Universal Remote Controlled Lamp Dimmer
Converting the Controller to 220VAC/50Hz Power

By: William J. Boucher, Document created: Apr.1 '01, Revised: July 8 '03

The most common question that I receive with regards to the lamp dimmer project is:

> How do I modify the controller and the software to operate on 220VAC/50Hz power supply?

Well, here's the answer.

Physical Changes:

- Change the capacitor C3 to 0.5uF/400V.
- Change R2 to about 680ohm with a 2W minimum power rating.
- Your Triac must withstand at least 400V as well but a 600V would be safer. Some of the triacs I listed as alternates later proved to not work because they required too much gate current. Typically, a triac with a gate rated for 5mA or less will work.

Software Changes:

To operate properly with 50Hz power, only a few constants need to be changed. To change program constants, you must locate the lines in the program source code (i.e. file: ircld501.src) that defines the value for each constant. Then, edit the values, save the file, and finally compile the program using the CVASM compiler. This produces a new object file with which you program the PICmicro. The CVASM compiler replaces the old Parallax compiler and is available for free download from http://www.tech-tools.com . I have provided files (IRCLD502.SRC and .OBJ) that have the following constants changed for 50Hz line operation.

The only constant values that have to be adjusted to work on 50Hz
are:

BitWidth: For the program to have correct timing to see a bit width = 1000us:

BitWidth = 1000us / loop time = 1000us / 50us,
BitWidth = 20

SpaceWidth : For the program to have correct timing to see a space width = 4166us:

SpaceWidth = 4166us / loop time
SpaceWidth = 4166us / 50us,
SpaceWidth = 83

TimerValue : To set the program's main loop frequency to achieve 200 (light levels), it has to run 200 loops for each half-cycle. With 50Hz line power there's (2 x 50Hz) or 100 halfcycles per second:

Since a half-cycle = 1 / (2 x line freq) = 1 / (2 x 50) = 10ms, the main loop freq will have to be 200 / 0.010s = 20,000 Hz (period = 50us).

TimerValue = 255 - ((xtal / 4) / (loopfreq x prescaler))
TimerValue = 255 - ((10000000 / 4) / (20000 x 2))
TimerValue = 255 - (2500000 / 40000)
TimerValue = 255 - 62.5
TimerValue = 192.5

You cannot use anything but an integer so try 193. Better to be hair too fast than too slow. I have found from experience that it is better to push the value even higher. I have used 207 with much success. The reason for this is to provide some tolerance to variance in the line frequency, the crystal frequency, the loop frequency, etc. If for any reason the loop frequency relative to the line frequency allows the phase angle counter to run beyond the zero crossing of the next half-cycle, the triac can be triggered early in the next half-cycle, making the lamp turn on close to 100%. Sometimes, if this effect is intermittent, the lamp will flicker severely. By running the main loop slightly faster, this problem is avoided.

At times, people have reported a lot of problems with their remotes (in other regions, i.e. the U.K.) activating the same function regardless of the button they press. All of my universal remotes worked when I programmed them to emulate RCA or GE devices, but if there are a lot of "RC5" signal type devices in your region, the lamp controller may need a bit more help. Just try it as it is and see if it works before getting into a panic. Decoding and saving more than two bytes of the signal can solve the problem. My program source code allows you to expand to up to 8 bytes per command and to save them in the eeprom. There are several sections in the program where lines containing SERDATAn and MEMDATAn values are commented out. You can enable more bytes just by un-commenting them appropriately.

Regarding code changes, it is also possible to make the system work with reduced resolution (say 100 levels of light) using a 4MHz PIC if you cannot source a 10MHz part. To do that, a lot of numbers have to change. Any time you change the crystal frequency, there are a lot of timing issues to address. This is handled by changing a series of constants. I haven't documented the procedures on how to calculate all of those constants so for now you should stick to a 10MHz crystal or figure it all out for yourself. Now that 20MHz flash PICs are commonly available, you could reprogram the project to double the resolution or to handle more than one lamp driver or to handle specific RC code protocols. In fact, the same basic hardware could be modified slightly to accept control signals from RF modules, serial comports, SPI masters, or whatever you like. You could add a dedicated IR signal decoder IC to work alongside the PIC if it designed for a specific IR protocol. If it is a universal type, you'd also have to tell it what IR protocol to decode. I expect that with some IR decoder ICs, the PIC would communicate this information via the SPI interface.

Good luck with your project.