HOLTEK I/O 8-Bit MTP
输入/输出型八位可多次编程单片机
初学者工具使用手册

二〇〇五年一月

注意：

使用手册中所出现的信息在出版当时相信是正确的，然而盛群对于说明书的使用不负任何责任。文中提到的应用目的仅仅是用来做说明，盛群不保证或表示这些没有进一步修改的应用将是适当的，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。盛群产品不授权使用于救生、维生器件或系统中做为关键器件。盛群拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网址 http://www.holtek.com.tw

目录

第一章

开始 1

简介

MTP(可多次烧写)使用者工具是一个认识 HOLTEK 8 位微控制器的快捷便利、低成本途径。它也可以作为 MTP 编程器和验证板。

特性

- 简单的印刷电路板
- 通过 PC 端 USB 口进行通讯
- 简单的 MTP 初学者工具软件应用
- 范例程序可供应用，修改简单
- 支持不同型号的 MTP 微控制器

初学者工具配备

- MTP 初学者工具印刷电路板
- USB 连接线
- HT-IDE3000 和初学者工具 CD-ROM
- 用户手册

安装 MTP 初学者工具

将 HT-IDE3000 CD 放入光驱，PC 会显示图 1-1。

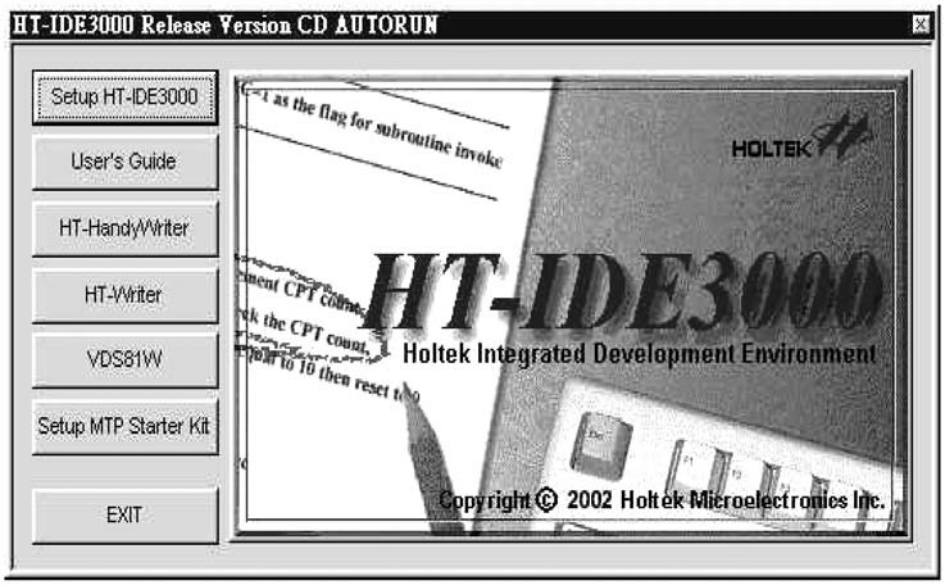


图 1-1

点击<Setup MTP Starter Kit>按键，PC 会显示图 1-2。

图 1-2

在图 1-3 中指定 MTP 初学者工具的安装目录，按下<Next>按键。

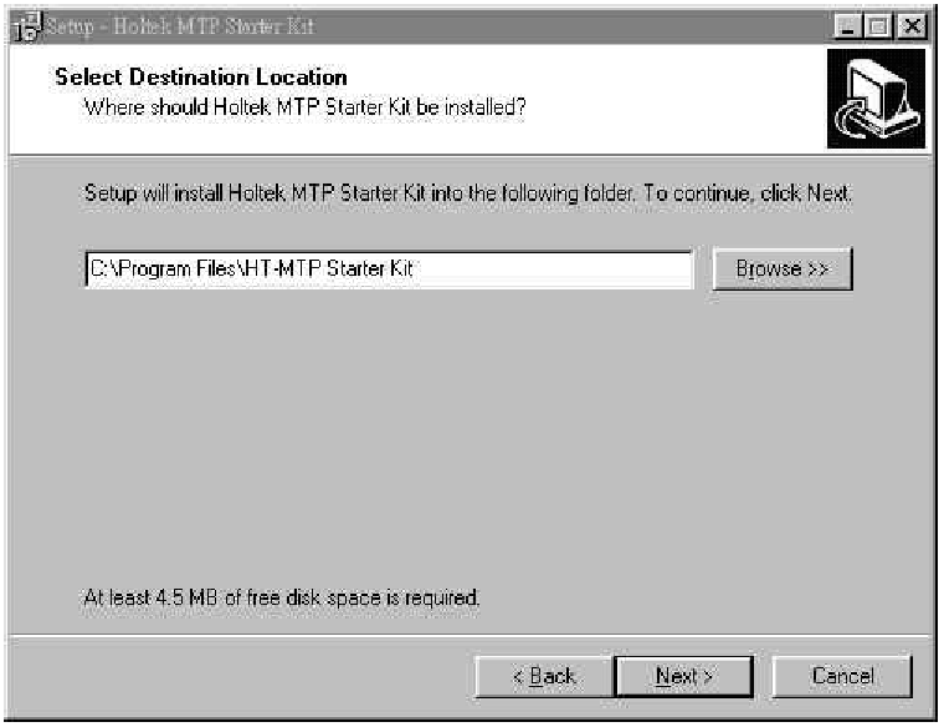


图 1-3

然后指定图 1-4 中的快捷名，它将会被放入操作系统开始菜单的程序组中。

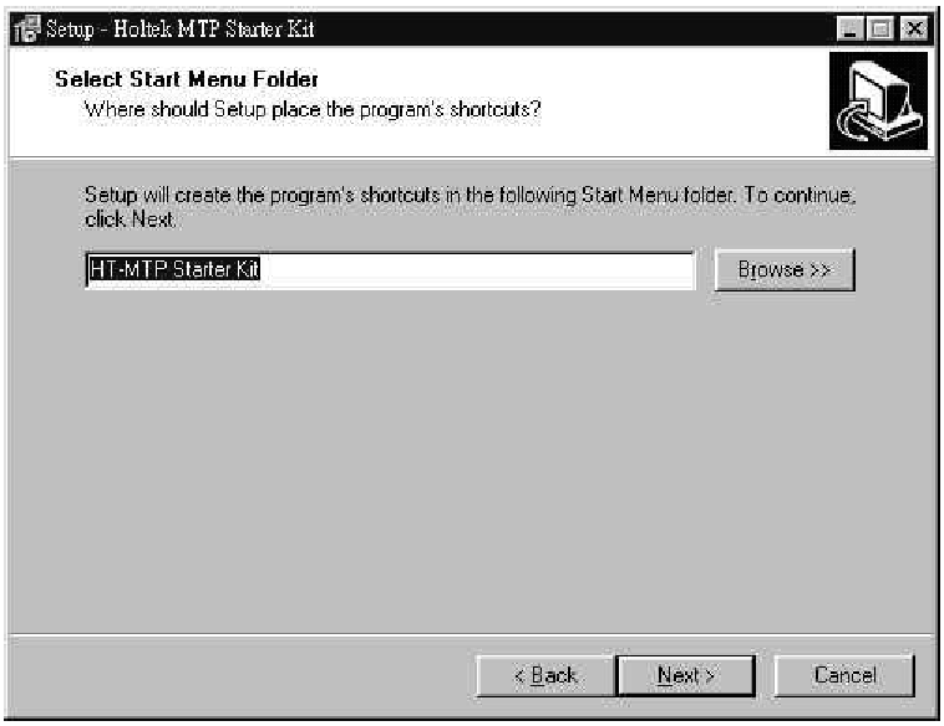


图 1-4

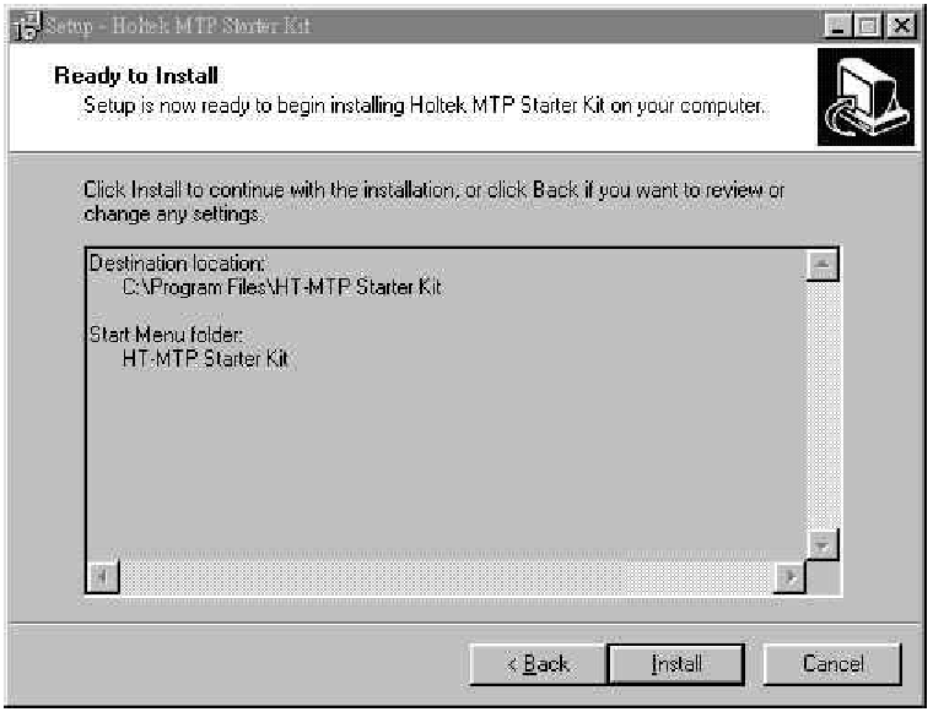

Setup - Holtek MTP Starter Kit
Ready to Install
Setup is now ready to begin installing Holtek MTP Starter Kit on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Program Files\HT-MTP Starter Kit
Start Menu folder:
HT-MTP Starter Kit
< Back
Install
Cancel

图 1-5

在图 1-5 中，按下<Install>开始安装。所有相关的文件将会复制到指定的目标地址。如果安装成功，会显示图 1-6。按下<Finish>按键，就可以执行 MTP 初学者工具了。

图 1-6

运行初学者工具内置例程

通过 USB 连接线，将 MTP 初学者工具 PCB 印刷板和 PC 的 USB 端口连结起来后，MTP 单片机内置的例程会自动运行。

LED 灯会依次亮起，两个 7 段显示码也会显示计数数据。

如果你下载其它的范例程序到 MTP 单片机中，那么内置的例程会被覆盖。需要的话，可以重新下载内置例程 LIGHT.MTP 到 MTP 单片机中。

如果是没有安装 USB 驱动的 Win95/98 系统，那么 PC 会显示图 1-7 ，请按下<Next>按键。

图 1-7

图 1-8 会显示，选择”Search the best driver for your device”，按下<Next>按键。

图 1-8

图 1-9 会显示。选择”CD-ROM Drive”并插入 Win98 系统光盘，按下<Next>按键顺序执行安装程序。

图 1-9

运行初学者工具范例程序

按下开始菜单，选择程序中的 MTP 初学者工具。如图 1-10 的主窗口会弹出来。

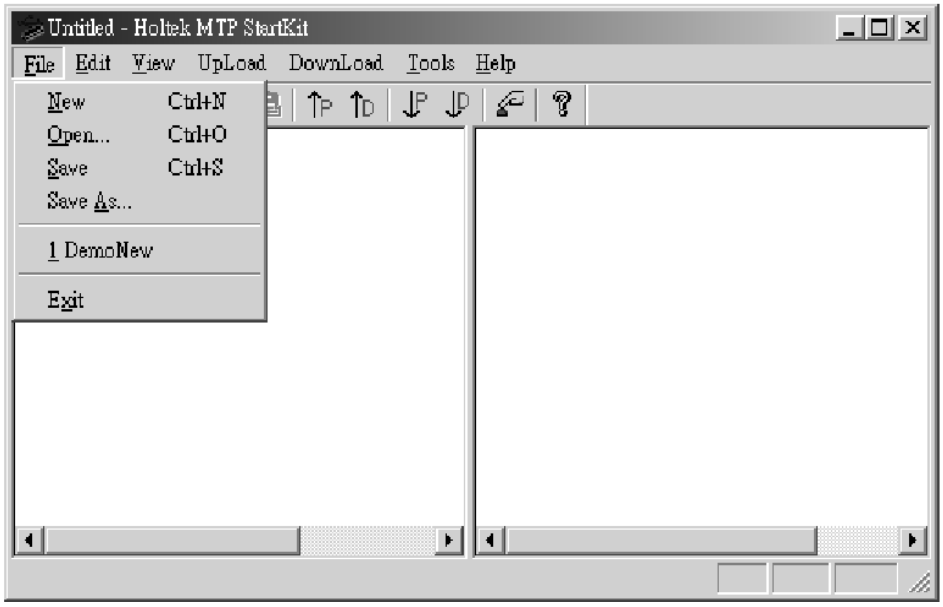


图 1-10

选择菜单 File/Open 打开一个范例程序文件（例如：CLOCK.MTP）。文件内容就会被显示在此窗口（如图 1-11）。窗口左侧框内显示程序的二进制代码，窗口右侧框内显示要被写入数据 EEPROM 的代码。

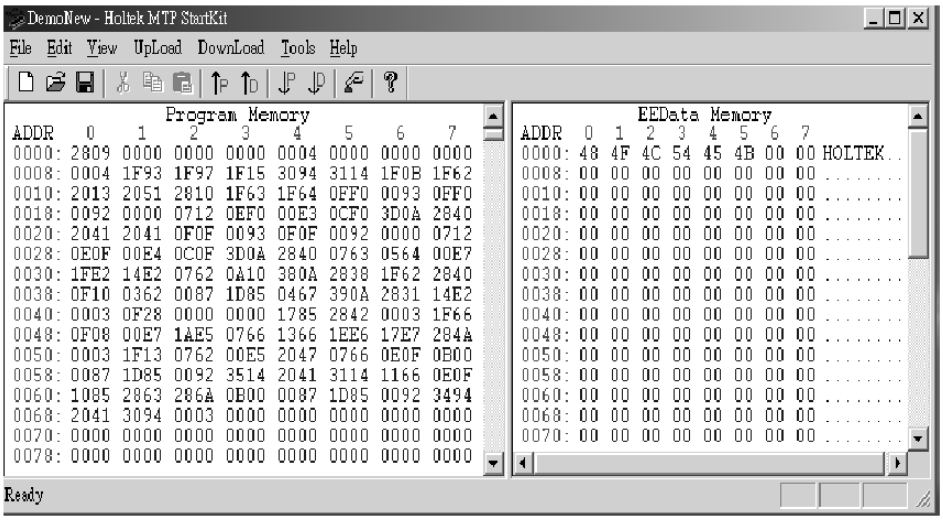


图 1-11

• 选择菜单 Download/ALL 如图 1-12。将程序和数据代码下载到 MTP 单片机 HT48E50 的对应地址中去。

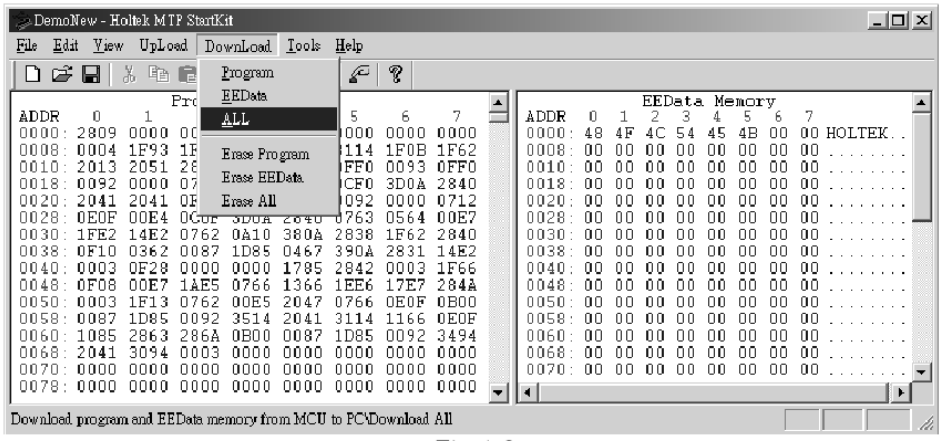


图 1-12

• 下载程序时，会出现显示下载百分比的进度条。下载结束后，按下 Reset 按键运行范例程序。

MTP 初学者工具菜单和命令

共有七个菜单，File，Edit，View，Upload，Download，Tools 和 Help。具体描述如下。

菜单——File

file 菜单包括如下的文件管理命令：

- New
  创建一个新文件
- Open
  打开已存在的.MTP 文件。在打开指定文件前，此命令自动关闭当前已经打开的文件。
- Save
  将当前窗口数据保存为当前文件。
- Save As
  将当前窗口数据另存为指定文件。
- Recent Files
  列出最新打开和关闭的四个文件。
- Exit
  退出初学者工具软件，回到 Windows 界面。

菜单——Edit

- Undo
  取消前一个编辑动作。
- Redo
  恢复前一个被取消的动作。
- Cut
  移除文件中被选中的行，并复制到剪贴板上。
- Copy
  将文件中被选中的行复制到剪贴板上。
- Paste
  将剪贴板上的内容复制到当前光标处。
- Delete
  删除选中内容。

- Find
  在当前编辑的文件中查找指定的一段内容。
- Replace
  在当前编辑的文件中，将一段内容替换成另一段指定的内容。

菜单——View

View 菜单包含如下命令，可用于调整初学者工具软件的窗口显示：

- Toolbar
  在窗口中显示工具栏。工具栏包括若干个快捷键，它们分别对应菜单中的相关项。如果鼠标的光标停留在某个快捷键上，那么对应的功能说明会在此快捷键旁边显示。如果按下某个快捷键，那么对应的命令就会被执行。
- Status Bar
  在窗口中显示状态栏。

菜单——Download

Download 菜单（如图 1-12）包含如下命令，可对 MTP 单片机进行编程：

- Program
  将当前打开的程序代码写入 MTP 单片机程序存储空间中。下载程序时，会出现显示下载百分比的进度条。
- EEPROM Data
  将当前打开的数据代码写入 MTP 单片机数据存储空间中。
- ALL
  将当前打开的程序和数据代码写入 MTP 单片机相对应的存储空间中。
- Erase Program
  清除 MTP 单片机程序存储空间中的代码。
- Erase EEData
  清除 MTP 单片机数据存储空间 EEPROM 中的代码。
- Erase All
  清除 MTP 单片机程序和数据存储空间 EEPROM 中的代码。

菜单——Upload

Upload 菜单（如图 1-13）包含如下命令，可读出 MTP 单片机中的数据：

- Program
  读出 MTP 单片机中的程序代码，并显示在当前打开的窗口。如果需要保存程序代码，请选择菜单 Files/Save As 命令。文件的扩展名是 . MTP。
- EEPROM Data
  读出 MTP 单片机中的 EEPROM 数据代码，并显示在当前打开窗口的右边框内。如果需要保存数据代码，请选择菜单 Files/Save As 命令。文件的扩展名是 . MTP。
- ALL
  读出 MTP 单片机中的程序代码和 EEPROM 数据代码，并分别显示在当前窗口的左边和右边框内。如果需要保存代码，请选择菜单 Files/Save As 命令。文件的扩展名是 . MTP。
- Verify Program
  读出 MTP 单片机中的程序代码，并与当前窗口中的程序代码比较是否一样，结果会显示在屏幕上。
- Verify Data
  读出 MTP 单片机中的 EEPROM 数据代码，并与当前窗口中的数据代码比较是否一样，结果会显示在屏幕上。
- Verify All
  读出 MTP 单片机中的程序代码和 EEPROM 数据代码，并与当前窗口中的代码比较是否一样，结果会显示在屏幕上。

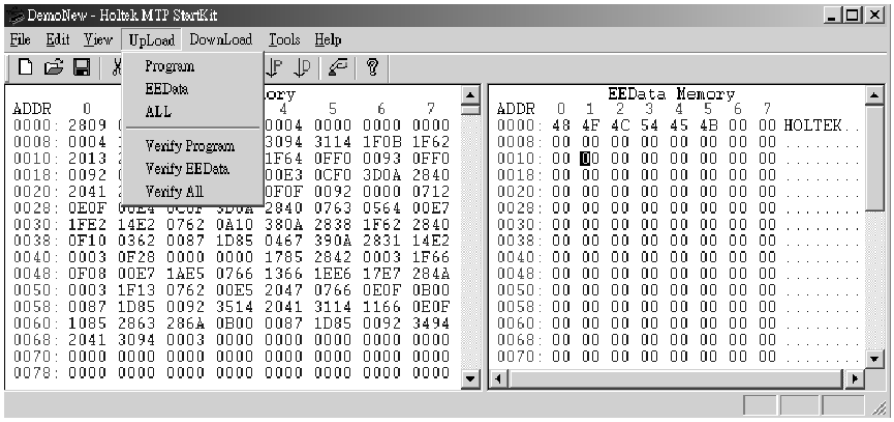


图 1-13

菜单——Tools

Tools 菜单（如图 1-14）包含如下命令，可设定 MTP 单片机的配置选项，并在编程后控制单片机的运行：

- View Option
  在屏幕上显示 MTP 单片机的配置选项。
- Reset
  执行复位动作，功能和工具栏上快捷键中的复位按键一样。
- Power On
  打开初学者工具板上的 MTP 单片机电源。这个命令用在替换 MTP 单片机的时候。在更换 MTP 单片机前会将单片机电源关闭，然后再次打开单片机电源，让新的 MTP 单片机运行程序。
- Power Off
  当你想替换初学者工具板上的 MTP 单片机时，无论这个新的 MTP 单片机是空片或者已经被写入程序了，与原来的 MTP 单片机型号相同或者不同，都需要使用这个命令来关闭 MTP 单片机的电源。替换之后，再执行 Power On 命令打开电源。

  这两条命令可以让用户无需拔除初学者工具板，就可以替换 MTP 单片机。

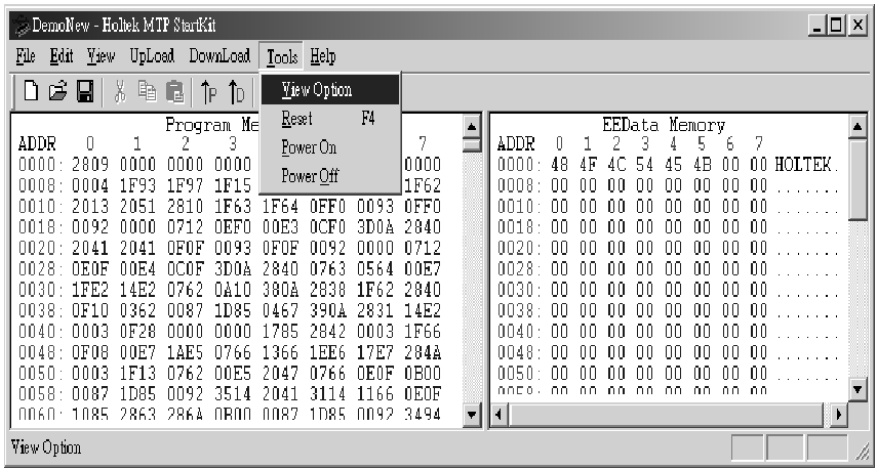


图 1-14

第二章

范例程序 2

这章介绍适用于 MTP 初学者工具的范例程序。范例程序被下载到初学者工具电路板上后，程序对应的相关功能就会被执行。每个范例程序都包含：程序说明、程序流程图、汇编语言和 C 语言程序代码。

源程序代码可以在CD或者硬盘的相关路径中找到。这些代码可以被用户修改，并下载到初学者工具板上验证。对此第三章会做详细说明。下面列出了包含的范例程序：

- 蜂鸣器
- 扫描键阵
- 时钟显示
- 访问 EEPROM 数据存储器
- 乐曲

范例程序 1：蜂鸣器

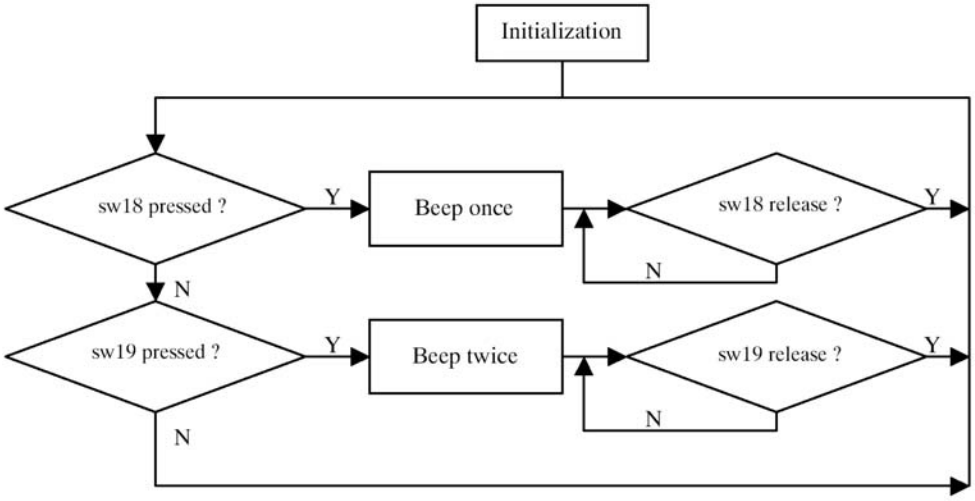
BZ 和 $\overline{\text{BZ}}$ 是一对值得称赞的蜂鸣器信号输出口。定时器 0（8 位内部定时器）或者定时器 1（16 位内部定时器）驱动蜂鸣器。对于定时器 0 或者定时器 1 来说，装载不同的预分频数和初值可以产生不同的蜂鸣器频率。BZ 和 $\overline{\text{BZ}}$ 首先要被置为输出状态。将 BZ 管脚置高就会启动蜂鸣器功能，反之就关闭蜂鸣器功能。

定时器 0 是一个 8 位的向上计数器，它的时钟源来自系统时钟或者外部时钟。时钟来源先被预分频器分频（8 位分频器），分频数可以由定时器控制寄存器 TMR0C 的三个位 PSC2，PSC1 和 PCS0 来决定。定时器 0 的计数值和初始值都会被装载入寄存器 TMR0，计数启动时 TMR0 向上累计直到溢出。

在范例程序中，我们选择定时器 0 作为蜂鸣器的时钟来源，定时器时钟来源是系统频率，预分频数是 1/4。定时器 0 的初值设为 0x80H。BZ（和 PB0 共享）作为输出口。我们可以通过两个按键来测试：sw18 和 sw19 如下：

- 按下 sw18 按键，蜂鸣器发出“嘟”一次，耗时 250 毫秒。
- 按下 sw19 按键，蜂鸣器发出“嘟”两次，每次耗时 125 毫秒，“嘟”声之间间隔 125 毫秒。

流程图



相关文件

BUZZER.MTP 是一个需要下载的文件。若你希望运行此范例程序，在菜单 File/Open 如图 1-7 中浏览并且指定此文件。**BUZZER.ASM** 是范例程序汇编语言源代码。**BUZZER.C** 是 C 语言版本。在下一章里会详细描述如何修改这些源文件，以满足用户自己的需要。

范例程序 2：扫描键阵

此范例程序提供了 4×4 的按键阵列和显示按下哪个按键的 7 段 LED。可显示数目是 1~16，如果无键按下或者无效键按下则显示“0”。

扫描方式

这里介绍扫描转换方式。首先端口 A 必须做以下设定：

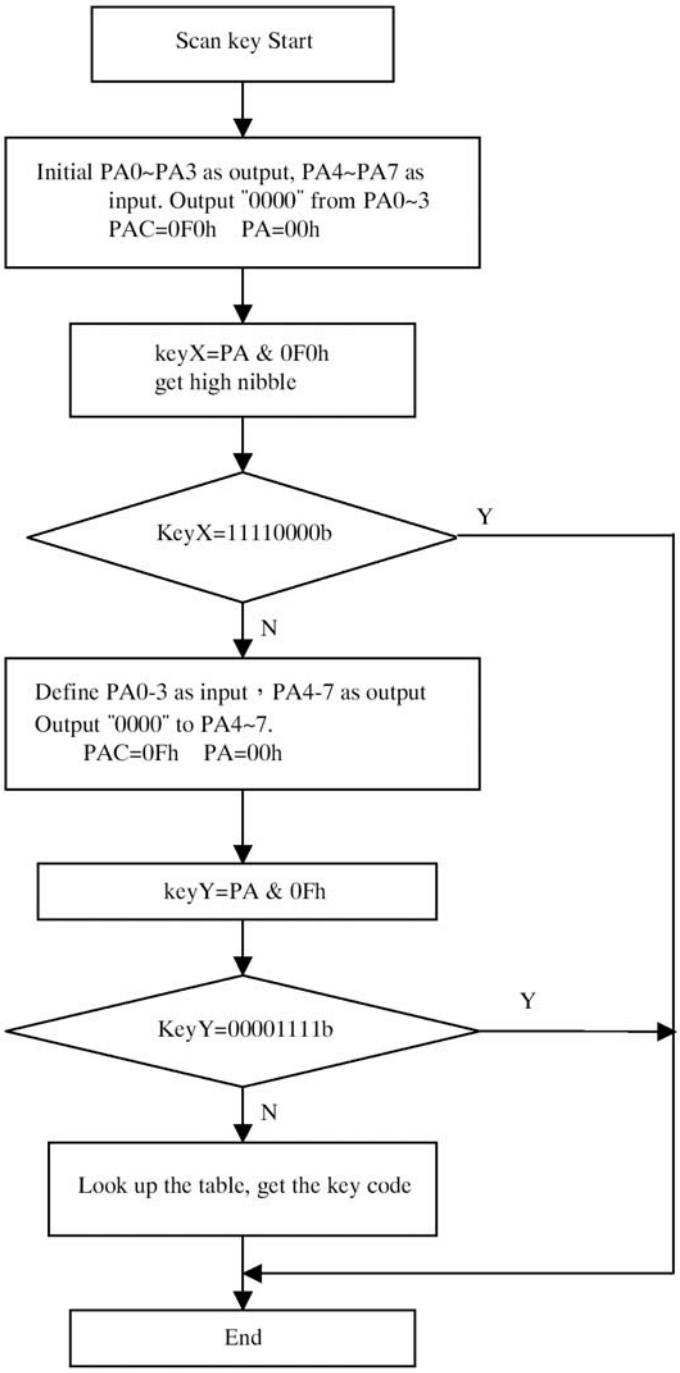
- 将端口 A 设成斯密特输入，来减弱按键抖动影响
- 将端口 A 设成内部上拉

在程序起始部分，将 PA0~PA3 设成输出状态，PA4~PA7 设成输入状态。然后输出“0000”给 PA0~PA3 并读入 PA4~PA7。由于 PA4~PA7 被设置为带有内部上拉，当没有按键按下时，读入的值是“1111b”。反之，若有按键按下，PA4~PA7（不为“1111b”）的值被存在变量“KeyX”。下一步，将PA0~PA3 变成输入状态，PA4~PA7 位输出状态，然后用相同的方法输出“0000b”给 PA4~PA7，并读入 PA0~PA3 的值。将读出的值放入“KeyY”。将“KeyY”和“KeyX”放入一个 8 位数据中，查按键状态表格就可以得到键值，最后显示键值在 LED 上。

相关文件

SCANKEY.MTP 是一个需要下载的文件。在菜单 File/Open 如图 1-7 中浏览并且指定此文件。**SCANKEY.ASM** 是范例程序汇编语言源代码。**SCANKEY1.C** 是 C 语言版本。

流程图


Scan key Start
Initial PA0~PA3 as output, PA4~PA7 as input. Output "0000" from PA0~3
PAC=0F0h PA=00h
keyX=PA & 0F0h
get high nibble
KeyX=11110000b
Y
N
Define PA0-3 as input，PA4-7 as output
Output "0000" to PA4~7.
PAC=0Fh PA=00h
keyY=PA & 0Fh
KeyY=00001111b
Y
N
Look up the table, get the key code
End

范例程序 3：时钟显示

此范例程序示范了如何使用定时器。当定时器开始工作，7 段 LED 会显示 1 秒~99 秒的时间。当超过 99 秒，LED 会复位到零，并从 1 秒开始重新计数。按下 sw18 按键可以停止定时器。再次按下 sw18 按键，定时器会从刚才停止计数的地方开始继续计数。Sw19 按键可以将计数时间值复位到“0”。

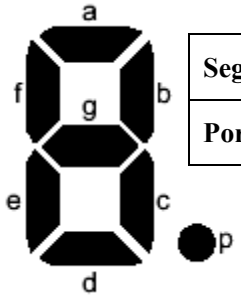
定时器 0 是作为秒表用的。它是一个 8 位内部定时器，时钟来源是系统时钟。定时器 0 的时基被设定成 1ms，因此当定时器 0 发生 1000 次中断就表示 1 秒时间已经计到了。为了实现上述功能，定时器 0 寄存器的初值必须事先计算完成。每收到一个内部时钟脉冲，寄存器的值就会向上增一。当定时器 0 计到 FFH 的时候，也就是定时器 0 溢出的时候，就会重新装载初值，并且产生一个内部中断信号。相对应的中断服务子程序会被执行。定时器 0 的值会被复位成设定的初始值，并开始继续计数。

按照如下步骤，计算定时器 0 寄存器的初值：

- 内部定时时钟来源是系统时钟，此程序中系统时钟为 4MHz
- 在 TMR0C 寄存器(0E0H)中设定时间预分频，此程序中设定 1：16 分频。因此一个内部时钟脉冲需要 1/（（4×10^{6}）÷16）=4×10^{-6} 秒
- 定时器 0 每隔 1ms 溢出一次并产生中断。假设需要计数 R 次后溢出，那么（4×10^{-6} ）×R=1ms=1×10^{-3} 秒

  因此 R=250，TMR0 的初始值应该是（256-250）或者说是 6

我们在定时器项目中用到了两个 7 段 LED。下表列出了对应的 Segment 和管脚之间的关系。



Segment	a	b	c	d	e	f	g	p
Port	PA2	PA3	PA4	PA7	PA5	PA1	PA0	PA6

这 7 段码是共阳结构。

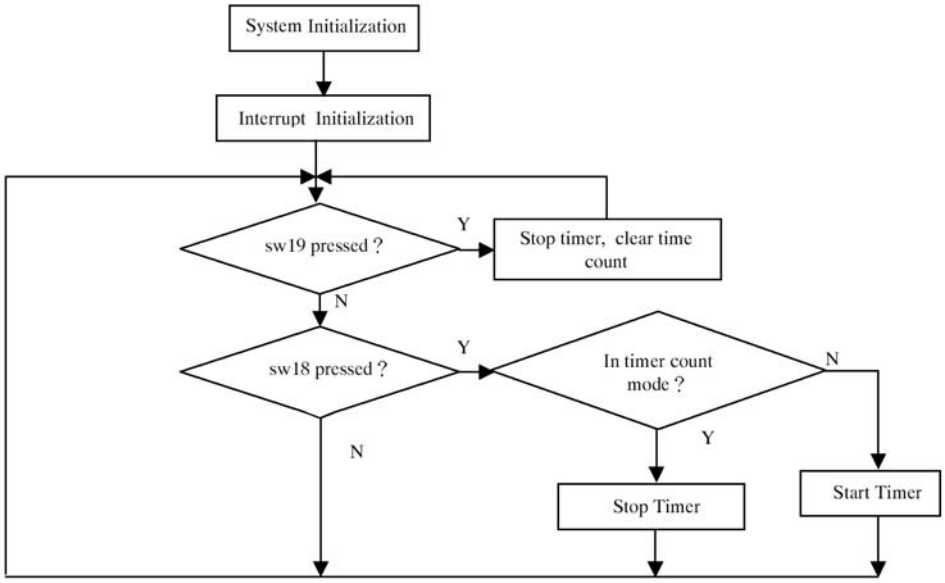
要点亮 LED 模块上的某个段，就要将对应的输出口设为“0”，此对应关系由外围电路决定。例如，要点亮 a 段，则需要将 PA2 输出口设为低电平。如下表格列出了字符“0~9”和“A~F”对应的 PA 端口码值。

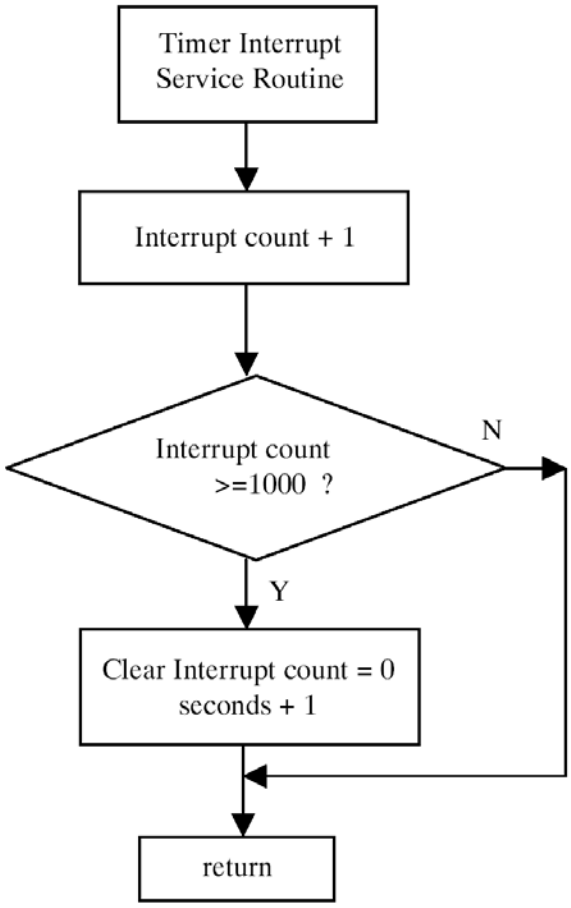
Character	0	1	2	3	4	5	6	7	8	9
Code	41h	E7h	52h	62h	E4h	68h	48h	E3h	40h	60h
Character	A	b	C	d	E	F				
Code	C0h	4Ch	59h	46h	58h	D8h				

相关文件

CLOCK.MTP 是一个需要下载的文件。在菜单 File/Open 如图 1-7 中浏览并且指定此文件。**CLOCK.ASM** 是范例程序汇编语言源代码。**CLOCK1.C** 是 C 语言版本。

流程图





范例程序 4：访问 EEPROM 数据存储器

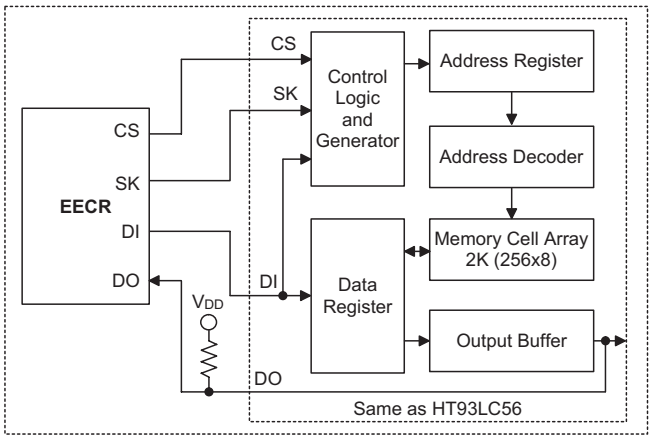
此范例程序会从 MTP 单片机的 EEPROM 数据存储器中读取数据，并将它作为初值赋给 7 段 LED 码显示。定时器开始从此初始值开始计数，并将它显示出来，直到指定按键按下。然后定时器停止计数，当前时间将被写入 EEPROM 数据存储器，等待下一次运行时取出。此范例程序的目的是展示如何访问 MTP 单片机中的 EEPROM 数据存储器。

EEPROM 数据存储器

EEPROM 数据存储器包含 256×8 位，在正常工作中可读可写。它要通过控制寄存器 EECR（[40H]在 Bank 1）间接访问。寄存器 EECR 只能使用 MP1 间接读写。

Label(EECR)	Bits	功能
——	0~3	未定义，读出为“0”
CS	4	EEPROM 数据存储器片选
SK	5	EEPROM 数据存储器串口时钟输入
DI	6	EEPROM 数据存储器串口数据输入
DO	7	EEPROM 数据存储器串口数据输出

访问 EEPROM 数据存储器需要三根串行通讯线，通过 EECR 界面写入。EEPROM 数据存储器有 256 个字，每个字包含 8 个位。EEPROM 数据存储器有 7 个指令：READ,ERASE,WRITE,EWEN,EWDS,ERAL 和 WRAL。对 HT48E50 来说，这些指令包含 12 个位：1 个起始位，2 个 op-code 位和 9 个地址位。通过写 CS,SK 和 DI，这些指令可以被输入到 EEPROM 中。这些在 DI 线上的串行指令数据会在 SK 上升沿写入 EEPROM 数据存储器。在读取周期中，DO 作为数据输出线，在 WRITE 或者 ERASE 周期，DO 处于忙/准备状态。当 DO 作为输入数据线或者处于忙/准备状态，CS 管脚必须置高；反之 DO 会处于一个高状态。指令发送成功之后，CS 必须马上置为低电平。上电后，系统默认处于 EWDS 状态。执行 ERASE 或者 WRITER 指令之前，需要先执行 EWEN 指令。接下来详细描述所有 7 条指令的功能。



EEPROM 数据存储器框图

READ

读取指令会将指定地址的数据送出到 DO 线上。在 SK 上升沿时，DO 线上的数据会跳变。8 位数据之前有一个逻辑“0”信号。无论 EWEN 或者 EWDS 指令执行与否，读取指令都可以有效执行。一个数据字被读取之后，EEPROM 内地址会自动加一，允许下一个连续的数据被读取，而不需要输入任何地址数据。地址数据会循环累加直到 CS 从高电平变为低电平。

EWEN/EWDS

EWEN/EWDS 指令可以使能或者禁止擦写动作有效执行。无论是处于上电或者掉电状态，芯片都会自动进入禁止擦写模式。执行 WRITE,ERASE,WRAL 或者 ERAL 指令之前，必须先执行 EWEN 指令使能擦写动作有效执行，反之，执行 ERASE/WRITE 指令就是无效的。EWEN 指令运行后，擦写动作使能状态就被启动，直到断电或者 EWDS 指令被运行。当处于禁止擦写状态时，就无法向 EEPROM 数据存储器进行擦写。这样做的话，内部数据就相当于被保护了。

ERASE

在擦写使能状态下，ERASE 指令可以擦除指定地址的数据内容。ERASE 指令码和指定地址送出后，数据在 CS 下降沿会被清除。由于芯片内部有自动时序发生器，提供所有时序信号给内部擦除动作，所以 SK 时钟不需要外部提供。内部擦除动作期间，当 CS 保持高电平我们可以检测到系统状态是忙还是空闲。忙状态时 DO 线保持低电平，直到擦除动作结束，DO 会恢复到高电平，此时其它指令可以被执行。

WRITE

擦写使能模式下，WRITE 指令可以将数据写入指定地址的 EEPROM 数据存储器。WRITE 指令码和指定地址送出后，数据在 CS 下降沿会被写入。由于芯片内部有自动时序发生器，提供所有时序信号给内部写入动作，所以 SK 时钟不需要外部提供。整个自动写入周期包括，擦除然后再写入。因此不需要在写入指令前擦除数据。内部写入动作期间，当 CS 保持高电平我们可以检测到系统状态是忙还是空闲。忙状态时 DO 线保持低电平，直到擦除动作结束，DO 会恢复到高电平，此时其它指令可以被执行。

ERAL

在擦写使能状态下，ERAL 指令可以擦除所有 256×8 存储器位单元为逻辑“1”。擦除所有内容指令送出后，数据在 CS 下降沿会被清除。由于芯片内部有自动时序发生器，提供所有时序信号给擦除所有内容动作，所以 SK 时钟不需要外部提供。内部擦除所有内容动作期间，当 CS 保持高电平我们可以检测到系统

状态是忙还是空闲。忙状态时 DO 线保持低电平，直到擦除动作结束，DO 会恢复到高电平，此时其它指令可以被执行。

WRAL

在擦写使能状态下，WRAL 指令可以写入所有 256×8 存储器。写入所有内容指令送出后，数据在 CS 下降沿会被写入。由于芯片内部有自动时序发生器，提供所有时序信号给写入所有内容动作，所以 SK 时钟不需要外部提供。内部写入所有内容动作期间，当 CS 保持高电平我们可以检测到系统状态是忙还是空闲。忙状态时 DO 线保持低电平，直到擦除动作结束，DO 会恢复到高电平，此时其它指令可以被执行。

指令	描述	起始位	指令码	地址	数据
READ	Read Data	1	10	X,A7~A0	D7~D0
ERASE	Erase Data	1	11	X,A7~A0	——
WRITE	Write Data	1	01	X,A7~A0	D7~D0
EWEN	Erase/Write Enable	1	00	11XXXXXXX	——
EWDS	Erase/Write Disable	1	00	00XXXXXXX	——
ERAL	Erase All	1	00	10XXXXXXX	——
WRAL	Write All	1	00	01XXXXXXX	D7~D0

注意：”X”表示不确定

擦写 EEPROM

由于 EECR 地址是在 BANK1 中的 40H，因此只有通过 MP1 间接寻址来读取或者写入。BANK1 中没有定义其它的寄存器，因此我们可以在程序起始将 BP 设为 1，MP1 设为 40H，这在整个程序中都不会有改动。通过 IAR1，我们可以对 EECR 读写来读取/写入 EEPROM 数据存储器。下面是采用 HOLTEK 汇编指令编写的读取/写入 EEPROM 数据存储器的范例。

```
// define the constant
CS     EQU     IAR1.4        ; CS is the 4th bit of IAR1 ( EERC register )
SK     EQU     IAR1.5        ; SK is the 5th bit of IAR1 ( EERC register )
DI     EQU     IAR1.6        ; DI is the 6th bit of IAR1 ( EERC register )
DO     EQU     IAR1.7        ; DO is the 7th bit of IAR1 ( EERC register )
_EECR EQU     40H
C_WRITECOM EQU 5<<5          ; Write data eeprom command
C_READCOM EQU 6<<5           ; Read data eeprom command

// define constant according to the MTP microcontroller type
ifdef    HT48E06  ; condition compile, for HT48E06
C_Addr_Length  EQU  7        ; total bits of  EEPROM address
C_Data_Length  EQU  8        ; total bits of EEPROM data
endif

ifdef     HT48E50             ; for HT48E50
C_Addr_LenghtEQU 8            ; total bits of EEPROM address
C_Data_LengthEQU 8            ; total bits of EEPROM data
endif
```

下面有四个子程序：WriteCommand,WriteAddr,WriteData 和 ReadData，用来实现 EEPROM 的访问动作。每个命令都包括一个起始位和两个指令码，总共需要写三位数据。

- 输出命令码

此程序输出读命令(110)或者写命令(101)的三位数据

这三个位存放在 EE_command 的第 7、6、5 位

WriteCommand:

```
MOV     A,3              ;Start bit and op code of a command
                         ;(total 3 bits)
MOV     COUNT,A
WriteCommand_0:
CLR      DI              ;set the bit value of a command
SZ      EE_command.7     ;if bit 7 of EE_command register is 1
SET      DI              ; then write 1,else writ 0 to DI

SET      SK              ; output the bit at the rising edge
CLR      SK              ; change SK to low to prepare write the next
                         ; bit
CLR      C               ; clear carry flag bit
RLC      EE_command      ;move to the next bit being written
SDZ      COUNT            ; all bits have been output ?
JMP      WriteCommand_0   ; no, continue to output
RET                       ; yes, return
```

- 输出 EEPROM 数据存储器地址

对于 HT48E50，需要输出 9 位地址，其中 8 位有效。其中第一位是“0”或者“1”都可以。对于 HT48E06，需要输出 7 位地址，其中 7 位都是有效地址。

WriteAddr :

```
MOV     A, C_Addr_Length   ; total bits of EEPROM address
MOV     COUNT, A           ; =8 (HT48E50) or 7 (HT48E06)
ifndef  HT48E06            ;only HT48E50 has the following
                            ; instructions
SET     SK                 ;output the first bit (don't care bit)
NOP
CLR     SK
endif

WriteAddr_0:          ;output 8 bits of address for HT48E50
CLR     DI            ;only the highest 7 bits are valid for HT48E06
SZ      ADDR.7
SET     DI
CLR     C
RLC     ADDR          ; move to next bit
SET     SK            ; generate a falling edge
NOP
CLR     SK
SDZ     COUNT         ; total address bits have been output ?
JMP     WriteAddr_0   ; no, continue to output
RET                   ; yes, return
```

- 写入 EEPROM 数据存储器数据

此程序输出 EEPROM 数据存储器指定地址的数据。写入 8 位数据。

WriteData:

```
MOV     A, C_Data_Length ; total data bit
MOV     COUNT, A

WriteData_0:
CLR      DI              ; output the data bit
SZ       WR_Data.7
SET      DI

CLR      C
RLC      WR_Data          ; move to the next bit
SET      SK               ; generate a falling edge, to write data

CLR      SK

SDZ      COUNT            ; total data bit have been output ?
JMP      WriteData_0      ; no, continue to output

CLR      CS               ; generate a TCDS size low pulse to start
NOP                       ; writing
SET      CS

SNZ      DO               ; wait for the completion of writing
JMP      $-1              ; still in writing
RET                       ; writing completed, return
```

- 读取 EEPROM 数据存储器数据

此程序读出 EEPROM 数据存储器的数据，并存入 WR_Data 缓存器。

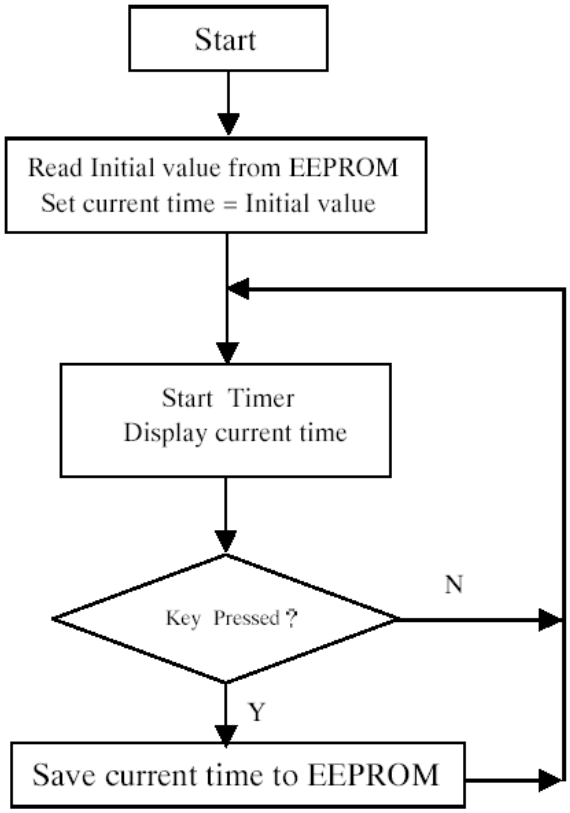
ReadData:

```
MOV     A, C_Data_Length    ; total read bits
MOV     COUNT, A

CLR     WR_Data             ; clear WR_Data

ReadData_0:
CLR     C                   ; prepare the place where the read bit
                            ; will be store
RLC     WR_Data             ; move read data to high bit
SET     SK                  ; EEPROM prepares the data at the rising
                            ;edge
SZ      DO                  ; read data bit, if it is 0, don't change
                            ; bit value
SET     WR_Data.0           ; if it is 1, set the WR_Data
CLR     SK

SDZ     COUNT               ; all data bits have been read ?
JMP     ReadData_0          ; no, continue to read

MOV     A, WR_Data          ; move the read data to accumulator ACC
RET
```

相关文件

EE_DATA.MTP 是一个需要下载的文件。在菜单 File/Open 如图 1-7 中浏览并且指定此文件。**EE_DATA.ASM** 是范例程序汇编语言源代码。
EE_DATA.C 是 C 语言版本。

流程图



范例程序 5：乐曲

此范例程序通过对 HT48E50 的两个内部定时器编程，使用蜂鸣器来播放乐曲。蜂鸣器输出的时钟来源就是 HT48E50 的两个定时器：定时器 0 和定时器 1。定时器 0 包含一个 8 位可编程向上计数器，定时器 1 包含一个 16 位可编程向上计数器。它们的时钟来源可以是外部信号或者系统频率。此范例程序中，蜂鸣器输出的时钟来源于定时器 1，而定时器 1 的时钟来源于系统时钟四分频，T1。
不同的计数器初值会产生不同的时间的中断溢出，因此也会产生不同的蜂鸣器频率输出。

此范例程序中，蜂鸣器输出的时钟来源于定时器 1。7 个基本音符频率在第五个八度（例如 Do, Re, Mi, Fa, So, La, Ti），分别是 523Hz, 587Hz, 659Hz, 698Hz, 783Hz, 880Hz 和 987Hz。

定时器 1 的初值计算公式如下（系统频率设定为 4MHz）

$$\frac{1}{(4\times10^{6})/4}\times R=\frac{1}{f_{NOTE}}$$

R 是定时器的计数值，F_{NOTE} 是音符频率。当 F_{NOTE} 等于 523Hz，即“Do”音符，此时 R 值等于 778h。因此定时器 1 初值寄存器的值应该等于 0-778h=0F888h。低字节是 88h，高字节是 0F8h。下表列出了和音符对应的定时器 1 初始值。

音符	频率（Hz）	定时器 1 初始值	
		TMR1H(Hex)	TMR1L(Hex)
Do	523	f8H	88H
Re	587	f9H	59H
Mi	659	faH	13H
Fa	698	faH	68H
So	783	fbH	03H
La	880	fbH	90H
Ti	987	fcH	06H

此范例程序中，定时器 0 作为音符长度周期的基础定时器。定时器 0 每个 8 毫秒产生一次中断，而音符长度设为这个时间因子的倍数。变量 T_Counter 作为一个倍数，那么音符长度=8ms×T_Counter。

依照索引数 PAT1，倍数可以从倍数表格中取出，以下就是表格中包括的 16 个倍数：

16，32，48，64，80，96，112，128，144，160，176，192，208，224，240，255

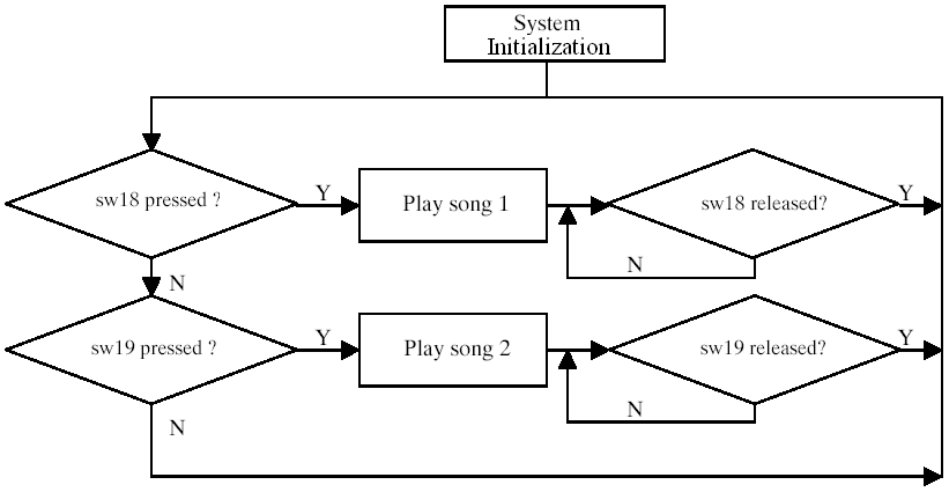
此范例程序中，PAT1 设为 4，则 T_Counter 等于 80，即倍数表格中第五个倍数。那么此音符等于 640ms(8×80ms)。将 PAT2 定义成双倍的 PAT1，那么它就等于 1280ms。倍数表格和 PAT1、PAT2 都可以根据需要调整。

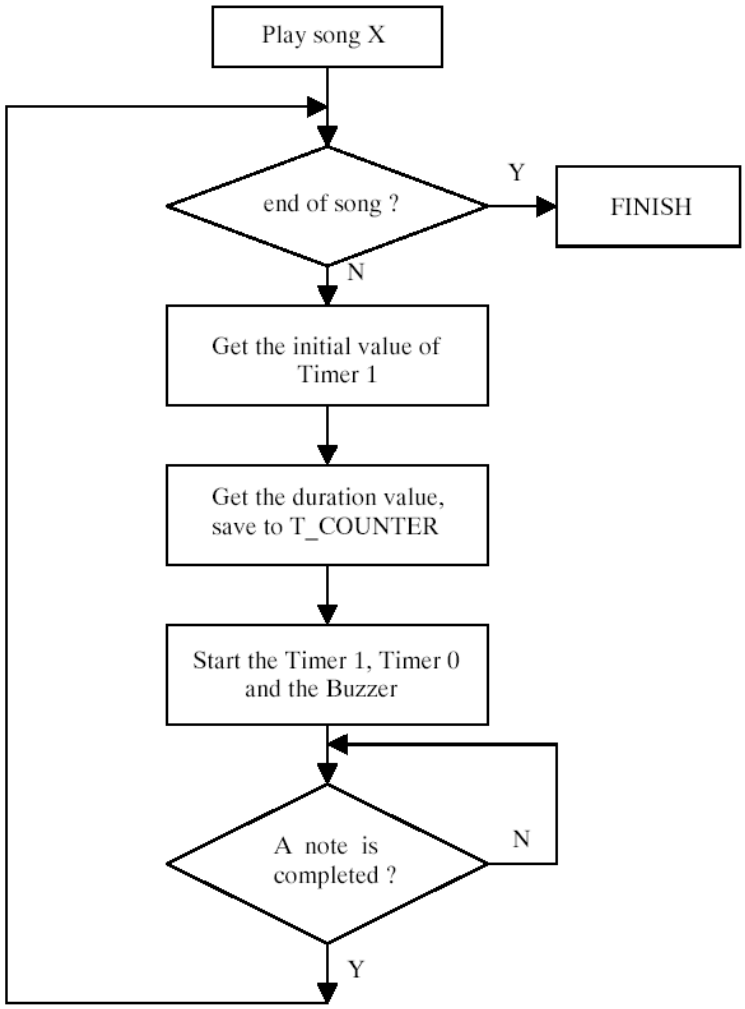
乐曲表格的数据格式
每个音符都是由音调和播放长度组成。音调索引数参考音符表格的第一个字节。音符长度表格中第二个字节是播放长度索引数。表格的最后两个数据，包括“0”，代表乐曲结束。

相关文件

MELODY.MTP 是一个需要下载的文件。在菜单 File/Open 如图 1-7 中浏览并且指定此文件。**MELODY.ASM** 是范例程序汇编语言源代码。**MELODY1.C** 是 C 语言版本。

流程图




Play song X
end of song ?
Y
FINISH
N
Get the initial value of Timer 1
Get the duration value, save to T_COUNTER
Start the Timer 1, Timer 0 and the Buzzer
A note is completed ?
N
Y

第三章

修改范例程序 3

这一章帮助用户修改范例程序来适应其它的应用。在开始之前，必须先在 PC 上安装 HT-IDE3000。

建立范例程序的新项目

- 选择 Start/Program Group/HT-IDE3000 进入 HT-IDE3000
- 选择 Project/New 命令新建一个项目

  设置项目名称为 buzzer

  选择 HT48E50 作为母体

  按下[OK]按键
- 显示配置选项对话框，将配置设定为下一段中的表格内容
- 选择 Project/Edit 命令，将源文件加入项目

  浏览目录，仅选择 BUZZER.ASM（或者 BUZZER.C）

  按下[ADD]按键，加入项目

  按下[OK]按键，完成项目编辑
- 选择 Project/Build 命令，编译项目

  产生需要被下载的文件 BUZZER.MTP
- 然后按照第一章的“运行初学者工具范例程序”，下载新文件并运行

对于其他范例程序，只要修改一下项目名称，源文件和配置选项，然后编译得到相关的.MTP 文件。下面的表格列出了每个范例程序的配置选项。

设置 DIP 开关选择时钟来源

正确的设置 Start Kit 上的 DIP 开关，可以选择 MTP 单片机 HT48E50 的时钟来源。如果选择晶振，那么位置 1 和 4 的开关拨到 ON，位置 2 和 3 的开关拨到 OFF。如果选择 RC，那么位置 2 的开关拨到 ON，其余开关位置拨到 OFF。可变电阻插入 Start Kit 面板上的 VR2 位置。电容值是 470pF。

配置选项表格

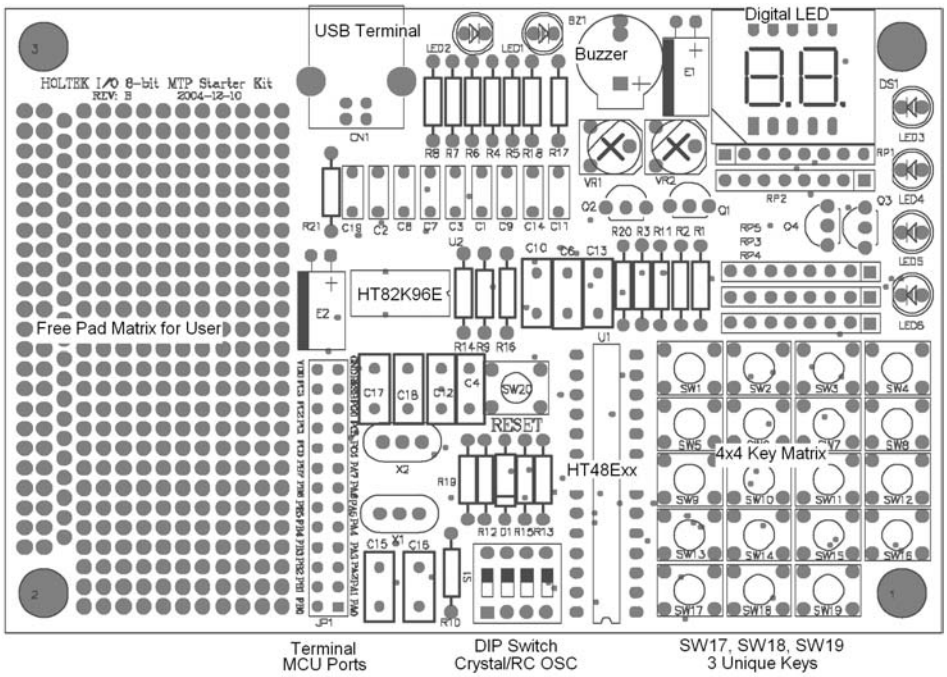
配置选项	BUZZER	Scan Key	Timer	EEPROM	Melody
System Frequency	4Mhz	4Mhz	4Mhz	4Mhz	4Mhz
Product	28 SKDIP-A	28 SKDIP-A	28 SKDIP-A	28 SKDIP-A	28 SKDIP-A
Wake-up PA0~PA7	Non-wake-up	Non-wake-up	Non-wake-up	Non-wake-up	Non-wake-up
Input Type PA	Schmitt Trigger	Schmitt Trigger	Schmitt Trigger	Schmitt Trigger	Schmitt Trigger
Pull-High PA	Pull-High	Pull-High	Pull-High	Pull-High	Pull-High
Pull-High PB	Pull-High	Pull-High	Pull-High	Pull-High	Pull-High
Pull-High PC	Pull-High	Pull-High	Pull-High	Pull-High	Pull-High
Pull-High PD	Pull-High	Pull-High	Pull-High	Pull-High	Pull-High
Pull-High PG	Pull-High	Pull-High	Pull-High	Pull-High	Pull-High
OSC	Crystal	Crystal	Crystal	Crystal	Crystal
WDT	Disable	Disable	Disable	Disable	Disable
CLR WDT	One Clear Instruction	One Clear Instruction	One Clear Instruction	One Clear Instruction	One Clear Instruction
WDT Clock Source	T1(System Clock/4)	T1(System Clock/4)	T1(System Clock/4)	T1(System Clock/4)	T1(System Clock/4)
PB01/Buzzer	BZ/$\overline{BZ}$	PB0/PB1	PB0/PB1	PB0/PB1	BZ/$\overline{BZ}$
LVR	Disable	Disable	Disable	Disable	Disable
LVR Voltage	3.0V	3.0V	3.0V	3.0V	3.0V
BZ_Source	Timer 0	Timer 0	Timer 0	Timer 0	Timer 1

第四章

目标板

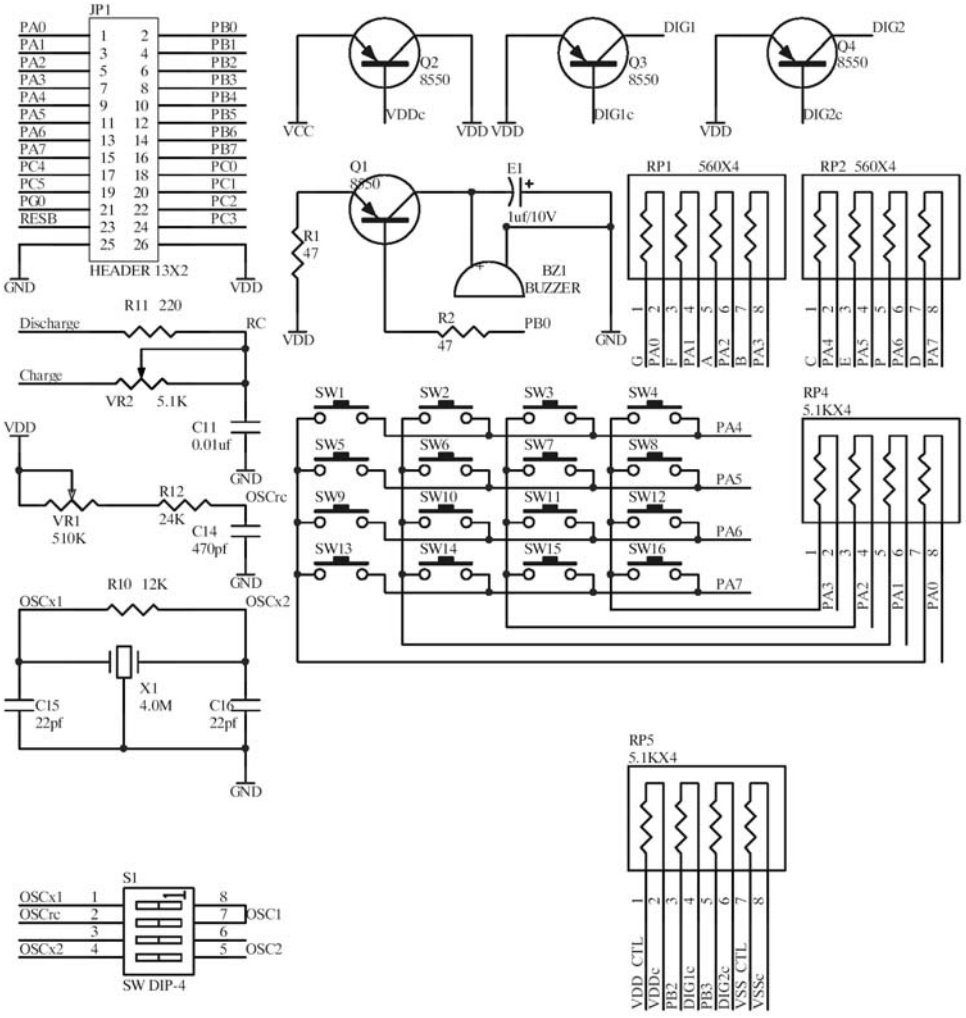
4

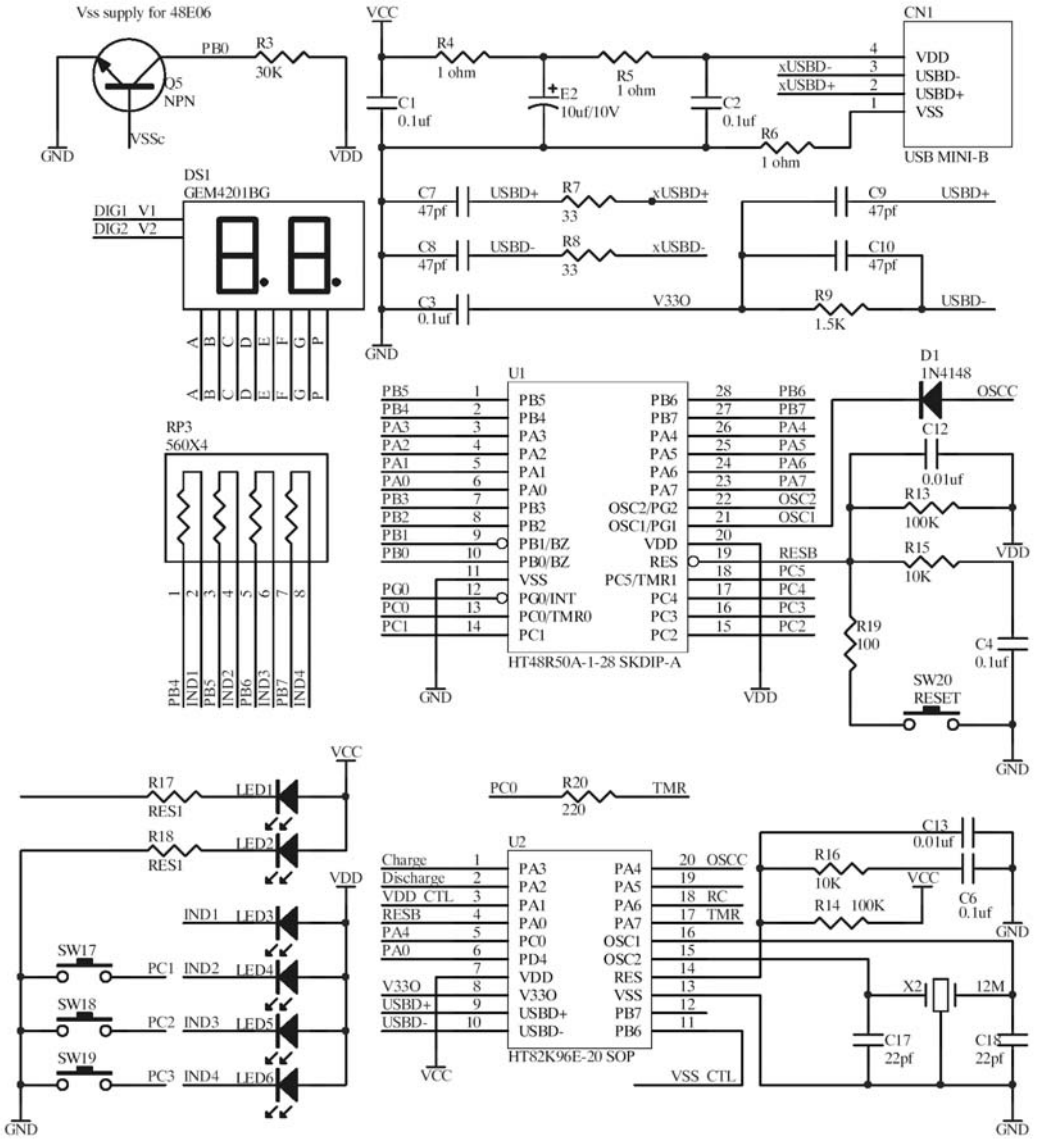
下图是 MTP 初学者工具印刷板布线图。HT48RXX 可以插入 28 SKDIP-A 封装的 HT48E50 单片机。运行正确的范例程序可做示范。



MTP 单片机的时钟来源 DIP 开关选项表格如下：

	DIP 开关位置			
时钟来源	1	2	3	4
晶振	On	Off	Off	On
RC 振荡	Off	On	Off	Off




Vss supply for 48E06
PB0
R3
30K
Q5
NPN
VSSc
GND
VDD
VCC
R4
1 ohm
C1
0.1uf
E2
10uf/10V
R5
1 ohm
C2
0.1uf
R6
1 ohm
CN1
VDD
USBD-
USBD+
VSS
USB MINI-B
xUSBD-
xUSBD+
DS1
GEM4201BG
DIG1 V1
DIG2 V2
C7
47pf
USBD+
R7
33
xUSBD+
C8
47pf
USBD-
R8
33
xUSBD-
C3
0.1uf
V33O
C9
47pf
C10
47pf
R9
1.5K
RP3
560X4
U1
HT48R50A-1-28 SKDIP-A
D1
1N4148
OSCC
C12
0.01uf
R13
100K
R15
10K
R19
100
C4
0.1uf
SW20
RESET
RESB
R17
RES1
R18
RES1
LED1
LED2
LED3
LED4
LED5
LED6
IND1
IND2
IND3
IND4
SW17
SW18
SW19
PC1
PC2
PC3
R20
220
TMR
U2
HT82K96E-20 SOP
Charge
Discharge
VDD_CTL
RESB
PA4
PA0
V33O
USBD+
USBD-
R16
10K
R14 100K
C13
0.01uf
C6
0.1uf
X2
12M
C17
22pf
C18
22pf
VSS_CTL

指令定义 5

ADC A, [m] Add Data Memory to ACC with Carry
指令说明 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回累加器。
功能表示 ACC ← ACC + [m] + C
影响标志位 OV , Z , AC , C

ADCM A, [m] Add ACC to Data Memory with Carry
指令说明 将指定数据存储器、累加器和进位标志位的内容相加后，把结果储存回指定数据存储器。
功能表示 [m] ← ACC + [m] + C
影响标志位 OV , Z , AC , C

ADD A, [m] Add Data Memory to ACC
指令说明 将指定数据存储器和累加器的内容相加后，把结果储存回累加器。
功能表示 ACC ← ACC + [m]
影响标志位 OV , Z , AC , C

ADD A, x Add immediate data to ACC
指令说明 将累加器和立即数的内容相加后，把结果储存回累加器。
功能表示 ACC ← ACC + x
影响标志位 OV , Z , AC , C

ADDM A, [m] Add ACC to Data Memory
指令说明 将指定数据存储器和累加器的内容相加后，把结果储存回指定数据存储器。
功能表示 [m] ← ACC + [m]
影响标志位 OV , Z , AC , C

AND A, [m] Logical AND Data Memory to ACC
指令说明 将存在累加器和指定数据存储器中的数据作 AND 的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“AND”[m]
影响标志位 Z

AND A, x Logical AND immediate data to ACC
指令说明 将存在累加器中的数据和立即数作 AND 的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“AND”x
影响标志位 Z

ANDM A, [m] Logical AND ACC to Data Memory
指令说明 将存在指定数据存储器和累加器中的数据作 AND 的运算，然后把结果储存回数据存储器。
功能表示 [m] ← ACC“AND”[m]
影响标志位 Z

CALL addr Subroutine call
指令说明 无条件地调用指定地址的子程序，此时程序计数器先加 1 获得下一个要执行的指令地址并压入堆栈，接着载入指定地址并从新地址继续执行程序，由于此指令需要额外的运算，所以为一个 2 周期的指令。
功能表示 Stack ← Program Counter + 1
Program Counter ← addr
影响标志位 None

CLR [m] Clear Data Memory
指令说明 指定数据存储器中的每一位均清除为 0。
功能表示 [m] ← 00H
影响标志位 None

CLR [m].i Clear bit of Data Memory
指令说明 指定数据存储器中的 i 位清除为 0。
功能表示 [m].i ← 0
影响标志位 None

CLR WDT Clear Watchdog Timer
指令说明 将 TO、PDF 标志位和 WDT 全都清零。
功能表示 WDT cleared
TO ← 0
PDF ← 0
影响标志位 TO , PDF

CLR WDT1 Pre-clear Watchdog Timer
指令说明 将 TO、PDF 标志位和 WDT 全都清零，请注意此指令要结合 CLR WDT2 一起动作且必须交替执行才有作用，重复执行此项指令而没有与 CLR WDT2 交替执行将无任何作用。
功能表示 WDT cleared
TO ← 0
PDF ← 0
影响标志位 TO , PDF

CLR WDT2 Pre-clear Watchdog Timer
指令说明 将 TO、PDF 标志位和 WDT 全都清零，请注意此指令要结合 CLR WDT1 一起动作且必须交替执行才有作用，重复执行此项指令而没有与 CLR WDT1 交替执行将无任何作用。
功能表示 WDT cleared
TO ← 0
PDF ← 0
影响标志位 TO , PDF

CPL [m] Complement Data Memory
指令说明 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1。
功能表示 [m] ← $\overline{[m]}$
影响标志位 Z

CPLA [m] Complement Data Memory with result in ACC
指令说明 将指定数据存储器中的每一位取逻辑反，相当于从 1 变 0 或 0 变 1，而结果被储存回累加器且数据存储器中的内容不变。
功能表示 ACC ← $\overline{[m]}$
影响标志位 Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory
指令说明 将存在累加器中的内容数值转换为 BCD（二进制转成十进制）数值，如果低 4 位大于 9 或 AC 标志位被置位，则在低 4 位加上一个 6，不然低 4 位的内容不变，如果高 4 位大于 9 或 C 标志位被置位，则在高 4 位加上一个 6，十进制的转换主要是依照累加器和标志位状况，分别加上 00H、06H、60H 或 66H，只有 C 标志位也许会被此指令影响，它会指出原始 BCD 数是否大于 100，并可以进行双精度十进制数相加。
功能表示 [m] ← ACC + 00H 或
[m] ← ACC + 06H 或
[m] ← ACC + 60H 或
[m] ← ACC + 66H
影响标志位 C

DEC [m] Decrement Data Memory
指令说明 将在指定数据存储器内的数据减 1。
功能表示 [m] ← [m] – 1
影响标志位 Z

DECA [m] Decrement Data Memory with result in ACC
指令说明 将在指定数据存储器内的数据减 1，把结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC ← [m] – 1
影响标志位 Z

HALT Enter power down mode
指令说明 此指令停止程序的执行并且关闭系统时钟，但数据存储器和寄存器的内容仍被保留，WDT 和预分频器(Prescaler)被清零，暂停标志位 PDF 被置位且 WDT 溢出标志位 TO 被清零。
功能表示 TO ← 0
PDF ← 1
影响标志位 TO , PDF

INC [m] Increment Data Memory
指令说明 将指定数据存储器内的数据加 1。
功能表示 [m] ← [m] + 1
影响标志位 Z

INCA [m] Increment Data Memory with result in ACC
指令说明 将指定数据存储器内的数据加 1，把结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC ← [m] + 1
影响标志位 Z

JMP addr Jump unconditionally
指令说明 程序计数器的内容被指定地址所取代，程序由新地址继续执行，当新地址被加载入时，必须插入一个空指令周期，所以此指令为 2 个周期的指令
功能表示 Program Counter ← addr
影响标志位 None

MOV A, [m] Move Data Memory to ACC
指令说明 将指定数据存储器的内容复制到累加器中。
功能表示 ACC ← [m]
影响标志位 None

MOV A, x Move immediate data to ACC
指令说明 将立即数载入至累加器中。
功能表示 ACC ← x
影响标志位 None

MOV [m], A Move ACC to Data Memory
指令说明 将累加器的内容复制到指定数据存储器。
功能表示 [m] ← ACC
影响标志位 None

NOP No operation
指令说明 空操作，接下来顺序执行下一条指令。
功能表示 No operation
影响标志位 None

OR A, [m] Logical OR Data Memory to ACC
指令说明 将存在累加器和指定数据存储器中的数据作 OR 的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“OR”[m]
影响标志位 Z

OR A, x Logical OR immediate data to ACC
指令说明 将存在累加器中的数据和立即数作 OR 的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“OR”x
影响标志位 Z

ORM A, [m] Logical OR ACC to Data Memory
指令说明 将存在指定数据存储器和累加器中的数据作 OR 的运算，然后把结果储存回数据存储器。
功能表示 [m] ← ACC“OR”[m]
影响标志位 Z

RET Return from subroutine
指令说明 将堆栈区的数据取回至程序计数器，程序由取回的地址继续执行。
功能表示 Program Counter ← Stack
影响标志位 None

RET A, x Return from subroutine and load immediate data to ACC
指令说明 将堆栈区的数据取回至程序计数器且累加器载入立即数，程序由取回的地址继续执行。
功能表示 Program Counter ← Stack
ACC ← x
影响标志位 None

RETI Return from interrupt
指令说明 将堆栈区的数据取回至程序计数器且中断功能通过 EMI 位重新被使能，EMI 是控制中断使能的主中断位(寄存器 INTC 的第 0 位)，如果在执行 RETI 指令之前还有中断未被响应，则这个中断将在返回主程序之前被响应。
功能表示 Program Counter ← Stack
EMI ← 1
影响标志位 None

RL [m] Rotate Data Memory left
指令说明 将指定数据存储器的内容向左移 1 个位，且第 7 位移回第 0 位。
功能表示 [m].(i+1) ← [m].i ; (i = 0~6)
[m].0 ← [m].7
影响标志位 None

RLA [m] Rotate Data Memory left with result in ACC
指令说明 将指定数据存储器的内容向左移 1 个位，且第 7 位移回第 0 位，而移位的结果储存回累加器且数据存储器中的内容不变。
功能表示 ACC.(i+1) ← [m].i ; (i = 0~6)
ACC.0 ← [m].7
影响标志位 None

RLC [m] Rotate Data Memory Left through Carry
指令说明 将指定数据存储器的内容连同进位标志位向左移 1 个位，第 7 位取代进位位且原本的进位标志位移至第 0 位。
功能表示 [m].(i+1) ← [m].i ; (i = 0~6)
[m].0 ← C
C ← [m].7
影响标志位 C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

指令说明 将指定数据存储器的内容连同进位标志位向左移 1 个位，第 7 位取代进位位且原本的进位标志位移至第 0 位，而移位的结果储存回累加器且数据存储器中的内容不变。

功能表示 ACC.(i+1) ← [m].i ; (i = 0~6)
ACC.0 ← C
C ← [m].7

影响标志位 C

RR [m] Rotate Data Memory right

指令说明 将指定数据存储器的内容向右移 1 个位，且第 0 位移回第 7 位。

功能表示 [m].i ← [m].(i+1) ; (i = 0~6)
[m].7 ← [m].0

影响标志位 None

RRA [m] Rotate Data Memory right with result in ACC

指令说明 将指定数据存储器的内容向右移 1 个位，且第 0 位移回第 7 位，而移位的结果储存回累加器且数据存储器中的内容不变。

功能表示 ACC.i ← [m].(i+1) ; (i = 0~6)
ACC.7 ← [m].0

影响标志位 None

RRC [m] Rotate Data Memory right through Carry

指令说明 将指定数据存储器的内容连同进位标志位向右移 1 个位，第 0 位取代进位位且原本的进位标志位移至第 7 位。

功能表示 [m].i ← [m].(i+1) ; (i = 0~6)
[m].7 ← C
C ← [m].0

影响标志位 C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

指令说明 将指定数据存储器的内容连同进位标志位向右移 1 个位，第 0 位取代进位位且原本的进位标志位移至第 7 位，而移位的结果储存回累加器且数据存储器中的内容不变。

功能表示 ACC.i ← [m].(i+1) ; (i = 0~6)
ACC.7 ← C
C ← [m].0

影响标志位 C

SBC A, [m] Subtract Data Memory from ACC with Carry

指令说明 将累加器中的数据与指定数据存储器内容和进位标志位的反相减，把结果储存回累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

功能表示 ACC ← ACC – [m] – $\overline{C}$

影响标志位 OV , Z , AC , C

SBCM A, [m] Subtract Data Memory from ACC with Carry and result in Data Memory

指令说明 将累加器中的数据与指定数据存储器内容和进位标志位的反相减，把结果储存回数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。

功能表示 [m] ← ACC – [m] – $\overline{C}$

影响标志位 OV , Z , AC , C

SDZ [m] Skip if Decrement Data Memory is 0

指令说明 将指定数据存储器的内容先减去 1 后，如果结果为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下面的指令。

功能表示 [m] ← [m] – 1
Skip if [m] = 0

影响标志位 None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC
指令说明 将指定数据存储器的内容先减去 1 后，如果结果为 0，则程序计数器再加 1 跳过下一条指令，此结果会被储存回累加器且指定数据存储器中的内容不变，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下面的指令。
功能表示 ACC ← [m] – 1
Skip if ACC = 0
影响标志位 None

SET [m] Set Data Memory
指令说明 将指定数据存储器的每一个位置位为 1。
功能表示 [m] ← FFH
影响标志位 None

SET [m].i Set bit of Data Memory
指令说明 将指定数据存储器的第 i 位置位为 1。
功能表示 [m].i ← 1
影响标志位 None

SIZ [m] Skip if increment Data Memory is 0
指令说明 将指定数据存储器的内容先加上 1 后，如果结果为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下面的指令。
功能表示 [m] ← [m] + 1
Skip if [m] = 0
影响标志位 None

SIZA [m] Skip if increment Data Memory is zero with result in ACC
指令说明 将指定数据存储器的内容先加上 1 后，如果结果为 0，则程序计数器再加 1 跳过下一条指令，此结果会被储存回累加器且指定数据存储器中的内容不变，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，则程序继续执行下面的指令。
功能表示 ACC ← [m] + 1
Skip if ACC = 0
影响标志位 None

SNZ [m].i Skip if bit i of Data Memory is not 0
指令说明 如果指定数据存储器的第 i 位不为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，程序继续执行下面的指令。
功能表示 Skip if [m].i ≠ 0
影响标志位 None

SUB A, [m] Subtract Data Memory from ACC
指令说明 将累加器中内容减去指定数据存储器的数据，把结果储存回累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – [m]
影响标志位 OV , Z , AC , C

SUBM A, [m] Subtract Data Memory from ACC with result in Data Memory
指令说明 将累加器中内容减去指定数据存储器的数据，把结果储存回数据存储器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 [m] ← ACC – [m]
影响标志位 OV , Z , AC , C

SUB A, x Subtract immediate Data from ACC
指令说明 将累加器中内容减去立即数，把结果储存回累加器。如果结果为负，C 标志位清除为 0，反之结果为正或 0，C 标志位设置为 1。
功能表示 ACC ← ACC – x
影响标志位 OV , Z , AC , C

SWAP [m] Swap nibbles of Data Memory
指令说明 将指定数据存储器的低 4 位与高 4 位互相交换。
功能表示 [m].3 ~ [m].0 ↔ [m].7 ~ [m].4
影响标志位 None

SWAPA [m] Swap nibbles of Data Memory with result in ACC
指令说明 将指定数据存储器的低 4 位与高 4 位互相交换，然后把结果储存回累加器且数据存储器的内容不变。
功能表示 ACC.3 ~ ACC.0 ← [m].7 ~ [m].4
ACC.7 ~ ACC.4 ← [m].3 ~ [m].0
影响标志位 None

SZ [m] Skip if Data Memory is 0
指令说明 如果指定数据存储器的内容为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，程序继续执行下面的指令。
功能表示 Skip if [m] = 0
影响标志位 None

SZA [m] Skip if Data Memory is 0 with data movement to ACC
指令说明 将指定数据存储器的内容复制到累加器，如果值为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，程序继续执行下面的指令。
功能表示 ACC ← [m]
Skip if [m] = 0
影响标志位 None

SZ [m].i Skip if bit i of Data Memory is 0
指令说明 如果指定数据存储器第 i 位为 0，则程序计数器再加 1 跳过下一条指令，由于取得下一指令时会要求插入一个空指令周期，所以此指令为 2 个周期的指令。如果结果不为 0，程序继续执行下面的指令。
功能表示 Skip if [m].i = 0
影响标志位 None

TABRDC [m] Read table (current page) to TBLH and Data Memory
指令说明 将表格指针 TBLP 所指的程序代码低字节(当前页)移至指定数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码(低字节)
TBLH ← 程序代码(高字节)
影响标志位 None

TABRDL [m] Read table (last page) to TBLH and Data Memory
指令说明 将表格指针 TBLP 所指的程序代码低字节(最后一页)移至指定数据存储器且将高字节移至 TBLH。
功能表示 [m] ← 程序代码(低字节)
TBLH ← 程序代码(高字节)
影响标志位 None

XOR A, [m] Logical XOR Data Memory to ACC
指令说明 将存在累加器和指定数据存储器中的数据作 XOR 的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“XOR”[m]
影响标志位 Z

XORM A, [m] Logical XOR ACC to Data Memory
指令说明 将存在指定数据存储器和累加器中的数据作 XOR 的运算，然后把结果储存回数据存储器。
功能表示 [m] ← ACC“XOR”[m]
影响标志位 Z

XOR A, x Logical XOR immediate data to ACC
指令说明 将存在累加器中的数据和立即数作 XOR 的运算，然后把结果储存回累加器。
功能表示 ACC ← ACC“XOR”x
影响标志位 Z

盛群半导体股份有限公司（总公司）
台湾新竹市科学工业园区研新二路 3 号
电话: 886-3-563-1999
传真: 886-3-563-1189

盛群半导体股份有限公司（台北业务处）
台湾台北市南港区园区街 3 之 2 号 4 楼之 2
电话: 886-2-2655-7070
传真: 886-2-2655-7373
传真: 886-2-2655-7383 (International sales hotline)

盛扬半导体有限公司（上海业务处）
上海宜山路 889 号 2 号楼 7 楼 200233
电话: 021-6485-5560
传真: 021-6485-0313

盛扬半导体有限公司（深圳业务处）
深圳市深南中路赛格广场 43 楼518031
电话: 0755-8346-5589
传真: 0755-8346-5590

盛扬半导体有限公司（北京业务处）
北京市西城区宣武门西大街甲 129 号金隅大厦 1721 室 100031
电话: 010-6641-0030，6441-7751，6441-7752
传真: 010-6641-0125

Holmate Semiconductor, Inc.（北美业务处）
46712 Fremont Blvd., Fremont, CA 94538
电话: 510-252-9880
传真: 510-252-9885

盛永半导体（深圳）有限公司
深圳市南山区高新技术产业园北区清华信息港 A 座 602 室
电话：0755-33639069
传真：0755-33639033

Sharing Success Through Excellence