

A miniATX Bench Power Supply

Project Abstract

June 2004

Submission #A3607

Introduction

Bench power supplies are basic components of electronics labs, and are widely available. But it is difficult to find a low-cost, reliable, multiple-output supply that provides external connectivity for remote monitoring and programming. The miniATX Bench Power Supply addresses this need by implementing a 100W five-output supply with RS232 and USB interfaces. A commodity miniATX computer case with built-in power supply is used as the backbone of the project. This combination of case and power supply is available for less than \$40.

The miniATX switching supply powers five output voltages that are monitored and controlled by the hardware and firmware in this project. The output voltages are:

- +3.3V at 3A
- +5V at 3A
- +12V at 3A
- Adjustable 0V-10V at 3A
- Adjustable -10V-0V at 1A

All five supplies have programmable current limits. These limits, as well as the adjustable voltages, are set by a front-panel LCD interface. When an overcurrent condition is detected, the miniATX Bench Power Supply turns off the supply until the user resets it from a front-panel pushbutton.

The front-panel LCD simultaneously displays measured voltages and currents, and also displays the speed of two cooling fans and two temperature sensors.

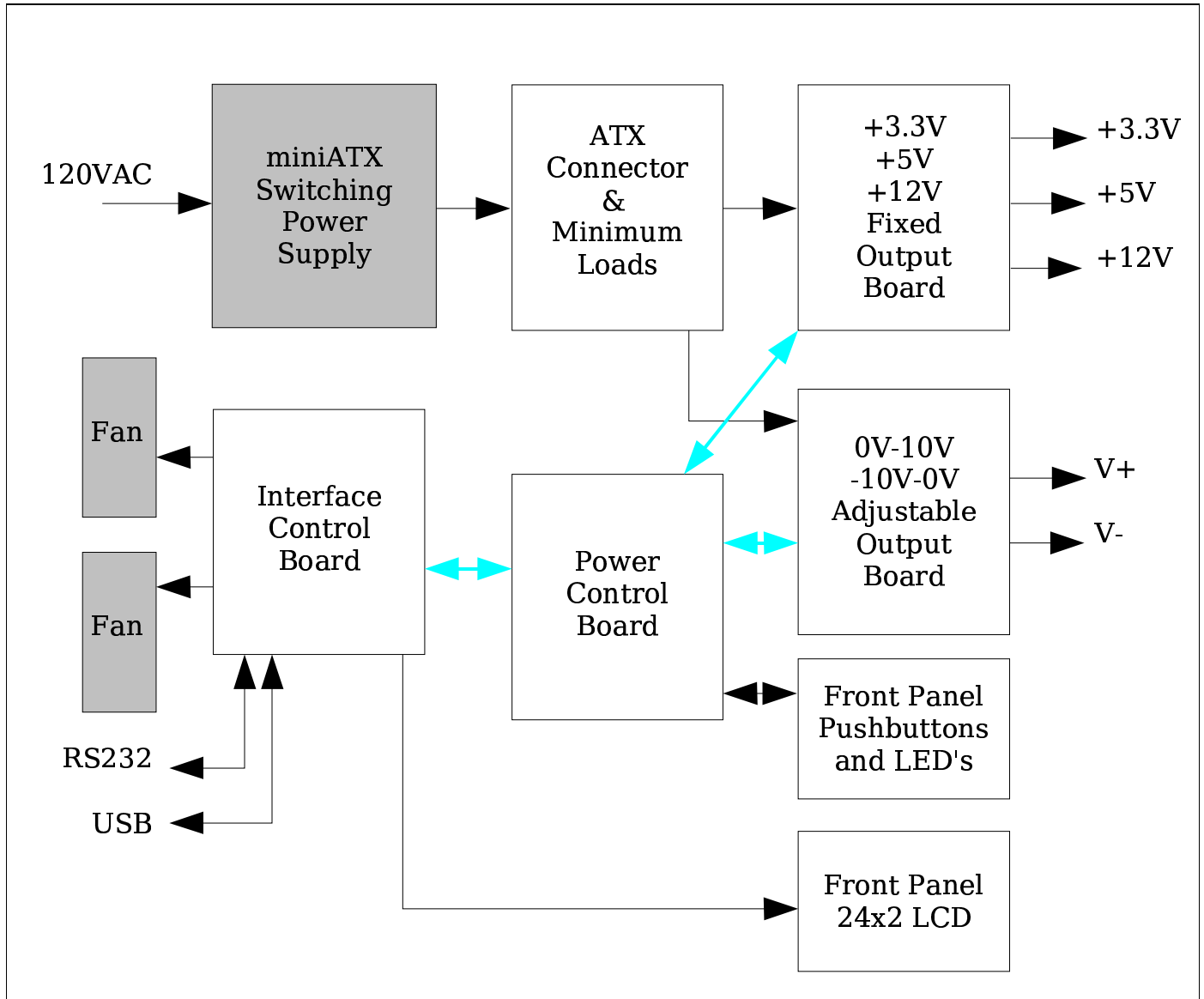
To support projects that require multiple supplies, a master power switch can enable and disable multiple outputs without having to turn the whole unit on and off.

The miniATX Bench Power Supply comes with hardware support for RS232 and USB interfaces, as well as an output for a small speaker to provide audio feedback. Together with in-system programming of firmware through the RS232 port, these interfaces give the user lots of flexibility to extend and customize the behavior of the power supply.

Four Atmel AVR microcontrollers are used in the design, with TWI interfaces between each microcontroller board for easy expansion or upgrading.

Block Diagram

A block diagram of the power supply is shown below. Gray blocks represent components that come with the computer case. Blue arrows represent TWI interfaces.



Photograph

A photograph of the front of the power supply is shown below. More photographs are available in the Pictures directory of the submission.



Software Code Sample

A sample of the code from the adjustable supply output board is shown below. This is the main loop of the firmware. It reads 5 A/D channels to measure supply voltages and currents, as well as the on-board temperature. Floating-point math is avoided by using rational approximations to real numbers. This code is available as the file `Abstract/codesample.c` in the project submission.

Schematic

The full schematic of the project comprises several pages thus is not included in this document. All schematic pages are available as PDF files in the `Abstract/Schematics` directory of the project submission.

```

// Check for completed conversion. Each conversion takes about 0.208ms and
// we cycle between 5 channels, so each channel is sampled about once every
// 1.04ms, or 961 samples per second.
if (ADCSR & _BV(ADIF)) {
    switch (ADMUX & 0x0F) {
        case 0: // -ADJ current sense. Must convert 3.98-3.717Iout to Ix100
                // Formula will be:
                //  $I_{x100} = ((ADCH/255)+4.096 - 3.98) / -3.717 + 100$ 
                //  $= ADCH * (-0.43214) + 107.08$ 
                //  $= 107 - ADCH * 105 / 243$ 
                uval = ADCH * (uint16_t)105 / (uint8_t)243;
                if (uval <= 107) {
                    Supplies[1].I = 107 - uval;
                } else {
                    Supplies[1].I = 0;
                }
                ADMUX = _BV(REFS0) | _BV(ADLAR) | 0x01;
                break;

        case 1: // -ADJ voltage sense. Must convert 0.308314xVout+0.2 to
                // Vx100. Formula will be:
                //  $V_{x100} = ((ADCH/255)+4.096-0.2) / 0.308314 + 100$ 
                //  $= ADCH * 5.20987 - 64.869$ 
                //  $= ADCH * 250 / 48 - 64.869$ 
                //  $= (ADCH*250 - 3114) / 48$ 
                uval = ADCH * (uint16_t)250;
                if (uval >= 3114) {
                    Supplies[1].V = (uval - 3114) / (uint8_t)48;
                } else {
                    Supplies[1].V = 0;
                }
                ADMUX = _BV(REFS0) | _BV(ADLAR) | 0x02;
                break;

        case 2: // +ADJ current sense. Must convert 1.176 V/A to Ix100
                // Formula will be:
                //  $I_{x100} = (ADCH/255)+4.096 / 1.176 + 100$ 
                //  $= ADCH * 1.36588$ 
                //  $= ADCH * 239 / 175$ 
                Supplies[0].I = ADCH * (uint16_t)239 / (uint8_t)175;
                ADMUX = _BV(REFS0) | _BV(ADLAR) | 0x03;
                break;

        case 3: // +ADJ voltage sense. Must convert 0.32622xVout to Vx100
                // Formula will be:
                //  $V_{x100} = (ADCH/255)+4.096 / 0.32622 + 100$ 
                //  $= ADCH * 4.9239$ 
                //  $= ADCH * 256 / 52;$ 
                Supplies[0].V = ADCH * (uint16_t)256 / (uint8_t)52;
                ADMUX = _BV(REFS0) | _BV(ADLAR) | 0x09;
                break;

        case 9: // Temperature sense. Sensor drives 10mV/deg.C. with
                // 0.5V at 0 degrees C. Formula will be:
                //  $T = ((ADCH/255)+4.096 - 0.5) / 0.01$ 
                //  $= ADCH * 1.6063 - 50$ 
                //  $= ADCH * 257 / 160 - 50$ 
                //  $= (ADCH*257 - 8000) / 160$ 
                // in degrees Celsius.
                TempA2D = (ADCH*(uint16_t)257 - 8000) / 160;
                ADMUX = _BV(REFS0) | _BV(ADLAR) | 0x00;
                break;
    }

    // Initiate the next conversion and clear ADIF flag.
    ADCSR |= _BV(ADSC);
}

```