

Battery Backup & Charger

KX4Z

Draft Version 1.0

May 11 2020

Introduction.

Amateur radio operators involved in emergency communication and others often need to have backup battery power for their station. It is advantageous if that backup power comes on-line immediately if there is a loss of AC power. There are excellent commercial solutions that accomplish this goal using P-channel power MOSFETs and associated circuitry. Some also provide for background charging of the battery.

There were multiple purposes to this project:

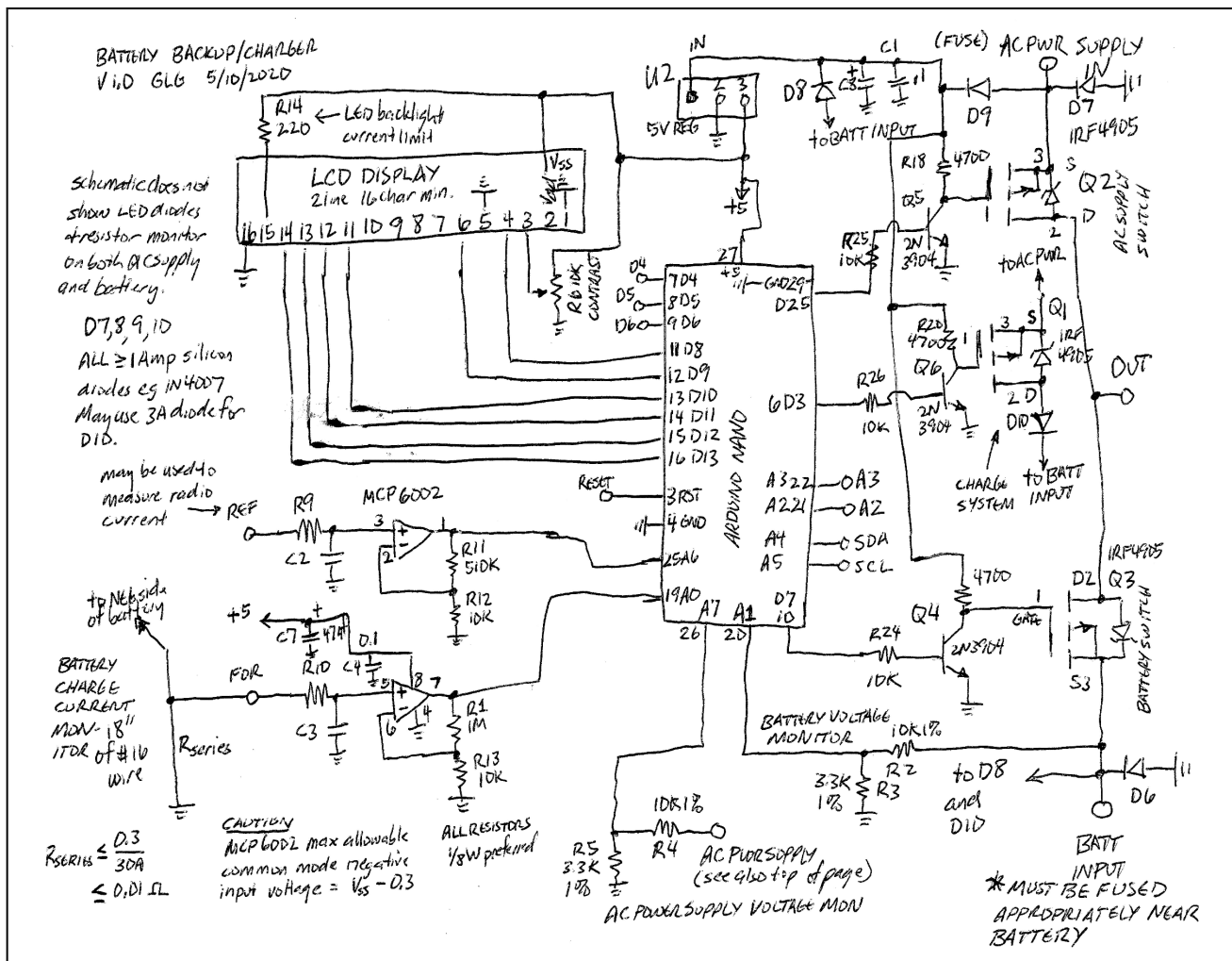
- Create an immediate backup battery switching system
- Provide for charging of the backup battery
- Possibly provide for multiple types of backup batteries
- Provide learning opportunities for amateur radio operators in construction
- Provide learning opportunities for amateur radio operators in Arduino programming

Desired specifications

Current handling capacity	Up to 25 Amps @ 13.8VDC
Switch-over time	<25 m sec
Battery charging	Initially for lead acid batteries with bulk and trickle charging; monitor battery voltage, and battery current
Display	Inexpensive 2-line 16 character display
Radio current monitoring	Optionally – measure radio current
Cost	minimize

DRAFT





HARDWARE NOTES

1. The IRF4905 MOSFETs have an on resistance of 0.02 ohms and will likely drop a lot less current than even a power Shottky diode. Nevertheless, for high currents (25A) and CONTINUOUS operation (e.g., FM) they may require a heatsink. The two power mosfets that would handle this level of current are at one end of the board allowing the addition of some heatsinking. This is not tested yet.
2. Short segments of wire in the negative leads of the battery and/or radio will be used to allow measurement of the dropped voltage and therefore series current, in battery and/or radio. It is important not to exceed negative 0.3VDC to the inputs of the MCP6002 op amp, therefore some loss in sensitivity for the battery charger shunt was accepted.
3. There is an unavoidable offset voltage (dependent on individual unit) in the MCP6002 of as much as 4.5 mV in either direction, resulting in an output offset of as much as 450 mV. There

is a temperature variation in this as well. If the offset is such that the output remains at 0V despite significant current flow that should have been measured, this will be a problem. Solutions include changing the connection of the (grounded) end of R13 and choosing components for an offset in the desired direction. POSITIVE offsets (those resulting in a positive op amp output voltage at rest) are easily handled in software.

4. For safety, a fuse should be added in the battery circuit of appropriate value for the WIRING UTILIZED TO THE BATTERY – and within 6 inches of the battery. 12 or 14 gauge wiring may be required if continuous currents of 25A or greater are planned.
5. The circuit board is somewhat “tight” and 1/8 watt resistors are encouraged, particularly on the input side of the Arduino.
6. The 16x2 LCD is designed to be mounted on the underside of the board, and the entire board mounted on standoffs underneath the panel of the enclosure.
7. Filter capacitors on the board can be anywhere from 25 to 100 uF at 25WVDC or greater.

MAJOR FUNCTIONS OF BOARD	
Measure output voltage of AC-based power supply	Accuracy of approximately 25 mV measurement
Measure output voltage of Battery	
Measure charging current to battery	Gross measurement of charging current, approximate measurement resolution 50 mA; based on tapping into a WIRE on the negative side of the battery so that we don't waste battery voltage.
Measure current usage by radio	Gross measurement of radio current usage; approximate measurement resolution 100 mA; based on tapping into the NEGATIVE WIRE feeding the radio so that we don't waste voltage.
Electronically connect/disconnect the AC-based power supply to the output terminal	We can choose when the AC based power supply is feeding the load. (This could also control a relay driving an AC unit as an alternate use of this system, for example)
Electronically connect/disconnect the DC battery to the output terminal	We can choose when the battery is feeding the load.
Electronically control charging of the DC battery by the AC -based power supply	I had to put a diode in this line so the maximum charging voltage will be about 13.1 VDC but that should be sufficient for our needs.
Polarity protection	Because of the way the MOSFETs are wired for this system we can't use them for the polarity protection (otherwise because of their inherent diode, we wouldn't be able to use them to turn the supplies on and off...) but do provide a reverse-connected DIODE across the line to blow a series fuse if a power supply is reverse connected
I2C Connection	This offers immense additional functions. For example, an I2C based display could be controlled. A frequency synthesizer chip Si5351 can provide three digitally controlled synthesized frequencies from this board. I2C based temperature/humidity/pressure sensors can act as altimeters, thermostats,, and even explosive gas detectors.

SOFTWARE

At the time of this writing, the software is very much in development stage

Libraries that must be included – utilize Arduino reference information to find and add these to your integrated development environment

Arduino Pin Assignments

Outputs (to the MOSFETs)

Inputs (both voltage and current measurements)

Control Constants

MILLISECONDINTERVAL – sets the reponse time of the system to power loss

SECONDSINTERVAL – sets the interval on which charging will be monitored

Structures and Variables

Pin numbers for the LCD display – system can be utilized on the GLG board and on others

Global variables (note line wrapping in this word processing document)

```
float zero_current_voltage; // the voltage we measure at BATTCURRENT pin
                             // when no current is flowing.
                             // Note: we may have a problem if it is
negative....
float battery_voltage;      // The voltage on the battery
float AC_supply_voltage;    // The voltage from the AC Supply
unsigned long current_milliseconds; // the current number of milliseconds since
starting
unsigned long next_millisecond_time; // time to make next measurements
unsigned long next_seconds_time;     // time to make next charging check
unsigned long next_resting_voltage_time; // next time to make a voltage
measurement of the battery
int desired_battery_charge_ma= 0;
int battery_charge_pwm = 0 ;        // 0 to 255
```

Function Declarations

```
void read_battery_voltage(); // function that reads the global variable
battery_voltage
void read_AC_supply_voltage(); // function that reads the global
AC_supply_voltage
void configure_charger(); // function to control charging
void MyMicroSecondsDelay(int microseconds); // microsecond non blocking delay
void MyDelay(int msec); // non-blocking milliseconds delay
```

Setup

Set the Digital pins that will control the MOSFETs to “output”

Turn OFF all the outputs so no current flows until the software is ready to go.

Initialize the Serial connection

Initialize the LCD display

Initialize the global variables

sets up a “watchdog timer” (an available function from a library) that if not called within every two seconds....will restart the code.

LOOP

(Note that in the Arduino environment, the LOOP function is repetitively executed; you don’t need to write it into a loop as it will loop on its own.)

Software actions are carried out on time intervals.

- A call to `wdt_reset()` is made so that this timer gets reset. If the code hangs, this call will fail to be made, causing the watchdog timer to reset the processor.
- If the time has advanced to the next `millisecond_time`, the AC-based supply voltage, and the battery voltage are both measured; if the AC-based supply voltage has declined unacceptably, the MOSFETS are switched to connect the battery to the output terminal, and a message is sent to the LCD and the serial monitor. `using_battery` is set to 1 if powering from the battery; 0 if using the AC powered supply.
- If the time has advanced to the (slower) time interval at which the charging system should be checked, the battery voltage (which may be in charging mode) is measured, and displayed on the top line of the display.
- Then, if we are using the AC-based power supply, it is appropriate to consider charging and thus the following is carried out:

Based on the measured battery voltage, the `desired_battery_charge_ma` is chosen.

The current charging current is then measured by making a number of measurements of the voltage from the current shunt in the negative lead of the battery.

Changes are then made to the pulse-width-command to be sent to the PWM output pin driving the charging MOSFET; this number can vary between 0 (0% duty cycle) and 255 (100% duty cycle).

A series of rules are used to slowly advance the actual charging toward the desired number; note that the effects of these choices will be evaluated only on the SECONDS intervals

```
if(instant_current_reading>desired_battery_charge_ma + 100)
battery_charge_pwm = battery_charge_pwm/2; // cut it way down
```

If we are more than 100 mA above the desired charging current, the duty cycle is cut in half.


```
if(instant_current_reading>desired_battery_charge_ma)battery_charge_pwm
=battery_charge_pwm-3;    // decrease by 1% of our range
```

If we are above the desired charge level, but by less than 100 mA, the charging duty cycle is reduced by 1%

```
if(instant_current_reading<desired_battery_charge_ma)battery_charge_pwm
=battery_charge_pwm+3;    // incese by 1% of our range
```

If the charging current is below the desired charging current, the duty cycle is increased by 1%. It may take several cycles of this hunting algorithm, but within a few minutes the charging should be very close to the desired charging.

A couple of sanity checks finish out the control

```
if(battery_charge_pwm <0) battery_charge_pwm = 0;
if(battery_charge_pwm >255) battery_charge_pwm = 255;    // can't go
beyond these limits!
```

This desired PWM modulation is then sent to the CHARGER pin control using

```
analogWrite(CHARGER ,battery_charge_pwm);
```

The next moment in time at which the battery needs to be measured and charging possibly adjusted is then set.

The loop is now done and repeats. Recognize that the arduino moves through this loop at FULL SPEED, hundreds of thousands of times per second, but only stops to carry out our orders at the appointed discrete time intervals, which are adjustable. Why? Because we wish to minimize radio frequency interference created by this switching system, and by using slower decision making intervals, we can reduce the repetitive base frequency of the interference, thus reducing the strength of harmonics that make it to our radio.