P87LPC76x I²C 被控器编程规范

(AN463 －I²C slave routines for the PP87LPC76x)

摘要

本文给出了简明的仅支持被控器的 I²C 软件规范（而不是主控器或主控器和被控器）和一个 ASM 演示程序。本文是应用指南 AN464“用 P87LPC76x 作为 I²C 总线的主控器”的补充。

概述

P87LPC76x 是带有支持 I²C 总线接口硬件的小封装高性能的单片机。P87LPC76x 通过软件编程可作为 I²C 总线的主控器、被控器或两者具备。这里我们给出一个 P87LPC76x 作为 I²C 总线被控器的编程例子。I²C 总线概述参见应用指南 AN464“用 P87LPC76x 作为 I²C 总线的主控器”。 AN464 中有 I²C 总线主控器演示程序。

本文程序演示了 P87LPC76x 作为 I²C 总线上的被控器的编程规范，它是应用指南 AN464 中程序的补充，AN464 的程序中 P87LPC76x 是作为 I²C 总线上的主控器。可以在 I²C 总线上运用两个 P87LPC76x 单片机互相通信，一个运行 AN464 的程序，另一个运行本文的程序。

本文和 AN464 中的程序可使 P87LPC76x 作为主控器或被控器，但不可两者具备；应用指南 AN435 中描述了在多主控器环境下，单片机在主控器和被控器之间变换的编程方法。

I²C 总线上的被控器的编程相对简单些。被控器不用初始化总线，但要响应主控器的初始化信息。被控器接收或传送数据时，都将产生一个应答信号。被控器不用考虑总线仲裁和那些不承认相应地址的器件。因为被控器被设定成不可以控制总线，我们不要求被控器解决总线冲突或“挂起”问题。如果总线没有激活，程序将被撤出，而不会与总线上的主控器（或多个主控器）交涉，主控器将会解决这些问题。

P87LPC76x 有一个字节寄存器硬件接口可以直接影响总线上器件的级别。软件与硬件寄存器一起构成 I²C 总线协议。硬件和处理寄存器的服务程序是密不可分的，这里我们再次重复 P87LPC76x I²C 总线主控器应用指南中的警戒：修改服务程序时一定要额外小心。

I²C 总线被控器的服务程序以中断方式传递每个信息，这样主控器的通信请求可以不与被控器上运行的程序同步，可以写一些不用中断方式驱动的被控器应用程序，注意在运行其它程序时不要丢失主控器的传送信息，但是程序员最好不要这样编程。被控器应该响应总线上主控器的异步请求，这样中断服务程序变得很重要，正如本文程序中的中断服务程序一样，可以轻易地解决这个问题。

演示程序

下面的主程序是一个简单的演示程序，程序中有两个数据缓冲区，一个用于数据接收，一个用于数据传送。当新的数据从 I²C 总线上送到数据接收缓冲区时，程序将把它写到数据传送缓冲区，所接收数据的第一个字节复制到 I/O 口 1，当总线主控器请求读数据时，I/O 口 0 将返回所需数据的第一个字节，而其它字节从数据传送缓冲区中读出。主控器可以比较接收的数据和传送的数据，这可作为主控器和被控器系统的简单测试。这种方式下，P87LPC76x 也可用作一个单字节的 I²C I/O 口。

程序在地址 0 处开始执行，即硬件复位后单片机开始执行程序，这里包含一个跳转指令跳到主程序。主程序在标号 Reset 处开始，一直至程序代码结束。主程序中复位程序初

始化堆栈指针和被控器的 I²C 总线地址(MyAddr)，以及清除数据缓冲区和软件标志位。在本程序中，数据接收和传送缓冲区都为 8 字节，其最大字节数由标号 MaxBytes 的值决定，可以通过改变 MaxBytes 的值和分配更多的数据内存给数据缓冲区实现长数据的处理。
设置寄存器 I2CFG 的第 7 位可使装置成为 I²C 被控器，同时也装入 CTVAL 的值（位 CT0 和 CT1 的值）它们的值由单片机晶振的频率确定。设置 I2CON 的相应标志位可使装置进入被控器等待模式。激活定时器 1 作为“看门狗”定时器检测总线的挂起，并且使能定时器 1 的中断。

初始化之后，程序转到 Mainloop 子程序，大多数时间下程序将在此“挂起”等待 I²C 中断的产生。当发生一个 I²C 总线请求时，将产生一个中断和运行服务程序，然后应返回到标号 Mainloop 处。如果接收到新的数据，服务程序将设置标志位 DatFlag,表明数据已更新。标志位 DatFlag 被设置之后，将离开 Mainloop 子程序，接着执行一段短程序，把更新内容的数据输入缓冲区拷贝到数据输出（传送）缓冲区。

如果在复写旧的数据传送缓冲区结束之前，发生一个新的总线中断，将会产生新旧混合的数据，这是不希望的。避免这种情况发生的方法是在拷贝数据到数据传送缓冲区之前关闭 I²C 总线中断（清除中断使能寄存器 IEN1 中位 IE2），并在拷贝操作结束时重新打开 I²C 总线中断。

拷贝过程结束后，DatFlag 将被清除，程序跳转回 Mainloop 处，等待下一个中断发生。如果产生的中断请求为传送数据，服务程序将不会设置 DatFlag，程序将停留在 Mainloop 处。

中断服务程序

每个 I²C 总线状态与中断服务程序相对应，在软件轮询基础上，I²C 总线状态和硬件相互作用。总线起动条件产生时将产生中断，使服务程序初始化以及在 33H 处开始指令。

在保持寄存器的值之后，除了 I²C 中断外，其它所有的中断都有效，因为在服务程序运行过程中需要响应其它的中断。原则上在整个系统中，I²C 中断的优先级比其它中断低。由于 I²C 硬件将保持时钟运行直到程序响应，其它合理的中断过程不会分防碍总线上数据的传送。

应该在程序循环时变换标志位 ATN 的值，激活 I²C 硬件，我们总线上的变化使程序从头开始，所以在中断使能时应清除标志位 E12 即屏蔽 I²C 中断（运用 ACALL 和 RETI 指令）

在标号 Slave 处，程序开始接收总线上的地址数据。在软件循环等待期间检测到标志位 ATN 被硬件置位后，程序读每一个新的地址数据位。必须注意的是，由于 I²C 总线是以单个位方式传送数据，所以软件程序应与硬件紧密配合。例如：在进入循环状态等待下一个数据位之前，软件必须清除总线起动状态；如果软件不清除起动状态，硬件将把总线时钟线（SCL）拉低，这样第一个地址数据位将不会产生（实际上起动状态不会持续进行，因为定时器 1 的溢出将使器件释放总线）。

接收字节子程序的一部分内容为接收前 8 位数据（地址+R/W）程序把接收到的地址数据与 MyAddr（运行本程序被控器地址数据）相比较，若两者不相等，被控器将被闲置并退出服务程序；如果两者相等，R/W 位经过检测后使程序转到相应标号。当 R/W 位为“0”时，主控器请求写操作，被控器应该接收写入它的数据；当 R/W 位为“1”时，主控器请求读操作，被控器应该传递数据（在标号 Read 处执行程序）。

在“主写”模式下，被控器应该在接收到地址字节数据后发送一个应答位，接着继续接收字节数据，而且在接收到每一个字节数据后发送一个应答位，并把这些数据传送到数据接收缓冲区。对于长的数据，当数据缓冲区满时（接收了 MaxByte 个字节），被控器从总线上读取一个额外字节，然后发送一个低电平应答信号，使主控器知道应停止发送数据

给被控器，然后设置 DatFlag 指明已接收到新数据，然后程序跳转到 MsgEnd 处，等待下一个停止信号或重复的起动信号。在出现停止信号后，被控器进入待机模式（GoIdle）并且离开总线服务程序。在重起动状态下，被控器再次起动在标号 Slave 处执行程序接收新的地址数据。

如果数据长度足够短，使数据接收缓冲区没有添满，子程序 RcvByte (在 WrtLoop 后调用) 将返回到停止条件，并且不设置标志位 DRDY，程序将退出循环，通过子程序 WLEx 设置 DatFlag,并转移程序到 MsgEnd 处运行。

在“主读”模式下，子程序 RdLoop 调用 XmitByte 把数据传送缓冲区中的数据一个字节一个字节地传送到总线上。当数据传送缓冲区中的所有数据被传送完毕，或主控器没有发送应答信号时，被控器将退出循环程序。注意，被控器传送数据后没有接到主控器的应答信号时，不一定是总线出现了问题，或许是因为接收数据的主控器正处于忙状态。I^2C 总线协议有一个操作可使接收主控器在接收到可接纳的最后一个字节后发出信号使被控器停止传送数据，但是总线协议不包含确定所传送数据长度的方法。

子程序

子程序与硬件以及总线上的活动紧密配合。子程序 XimitByte 用于传送一个字节和接收应答位。字节接收程序有不同的人口指针用于接收数据，接收地址字节数据或处理应答信号。当检测到起动信号或停止信号时，子程序立即返回调用处，调用程序检测标志位确定是否整个字节数据都已收到（标志位 DRDY 将被设置）或者产生了一个起动条件或停止条件。

RcvByte 处程序代码表明共有 9 位数据从总线上读出，其中第一位数据不属于接收字节，它是上次所传送字节或地址的应答信号。读取寄存器 I2DAT 中的应答位将清除数据发送状态和标志位 DRDY，释放 SCL，允许总线处理下一个数据位，所返回的应答位保留在进位标志位中，而实际所接收的数据字节保留在寄存器 Acc 中。

定时器 1 的中断代码在地址 73H 处执行，地址 73H 处有一个跳转指令把程序跳到定时器 1 服务程序。定时器 1 中断由看门狗定时器产生，即在数据传送过程中 I^2C 总线“挂起”长时间后产生这个中断。被控器则清除总线接口，在标号 Reset 处重新运行程序。

P87LPC76x I^2C 总线被控器程序

本程序为 P87LPC76x I^2C 总线被控器演示程序，本程序中的传送和接收数据缓冲区各为 8 字节。主程序把接收到的数据拷贝到传送数据缓冲区，这样总线上的主控器可以读回所传送的数据。

缓冲区地址 0 映象到端口 1，这样 I^2C 写操作将影响端口的输出，不包括 I^2C 管脚 P1.6 和 P1.7。I^2C 读操作将把数据返回到端口 0。在一次 I^2C 总线传送数据过程中，只可以写 8 个字节数据，多余的字节不接收。同样的，在一次 I^2C 总线传送数据过程中，只能读 8 个字节数据。本程序不支持子地址用于访问缓冲区。

```
; 数值定义
CTVAL       equ     02h             ; I²C 中断中标志位 CT1，CT0 值
MaxBytes    equ     8               ; 传送或接收数据的最大字节数

SlvAdr      equ     7Eh             ; 被控器 I²C 从地址
```

```
; SlvAdr    equ     76h             ; I²C 被控器 LED 显示地址



; 屏蔽 I2CFG 标志位
BTIR        equ     10h             ; 屏蔽 TIRUN
BSLAV       equ     80h             ; 屏蔽 SLAVEN

; 屏蔽 I2CON 标志位
BCXA        equ     80h             ; 屏蔽 CXA
BIDLE       equ     40h             ; 屏蔽 IDLE
BCDR        equ     20h             ; 屏蔽 CDR
BCARL       equ     10h             ; 屏蔽 CARL
BCSTR       equ     08h             ; 屏蔽 CSTR
BCSTP       equ     04h             ; 屏蔽 CSTP

; 分配 I²C 程序内存
RcvDat      data    10h             ; I²C 接收数据缓冲区（8 字节）
                                    ; 地址为 10h～17h
XmtDat      data    18h             ; I²C 传送数据缓冲区 （8 字节）
                                    ; 地址为 18h～1Fh
flags       data    20h             ; I²C 软件状态标志
NoAck       bit     Flags.7         ; 保持低电平应答信号标志位
DatFlag     bit     Flags.6         ; 表明是否已发生 I²C 写操作
BitCnt      data    21h             ; I²C 位计数器
ByteCnt     data    22h             ; 发送/接收字节计数器
MyAddr      data    24h             ; 保存被控器地址
AdrRcvd     data    25h             ; 保存所接收的从地址＋R/W
RWFlag      bit     AdrRcvd.0       ; 被控器读/写标志位

; ***************************************************************************
;                                 程序代码
; ***************************************************************************

; 复位和中断向量

            ajmp    Reset           ; 复位向量在地址 0

; 当被控器闲置时 I²C 中断用于检测总线的起动条件

            org     33h             ; I²C 中断
            push    psw             ; 保存 psw
            push    Acc             ; 保存 Acc
            clr     ei2             ; 清除 I²C 中断
```

```
            acall   ClrInt          ; 开放其它中断
            ajmp    Slave


; 定时器 1 溢出中断服务程序

            org     0073h           ; 定时器 1 中断地址
TimerI:     setb    CLRTI           ; 清除定时器 1 中断
            mov     I2CFG,#0        ; 关闭 I²C
            mov     I2CON,#BCXA+BCDR+BCARL+BCSTR+BCSTP ; 复位 I²C 标志位
            clr     TIRUN
            acall   ClrInt          ; 清除中断悬挂状态
            ajmp    Reset           ; 重新开始
ClrInt:
            reti
                                                                        ;
;*************************************************************************
;                       传送和接收数据主程序
; *************************************************************************

            org     0100h
Slave:
            mov     I2CON,#BCARL+BCSTP+BCSTR+BCXA  ; 清除起动状态
            jnb     ATN,$           ; 等待下一个数据位
            mov     BitCnt,#7       ; 送位总数
            acall   RcvB2           ; 取剩下的从地址位
            mov     AdrRcvd,A       ; 保存所接收的地址＋R/W 位
            clr     Acc
            cjne    A,MyAddr,GoIdle  ; 若地址不匹配，则进入空闲状态
            jb      RWFlag,Read     ; 读或写?
            mov     R0,#RcvDat      ; 设置数据接收缓冲区指针
            mov     ByteCnt,#MaxBytes  ; 最多可接收的数据为 4 字节
WrtLoop:
            acall   SendAck         ; 送应答位
            acall   RcvByte         ; 取主控器的数据
            jnb     DRDY,WLEx       ; 状态结束?
            mov     @R0,A           ; 保存数据
            inc     R0              ; 增益缓冲区指针
            djnz    ByteCnt,WrtLoop ; 若缓冲区未满，则继续结束数据
            acall   SendAck         ; 送应答位
            acall   RcvByte         ; 取下一个数据但不保存
            mov     I2DAT,#80H      ; 送反应答位
            JNB     ATN,$           ; 等待送应答位
WLEx:
```

```
            setb    DatFlag             ; 设置 DatFlag 表明已收到数据
            sjmp    MsgEnd              ; 缓冲区已满，进入空闲模式
read:
            mov     R0,#XmtDat          ; 设置传送数据缓冲区
            mov     ByteCnt,#MaxBytes   ; 传送数据的最大字节数
            acall   SendAck             ; 送地址应答位
RdLoop:
            mov     A,@R0               ; 取缓冲区中数据字节数
            cjne    R0,#XmtDat,RdL1     ; 若缓冲区地址为 0，
            mov     A,P0                ; 则端口 0 数据返回到缓冲区
RdL1:
            inc     R0                  ; 增益缓冲区指针
            acall   XmitByte            ; 送数据字节数
            jb      NoAck,RLEx          ; 若无应答信号，则退出
            djnz    ByteCnt,RdLoop      ; 若还有多余数据要传送到缓冲区 1 时退出
            mov     A,P3
RLEx:
            sjmp    MsgEnd              ; 结束，进入空闲模式
MsgEnd:
            jnb     ATN,$               ; 等待停止信号或重复的起动信号
            jb      STR,Slave           ; 若是重复起动信号，则进入被控器模式
                                        ; 否则进入空闲模式
GoIdle:
            mov     I2CON,#BCSTP+BCXA+BCDR+BCARL+BIDLE  ; 进入空闲模式
            pop     Acc                 ; 恢复 Acc
            pop     psw                 ; 恢复 psw
            setb    EI2                 ; 开放 I²C 中断
            ret

; ***************************************************************************
;                                子程序
; ***************************************************************************

; 字节数据传送程序
; 数据在寄存器 A 中

XmitByte:   mov     bitCnt,#8           ; 数据设为 8 位

XmBit:      mov     I2DAT,A             ; 送数据位
            rl      A                   ; 取下一个数据位
            jnb     ATN,$               ; 等待传送数据位
            djnz    BitCnt,XmBit        ; 重复操作直到所有数据被传送
            mov     I2CON,#BCDR+BCXA    ; 转换到接收数据模式
            jnb     ATN,$               ; 等待应答位
```

```
            mov    Flages,I2DAT        ; 保存应答位
            ret
```

; 接收字节程序
; SendAck:送应答位
; RcvByte:接收一个字节数据
; RcvB2 :接收 I²C 数据的一部分，用于接收 7 位的被控器地址数
; 数据返回到 Acc

```
SendAck:    mov      I2DAT,#0           ; 送接收应答位
            JNB      ATN,$              ; 等待应答位的传送
            ret
            ;
RcvByte:    mov       BitCnt,#8         ; 设置位数值
RcvB2:      clr       A                 ; 接收字节数据初始化
RBit:       orl       A,I2DAT           ; 取数据位，清除 ATN
            rl        A                 ; 数据左移
            jnb       ATN,$             ; 等待下一个数据位
            jnb       DRDY,RBEX         ;  若无数据位，则退出
            djnz      BitCnt,RBit       ; 重复操作取 7 个数据位
            mov       C,RDAT            ; 取最后一个数据位，但不清除 ATN
            rlc       A                 ; 形成完整的字节数据
RBEx:
            ret
            ;

; ***************************************************************************
;                              主程序
; ***************************************************************************

Reset:
            mov       SP,#2Fh           ; 设置堆栈区
            mov       R0,#RcvDat        ; 设置指针到数据区
            mov       R1,#2*MaxBytes    ; 设置缓冲区长度计数器
RLoop:
            mov       @R0,#0            ; 清除缓冲区内容
            inc       R0                ; 增益 R0 到下一个缓冲区
            djnz      R1,RLoop          ; 重复操作
            mov       AdrRcvd,#0
            mov       MyAddr,#SlvAdr    ; 设置被控器地址
            mov       Flags,#0          ; 清除系统标志
            setb      EI2               ; 开放 I²C 中断
```

```
        setb    ETI             ; 开放定时器 1 中断
        setb    EA              ; 开放系统中断
        mov     I2CFG,#BSLAV+CTVAL  ; 使能被控器功能
        mov     I2CON,#BCSTR+BCSTP+BXCA+BCDR+BCARL_BIDLE
                                ; 使被控器进入空闲模式
        setb    TIRUN           ;  打开定时器 1
```

；以下程序在发生 I²C 写操作时，把所接收的第一个字节数据拷贝到端口 0，
；同时把数据输入缓冲区剩下的数据拷贝到数据输出缓冲区中。

```
MainLoop:
        jnb     DatFlag,$       ; 等待数据传送
        mov     pcon,#01h       ; 进入空闲模式
        clr     EA              ; 数据传送时关闭中断
        mov     A,RcvDat+1      ; 取第一个字节数据（第二个缓冲区）
        orl     A,#0Ch          ; 屏蔽 I²C 管脚防止出错
        mov     P1,A            ; 保存数据到端口 1
        mov     R0,#RcvDat      ; 设置数据输入缓冲区头指针
        mov     R1,#XmtDat      ; 设置数据输出头指针
        mov     R2,#MaxBytes    ; 设置缓冲区长度计数器
ML2:
        mov     A,@R0           ; 从数据输入缓冲区中取数据
        mov     @R1,A           ; 保存数据到输出缓冲区
        inc     R1              ; 增益数据输入缓冲区指针
        inc     R0              ; 增益数据输出缓冲区指针
        djnz    R2,ML2          ; 重复操作到整个缓冲区被更新
        clr     DatFlag         ;  清除 I²C 传送数据标志
        setb    EA              ; 传送数据结束，开放中断
        sjmp    Mainloop        ; 等待下一个 I²C 数据传送
        org     0fd00h          ; EPROM 配置字节（寄存器 UCFG1）
        db      038h            ; 关闭 WDT，使能 RST，端口复位时置为高
                                ; 掉电复位电压置为 2.5V，CLKR 为 1，振荡器
                                ; 频率范围为 4MHz～20MHz
        ;
        end
```