Tesla Founders Blog

August 18, 2009

The Roadster Foundry Mobile Charger

Filed under: General — mfeberhard @ 5:16 pm

I designed the Roadster Foundry Mobile Charger because I wanted to be able to charge my Tesla Roadster from whatever electrical outlets I might find during my travels. I designed it to take a variety of Plug Adapters, and to program my car's charging current based on the Plug Adapter I chose.

I couldn't purchase the EV connector that Tesla uses on its charger, so I disassembled and re-used the connector that came with my Tesla MC120. Disassembly and reassembly of this connector is a bit difficult; perhaps I will write about how it's done sometime if you are interested. All other parts come from online suppliers like Digikey.

I needed a variety of special tools to design and build the charger. For example, I needed an oscilloscope to decode the signals generated by my Tesla high-power charger. (I sold my old 'scope to NuvoMedia, the company Marc and I founded before we founded Tesla Motors.) I found an excellent, calibrated Tektronix 2465B on eBay for $800. I needed a big drill bit and matching tap to modify the strain relief mount on the Tesla connector so that I could use thicker wire. I needed a big crimping tool to crimp the Radsok contacts for the Tesla connector onto the wires. I needed a specialty "security" screwdriver to unscrew a screw on the Tesla connector. Each of these tools has a story, fodder for future blogs…

I had the circuit boards printed by Express PCB. The front panels were machined by Front Panel Express. I fabricated the boxes on my drill press, starting with a standard Carlon waterproof 4"X4"X4" NEMA electrical enclosures.

My charger works well, and a few friends with Teslas asked if I could build one for them. I did. A few more people heard about my charger and asked if I could build one for them, and I did. Pretty soon I had built 14 of them, with orders for another 20 or so. Rather than go into business making chargers, I have recently given the project to James Morrison's Electric Vehicle Components, and he is now building and selling these chargers. Carolyn is very happy about this.

In this blog, I describe how the Roadster Foundry Mobile Charger works. I include schematics and firmware listings for your amusement.I apologize in advance for some of the crummy drawings – they were imported from Microsoft Word, which mangles everything. Also, I see that the firmware at the end is getting cropped off at the right; I will fix this I think by changing the formatting of this blog – in the next few days, I hope.

This is actually the second design; the first was based on the Basic Stamp, a simple microprocessor development system that is programmed in Basic. When I realized that I would be making more than one of these, I was kind of embarrassed by the Basic Stamp version (plus the Basic Stamp itself was too expensive at $50 each), so I chucked the design and redesigned around a $0.85 PIC microprocessor that I programmed in manly assembly language. (I re-used the Basic Stamp to make a Halloween costume for Carolyn.)

What the Charger Does

The Roadster Foundry Mobile Charger allows you to charge a Tesla Roadster from standard electrical outlets, including the following:

Type	Voltage	Rating	Plug Configuration	Example Receptacle	Usage

NEMA 5-20	120V	20A			Standard USA 120V plug. Note that some circuits are 15-amps, while most are 20-amps.
NEMA 10-30	208V-240V	30A			This is the most common plug used for dryers. Since dryers are often located in the garage, this one is very handy.
NEMA 14-30	208V-240V	30A			This connector is common on newer dryer installations.
NEMA 10-50	208V-240V	50A			This is commonly used for ovens and ranges, and is less likely to be found anyplace useful for charging.
NEMA 14-50	208V-240V	50A			This is the most common connector in RV parks, and is sometimes referred to as the RV connector.

To charge, you need to assemble a complete Charger from two components: the 20-foot long Charge Cable (which plugs into the car's charge inlet) and the short Plug Adapter that fits the electrical receptacle that you intend to use for charging – one of the types shown above.

Charge Cable & Some Plug Adapters

When you attach a Plug Adapter to the Charge Cable, the complete charger is automatically programmed for the charging rate of the electrical outlet type that you plan to use.

Plug Adapter mated to Charge Cable

The charger, in turn, detects the presence of a car at the other end, and if it finds one there, signals the car to limit its charging rate to 80% of the rating for the type of electrical receptacle used. (The National Electrical Code allows an EV to use no more than 80% of a circuit's rated ampacity.)

The 240-volt Plug Adapters also contain thermal switches that tell the car to stop charging if the Plug Adapter gets hot. This is because some electrical receptacles, particularly those in outdoor locations, get worn out and don't make good electrical connections anymore. A poor electrical connection can cause the receptacle and the Plug Adapter to overheat, potentially melting the plastic or even causing a fire.

How it works

The actual charger for the Tesla Roadster – the circuitry that converts whatever AC is supplied by the electrical outlet to whatever DC voltage that the batteries require – is actually part of the Roadster itself. The Roadster Foundry Mobile Charger simply connects the electrical outlet to

the Roadster, and signals the Roadster the maximum allowed current for the particular type of electrical outlet being used.

Note that the Roadster Foundry Mobile Charger and the Roadster itself both don't care about voltage – 120 volts or 240 volts are both fine, as are voltages in between, such as 208 volts.

The Tesla charging connector contains 4 contacts. Two are for the charging current, one is safety (green wire) ground, and the last is called the Control Pilot, which is used to signal maximum current. This signal works a lot like the Control Pilot in the old "Avcon" J1772 charging standard – so much so that you could make a Plug Adapter that goes from an Avcon charger to a Tesla Roadster with no conversion circuitry at all.

Plug Adapters

The Plug Adapters plug into the Roadster Foundry Mobile Charger with a 4-contact twist-lock connector that is rated for 50 amperes. (This particular connector is called a "California Standard" connector, and is the lowest-cost 50-amp connector type with adequate quality that I could find. I use the "marine" ones made by Marinco rather than the Hubbell version because the Marinco connectors are sealed better, and is made of plastic rather than plated steel.)

Two of the contacts are line current for charging, one is safety (green wire) ground, and the last is used as a "Diode Sense" line to set the maximum charging rate. For 240-volt charging, the line current contacts connect to the red and black circuit wires. For 120-volt charging, they connect to the black and white circuit wires. (See the schematics later on in this blog.)

Each Plug Adapter signals its ampacity with a single diode between the Diode Sense line and the ground line. The presence and direction of this diode indicate the ampacity according to the following table.

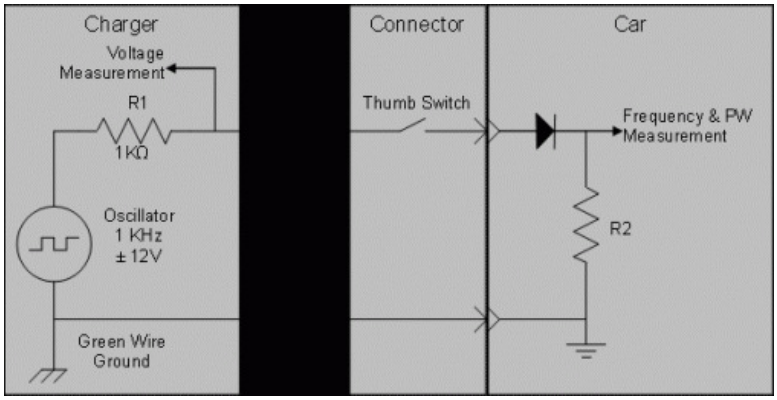
Each Plug Adapter, except the 120V Plug Adapter, also contains a thermal switch that shorts this signal out when hot, to indicate a fault. The thermal switch is glued with epoxy to the inside of the plastic plug housing.

Note that the one could build a 240-volt, 20-amp Plug Adapter that also would not have a diode. The only difference between such a Plug Adapter and the 120V Plug Adapter would be the type of connector on its end.

Diode Orientation	Circuit Ampacity	Maximum Charging Rate
None	20A	16A
Cathode to ground	30A	24A
Anode to ground	50A	40A
Short	Fault	0

The Control Pilot signal

The signaling circuit between the Charger and the car conceptually looks like this: (This is the J1772 equivalent circuit.)



The value of R2 depends on the state of the car; it has one value when the car is not yet ready to charge and another value when the car is ready to charge or is actually charging.

The sequence of operation is as follows:

1. The Charger sets its no-load output voltage to a steady +12 volts, and then measures the voltage on the Control Pilot signal. It waits in this state until it sees the voltage on the Control Pilot signal drop enough to indicate the presence of a car.
2. The Charger examines the Plug Adapter (checking for presence and orientation of the diode) to determine the maximum charging rate.
3. When it detects a car, the Charger generates a continuous 1 kHz square wave with a pulse width that indicates the maximum allowed charging rate as defined in the following table.
4. The car measures the frequency and pulse width of the Control Pilot signal to determine if a legitimate charger is attached, and to determine its maximum charging rate.
5. The car closes its contactors and begins charging at a rate no higher than that allowed by the Charger.
6. During charging, the Charger monitors the temperature sensor in its Plug Adapter, and stops generating a square wave if the Plug Adapter is too hot. (The Plug Adapter and the receptacle will overheat if the receptacle is dirty, damaged, or worn out – not entirely uncommon for receptacles in public locations.) In this state, the Charger blinks all its lights quickly. It will continue to do so until the Plug Adapter cools off and the Charger is reset (by unplugging the Charger from the receptacle, and then plugging it in again).
7. The Charger also monitors the voltage of the Control Pilot signal when it is high. If the voltage indicates that the car is no longer present, it stops generating a square wave. In this state, it slowly blinks the blue light, with the others off.
8. During charging, the car monitors the square wave. If it goes out of spec, the car may indicate a charging fault, either on its touch screen or by turning the charge inlet light red, or both.
9. During charging, the car will stop charging and indicate a fault (on its touch screen and by turning the charge inlet red) if it detects a ground fault.
10. Note that the car may choose to charge at a rate that is below the rate signaled by the Charger.

Control Pilot Specifications

Item	Nominal	Min	Max	Note
R1	1,000 Ω	980 Ω	1020 Ω	Control Pilot source resistance
$R2_{CONNECT}$	2,740 Ω	2,657 Ω	2,883 Ω	Car load resistance, car present but not ready
$R2_{READY}$	1,000 Ω	970 Ω	1,030 Ω	Car load resistance, car ready to charge

$V_{OC}H$	+12.0 V	+11.25 V	+12.75 V	Control Pilot high voltage, open circuit
$V_{OC}L$	-12.0 V	-12.75 V	-11.25 V	Control Pilot low voltage, open circuit
$V_{CONNECT}H$	9.0 V	8.25 V	9.75 V	Control pilot high voltage, car present
$V_{READY}H$	6.0 V	5.25 V	6.75 V	Control pilot high voltage, car ready
F_{OSC}	1,000 Hz	995 Hz	1,005 Hz	Oscillator frequency
PW_{12A}	200 µsec	195 µsec	260 µsec	Control Pilot pulse width, 12-amp charging[1,2]
PW_{16A}	267 µsec	260 µsec	395 µsec	Control Pilot pulse width, 16-amp charging[1]
PW_{24A}	400 µsec	395 µsec	525 µsec	Control Pilot pulse width, 24-amp charging[1]
PW_{32A}	533 µsec	525 µsec	660 µsec	Control Pilot pulse width, 32-amp charging[1,2]
PW_{40A}	668 µsec	660 µsec	795 µsec	Control Pilot pulse width, 40-amp charging[1]

Notes: 1. Pulse Width is the high-time of the Control Pulse signal, measured as the signal crosses through 0 volts.

2. Not supported by the Mobile Charger.

The Control Pilot Generator

The printed circuit board assembly (PCBA) inside the Roadster Foundry Mobile Charger (called the Control Pilot Generator) looks at the diode in the Plug Adapter and then signals the car via the Control Pilot signal. It also indicates what's up on 4 LEDs. See the Control Pilot Generator schematic to understand this section.

The PCBA is a 2-layer board with components on both sides. All components are through-hole type. When completed (and also before the bulky power supply was installed), both sides of the PCBA have been sprayed with a silicone conformal coating, primarily to prevent corrosion of the connections over time.

The Control Pilot Generator comprises the following circuits:

- The power supply
- The PIC microprocessor
- The Control Pilot driver
- The Control Pilot Sense circuit
- The Diode driver
- The Diode Sense circuit

The Power Supply

PS1, F1,F2, U1, C1-C3, C5

The power supply comprises a switching power supply made by Lambda, which takes anything between 85VAC and 265VAC (at 50 Hz or 60 Hz) in and produces +12V and – 12V. It is

protected by a pair of fuses on its inputs. Its output feeds a 7805 linear regulator, which produces +5V for the PIC microprocessor and the LEDs.

The high-voltage section of the PCB was laid out with safety in mind. The traces are separated from the low-voltage section and from each other as much as possible, including skipping every other pin on connector J1. Where possible, the high-voltage traces are beneath PS1 where they cannot be touched. When mounted to the lid of the box, all solder contacts in the high-voltage section are facing the lid to reduce opportunities to touch high voltage.

The PIC

U3, X1, J2, D1-D4, D7, R5-R8, R15, C4, C6, C7

The PIC microprocessor figures out what to do from its inputs, does the right thing on its outputs, and indicates what it is doing on its LEDs. To keep the Control Pilot signal timing within spec, the PIC's clock is derived from a crystal.

The PIC's EEPROM can be programmed in-circuit, using a Microchip PicKit 2 in-circuit programming device, plugged into connector J2. The circuit comprising D7, R15, and C4 holds the programming pin at the correct voltage, as recommended by Microchip.

The PIC indicates its state to the user via the 4 LEDs, D1-D4. (I used a variety of LED colors because I think the different LED colors are cool, particularly the ridiculously expensive cyan LEDs.)

As mentioned, the PIC is programmed in assembly language. This particular version of the PIC is quite primitive, with no timers and no interrupts. Since it must produce accurate timing signals on its outputs, the timing of the code has been accounted for in every routine, and code loops are used to produce the timing with perfect accuracy. (I verified my code timing with the oscilloscope.)

To minimize the opportunities for incorrect behavior due to glitches induced by electrostatic discharge or electromagnetic interference, all inputs are filtered in software, requiring several identical reads of each input before an input change is believed.

Read the firmware source code here. It is pretty self-explanatory for those of you with some experience in assembly language

The Control Pilot Driver

1/2 of U2, D8, D9, R2, R9, R11, C10

This circuit drives the Control Pilot signal, as generated by the PIC, to the correct voltages and with the correct output impedance.

The Control Pilot signal is driven with a fancy rail-to-rail op amp that compares the PIC's signal output voltage to a reference voltage made from 2 forward-biased diodes, causing the opamp to swing between one rail (+12V) and the other rail (-12V). Its output drives a 1K resister and a 300 pF capacitor, to meet the J1772 output impedance spec.

The Control Pilot Sense Circuit

D5, D6, R1, R4

This circuit allows the PIC to read the voltage of the Control Pilot signal during positive swings of the Control Pilot signal, so that the PIC can determine whether or not a car is present, and if that car is charging or not.

The two resistors of the Control Pilot sense circuit scale the positive swing of the Control Pilot signal such that it can be read by the PIC's D/A converter as a voltage between 0V and +5V. The resistor values of this divider were chosen such that the input impedance of the Control Pilot sense circuit is quite high relative to the output impedance of the Control Pilot driver. The Schottky diode, D6, prevents the PIC's A/D input from going much below 0V during negative

swings of the Control Pilot signal. The zener diode, D 5, prevents the PIC's A/D input from ever going above 4.7V, particularly in the case of a failure of R1.

The Diode Driver

1/2 of U2, D8, D9, R3, R12, R13

The diode driver uses a left-over gate in the dual rail-to-rail opamp to drive the diode in the Plug Adapter to +12V and -12V under control of the PIC.

The Diode Sense Circuit

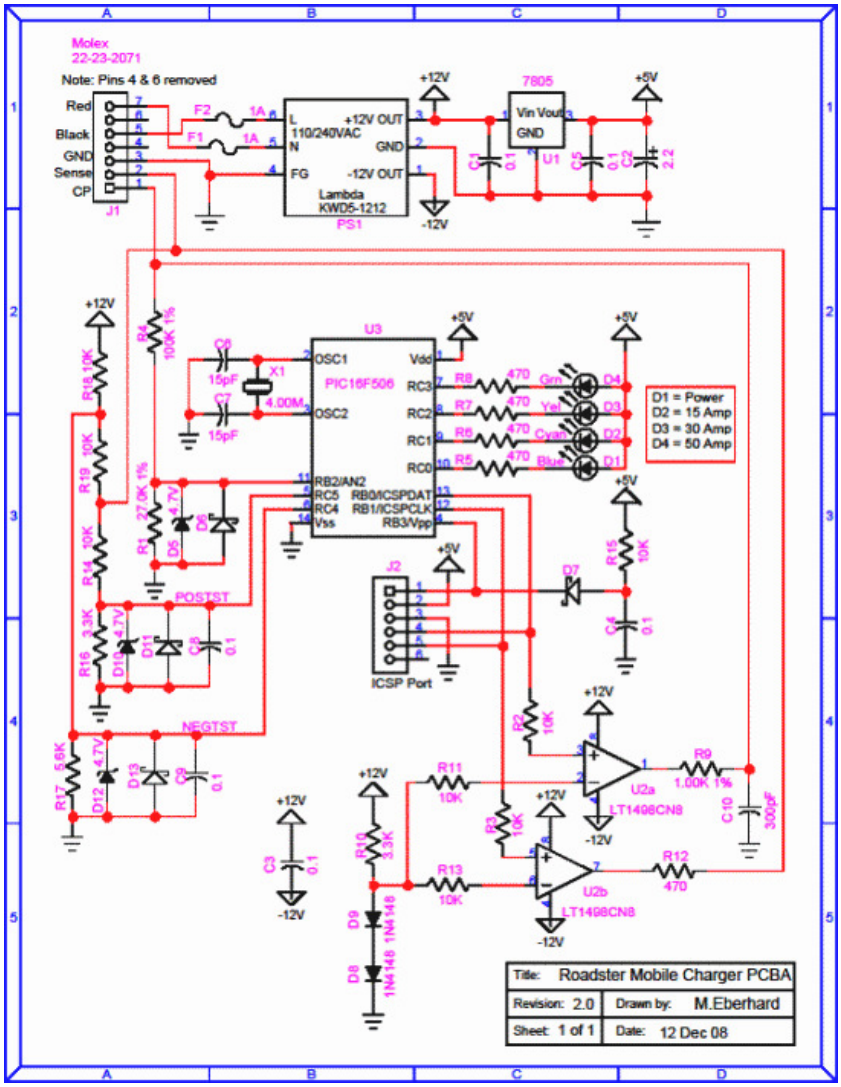
D10-D13, C8, C9, R14, R16-R19

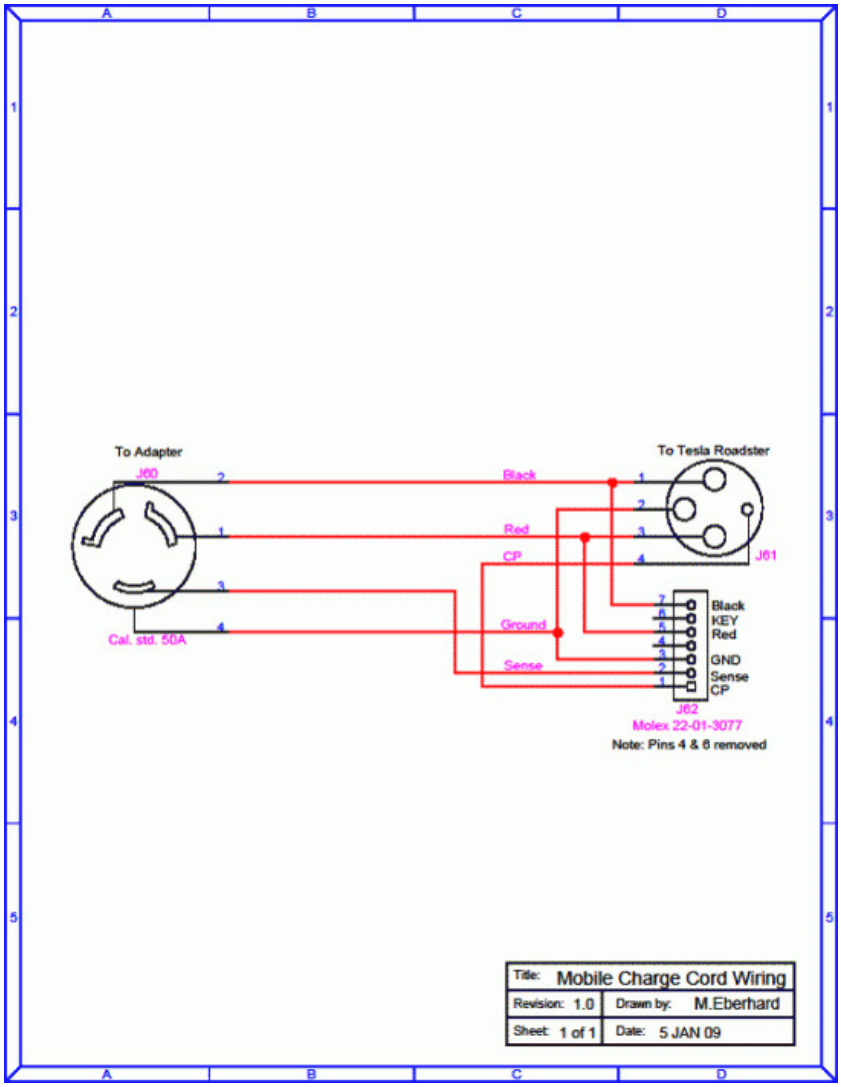
The PIC uses the Diode Sense circuit together with the Diode Driver to sense the presence and orientation of a diode (or shorted thermal switch) in the Plug Adapter. The Diode Sense circuit comprises two similar sub-circuits, each is a resistor divider, a pair of protection diodes, and a filter capacitor.

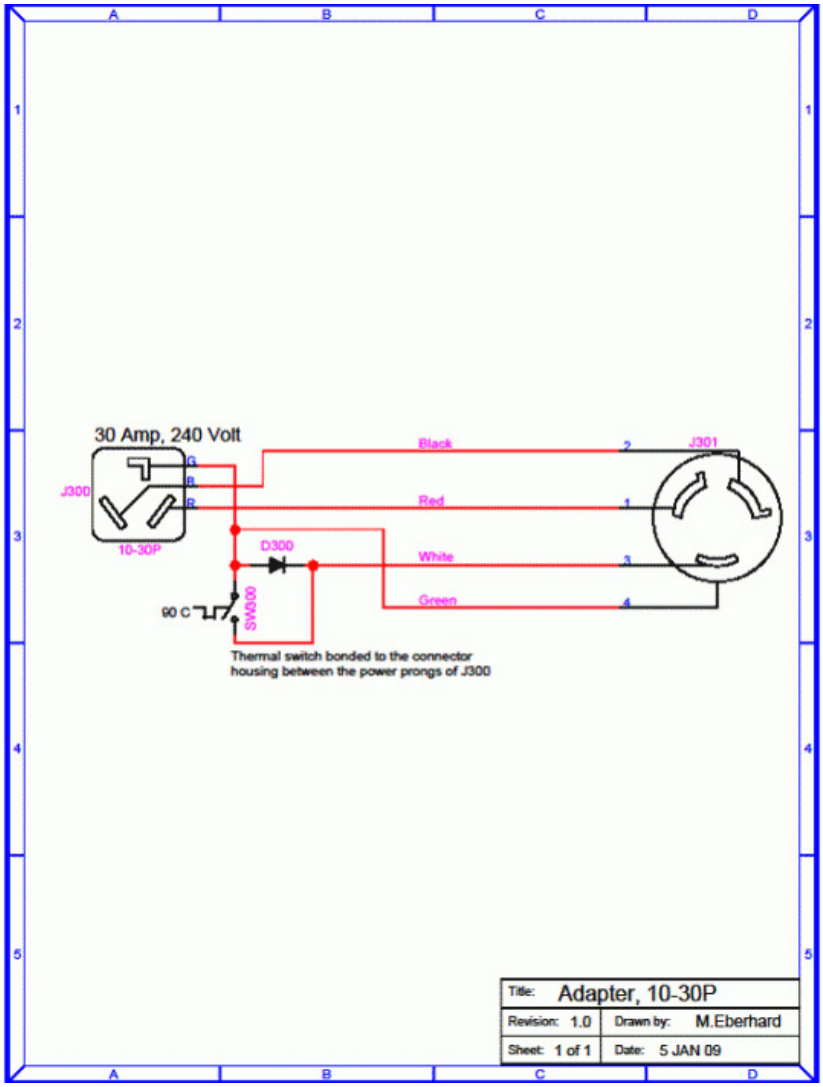
The first sub-circuit comprises D10, D11, R14, R16, and C8. This circuit allows the PIC to detect the presence of a diode with anode to ground – or perhaps a shorted thermal switch – that will prevent input RC5 from swinging much above ground when the Diode Driver is driven to +12V.

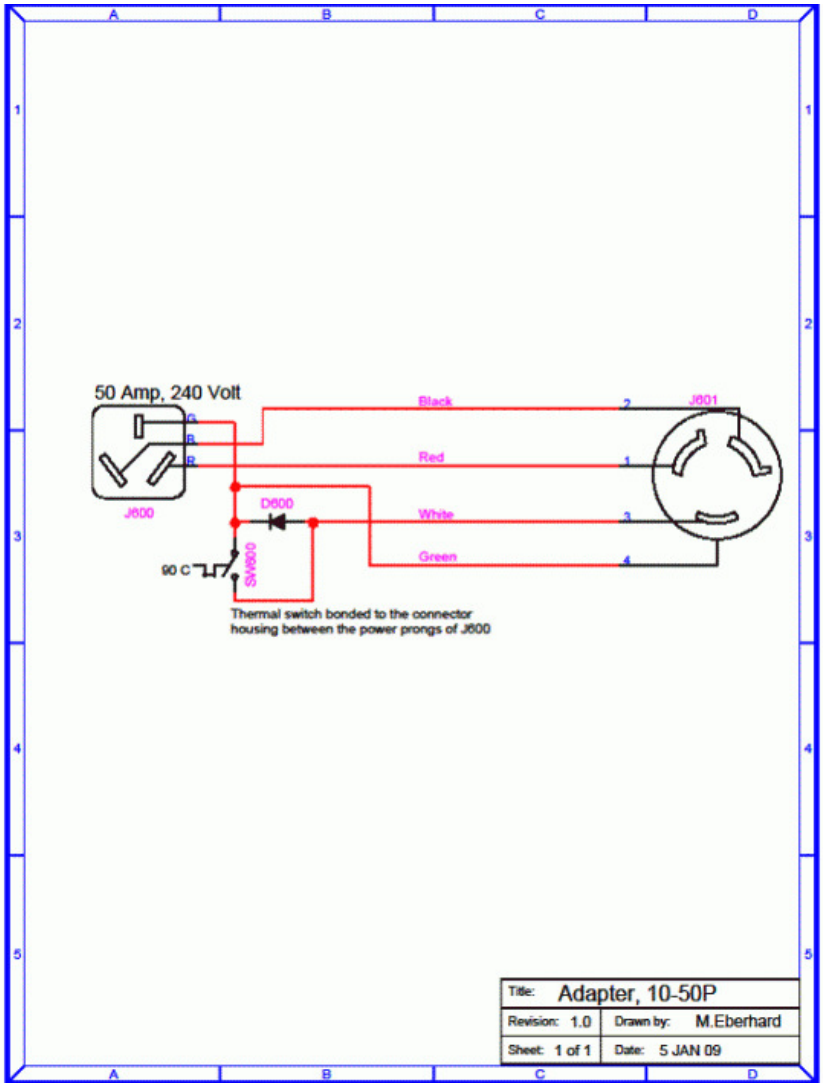
The second sub-circuit comprises D12, D13, R17-R19, and C9. This circuit allows the PIC to detect the presence of a diode with cathode to ground – or perhaps a shorted thermal switch – that will prevent input RC4 from swinging below 3.1V when the Diode Driver is driven to -12V.

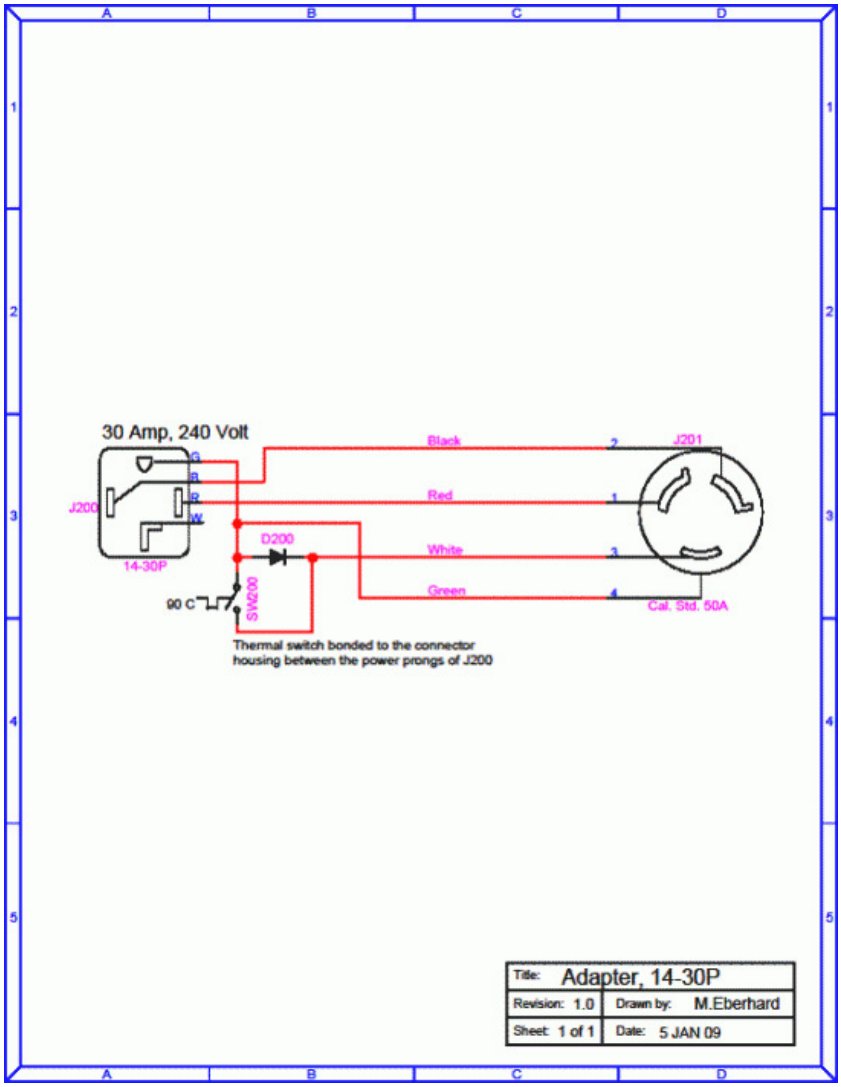
Schematics

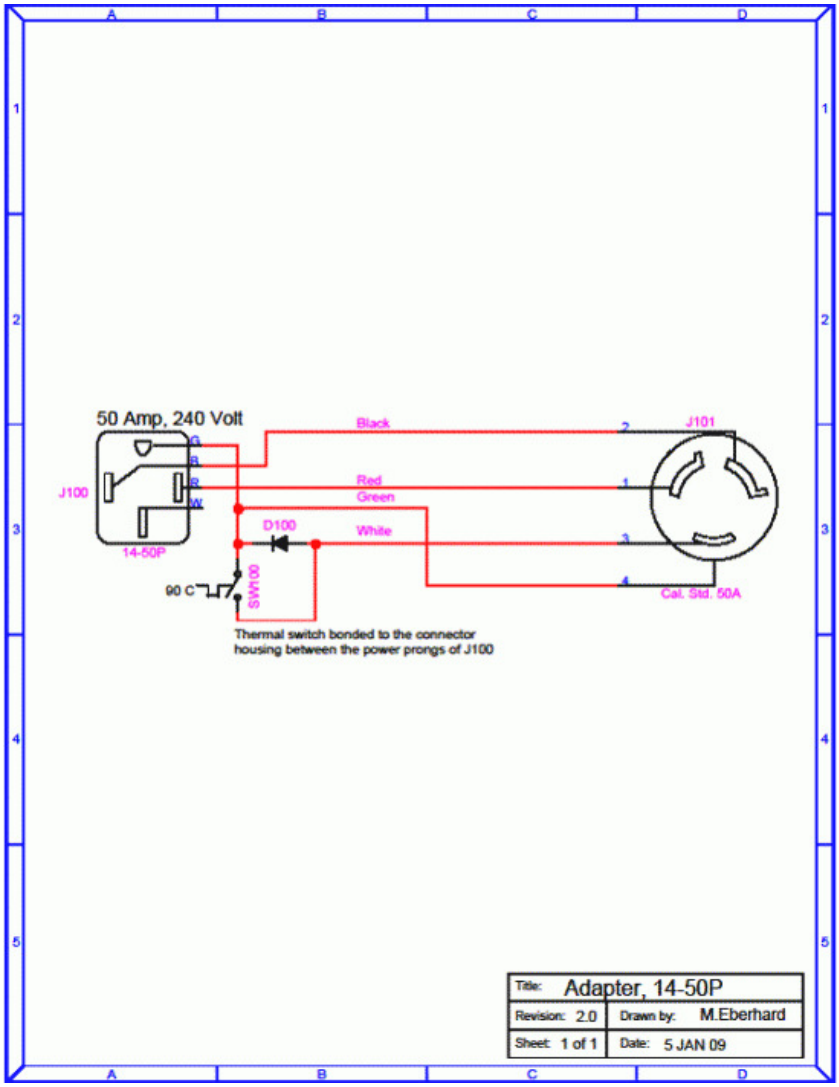

Molex
22-23-2071
Note: Pins 4 & 6 removed
Red
Black
GND
Sense
CP
J1
F2
F1
1A
1A
L
110/240VAC
N
FG
+12V OUT
GND
-12V OUT
Lambda
KWD5-1212
PS1
+12V
-12V
7805
Vin Vout
GND
U1
+5V
C1 0.1
C5 0.1
C2 2.2
U3
PIC16F506
OSC1
OSC2
Vdd
RC3
RC2
RC1
RC0
RB2/AN2
RC5 RB0/ICSPDAT
RC4 RB1/ICSPCLK
Vss RB3/Vpp
X1
4.00M
C6 15pF
C7 15pF
R4 100K 1%
R13 10K
R19 10K
R14 10K
R16 3.3K
R17 6.8K
R1 27.0K 1%
R8 470 Grn
R7 470 Yel
R6 470 Cyan
R5 470 Blue
D4
D3
D2
D1
D1 = Power
D2 = 15 Amp
D3 = 30 Amp
D4 = 50 Amp
D5 4.7V
D6
D10 4.7V
D11
D12 4.7V
D13
C8 0.1
C9 0.1
POSTST
NEGTST
J2
ICSP Port
D7
R15 10K
C4 0.1
R2 10K
R11 10K
R3 10K
R10 3.3K
R13 10K
U2a
U2b
LT1498CN8
R9 1.00K 1%
R12 470
C10 300pF
C3 0.1
D9 1N4148
D8 1N4148
Title: Roadster Mobile Charger PCBA
Revision: 2.0
Drawn by: M.Eberhard
Sheet: 1 of 1
Date: 12 Dec 08

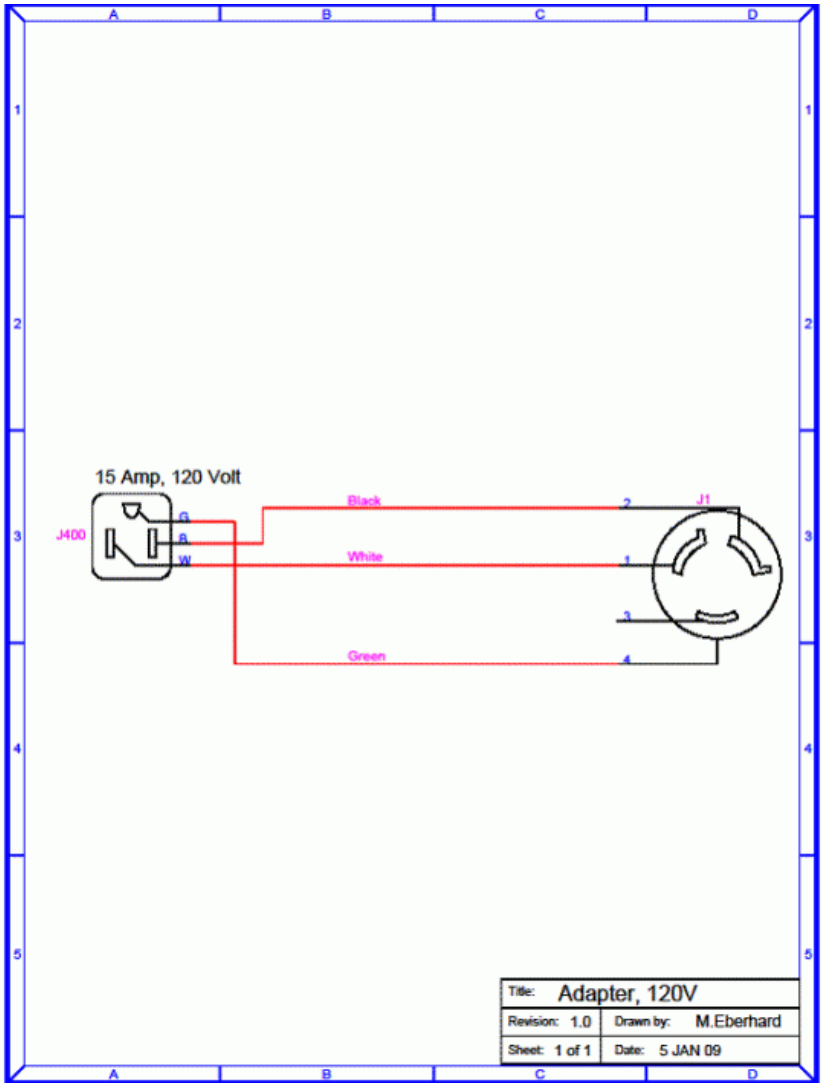

To Adapter
J60
2
1
3
4
Cal. std. 50A
Black
Red
CP
Ground
Sense
To Tesla Roadster
1
2
3
4
J61
7
6
5
4
3
2
1
Black
KEY
Red
GND
Sense
CP
J62
Molex 22-01-3077
Note: Pins 4 & 6 removed
Title: Mobile Charge Cord Wiring
Revision: 1.0
Drawn by: M.Eberhard
Sheet: 1 of 1
Date: 5 JAN 09


30 Amp, 240 Volt
J300
10-30P
G
B
R
Black
Red
White
Green
J301
2
1
3
4
D300
90 C
SW300
Thermal switch bonded to the connector
housing between the power prongs of J300
Title: Adapter, 10-30P
Revision: 1.0
Drawn by: M.Eberhard
Sheet: 1 of 1
Date: 5 JAN 09


50 Amp, 240 Volt
G
B
R
J600
Black
Red
White
Green
2
1
3
4
J601
D600
90 C
SW600
Thermal switch bonded to the connector
housing between the power prongs of J600
Title: Adapter, 10-50P
Revision: 1.0
Drawn by: M.Eberhard
Sheet: 1 of 1
Date: 5 JAN 09


30 Amp, 240 Volt
J200
14-30P
G
B
R
W
D200
90 C
SW200
Black
Red
White
Green
2
1
3
4
J201
Cal. Std. 50A
Thermal switch bonded to the connector
housing between the power prongs of J200
Title: Adapter, 14-30P
Revision: 1.0
Drawn by: M.Eberhard
Sheet: 1 of 1
Date: 5 JAN 09


50 Amp, 240 Volt
J100
14-50P
G
B
R
W
Black
Red
Green
White
D100
90 C
SW100
J101
2
1
3
4
Cal. Std. 50A
Thermal switch bonded to the connector
housing between the power prongs of J100
Title: Adapter, 14-50P
Revision: 2.0
Drawn by: M.Eberhard
Sheet: 1 of 1
Date: 5 JAN 09



Firmware

Main Code

```
    title "Roadster Foundry Charger Revision 2.01"
    subtitle "Copyright (C) 2009 Martin Eberhard"
    list b=4, c=132, n=80, p=PIC16F506
;_____________________________________________________________________________
;|                                                                           |
;|                                ROADSTER FOUNDRY                           |
;|___________________________________________________________________________|

;*****************************************************************************
;* Charger Firmware Main Code                                                *
;*                                                                           *
;*    Revision    Date         Author        Comments                        *
;*        1.01    2 Dec 08    M. Eberhard   Created                         *
;*        1.02    5 Dec 08    M. Eberhard   Added blinking while car connected  *
;*                                          but not ready/charging.          *
;*        1.03    7 Dec 08    M. Eberhard   Added firmware rev indicator    *
;*        1.04   27 Jan 09    M. Eberhard   Reversed LED order to match front   *
;*                                          panel labels; changed labeling from *
;*                                          50, 30, 15 amps to 40, 24, 16 amps  *
;*        1.05    31Jan09     M. Eberhard   Fix power LED while waiting for car *
;*        2.01    21May09     M. Eberhard   Change from 12-amp to 16-amp    *
;*****************************************************************************
REV_MAJOR       equ     .2      ;note: 0 is an illegal value
REV_MINOR       equ     .1      ;note: 0 is an illegal value
;*****************************************************************************
```

```
;* This program looks at the level if the DIODE line to the adaptor.            *
;*   o  40-amp charging is indicated by a diode in the adapter from the DIODE    *
;*      line to ground such that the DIODE line cannot go higher than 0.7V, but  *
;*      can go to -12V. For 40-amp charging, the 40-amp LED is turned on and a   *
;*      1 kHz square wave with a 667 uSec high time is generated on the Control  *
;*      Pilot line to the car.                                                   *
;*   o  24-amp charging is indicated by a diode in the adapter from the DIODE    *
;*      line to ground such that the DIODE line cannot go lower than -0.7V, but  *
;*      can go to +12V. For 24-amp charging, the 24-amp LED is turned on and a   *
;*      1 kHz square wave with a 400 uSec high time is generated on the Control  *
;*      Pilot line to the car.                                                   *
;*   o  16-amp charging is indicated by an open circuit in the adapter from the  *
;*       DIODE line to ground such that the DIODE line go to both +12V and -12V. *
;*      For 16-amp charging, the 16-amp LED is turned on and a 1 kHz square      *
;*      wave with a 267 uSec high time is generated on the Control Pilot line    *
;*      to the car.                                                              *
;*   o  Temperature sensors in the adapters can short the DIODE line to ground   *
;*      such that the DIODE line cannot be driven far from ground in either      *
;*      polarity. When over-temperature is detected, the Power LED is flashed    *
;*      at 2 Hz, and the Control Pilot signal is held at +12V.                   *
;*                                                                               *
;* This program measures the voltage on the CP line as late as possible while    *
;* CP is high. Based on the voltage it measures (see below), it decides whether  *
;* a car os present, and if so, whether the car is ready to charge (and is       *
;* probably also charging) or not.                                               *
;*   o  If no car is present, this firmware will blink the green LED slowly      *
;*      (about 0.5 Hz), with the other 3 LEDs off.                               *
;*   o  If a car is present but not ready, the green LED is steady-on, and the   *
;*      LED for the charging rate specified by the adapter blinks slowly (about  *
;*   o  0.5 Hz).                                                                 *
;*   o  If the car is ready and charging, then the green LED and the LED for     *
;*      the charging rate specified by the adapter are both steady-on.           *
;*                                                                               *
;* A/D Converter Results Interpretation                                          *
;*                                                                               *
;* The A/D Converter reads a divider on the CP line such that the voltage on     *
;* pin AD2 is 27/127 * Control Pulse voltage. To protect the PIC hardware,       *
;* this is diode-clipped from exceeding 4.7 volts ore going below -0.3 volts.    *
;*                                                                               *
;* CP_VOLTAGE Interpretation while CP is driven high:                            *
;*                                                                               *
;*   CP Voltage          AD2 Voltage      AD2 Value       Meaning                *
;*  11.25V<CP<12.75V  2.39V<AD2<2.71V   122<AD2<139   No car present            *
;*       10.50V            2.23V            114       This firmware's threshold  *
;*   8.25V<CP<9.75V   1.75V<AD2<2.07V    89<AD2<106   Car is present             *
;*        7.50V            1.59V             81       Threshold                  *
;*   5.25V<CP<6.75V   1.12V<AD2<1.44V    57<AD2< 74   Car is ready to charge     *
;*                                                                               *
;* Reads of the adapter diode and presence of a car are debounced - requiring    *
;* multiple sequential reads to be the same before doing anything new.           *
;* sequential reads are separated by a chunk of time.                            *
;*                                                                               *
;* This is an absolute program - no linker is required. However, the current     *
;* version ov MPLAB IDE (v8.10) does not work in assembly-only mode, so you      *
;* must use "Project>Build All" (which invokes the linker) to assemble this      *
;* program.                                                                      *
;*                                                                               *
;* Page references in the comments refer to the appropriate pages in Microchip   *
;* document DS41268D, "PIC12F510/16F506 Data Sheet."                             *
;*********************************************************************************
    page
;*********************************************************************************
;* Include standard header and charger header                                    *
;*********************************************************************************
    #include    P16F506.INC
    #include    CHARGER.INC
;*********************************************************************************
;* Reset Vector                                                                  *
;*********************************************************************************
    org RESET_VECTOR
    goto    init

;*********************************************************************************
;* Subroutines here because they must be in the first half of the page.          *
;*********************************************************************************
```

```
;*******************************************************************************
;* Subroutine ReadDiode                                                        *
;*                                                                             *
;* Read and debounce NEG_DIODE and POS_DIODE Subroutine                        *
;*                                                                             *
;* On entry: DIODE is set or cleared, making one of NEG_DIODE or POS_DIODE     *
;* reads meaningful.                                                           *
;*                                                                             *
;* Calls:    uSec                                                              *
;*                                                                             *
;* On exit: W = 0                                                              *
;*          R0 trashed                                                         *
;*          R1 = stable read of PORTC                                          *
;*******************************************************************************
ReadDiode:
        movlw    HOT_COUNT_MAX + .1
        movwf    HOT_COUNT
        movlw    0x0

;spin until we get HOT_COUNT_MAX identical reads of POS_DIODE and NEG_DIODE
RDLoop:
            movwf   R1            ;R1 remembers what the last read found

            movlw   DIODE_DELAY
            call    uSec          ;trashes R0

            movlw   (0x1 << POS_DIODE) | (0x1 << NEG_DIODE)
            andwf   PORTC,W       ;read POS_DIODE test bit

            xorwf   R1,F          ;test for same (Z set if same)
                                  ;R1 trashed; W has new value
            btfsc   STATUS,Z      ;if this read is like the last one, dec counter
            decf    HOT_COUNT,F

            btfss   STATUS,Z      ;quit loop if we counted down to 0
            goto    RDLoop

        movwf   R1
        retlw   0x0
    page
;*******************************************************************************
;* Subroutine TestCP                                                           *
;*                                                                             *
;* Read and interpret Control Pulse Subroutine                                 *
;*                                                                             *
;* On entry: Don't care                                                        *
;*                                                                             *
;* Calls:    None                                                              *
;*                                                                             *
;* On exit: W = 0                                                              *
;*          C = 0 if a car was detected                                        *
;*          R0 = A/D Result                                                    *
;* Time:    24 cycles                                                          *
;*******************************************************************************
TestCP:
TEST_CP_CYCLES    equ    .24    ;cycles consumed during this subroutine

        bsf   ADCON0,GO    ;(1 cycle) start the A/D conversion

                           ;this loop will spin 5 times, consuming 17 cycles)
WaitADC:    btfsc   ADCON0,NOT_DONE    ;wait for conversion to complete
            goto    WaitADC

        movf    ADRES,W    ;(1 cycle)
        movwf   R0         ;(1 cycle) save A/D result
        movlw   NO_CAR     ;(1 cycle) if A/D result is higher than this, then no car
        subwf   R0,W       ;(1 cycle)compare the A/D result to the threshold

        retlw   0x0        ;(2 cycles)

    page
;*******************************************************************************
;* Subroutine Pulse                                                            *
;*                                                                             *
```

```
;* Create high-pulse on CP and test for car presence Subroutine                 *
;*                                                                              *
;* The CP signal is sampled about 39 uSec before the end of the high pulse.     *
;* This gives a minimum of 200-39=141 uSec for the CP signal to overcome RC     *
;* delay and reach its final value.                                             *
;*                                                                              *
;* If CAR_COUNT_MAX successive calls to this subroutine find no car present,    *
;* rudely jump to init.                                                         *
;*                                                                              *
;* On entry: W = low-time delay (0 means 256)                                   *
;*           R1 = high-time delay                                               *
;*           R2 = LED mask for particular charging rate                         *
;*           R3 = Port C value                                                  *
;*           CAR_COUNT has a down-count of the number of sequential times we    *
;*              did not see a car present                                       *
;*           CP bit (in PORTB) is high                                          *
;*                                                                              *
;* Calls:    uSec                                                               *
;*           TestCP                                                             *
;*                                                                              *
;* On exit: W = 0                                                               *
;*          R0-R1 trashed                                                       *
;*          CAR_COUNT is decremented if no car found,                           *
;*             set to CAR_COUNT_MAX + 1 if car detected.                        *
;*          CP bit (in PORTB) is low                                            *
;*                                                                              *
;* high time: (R1 * 4) + 28 + TEST_CP_CYCLES cycles                             *
;* low time: (W * 4) + 8 cycles                                                 *
;********************************************************************************
Pulse:
PULSE_HIGH_CYCLES   equ   TEST_CP_CYCLES + .28 ;count cycles while CP is high in this subroutine
PULSE_LOW_CYCLES    equ   .8        ;count cycles while CP is low here

        call    uSec                ;(W + 3 cycles low) Finish the low pulse
        bsf     PORTB, CP           ;(1 cycle low) Set CP high

        movf    R1,W                ;(1 cycle high) Get high delay
        call    uSec                ;(W * 4 + 3 cycles high)

;See if there is still a car around
        call    TestCP              ;(2+TEST_CP_CYCLES cycles high) C=0 if car there, R0=ADRES
        movlw   CAR_COUNT_MAX + .1  ;(1 cycle)
        btfss   STATUS,C            ;(2 cycles high if taken, 1 otherwise)
        movwf   CAR_COUNT           ;(1 cycle high)car detected: restart count

        decf    CAR_COUNT,F         ;(1 cycle high)
        btfsc   STATUS,Z            ;(2 cycles high if taken)
        goto    init                ; Start over if the car disappeared

;See if the car is not actually charging and we need to blink the given LED
        movlw   CAR_READY           ;(1 cycle high)
        subwf   R0,W                ;(1 cycle) C=1 if car is charging
        btfss   STATUS,C            ;(2 cycles high if taken, 1 otherwise)
        goto    IsCharging          ;(2 cycles)

;the car is not charging: decrement the 16-bit blink counter
        decf    TIMER_HIGH,W        ;(1 cycle high)16-bit counter decrement
        decfsz  TIMER_LOW,F         ;(1 cycle high)
        movf    TIMER_HIGH,W        ;(1 cycle high)
        movwf   TIMER_HIGH          ;(1 cycle high)
;See if the 16-bit blink counter reached zero and bail of not
        iorwf   TIMER_LOW,W         ;(1 cycle high)Test for underflow, and restart timer if so
        btfss   STATUS,Z            ;(2 cycles high if taken, 1 otherwise)
        goto    NoLEDChange         ;(2 cycles high)

;the 16-bit blink counter went to 0, so reload the upper byte
        movlw   READY_BLINK         ;(1 cycle high)
        movwf   TIMER_HIGH          ;(1 cycle high)
        movf    R2,W                ;(1 cycle high)
        xorwf   R3,W                ;(1 cycle high)
        movwf   R3                  ;(1 cycle high)
        movwf   PORTC               ;(1 cycle high)
;set CP low and we are done
SetLowDone:
        bcf     PORTB,CP           ;(1 cycle high) set CP low
```

```
        retlw   0x0               ;(2 cycles low)

IsCharging:                       ;need 12 cycles
;Car is charginng: turn on the correct LED and the power LED
        movlw  (0x1 << NOT_GREEN) | (0x1 << NOT_YELLOW) | (0x1 << NOT_CYAN)
                                  ;(1 cycle high)
        xorwf   R2,W              ;(1 cycle high)
        movwf   PORTC             ;(1 cycle high)
;stall for the correct number of cycles
        goto    skip1             ;(2 cycles high)
skip1:  goto    skip2             ;(2 cycles high)
skip2:
NoLEDChange:                      ;need 5 cycles
        goto    skip3             ;(2 cycles high)
skip3:  nop                       ;(1 cycle high)
        goto    SetLowDone        ;(2 cycles high)

    page
;*****************************************************************************
;* Subroutine mSec                                                           *
;*                                                                           *
;* Millisecond Delay Subroutine                                              *
;*                                                                           *
;* Waste time for (W * 4000) cycles (4 mSec)                                 *
;*                                                                           *
;* On entry: W = desired delay (0 means 256)                                 *
;*                                                                           *
;* Calls:    None                                                            *
;*                                                                           *
;* On exit: W = 0, R0=0, R1 trashed                                          *
;*****************************************************************************
mSec:    movwf   R1

mSecLoop:
            call    uSec1000
            call    uSec1000
            call    uSec1000
            call    uSec1000
            decfsz  R1,F
            goto    mSecLoop

        retlw  0x0                ;(2 cycles)

;*****************************************************************************
;* Subroutine uSec1000                                                       *
;*                                                                           *
;* 1,000 Microsecond Delay Subroutine                                        *
;*                                                                           *
;* Waste time for 1,000 cycles                                               *
;*                                                                           *
;* Calls: None (falls into uSec)                                             *
;*                                                                           *
;* On exit: W = 0, R0=0                                                      *
;*****************************************************************************
uSec1000:
            movlw    .242         ;account for the time of the calls, etc.

; fall into uSec

;*****************************************************************************
;* Subroutine uSec                                                           *
;*                                                                           *
;* Microsecond Delay Subroutine                                              *
;* (See page 40) With a 4 MHz crystal, 1 cycle = 1 uSec.                     *
;*                                                                           *
;* Waste time for (W * 4) + 1 cycles                                         *
;*                                                                           *
;* On entry: W = desired delay (0 means 256)                                 *
;*                                                                           *
;* Calls:    None                                                            *
;*                                                                           *
;* On exit: W = 0, R0=0                                                      *
;*****************************************************************************
uSec:
uSecLoop:    movwf  R0           ;(1 cycle) the slow way to make loop 4 cycles
```

```
            decfsz  R0,W        ;(1 cycle except last time, which takes 2 cycles)
            goto    uSecLoop    ;(2 cycles)

        retlw  0x0              ;(2 cycles)

    page
;*************************************************************************
;* Initialization Routine                                                *
;*************************************************************************
init:
        movlw   OPTION_INIT
        option

        movlw   TRISB_INIT
        tris    0x6
        movlw   TRISC_INIT
        tris    0x7

        movlw   CM1CON0_INIT
        movwf   CM1CON0
        movlw   CM2CON0_INIT
        movwf   CM2CON0

        movlw   VRCON_INIT
        movwf   VRCON

        movlw   ADCON0_INIT
        movwf   ADCON0

        movlw   PORTB_INIT
        movwf   PORTB

        movlw   PORTC_INIT
        movwf   PORTC        ;This turns on the green LED

        movlw   FSR_INIT
        movwf   FSR

        clrf    STATUS      ;(page 18)make sure CALLs work correctly - clear bits 7-5
;Fall into BlinkRev

;*************************************************************************
;* Indicate the firmware version number on the LEDs                      *
;*                                                                       *
;* This turns on the Cyan LED, then blinks out the major revision number on      *
;* the Green LED (leaving it off when done), then blinks out the minor rev       *
;* number on the Yellow LED (leaving it off when done), then turns off the       *
;* Cyan LED.                                                             *
;*                                                                       *
;* On entry: The Blue (power) LED is on                                  *
;*                                                                       *
;* Calls:    mSec (which calls uSec)                                     *
;*                                                                       *
;* On Exit: NOT_BLUE = 0                                                 *
;*          W, R0, R1, R2, CAR_COUNT trashed                             *
;*************************************************************************
BlinkRev:
    bcf     PORTC,NOT_CYAN         ;turn on LED to indicate rev number output
    movlw   REV_BLINK              ;delay to look pretty
    call    mSec

    movlw  REV_MAJOR               ;Major revision number first
    movwf  R2
MajorLoop:
        bcf     PORTC,NOT_GREEN    ;turn led on
        movlw   REV_BLINK
        call    mSec
        bsf     PORTC,NOT_GREEN    ;turn led off
        movlw   REV_BLINK
        call    mSec
        decfsz  R2,F
        goto    MajorLoop

    movlw   REV_MINOR               ;Next the minor revision number
    movwf   R2
```

```
MinorLoop:
        bcf     PORTC,NOT_YELLOW    ;turn led on
        movlw   REV_BLINK
        call    mSec
        bsf     PORTC,NOT_YELLOW    ;turn led off
        movlw   REV_BLINK
        call    mSec
        decfsz  R2,F
        goto    MinorLoop

    bsf   PORTC,NOT_CYAN            ;All done: yellow LED off

;Fall into CarWait
    page
;*****************************************************************************
;* Hang and blink Blue (power) LED slowly if no car is detected              *
;*                                                                           *
;*                                                                           *
;* On entry: Don't care                                                      *
;*                                                                           *
;* Calls:    TestCP, uSec                                                    *
;*                                                                           *
;* On Exit: NOT_BLUE = 0 (Power LED is on)                                   *
;*          W, R0, R1, R2, CAR_COUNT trashed                                 *
;*****************************************************************************
CarWait:
        movlw   CAR_COUNT_MAX + .1
        movwf   CAR_COUNT

; Loop here until we see the car present for CAR_COUNT_MAX reads in a row
; read CP twice per half-cycle of the LED.
; R2 is used to do every-other-time for half-cycle. Initial value does not matter.
CarWaitLoop:
            movlw   (0x1 << NOT_BLUE)   ;toggle the LED
            xorwf   R2,F                ;toggle relevent bit in R2
            andwf   R2,W                ;mask off all the other bits
            xorwf   PORTC,F

            movlw   WAIT_BLINK/2       ;delay for a long while
            call    mSec               ;trashes R0, R1

            call    TestCP             ;C = 0 if a car is there

            movlw   CAR_COUNT_MAX + .1 ;decrement counter if car detected
            btfsc   STATUS,C
            movwf   CAR_COUNT          ;no car: restart counter

            decfsz  CAR_COUNT,F
            goto    CarWaitLoop

;Fall into HotTest
    page
;*****************************************************************************
;* Hot Test                                                                  *
;*                                                                           *
;* Read the diode in the adaptor, debounce it, and test for too hot.         *
;* The adaptor will connect the sense wire to ground when it is hot.         *
;* To test for connection to ground, DIODE must be driven high and tested with   *
;* DIODE_POS, then driven low and tested with DIODE_NEG.                     *
;*                                                                           *
;* On entry: DIODE = 1 a long time ago                                       *
;*           CP = 1                                                          *
;*                                                                           *
;* Calls: ReadDiode (which also calls uSec)                                  *
;*          uSec                                                             *
;*                                                                           *
;* On Exit: DIODE = 0                                                        *
;*          POS_DIODE bit of R2 = 1 if the DIODE line was driven to +12V     *
;*          NEG_DIODE bit of R2 = 1 if the DIODE line was driven to -12V     *
;*          all other bits of R2 = 0, and R2 does not equal 0                *
;*          Power LED is on, other LEDs off, R3 = value written to Port C    *
;*          W, R0, R1 trashed                                                *
;*****************************************************************************
HotTest:
        movlw   PORTC_INIT
```

```
        movwf   PORTC               ;This turns on the power LED, the others off
        movwf   R3

;Wait for a stable read of the diodes while DIODE is high
        call    ReadDiode

        bcf     PORTB,DIODE         ;drive diode test line low

        movlw   (0x1 << POS_DIODE)  ;save the positive diode result
        andwf   R1,W                ;R1 has the stable read of the diode
        movwf   R2                  ;r2=0 means DIODE did not go high

        movlw   DIODE_DELAY         ;wait for DIODE line to stabilize
        call    uSec                ;trashes R0

        call    ReadDiode

        movlw   (0x1 << NEG_DIODE)  ;invert and save the negative diode result
        xorwf   R1,F
        andwf   R1,W                ;r1=0 means DIODE did not go low
        iorwf   R2,F                ;combine negative read with positive read
                                    ;r2 = 0 means DIODE line shorted to ground

        btfss  STATUS,Z             ;adapter is hot if DIODE line is shorted to ground
        goto   TestHighRate

;fall into HotHang if adapter is hot

;*****************************************************************************
;* Hot Hang                                                                  *
;*                                                                           *
;* Hang forever and fast-blink all LEDs since the adapter temp is hot        *
;* On entry: NOT_BLUE = 0                                                    *
;*           NOT_CYAN = 1                                                    *
;*           NOT_YELLOW = 1                                                  *
;*           NOT_GREEN = 1                                                   *
;*           R3 = Port C value                                               *
;* (there is no exit.)                                                       *
;*****************************************************************************
HotHang:
        bsf    PORTB,CP    ;drive control pulse high permanently, just in case

HotHangLoop:
            movlw  ERROR_BLINK
            call   mSec            ;trashes R0, R1

            movlw  (0x1 << NOT_BLUE) | (0x1 << NOT_CYAN) | (0x1 << NOT_YELLOW) | (0x1 << NOT_GREEN)
                                   ;toggle all LEDs
            xorwf  PORTC,F
            goto   HotHangLoop     ;loop here forever.
    page
;*****************************************************************************
;* 40-amp charging loop                                                      *
;*                                                                           *
;* Test for 40-amp charging and temperature okay.                            *
;* Create 1 KHz Control Pilot (CP) signal if so.                             *
;*    High Time: nominally 667 uSec, and between 660 uSec and 795 uSec       *
;*    Frequency: nominally 1000 Hz, between 995 Hz and 1005 Hz               *
;* 40-amp charging is indicated by a diode from the DIODE line to ground, such   *
;* that the DIODE line cannot go higher than 0.7V, but can go to -12V.       *
;*                                                                           *
;* Hot will be indicated by NEG_DIODE becoming 1 while DIODE = 0.            *
;*                                                                           *
;* On entry: DIODE = 0                                                       *
;*            POS_DIODE bit of R2 = 1 if the DIODE line was driven to +12V   *
;*            NEG_DIODE bit of R2 = 1 if the DIODE line was driven to -12V   *
;*            all other bits of R2 = 0, and R2 does  not equal 0             *
;*            R3 = Port C value                                              *
;*                                                                           *
;* Calls: Pulse (which calls uSec and TestCP)                                *
;*                                                                           *
;* Register usage:                                                           *
;*        R0 is trashed in some subroutines                                  *
;*        R1 is a temp register, trashed regularly                           *
;*        R2 is a mask for the Green LED in PORT C                           *
```

```
;*        HOT_COUNT counts down sequential times the adapter seemed hot          *
;*        CAR_COUNT counts down sequential times the car seemed gone             *
;*                                                                               *
;* Exit conditions: Car really not present causes restart to init                *
;*                    Adapter really hot causes rude jump to HotWait             *
;*                                                                               *
;* Scope Measurements:                                                           *
;*   High time = 668.7 uSec                                                      *
;*   Frequency = 1000 Hz (Period = 1000 uSec)                                    *
;*********************************************************************************
TestHighRate:
;initialize counters
        movlw  HOT_COUNT_MAX + .1
        movwf  HOT_COUNT
        movlw  CAR_COUNT_MAX + .1
        movwf  CAR_COUNT
        movlw  READY_BLINK
        movwf  TIMER_HIGH

;see if the adapter really is a 40-amp one
    btfsc  R2,POS_DIODE           ;not 40 amps if DIODE line went to +12V
    goto   TestMidRate

    movlw  (0x1 << NOT_GREEN)     ;get ready to turn on 40-amp LED
    movw   R2

;*-------------------------------------------------------------------------*
;* Count the cycles while CP is high and low in this routine               *
;*-------------------------------------------------------------------------*
TEST40_LOW_CYCLES    equ    PULSE_LOW_CYCLES + .12
TEST40_HIGH_CYCLES   equ    PULSE_HIGH_CYCLES

LoopHighRate:
;*-------------------------------------------------------------------------*
;* Test NEG_DIODE (With DIODE low) to see if the adapter is hot.           *
;* (High means hot.) If it really is hot (it seems hot too many times      *
;* in a row), then abort rudely!                                           *
;*-------------------------------------------------------------------------*

        movlw  HOT_COUNT_MAX + .1   ;(1 cycle low)
        btfss  PORTC,NEG_DIODE      ;(2 cycles low if taken, 1 otherwise)
        movwf  HOT_COUNT            ;(1 cycle low) not hot: restart count

        decf   HOT_COUNT,F          ;(1 cycle)
        btfsc  STATUS,Z             ;(2 cycles low) Z means underflow
        goto   HotTest              ;double-check for too hot

;*-------------------------------------------------------------------------*
;* Finish the low pulse, set CP high, then delay for the maximum time      *
;* before checking to see if the car is still there.                       *
;* A car is not there if two reads during two successive cycles of CP      *
;* both indicate no car present. If no car is there, go init.              *
;*-------------------------------------------------------------------------*

        movlw  (HIGH_40AMP-TEST40_HIGH_CYCLES)/.4  ;(1 cycle low) high delay in R1
        movwf  R1                   ;(1 cycle low)
        movlw  ((CP_PERIOD - HIGH_40AMP) - TEST40_LOW_CYCLES)/.4
                                    ;(1 cycle low)

        call   Pulse                ; high time: (R1 * 4) + PULSE_HIGH_CYCLES
                                    ; low time: (W * 4) + 2 + PULSE_LOW_CYCLES
        goto   LoopHighRate         ;(2 cycles low)

    page
;*********************************************************************************
;* 24-amp charging loop                                                          *
;*                                                                               *
;* Test for 24-amp charging and temperature okay.                                *
;* Create 1 KHz Control Pilot (CP) signal if so.                                 *
;*    High Time: nominally 400 uSec and between 395 uSec and 525 uSec            *
;*    Frequency: nominally 1000 Hz, between 995 Hz and 1005 Hz                   *
;* 24-amp charging is indicated by a diode from the DIODE line to ground, such   *
;* that the DIODE line cannot go lower than 10.7V, but can go to +12V.           *
;*                                                                               *
;* Hot will be indicated by POS_DIODE becoming 0 while DIODE = 1.                *
```

```
;*                                                                              *
;* On entry: DIODE = 0                                                          *
;*           POS_DIODE bit of R2 = 1, since the DIODE line was driven to +12V   *
;*           NEG_DIODE bit of R2 = 1 if the DIODE line was driven to -12V       *
;*           all other bits of R2 = 0                                           *
;*           R3 = Port C value                                                  *
;*                                                                              *
;* Calls: Pulse (which calls uSec and TestCP)                                   *
;*                                                                              *
;* Register usage:                                                              *
;*        R0 is trashed in some subroutines                                     *
;*        R1 is a temp register, trashed regularly                              *
;*        R2 is a mask for the Yellow LED in PORT C                             *
;*        HOT_COUNT counts down sequential times the adapter seemed hot         *
;*        CAR_COUNT counts down sequential times the car seemed gone            *
;*                                                                              *
;* Exit conditions: Car really not present causes restart to init               *
;*                  Adapter really hot causes rude jump to HotWait              *
;*                                                                              *
;* Scope Measurements:                                                          *
;*   High time = 399.5 uSec                                                     *
;*   Frequency = 1000 Hz (Period = 1000 uSec)                                   *
;********************************************************************************
TestMidRate:
    bsf    PORTB,DIODE             ;plan ahead to test for hot - set DIODE high
    movlw  DIODE_DELAY             ;wait for it to settle
    call   uSec

;see if the adapter really is a 24-amp one
    btfsc  R2,NEG_DIODE            ;not 24 amps if DIODE line went to -12V
    goto   TestLowRate

    movlw   (0x1 << NOT_YELLOW)    ;get ready to turn on 24-amp LED
    movwf   R2

;*------------------------------------------------------------------------*
;* Count the cycles while CP is high and low in this routine              *
;*------------------------------------------------------------------------*
TEST24_LOW_CYCLES    equ    PULSE_LOW_CYCLES + .12
TEST24_HIGH_CYCLES   equ    PULSE_HIGH_CYCLES

LoopMidRate:
;*------------------------------------------------------------------------*
;* Test POS_DIODE (With DIODE high) to see if the adapter is hot.         *
;* (Low means hot.) If it really is hot (it seems hot too many times in   *
;* a row), then abort rudely!                                             *
;*------------------------------------------------------------------------*

        movlw  HOT_COUNT_MAX + .1  ;(1 cycle low)
        btfsc  PORTC,POS_DIODE     ;(2 cycles low if taken, 1 otherwise)
        movwf  HOT_COUNT           ;(1 cycle low) not hot: restart count

        decf   HOT_COUNT,F         ;(1 cycle low)
        btfsc  STATUS,Z            ;(2 cycles low) Z means underflow
        goto   HotTest             ;double-check for too hot

;*------------------------------------------------------------------------*
;* Finish the low pulse, set CP high, then delay for the maximum time     *
;* before checking to see if the car is still there.                      *
;* A car is not there if two reads during two successive cycles of CP     *
;* both indicate no car present. If no car is there, go init.             *
;*------------------------------------------------------------------------*

        movlw (HIGH_24AMP-TEST24_HIGH_CYCLES)/.4  ;(1 cycle low) put high delay in R1
        movwf  R1                  ;(1 cycle low)
        movlw  ((CP_PERIOD - HIGH_24AMP) - TEST24_LOW_CYCLES)/.4
                                   ;(1 cycle low)

        call   Pulse               ; high time: (R1 * 4) + PULSE_HIGH_CYCLES
                                   ; low time: (W * 4) + 2 + PULSE_LOW_CYCLES
        goto   LoopMidRate         ;(2 cycles low)

    page
;********************************************************************************
;* 16-amp charging loop                                                         *
```

```
;*                                                                              *
;* Must be 16-amp charging. Test for temperature okay.                          *
;* Create 1 KHz Control Pilot (CP) signal if so.                                *
;*    High Time: nominally 267 uSec and between 260 uSec and 395 uSec           *
;*    Frequency: nominally 1000 Hz, between 995 Hz and 1005 Hz                  *
;* 16-amp charging is indicated by nothing connected to the DIODE line, such    *
;* that the DIODE line can go to -12V, and also can go to +12V.                 *
;*                                                                              *
;* Hot will be indicated by POS_DIODE becoming 0 while DIODE = 1.               *
;*                                                                              *
;* On entry: DIODE = 1                                                          *
;*           POS_DIODE bit of R2 = 1, since the DIODE line was driven to +12V   *
;*           NEG_DIODE bit of R2 = 1, since the DIODE line was driven to -12V   *
;*           all other bits of R2 = 0                                           *
;*           R3 = Port C value                                                  *
;*                                                                              *
;* Calls: Pulse (which calls uSec and TestCP)                                   *
;*                                                                              *
;* Register usage:                                                              *
;*        R0 is trashed in some subroutines                                     *
;*        R1 is a temp register, trashed regularly                              *
;*        R2 is a mask for the Cyan LED in PORT C                               *
;*        HOT_COUNT counts down sequential times the adapter seemed hot         *
;*        CAR_COUNT counts down sequential times the car seemed gone            *
;*                                                                              *
;* Exit conditions: Car really not present causes restart to init               *
;*                  Adapter really hot causes rude jump to HotWait              *
;*                                                                              *
;* Scope Measurements:                                                          *
;*   High time = ??? uSec                                                       *
;*   Frequency = 1000 Hz (Period = 1000 uSec)                                   *
;********************************************************************************
TestLowRate:
    movlw   (0x1 << NOT_CYAN)        ;get ready to turn on 16-amp LED
    movwf   R2

;*------------------------------------------------------------------------*
;* Count the cycles while CP is high and low in this routine              *
;*------------------------------------------------------------------------*
TEST16_LOW_CYCLES    equ    PULSE_LOW_CYCLES + .12
TEST16_HIGH_CYCLES   equ    PULSE_HIGH_CYCLES

LoopLowRate:
;*------------------------------------------------------------------------*
;* Test POS_DIODE (With DIODE high) to see if the adapter is hot.         *
;* (Low means hot.) If it really is hot (it seems hot too many times in   *
;* a row), then abort rudely!                                             *
;*------------------------------------------------------------------------*

        movlw  HOT_COUNT_MAX + .1  ;(1 cycle low)
        btfsc  PORTC,POS_DIODE     ;(2 cycles low if taken, 1 otherwise)
        movwf  HOT_COUNT           ;(1 cycle low) not hot: restart count

        decf   HOT_COUNT,F         ;(1 cycle)
        btfsc  STATUS,Z            ;(2 cycles low) Z means underflow
        goto   HotTest             ;double-check for too hot

;*------------------------------------------------------------------------*
;* Finish the low pulse, set CP high, then delay for the maximum time     *
;* before checking to see if the car is still there.                      *
;* A car is not there if two reads during two successive cycles of CP     *
;* both indicate no car present. If no car is there, go init.             *
;*------------------------------------------------------------------------*

        movlw  (HIGH_16AMP - TEST16_HIGH_CYCLES)/.4  ;(1 cycle low) put high delay in R1
        movwf  R1                  ;(1 cycle low)
        movlw  ((CP_PERIOD - HIGH_16AMP) - TEST16_LOW_CYCLES)/.4
                                   ;(1 cycle low)

        call   Pulse               ; high time: (R1 * 4) + PULSE_HIGH_CYCLES
                                   ; low time: (W * 4) + 2 + PULSE_LOW_CYCLES
        goto   LoopLowRate         ;(2 cycles low)

    end
```

Charger Header File

```
;_____________________________________________________________________________
;|                                                                            |
;|                                ROADSTER FOUNDRY                            |
;|____________________________________________________+_______________________|
;*****************************************************+***********************
;* Charger Firmware Header File                                               *
;*                                                                            *
;* o  Tell the assembler to set up the configuration register when the chip is   *
;*    programmed: External oscillator, Master Clear disabled, no code         *
;*    protection, no watchdog                                                 *
;* o  Name the I/O pins. Note that pin CP_VOLTAGE is defined as an ADC input.    *
;* o  Define inital values for all the PIC registers                          *
;* o  Define a bunch of useful constants                                      *
;* o  Name a few RAM locations that are used as registers and counters        *
;*                                                                            *
;* Page references in the comments refer to the appropriate pages in Microchip   *
;* document DS41268D, "PIC12F510/16F506 Data Sheet."                          *
;*****************************************************************************
    __config     _XT_OSC & _MCLRE_OFF & _CP_OFF & _WDT_OFF

;*-----------------------------------------------*
;* I/O pin definitions                           *
;*-----------------------------------------------*
CP              equ    RB0    ;output to control pulse signal to car
DIODE           equ    RB1    ;output to drive +/-12V into programming diode
CP_VOLTAGE      equ    RB2    ;(AN2) positive voltage measurement for Control Pulse
Vpp             equ    RB3    ;RB3 is only used during PIC programming
OSC2            equ    RB4    ;RB4 is used for the external xtal
OSC1            equ    RB5    ;RB5 is used for the external xtal
NOT_BLUE        equ    RC0    ;active low output to LED
NOT_CYAN        equ    RC1    ;active low output to LED
NOT_YELLOW      equ    RC2    ;active low output to LED
NOT_GREEN       equ    RC3    ;active low output to LED
NEG_DIODE       equ    RC4    ;negative-sense for programming diode
POS_DIODE       equ    RC5    ;positive-sense for programming diode

;*-----------------------------------------------*
;* System equates                                *
;*-----------------------------------------------*

RESET_VECTOR    equ    0x0    ;defined by PIC hardware

;*-----------------------------------------------*
;* PIC register initial values                   *
;*-----------------------------------------------*

OPTION_INIT     equ    (1 << NOT_RBWU) | (1 << NOT_RBPU) | PSA | ( 1 << PS0) | (1 << PS1) | (1 << PS2)
                       ;(page 22)
                       ;Disable wakeup on pin,
                       ;Disable weak pullups,
                       ;Timer0 source = instruction cycle clock
                       ;T0SE = don't care
                       ;Prescaler assigned to WDT (Watchdog timer)
                       ;Prescaler = 1:128

TRISB_INIT      equ    0xFC  ;(page 34)RB0 & RB1 are outputs, the rest of RB are inputs

TRISC_INIT      equ    0xF0  ;(page 34)RC0-RC3 are outputs, the rest of RC are inputs

CM1CON0_INIT    equ    (0x1 << NOT_C1OUTEN) | (0x1 << NOT_C1WU)
                       ;(page 44) Disable comparator 1

CM2CON0_INIT    equ    (0x1 << NOT_C2OUTEN) | (0x1 << NOT_C2WU)
                       ;(page 45) Disable comparator 2

VRCON_INIT      equ    0x0   ;Voltage reference off

FSR_INIT        equ    0x0   ;page 0

ADCON0_INIT     equ    (0x1 << ANS0) | (0x1 << ADCS1) | (0x1 << CHS1) | (0x1 << ADON)
                      ;(page 51-53) enable ADC with AN2 as input, clock = Fosc/4
```

```
PORTB_INIT      equ    (0x1 << CP) | (0x1 << DIODE)
                       ;both outputs high

PORTC_INIT      equ    (0x1 << NOT_GREEN) | (0x1 << NOT_YELLOW) | (0x1 << NOT_CYAN)
                       ;only power LED on
    page
;*-----------------------------------------------*
;* RAM usage                                     *
;*-----------------------------------------------*
;general purpose registers, visible in every page
R0           equ    0x0d
R1           equ    0x0e
R2           equ    0x0f

;general purpose registers that change with FSR. (We don't change FSR, however.)
CAR_COUNT    equ    0x10        ;counts sequential occurrences of no car detected
HOT_COUNT    equ    0x11        ;counts sequential occurrences of hot adapter detected
TIMER_LOW    equ    0x12        ;used to time blinking while producing CP
TIMER_HIGH   equ    0x13
R3           equ    0x14

;*-----------------------------------------------*
;* Useful constants                              *
;*-----------------------------------------------*
NO_CAR           equ    .114   ;if ADC read is higher than this, then no car is detected
CAR_READY        equ    .81    ;if ADC read is lower than this, then the car is charging

CAR_COUNT_MAX    equ    .4     ;car is gone if it seems gone this many cycles in a row
HOT_COUNT_MAX    equ    .4     ;adapter is hot if it seems hot this many cycles in a row.

DIODE_DELAY      equ    .250   ;delay after changing DIODE before reading NEG_DIODE
                               ;or POS_DIODE, units = 4 uSec
WAIT_BLINK       equ    .250   ;(units are 4 mSec) blink rate while no car
ERROR_BLINK      equ    .63    ;(units are 4 mSec) blink rate when error
READY_BLINK      equ    .4     ;(units are 256 mSec) blink rate while car present but not ready
REV_BLINK        equ    .125   ;(units are 4 mSec) blink rate for displaying firmware revision

;These are the timing specifications for the CP signal (these work best when divisibly by 4.)
CP_PERIOD        equ    .1000  ;period in uSec for CP signal
HIGH_40AMP       equ    .668   ;high time in uSec for 40-amp CP
HIGH_24AMP       equ    .400   ;high time in uSec for 24-amp CP
HIGH_16AMP       equ    .268   ;high time in uSec for 16-amp CP

    page
```

Standard Header File

```
LIST
; P16F506.INC  Standard Header File, Version 1.10    Microchip Technology, Inc.
        NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F506 microcontroller.  These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;               C:\ MPASM MYFILE.ASM /P16F506
;       2. LIST directive in the source file
;               LIST   P=16F506
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;          Revision History
;
;==========================================================================

;Rev:   Date:    Reason:

;1.00   12/13/04  Initial Release
;1.01   13/15/04  Added EC osc mode, COrrected CP on
```

```
;1.02   07/14/05  Updated Comparator names, comparator register bit names, and Oscillator fuse options
;1.03   08/26/05  Added port bit names

;==========================================================================
;
;       Verify Processor
;
;==========================================================================

        IFNDEF __16F506
           MESSG "Processor-header file mismatch.  Verify selected processor."
        ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                           EQU     H'0000'
F                           EQU     H'0001'

;----- Register Files --------------------------------------------------

INDF                        EQU     H'0000'
TMR0                        EQU     H'0001'
PCL                         EQU     H'0002'
STATUS                      EQU     H'0003'
FSR                         EQU     H'0004'
OSCCAL                      EQU     H'0005'
PORTB                       EQU     H'0006'
PORTC                       EQU     H'0007'
CM1CON0                     EQU     H'0008'
ADCON0                      EQU     H'0009'
ADRES                       EQU     H'000A'
CM2CON0                     EQU     H'000B'
VRCON                       EQU     H'000C'

;----- STATUS Bits --------------------------------------------------

RBWUF                       EQU     H'0007'
CWUF                        EQU     H'0006'
PA0                         EQU     H'0005'
NOT_TO                      EQU     H'0004'
NOT_PD                      EQU     H'0003'
Z                           EQU     H'0002'
DC                          EQU     H'0001'
C                           EQU     H'0000'

;----- OPTION Bits --------------------------------------------------

NOT_RBWU                    EQU     H'0007'
NOT_RBPU                    EQU     H'0006'
T0CS                        EQU     H'0005'
T0SE                        EQU     H'0004'
PSA                         EQU     H'0003'
PS2                         EQU     H'0002'
PS1                         EQU     H'0001'
PS0                         EQU     H'0000'

;----- OSCCAL Bits --------------------------------------------------

CAL6                        EQU     H'0007'
CAL5                        EQU     H'0006'
CAL4                        EQU     H'0005'
CAL3                        EQU     H'0004'
CAL2                        EQU     H'0003'
CAL1                        EQU     H'0002'
CAL0                        EQU     H'0001'

;----- CM1CON0 Bits --------------------------------------------------

C1OUT                       EQU     H'0007'
NOT_C1OUTEN                 EQU     H'0006'
C1POL                       EQU     H'0005'
```

```
NOT_C1T0CS                  EQU     H'0004'
C1ON                        EQU     H'0003'
C1NREF                      EQU     H'0002'
C1PREF                      EQU     H'0001'
NOT_C1WU                    EQU     H'0000'

;----- ADCON0 Bits --------------------------------------------------------

ANS1                        EQU     H'0007'
ANS0                        EQU     H'0006'
ADCS1                       EQU     H'0005'
ADCS0                       EQU     H'0004'
CHS1                        EQU     H'0003'
CHS0                        EQU     H'0002'
GO                          EQU     H'0001'
NOT_DONE                    EQU     H'0001'
ADON                        EQU     H'0000'

;----- CM2CON0 Bits -------------------------------------------------------

C2OUT                       EQU     H'0007'
NOT_C2OUTEN                 EQU     H'0006'
C2POL                       EQU     H'0005'
C2PREF2                     EQU     H'0004'
C2ON                        EQU     H'0003'
C2NREF                      EQU     H'0002'
C2PREF1                     EQU     H'0001'
NOT_C2WU                    EQU     H'0000'

;----- VRCON Bits ---------------------------------------------------------

VREN                        EQU     H'0007'
VROE                        EQU     H'0006'
VRR                         EQU     H'0005'
VR3                         EQU     H'0003'
VR2                         EQU     H'0002'
VR1                         EQU     H'0001'
VR0                         EQU     H'0000'

;----- PORTB Bits ---------------------------------------------------------
RB0                         EQU     H'0000'
RB1                         EQU     H'0001'
RB2                         EQU     H'0002'
RB3                         EQU     H'0003'
RB4                         EQU     H'0004'
RB5                         EQU     H'0005'

;----- PORTC Bits ---------------------------------------------------------
RC0                         EQU     H'0000'
RC1                         EQU     H'0001'
RC2                         EQU     H'0002'
RC3                         EQU     H'0003'
RC4                         EQU     H'0004'
RC5                         EQU     H'0005'

;==========================================================================
;
;       RAM Definition
;
;==========================================================================

        __MAXRAM H'7F'

;==========================================================================
;
;       Configuration Bits
;
;==========================================================================

_IOSCFS_ON                  EQU     H'0FFF'
_IOSCFS_OFF                 EQU     H'0FBF'
_MCLRE_ON                   EQU     H'0FFF'
_MCLRE_OFF                  EQU     H'0FDF'
_CP_ON                      EQU     H'0FEF'
_CP_OFF                     EQU     H'0FFF'
```

```
_WDT_ON                         EQU     H'0FFF'
_WDT_OFF                        EQU     H'0FF7'
_LP_OSC                         EQU     H'0FF8'
_XT_OSC                         EQU     H'0FF9'
_HS_OSC                         EQU     H'0FFA'
_EC_OSC                         EQU     H'0FFB'
_IntRC_OSC_RB4EN                EQU     H'0FFC'
_IntRC_OSC_CLKOUTEN             EQU     H'0FFD'
_ExtRC_OSC_RB4EN                EQU     H'0FFE'
_ExtRC_OSC_CLKOUTEN             EQU     H'0FFF'

        LIST
```

Leave a Comment

No Comments Yet »

No comments yet.

RSS feed for comments on this post. TrackBack URI

Leave a comment

Name (required)

Mail (will not be published) (required)

Website

Submit Comment

Notify me of follow-up comments via email.

- ## Recent Comments
  - David Kosowsky on The Future Starts Now.
  - Gabe on The Future Starts Now.
  - Rob D. on The Future Starts Now.
  - Steve S. on The Future Starts Now.
  - Gabe on The Future Starts Now.
- ## Pages
  - About
  - Downloads
- ## a

  Select Category
- ## Blogroll
  - Tesla fan club

- The Oil Drum
- WordPress.com

Blog Stats

- 421,655 hits