International Workshop on Automobile, Power and Energy Engineering

The IIC interface based on ATmega8 realizes the applications of PS/2 keyboard/mouse in the system

Dai Jiejie\textsuperscript{a}\textsuperscript{*}, Song Hui\textsuperscript{b}

\textsuperscript{a}School of Automation Science and Electrical Engineering, Beihang University New Main Building D645, Beijing 100191, China
\textsuperscript{b}School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Abstract

This paper introduces the PS/2 keyboard/mouse protocol and the transmission characteristics of IIC bus interface, giving the design and hardware circuit for converting the PS/2 keyboard/mouse interface to IIC interface. System analyzes protocols and data, and then transmits the data on the basis of PS/2 keyboard protocol by the ATmega8 microcontroller to the main control system. Thereby data from PS/2 keyboard/mouse is handled and output by the ATmega8 microcontroller to other devices, providing a feasible solution for the next step to achieve the keyboard/mouse interaction with the host.

© 2010 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Society for Automobile, Power and Energy Engineering

Keywords: PS/2 keyboard/mouse; IIC interface; ATmega8 microcontroller; data transmission

1. Introduction

AVR single-chip microcomputers are widely used in process control, data acquisition and other fields because of its large program memory, low price, fast processing and powerful process ability [1]. Besides, as PS/2 keyboard/mouse has a good serviceability, high stability and versatility, it has been widely used in microcomputer and project. This paper introduces an implementation about data transition between keyboard and host computer based on Atmel’s ATmega8 microcontroller.

\textsuperscript{*} Corresponding author. Tel.:+1-591-118-5077; fax: +0-108-231-6564.
\textit{E-mail address}: secess@163.com.
2. PS/2 Protocol

PS/2 Protocol is a type of Synchronous bidirectional serial protocol used in communication between peripheral and host. Keyboard/mouse can send data to the host and the host can send data to the device, but the host always has priority over the bus. All data is arranged in bytes, each byte includes 11~12 bits: 1 start bit (0), 8 data bits, 1 odd parity bit, and 1 stop bit (1). Data sent from the keyboard/mouse to the host is read on the falling edge of the clock signal (when Clock goes from high to low); data sent from the host to the keyboard/mouse is read on the rising edge (when Clock goes from low to high). When the host sends the data to the keyboard/mouse, a handshaking bit is sent from the device to acknowledge the packet was received. This bit is not present when the device sends data to the host.

2.1. Peripheral sent host receive

When the keyboard or mouse wants to send information, it first checks Clock to make sure it’s at a high logic level. If it's not, the host is inhibiting communication and the device must buffer any to-be-sent data until it regains control of the bus. If the Clock line is high, the device can begin to transmit its data. Sequence is shown in figure 1.

![Fig. 1. Timing diagram while peripheral is sending](image)

2.2. Host sent peripheral receive

First of all, the PS/2 device always generates the clock signal. If the host wants to send data, it must first put the Clock and Data lines in a "Request-to-send" state as follows: Inhibit communication by pulling Clock low for at least 100 microseconds; Apply "Request-to-send" by pulling Data low, then release Clock. The device should check for this state at intervals not to exceed 10 milliseconds. The host changes the Data line only when the Clock line is low, and data is latched on the rising edge of the clock pulse. This is opposite of what occurs in device-to-host communication. The sequential relationship in the process is shown in figure 2 below [2].

![Fig. 2. Timing diagram while controller is sending](image)

2.3. Keyboard working principles

Keyboard-processor scans or monitors the matrix of keys. If any key is being pressed or released, the keyboard will send a packet of information called a "scan code" to the computer. There are two different types of scan codes: “make code” (1-byte connects scan code) and “break code” (2-bytes break scan
code). Every key is assigned its own unique make code and break code so the host can determine exactly what happened to which key by looking at a single scan code. Scan code associated with key position, and have nothing to do with that key ASCII code.

2.4. Mouse working principles

PS2 mouse has four standard modes of operation. (1) “Reset” mode: The mouse enters Reset mode at power-up or after receiving the "Reset" (0xFF) command from the host. (2) "Stream" mode: After power-up or Reset finishes executing, the mouse will enter this default mode. (3) “Remote” mode: When the host sends "Set Remote Mode" (0xF0) command to the mouse, the mouse enters this mode. (4) “Wrap” mode: This mode is used to test whether the connection between the mouse and the host is correct.

When the mouse moves by a minimum distance or a key is pressed, the mouse will send the distance and the state of buttons to the host by one or more messages. This information includes: initialize report, the direction of movement, the distance and button state [3].

3. IIC Interface

The IIC protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the IIC bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the IIC protocol. Each data bit transferred on the IIC bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions. All address packets transmitted on the IIC bus are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed; otherwise a write operation should be performed. An address packet consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively. During a data transfer, the Master generates the clock and the START and STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signaled by the Receiver pulling the SDA line low during the ninth SCL cycle. A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. Figure3 shows a typical data transmission [4].

![Fig.3. Typical data transmission](image)

4. System Design

4.1. Hardware principle

In this paper, we use the ATMEL Company’s ATmega8 microcontroller to communicate with PS/2 keyboard (mouse). We will take keyboard’s communication for example and its hardware schematic is shown in Figure 4. Because PS/2 interface uses a synchronous bidirectional serial protocol and CLK
signal is always generated by the keyboard, the following two programs can be adopted: one is that the keyboard clock line connects to the microcontroller’s interrupt line and the data line connects to microcontroller’s any one of I/O ports, to achieve bidirectional communication by querying the state of DATA bit. Another is that do not changes the position of the keyboard clock line CLK connection, while the data line connects to the microcontroller’s RXD (also for I/O port). When the microcontroller sends data, MCU external interrupt is triggered by CLK, and then use RXD as its I/O port function for data transmission; when the keyboard sends data, the I/O port will switch to the RXD function, using synchronous/asynchronous serial interface USART of the MCU to receive data. Thus, the host doesn’t generate an interruption until it receives a data. Compared with the second option, although there are more frequent interrupts, the first realization doesn’t need to consider the adaptive chip baud rate nor relatively complicated hardware circuit design. So the first program is chosen.

CLK line and DATA line are respectively connected to MCU’s INT0 (PD2) and INT1 (PD3). IIC Interface SDA and SCL are connected to the main control system. Enable INT0 and shield INT1 (INT1 as the general I/O pin) when the system is on working.

Fig.4. Keyboard communication schematic

4.2. Software realization

PS/2 interface program flow chart is as follows:

Fig.5. PS/2 keyboard communication flow chart

4.3. Part of routine design

This program adopts CodeVision AVR C language compiler. When the keyboard gives an input, CLK’s falling edge triggers external interrupts INT0, the program receives the data sent by the keyboard in External Interrupt subroutine. When the clock and data line are both high at the same time, the data transmits is turned on. The start bit (0) is output to the data line (to note here is; after sending a bit, it needs to check Clock to make sure that the host doesn’t inhibit the PS/2 device. If is inhibited, transmit
will be stopped). The keyboard output 8-bit data to the data line then check odd parity [5]. If parity is decoded correctly, it will finally output the stop bit (1). The programs of receiving 8-bit data and checking odd parity are as following:

```c
If (bBitCounter > 1 && bBitCounter < 10)
{
    bKeyCode >>= 1; //receive 8-bit data
    If (DAT){bKeyCode |= 0x80; bOddParity++;} return;
} Else if (bBitCounter == 10)
{If (bKeyCode ^ (bOddParity & 0x01)) //odd parity correct
    oddstatus=1; //odd parity correct flag
    PS/2_KeyDeCode (bKeyCode);
    PSDAT=PS/2KeyStatus.bKeyAsiic; }
Else { bBitCounter = 0 ;} return; }
```

In accordance with the I2C protocol, data is transferred to the main control chip by the I2C bus.

```c
Void I2C_W (unsigned char PSDAT)
{TWCR = (1 << I2CNT)|(1 << TWSTA)|(1 << TWEN); //clear Interrupt, send Start condition
    While (! (TWCR & (1 << I2CNT))); // wait for completion of Start condition
    TWDR = SLAW_XX; //send Write operation
    TWCR = (1 << I2CNT)|(1 << TWEN); //do the write operation
    While (! (TWCR & (1 << I2CNT))); // wait for completion of Write operation
    TWDR = PSDAT; //Read the data
    TWCR = (1 << I2CNT)|(1 << TWEN); //start sending
    While (! (TWCR & (1 << I2CNT)));
    TWCR = (1 << I2CNT)|(1 << TWEN)|(1 << TWSTO); } //stop
```

5. Conclusion

When involved with text input, value modification, liquid crystal display and some other operations, we usually has to design the keyboard circuit, interface and control program in a traditional single chip microcomputer control system, which not only occupy hardware resource but also cost a great deal of software resource. This paper designs a PS/2 interface circuit and use the keyboard to control interrupt in the program, occupying less system resources and having a lower cost. This paper provides a convenient implementation for a variety of keyboard control and visualization of operation.

Acknowledgements

We gratefully acknowledge the support and collaboration of Professor Chen Shenggong in this project.

References
