```
 1 #define IDLE_WAIT_NO 1000
 2
 3 // include files
 4 #include <io.h>                // for I/O control
 5 #include "const.h"             // constant definition
 6 #include "mouse.h"             // mouse control constant definition
 7
 8 // functions prototype
 9 #include "avrps2.h"
10
11 unsigned char put_mouse(unsigned char out_data) // マウス制御コマンド送出
12 {
13     unsigned char i, oddparity, ret;
14
15     oddparity = 0;
16     PS2CLK_LOW;                 //[1] PS/2 clock signal drive (0)
17     wait_60usec();              //    wait 60 [uSec]
18     PS2DATA_LOW;                //[2] start bit
19     PS2CLK_HIGH;                //[3] PS/2 clock signal drive end
20     wait_psclk_posedge();       //[4]
21     wait_psclk_negedge();       //[5]
22     for(i=0; i<8; i++) {        //    data out (LSB -> MSB)
23         if(out_data & 0x01) {   //[6][9]...
24             PS2DATA_HIGH;
25             oddparity++;
26         } else {
27             PS2DATA_LOW;
28         }
29         out_data >>= 1;
30         wait_psclk_posedge();   //[7]..[10]
31         wait_psclk_negedge();   //[8]..[11]
32     }
33     if(oddparity & 0x01) {      //[12] odd parity bit
34         PS2DATA_LOW;
35     } else {
36         PS2DATA_HIGH;
37     }
38     wait_psclk_posedge();       //[13]
39     wait_psclk_negedge();       //[14]
40     PS2DATA_HIGH;               //[15] stop bit
41     wait_psclk_posedge();       //[16]
42     wait_psclk_negedge();       //[17]
43     if(PS2DATA_EQ_HIGH) {       //[18]
44         ret = ERROR;
45     } else {
46         ret = NO_ERROR;
47     }
48     wait_psclk_posedge();       //[19]
49     return(ret);
50 }
51
52 unsigned char get_mouse1(unsigned char *data)   // マウス・データ取得(１バイト)
53 {
54     int i;
55     unsigned char status;
56     unsigned char temp;
57     unsigned char parity;
58
59     if(!(PS2CLK_EQ_HIGH)) {     //(1) if PS/2 clock == LOW
60         *data =  0x0E;          //      then over run error
61         return(OVERRUN);
62     }
63     temp   = 0;
64     status = 0;
65     parity = 0;
66
67     wait_psclk_negedge();       //(2) 1st negative edge of mouse clock
68     if(PS2DATA_EQ_HIGH) {       //(3) start bit check
69         data = 0;
70         return(FRAME);
71     }
72
73     for(i=0; i<8; i++) {
74         wait_psclk_posedge();   //(4)..(7)
75         wait_psclk_negedge();   //(5)..(8)
76         temp >>= 1;
77         if(PS2DATA_EQ_HIGH){    //(6)..(9) data bit sample
78             temp |= 0x80;
79             parity++;
80         }
81     }
82
83     wait_psclk_posedge();                       //(10)
84     wait_psclk_negedge();                       //(11)
85     parity += (PS2DATA_EQ_HIGH) ? 1 : 0;        //(12) parity bit
86
87     wait_psclk_posedge();                       //(13)
88     wait_psclk_negedge();                       //(14)
89     status |= (PS2DATA_EQ_HIGH) ? 0 : FRAME;    //(15) stop bit
90     status |= (parity & 0x01)   ? 0 : PARITY;
```

```
 91     *data = temp;
 92     wait_psclk_posedge();                       //(16)
 93     return(status);
 94 }
 95
 96 unsigned char get_mouse3(unsigned char data[])  // マウス移動データ取得(３バイト)
 97 {
 98     unsigned char status;
 99     long int count;
100
101     count = 0;
102     while(count<=IDLE_WAIT_NO) {        // overrun check
103         if(PS2CLK_EQ_HIGH) {    //
104             count++;
105         } else {
106             count = 0;
107         }
108     }
109
110     status  = get_mouse1(&data[0]);     // input 3-byte from mouse
111     status |= get_mouse1(&data[1]);
112     status |= get_mouse1(&data[2]);
113     return(status);
114 }
115
116 unsigned char init_PS2(void)                    // PS/2マウス初期化
117 {
118     unsigned char data1, data2, data3, status;
119
120     status = put_mouse(0xFF);   // initilize command
121     get_mouse1(&data1);         // 0xFA (ack. from mouse)
122     get_mouse1(&data2);         // 0xAA (self test OK)
123     get_mouse1(&data3);         // 0x00 (mouse ID)
124     return(status);
125 }
126
127 unsigned char start_PS2(void)                   // ストリーミング・モード開始
128 {
129     unsigned char data1, status;
130
131     status = put_mouse(0xF4);   // start streaming mode command
132     get_mouse1(&data1);         // 0xFA (ack. from mouse)
133     return(status);
134 }
135
136 void wait_psclk_posedge(void)                   // マウスクロック↑ウェイト
137 {
138     while(!(PS2CLK_EQ_HIGH)) { }
139 }
140
141 void wait_psclk_negedge(void)                   // マウスクロック↓ウェイト
142 {
143     while(PS2CLK_EQ_HIGH) { }
144 }
```