V-USB

A Firmware-Only USB Driver for the AVR

Navigation

About this Wiki
Hardware Considerations
USB Device Classes
Driver API
Host Software
Features by Example
Troubleshooting
Projects Hosted on Wiki

Utilities

Add a Page
Recent Changes
Become a Member
Site Members
Manage this Site

edit menu

Hardware Considerations

Prototyping board:

Many people ask for a prototyping board for V-USB. There are a few available:

Metaboard. This board is not available for sale, but it is very easy to build it yourself. It's a single sided board which can be easily etched at home.

MHVBoard. A 20MHz ATmega328P kit based on the Metaboard, and available for purchase. Includes Arduino shield & breadboard compatibility, proto area, TTL serial & ICSP.

USnooBie. ATmega328P based development kit with V-USB circuitry. Designed in a thin shape for breadboard use. Provides bootloading features and selectable operating voltage.

AVR Stick. This board combines an ATtiny85 with the recommended V-USB hardware on a small stick shaped board which plugs directly into an USB port.

DuinoThumb. This is a ATmega168 based development board that features V-USB hardware with a USB-A male connector. The pins of the ATmega168 are connected to female pin headers.

The voltage level problem

Devices based on V-USB cannot comply to the electrical specifications of the USB standard because the AVR's I/O characteristics don't match the requirements of USB. However, we can get close enough to make it work reliably.

Each end of the communication (the host and the device) requires particular voltage levels for a High or Low signal. The other end must generate voltages in these ranges for reliable communication. Exact characteristics are defined in the USB specification.

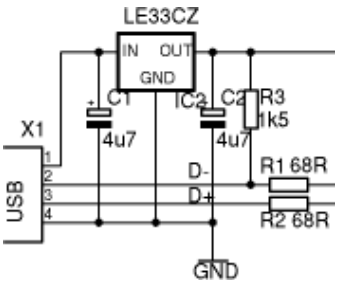
Since the host side is fixed, we must accommodate the levels sent and expected. The host sends 0 V for a Low level and 3.3 V for a High level. These levels are pretty precise, even if the specification allows tolerances. This is usually no problem, regardless of the supply voltage of the AVR. Even 5 V designs detect 3.3 V reliably as High level.

The other direction is more problematic: The host expects 0 V to 0.8 V for a Low level and 2 V to 3.6 V for a High level. Some hosts allow up to 5 V for High level, but some don't. In order to meet this requirement, we must either power the AVR at a voltage between 2 and 3.6 V or insert some kind of level conversion in the USB data lines.

Solution A: Reduced supply voltage for the AVR

One solution to the level conversion problem is to supply the AVR with a reduced

voltage, usually in the range of 3.3 to 3.6 V. Since the USB power supply is in the range of 4.5 to 5.25 V, we need a low drop voltage regulator. Advantages of this solution are:



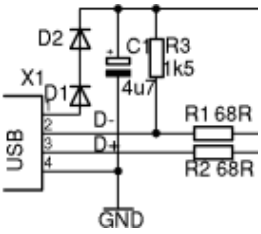
- Clean solution, fast transitions on D+ and D-.
- Good noise immunity for signal input.

Disadvantages:

- Low drop regulators are often expensive and harder to obtain.
- Many AVRs are not specified for the required clock rates at supply voltages below 4.5 V.
- Other chips on the board may not be specified for 3.3 V.

Note that using low-drop regulators LE33CZ, or even worse, LM1117S33, imposes additional supply current (quiescent current) that violates the 500 µA limit for USB standby. Because the 1.5 kΩ pull-up contributes 200 µA, only 299 µA are allowed for the regulator if the ATmega is in its deep standby drawing less than 1 µA. For compliance with the USB standard, very low-power low-drop regulators like TPS71533DCK are required. Small quantities can be obtained from Texas Instruments as samples for free.

Another variation of this approach is to reduce the supply voltage with two rectifier diodes in series. This gives roughly a drop of 1.2 to 1.4 V, usually enough to make things work. The missing regulation may be problematic for analog circuitry or if you use the ADC. It depends on the power consumption of the device and the PC side voltage supply stability.
Additional advantages:



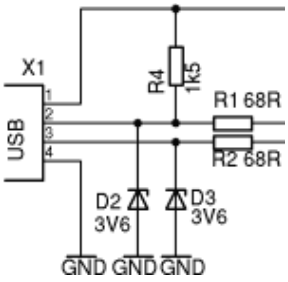
- Low cost, easily orderable devices.
- No quiescent current, makes this solution ideal for USB compliant standby mode.

Additional disadvantages:

- No regulation, only voltage step-down.
- Unregulated voltage problematic for analog circuitry, ADC usage, and maybe attached digital circuitry.

Solution B: Level conversion on D+ and D-

Instead of reducing the AVR's power supply, we can limit the output voltage on D+ and D- with Zener diodes. We recommend 3.6 V low power types, those that look like 1N4148 (usually 500 mW or less). Low power types are required because they have less capacitance and thus cause less distortion on the data lines. And 3.6 V is better than 3.3 V because 3.3 V diodes yield only ca. 2.7 V in conjunction with an 1.5 kΩ (or more exactly 10 kΩ) pull-up resistor. With 3.3 V diodes, the device may not be detected reliably.



If you use Zener diodes for level conversion, please measure the voltage levels to make sure that the diodes you have chosen match the requirements.

Advantages of the Zener diode approach:

- Low cost.
- Easy to obtain.
- Entire design can be at 5 V.
- AVR can be clocked at high rates.

Disadvantages:

- Not a clean solution, a compromise between all parameters must be found.
- Zener diodes come with a broad range of characteristics, especially at low currents, results may not be reproducible.
- High currents when sending high-level.
- High level is different for signaling and in idle state because signaling uses high currents to drive the diodes while idle state is driven by a 1.5 kΩ pull-up resistor.

Which clock rate should I choose?

V-USB can currently handle clock rates of 12 MHz, 12.8 MHz, 15 MHz, 16 MHz, 16.5 MHz, 18 MHz and 20 MHz. These clock rates are *precise*! A crystal with 11.9 MHz won't work! Only the 16.5 MHz and 12.8 MHz variants allow a deviation of up to 1%. They are therefore suitable for RC oscillators. 16.5 MHz is suitable for devices which can derive a 16 MHz clock from an RC oscillator (e.g. ATTiny25/45/85 and ATTiny26), 12.8 MHz can usually be reached by calibrating the 8 MHz RC oscillator. Calibration of the RC oscilator is described at examples.

The driver uses different code in the assembler routines for each clock rate. All assembler modules should be equally reliable. Your decision can therefore be based on the following criteria:

- Do you want to use the RC oscillator? Then you need the 16.5 MHz module.
- Is code space very tight? Then we recommend the 16 MHz or 20 MHz module, these have the least code size.
- Is the AVR powered at low voltage? Then we recommend 12 MHz for reliable operation.
- If you want to include CRC checks on USB transfers (available in vusb-releases since 2009-03-23) use 18 MHz.
- Do other parts of the code or hardware require a particular clock rate? Well, then you have no choice and are lucky if the rate is supported by V-USB.

Designs based on the RC oscillator need calibration. EasyLogger starting with version 2008-02-28 includes code for auto-calibration based on the USB frame clock. This code is now also included in directory `libs-device` of the driver package.

page revision: 24, last edited: 29 Oct 2011, 17:09 (1543 days ago)
Edit Tags History Files Print Site tools + Options

Powered by Wikidot.com Help | Terms of Service | Privacy | Report a bug | Flag as objectionable

Unless otherwise stated, the content of this page is licensed under Creative Commons Attribution-ShareAlike 3.0 License

Other interesting sites

Scion: Monsters and Mosh Pits

Хроника Убийцы Короля

Electromagnetic (Hyper)Sensitivity

Causes of Electro Smog, EMF and Wireless Devices

Connors Campaigns

My dashboard for RPGs, Music, Sports, and more.