To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

H8S/2117

PS2 Mouse and Keyboard Sample Program

Introduction

This application note accompanies the PS2 mouse and key -board sample code for the H8S2117 development board. The software was developed and tested on the YTD08EV211701 (Common KBC Development board).

Target Device

H8S2117

Contents

1. Overview 2

2. The PS2 Standard... 2

3. Program Description... 2

4. Using the program in your application... 5

1. Overview

The PS2 sample program is a demonstration of how to configure the block and create a driver that can detect PS2 device types, parse scan codes and notify higher layers of different PS2 events. The program is written so that a single PS2 port can be used for either a mouse or a keyboard. The program is capable of changing buffer sizes and parsing the scan codes depending on the device plugged in. The PS2 block supports a total of three channels; the code is written for channel 1, but can be easily reconfigured to run for any of the other two channels.

Compile and download the program to the MCU with an E10A. Open a watch window, add the variable ps2_rxbuf[] to it and enable real-time variable update by clicking on the "R" above the watch window. Once the program is running, connect a mouse or keyboard to the PS2 port (channel 1). If a keyboard is connected, pressing keys will display corresponding make and break codes in the buffer. If a mouse is connected, moving it about will display mouse packets in the buffer.

The program determines if a keyboard or mouse is plugged in on start-up, so in order to use a different device, reset and run the sample code before plugging in the new device.

2. The PS2 Standard

PS2 is a widely used keyboard and mouse communication standard developed by IBM. Even with the advent of USB based systems, the PS2 still remains a commonly used interface. The connector is either the standard 5 pin DIN on the 6 pin mini-DIN, both of which are electrically similar. The signals present on the connector are Clock, Data, Ground and Vcc. The order is different for the 5 pin and 6 pin connectors.

The clock and data lines are bidirectional and logic high is the "idle" state. The PS2 protocol is serial synchronous and the electrical interface is designed to prevent collision between the host and device packets during communication. In a PS2 system, the clock (typically 10 KHz) is generated by the device although the host has ultimate control over operation. In a typical system, the host is a PC with the device being either a keyboard or mouse. A data packet has 11 bits with 1 start bit, 8 data bits, 1 parity bit and 1 stop bit.

Communication arbitration between the host and device is performed based on the state of the clock and data lines.

There are essentially three states used:

a. Data and clock lines are high: This is the "idle" state and as such the system available for use by the slave or host.

b. Data line is high and clock is low: This is called "inhibited" state. This is usually enforced by the host firmware by pulling down the clock line. In this state, no further communication takes place until the host releases the clock line. The does may do this to prevent the device from sending any more packets while processing previously received packets from the device

c. Data line is low and clock line is high: This known as the "Request to send" state and is performed by the host. The standard sequence is to pull down the clock line for a 100 microseconds, pull down the data line and then release the clock line. This lets the device know that the host wants to send information to it. The device will then cease communication and start generating the clock required by the host to send the data packets.

2.1 Communication Protocol

On power on or on receiving the Reset command, the device performs the Basic Assurance Test and sends out either a BAT Success (0xAA) or a BAT Failure (0xFC) code to the host. In mouse devices, this is followed by the Mouse ID packet which is 0x00 for standard mice and 0x04 for the Microsoft Intelli-mouse. Most commands to the device are acknowledged with an ACK (0xFA) packet. There are defined commands that the host can send to reset the device, enable or disable operation configure operational parameters etc.

For detailed information on the PS2 standard and operation, refer to [1]

3. Program Description

Main routines in the sample program

a. Main (Application code)

b. PS2 Initialization.

c. Tick timer initialization: 1 millisecond tick timer used as time base for timeout routine.

d. SendCmd: Send commands to the PS2 device

e. Alarm: Configurable timeout routine with max timeout of 65536 milliseconds

f. Channel 1ISR: Receives and parses data from the PS2 device

The program essentially has two layers, one which handles the data transmission and reception part and the other which handles the higher level state machine for command transfer and response decoding.

The lower layer is mainly composed of two functions namely the SendCmd() function and the ISR.

The SendCmd() function takes the one byte command to be transmitted as argument and returns a success or failure status for the transmit operation. The function does not check if the device actually responds with an ACK; that has to be done by the application layer.

The ISR parses received data packets and assembles them into scan codes. Since the scan code size varies depending on the transaction type (ACK, device ID, make code, break code) and device type, the program takes this into account and only notifies the application layer once a complete scan code for the particular device or transaction type is assembled. The ISR then holds down the clock line and sends a notification via the PROCESS_REQ bit so that the application layer can process the assembled scan code. The Clock line has to be released by the application layer once the scan code is processed. Currently the program is limited to assembling scan codes for the standard keyboard, standard mouse and Intelli-Mouse.

The application layer uses the SendCmd() function and the ISR notification to establish communication with the device. The application layer sends commands, and checks for proper response from the device. This layer also has to process the assembled scan-codes.

Communication between the driver layer and the application layer is established by way of three variables.

a. uchar PS2_State: Individual bits in this variable are used to convey error messages and process requests to the application layer. The application layer has to clear the corresponding bits after processing them

PROCESS_REQ	READY	BAT_FAILED	NO_DEVICE	UNUSED	TX_ERR	STOP_ERR	PARITY_ERR

Figure 1: PS2_State bit-field definition

b. uchar PS2_ rxbuf[]: This is the data structure that is populated with the make and break codes. The PROCESS_REQ bit in PS2_State is asserted once a scan code is assembled

c. scan_code_size: The size of the assembled make or break code is conveyed in the variable.

The program is configured to run on Channel 1, but can be easily configured to run on the other two channels on this device.

There is also a tick timer function provided by the Timer0. It is currently configured for a 1 millisecond tick, and is used as the timer based for the alarm() timeout routines.

3.1 Pseudo-code

3.1.1 Driver layer

This layer consists of the PS2 ISR and the SendCmd() function

Pseudo-code for the ISR:

```
Check for any errors (Stop Bit, Parity Bit) in the received data.
If there are errors
{
        set the appropriate flags in the KBD_State variable and set the Process Request bit for the application layer.
}
else if there are no errors
{
        Read device_type to determine if the device type is known.
        If device type is known
        {
```

```
        parse the byte so that a packet can be assembled based on the device type
    }
    If not
    {
        then set the scan code size to a default of 2
    }
    If a packet has been assembled
    {
        set the Process Request bit to notify the application layer.
        Hold the clock low so that further packets are not received until the application layer processes the
        packet
    }
    if more bytes are needed until a packet is assembled, then clear the interrupt request so that clock line is
    released to allow device to send more data
}
```

Pseudo-code for the SendCmd() function

```
Disable reception
Write the transmit data to the transmit buffer
Check to see if Clock and Data lines are both inactive (logic high)
if not
{
    terminate transmit operation since device is still sending data
    exit function with a failure return code
}
Pull down clock line, then pull down data line and then release clock line to indicate to device that host needs to send
data.
Wait with timeout for data transmission to complete.
Re-enable data reception
exit from the function with a success return code
```

3.1.2 Application layer

Pseudo-code for the main loop

```
Initialize tick timer and the PS2 block

do this forever
{
    Send PS2 device reset command until a device responds with ACK and successful power on self test (BAT)
    code
    if device responds with the proper BAT code
    {
        get the device_ID to identify the device as a mouse or keyboard or intellimouse.
        update the device_type variable with this information
        if the device is a mouse
        {
            then send out the Enable_reporting command so that the mouse streams data packets.
        }
        do this forever
        {
            if the lower layer has asserted the Process_request bit
            {
                If any error bits were set, ask device to resend the byte and increment the error
                count.
                clear the Process_request bit and release the Clock line so that the device can send
                data
            }
            if the error count has exceeded a present limit, break out of this loop to the reset_device line
        }
    }
}
```

4. Using the program in your application

In its current state, the lower layer supports decoding of Keyboard, Standard Mouse and Microsoft® Intellimouse® packets. Once a complete packet (scan code, ACK, BAT code or Device ID) is assembled, a request flag is set for the application layer to process the received packet. The program also holds down the Clock line to prevent any further data packets from being sent by the device until the previous packet is processed. The software is written in a non-blocking manner with timeout routines.

The main program is an example of how one would use the program in a PS2 application. The main routine can be created as a separate thread if running in a multithreaded environment. The thread will periodically check to see if the lower layer has asserted the process packet request, and act accordingly.

To switch operation to a different channel, change the #define KBC_Channel accordingly and move the ISR to the corresponding vector. To add the functions to a separate application, include the PS2_functions.c and PS2_includes.h in the project. The ISR from the intprg.c will need to be moved over as well. Some functions that handle error processing under different conditions are left as stubs and may be populated depending on the application requirements.

Reference

1. www.computer-engineering.org/

Website and Support

Renesas Technology Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record <revision history,rh>

Rev.	Date	Description	
		Page	Summary
1.00	Oct.1.09	—	First edition issued