1 Oct 98

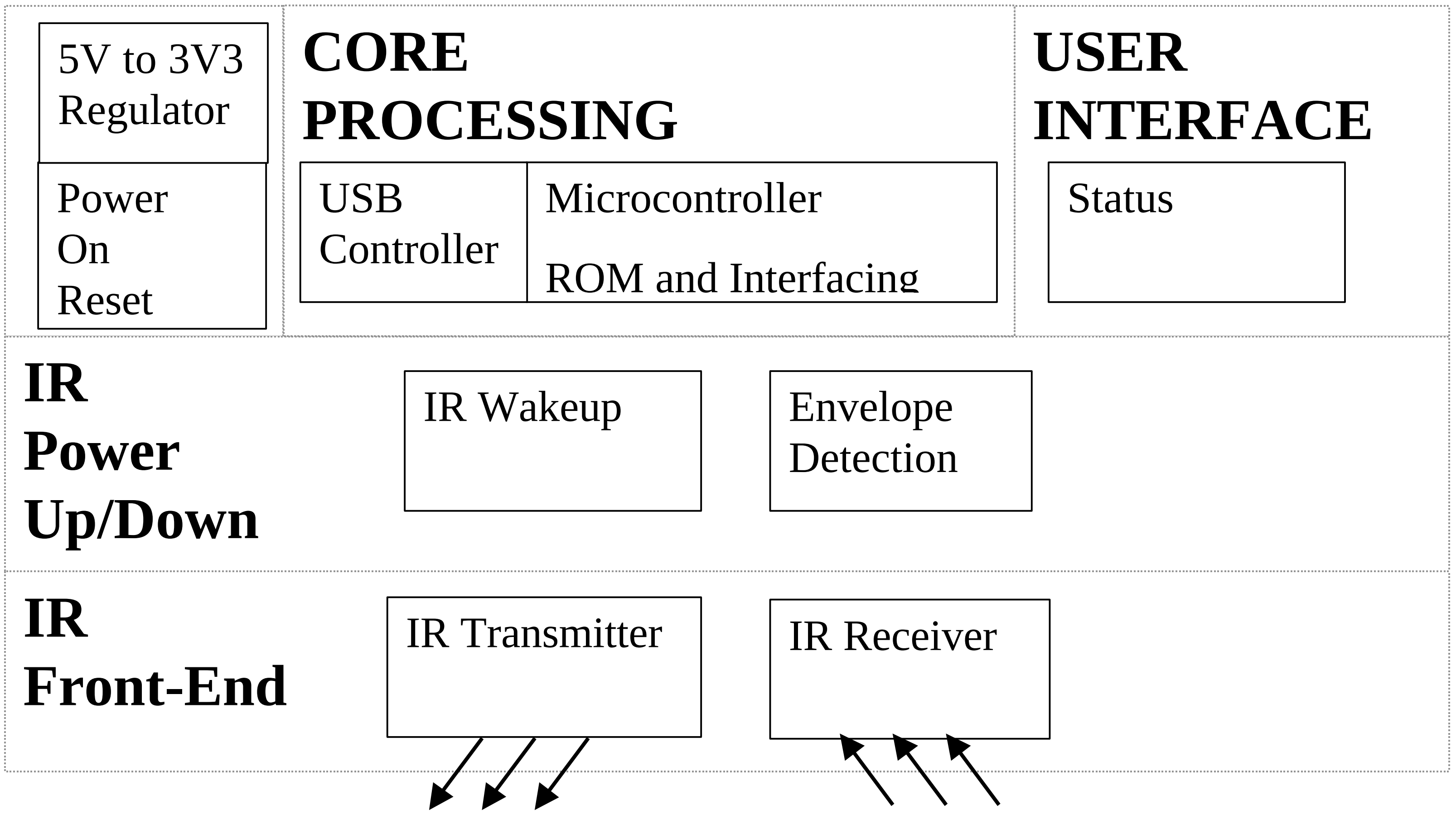
Implementing a USB-to-Infrared (Philips RCMM) Dongle

- USB IR HID Device
  (OVU1000, Reference Design Ver 1.1)

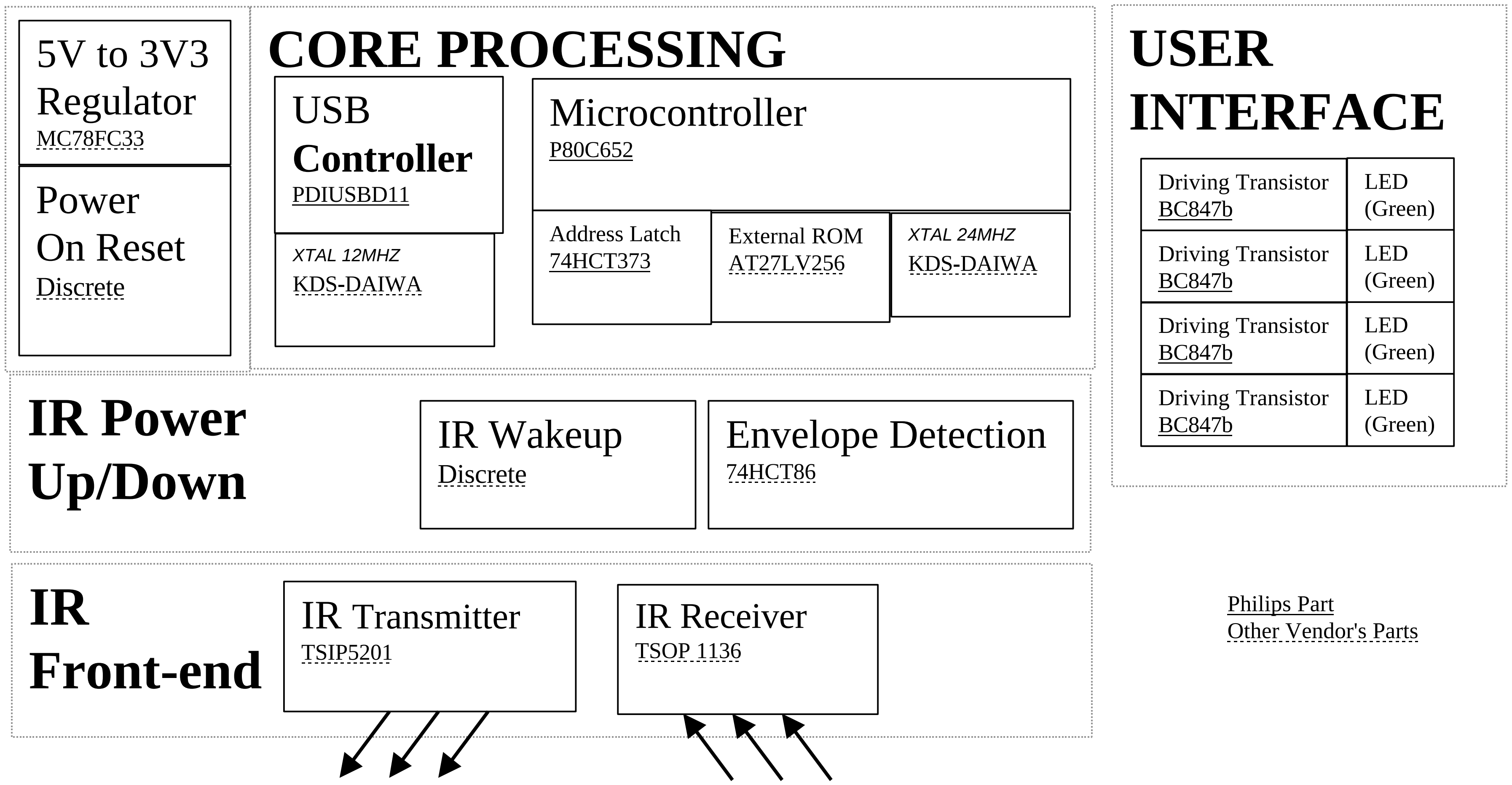
- USB Compound Hub with IR HID Device

Author: Wim, Lemay
Edited by: Wei Leong, Chui

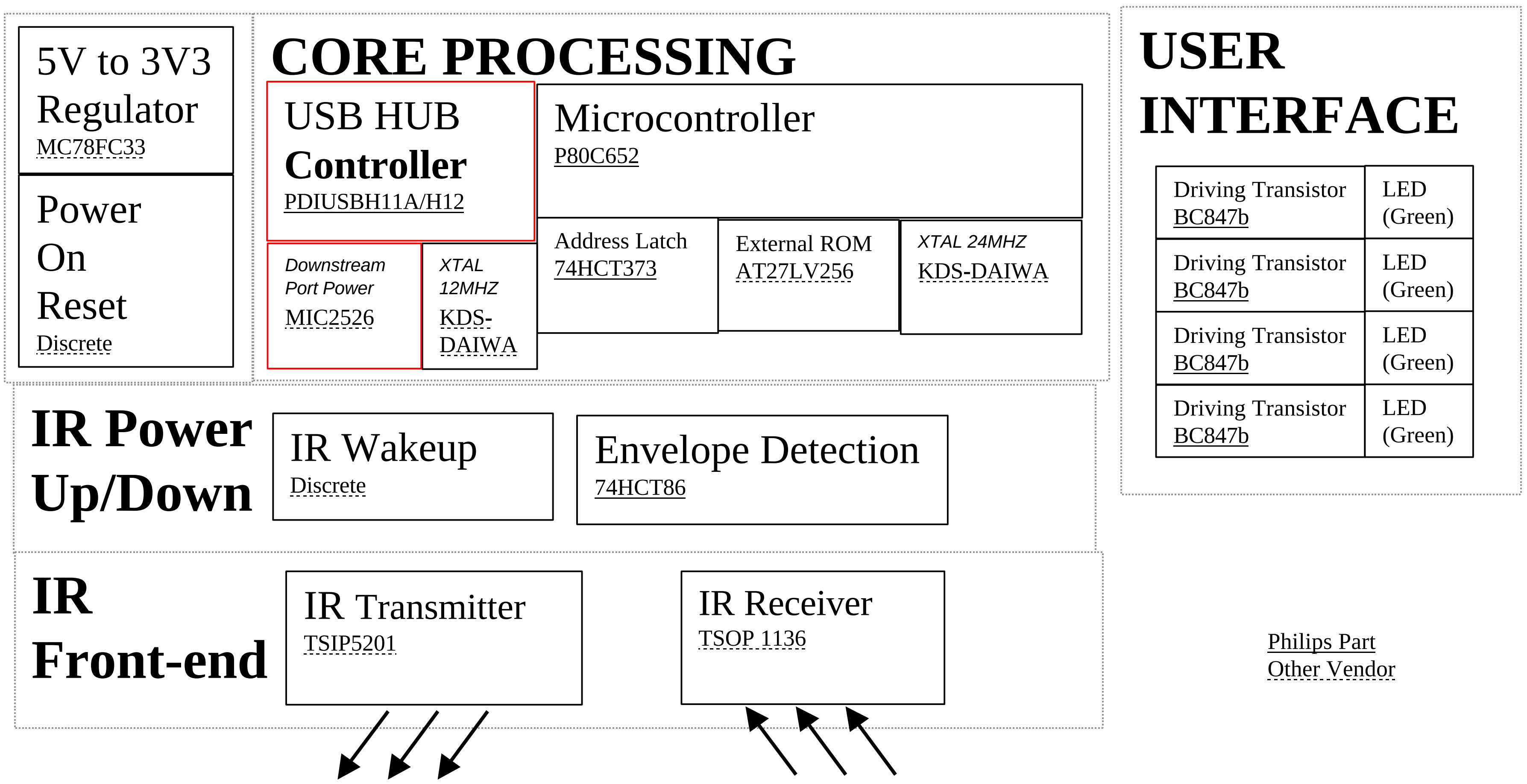
GENERAL HARDWARE BLOCKS FOR USB IR DONGLE



Design Blocks for USB IR Dongle: Infrared Keyboard/Mouse/Gamepad



DESIGN BLOCKS FOR USB COMPOUND HUB WITH IR HID SUPPORT – IR KEYBOARD/MOUSE/GAMEPAD



Philips Part
Other Vendor

TECHNICAL DESCRIPTION

GENERAL

This application note describes the implementation of
- A USB-IR dongle; and
- A USB compound hub with a USB-IR dongle device.

The main difference in the two designs lie in the USB Controller used. To implement a USB device, the PDIUSBD11 (D11) general-purpose USB interface is selected. To implement a compound hub with USB-IR functionality, you must use PDIUSBH11A (H11A) or PDIUSBH12 (H12). The device functionality is the same for all the designs, H11A provides 4 USB walk-up ports and the H12 provides only 2 USB walk-up ports.

The firmware implementation for the USB-IR dongle on the microcontroller can be divided into two parts. The microcontroller has to service the USB controller and at the same time decode and encode the data for the IR front-end. The hub functionality adds another 2 Kbytes of code onto the firmware.

The USB-IR dongle portion uses the D11/H11A/H12 as the USB interface for implementing HID functionality. This includes Keyboard, Mouse and Gamepad functionality.

For the USB-IR device only, it enumerates as a multi-interface composite device. The individual interfaces, each conforms to the HID Class definition, be it a HID Keyboard, a HID mouse or a HID gamepad. The default HID device driver from Windows 98 is loaded. Thus, the whole system may be implemented ready to ship without a need to develop your own device driver.

For the compound hub with the USB-IR device, the hub is first enumerated. The generic driver for Windows 98 is loaded before it enumerates the embedded USB-IR device. The enumeration of the USB-IR device follows the same procedure as explained in the previous paragraph.

The USB-IR device translates USB requests into IR signals conforming to the Philips Infrared RCMM (1-way) and Infrared RCMM^2 (2-way) protocols. The RCMM protocol is proprietary; an NDA would be required to obtain the protocol description. The contact person for the NDA is listed under the Contacts section.

Implementing a USB-to-Infrared Dongle

USB Interface Controller

D11, H11A and H12 are full-speed (12 MHz) USB Devices. Thus, a full-speed USB cable is required. The common mode chokes for EMI do not need to be added because both interface devices have built-in skew control for the D+/D- signaling lines. A termination resistor of 24 ohms for the transmission line is required. D11, H11A and H12 incorporate a SoftConnect™ internal resistor for the D+ pull-up detection.

The USB interface controllers run on 3.3V. A step-down voltage regulator from 5V to 3.3V is used. The microcontroller communicates with D11/H11A/H12 via I^2C. The command registers for D11/H11A/H12 may be obtained from the data sheets of the devices at www.flexiusb.com.

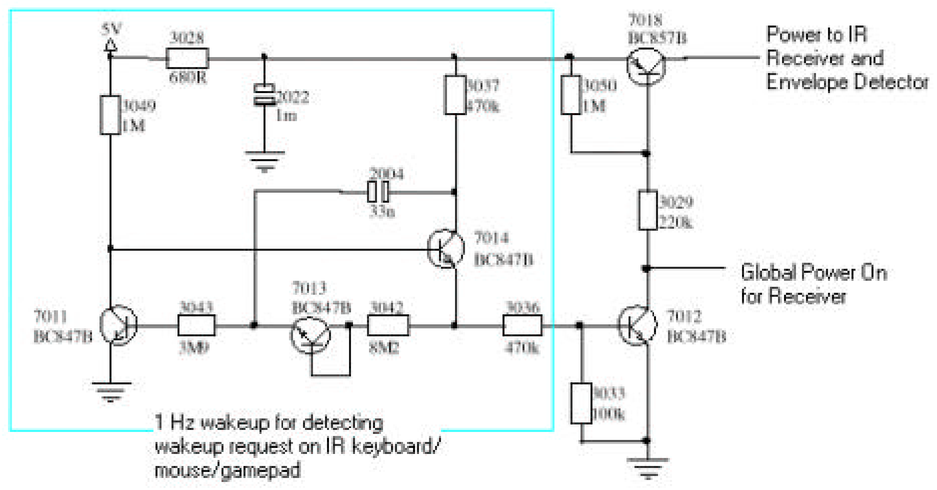
D11/H11A/H12 will go into sleep mode 1 millisecond after detecting the absence of USB traffic for 3 milliseconds. The suspend pin (Open-Drain) from the USB controllers will go high immediately after the suspend detection. Upon sensing a suspend pin, the microcontroller needs to finish whatever it is processing and go into deep sleep so that the total suspend current is less than 500 µA. When it is processing a critical event, it may pull the suspend pin low to hold off the suspend condition; this may be done for a maximum of 10 milliseconds.

The suspend condition is terminated when USB traffic resumes. Thus the system must be responsible for waking up the microcontroller. Or, the USB device can initiate a resume signaling to wake up the host.

Power on Reset

The charging ramp voltage across the 2017 capacitor will trigger a transistor and give a pulse to the input of ExOR gates , which reset the microcontroller.

IR wake up


5V
3028
680R
3049
1M
2022
1m
2004
33n
3037
470k
3050
1M
7018
BC857B
Power to IR Receiver and Envelope Detector
3029
220k
7014
BC847B
Global Power On for Receiver
7011
BC847B
3043
3M9
7013
BC847B
3042
8M2
3036
470k
7012
BC847B
3033
100k
1 Hz wakeup for detecting wakeup request on IR keyboard/ mouse/gamepad

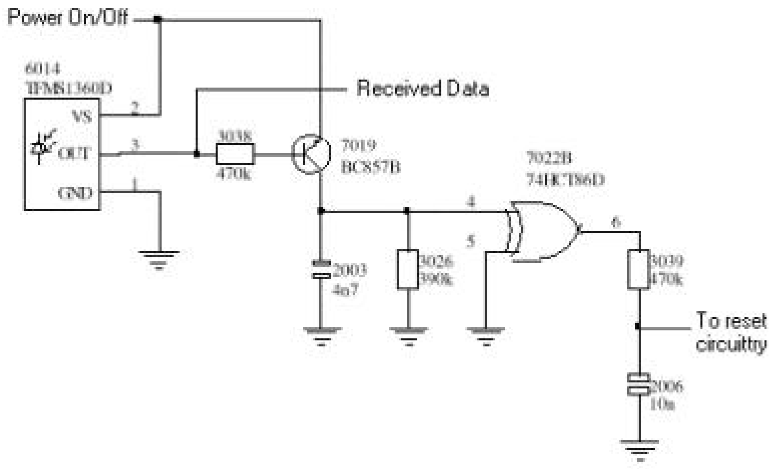
This is a -Hz oscillator with an on-time of around 20%, which gives power to the IR receiver. When the microcontroller is working this oscillator is frozen to give constant power to the receiver.

IR Receiver

TSOP1136 chip has a built-in Automatic Gain Control and a demodulator of the 36-kHz Modulation.

Implementing a USB-to-Infrared Dongle

ENVELOPE DETECTOR



Integrates the receiver pulses and makes a reset of the microcontroller. Under normal working conditions, this part of the circuitry remains dormant. The device kicks into action only after it is put into suspend. It will be enabled every 1 second to scan for any IR signals from the remote IR keyboard/mouse/gamepad.

ROM AND INTERFACING

The microcontroller is configured to use an external EPROM, the program fetch done over the multiplexed address/data bus. A data latch separates the address byte from the data bytes. The processing power is in the P80C652 microcontroller, which is running at 24 MHz. The 652 was selected because it can go into power-down mode; this is crucial to pass the power requirement during suspend.

INDICATION LEDS

The four red indications LEDs are driven by BC847B NPN transistors and connect directly to the output of the microcontroller.

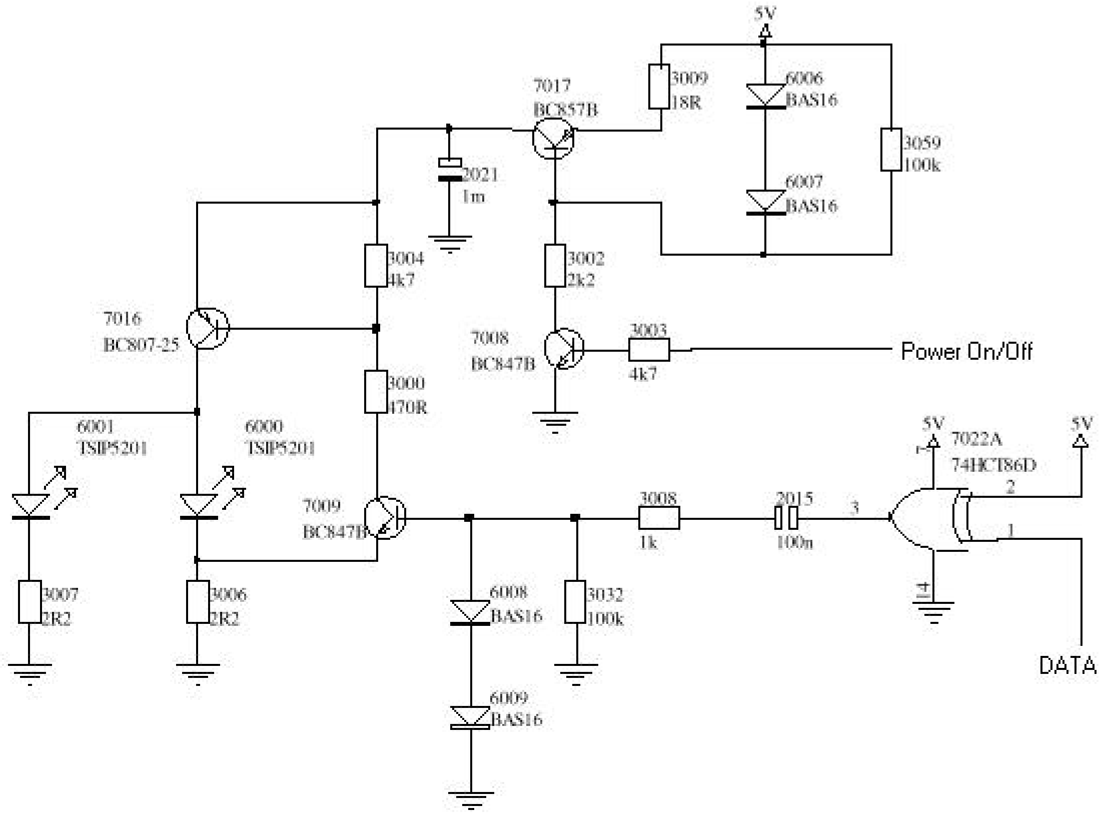
FIRMWARE

To appreciate the hardware, we must understand the firmware. It is, however, not the aim of this application note to go into the details of the firmware. A brief description below would explain the need of the circuitry as shown in the schematics.

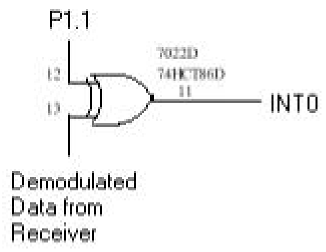
36 kHz Modulated IR Transmitter Circuitry

All IR signaling is modulated on a 36-kHz carrier. This is done by the microcontroller whose output pin modulates the data to the IR LEDs. Much of the circuitry here is used to power down the IR LEDs while in Suspend State; this is to meet the stringent requirement of 500 µA maximum suspend current.

Implementing a USB-to-Infrared Dongle



Receiver circuitry



The IR data from the TSOP1136 chip is already demodulated. Processing of this data is done via software. The microcontroller is interrupted on every transition of the received data pulse. Since the microcontroller interrupt responds to only falling/low-level trigger, an ExOR gate is used to invert the received data from the TSOP1136.

The Philips protocol for RC5 is coded in biphase format as follows:

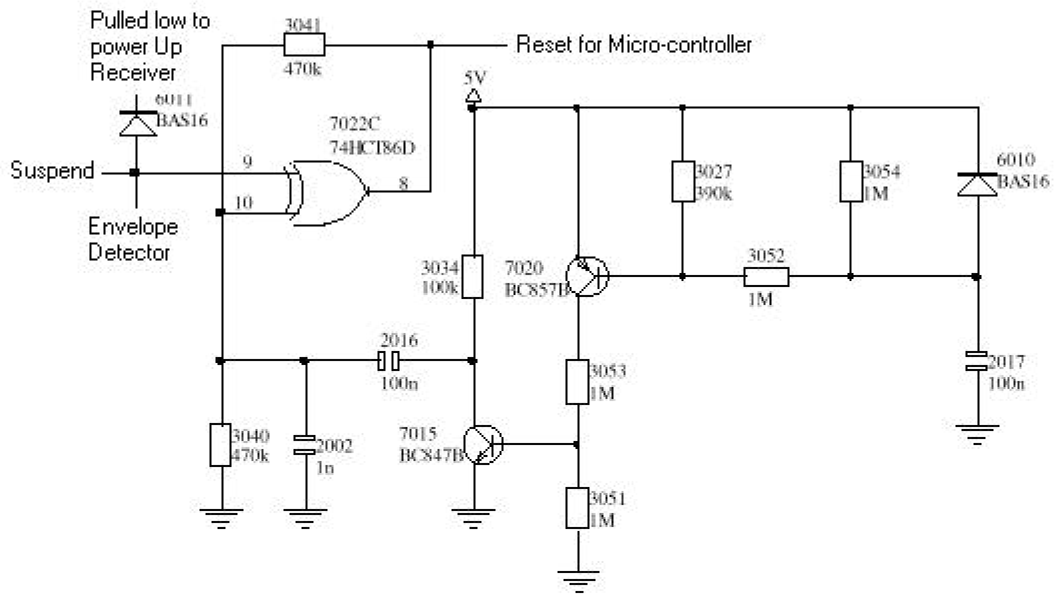
```
|---+   |      |   +---|
|   |   |      |   |   |
|   |   |      |   |   |
|   +---|      |---+   |
 <----->        <----->
  1.7778ms       1.7778ms
  Logic 0        Logic 1
```

The actual RCMM protocol document requires an NDA from the contact list.

Implementing a USB-to-Infrared Dongle

Reset and Recover from Suspend

The reset pin is tied as shown in the diagram below.



The reset to the microcontroller serves a dual purpose. The first is to do a proper power up reset. The second is to provide a wakeup signal to the microcontroller by pulling and releasing the reset line. There are two sources of waking the system from suspend. The first would be a host-initiated wakeup. In that situation, the suspend pin of the D11 would be pulled low. The second situation occurs when the wakeup is generated from the IR keyboard or IR mouse. When any key is pressed, this generates some toggling on the IR receiver, which is designed to wake up every second. Any detected IR signals during the “awake” frame would generate an enveloped pulse that triggers a Reset pin to wake the microcontroller from power-down mode.

USER FUNCTIONAL DESCRIPTION

The OVU1000 reference design from Philips can communicate through a Philips developed and owned Infrared RCMM (1-way) and Infrared RCMM² (2-way) 36-kHz modulated protocol with the following wireless peripherals:

- Philips Wireless Keyboard/Mouse (KW10XX)
- Philips Wireless Gamepads (GP3020): Up to four simultaneously operated gamepads without loss of data and with minimized latency

The OVU1000 can also be upgraded, by only programming the EPROM with other software, to support additional wireless peripherals:

- PHILIPS Remote Controls with Trackball, FSR (Force Sensitive Resistor) and other PHILIPS Remote Controls.

The Remote Control can be used as a pointing device and to control the HID controls of the WINDOWS 98™ operating system controls. (Volume, Source Selection, …)

Compatibility

The OVU1000 is compatible with the USB specification 1.1 and compliant with USB HID specification.
It complies with the Power Consumption specification described in the USB specification by using Power Control Hardware.

The device works with Windows 98™.
OVU1000 can be used with Windows 95™ if OSR2.1 and a proper HID driver are installed.

OTHER ENCLOSURES

- Attached you will find the Schematic in PDF format
  - ➢ USBIRH.pdf (PDF file for the schematic of an IR receiver with PDIUSBH11)
  - ➢ USBIRD.pdf (PDF file for the schematic of an IR receiver with PDIUSBD11)

- Attached you will find the Bill Of Materials in PDF format
  - ➢ USBIRH_BOM.pdf (Bill of Materials for IR receiver with PDIUSBH11)
  - ➢ USBIRD_BOM.pdf (Bill of Materials for IR receiver with PDIUSBD11)