**  概要:
**          初始化 TM00_模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void TM00_Init( )
{
        TMC00=0x00;
                                        /* 内部计数时钟*/
        PRM00 |= TM_TM00_CLOCK;
        /* TM00 方波输出*/
        ClrIORBit(CRC00,0x01);
        CR000 = TM_TM00_SQUAREWIDTH;
        CR010 = 0xffff;

        ClrIORBit(P0,0x2);                      /* TO00(P01) 输出 */
        ClrIORBit(PM0,0x2);
        TOC00=0x7;                              /* 初始化低*/
}
/*
**---------------------------------------------------------------------------
**
**  概要:
**          启动 TM00 计数器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void TM00_Start()
{
        TMC00 = 0x0c;                           /* 启动方波输出*/
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**          停止 TM00 计数器和清零计数寄存器。
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void TM00_Stop()
{
        TMC00=0x0;
```

```
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**          改变 TM00 状态。
**
**  参数:
**          USHORT* :       array_reg
**          USHORT :        array_num
**  返回:
**          MD_OK
**          MD_ERROR
**
**---------------------------------------------------------------------------
*/
MD_STATUS TM00_ChangeTimerCondition(USHORT* array_reg,USHORT array_num)
{
  switch (array_num){
        case 2:
              CR010=*(array_reg + 1);
        case 1:
              CR000=*(array_reg + 0);
              break;
        default:
              return MD_ERROR;
        }
             return MD_OK;
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**          初始化 TMH0 模块。
**
**  参数:
**          无
**
**  返回:
**          无
**
**---------------------------------------------------------------------------
*/
void TMH0_Init( void )
{
          ClrIORBit(TMHMD0, 0x80);
                              /* 计数时钟=fx/1024 */
          TMHMD0 |= (TM_TMH0_CLOCK << 4);

          SetIORBit(PR0H, 0x10);                          /* 低优先级*/
          ClrIORBit(IF0H, 0x10);
          /* TMH0 间隔定时器*/
          ClrIORBit(TMHMD0, 0x8c);
          CMP00 = TM_TMH0_INTERVALVALUE;
}

/*
**---------------------------------------------------------------------------
**
**  概要:
**          启动 TMH0 计数器。
**
**  参数:
**          无
**
**  返回:
```

```
**          无
**
**-------------------------------------------------------------------------
*/
void TMH0_Start( void )
{
        SetIORBit(TMHMD0, 0x80);
        ClrIORBit(MK0H, 0x10);                  /* 允许 INTTMH0 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          停止 TMH0 计数器操作。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
void TMH0_Stop( void )
{
        ClrIORBit(TMHMD0, 0x80);
        SetIORBit(MK0H, 0x10);                  /* 禁止 INTTMH0 */
}

/*
**-------------------------------------------------------------------------
**
**  概要:
**          改变 TMH0 状态。
**
**  参数:
**          UCHAR* :        array_reg
**          UCHAR :         array_num
**  返回:
**          MD_OK
**          MD_ERROR
**
**-------------------------------------------------------------------------
*/
MD_STATUS TMH0_ChangeTimerCondition(UCHAR* array_reg,UCHAR array_num)
{
  switch (array_num){
      case 2:
          CMP10=*(array_reg + 1);
      case 1:
          CMP00=*(array_reg + 0);
          break;
      default:
          return MD_ERROR;
      }
        return MD_OK;
}
```

10.7.5 Timer_user.c

```
/*
*****************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
```

```
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        timer_user.c
**  概要 :          此文件执行定时器模块的设备驱动程序
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
*******************************************************************************
*/

#pragma sfr
#pragma interrupt  INTTMH0 MD_INTTMH0
/*
*******************************************************************************
**  包含文件
*******************************************************************************
*/
#include "macrodriver.h"
#include "timer.h"

/* 增加包括用于按键的程序*/
#include "sw_0537.h"

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

/* 定时器 00, 定时器 01 脉冲宽度测量*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**      TMH0 INTTMH0 中断服务程序
**
**  参数:
**      无
**
**  返回:
**      无
**
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTTMH0( )
{
    /* 调用按键 ISR 程序用于按键去抖动*/
    sw_isr();
}
```

10.8 钟表定时器(WTM)的时间保持

10.8.1 Main.c

```
/*  main.c 用于 NEC 定时器应用笔记, 钟表定时器部分*/
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        main.c
**  概要 :          此文件执行主函数
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器:         NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "watchtimer.h"

/* 增加包括非 Applilet 产生的函数*/
#include "sw_0537.h"        /* 用于 M-Station 的按键功能*/
#include "led_0537.h"       /* 用于 M-Station 的 LED 功能*/

/*
************************************************************************************
**  宏定义
************************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          主函数
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void main( void )
{
unsigned char sw_val;
unsigned char last_sec;
```

```
        IMS = MEMORY_IMS_SET;
        IXS = MEMORY_IXS_SET;
        /* TODO. 增加用户代码*/

        led_init();                     /* 初始化 LED */
        sw_init();          /* 初始化按键 */
        sw_set_debounce(10);    /* 设置去抖动计数器 10 (10 x 1 ms) */

        Seconds = 0;    /* 初始化秒计数器*/
        SecHalf = 0;
        SecondTimer = ST_ON;
        led_dig_bcd(Seconds);   /* 显示秒计数*/
        last_sec = 0;

        WT_Start();                     /* 启动钟表定时器*/

        while(1){
                /* 新秒? */
                if (Seconds != last_sec) {
                        led_dig_bcd(Seconds);   /* 显示新秒*/
                        last_sec = Seconds;
                }

                /* 检查按键 */
                sw_val = sw_get();          /* 按键按下检查*/
                switch (sw_val) {
                case SW_LD_RU:
                        Seconds = 0;    /* 秒清零*/
                        SecHalf = 0;
                        led_dig_bcd(Seconds);   /* 显示 */
                        last_sec = 0;
                        while ( (sw_val = sw_get()) != SW_LU_RU)
                                ;           /* 等待按键再次抬起*/
                        break;
                case SW_LU_RD:
                        /* 按下 SW3, 停止或启动秒定时器*/
                        if (SecondTimer == ST_OFF)
                                SecondTimer = ST_ON;
                        else
                                SecondTimer = ST_OFF;
                        while ( (sw_val = sw_get()) != SW_LU_RU)
                                ;           /* 等待按键再次抬起*/
                        break;
                default:
                        /* 同时抬起同时按下-忽略 */
                        break;
                }
        }
}
```

10.8.2 Systeminit.c

```
/*
***********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        systeminit.c
**  概要 :          此文件执行宏初始化。
```

```
**  APIlib :          NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :            uPD78F0537
**
**  编译器 :          NEC/CC78K0
**
*******************************************************************************
*/
/*
*******************************************************************************
**  包含文件
*******************************************************************************
*/
#include "macrodriver.h"
#include "system.h"
#include "port.h"
#include "watchtimer.h"
/*
*******************************************************************************
**  宏定义
*******************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          初始化每个宏
**
**  参数:
**          无
**
**  返回:
**          无
**-----------------------------------------------------------------------------
*/
void SystemInit( void )
{
        /* 时钟发生器初始化*/
        Clock_Init();
        /* 端口初始化 */
        PORT_Init();
        /* WT初始化 */
        WT_Init();
}
/*
**-----------------------------------------------------------------------------
**
**  概要:
**          初始化硬件设置
**
**  参数:
**          无
**
**  返回:
**          无
**-----------------------------------------------------------------------------
*/
void hdwinit( void )
{
        DI( );
        SystemInit( );
        EI( );
}
```

10.8.3 Watchtimer.h

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

```
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        watchtimer.h
**  概要 :          此文件执行钟表定时器模块设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
*******************************************************************************
*/
#ifndef _MDWATCHTIMER_
#define _MDWATCHTIMER_
/*
*******************************************************************************
**  宏定义
*******************************************************************************
*/
void WT_Init( void );
MD_STATUS WT_Start( void );
MD_STATUS  WT_Stop( void );
void WT_User_Init( void );
__interrupt void MD_INTWT( void );
__interrupt void MD_INTWTI( void );

/* 增加秒计数全局变量*/
extern UCHAR SecHalf;               /* 半秒计数器*/
extern UCHAR Seconds;               /* 秒计数*/
extern UCHAR SecondTimer;           /* 定时器开/关控制 */
/* 增加 SecondTimer 定义 */
#define ST_ON    1
#define ST_OFF  0

#endif
```

10.8.4 Watchtimer.c

```
/*
*******************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        watchtimer.c
**  概要 :          此文件执行钟表定时器模块设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
```

```
*****************************************************************************
*/
/*
*****************************************************************************
** 包含文件
*****************************************************************************
*/
#include "macrodriver.h"
#include "watchtimer.h"
/*
*****************************************************************************
** 宏定义
*****************************************************************************
*/
/*
**-------------------------------------------------------------------------
**
** 概要:
**      初始化钟表定时器模块。
**
** 参数:
**      无
**
** 返回:
**      无
**-------------------------------------------------------------------------
*/
void WT_Init( void )
{
        WTM = 0;
        WTPR = 1;               /*低优先级*/
        WTMK = 0;
        WTIPR = 1;              /* 低优先级*/
        WTIMK = 0;
        WTM.7 = 1;              /* 钟表定时器时钟: fw = fsub */
        ClrIORBit(WTM, 0x0c);   /* 钟表时间: 2^14/fw (在 32.768Kz 的 0.5s) */
        ClrIORBit(WTM, 0x70);   /* 间隔时间: 2^5/fw */
        SetIORBit(WTM, 0x10);
        WT_User_Init( );
}
/*
**-------------------------------------------------------------------------
**
** 概要:
**      此功能在停止后重启钟表定时器。
**
** 参数:
**      无
**
** 返回:
**      MD_OK
**-------------------------------------------------------------------------
*/
MD_STATUS WT_Start( void )
{
        /* 允许钟表定时器操作*/
        WTM1 = 1;
        /* 启动 5 位计数器 */
        WTM0 = 1;
        WTMK = 0;               /* 允许 INTWT */
        WTIMK = 0;              /* 允许 INTWTI */

        return MD_OK;
}
/*
**-------------------------------------------------------------------------
**
** 概要:
```

```
**          此功能停止钟表定时器。
**
**  参数:
**          无
**
**  返回:
**          MD_OK
**--------------------------------------------------------------------------------
*/
MD_STATUS WT_Stop( void )
{
        WTMK = 1;                   /* 禁止 INTWT */
        WTIMK = 1;                  /* 禁止 INTWTI */
        /* 停止 5 位计数器*/
        WTM1 = 0;
        /* 停止钟表定时器操作*/
        WTM0 = 0;

        return MD_OK;
}
```

10.8.5 Watchtimer_user.c

```
/*
********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        watchtimer_user.c
**  概要 :          此文件执行钟表定时器模块设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
********************************************************************************
*/

#pragma interrupt INTWT MD_INTWT
#pragma interrupt INTWTI MD_INTWTI
/*
********************************************************************************
**  包含文件
********************************************************************************
*/
#include "macrodriver.h"
#include "watchtimer.h"

/* added include to find definition for sw_isr() and LED_PAT_DP */
#include "sw_0537.h"
#include "led_0537.h"

/*
********************************************************************************
**  宏定义
********************************************************************************
*/
/* 增加全局变量*/
```

```
UCHAR SecHalf;              /* 半秒计数器*/
UCHAR Seconds;                      /* 秒计数*/
UCHAR SecondTimer;      /* 控制定时器开/关*/

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          This function is an empty function for user code when initializing.
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
void WT_User_Init( void )
{
        /* TODO. 增加用户代码*/
}

/*
**-----------------------------------------------------------------------------
**
**  概要:
**          INTWTI 中断服务程序。
**
**  参数:
**          无
**
**  返回:
**          无
**
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTWT( void )
{
        /* TODO. 增加用户定义中断服务程序*/
        /* 每 0.5 秒产生一次中断*/
        if (SecondTimer == ST_OFF)
                return;    /* 定时器停止*/

        SecHalf++;                              /* 累加半秒计数器*/
        if (SecHalf == 1) {
                P7 &= LED_PAT_DP;       /* 启动小数点显示半秒*/
        }

        if (SecHalf > 1) {
                /* 2 个半秒发生*/
                SecHalf = 0;                    /* 复位半秒计数器 */
                Seconds++;                              /* 累加秒 */
                if (Seconds == 60) {
                        Seconds = 0;       /* 60 秒循环*/
                }
        }
}

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

```
**
**  返回:
**          无
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTWTI( )
{
        /* TODO. 增加用户定义中断服务程序*/
        /* 每毫秒产生一次中断*/
        sw_isr(); /* 去抖动按键 */
}
```

10.9 所有演示程序通用文件列表

本节文件列表可被所有演示程序使用。一些演示程序仅使用部分文件。

10.9.1 Macrodriver.h

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        macrodriver.h
**  概要 :          常规头文件
**  APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  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    /* 发送数据完成*/
#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     /* 无可用资源*/
#define MD_PARITYERROR          MD_ERRORBASE+0x2     /* UARTn 奇偶校验错误*/
#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     /* 错误论点输入错误*/
#define MD_TIMINGERROR          MD_ERRORBASE+0x6     /* 错误时序操作错误*/
#define MD_SETPROHIBITED        MD_ERRORBASE+0x7     /* 禁止设置*/
#define MD_DATAEXISTS           MD_ERRORBASE+0x8     /* 数据转送给下一 TXBn 寄存器 */
#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 从设备发送结束*/
#define MD_SLAVE_RCV_END        MD_STATUSBASE+0x11   /*IIC 从设备接收结束*/
#define MD_MASTER_SEND_END      MD_STATUSBASE+0x12   /*IIC 主设备发送结束*/
#define MD_MASTER_RCV_END       MD_STATUSBASE+0x13  /*IIC 主设备接收结束*/

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

/* IMS 和 IXS 的值 */
#define  MEMORY_IMS_SET                 0xCC
#define  MEMORY_IXS_SET                 0x00
/* 清零 IO 寄存器位和设置 IO 寄存器位 */
#define ClrIORBit(Reg, ClrBitMap)   Reg &= ~ClrBitMap
#define SetIORBit(Reg, SetBitMap) Reg |= SetBitMap

enum INTLevel { Highest, Lowest };


#define  SYSTEMCLOCK  20000000
#define  SUBCLOCK     32768
#define  MAINCLOCK    20000000
#define  FRCLOCK      8000000
#define  FRCLOCKLOW   240000


#endif
```

10.9.2 System.h

```
/*
*******************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        system.h
**  概要 :          此文件执行系统模块的设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  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
#define  CG_Mainosc       0x14

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
```

10.9.3 System.c

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        system.c
**  概要 :          此文件执行系统模块的设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  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:
```

```
**         无
**-------------------------------------------------------------------------
*/
void Clock_Init( void )
{
        USHORT i;
        UCHAR temp_stabset, temp_stabwait;
        SetIORBit(PM12, 0x06);                  /* P121/122 输入模式*/
        ClrIORBit(OSCCTL, 0x80);        /* X1/X2 输入模式*/
        SetIORBit(OSCCTL, 0x40);
        ClrIORBit(MOC, 0x80);
        /* OSC 稳定时间*/
        temp_stabset = CG_X1STAB_STA;
        do{
                temp_stabwait = OSTC;
                temp_stabwait &= temp_stabset;
        }while(temp_stabwait != temp_stabset);
        OSTS = CG_X1STAB_SEL;
        for(i = 0; i <= 20; i++){               /* 等待 5us */
                NOP();
        }
        SetIORBit(OSCCTL, 0x01);                /* 10MHz<fx<=20MHz */
        SetIORBit(MCM, 0x05);                   /* X1 用于 CPU 操作 */
        SetIORBit(PM12, 0x18);                  /* P123/124 输入模式*/
        ClrIORBit(OSCCTL, 0x20);        /* XT1 输入模式*/
        SetIORBit(OSCCTL, 0x10);
        ClrIORBit(RCM, 0x01);
        PCC = CG_CPU_CLOCKSEL;
}
```

10.9.4 Port.h

```
/*
*****************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        port.h
**  概要 :          此文件执行端口模块设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
*****************************************************************************
*/
#ifndef   _MDPORT_
#define   _MDPORT_
/*
*****************************************************************************
**  宏定义
*****************************************************************************
*/
#define   PORT_PM0      0xff
#define   PORT_PU0      0x0
#define   PORT_P0       0x0
#define   PORT_PM1      0xff
#define   PORT_PU1      0x0
#define   PORT_P1       0x0
#define   PORT_PM2      0xff
```

```
#define  PORT_P2          0x0
#define  PORT_PM3         0xff
#define  PORT_PU3         0x6
#define  PORT_P3          0x0
#define  PORT_PM4         0xf0
#define  PORT_PU4         0x0
#define  PORT_P4          0x0
#define  PORT_PM5         0xf0
#define  PORT_PU5         0x0
#define  PORT_P5          0x0
#define  PORT_PM6         0xff
#define  PORT_P6          0x0
#define  PORT_PM7         0x0
#define  PORT_PU7         0x0
#define  PORT_P7          0x0
#define  PORT_PM12        0xff
#define  PORT_PU12        0x0
#define  PORT_P12         0x0
#define  PORT_P13         0x0
#define  PORT_PM14        0xff
#define  PORT_PU14        0x0
#define  PORT_P14         0x0
#define  PORT_ADPC        0x0

void PORT_Init( void );

#endif
```

10.9.5 Port.c

```
/*
************************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        port.c
**  概要 :          此文件执行端口模块设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
************************************************************************************
*/
/*
************************************************************************************
**  包含文件
************************************************************************************
*/
#include "macrodriver.h"
#include "port.h"
/*
************************************************************************************
**  常数
************************************************************************************
*/
/*
**-----------------------------------------------------------------------------
**
**  概要:
```

```
**          初始化 I/O 模块。
**
**  参数:
**          无
**
**  返回:
**          无
**--------------------------------------------------------------------------------
*/
void PORT_Init( void )
{
        /*初始化端口寄存器*/
        P0 = PORT_P0;
        P1 = PORT_P1;
        P2 = PORT_P2;
        P3 = PORT_P3;
        P4 = PORT_P4;
        P5 = PORT_P5;
        P6 = PORT_P6;
        P7 = PORT_P7;
        P12 = PORT_P12;
        P13 = PORT_P13;
        P14 = PORT_P14;

        /* 初始化上拉电阻选项寄存器*/
        PU0 = PORT_PU0;
        PU1 = PORT_PU1;
        PU3 = PORT_PU3;
        PU4 = PORT_PU4;
        PU5 = PORT_PU5;
        PU7 = PORT_PU7;
        PU12 = PORT_PU12;
        PU14 = PORT_PU14;

        /* 初始化模式寄存器*/
        PM0 = PORT_PM0;
        PM1 = PORT_PM1;
        PM2 = PORT_PM2;
        ADPC = PORT_ADPC;
        PM3 = PORT_PM3;
        PM4 = PORT_PM4;
        PM5 = PORT_PM5;
        PM6 = PORT_PM6;
        PM7 = PORT_PM7;
        PM12 = PORT_PM12;
        PM14 = PORT_PM14;
}
```

10.9.6 Ad.h

```
/*
*********************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        ad.h
**  概要 :          此文件执行用于 AD 模块的设备驱动程序。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
```

```
** 编译器 :        NEC/CC78K0
**
*****************************************************************************
*/
#ifndef  _MDAD_
#define  _MDAD_
/*
*****************************************************************************
** 宏定义
*****************************************************************************
*/
#define  AD_ADPC         0x0
#define  AD_ADS          0x0
#define  AD_PM2          0xff
#define  AD_ADM          0x38

enum INTMode { INTMode0, INTMode1, INTMode2 };

void AD_Init( void );
MD_STATUS AD_Start( void );
MD_STATUS AD_Read( USHORT* buffer );
MD_STATUS AD_Stop( void );
MD_STATUS AD_Stop( void );

#endif
```

10.9.7 Ad.c

```
/*
*****************************************************************************
**
** 此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
** 78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
** 版权(C) NEC Electronics Corporation 2002 - 2005
** 归 NEC Electronics Corporation 版权所有。
**
** 若使用此程序用户必须自己承担责任。
** NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
** 文件名 :        ad.c
** 概要 :          此文件用来执行设备 AD 模块驱动。。
** APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
** 设备 :          uPD78F0537
**
** 编译器 :        NEC/CC78K0
**
*****************************************************************************
*/
/*
*****************************************************************************
** 包含文件
*****************************************************************************
*/
#include "macrodriver.h"
#include "ad.h"
/*
**-----------------------------------------------------------------------------
**
** 概要:
**      此功能初始化 A/D 模块。
**
** 参数:
**      无
**
** 返回:
**      无
```

```
**-----------------------------------------------------------------------------
*/
void AD_Init( void )
{
        ADPC = AD_ADPC;
        PM2 = PM2 | AD_PM2;
        ADS = AD_ADS;
        ClrIORBit(ADM, 0x02);                   /* 2.7V<=AVref<=5.5V */
        ADM = ADM & AD_ADM;
}
/*
**-----------------------------------------------------------------------------
**
**  概要:
**      此功能启动 A/D 转换器。
**
**  参数:
**      无
**
**  返回:
**      MD_OK
**-----------------------------------------------------------------------------
*/
MD_STATUS AD_Start( void )
{
        int delay = 200;
        SetIORBit(ADM, 0x01);                   /* 操作比较器 */
        while(delay--);

        SetIORBit(ADM, 0x80);
        return MD_OK;
}
/*
**-----------------------------------------------------------------------------
**
**  概要:
**      此功能在 A/D 转换完成后调用,
**      并返回缓冲器中的转换结果。
**
**  参数:
**      写入转换结果的地址缓冲器。
**
**  返回:
**      MD_OK
**
**-----------------------------------------------------------------------------
*/
MD_STATUS AD_Read( USHORT* buffer )
{
        *buffer = (USHORT)( ADCR >> 6 );
        return MD_OK;
}
/*
**-----------------------------------------------------------------------------
**
**  概要:
**      此功能停止 A/D 转换器。
**
**  参数:
**      无
**
**  返回:
**      MD_OK
**-----------------------------------------------------------------------------
*/
MD_STATUS AD_Stop( void )
{

        ClrIORBit(ADM, 0x81);
```

```
    return MD_OK;
}
```

10.9.8 Ad_user.c

```
/*
*******************************************************************************
**
**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
**
**  版权(C) NEC Electronics Corporation 2002 - 2005
**  归 NEC Electronics Corporation 版权所有。
**
**  若使用此程序用户必须自己承担责任。
**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
**
**  文件名 :        ad_user.c
**  概要 :          此文件执行设备 AD 模块驱动。
**  APIlib :        NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
**
**  设备 :          uPD78F0537
**
**  编译器 :        NEC/CC78K0
**
*******************************************************************************
*/
/*
*******************************************************************************
**  包含文件
*******************************************************************************
*/
#include "macrodriver.h"
#include "ad.h"
```

10.9.9 Led_0537.h

```
/* led_0537.h                                                                    */
/*          用于 M-78F0537 CPU 板 LED 数字显示头文件          */
/* 版本 1.1         01-13-2006                                                   */

#ifndef _LED_0537_H
#define _LED_0537_H

/*****************************************************************/
/* 定义                                    */
/*****************************************************************/

/* LED 图形十进制，十六进制数 */
/*  位,     ---A---    */
/*   0=on 1=off                                  |     |    */
/* 位 0 = segment A                  F      B  */
/* 位 1 = segment B                  |      |    */
/* 位 2 = segment C                   ---G---   */
/* 位 3 = segment D                  |      |    */
/* 位 4 = segment E                  E      C  */
/* 位 5 = segment F                  |      |    */
/* 位 6 = segment G                   ---D--- DP          */
/* 位 7 = 小数点                                 */

#define LED_PAT_0   0xC0
#define LED_PAT_1   0xF9
#define LED_PAT_2   0xA4
#define LED_PAT_3   0xB0
#define LED_PAT_4   0x99
#define LED_PAT_5   0x92
```

```
#define LED_PAT_6   0x82
#define LED_PAT_7   0xF8
#define LED_PAT_8   0x80
#define LED_PAT_9   0x98
#define LED_PAT_A   0x88
#define LED_PAT_B   0x83
#define LED_PAT_C   0xC6
#define LED_PAT_D   0xA1
#define LED_PAT_E   0x86
#define LED_PAT_F   0x8E
#define LED_PAT_BLANK 0xFF
#define LED_PAT_DP              0x7F
#define LED_PAT_DASH  0xBF
#define LED_PAT_ULINE  0xF7
#define LED_PAT_OLINE  0xFE
#define LED_PAT_EQUAL0xB7

/*********************************************************************/
/* 输出功能                                          */
/*********************************************************************/
extern void led_init(void);                                         /* 初始化 LED 输出端口*/
extern void led_out_right(unsigned char val); /* 在右 LED 输出值          */
extern void led_out_left(unsigned char val);   /* 在左 LED 输出值                  */
extern void led_dig_right(unsigned char num);          /* 在右 LED 显示数          */
extern void led_dig_left(unsigned char num); /* 在左 LED 显示数          */
extern void led_dig(unsigned char num);                         /* 显示十六进制数            */
extern void led_dig_bcd(unsigned char bcdnum);       /* 显示 BCD 数字             */

#endif    /* _LED_0537_H */
```

10.9.10 Led_0537.c

```
/*        led_0537.c - LED 显示程序                                  */
/*                用于 M-78F0537 CPU 板 M-Station 基板       */
/*  版本: 1.1       01-13-2006                                        */

/*                P70-P77 = 右数字 (LED2)输出                        */
/*                P40-P43 = 左数字(LED1)位 0-3       输出*/
/*                P50-P53 = 左数字(LED1)位 4-7 输出          */

/*        连接端口到 M-Station 1.1 上的 LED, 按以下连接 ROW1 和 ROW2.
          连接端口到 M-Station 2 上的 LED, 确保定义 SBxx 。
          端口      LED             M-Station 1.1      M-Station 2.2
          ----      ---             -------------      -------------
          P70             2-A             R1.25 - R2.25     SB27
          P71             2-B             R1.26 - R2.26     SB28
          P72             2-C             R1.27 - R2.27     SB29
          P73             2-D             R1.28 - R2.28     SB30
          P74             2-E             R1.29 - R2.29     SB31
          P75             2-F             R1.30 - R2.30     SB32
          P76             2-G             R1.31 - R2.31     SB33
          P77             2-DP    R1.32 - R2.32     SB34

          P40             1-A             R1.17 - R2.17     SB35
          P41             1-B             R1.18 - R2.18     SB36
          P42             1-C             R1.19 - R2.19     SB37
          P43             1-D             R1.20 - R2.20     SB38

          P50             1-E             R1.21 - R2.21     SB39
          P51             1-F             R1.22 - R2.22     SB40
          P52             1-G             R1.23 - R2.23     SB41
          P53             1-DP    R1.24 - R2.24     SB42
*/

/* 注: 在 M-Station Base V1.0 原型, P40-P53 */
/*  定位 ROW4.1-8, 并需要预先       */
/*  连接 ROW2.17-24 驱动 LED1.                                      */
```

```
/* 需要程序声明 SFR's       */
#pragma sfr

#include "led_0537.h"

/* 7 段数字位表 */
static unsigned char dig_tab[] = {
  LED_PAT_0,    /* 0 */
  LED_PAT_1,    /* 1 */
  LED_PAT_2,    /* 2 */
  LED_PAT_3,    /* 3 */
  LED_PAT_4,    /* 4 */
  LED_PAT_5,    /* 5 */
  LED_PAT_6,    /* 6 */
  LED_PAT_7,    /* 7 */
  LED_PAT_8,    /* 8 */
  LED_PAT_9,    /* 9 */
  LED_PAT_A,    /* A */
  LED_PAT_B,    /* B */
  LED_PAT_C,    /* C */
  LED_PAT_D,    /* D */
  LED_PAT_E,    /* E */
  LED_PAT_F     /* F */
};

/*        void led_init(void) */
/*                设定端口为 LED 数字显示*/
void led_init(void)
{
#if 0       /*由 Applilet 进行端口初始化在 Port_Init()*/
        PM7 = 0x00;             /* 设置所有端口 7 为输出 */
        PM4 = 0x00;             /* 设置所有端口 4 为输出*/
        PM5 = 0x00;             /* 设置所有端口 5 为输出*/
#endif
}

/* void led_out_right(unsigned char val) */
/*                在右 LED 输出原始数据*/
void led_out_right(unsigned char val)
{
        P7 = val;
}

/* void led_out_left(unsigned char val) */
/*                在左 LED 输出原始数据*/
void led_out_left(unsigned char val)
{
        P4 = val & 0x0F;
        P5 = (val >> 4) & 0x0F;
}

/* void led_dig_right(unsigned char num) */
/*        在右 LED 显示数字*/
void led_dig_right(unsigned char num)
{
        if (num > 0x0F) {
                led_out_right(LED_PAT_BLANK);
                return;
        }
        led_out_right(dig_tab[num]);
}

/* void led_dig_left(unsigned char num) */
/*        在左 LED 显示数字 */
void led_dig_left(unsigned char num)
{
        if (num > 0x0F) {
                led_out_left(LED_PAT_BLANK);
                return;
        }
```

```
        led_out_left(dig_tab[num]);
}

/* void led_dig(unsigned char num) */
/*              显示十六进制数字*/
/*              num – 数字显示 */
/*                      位 0-3 右数字*/
/*                      位 4-7 左数字*/
void led_dig(unsigned char num)
{
        led_out_right(dig_tab[num & 0x0F]);
        led_out_left(dig_tab[(num >> 4) & 0x0F]);
}

/* void led_dig_bcd(unsigned char bcdnum) */
/*              显示两位 BCD 代码 bcdnum */
/*              bcdnum - number 按 BCD 显示 */
/*                      0 - 9    显示右面十进制数字，左空白*/
/*                      10 - 99  显示两位小数*/
/*                      100 - 255 显示空白 */
void led_dig_bcd(unsigned char bcdnum)
{
unsigned char tens_dig;

        if (bcdnum > 99) {
                led_out_right(LED_PAT_BLANK);   /* 显示两个数字空白*/
                led_out_left(LED_PAT_BLANK);
                return;
        }

        if (bcdnum < 10) {
                led_out_right(dig_tab[bcdnum]);     /* 显示右 LED */
                led_out_left(LED_PAT_BLANK);                /* 左 LED 空白*/
                return;
        }

        /* 10 <= bcdnum <= 99 */
        tens_dig = 0;
        do {                                    /* 计算十的位置和余数 */
                bcdnum -= 10;     /* 按 10 倍数减 */
                tens_dig++;                     /* 当计数按十位数加 */
        } while (bcdnum >= 10);
        /* tens_dig 十的位置 */
        /* bcdnum 余数*/
        led_out_right(dig_tab[bcdnum]);
        led_out_left(dig_tab[tens_dig]);
}
```

10.9.11 Sw_0537.h

```
/* sw_0537.h       */
/*          M-78F0537 CPU 板头文件用于按键读取基础板*/
/*  版本: 1.1       01-13-2006          */

#ifndef _SW_0537_H
#define _SW_0537_H

/*****************************************************************/
/* 定义                                  */
/*****************************************************************/

/* 按键输入符号定义*/
/* SW2 = left switch = P31 */
/* SW3 = right switch = P32 */
/*                                                                                                 P32
                P31 */
#define  SW_LU_RU       0x06     /* left up, right up  1                  1         */
#define SW_LD_RU        0x04     /* left down, right up          1                  0          */                                  211
```

```
#define SW_LU_RD        0x02    /* left up, right down          0                   1          */
#define SW_LD_RD        0x00    /* left down, right down 0              0          */

#define  SW_DEF_DEB_COUNT   8          /* 定义去抖动计数器          */

/*********************************************************************/
/* 输出功能                                                          */
/*********************************************************************/
extern void sw_init(void);                          /* 初始化端口和按键输入变量*/
extern unsigned char sw_chk(void); /* 获取未去抖动按键输入*/
extern unsigned char sw_get(void); /* 获取去抖动按键输入*/
extern void sw_set_debounce(unsigned char count);   /* 设置去抖动计数*/
extern void sw_isr(void);                           /* 由定时器 ISR 调用去抖动程序*/

#endif    /* _SW_0537_H */
```

10.9.12 Sw_0537.c

```
/*          sw_0537.c –按键输入程序                                 */
/*                    用于 M-78F0537 CPU 板 M-Station 板          */
/*  版本: 1.1         01-13-2006                                                */
/*                    P31 = 输入左按键  (SW2)                                 */
/*                    P32 = 输入右按键  (SW3)                                 */
/*          连接端口到 M-Station 1.1 上的按键, 按以下连接 ROW1 和 ROW2。
            连接端口到 M-Station 2 上的按键, 确保按 SBxx 定义连接。

            端口      按键      M-Station 1.1       M-Station 2.2
            ----      ---               -------------       -------------
            P31               SW2               R1.5 - R2.5               SB7
            P32               SW3               R1.6 - R2.6               SB8
*/
/* 需要程序声明 SFR's       */
#pragma sfr

#include "sw_0537.h"

/*按键处理变量*/
static unsigned char sw_last;                       /* 最后去抖动按键值*/
static unsigned char sw_new;                        /*去抖动新值*/
static unsigned char sw_deb_value;/*去抖动计数器值*/
static unsigned char sw_deb_count;/* 去抖动计数器 */

/*          void sw_init(void) */
/*                    设置端口为转换输入 */
void sw_init(void)
{
#if 0       /*由 Applilet 初始化 Port_Init()*/
            /* 设置 P31 和 P32 为输入*/
            PM3.1 = 1;
            PM3.2 = 1;
            /* 设置 P31 和 P32 上拉 */
            PU3.1 = 1;
            PU3.2 = 1;
#endif
            /* 设置静态变量*/
            sw_last = SW_LU_RU;     /* 默认右抬起,左抬起 (无按键按下) */
            sw_deb_value = SW_DEF_DEB_COUNT;    /* 设置默认计数器值*/
            sw_deb_count = SW_DEF_DEB_COUNT;    /* 设置计数器值最大 */
}

/* unsigned char sw_chk(void) */
/*          从按键返回输入, 未去抖动*/
unsigned char sw_chk(void)
{
            return P3 & 0x06;
}
```

```
/* void sw_set_debounce(unsigned char count) */
/*          设置去抖动计数器值*/
void sw_set_debounce(unsigned char count)
{
        sw_deb_value = count;     /* 设置新的去抖动计数器值*/
        sw_deb_count = count;     /* 设置计数器到最大 */
}

/* unsigned char sw_get(void) */
/*          返回去抖动按键输入 */

unsigned char sw_get(void)
{
        return sw_last;
}

/* void sw_isr( void ) */
/* 由周期定时器中断调用此程序并按键去抖动*/
/* 新值之后认为是 sw_deb_value 稳定时间, sw_last 被更新*/
void sw_isr( void )
{
unsigned char val;

        val = sw_chk();                 /* 获取当前值*/
        /* 如果值和以前一样没有变化；重设去抖动并返回*/
        if (val == sw_last) {
                sw_deb_count = sw_deb_value;     /* 重设去抖动计数器到最大*/
                return;
        }

        /* val != sw_last, 有新输入*/
        /* 如果和以前的值不同是个新值, */
        /* 设置 NEW 为一, 重设去抖动计数器并返回*/
        if (val != sw_new) {
                sw_new = val;
                sw_deb_count = sw_deb_value;
                return;
        }

        /* val != sw_last, val == sw_new */
        /* 去抖动计数器减 */
        sw_deb_count--;

        /* 如果减为零，得到与 sw_new 相同的 */
        /* 用于去抖动计数时间，现在是去抖动按键值*/
        if (sw_deb_count == 0) {
                sw_last = val;
                sw_deb_count = sw_deb_value;
                return;
        }
        /* 如果仍为抖动，返回*/
        return;
}
```

10.9.13 Option.inc

```
;*****************************************************************************
;**
;**  此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
;**  78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
;**
;**  版权(C) NEC Electronics Corporation 2002 - 2005
;**  归 NEC Electronics Corporation 版权所有。
;**
;**  若使用此程序用户必须自己承担责任。
;**  NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
```

```
;**
;** 文件名：        option.asm
;** 概要：          此文件执行选项字节/安全 ID 设置
;** APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
;**
;** 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
```

10.9.14 Option.asm

```
;*****************************************************************************
;**
;** 此设备驱动文件由 Applilet 产生适用于 78K0/KB2, 78K0/KC2,
;** 78K0/KD2, 78K0/KE2 和 78K0/KF2 8 位单片机。
;**
;** 版权(C) NEC Electronics Corporation 2002 - 2005
;** 归 NEC Electronics Corporation 版权所有。
;**
;** 若使用此程序用户必须自己承担责任。
;** NEC Electronics Corporation 对使用此程序出现问题的消费者和第三方不负担任何责任。
;**
;** 文件名：        option.asm
;** 概要：          此文件执行选项字节/安全 ID 设置
;** APIlib:         NEC78K0KX2.lib V1.01 [2005 年 8 月 9 日]
;**
;** 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
```

［备忘录］

NEC

传真信息

来自：

姓名：

公司：

电话：　　传真：

地址：

虽然NEC已经采取各种可行措施以确保供给客户完整的，无误的以及最新的文档，但我们也愿意接受可能出现的错误。尽管我们已经采取谨慎态度和预防措施，但你还有可能遇到文档问题，如果要向我们举报错误或提出建议，请完成这个表格。

感谢你的支持。

[北京]
日电电子（中国）有限公司
中国北京市海淀区知春路 27 号
量子芯座 7，8，9，15 层
电话：（+86）10-8235-1155
传真：（+86）10-8235-7679

[深圳]
日电电子（中国）有限公司深圳分公司
深圳市福田区益田路卓越时代广场大厦 39 楼 3901，3902，3909 室
电话：（+86）755-8282-9800
传真：（+86）755-8282-9899

上海恩益禧电子国际贸易有限公司
中国上海市浦东新区银城中路 200 号
中银大厦 2511-2512 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

[上海]
日电电子（中国）有限公司上海分公司
中国上海市浦东新区银城中路 200 号
中银大厦 2409-2412 和 2509-2510 室
电话：（+86）21-5888-5400
传真：（+86）21-5888-5230

[香港]
香港日电电子有限公司
香港九龙旺角太子道西 193 号新世纪广场
第 2 座 16 楼 1601-1613 室
电话：（+852）2886-9318
传真：（+852）2886-9022
2886-9044

我想举报错误/提出如下意见：

文档标题：

文档号：　　页码：

如果有可能，请传真引用页及图片。

文档等级	很好	较好	可接受	不好
清晰度	☐	☐	☐	☐
技术准确度	☐	☐	☐	☐
结构	☐	☐	☐	☐

CS 99。1

［备忘录］