PHILIPS

Philips Semiconductors

Interconnectivity

**2 December 1998**

# Application Notes

# USB Keyboard using PDIUSBH11A

Philips Semiconductors - Asia Product Innovation Centre
Visit http://www.flexiusb.com

**Disclaimers:**

**Life Support** – These products are not designed for use in life support appliances, devices or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling thes products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

**Right to make change** – Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products and makes no representations or warranties that these products are free from patent, copyright or mask work right infringement, unless otherwise specified.

**Application information** – Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further test or modification

LICENSES

**Purchase of Philips I$^2$C components**

Purchase of Philips I$^2$C components conveys a license under the Philips I$^2$C patent to use the components in the I$^2$C system provided the system conforms to the I$^2$C specification defined by Philips. This specification can be ordered using the code 9398 393 40011.

# Application Notes: USB Keyboard Hub using PDIUSBH11A

## Introduction

The PDIUSBH11A (H11A) implements a USB Hub and USB interface. It can be interfaced to a micro-controller using a minimum of 3 I/O lines. These are a pair of I$^2$C lines and an interrupt line.

The H11A can be configured in two modes:

Mode 1 - as a Hub + single function.
Mode 2 - as a Hub + three embedded functions.

These application notes describe the Mode 1 configuration. There are a total of 4 endpoints for Mode 1:

Endpoint O (the default control endpoint) and endpoints 1-3 are generic. All four endpoints can be used as either interrupt, bulk or control endpoints. Each endpoint has a buffer size of 8 bytes.

In Mode 1, only the default control endpoints and an interrupt endpoint are used.

### HARDWARE DESCRIPTION

The USB Keyboard Hub makes use of Philips H11A and 8051 Micro-Controller (MCU). The USB Keyboard Hub supports one USB upstream and 4 downstream ports. The keyboard matrix implements 16 output and 8 input MCU lines, as follows:

- I$^2$C communicates with H11A, using 3 I/O MCU lines.
- PS/2 communicates with the PC 2 I/O MCU lines.
- Num Lock, Caps Lock and Scroll Lock LEDs use 3 I/O MCU lines.

## Firmware description

Files description:

1. **kbhub213.asm** is the source file, containing all the routines.
2. **transctn.typ** contains constant values for various transaction codes, between the Hub and the Host.
3. **H11.cmd**. contains the I2C commands of the H11A.

## USB Class description

The firmware implements an HID-compliant composite device. This is a USB device with a single configuration and one interface. Endpoint 1 is used as the interrupt endpoint for key-pressed data. The report descriptor implements a boot-class keyboard.

## Flow Chart

The main loop polls the USB interrupt from the H11A; it also runs a routine to detect key-press events.

### a) Main loop

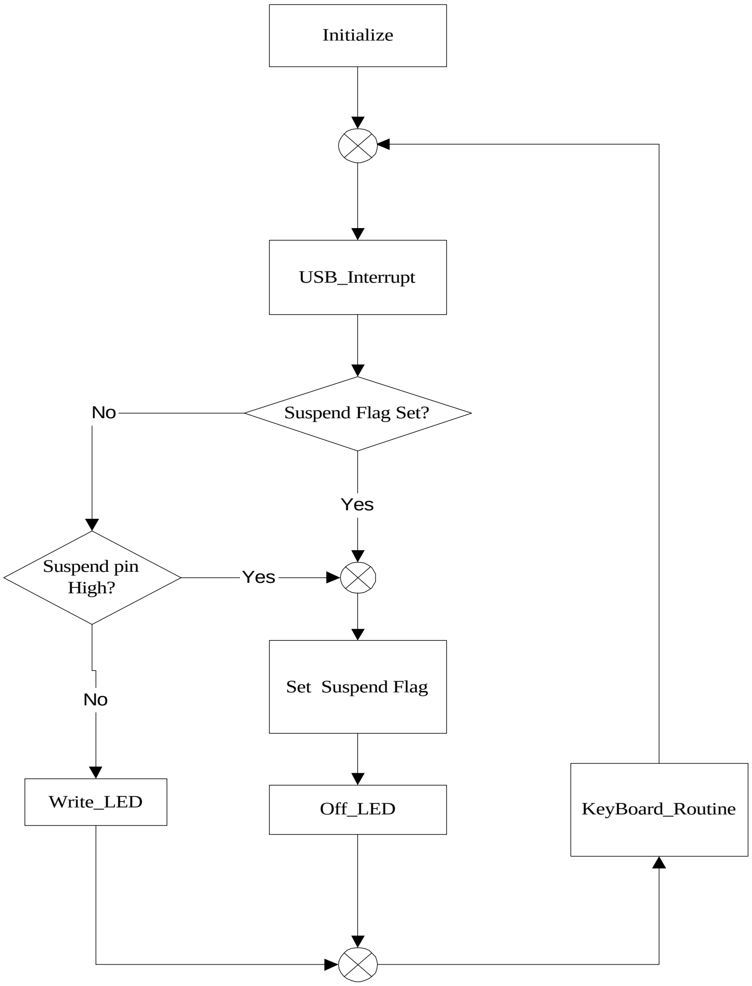


Fig. 1: MAIN ROUTINE

This routine initialises H11A and loops for H11A Interrupt pin to go LOW. The routine branches to USB_INTERRUPT on a HIGH-to-LOW transition on the Interrupt pin.

### b) USB Interrupt

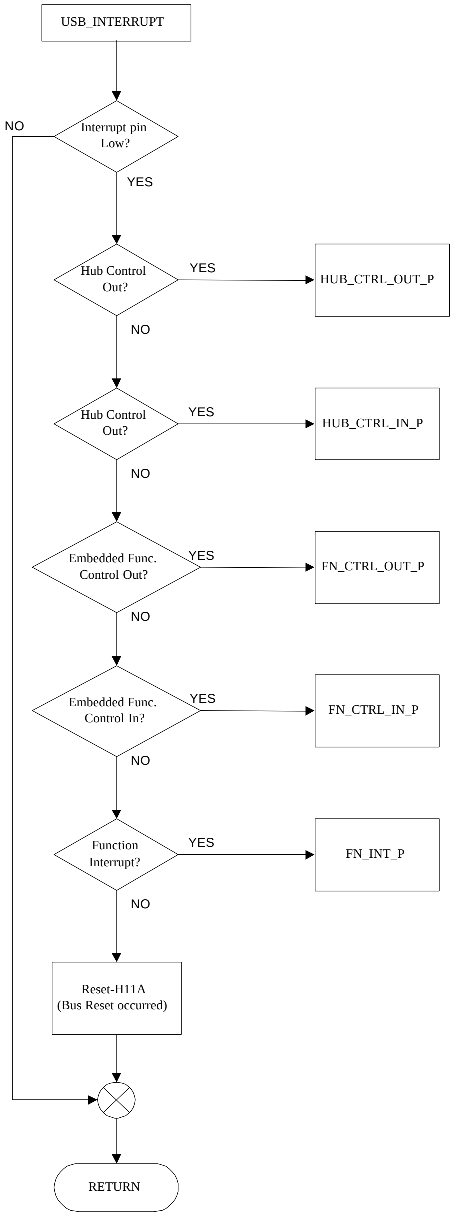


FIG. 2: USB INTERRUPT

This routine checks for the source causing the Interrupt pin to go LOW and branches to the respective routines. The source of the Interrupt can be from:
Hub_Control_Output_Endpoint
Hub_Control_Input_Endpoint
Embedded_Control_Output_Endpoint
Embedded_Control_Input_Endpoint
Embedded_Interrupt_Endpoint

### c) Hub Control OUT

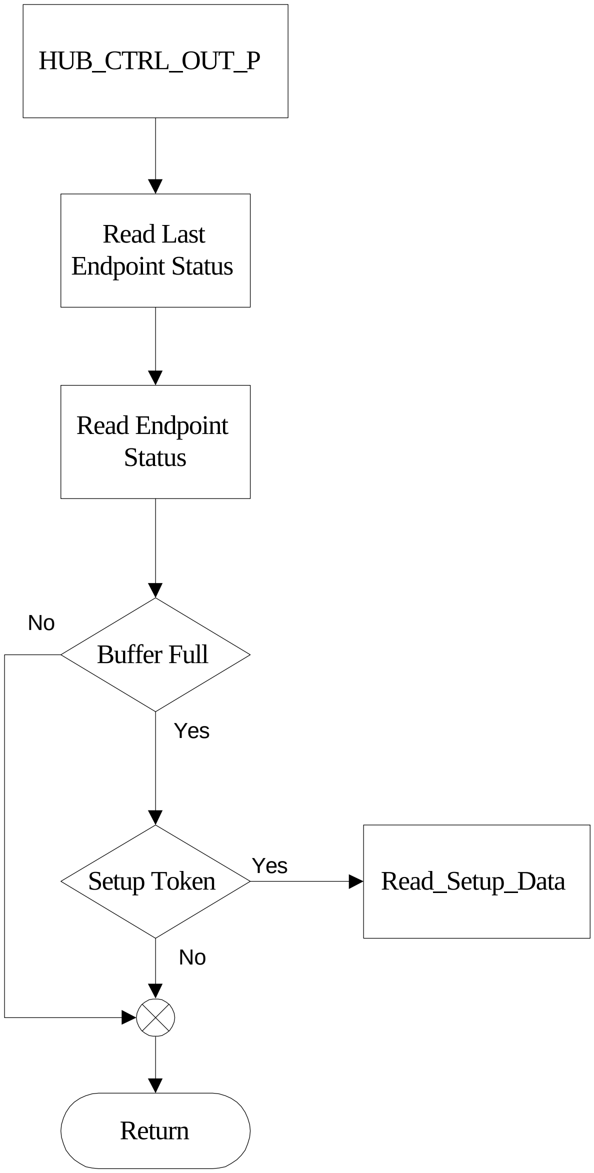


Fig. 3: HUB CONTROL OUT

This routine is executed when H11A receives a Setup packet on the upstream. The packet is rejected if all 8 bytes are not received, or if the received packet is not a Setup packet. When a Setup packet is received, the firmware branches to READ_SETUP_DATA to decipher the Setup packet and to execute respective routines.

### d) Hub Control IN routine

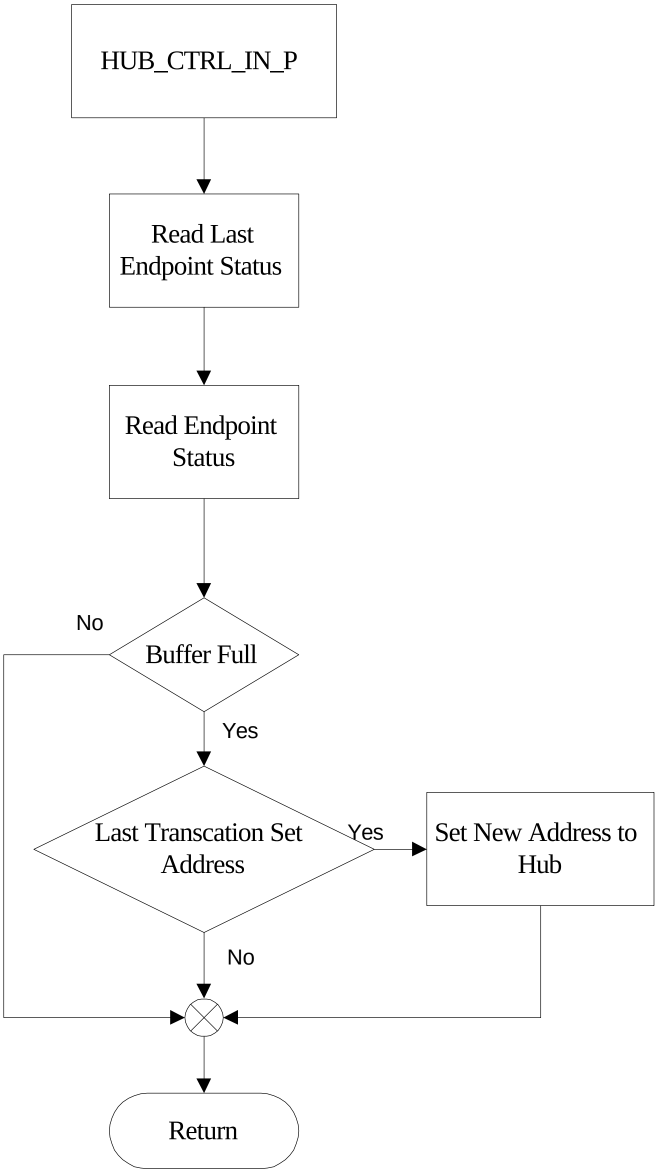


FIG 4: HUB CONTROL IN

This routine is executed when H11A receives an IN token on the upstream. The packet is rejected if all 8 bytes are not received. The firmware branches to MORE_MESSAGES if any data remains from the previous IN token. If the last transaction was SET_ADDRESS, the Hub address is changed to the new address during this transaction.

## e) Function Control OUT routine

The deciphered SETUP token is kept as the transaction code which is defined in the file **transctn.typ.** The actual data transfer is performed by the Function-Control-IN routine.

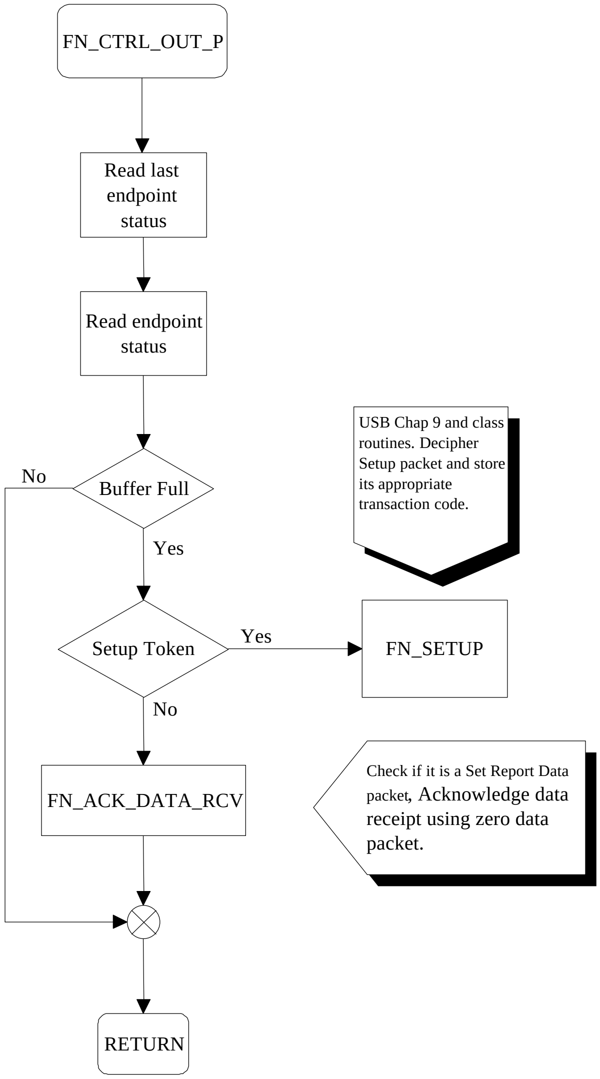

FN_CTRL_OUT_P
Read last endpoint status
Read endpoint status
USB Chap 9 and class routines. Decipher Setup packet and store its appropriate transaction code.
No
Buffer Full
Yes
Setup Token
Yes
FN_SETUP
No
FN_ACK_DATA_RCV
Check if it is a Set Report Data packet, Acknowledge data receipt using zero data packet.
RETURN

FIG 5: FUNCTION CONTROL OUT

This routine is executed when H11A receives a Setup packet on the upstream. The packet is rejected if all 8 bytes are not received, or if the received packet is not a Setup packet. When a Setup packet is received, the firmware branches to READ_SETUP_DATA to decipher the Setup packet and to execute respective routines.

## f) Function Control IN

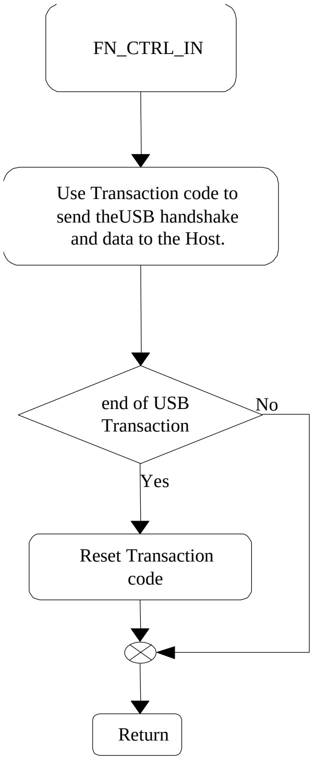
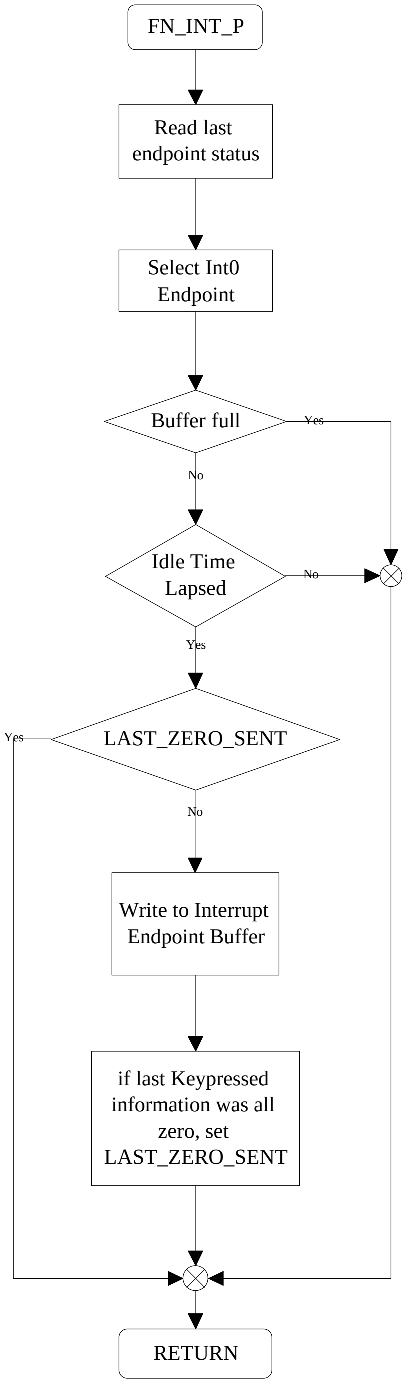

FN_CTRL_IN
Use Transaction code to send theUSB handshake and data to the Host.
end of USB Transaction
No
Yes
Reset Transaction code
Return

FIG 6: FUNCTION CONTROL IN

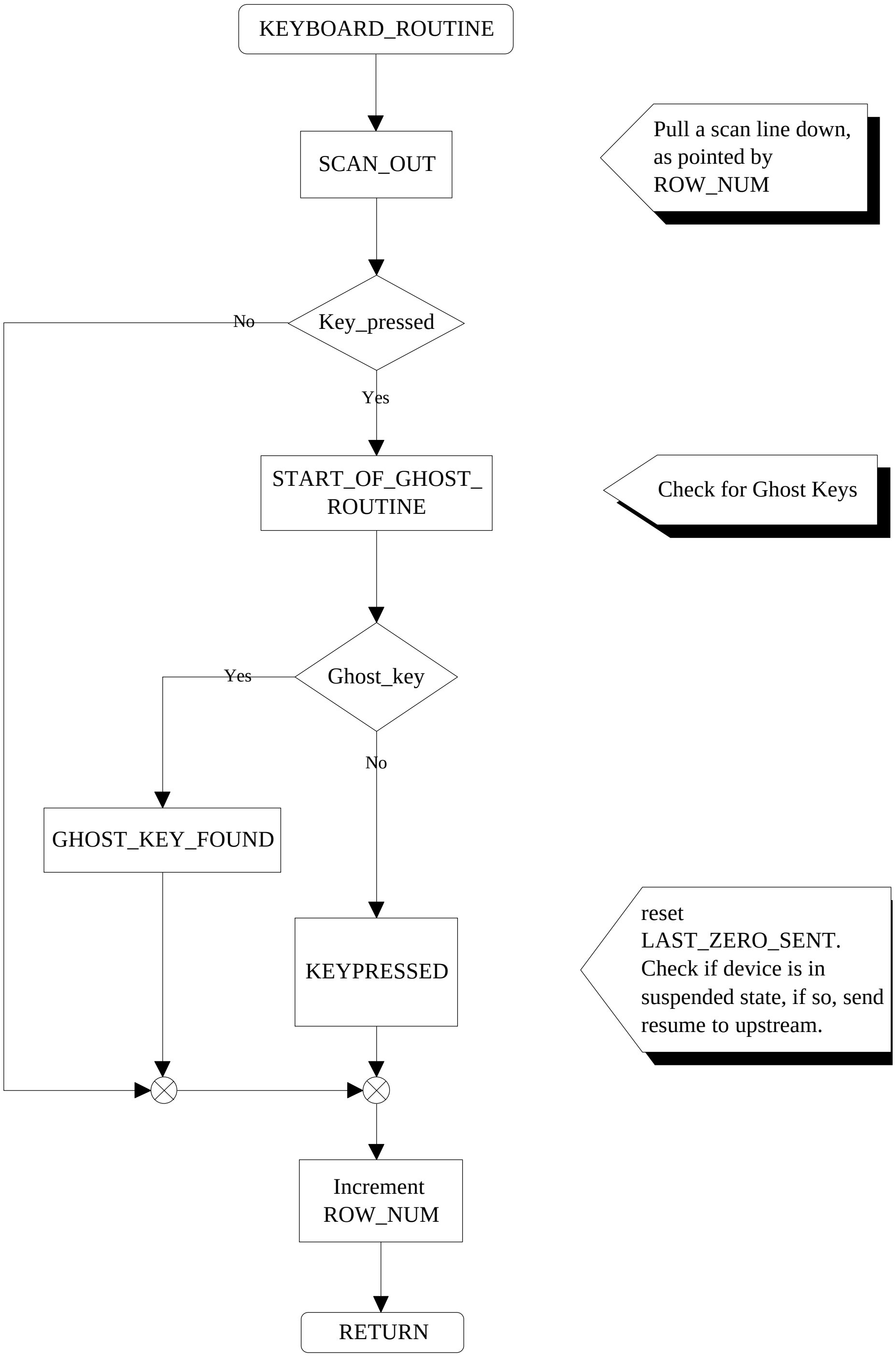
This routine is executed when H11A receives an IN token on the upstream. The packet is rejected if all 8 bytes are not received. The firmware branches to MORE_MESSAGES if any data remains from the previous IN token. If the last transaction was SET_ADDRESS, the Hub address is changed to the new address during this transaction.

## g) Function-Interrupt routine.

H11A should be configured to operate in Debug mode, to enable polling by the Host on a specific endpoint. fsIn the USB Keyboard Hub, one of the endpoints is configured as an Input with an Interrupt attribute. This means that H11A generates an Interrupt for every Input token received from the Host on this endpoint


FN_INT_P
Read last endpoint status
Select Int0 Endpoint
Buffer full
Yes
No
Idle Time Lapsed
No
Yes
LAST_ZERO_SENT
Yes
No
Write to Interrupt Endpoint Buffer
if last Keypressed information was all zero, set LAST_ZERO_SENT
RETURN

### h) The keyboard routine


KEYBOARD_ROUTINE
SCAN_OUT
Pull a scan line down, as pointed by ROW_NUM
Key_pressed
No
Yes
START_OF_GHOST_ROUTINE
Check for Ghost Keys
Ghost_key
Yes
No
GHOST_KEY_FOUND
KEYPRESSED
reset LAST_ZERO_SENT. Check if device is in suspended state, if so, send resume to upstream.
Increment ROW_NUM
RETURN