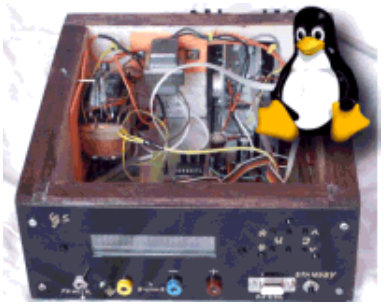
LinuxFocus article number 251
http://linuxfocus.org

A Microcontroller based DC power supply



by Guido Socher (homepage)

About the author:

Guido loves Linux not only because it is fun to discover the great possibilities of this systems but also because of the people involved in its design.

Abstract:

This article is the 4th article in the LinuxFocus AT90S4433 Microcontroller series. I suggest you to read the previous articles on Atmel Microcontrollers programming with regards to:

1. How to install and use the Linux AVR development environment and how to build the programmer hardware:
   March 2002, Programming the AVR Microcontroller with GCC
2. How to make your own printed circuit board:
   May 2002, A LCD control panel for your Linux server
3. How to build the case/box for your power supply:
   September 2002, Frequency counter 1Hz-100Mhz with LCD display and RS232 interface

One of the most important devices for your workshop at home is a good, and reliable DC power supply. In this article we will build such a power supply. It will be Microcontroller controlled. It has a LCD display, and you can send it commands from your Linux computer via RS232 interface. It has a very robust design.

This article shows also how versatile Microcontrollers are. It is however not the simplest circuit.
If you are just looking a simple DC power supply then take a look at the "simple DC power". The simple DC is good if you just need a small power supply unit for the other electronic experiments in LinuxFocus. It has however nothing to do with Linux and software in general.
Even if you finally only build the "simple DC power" unit you can read on and learn many interessting aspects about Microcontrollers.

Introduction

This Microcontroller bases DC power supply is not the simplest circuit but I can assure you that you will not regret the time needed to build it. It is very robust and reliable. It is also technically very interesting, because you will learn how to generate a analog DC voltage with a Microcontroller without using a DA-converter chip.

You need a lot of parts for this article but those are only cheap standard parts. This power supply is not expensive.

What you need

See this part list for a listing of all the parts that you need. You can also see the needed parts with their values in the schematic below.
Our power supply comes in 3 variants. Except for the transformer and one resistor there is only a modification in the software required. All other parts are identical for all 3 options:

1. 0-16V Imax=2.2A
   buy a transformer with 15V 2.5A
2. 0-24V Imax=2.2A
   buy a transformer with 24V 2.5A
3. 0-30V Imax=3A
   buy a transformer with 30V 3A

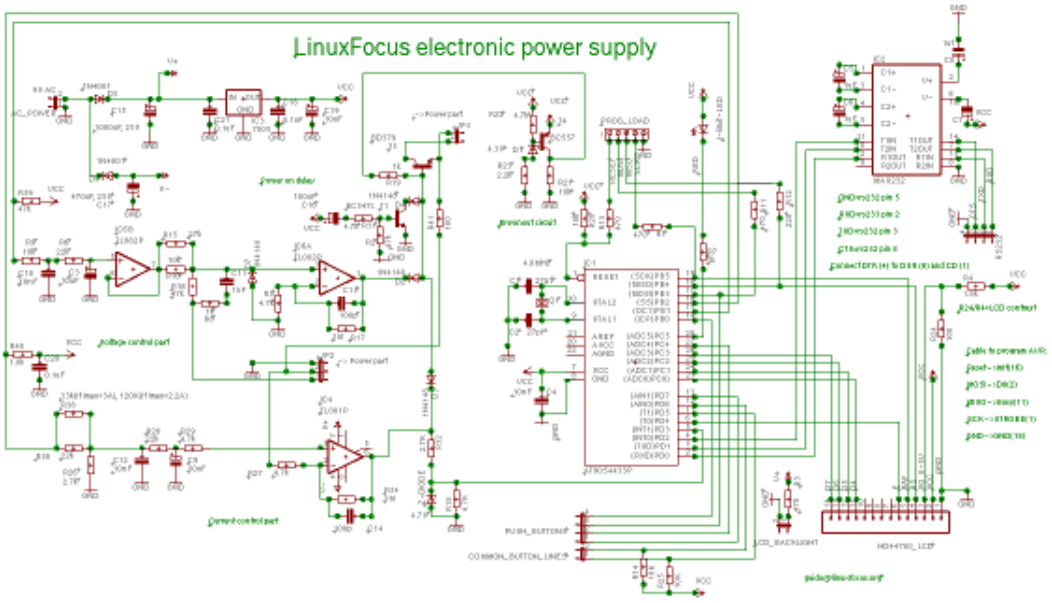
Note: In all three cases you need of course the additional 9V, 100mA transformer for the power to the main board.

Schematic and board

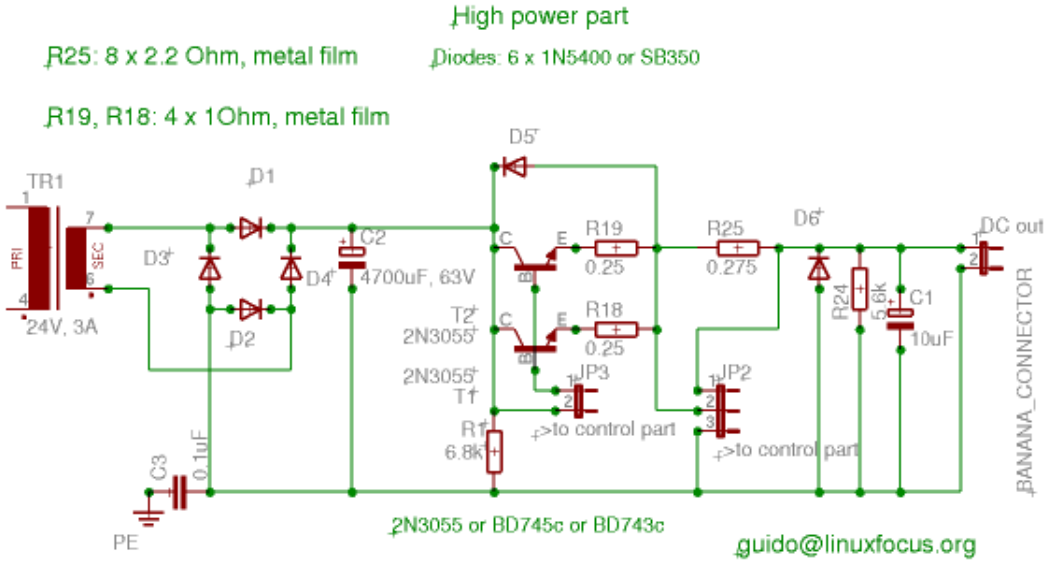
I used eagle for Linux to design the schematic and board. The eagle files are also included in the tar.gz package together with the software. You can download it at the end of the article.

The circuit is divided into 2 parts. One main part and one part that should be close to the power transistors. Below you see 2 independent schematic diagramms for the two parts but they are finally to be connected via wires.

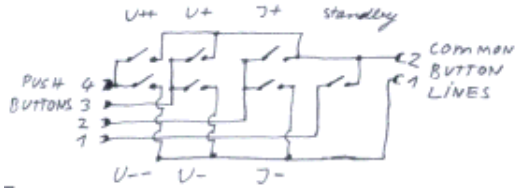
The main schematic (click on it for a bigger picture):



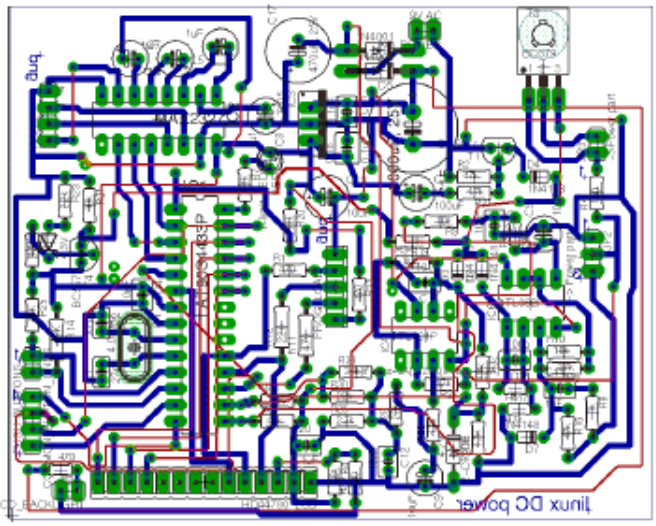
The schematic for the high power part (click on it for a bigger picture):



How to connect the push buttons in a matrix (click on it for a bigger picture):



The main board, top view (click on it for a bigger picture):



The board is specifically designed for hobby electronic. Only the blue layer is meant to be etched as a printed circuit board. The red lines are wires. It’s much easier and less accuracy is required to build a single sided printed circuit board. You can lay the wires (red) such that they have the shortest length. I could not do that in eagle.

The few parts in the high power part of the power supply can be mounted on a standard prototyping boards (those boards with many holes). The main board and the power part is connected via wires (JP2 and JP3). You will notice that the ground wire from the main part connects plus DC out. This is correct and it is the reason why we need two separate transformers for (one for the power part and one for the logic part with Microcontroller and operational amplifiers).

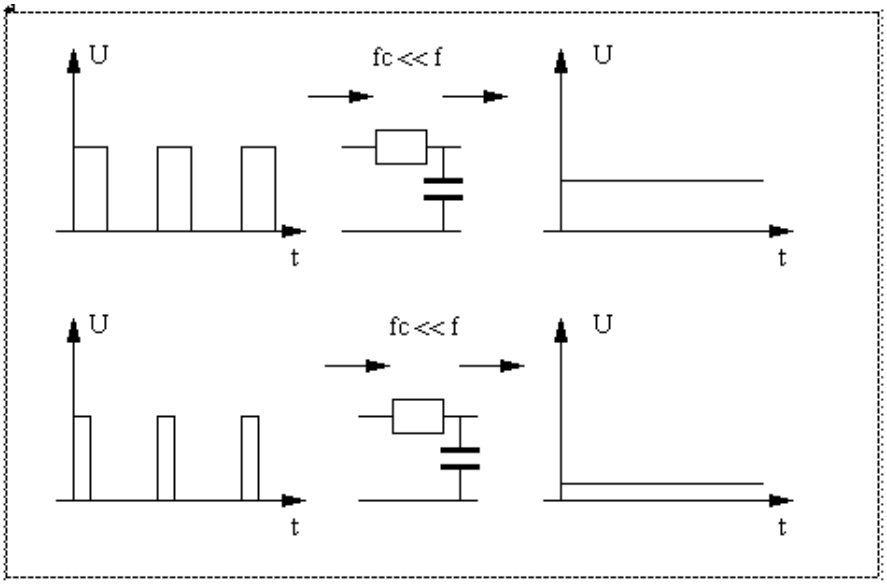
How it works

Looking at the main schematic you can see that it consists of 2 logical parts. One is is marked in the schematic as "current control" and the other "voltage control". These are 2 independent control loops. The one loop controls the output voltage and the other the voltage drop over the 0.275 Ohm resistor in the power part. The voltage drop is equivalent to the current. The two control parts are "combined" via diodes D2 and D3. These diodes form and analog electrical OR-gate. That is if the current is too high then the current control part lowers the voltage until it is below the limit otherwise (current not too high) the voltage control part is in charge of regulating the output voltage.

This logical OR works because the transistor T3 is connected via R19 to +5V. If there where no operational amplifiers connected behind D2 and D3 then you would get maximum output power. The operational amplifiers in the control loops control the output by taking away the +5V from T3 (pull it as much as needed to ground).

The voltage control loop controls the output voltage according to the voltage level that it gets on pin 5 of IC6B. In other words the voltage on pin 5 is equivalent to the output multiplied by the amplification factor which is determined by the resistors R15, R10 and R16. The same goes for the current except that it is the voltage on resistor R30 which is equivalent to the max. output current.

In order to set the max current or regulate the output of the power supply we just need to supply appropriate voltages on the two points (pin 5 of IC6B and resistor R30). This is what the Microcontroller does.... but how can a Microcontroller generate and regulate a reference DC voltage? Take a look at the following picture:



What you see in this picture is how a pulsed signal can be transformed into a DC signal. All you need to do is run it through a low pass filter with a cut off frequency a hundred (or more) times lower than the signal frequency. Since our Microcontroller runs at 4Mhz it is not so difficult to design such a low pass filter. Even if we implement the signal generation with software we will still get a few kHz and the filter will still be very small.

The difference in the picture between the upper and the lower diagram is called pulse width modulation. By changing the length of the pulses we can change the DC voltage behind the filter.

Cool, isn't it? We can generate a exact DC voltage from a digital signal!

The AT90S4433 Microcontroller has two internal counters. One is 16bit wide and one is 8bit wide. The 16bit counter has the possibility to use pulse width modulation (PWM) which is already implemented in hardware in the AT90S4433 chip with a resolution of 10bit. The 8bit counter does not have that but we can implement it in software. It is still fast enough. We use the 16bit counter for voltage regulation this gives us 10bit=1023 steps of resolution for the voltage control. The output current is controlled with the 8bit wide counter and it gives us 255 steps to control 1-3000mA. That means we have an accuracy of about 12mA (or less). This is still sufficient for current control.

All the other parts in the circuit are for power supply and reference voltage (the 7805 is our reference point) and for ensuring that the power supply does not behave unstable when switched on or off.

The software

The software for the Microcontroller uses many aspects which you already know from the previous articles (uart for rs232, lcd display, counters in interrupt mode). You can take a look at it here:

linuxdcp.c.

Interesting is perhaps the software PWM (Pulse Width Modulation). The variable ipwm_phase implements together with ipwm_h the PWM for the current. We just run the 8bit counter in interrupt mode and every time it generates an overflow the function "SIGNAL(SIG_OVERFLOW0)" is called. Here we check the ipwm_phase to check if we should generate a 1 or a 0 at the output and then we restart the timer. Easy.

The software is not complicated at all but to understand it exactly you need to read the data sheet of the 4433 (see references).

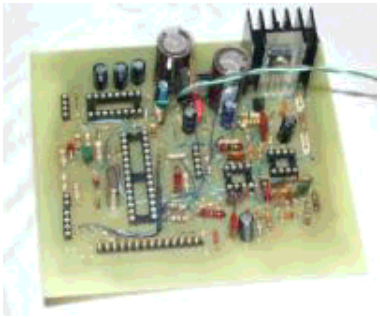
The 4433 is a 8bit Microcontroller and its mathematical capabilities are limited. The functions divXbyY and multiXbyY implement 24bit math which we need to accurately calculate the pulse width from a given voltage set be the user.

Our power supply has 7 buttons. 6 buttons are available to step the current and voltage levels and one button is "standby". Using the standby button you can temporary switch off the power and still change the voltage and current limits. The state of the buttons is "pulled" in the main loop in the program. The ignorebutton variable is used to debounce the buttons. When you press a button with your finger it bounces up and down a bit. As a human you will not notice this but the Microcontroller is so fast that it would see on, off, on, off... The ignorebutton counter waits a bit after a button press to avoid this bouncing.

Making the printed circuit board

The software package contains a postscript file (linuxDCpower.ps) for the printed circuit board. Personally I find that the pads are always a bit too small. Therefore I strongly recommend to enlarge them a bit with a paint marker before you etch the board. The process how to make a board at home is described in: May 2002, A LCD control panel for your Linux server.
How to build a cheap but good looking case for your power supply is described in the "September 2002, Frequency counter 1Hz-100Mhz with LCD display and RS232 interface" article. You can see the case and front panel that I made on the right. Click on the images for bigger pictures.

The main board:



The case for the power supply. Wood on the sides, sheet metal for bottom part, top and front:

The front panel:

Testing

Like any circuit that you have soldered together it is a good idea to not directly connect it to full power supply but rather test it step wise. This is to find faults that you have made while building the circuit.

1. Assemble the main board with all the parts but do not put the ICs into the sockets.
2. Take a 9V battery and connect plus to the pin 2 and minus to pin 1 on the connector marked in the schematic with AC_POWER. Use a voltmeter and check that you have +5V on the max232 between pin 8 and 16 and on the Microcontroller pins 7 and 8. On the operational amps you should have almost 9V on the positive power pin.
3. Now turn the 9V battery (pin 1 to plus and pin 2 to minus) and check that you have around -9V on the negative power pins of the operational amps.
4. If all the tests until here are passed then the power supply of the main board works and it is save to insert the max 232 and the Microcontroller into their sockets.
5. Use again the 9V battery and connect is such that you have the +5V supply working (see above). Connect the programmer cable to the parallel port and the connector for programming the board. Unpack the software package (for download see references chapter), "cd" into the directory that is created and type:
   make avr_led_lcd_test.hex
   make testload
   make ttydevinit

   Now the test software should be loaded to the board. On the LCD display you should see "hello", the red LED should blink and if you connect your computer to the rs232 you should see "ok" being printed (initialize the rs232 line with ttydevinit, then type cat /dev/ttyS0, or cat /dev/ttyS1 for COM2).
6. Now assemble the power part but do not connect the main transformer yet. Instead connect the 9V battery to the cables where the transformer would be connected. No matter in which direction the battery is connected the 4700uF capacitor should always charge up to around 9V. Check this with

a voltmeter.

7. When the last test step is passed do some final checking of the wires and then connect all the transformers and power on. With no operational amplifiers in the sockets you should get the max. output voltage out of the power supply. Measure this but take care to not cause any short circuit otherwise you blow up the power transistors since there is no current limitation yet.
8. Power down insert all the operational amplifiers and connect again the programmer cable, power on and type:
   make
   make load
9. Now the power supply should be fully functional. Note that while the programmer cable is still connected the output is slightly off. Disconnect it to get accurate output voltage and current.

Here it is: Our own power supply

You have seen above that there are 3 options available dependent on what transformer you use. The default software is for 16V, 2.2A output. To change this edit the file linuxdcp.c and search for: MAX_U, IMINSTEP, MAX_I, and in the function set_i you need to change the calibration if you have 3A maximum output. The code is well commented and you will see what you need change.

Finally here are a few pictures of the power supply as I have build it. It was quite some work but it really is a very good and robust power supply. The time was well invested since a lab-power supply is really one of the most used things.

Using the power supply

It is probably almost obvious how you use the power supply. You have 4 buttons to set the output voltage. 2 buttons to step up/down by 1V and 2 buttons to step up/down by 0.1V. The current limit can be set also with 2 buttons. Here the stepping is not linear. For smaller values you can increment or decrement by 50mA. For values over 200mA you can step in 100mA units and above 1A in 200mA units. That way it is easy to step through the whole range with just 2 buttons.
The standby button can be used to temporary switch off the power without the need to set the values again when you switch on.
The red LED will go on when you reach the current limit and it will blink in standby mode.
The power supply can also be totally controlled via ascii commands over the rs232 serial line. The following commands are available:

u=X set the voltage (e.g u=105 set voltage to 10.5V)
i=Xmax set the max current (e.g i=500 sets the current limit to 500mA)
s=1 or s=0 set to standby
u=? or i=? or s=? print the current settings. This will produce a printout that looks e.g like this:
u: 50 s:0 i: 100 l:0
u: means voltage=50 =5V, s:0 means standby off, i: 100 is 100mA, and l:0 means current limit is not reached.

Using this acsii command language you could also write a graphical user interface for the power supply. To use the rs232 line you need to initialize it first with the command ttydevinit. ttydevinit is included in the software package. This is also described in the September 2002, Frequency Counter article.

As you have seen in the schematic diagram above we use 2 transformers and the ground plane of the control logic is connected to the positive DC output. The two transformers separate the voltages and there is normally no problem with this setup. We need to connect things like that to have the right polarity for the feedback loops of the operational amplifiers. **A word of waring**: This setup means also that the ground line of RS232 line is connected to the positive DC output! In other words you can not use the RS232 line if you want to use the power supply with other parts that are connected somehow to the ground line of your computer. It might be an idea to put a label on the case of the power supply saying "ground line of RS232 connection is connected to positive DC output line". If you want to make sure that there is no way to cause a short circuit through the ground wire of the RS232 line then either use a battery powered laptop or make sure that the ciruit powered by the power supply does not have any other connections or do not use the RS232 command interface. Also don’t be too shocked by this warning. If do not go above 250mA with the current limitation of the power supply then the red led will tell you when you made a mistake and there is no danger for your computer even if you did something stupid.

Security

This circuit contains a transformer which is connected to then main power supply (230V or 110V dependent on your country). Please ensure proper insulation. If you have never worked with power supplies then ask an experienced person to check your circuit with regards to insulation and security

before you connect the first time.

Tuning

The software for the power supply is already calibrated. Most likely you will not have to change anything there. Hardware wise the calibration depends only on the 7805, R15, R10, R16 and R38, R30, R26. Only those parts influence the voltage and current levels. If you want to do fine tuning you can either change those resistor or you can modify the software. Note that a connected programmer cable influences the output. Before you make measurements you should disconnect the cable. In software you can do the changes in the functions set_u and set_i. It's commented in the code of linuxdcp.c

References

- The uisp AVR programmer software: www.amelek.gda.pl/avr/
  local copy: uisp-20011025.tar.gz
- How to build the programmer hardware and install the AVR compiler:
  March 2002, Programming the AVR Microcontroller with GCC
- The source code for this article linuxdcpower-0.1.tar.gz, 1201K . The circuit diagram, the Eagle files and screen shoots are as well included.
- All software (updates will be listed here) and documents :software/datasheets
- Datasheet for bd379 bd379.pdf 44K
- Datasheet for TL082 TL082.pdf 110K
- Datasheet for TL071 TL071.pdf 268K
- Datasheet for 2n3055 2n3055.pdf 64K
- Datasheet for MAX232 MAX220-MAX249.pdf 448K
- Datasheet for ST232, a cheap variant, often sold instead of the real MAX232 st232.pdf 100K
- Datasheet for Atmel AT90S4433 avr4433.pdf 2356K
- The atmel website: www.atmel.com/
- Eagle for Linux cadsoftusa.com

2005-01-14, generated by lfparser_pdf version 2.51