XICOR RTC X12xx-EVM
评估板使用说明

VER 1.0 版

武汉・力源

2004 年 8 月

一、如何开始使用本评估板

XICOR RTC X12xx-EVM 评估套件旨在帮助用户对 XICOR 公司的 X12xx 系列实时时钟(RTC)器件进行功能评估和性能测试。

使用此套件时请先按第二节中方法进行软件安装，然后用所配的 RS232 串口电缆将评估板与 PC 机的串口连接，运行所安装和 PC 机软件 RTC.EXE 应用程序即可进行评估和测试。

二、软件安装说明

本软件为绿色软件，直接拷贝即可使用，不会改变机器的任何配置信息。

最低安装配置：INTEL 486，内存 16M，硬盘空余空间 10M，1 个有效的 RS-232 串行接口

可运行的平台：WINDOWS 98/NT/2000/XP

安装步骤：

1）在需安装的硬盘中新建目录，例如：C:\RTC

2）将安装光盘中根目录中文件拷贝到新建目录中，包括：

RTC.EXE	应用程序
IIC.TXT	IIC 示例源程序
AN.TXT	应用笔记文本文件
UM.TXT	使用说明文本文件

3）直接运行 RTC.EXE 应用程序，如果提示缺少运行时库文件，请将安装光盘中 DLLOCX 目录下的相应文件拷贝到新建目录或操作系统目录如“ C:\WINDOWS\SYSTERM32 ”。可能需要的文件包括：

MSVBVM60.DLL	VB 运行动态库
COMMCTL32.OCX	通用控件对象
MSCOMM32.OCX	COMM 控件对象
RICHTX32.OCX	RICHTEXT 控件对象

如果你没有 PDF 文件的浏览器请从光盘的 TOOLS 目录中安装 READER 中文 6.0 版，TOOLS 目录中还包括了一个 WORD 阅读器。如果希望在硬盘中能够浏览 RTC 芯片的数据手册或应用笔记，将安装光盘中 DOCS 目录下文件拷贝到硬盘目录中，例如：C:\RTC\DOCS。

光盘目录及文件说明：

```
|
|    README.TXT
|    RTC.EXE
|    IIC.TXT
|    AN.TXT
|    UM.TXT
|-----DLLOCX
|          |    MSVBVM60.DLL
|          |    COMMCTL32.OCX
|          |    MSCOMM32.OCX
|          |    RICHTX32.OCX
|-----DOCS
|          |    RTC 数据手册应用笔记（PDF 或 MTHML 文件）
|          |    XICOR RTC X12xx-EVM 评估板使用说明.DOC
|          |    RTC10SCH.PDF
|          |    RTC10BRD.PDF
|-----TRAIN
|          |    XICOR RTC 培训资料（PDF 或 PPT 文件）
|-----TOOLS
          |    ADBERDR60_CHS_FULL.EXE
          |    WD97VWR32.EXE
```

三、PC 机软件使用说明

（一）操作界面如图 1 所示

图 1　PC 软件操作界面

（二）使用步骤

1．使用 RS-232 串行电缆连接 PC 机串行口与评估板
2．运行 RTC.EXE 应用程序，程序自动探测评估板。如果找到，界面的状态条中会显示“ 成功连接评估板于串行口 x ”，如果没有找到，会提示“ 没有探测到评估板 ”，此时请检查连接是否正确以评估板板的配置，请参见关于评估板一节。检查后需使用命令菜单中的手动连接，选定串口后再次与评估板连接，直到连接成功。
3．在器件菜单选择与评估板上相一致的器件，使用命令菜单中的读取时间，读取评估板上 RTC 的时间。如果是第一次使用评估板或者 RTC 的主电和备用电池曾全部失效，应先使用命令菜单中的设置时间命令，这样才能启动 RTC 开始工作。
4．使用命令菜单中的状态/控制命令，可对 RTC 的所有状态或控制寄存器进行读或写操作。
5．使用功能菜单，可对当前选择的 RTC 器件的所有功能模块进行测试。
6．使用帮助菜单可获得简单帮助、应用笔记、IIC 源程序及版本信息。
7．选择界面下方的自动读取时间，将自动读取评估板的时间；选择自动监测报警或事件，可对 RTC 中的报警或事件进行实时监视，一旦有报警或事件发生，界面中的指示灯变亮，同时弹出警报对话框。
8．在界面中的当前 RTC 的资料可点击进行浏览。

四、评估板使用说明

（一）概要说明

XICOT RTC X12xx 评估板可用于对 XICOR 的时钟芯片 X12xx 系列进行测试和评估，支持器件包括 X1205、X1226、X1227、X1228、X1208、X1209，封装包括 SOIC、TSSOP、MSOP。

XICOR RTC X12xx 评估板上包括一颗 MCS-51 兼容的 MCU，RS-232 驱动器以及电源电路。评估板与 PC 机上软件配合使用，MCU 负责解释 PC 机发来的命令，执行相应操作再将结果返回给 PC 机。通信采用 19200bps，8 位数据，1 位起始位，1 位停止位，无奇偶校验方式。

（二）评估板原理框图及 PCB 顶层丝网图

原理框图如图 2 所示。

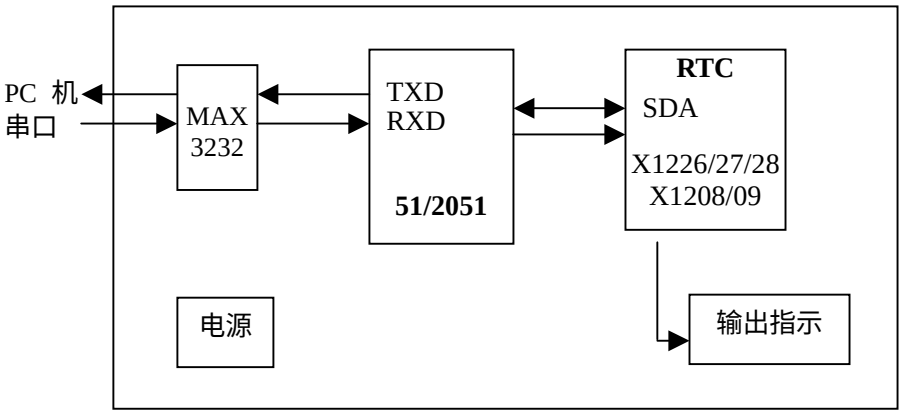

图 2　RTC 评估板框图

PCB 板顶层丝网图如图 3 所示。

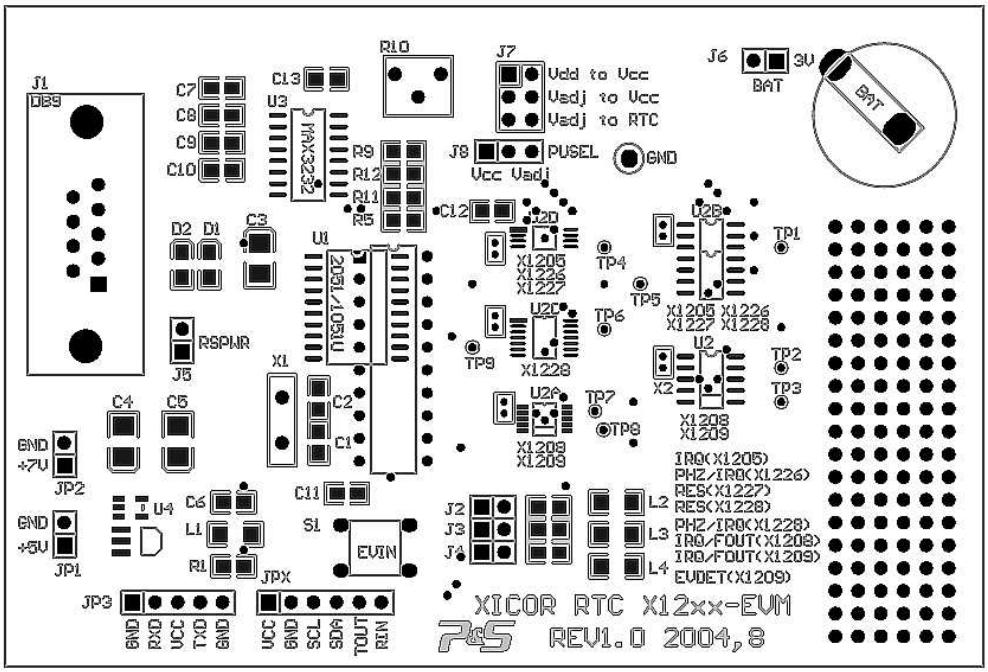

图 3　评估板 PCB 顶层丝网图

（三）使用说明

1．评估板供电选择

评估板可采用多种供电方式选择

- 从 PC 机串口获得电源，短路跳接 J5（*）
- 外部提供+5V 直流稳压电源，从 JP1 接入，不使用板上的稳压电路
- 外部提供+7~15V 直流电源，从 JP2 接入，使用板上的稳压电路
- 从 USB 口获取电源：从 JP3（USBP）接入，需使用 USB 专用电缆，可选配或自制，自制方法见本节（四）小节。

* 注意，在使用串口获取电源时，由于 PC 机的串口驱动能力的不同，获得的电压可能略低于 5V，因此部分 5V 的 RTC 器件可能会出现操作失败，此时请使用其他供电方式。

2．非 5V 应用及电压调节

为使用非 5V 系统电压的 RTC 器件，评估板的 MCU 和 232 设计为 3V/5V 兼容。通过 J7 的不同配置，可实现非 5V 电压和电压调节功能。

J7 的连接示意图如图 4 所示：

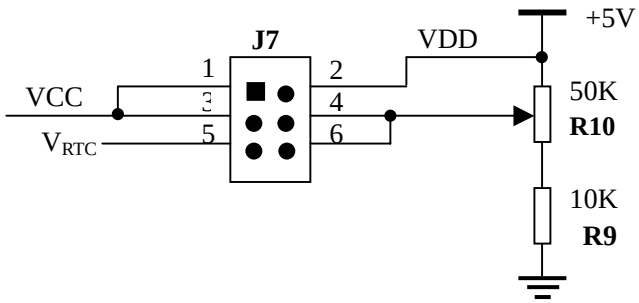

图 4　电源选择 J7 连接示意图

VDD：板上经稳压的+5V 电源

VCC：评估板的系统电源，MCU 和 232 驱动电路，不包括 RTC 器件

V_{RTC}：仅RTC器件的电源

J7 的连接方式：

- 短接 1-2：将板上稳压+5V 电源接入评估板的系统电源（MCU 和 232 器件）
- 短接 3-4：将板上稳压+5V 电源经电位器分压后接入评估板的系统电源（MCU 和 232 器件）
- 短接 5-6：将板上稳压+5V 电源经电位器分压后接到 RTC 的电源
- 短接 3-5：将板上稳压+5V 电源直接引入到 RTC 的电源

3．使用跳接线短接 J6，使用板上的电池作为 RTC 的后备电源

4．指示说明

RTC 的输出信号 IRQ、PHZ、RES、EVDET，在评估板上以三个发光二极管进行指示。所有信号低电平有效，即输出为低时，二极管亮。详见表 1。

表 1　LED 指示说明

	X1205	X1226	X1227	X1228	X1208	X1209
L2	IRQ	PHZ/IRQ	RESET	RESET		
L3				PHZ/IRQ	IRQ/FOUT	IRQ/FOUT
L4						EVDET

跳接 J2，J3，J4 选择是否使用指示 L2，L3，L4。

5．RTC 芯片的安装

评估板上的 U2、U2A、U2B、U2C 四个位置可供 RTC 器件安装，但任何时候板上只能有一个 RTC 器件。详见表 2。

表 2　RTC 器件安装说明

	X1205	X1226	X1227	X1228	X1208	X1209
U2					SOIC8	SOIC10
U2A					MSOP8	MSOP10
U2B	SOIC8*	SOIC8*	SOIC8*	SOIC14		
U2C				TSSOP14		
U2D	TSSOP8	TSSOP8	TSSOP8			

注*：U2B 安装 SOIC 的 X1205/26/27 时 1 脚对应 4 号焊盘，即靠下安装；
其他器件 1 脚对应 1 号焊盘。

6．测试点说明

评估板上的测试点 TP1-TP9 用于用户对 RTC 器件的输出信号进行测量。详见表 3。

表 3　测试点指示说明

测试点	器件型号	信号	测试点	器件型号	信号
TP1	X1205	IRQ	TP2	X1208	IRQ/FOUT
	X1226	PHZ/IRQ		X1209	IRQ/FOUT
	X1227	RESET	TP5	X1228	PHZ/IRQ
	X1228	RESET	TP6	X1228	PHZ/IRQ
TP3	X1209	EVDET	TP7	X1208	IRQ/FOUT
TP4	X1205	IRQ		X1209	IRQ/FOUT
	X1226	PHZ/IRQ	TP8	X1209	EVDET
	X1227	RESET	TP9	X1228	RESET

（四）自制 USB 连接电缆

1. 所需材料：如了图 5。
   USB 标准延长线一根
   DB9/F（孔）插头一个
   254 标准 5 芯接插件一套

图 5　自制 USB 连接电缆所需材料

2. 制作步骤：
   1）将 USB 延长线在靠 PC 插头 40 厘米左右处剪断
   2）测量出 USB 线中电源正和负，通常红为正，黑为负
   3）打开 DB9 接头外壳，与 USB 插头照图 6 左部进行连接

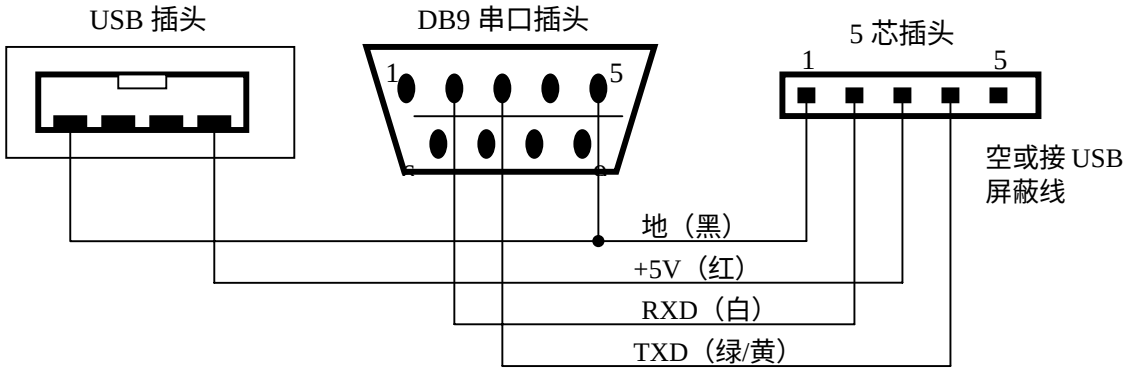

图 6　USB 连接电缆连接示意图

4）剪掉 USB 线的另一头，用 5 芯接头照图 6 右部方式连接，实物连法如图 7。

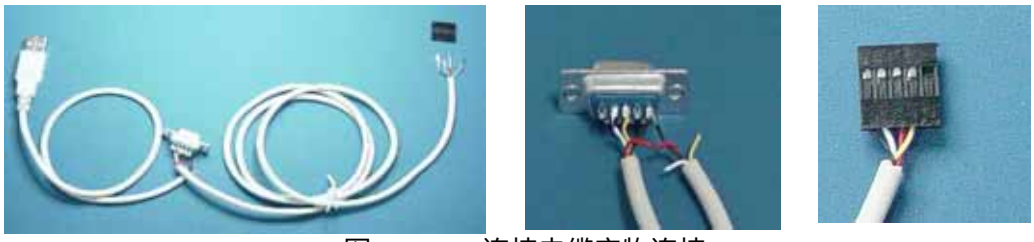

图 7　USB 连接电缆实物连接

5）将 DB9 插的外壳装上，制作好的样品见图 8。

图 8　USB 连接电缆样品

6）使用时 USB 头和 DB9 接 PC 机的 USB 口和串口，另一端 5 芯插头连到评估板的 JP3，5 芯插头方向应与板上的标识一致。

附录一　　RTC 评估板电路图

参见PDF文件：rtc10sch.pdf

附录二　X1226/27/28 与 8051 演示例程

1. 初始化启动 RTC

```
    lcall   ResetIIC      ；复位 RTC
    lcall   set_WEL       ；置 WEL
    lcall   set_RWEL        ；置 RWEL
    lcall   start_rtc     ；写 RTC 中的时间，启动 RTC
```

2. 读时间

```
    mov     dID,#6fH                    ；RTC 器件 ID '1101111'
    mov     addh,#0H                    ；高字节地址 00
    mov     addl,#30H                   ；RTC SRAM 低字节地址 30H ，仅 X1205/26/7/8
    mov     r_len,#08                   ；读 RTC 共 8 个字节
    mov     f_len,r_len                 ；置缓冲区长度为读缓冲区长度
    lcall   read_device
```

3. 写 WEL、RWEL、RTC 子程序

```
set_WEL:
    mov     dID,#6FH                    ； RTC 器件 ID '1101111'
    mov     addh,#0H        ；高字节地址，仅 X1205/26  /27/28，对 X1208/09 使用 FF 代替
    mov     addl,#3FH       ；低字节地址
    mov     R0,#f_buf           ；R0 指向缓冲区
    mov     w_len,#1                    ；写入的字节数
    mov     @R0,#02H                    ；写 SR, 置 WEL
    ajmp    write_device        ；调用写器件子程序

set_RWEL:
    mov     dID,#6FH                    ；RTC 器件 ID '1101111'
    mov     addh,#0H
    mov     addl,#3FH
    mov     R0,#f_buf
    mov     w_len,#1
    mov     @R0,#06H                    ； 写 SR, 置 WEL 和 RWEL
    ajmp    write_device

start_rtc:
    mov     dID,#6fH                    ；RTC 器件 ID '1101111'
    mov     addh,#0H
    mov     addl,#30H
    mov     R0,#f_buf
    mov     w_len,#8                    ；启动时间于：2004 年 7 月 1 日（四），11：22：33
    mov     @R0,#33H        ；秒
    inc     R0
    mov     @R0,#22H        ；分
    inc     R0
    mov     @R0,#11H        ；小时
    inc     R0
    mov     @R0,#01H        ；日
    inc     R0
```

```
        mov     @R0,#07H        ；月
        inc     R0
        mov     @R0,#04H        ；年
        inc     R0
        mov     @R0,#04H        ；星期
        inc     R0
        mov     @R0,#20H        ；世纪

        mov     R0,#f_buf
        ajmp    write_device
```

4. 读/写器件子程序

```
read_device:
        mov     R0,#f_buf       ；初始化读缓冲区
        mov     DPH,addh        ；高字节地址送 DPH
        mov     DPL,addl        ；低字节地址送 DPL
        lcall   RandomRead          ；调随机读子程序
        mov     @R0,A       ；保存读出结果
        inc       R0
        djnz    r_len,read_n        ；是多字节读
        ret
read_n:
        mov     R1,r_len
        lcall   SeqRead         ；调用连续读子程序
        ret

write_device:
        mov     R1,w_len
        mov     DPH,addh
        mov     DPL,addl
        lcall   ProgPage        ；调用连续写命令
        ret
```

5. IIC 通信子程序

```
;*******************************************************************************
;**
;**   DESCRIPTION:
;**
;**   This file contains general utility routines, written in the 80C51 assembly
;**   language, which are used to interface the 80C51 to standard two-wire(IIC)
;**   Serial Products. The interface between the 80C51 and IIC Device consists
;**   of a clock (SCL) and a bidirectional data line (SDA).
;**   The communication interface uses 2 pins from Port 1(P10 = SCL and P11 = SDA).
;**   The following table lists all the subroutines in this file with a brief
;**   description:
;**
;**             Start: Generate the start condition
;**             Stop: Generate the stop condition
;**             ResetIIC: Issues the appropriate commands to force device reset
;**             ProgPage: Transfer multible bytes from RAM buffer to IIC Device
```

```
;**             ProgByte: Transfer byte from RAM buffer to IIC Device
;**             SeqRead: Read multiple bytes, starting from current address pointer
;**             RandomRead: Read a byte from a specific memory location
;**             ACKPoll: Return when the write cycle completes.
;**             OutACK: Process the acknowledge output cycle
;**             GetACK: process the acknowledge from the slave device
;**
;**       NOTE 1. These rountines can run at 12.0MHz, While IIC Maxspeed is 400K
;**                  2. The IIC Device must based on 16-bits address, so the sequence:
;**
;**            START     Slave    Address    Address     Data       STOP
;**                  \XXXXXXXW AAAAAAAA AAAAAAAA DDDDDDDD /
;**                       ACK          ACK          ACK          ACK
;**
;*****************************************************************************
;*****************************************************************************
;*                                PROGRAM CONSTANTS
;*****************************************************************************
SDAbit      EQU      00H          ; PORT-1 BITS FUNCTIONING AS BIDIRECTIONAL
SCLbit      EQU      01H          ; SERIAL DATA (SDA) AND SERIAL CLOCK OUTPUT (SCL)

MaxDelay    EQU      0FFH         ; NUMBER OF TIMES TO CHECK ACKNOWLEDGE POLLING

DeviceID    EQU      6FH          ; DEVICE SLAVE ADDRESS
                                  ; X1208\1209 SLAVE ADDRESS: 1101111B

;*****************************************************************************
;*****************************************************************************
;** Name: SeqRead
;** Description: Read sequentially from the IIC Device.
;** Function: This subroutine extracts contents of the IIC Device and stores
;**             them into the specified RAM buffer. The total number of bytes to
;**             read should be provided along with the buffer address. This
;**             routine assumes that the address pointer has already been
;**             initialized using the RandomRead routine.
;**
;** Calls:              Start, SlavAddr, InByte, StopRead
;** Input:              R0 = RAM Buffer Base Address
;**                       R1 = Number of bytes to read
;** Output:             None
;** Register Usage:     A, R0, R1
;*****************************************************************************
        ORG      0300H                    ; IIC.ASM MAPPED IN: 300~3D2H

SeqRead:
      acall    Start                    ; START
      setb     c                          ; [C=1] READ OPERATION BIT
      acall    SlavAddr                 ; SEND THE SLAVE ADDRESS BYTE
SeqReadLoop:
      acall    InByte                   ; START READING FROM THE CURRENT ADDRESS
      mov      @R0,A                        ; TOTAL NUMBER OF BYTES TO READ OUT OF
      inc      R0                         ; IIC Device
      djnz     R1,SeqReadNxt
```

```
      ajmp      StopRead                    ; END OF READ OPERATION
SeqReadNxt:
      acall     OutACK                      ; SEND AN ACKNOWLEDGE TO THE DEVICE
      ajmp      SeqReadLoop

;*****************************************************************************
;** Name: RandomRead
;** Description: Reads content of the IIC Device at a specific location.
;** Function: This subroutine sends out the command to read the content of a
;**                  memory location specified in the DPTR.
;**
;** Calls:                  Start, InByte, SlavAddr, OutByte StopRead
;** Input:                  DPTR = Address of the byte
;** Output:                 A = Read value
;** Register Usage:         A
;*****************************************************************************
RandomRead:
      acall     Start                     ; START
      clr       C                           ; [C=0] WRITE OPERATION BIT
      acall     SlavAddr                  ; SEND THE SLAVE ADDRESS BYTE
      mov       A,DPH                       ; LOAD THE UPPER BYTE OF THE ADDRESS
        cjne      A,#0FFH,Rand16
        ajmp      Rand8
Rand16:
        acall     OutByte                   ; ADDRESS SHIFT OUT TO THE DEVICE
Rand8:
      mov       A,DPL                       ; LOAD THE LOWER BYTE OF THE PAGE
      acall     OutByte                   ; ADDRESS SHIFT OUT TO THE DEVICE
      acall     Start                     ; START
      setb      C                           ; [C=1] READ OPERATION BIT
      acall     SlavAddr                  ; SEND THE SLAVE ADDRESS BYTE
      acall     InByte                    ; SHIFT IN A BYTE FROM THE DEVICE
      ajmp      StopRead                    ; END OPERATION

;*****************************************************************************
;** Name: StopRead
;** Description: Terminate read operation.
;** Function: This subroutine sends out the command to end reading content of a
;**                  IIC Device.
;** Calls:                  ClockPulse, Stop
;** Input:                  None
;** Output:                 None
;** Register Usage:         None
;*****************************************************************************
StopRead:
      setb      P1.SDAbit                 ; MAKE SURE THAT THE DATA LINE IS HIGH
      acall     ClockPulse
      jmp       Stop                      ; END OPERATION

;*****************************************************************************
;** Name: ProgPage
;** Description: Update a page of the Serial Memory.
;** Function: This subroutine transfers the contents of the given buffer to the
```

```
;**                 Serial Memory. The caller program must supply the page number
;**                 of the Serial Memory to update and the base address of the
;**                 RAM buffer.
;** Calls:                  Start, SlavAddr, OutByte, Stop
;** Input:                  R0 = RAM Buffer Base Address, DPTR = Address of programm
;**                           R1 = Number of bytes to write
;** Output:                 None
;** Register Usage:       A, R0, R1
;*********************************************************************************

ProgPage:
      acall     Start                       ; START
      clr       C                             ; [C=0] WRITE OPERATION BIT
      acall     SlavAddr                    ; SEND THE SLAVE ADDRESS BYTE
      mov       A,DPH                         ; LOAD THE HIGHER BYTE OF ADDRESS
        cjne      A,#0FFH,ProgP16
        ajmp      ProgP8
ProgP16:
        acall     OutByte
ProgP8:
      mov       A,DPL                         ; LOAD THE LOWER BYTE OF ADDRESS
      acall     OutByte                     ; AND SHIFT OUT TO THE DEVICE
ProgPageNxt:
      mov       A,@R0                         ; TO THE Serial MEMORY
      acall     OutByte                     ; R0 SHOULD BE POINTING TO THE BUFFER
      inc       R0                            ; TOTAL NUMBER OF BYTES TRANSFERRED
      djnz      R1,ProgPageNxt              ; TO THE Serial SHOULD NOT EXCEED
                                    ; THE PAGE SIZE
      ajmp      Stop                          ; END OF THE OPERATION

;*********************************************************************************
;** Name: ProgByte
;** Description: Write a byte to IIC Device.
;** Function: This subroutine transfers the contents of Acc to the IIC
;**                 Device. The upper byte of the word address    is contained in
;**                 (DPH) and the lower byte is contained in (DPL).    A two byte
;**                 address is required for both page and byte write commands.
;**
;** Calls:                  Start, SlavAddr, Outbyte, Stop
;** Input:                  Acc = byte to be written, DPTR = Address of programm
;** Output:                 None
;** Register Usage:       B
;*********************************************************************************
ProgByte:
      xch       A,B                         ; SAVE DATA BYTE
      acall     Start                       ; START
      clr       c                             ; [C=0] WRITEOPERATION BIT
      acall     SlavAddr                    ; SEND THE SLAVE ADDRESS BYTE
      mov       A,DPH                         ; LOAD THE UPPER BYTE OF THE WORD ADDRESS
        cjne      A,#0FFH,ProgB16
        ajmp      ProgB8
ProgB16:
        acall     OutByte                     ; SEND OUT THE HIGHER BYTE OF THE ADDRESS
```

```
ProgB8:
      mov       A,DPL                       ; LOAD THE LOWER BYTE OF THE WORD ADDRESS
      acall     OutByte                     ; SEND OUT THE LOWER BYTE OF THE ADDRESS
      xch       A,B                         ; LOAD THE DATA TO BE SENT IN THE ACC
      acall     OutByte                     ; SEND OUT THE DATA TO THE SERIAL MEMORY
      ajmp      Stop                        ; END OF THE OPERATION

;*****************************************************************************
;** Name: SlavAddr
;** Description: Build the slave address for the IIC Device.
;** Function: This subroutine concatenates the bit fields for Device ID,
;**                  the high address bits and the command bit. The resultant
;**                  byte is then transmitted to the IIC Device.
;** Calls:                   OutByte
;** Input:                   C = COMMAND BIT (=0 WRITE, =1 READ)
;** Output:                  None
;** Register Usage:        A
;*****************************************************************************
SlavAddr:
      mov       A,dID                 ; IF IS FIXED DEVICE USE:'#DeviceID'
      rlc       A                           ; MERGE THE COMMAND BIT
      ajmp      OutByte                     ; SEND THE SLAVE ADDRESS

;*****************************************************************************
;** Name: OutByte
;** Description: Sends a byte to the IIC Device
;** Function: This subroutine shifts out a byte, MSB first, through the
;**                  assigned SDA/SCL lines on port 1.
;** Calls:                   ClockPulse, GetACK
;** Input:                   A = Byte to be sent
;** Return Value:          None
;** Register Usage:        A
;*****************************************************************************
OutByte:
      setb      C
OutByteLoop:
      rlc       A                           ; SHIFT OUT THE BYTE, MSB FIRST
      jnc       OutByte0
      setb      P1.SDAbit
      ajmp      OutByte1
OutByte0:
      clr       P1.SDAbit
OutByte1:
      acall     ClockPulse                  ; CLOCK THE DATA INTO THE IIC Device
      cjne      A,#10000000b,OutByteNxt ; SKIP IF NOT DONE
      jmp       GetACK                      ; CHECK FOR AN ACK FROM THE DEVICE
OutByteNxt:
      clr       C                           ; LOOP IF ALL THE BITS HAVE
      ajmp      OutByteLoop                 ; NOT BEEN SHIFTED OUT

;*****************************************************************************
;** Name: InByte
;** Description: Shifts in a byte from the IIC Device
```

```
;** Function: This subroutine shifts in a byte, MSB first, through the
;**                   assigned SDA/SCL lines on port 1. After the byte is received
;**                   an ACK bit is sent to the IIC Device.
;** Calls:                    ClockPulse
;** Input:                    None
;** Return Value:          A = Received byte
;** Register Usage:       A
;*********************************************************************************
InByte:
      mov       A,#00000001b
      setb      P1.SDAbit                    ; SDA LINE FORCED HIGH
InByteNxt:
      acall     ClockPulse                   ; CLOCK THE Serial MEMORY AND SHIFT
      rlc       A                             ; INTO ACC. THE LOGIC LEVEL ON THE SDA
      jnc       InByteNxt                    ; LINE. THE DEVICE OUTPUTS DATA ON SDA
      ret                                    ; MSB FIRST

;*********************************************************************************
;** Name: ClockPulse
;** Description: Generate a clock pulse
;** Function: This subroutine forces a high-low transition on the assigned SCL
;**                   pin of port 1. It also samples and returns the SDA line state
;**                   during high clock period.
;** Calls:                    None
;** Input:                    None
;** Return Value:          C = SDA line status
;** Register Usage:       None
;*********************************************************************************
ClockPulse:
        nop
      setb      P1.SCLbit                    ; BASED ON A 12MHz SYSTEM CRYSTAL THE
      nop                                     ; BUS CYCLE TIME IS 1 MICROSEC.
      nop
        nop
      clr       C                             ;
      jnb       P1.SDAbit,ClockPulseLo    ;
      setb      C
ClockPulseLo:
      clr       P1.SCLbit                    ; LOWER THE CLOCK LINE
      ret

;*********************************************************************************
;** Name: OutACK
;** Description: Send out an ACK bit to the IIC Device
;** Function: This subroutine clocks an ACK bit to the IIC Device.
;**                   The ACK cycle acknowledges a properly received data by lowering
;**                   the SDA line during this period (9th clock cycle of a received
;**                   byte).
;** Calls:                    ClockPulse
;** Input:                    None
;** Return Value:          None
;** Register Usage:       None
;*********************************************************************************
```

```
OutACK:
      clr       P1.SDAbit                    ; MAKE SURE THAT THE DATA LINE IS LOW     ...
      jmp       ClockPulse                   ; CLOCK OUT THE ACK CYCLE

;*************************************************************************
;** Name: GetACK
;** Description: Clock the IIC Device for ACK cycle
;** Function: This subroutine returns the sampled logic level on the SDA during
;**                  high clock cycle. The IIC Device holds the SDA line HIGH if
;**                  it does not receive the correct number of clocks or it's stuck in
;**                  a previously initiated write command.
;** Calls:                  ClockPulse
;** Input:                  None
;** Return Value:           C = ACKnowledge bit
;** Register Usage:         None
;*************************************************************************
GetACK:
      setb      P1.SDAbit                    ; FORCE SDA LINE HIGH
      acall     ClockPulse                   ; GENERATE ONE CLOCK PULSE
      ret

;*************************************************************************
;** Name: ACKPoll
;** Description: Wait for an ACK from the IIC Device
;** Function: This subroutine monitors the IIC Device response
;**                  following a dummy write command. The program returns when it
;**                  either receives an ACK bit or the maximum number of tries is
;**                  reached.
;** Calls:                  Start, SlavAddr, Stop
;** Input:                  None
;** Return Value:           C = ACKnowledge bit [=0 ACK ,=1 No ACK was received]
;** Register Usage:         A, R1
;*************************************************************************
ACKPoll:
      mov       R1,#MaxDelay                 ; LOAD MAX NO. OF ACK POLLING CYCLE
ACKPollnxt:
      acall     Start                        ; START THE ACK POLL CYCLE AND
      clr       C                            ; [C=0] WRITE OPERATION BIT
      acall     SlavAddr                     ; SEND THE SLAVE ADDRESS. THEN
                                             ; MONITOR THE SDA LINE FOR AN ACK FROM
                                             ; THE IIC Device. TERMINATE THE
      acall     Stop                         ; OPERATION BY A STOP CONDITION.
      jnc       ACKPollExit                  ; EXIT IF THE ACK WAS RECEIVED
        djnz      R1,ACKPollnxt              ; LOOP WHILE THE MAXIMUM NO. OF CYCLES
                                             ; HAVE NOT EXPIRED. ELSE RETURN WITH C=1
ACKPollExit:
      ret

;*************************************************************************
;** Name: Start
;** Description: Send a start command to the IIC Device
;** Function: This subroutine generates a start condition on the bus. The start
;**                  condition is defined as a high-low transition on the SDA
```

```
;**                 line while the SCL is high. The start is used at the beginning
;**                 of all transactions.
;** Calls:                 None
;** Input:                 None
;** Return Value:          None
;** Register Usage:        None
;*************************************************************************
Start:
      setb      P1.SDAbit                ; FORCE THE SDA LINE HIGH
      setb      P1.SCLbit                ; FORCE THE SCL CLOCK LINE HIGH
        nop
        nop
        nop
      clr       P1.SDAbit                ; BEFORE TAKING THE SDA LOW
      nop
      nop
      nop
      nop
      clr       P1.SCLbit                ; FORCE THE SCL LOW
      ret

;*************************************************************************
;** Name: Stop
;** Description: Send stop command to the IIC Device
;** Function: This subroutine generates a stop condition on the bus. The stop
;**                 condition is defined as a low-high transition on the SDA
;**                 line while the SCL is high. The stop is used to indicate end
;**                 of current transaction.
;** Calls:                 None
;** Input:                 None
;** Return Value:          None
;** Register Usage:        None
;*************************************************************************
Stop:
      clr       P1.SDAbit                ; FORCE THE SDA LOW BEFORE TAKING
      setb      P1.SCLbit                ; THE SCL CLOCK LINE HIGH
      nop
      nop
      nop
      nop
      setb      P1.SDAbit                ; FORCE THE SDA HIGH (IDLE STATE)
      ret

;*************************************************************************
;** Name: ResetIIC
;** Description: Resets the IIC Device
;** Function: This subroutine is written for the worst case. System interruptions
;**                 caused by brownout or soft error conditions that reset the main
;**                 CPU may have no effect on the internal Vcc sensor and reset
;**                 circuit of the IIC Device. These are unpredictable and
;**                 random events that may leave the IIC Device interface
;**                 logic in an unknown state. Issuing a Stop command may not be
;**                 sufficient to reset the IIC Device.
```

```
;** Calls:                Start, Stop
;** Input:                None
;** Return Value:         None
;** Register Usage:       R0
;*****************************************************************************
ResetIIC:
      mov      R0,#0AH                    ; APPLY 10 CLOCKS TO THE DEVICE. EACH
ResetNxt:
      acall    Start                      ; CYCLE CONSISTS OF A START/STOP
      acall    Stop                        ; THIS WILL TERMINATE PENDING WRITE
      djnz     R0,ResetNxt                 ; COMMAND AND PROVIDES ENOUGH CLOCKS
      ret                                  ; FOR REMAINING BITS OF A READ OPERATION

;*****************************************************************************
;** END OF IIC Device INTERFACE SOURCE CODE
;*****************************************************************************
```

* 以上程序在 10.0592MHZ 晶振下使用 AT89C51、GMS97LS2051 测试全部通过，运行正确。
* 所有例程仅供参考，本公司不承担用户因应用上述程序所造成的一切后果。