附录四 PS/2 设备例程

These routines can be used to emulate a PS/2 mouse or keyboard. They were written for a PIC16F84 @ 4.61 MHz +/- 25% (perfect for a 5k/20pF RC oscillator). For more information about the PS/2 mouse, keyboard, and their protocol, check out one of the folowing links:
这个例程可用于仿真 PS/2 鼠标或键盘。适用于 PIC16F84，振荡频率为 4.61MHz +/-25%（最好用 5k/20pF RC 振荡器）。关于 PS/2 鼠标、键盘及它们的协议，检查下列的连接：

The AT Keyboard Interface
The PS/2 Mouse Interface
PS/2 Mouse/Keyboard Protocol
（译者注：中文译文的第二章、第三章和第一章。向上翻！）

Header:

```
;-----------------------------------------------------------------------------
; CLOCK/TIMING INFORMATION:
;-----------------------------------------------------------------------------
;
;     PS/2 bus clock low time =    40 us +/- 25% (30 us - 50 us)
;     PS/2 bus clock high time = 40 us +/- 25% (30 us - 50 us)
;     RC osc @ 20pF/5k = 4.61 MHz +/- 25% (3.50 MHz - 5.76 MHz)
;     1 instruction cycle @ 4.61 MHz (RC) = 0.87 us +/- 25% (0.65 us - 1.09 us)
;     Optimum PS/2 bus clock low time @4.61MHz = 45.97 instruction cycles
;     Actual PS/2 bus clock low time = 46 instruction cycles
;     Actual PS/2 bus clock low time @4.61MHz (RC) = 40.0us +/- 25% (30us-50us)
;     Actual PS/2 bus clock frequency @461MHz (RC) = 12.5 kHz +/- 25% (10.0kHz-16.7kHz)
;-----------------------------------------------------------------------------
;           HEADER:
;-----------------------------------------------------------------------------
   TITLE           "PS/2 Device Routines"
   SUBTITLE        "By Adam Chapweske"
   LIST            P=16F84
   INCLUDE         "p16f84.inc"
   RADIX           DEC
   ERRORLEVEL    -224, 1
   __CONFIG        _CP_OFF & _WDT_OFF & _RC_OSC


;-----------------------------------------------------------------------------
;          DEFINES:
;-----------------------------------------------------------------------------
#DEFINE DATA    PORTB, 7
#DEFINE CLOCK PORTB, 6

;-----------------------------------------------------------------------------
; RAM ALLOCATION:
```

```
;-------------------------------------------------------------------------------
  cblock
        TEMP0
        RECEIVE
        PARITY
        COUNTER
  endc
```

Required Routines & Macros:

```
;-------------------------------------------------------------------------------
;       MACROS:
;-------------------------------------------------------------------------------
Delay   macro   Time            ;Delay "Cycles" instruction cycles
  if (Time==1)
        nop
        exitm
  endif
  if (Time==2)
        goto $ + 1
        exitm
  endif
  if (Time==3)
        nop
        goto $ + 1
        exitm
  endif
  if (Time==4)
        goto $ + 1
        goto $ + 1
        exitm
  endif
  if (Time==5)
        goto $ + 1
        goto $ + 1
        nop
        exitm
  endif
  if (Time==6)
        goto $ + 1
        goto $ + 1
        goto $ + 1
        exitm
  endif
  if (Time==7)
```

```
            goto $ + 1
            goto $ + 1
            goto $ + 1
            nop
            exitm
    endif
    if (Time%4==0)
            movlw (Time-4)/4
            call Delay_Routine
            exitm
    endif
    if (Time%4==1)
            movlw (Time-5)/4
            call Delay_Routine
            nop
            exitm
    endif
    if (Time%4==2)
            movlw (Time-6)/4
            call Delay_Routine
            goto $ + 1
            exitm
    endif
    if (Time%4==3)
            movlw (Time-7)/4
            call Delay_Routine
            goto $ + 1
            nop
            exitm
    endif
    endm

;-------------------------------------------------------------------------------------------
; DELAY:
;-------------------------------------------------------------------------------------------
;Delays 4w+4 cycles (including call,return, and movlw) (0=256)
Delay_Routine     addlw     -1                    ;Precise delays used in I/O
                  btfss     STATUS, Z
                  goto       Delay_Routine
                  return
```

ByteOut:

Sends a byte in w to the host. Returns 0xFE if inhibited during transmission. Returns 0xFF if host interrupts to send its own data. Returns 0x00 if byte sent successfully.

```
;-------------------------------------------------------------------------------
; OUTPUT ONE BYTE:    - TIMING IS CRITICAL!!!
;-------------------------------------------------------------------------------
ByteOut         movwf   TEMP0
InhibitLoop     btfss   CLOCK           ;Test for inhibit
                goto    InhibitLoop
                Delay   50
                btfss   CLOCK
                goto    InhibitLoop
                btfss   DATA            ;Check for request-to-send
                retlw   0xFF
                clrf    PARITY
                movlw   0x08
                movwf   COUNTER
                movlw   0x00
                call    BitOut          ;Start bit (0)
                btfss   CLOCK           ;Test for inhibit
                goto    ByteOutEnd
                Delay   4
ByteOutLoop     movf    TEMP0, w
                xorwf   PARITY, f
                call    BitOut          ;Data bits
                btfss   CLOCK           ;Test for inhibit
                goto    ByteOutEnd
                rrf     TEMP0, f
                decfsz  COUNTER, f
                goto    ByteOutLoop
                Delay   2
                comf    PARITY, w
                call    BitOut          ;Parity bit
                btfss   CLOCK           ;Test for inhibit
                goto    ByteOutEnd
                Delay   5
                movlw   0xFF
                call    BitOut          ;Stop bit (1)
                Delay   48
                retlw   0x00

ByteOutEnd      bsf     STATUS, RP0
                bsf     DATA
                bsf     CLOCK
                bcf     STATUS, RP0
                retlw   0xFE

BitOut          bsf     STATUS, RP0
                andlw   0x01
```

```
                btfss   STATUS, Z
                bsf     DATA
                btfsc   STATUS, Z
                bcf     DATA
                Delay   21
                bcf     CLOCK
                Delay   45
                bsf     CLOCK
                bcf     STATUS, RP0
                Delay   5
                return
```

ByteIn:

Reads a byte from the host. Result in "RECEIVE" register. Returns 0xFE in w if host aborts transmission. Returns 0xFF in w if framing/parity error detected. Returns 0x00 in w if byte received successfully.

```
;-------------------------------------------------------------------------------------------
; READ ONE BYTE: - TIMING IS CRITICAL!!!
;-------------------------------------------------------------------------------------------
ByteIn          btfss   CLOCK           ;Wait for start bit
                goto    ByteIn
                btfsc   DATA
                goto    ByteIn
                movlw   0x08
                movwf   COUNTER
                clrf    PARITY
                Delay   28
ByteInLoop      call    BitIn           ;Data bits
                btfss   CLOCK           ;Test for inhibit
                retlw   0xFE
                bcf     STATUS, C
                rrf     RECEIVE, f
                iorwf   RECEIVE, f
                xorwf   PARITY,f
                decfsz  COUNTER, f
                goto    ByteInLoop
                Delay   1
                call    BitIn           ;Parity bit
                btfss   CLOCK           ;Test for inhibit
                retlw   0xFE
                xorwf   PARITY, f
                Delay   5
ByteInLoop1     Delay   1
                call    BitIn           ;Stop bit
                btfss   CLOCK           ;Test for inhibit
                retlw   0xFE
```

```
                xorlw   0x00
                btfsc   STATUS, Z
                clrf    PARITY
                btfsc   STATUS, Z           ;Stop bit = 1?
                goto    ByteInLoop1         ;   No--keep clocking.

                bsf     STATUS, RP0         ;Acknowledge
                bcf     DATA
                Delay   11
                bcf     CLOCK
                Delay   45
                bsf     CLOCK
                Delay   7
                bsf     DATA
                bcf     STATUS, RP0

                btfss   PARITY, 7           ;Parity correct?
                retlw   0xFF                ;   No--return error

                Delay   45
                retlw   0x00

BitIn           Delay   8
                bsf     STATUS, RP0
                bcf     CLOCK
                Delay   45
                bsf     CLOCK
                bcf     STATUS, RP0
                Delay   21
                btfsc   DATA
                retlw   0x80
                retlw   0x00
```

附录五 PS/2 主机例程

These routines can be used to interface a PS/2 mouse or keyboard.
这些例程用于和 PS/2 鼠标或键盘进行接口。

PS2get:
This routine reads a byte from the PS/2 device (keyboard or mouse). Result in w.

```
PS2get          call        PS2getBit           ;Get/ignore the start bit
                movlw       0x08                ;Load Counter
                movwf       COUNTER
PS2getLoop      bcf         STATUS, C
                rrf         TEMP0, f
                call        PS2getBit           ;Read a data bit from the keyboard/mouse
                iorwf       TEMP0, f
                decfsz      COUNTER, f          ;Read 8 data bits yet?
                goto        PS2getLoop
                call        PS2getBit           ;Get/ignore parity bit.
                call        PS2getBit           ;Get/ignore stop bit
                movf        TEMP0, w            ;Result in w.
                return
PS2getBit       btfss       CLOCK               ;Make sure clock is high.
                goto        $ - 1
                btfsc       CLOCK
                goto        $ - 1
                goto        $ + 1
                btfss       DATA                ;Read data.
                retlw       0x00
                retlw       0x80
```

PS2cmd:
This routine sends a byte in w to a PS/2 mouse or keyboard. TEMP0, TEMP1, and TEMP2 are general purpose registers. CLOCK and DATA are assigned to port bits, and "Delay" is a self-explainatory macro. DATA and CLOCK are held high by setting their I/O pin to input and allowing an external pullup resistor to pull the line high. The lines are brought low by setting the I/O pin to output and writing a "0" to the pin.
PS2cmd:

```
        movwf   TEMP0           ;Store to-be-sent byte
        movlw   0x08            ;Initialize a counter
        movwf   TEMP1
        clrf    TEMP2           ;Used for parity calc
```

```
        bsf     STATUS, RP0
        bcf     CLOCK
        bcf     STATUS, RP0
        bcf     CLOCK           ;Inhibit communication
        Delay   100             ;for at least 100 microseconds
        bsf     STATUS, RP0
        bcf     DATA
        bcf     STATUS, RP0
        bcf     DATA            ;Pull DATA low
        Delay   5
        bsf     STATUS, RP0
        bsf     CLOCK           ;Release CLOCK
        bcf     STATUS, RP0
PS2cmdLoop:
        movf    TEMP0, w
        xorwf   TEMP2, f        ;Parity calc
        call    PS2cmdBit       ;Output 8 data bits
        rrf     TEMP0, f
        decfsz  TEMP1, f
        goto    PS2cmdLoop
        comf    TEMP2, w
        call    PS2cmdBit       ;Output parity bit
        movlw   0x01
        call    PS2cmdBit       ;Output stop bit (1)
        btfsc   CLOCK           ;Wait for acknowledge
        goto    $ - 1
        btfss   CLOCK
        goto    $ - 1
        return

PS2cmdBit:
        btfsc   CLOCK           ;Wait for CLOCK=low
        goto    $ - 1
        bsf     STATUS, RP0
        andlw   0x01
        btfss   STATUS, Z       ;Set/Reset DATA line
        bsf     DATA
        btfsc   STATUS, Z
        bcf     DATA
        bcf     STATUS, RP0
        btfss   CLOCK           ;Wait for CLOCK=high
        goto    $ - 1
        return
```

附录六 PS/2 "Access" Mouse

This is a fully-functional PS/2 mouse written for the PIC16F84 microcontroller. It can be adapted to virtually any inputs, which gives the user a lot of flexability in how he/she controls the computer. It was developed to give computer access to people with physical disabilities, but I'm sure you can find many additional uses for this project.
这是一个使用 PIC16F84 微控制器的全功能的 PS/2 鼠标。它实际上可以适合任何输入，这就给了用户巨大的灵活性来控制他们的计算机。开发它出来让有残障的人们能访问计算机，但我确信你可以为这个工程找到更多额外的用途。

Feel free to use the code for non-commercial purposes only. You may distribute the code only if it is unmodified from its orginal form. I do not imply any warrenties or guarantees with this code. Use at your own risk. Enjoy!

Click on the following links to get the files:

- Access Mouse v1.50 - MPASM source code
- Access Mouse v1.50 - Schematic diagram (jpg)
- Access Mouse v1.51 - MPASM source code
- Access Mouse v1.51 - Schematic diagram (jpg)

VERSION 1.51:

- All inputs are active low.
- Speed is controlled in software by adjusting "PERIOD" and "DISTANCE" constants.
- All inputs, including "Clock" and "Data" may be assigned to any I/O pin.

VERSION 1.50:

- Internal PORTB pullups are enabled and all inputs are active low.
- All movement/button inputs may be assigned to any pin on PORTB.
- Speed is controlled by adjusting a variable resistor.
- An LED indicates the mouse's status.
- Potientiometer and LED may be assigned to any pin on PORTA.

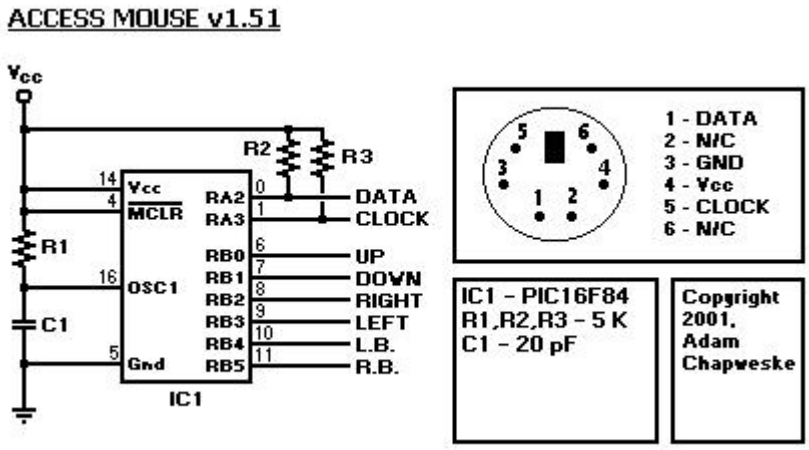
Just to give you an example of how these may be used, if you were to connect a condensor microphone element to a 339 Quad Comparator, then connect the output of the comparator to the left mouse button input pin on the PIC, you will be able to emulate a mouse click by blowing on the microphone.
If you find any bugs or have any comments, please send me an email. You may also email me your questions, but I won't have time to respond to most of them. This is a work under progress. Check back every few weeks for updated code and additions to this page.

For more information related to this project, try the following links:

- The PS/2 Mouse Interface
- PS/2 Mouse/Keyboard Protocol
- PIC Code/Projects
- Adam's Micro-Resources Home

- More Links

Access Mouse 1.51 版的电路图：



Access Mouse 1.51 版的软件：（比较长的样子，我没有核对，小心！）

```
;    PS/2 Mouse Emulator by Adam Chapweske (chap0179@tc.umn.edu)                    v1.51
;    http://panda.cs.ndsu.nodak.edu/~achapwes/
;    This was written for the PIC16F84 with an RC oscillator @ 20pF/5kohm
;    (will work with any oscillator between 3.50 MHz - 5.76 MHz)
;
;    FEEL FREE TO USE THIS CODE FOR NON-COMMERCIAL PURPOSES ONLY.    YOU MAY DISTRIBUTE
;    THIS CODE ONLY IF IT IS UNMODIFIED FROM ITS ORIGINAL FORM AND IT MUST CONTAIN THIS
;    HEADING.    I DO NOT IMPLY ANY WARANTEES OR GUARANTEES.    USE AT YOUR OWN RISK.    ENJOY!!!
;
;    Copyright 2001, Adam Chapweske

;-----------------------------------------------------------------------------
; CLOCK/TIMING INFORMATION:
;-----------------------------------------------------------------------------
;
;    PS/2 bus clock low time =    40 us +/- 25% (30 us - 50 us)
;    PS/2 bus clock high time = 40 us +/- 25% (30 us - 50 us)
;    RC osc @ 20pF/5k = 4.61 MHz +/- 25% (3.50 MHz - 5.76 MHz)
;    1 instruction cycle @ 4.61 MHz (RC) = 0.87 us +/- 25% (0.65 us - 1.09 us)
;    Optimum PS/2 bus clock low time @4.61MHz = 45.97 instruction cycles
;    Actual PS/2 bus clock low time = 46 instruction cycles
;    Actual PS/2 bus clock low time @4.61MHz (RC) = 40.0us +/- 25% (30us-50us)
;    Actual PS/2 bus clock frequency @461MHz (RC) = 12.5 kHz +/- 25% (10.0kHz-16.7kHz)

;-----------------------------------------------------------------------------
;            HEADER:
;-----------------------------------------------------------------------------
            TITLE           "PS/2 Mouse Emulator"
            SUBTITLE        "Copyright 2001, Adam Chapweske"
```

```
        LIST    P=16F84
        INCLUDE "p16f84.inc"
            RADIX       DEC
        ERRORLEVEL  -224, 1
            __CONFIG    _CP_OFF & _WDT_OFF & _XT_OSC

;-----------------------------------------------------------------------------------------
;   DEFINES:
;-----------------------------------------------------------------------------------------
#DEFINE  DATA     PORTA, 2  ;May be assigned to any I/O pin
#DEFINE  CLOCK    PORTA, 3  ;May be assigned to any I/O pin
#DEFINE PS2_Yp    PORTB, 0  ;May be assigned to any I/O pin
#DEFINE  PS2_Yn   PORTB, 1  ;May be assigned to any I/O pin
#DEFINE  PS2_Xp   PORTB, 2  ;May be assigned to any I/O pin
#DEFINE  PS2_Xn   PORTB, 3  ;May be assigned to any I/O pin
#DEFINE  PS2_Bl   PORTB, 4  ;May be assigned to any I/O pin
#DEFINE  PS2_Br   PORTB, 5  ;May be assigned to any I/O pin


#DEFINE  PERIOD   20        ;Time between reading of inputs.   Min=(osc frequency)/204800
#DEFINE  DISTANCE 2         ;Amount by which X/Y counters are incremented/decremented

;-----------------------------------------------------------------------------------------
;   RAM ALLOCATION:
;-----------------------------------------------------------------------------------------

  cblock 0x0C

    TEMP0, TEMP1
    RECEIVE, PARITY, COUNTER   ;Used in I/O routines
    REPORT_RATE, RESOLUTION    ;Used for responses to status requests
    FLAGS, XY_FLAGS
    dBUTTONS                   ;"Delta Button States"
    X_COUNTER
    Y_COUNTER
  endc

;-----------------------------------------------------------------------------------------
;FLAGS:
; bit 7 -- Always 0
; bit 6 -- Stream(0)/Remote(1) mode
; bit 5 -- Disable(0)/Enable(1) reporting
; bit 4 -- 1:1(0)/2:1(1) Scaling
; bit 3 -- Always 0
; bit 2 -- Always 0
; bit 1 -- Always 0
```

```
; bit 0 -- Always 0

MODE      equ   6
ENABLE    equ   5
SCALE     equ   4

;-----------------------------------------------------------------------------------------
;XY_FLAGS:
; bit 7 -- Y Counter overflow
; bit 6 -- X Counter overflow
; bit 5 -- Y counter sign bit
; bit 4 -- X counter sign bit
; bit 3 -- Always 1
; bit 2 -- Always 0 (middle button)
; bit 1 -- Previous right button state
; bit 0 -- Previous left button state

yOVF      equ   7
xOVF      equ   6
ySIGN     equ   5
xSIGN     equ   4

;dBUTTONS
; bit 7 -- Always 0
; bit 6 -- Always 0
; bit 5 -- Always 0
; bit 4 -- Always 0
; bit 3 -- Always 0
; bit 2 -- Always 0
; bit 1 -- Change in right buton state
; bit 0 -- Change in left button state

;-----------------------------------------------------------------------------------------

   cblock          ;Contains to-be-sent packet and last packet sent
      LENGTH
      SEND1
      SEND2
      SEND3
   endc

;-----------------------------------------------------------------------------------------
;            MACROS:
;-----------------------------------------------------------------------------------------

;Delay "Cycles" instruction cycles
```

```
Delay           macro       Time
  if (Time==1)
          nop
          exitm
  endif
  if (Time==2)
          goto  $ + 1
          exitm
  endif
  if (Time==3)
          nop
          goto  $ + 1
          exitm
  endif
  if (Time==4)
          goto  $ + 1
          goto  $ + 1
          exitm
  endif
  if (Time==5)
          goto  $ + 1
          goto  $ + 1
          nop
          exitm
  endif
  if (Time==6)
          goto  $ + 1
          goto  $ + 1
          goto  $ + 1
          exitm
  endif
  if (Time==7)
          goto  $ + 1
          goto  $ + 1
          goto  $ + 1
          nop
          exitm
  endif
  if (Time%4==0)
          movlw     (Time-4)/4
          call   Delay_us
          exitm
  endif
  if (Time%4==1)
          movlw     (Time-5)/4
          call   Delay_us
```

```
            nop
            exitm
    endif
    if (Time%4==2)
            movlw       (Time-6)/4
            call    Delay_us
            goto  $ + 1
            exitm
    endif
    if (Time%4==3)
            movlw       (Time-7)/4
            call    Delay_us
            goto  $ + 1
            nop
            exitm
    endif
            endm
;-------------------------------------------------------------------------------------------
;           ORG 0x000:
;-------------------------------------------------------------------------------------------

            org    0x000
            goto  Start

;-------------------------------------------------------------------------------------------
;       HANDLE COMMAND:
;-------------------------------------------------------------------------------------------

    if (high Table1End != 0)
            ERROR     "Command handler table must be in low memory page"
    endif

Command         movlw       0x04        ;Test for a resolution value
            subwf       RECEIVE, w
            bnc    SetResolution
            movlw       0xC8        ;Test for report rate value
            subwf       RECEIVE, w
            bnc    SetReportRate
            movlw       0xE6        ;0xE6 is lowest code
            subwf       RECEIVE, w
            bnc    MainLoop
HandlerTable    addwf       PCL, f              ;Add offset
            goto  Mouse_E6 ;0xE6 - Set Scaling 1:1
            goto  Mouse_E7 ;0xE7 - Set Scaling 2:1
            goto  MainLoop  ;0xE8 - Set Resolution
            goto  Mouse_E9 ;0xE9 - Status Request
```

```
            goto  Mouse_EA ;0xEA - Set Stream Mode
            goto  Report       ;0xEB - Read Data
            goto  MainLoop  ;0xEC - Reset Wrap Mode
            goto  MainLoop  ;0xED -
            goto  WrapMode ;0xEE - Set Wrap Mode
            goto  MainLoop  ;0xEF
            goto  Mouse_F0  ;0xF0 - Set Remote Mode
            goto  MainLoop  ;0xF1
            goto  Mouse_F2  ;0xF2 - Read Device Type
            goto  MainLoop  ;0xF3 - Set Report Rate
            goto  Mouse_F4  ;0xF4 - Enable
            goto  Mouse_F5  ;0xF5 - Disable
            goto  Mouse_F6  ;0xF6 - Set Default
            goto  MainLoop  ;0xF7
            goto  MainLoop  ;0xF8
            goto  MainLoop  ;0xF9
            goto  MainLoop  ;0xFA
            goto  MainLoop  ;0xFB
            goto  MainLoop  ;0xFC
            goto  MainLoop  ;0xFD
            goto  PacketOut  ;0xFE - Resend
Table1End goto  Reset        ;0xFF - Reset


;-------------------------------------------------------------------------------------------
;       START:
;-------------------------------------------------------------------------------------------

Start       clrf   PORTA
            clrf   PORTB
            bsf    STATUS, RP0    ;(TRISA=TRISB=0xFF by default)
            movlw      0x57        ;Timer mode, assign max. prescaler, enable pullups
            movwf      OPTION_REG
            bcf    STATUS, RP0
            movlw      0x08        ;Bit 3 always = 1, clear previous button states
            movwf      XY_FLAGS
;           goto  Reset


;-------------------------------------------------------------------------------------------
;           Reset Mode:
;-------------------------------------------------------------------------------------------
Reset       movlw      0xAA
            movwf      SEND1           ;Load BAT completion code
            call   LoadDefaults
            clrf   SEND2           ;Load Device ID (0x00)
            movlw      0x02
```

```
            movwf   LENGTH
            call    BATdelay
            goto    PacketOut  ;Output 2-byte "completion-code, device ID" packet

;-----------------------------------------------------------------------------
;           Stream/Remote Mode:
;-----------------------------------------------------------------------------
MainLoop    clrf    X_COUNTER   ;Clear movement counters
            clrf    Y_COUNTER

MainLoop1   btfss   DATA        ;Check for host request-to-send
            goto    PacketIn
            movlw   PERIOD      ;Report period
            subwf   TMR0, w
            btfss   STATUS, C   ;TMR0=report period?
            goto    MainLoop1   ;   No--loop
            clrf    TMR0        ;   Yes--reset TMR0, then read inputs...
            call    ReadInputs
            btfsc   FLAGS, MODE ;Stream(0)/Remote(1) mode
            goto    MainLoop1
            btfss   FLAGS, ENABLE   ;Disable(0)/Enable(1) reporting
            goto    MainLoop1
            movf    X_COUNTER, w    ;Test for X-movement
            iorwf   Y_COUNTER, w    ;Test for Y-movement
            iorwf   dBUTTONS, w ;Test for change in button states
            bz      MainLoop1
;           goto    Report

;-----------------------------------------------------------------------------
;       REPORT:
;-----------------------------------------------------------------------------

Report      movf    dBUTTONS, w
            xorwf   XY_FLAGS, f ;Find current button state
            movf    XY_FLAGS, w
            movwf   SEND1
            movf    X_COUNTER, w
            movwf   SEND2
            movf    Y_COUNTER, w
            movwf   SEND3
            movlw   0x03        ;Movement data report length
            movwf   LENGTH
;           goto    PacketOut

;-----------------------------------------------------------------------------
;       OUTPUT PACKET
```

```
;-------------------------------------------------------------------------------------------

PacketOut  movlw     SEND1          ;First byte of packet
           movwf     FSR
           movf LENGTH, w     ;Length of packet
           movwf     TEMP1
PacketOutLoop   movf INDF, w          ;Get data byte
           call   ByteOut          ;   Output that byte
           xorlw0xFF       ;Test for RTS error
           bz     PacketIn
           xorlw0xFE ^ 0xFF     ;Test for inhibit error
           bz     PacketOut
           incf   FSR, f          ;Point to next byte
           decfsz     TEMP1, f
           goto   PacketOutLoop
           goto   MainLoop

;-------------------------------------------------------------------------------------------
;      READ PACKET
;-------------------------------------------------------------------------------------------

PacketIn   call   ByteIn
           xorlw0xFF       ;Test for parity/framing error
           bz     Mouse_ERR
           xorlw0xFE ^ 0xFF     ;Test for inhibit error
           bz     MainLoop1
           movlw      0xFE        ;Test for "Resend" command
           xorwf      RECEIVE, w
           bz     PacketOut
Acknowledge     movlw      0xFA        ;Acknowledge
           call   ByteOut
           goto   Command

;-------------------------------------------------------------------------------------------
;      READ INPUTS:
;-------------------------------------------------------------------------------------------

ReadInputsmovlw      DISTANCE

           btfss  PS2_Xp          ;Read inputs
           addwf      X_COUNTER, f
           btfss  PS2_Yp
           addwf      Y_COUNTER, f
           btfss  PS2_Xn
           subwf      X_COUNTER, f
           btfss  PS2_Yn
```

```
        subwf   Y_COUNTER, f

        bcf     XY_FLAGS, xSIGN
        btfsc   X_COUNTER, 7
        bsf     XY_FLAGS, xSIGN
        bcf     XY_FLAGS, ySIGN
        btfsc   Y_COUNTER, 7
        bsf     XY_FLAGS, ySIGN

        movf    XY_FLAGS, w     ;Get previous button states
        andlw   b'00000111'
        btfss   PS2_Bl          ;Find changes in button states
        xorlwb'00000001'
        btfss   PS2_Br
        xorlwb'00000010'
        movwf   dBUTTONS        ;Save *change* in button state
        retlw 0x00

;-------------------------------------------------------------------------------
;       WRAP MODE:
;-------------------------------------------------------------------------------

WrapMode btfsc  DATA            ;Wait for RTS
        goto    WrapMode
        call    ByteIn          ;Read one byte from host
        xorlw0xFE       ;Test for aborted transmission
        bz      WrapMode
        movf    RECEIVE, w
        xorlw0xFF       ;Test for "Reset" command
        bz      Acknowledge
        xorlw0xFF^0xEC      ;Test for "Reset Wrap Mode" command
        bz      Acknowledge
        xorlw0xEC
        call    ByteOut         ;Else, echo
        goto    WrapMode

;-------------------------------------------------------------------------------
;       LOAD DEFAULT VALUES:
;-------------------------------------------------------------------------------

LoadDefaults    movlw   100         ;Default report rate
        movwf   REPORT_RATE
        movlw   0x02        ;Default resolution
        movwf   RESOLUTION
        clrf    FLAGS           ;Stream mode, 1:1 scaling, disabled
        retlw 0x00
```

```
;-------------------------------------------------------------------------------
;       EMULATE BAT:
;-------------------------------------------------------------------------------
BATdelay  clrf    TEMP0             ;Used for a 400 ms delay at power-on
          clrf    TEMP1
DelayLoop Delay   6
          decfsz  TEMP0, f
          goto    DelayLoop
          decfsz  TEMP1, f
          goto    DelayLoop
          retlw   0x00

;-------------------------------------------------------------------------------
;       HANDLE COMMANDS:
;-------------------------------------------------------------------------------

SetResolution     movf RECEIVE, w
          movwf   RESOLUTION
          goto    MainLoop

SetReportRate     movf RECEIVE, w
          movwf   REPORT_RATE
          goto    MainLoop

;0xE6 - Set Scaling 1:1
Mouse_E6  bcf     FLAGS, SCALE
          goto    MainLoop

;0xE7 - Set Scaling 2:1
Mouse_E7  bsf     FLAGS, SCALE
          goto    MainLoop

;0xE9 - Status Request
Mouse_E9  movf    FLAGS, w
          btfss   PS2_Bl
          iorlw   0x04
          btfss   PS2_Br
          iorlw   0x01
          movwf   SEND1
          movf    RESOLUTION, w
          movwf   SEND2
          movf    REPORT_RATE, w
          movwf   SEND3
          movlw   0x03
          movwf   LENGTH
```

```
            goto  PacketOut

;0xEA - Set Stream Mode
Mouse_EA bcf    FLAGS, MODE
            goto  MainLoop

;0xF0 - Set Remote Mode
Mouse_F0 bsf    FLAGS, MODE
            goto  MainLoop

;0xF2 - Get Device ID
Mouse_F2 clrf   SEND1
            movlw       0x01
            movwf       LENGTH
            goto  PacketOut

;0xF4 - Enable Reporting
Mouse_F4 bsf    FLAGS, ENABLE
            goto  MainLoop

;0xF5 - Disable Reporting
Mouse_F5 bcf    FLAGS, ENABLE
            goto  MainLoop

;0xF6 - Set Default
Mouse_F6 call   LoadDefaults
            goto  MainLoop

;Invalid command
Mouse_ERR       movlw       0xFE
            call   ByteOut
            goto  MainLoop

;-------------------------------------------------------------------------------
;       OUTPUT ONE BYTE:    - TIMING IS CRITICAL!!!
;-------------------------------------------------------------------------------

ByteOut         movwf       TEMP0
InhibitLoop     btfss  CLOCK            ;Test for inhibit
            goto  InhibitLoop
            Delay       100         ;(50 microsec = 58 clock cycles, min)
            btfss  CLOCK
            goto  InhibitLoop
            btfss  DATA             ;Check for request-to-send
            retlw 0xFF
            clrf   PARITY
```

```
        movlw   0x08
        movwf   COUNTER
        movlw   0x00
        call    BitOut          ;Start bit (0)
        btfss   CLOCK           ;Test for inhibit
        goto    ByteOutEnd
        Delay   4
ByteOutLoop     movf TEMP0, w
        xorwf   PARITY, f
        call    BitOut          ;Data bits
        btfss   CLOCK           ;Test for inhibit
        goto    ByteOutEnd
        rrf     TEMP0, f
        decfsz  COUNTER, f
        goto    ByteOutLoop
        Delay   2
        comf PARITY, w
        call    BitOut          ;Parity bit
        btfss   CLOCK           ;Test for inhibit
        goto    ByteOutEnd
        Delay   5
        movlw   0xFF
        call    BitOut          ;Stop bit (1)
        Delay   48
        retlw 0x00
ByteOutEnd      bsf     STATUS, RP0
        bsf     DATA
        bsf     CLOCK
        bcf     STATUS, RP0
        retlw 0xFE

BitOut          bsf     STATUS, RP0
        andlw   0x01
        btfss   STATUS, Z
        bsf     DATA
        btfsc   STATUS, Z
        bcf     DATA
        Delay   21
        bcf     CLOCK
        Delay   45
        bsf     CLOCK
        bcf     STATUS, RP0
        Delay   5
        return

;-------------------------------------------------------------------------------------------
```

```
;       READ ONE BYTE: (Takes about 1ms) - TIMING IS CRITICAL!!!
;-------------------------------------------------------------------------------------------

ByteIn          btfss  CLOCK           ;Test for Request-to-send
                retlw 0xFE
                btfsc  DATA
                retlw 0xFE
                movlw       0x08
                movwf       COUNTER
                clrf   PARITY
                Delay       28
ByteInLoop      call   BitIn        ;Data bits
                btfss  CLOCK           ;Test for inhibit
                retlw 0xFE
                bcf    STATUS, C
                rrf    RECEIVE, f
                iorwf RECEIVE, f
                xorwf       PARITY,f
                decfsz      COUNTER, f
                goto   ByteInLoop
                Delay       1
                call   BitIn        ;Parity bit
                btfss  CLOCK           ;Test for inhibit
                retlw 0xFE
                xorwf       PARITY, f
                Delay       5
ByteInLoop1     Delay       1
                call   BitIn        ;Stop bit
                btfss  CLOCK           ;Test for inhibit
                retlw 0xFE
                xorlw0x00
                btfsc  STATUS, Z
                clrf   PARITY
                btfsc  STATUS, Z       ;Stop bit = 1?
                goto   ByteInLoop1     ;    No--keep clocking.

                bsf    STATUS, RP0     ;Acknowledge
                bcf    DATA
                Delay       11
                bcf    CLOCK
                Delay       45
                bsf    CLOCK
                Delay       7
                bsf    DATA
                bcf    STATUS, RP0
```

```
            btfss  PARITY, 7 ;Parity correct?
            retlw 0xFF          ;    No--return error

            Delay       45
            retlw 0x00

BitIn       Delay       8
            bsf    STATUS, RP0
            bcf    CLOCK
            Delay       45
            bsf    CLOCK
            bcf    STATUS, RP0
            Delay       21
            btfsc  DATA
            retlw 0x80
            retlw 0x00

;-------------------------------------------------------------------------------------------
;       DELAY:
;-------------------------------------------------------------------------------------------
;Delays 4w+4 cycles (including call,return, and movlw) (0=256)
Delay_us    addlw       -1              ;Precise delays used in I/O
            btfss  STATUS, Z
            goto  Delay_us
            return

            end
```

附录六 其他资源/参考

上面文章中的一些链接在 PDF 中好像不能点击，我又放了一份在这里，共大家参考：

The AT Keyboard - My page on AT keyboards
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/keyboard/atkeyboard.html
The PS/2 Mouse - My page on the PS/2 mouse
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/mouse/mouse.html

Keyboard Scan Codes - My collection of scan code sets, verified in hardware.
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/keyboard/scancodes.html
PS/2 Mouse/Keyboard Protocol - Protocol used by AT and PS/2 keyboards.
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/PS2/ps2.htm
Keyboard Code/Projects - My keyboard projects and source code.
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/code/code.html

National Semiconductor - "Super I/O" chipset datasheets.
http://www.national.com/
IBM Archives - Non-technical historical information.
http://www-1.ibm.com/ibm/history/
Samtech, Holtech - Keyboard encoder datasheets.
http://www.samtech.com/
http://www.holtech.com/
Sci.Electronics.Repair - PC Keyboard FAQ.
http://www.repairfaq.org/
Holtek - Informative datasheets on many different PS/2 mice (and other peripherals).
http://www.holtek.com/products/computer/
EMC - More inormative datasheets on many different PS/2 mice (and an ADB mouse).
http://www.emc.com.tw/product/p_pc_mc.asp
Synaptics Touchpad Interfacing Guide -Very informative!
http://www.synaptics.com/decaf/utilities/tp-intf2-4.PDF
PS/2 Keyboard and Mouse Protocols - Timing diagrams.
http://www.networktechinc.com/ps2-prots.html
More links - Many more links to related information.
http://www-dev.ri.cmu.edu:8080/pub_files/pub1/brennemann_a_e_1995_2/brennemann_a_e_1995_2.pdf

Adam Chapweske's Homepage - Information about me.
http://panda.cs.ndsu.nodak.edu/~achapwes/
More Links - Many more links to related resources.
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/Links.html
Email me - Questions/comments?
mailto:achapwes@panda.cs.ndsu.nodak.edu
还有我的：shouxj@sohu.com 或者去 c51bbs.com 找我吧。