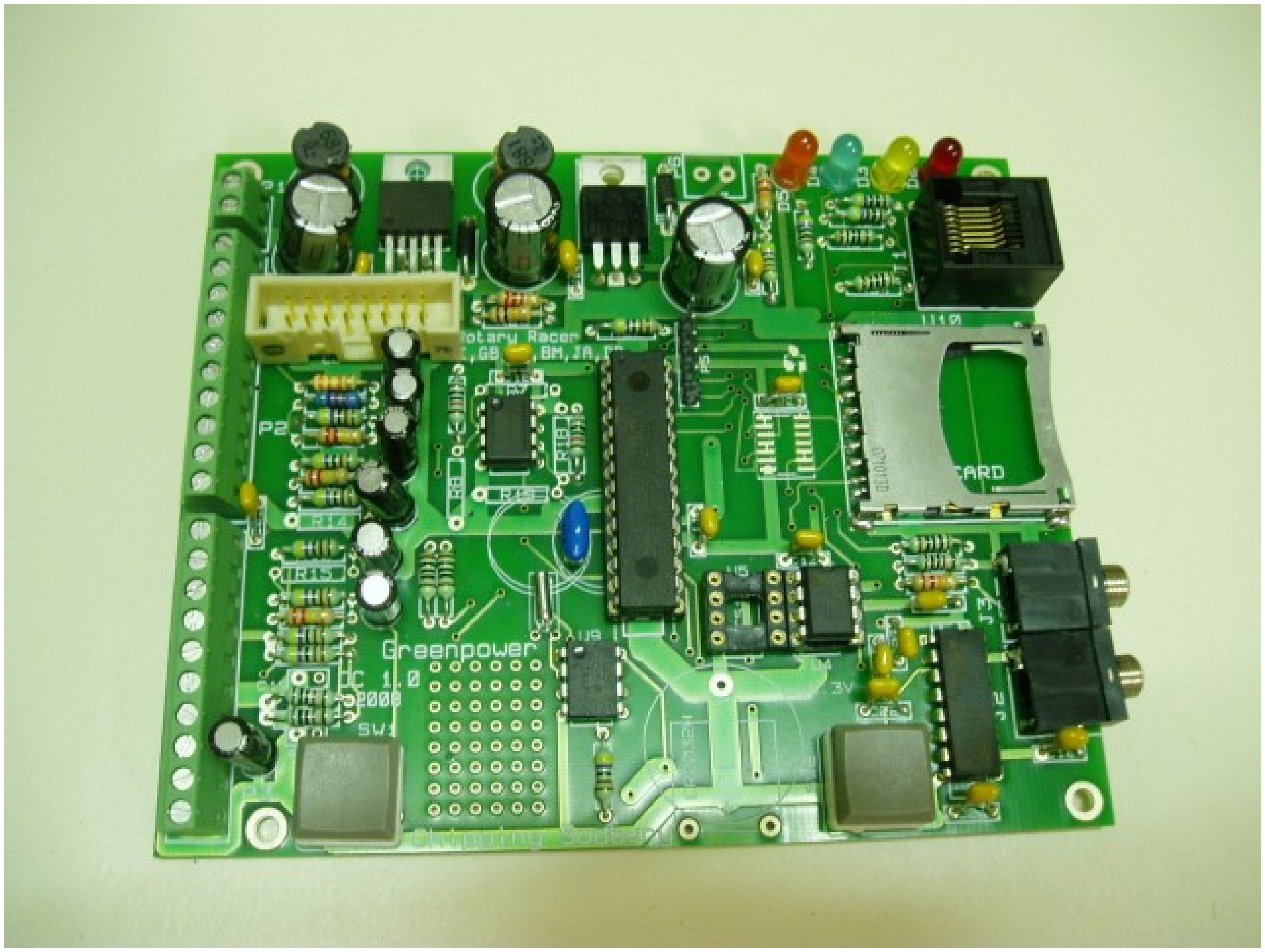
Greenpower Car Computer

Chipping Sodbury School – Rotary Racer

CC1.0 – 2008-09-08 - T.Barnaby

Introduction

This document gives a description of the Greenpower car computer developed for the Rotary Racer. The Rotary Racer is an electric racing car developed by the pupils and parents of the Greenpower Rotary Racer team at Chipping Sodbury secondary school.



The board was designed after playing with a prototype system using PicAxe computer modules on the Rotary Racer Greenpower car. The circuit design was done by a parent, the PCB was laid out by a parent with the pupils laying out some of the tracks. The PCB was sent out to a PCB production house for manufacture. The components were soldered in place by the pupils with help by the parents. Test LED code was done by some of the pupils.

Overview

The main purpose of this unit is to provide instrumentation and feedback for the driver and pit crew on the cars operation and performance. This is useful for developing the car and making sure it is running correctly during the race. It also has the ability to control the motor speed, cooling fan and other items. It has been designed to be simple, but as flexible as possible so that it can be used in different car designs in different ways.

In a typical use it can perform the following functions:

- Calculate and display the cars current and average speed
- Calculate and display the distance traveled
- Measure and display the battery voltage
- Measure and display motor current
- Measure and display the motors temperature
- Log all of the data to non-volatile memory for later recall.
- 3 LED's for driver information.
- High efficiency switched mode and secondary linear regulated power supply: Runs off car's 24Volt power supply (7 – 24 Volt operation)
- Battery backed up real-time clock for date and time.
- Low power (Approx 15mA at 15Volts 0.21 Watts)
- Can be programmed in PicAxe Basic or 'C'.
- Serial RS232 communication to Laptop computer for data download.

It can also perform the following functions:

- Calculate and display the battery charge in % for each battery set
- Control motor speed if motor speed controller is present. This can limit the current to the motor and can automatically set the maximum motor speed based on the charge present in the batteries and the race duration.
- Control the speed of a cooling fan for the motor to save power on this device.
- Control gearbox if gearbox change servo is present.
- To store the data into a removable SDCARD that can store up to a years worth of data sampled every second.
- Extra RS-232 port for live radio telemetry back to the pit crew.
- Use a simple 2-line LCD panel for display with 4 user input switches.
- I2C serial bus for extensions.

Some Features of the Car Computer:

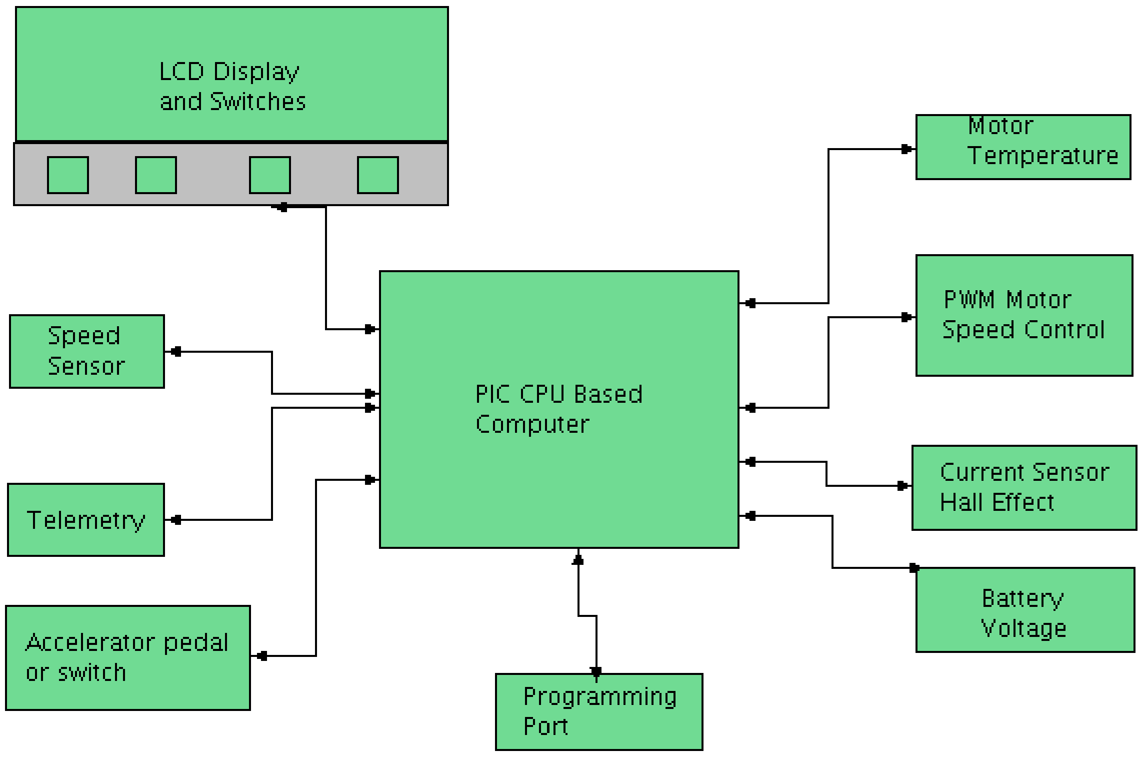
- Easily built by pupils.
- Can be programmed using free Basic or 'C' software development tools under Linux or Microsoft Windows.
- Programs in PicAxe Basic or 'C' can be downloaded to the board using a serial cable. No need for special programming hardware.
- Flexible hardware can be modified for different purposes and there is a small patch area for extra components.
- High efficiency switched mode power supply built in with power line filters to reduce voltage spike interference from the cars electronics.
- Low power requirements 0.21Watts.
- Real-time clock with battery backup.
- Watchdog timer to automatically reset computer if it fails.
- The processor has 25 pins available for digital I/O or analogue inputs. Up to 5 analogue input channels can be connected. The PIC processor has a 10bit ADC.
- Operational Amplifier and RC filtering for analogue signal conditioning.
- Simple screw terminals to connect power and input/output lines.
- Reset push button and user push button on board.

- Four LED's on-board for development or other purposes.
- Daughter board connector for Motor Speed Controller or other purposes. This has power, I2C bus and two PWM control signals.
- Display interface RJ45 connector. This has power, I2C bus and I/O lines available for LED and/or switch interfaces.
- Up to 64Kbytes of on board non-volatile FLASH memory.
- Interface to SD card for over 8GBytes of FLASH memory support.
- Two serial port jacks using PicAxe pin-out. One is a true RS232 interface, the other is a PicAxe bit bashed serial interface.
- On-board MOSFET output for high current drive for replay, fan or other purposes.

The design of the unit has been made as flexible as possible to allow for different uses in different cars. Its PCB has been designed for easy assembly by pupils with both through hole components and a few surface mount components.

It is based on a simple PIC Microprocessor that can be easily programmed by pupils and parents to perform different tasks. The program is stored in non-volatile memory within the PIC micro-processor. The processor has 25 pins available for digital I/O or analogue inputs. Up to 5 analogue input channels can be connected.

Typical Usage



As an example of its use in the Rotary Racer, the computer can measure the following car parameters.

Time	Every Second	A battery backed up clock keeps the date and time to the nearest second.
Speed	Every Second	A reed relay/magnet combination or hall-effect sensor pulses an input on each wheel revolution. The time in ms for a number of pulses is used to determine the cars speed.
Distance	Every Second	The input from the speed sensor is used to determine the distance travelled.
Battery Voltage	Every Second	The Battery voltage is measured using a 10bit ADC in the computer
Motor Current	Every 0.1 Seconds	The battery current is measured by an external hall effect device and read every 0.1 seconds using the computers 10bit ADC.
Battery pack number	Every 1 second	The battery pack plug/socket has a pin to indicate which battery pack is installed, 0 or 1.
Throttle	Every 0.1 seconds	The throttle position is read every 0.1 seconds using the computers 10bit ADC.
Motor Temperature	Every second	The motors temperature is measured every second using the computers 10bit ADC.

In the Rotary Racer the Car Computer is connected to a Motor Speed controller of our own design. The Car Computer generates the PWM (Pulse Width Modulated) to control the motor. The Motor Speed Controller simply amplifies this signal to drive the motor.

Car Computer Board

The car computer board is implemented on a 120mm x 100mm dual layer PCB. It is based around a PIC microprocessor. Two main variants are catered for: a PIC 16F886 programmed as a PICAXE 28X1 for Basic programming and a PIC18F2420 or PIC 18F2520 which can be used when programming in the 'C' language. Other PIC processor variants may be used in the board.

The board has the following main components on board:

Component	Type	Usage
Processor	PIC 16F886 or PIC18F2520	Main microprocessor PIC16F886.pdf and PIC18F2420.pdf PICAXE 28X1 is a 16F886
Clock	DS1307	Real time clock chip with battery backup. http://www.rev-ed.co.uk/docs/DS1307.pdf
EEPROM I2C	24LC256	Memory storage device, up to two can be installed in the two 8pin sockets. Used for data logging. http://www.rev-ed.co.uk/docs/24lc256.pdf
RAM I2C		Optional: Temporary memory storage device can be installed in one of the two 8pin sockets.

Rs232 Interface 1	PICAXE compatible	Programing port for PICAXE programing, or can be used as a simple RS232 port.
RS232 Interface 2	MAX232	RS232 serial communications interface to computer or wireless telemetry. Also used for 'C' programming port.
Regulator	LM2575	Optional: Switched mode regulator for efficient 5 - 12V power supply from (12 to 24 Volts)
Regulator	LM705	Optional: Linear 5V regulator for accurate, low-noise power supply suitable for high quality ADC.
SDCARD		Optional: SDCARD interface for mass storage of data. Up to 8 Gigabyte cards supported via SPI interface.
OPAMP	LM6132	Two low noise analogue operational amplifiers on two inputs for signal conditioning.
MOSFET	IRFR024N	Optional: High power MOSFET switch for PWM fan control or relay switching etc.
LED's	Red,Yellow,Green and Orange	There are 4 LED's for status and debugging
Switches	Push Button	There is a reset switch and a user switch
PROGPORT		The board has the conventional 5 pin PIC programing connection for low level chip programming using a standard PIC programmer.

The board has the following connections:

Connection	Usage
Power	7 - 24Volt 's can be used as the power supply assuming at least one of the regulators is installed.
Data Inputs/Outputs	Up to 13 digital inputs. 8 of these can be analogue inputs and two have operational amplifiers. The board has been designed to provide: ● 6 inputs having optional RC filters and attenuators that can be used to reduce noise. 5 of these can be analogue. ● 1 switch Input ● 4 Digital outputs for LED's (1 high current) ● 2 Digital PWM outputs
Digital PWM Outputs	The are two Digital PWM outputs for Motor speed and/or Fan control etc
Digital High current Outputs	There is a high current MOSFET driven digital output.
I2C	A two wire I2C interface is provided for display control and/or other purposes. There are a number of chips available that support the I2C bus standard.
RS232	Two RS232 three wire interfaces are provided for control or

	programming. One of these is PICAXE compatible the other is a full RS232 voltage (+-9V) port using the PIC's hardware RS232 interface and has interrupt capability.

Car Computer Build options

As previously stated the Car Computer board has been designed to be as flexible as possible. The following lists the main build options provided:

- On-board power regulation. The board has an optional, high efficiency, switched mode regulator that can lower the 24 Volt power supply to provide the necessary 5Volt supply. There is an alternative linear 5 volt regulator as well. It is also possible to use the two regulators in tandem with the switch mode regulator generating, say, 8 Volts and the linear regulator reducing this to the 5 Volt supply line. This method improves the power supply efficiency and can make the ADC measurements more accurate.
- Build with PICAXE with Basic language support or PIC18F2520 for 'C' support. This requires plugging in the appropriate processor chip.
- The SDCARD socket can be built onto the board for more data storage. This may only work with the PIC18F2520 programmed with the 'C' language.
- An extra RS232 port can be built onto the board for communications for a telemetry system etc.
- A high current MOSFET transistor can be built on the card for driving a high current device such as a fan or relay.

Software Development

The software can be developed using the PICAXE Basic language system or in the 'C' language.

The PICAXE basic language system is used by students. It is fairly simple to learn and is well documented. It uses the commercial, but free, PICAXE development system. It does lack a few features though, such as real interrupts which can make it difficult to implement some processing and the chips do not have a great deal of program space. The Basic language is also a little awkward to use.

The 'C' language is very flexible, high performance, but less well documented. It uses the free open source SDCC 'C' compiler.

We have done all of the software development under the Linux operating system although it is also possible to use Microsoft Windows.

There are simple examples of code using both programming languages available.

As the PIC microprocessor is a small low level device, it is normally necessary to read and understand the chips data sheet in order to program it successively. The data sheet is available on the Rotary Racers Wiki website under CarComputer.

Software Development using PicAxe Basic

The PicAxe basic development software can be downloaded from: http://www.rev-ed.co.uk

This software works under Microsoft Windows. In may run under Linux using the latest Wine Microsoft Windows emulator.

The PicAxe chips come with a small RS232 boot loader ready programmed inside. This allows the Picaxe to be connected to a PC or Laptop via the RS232 port to reprogram the PicAxe with a new program. The Picaxe company can provide a suitable 9pin RS232 to stereo jack download lead or you can make one up. The Car Conputers J3 socket should be used for PicAxe Basic programming.

There is some simple Picaxe example code on the Rotary Racer Wiki website at http://portal.beam.ltd.uk/greenpower/carComputer

Software Development using 'C'

There are a number of 'C' compilers available for the PIC processor. The one we have used is the Open Source, free, SDCC compiler. Although this is not very well optimised it has suited our needs. It can be downloaded from: http://sdcc.sourceforge.net/. It is available for Linux as well as Microsoft Windows. We have used it under Linux Fedora 8 and have a simple to install RPM package for this system available.

We use the free “Tiny” RS232 boot loader program for this environment. Information on this is at: http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm. We can provide the PIC chips with this boot loader already programmed into the chip. It is then possible to simply program the chip by downloading a new program over the serial port. We have developed a simple command line program, named, **picProgrammer** to program the chip in this way under Linux. Under MSWindows the TinyBld program can be used. The Car Computers J2 socket should be used for programming in 'C'.

The procedure for developing using 'C' is as follows:

1. Edit the source code using your favorite text file editor. We used Nedit. Alternatively an IDE (Integrated Development Environment or GUI based tool), can be used. The open source piklab tool is available for Linux and MSWindows.
2. Compile the source code file or set of files into PIC executable instruction file in HEX format using the SDCC compiler. We use the “make” utility to do this. Simply run the command “make” in the source directory to build the code. The piklab IDE can also do this.
3. Use the picProgrammer program to write the HEX file to the PIC. To do this run the program with the file name of the hex file created as an argument, then press the reset button on the Car Computer. The bootloader is active for about 1 second after the reset button has been pressed. The piklab IDE can also do this.
4. The Bootloader program in the PIC will download the new code and start running it.

Notes on the 'C' Language

The PIC 'C' compiler is not a full 'C' compiler and has a number of extensions for handling the simple 8bit PIC processor family. The PIC processor is essentially an 8bit microprocessor, that is, it natively works with 8bit numbers either signed numbers in the range: -128 to 127 or unsigned numbers in the range: 0 to 255. The PIC has some support for 16bit numbers (-32768 to 32767 or 0 to 65535) and the 'C' compiler can generate the necessary set of 8bit instructions to implement the rest of the 16bit and indeed 32bit arithmetic operations although the code size increases and the speed goes down.

The 'C' compiler can also generate the code to perform 32bit floating point calculations, but this results in large and slow code and so is generally best avoided.

So it is best to work in, low bit count, integer arithmetic. To achieve a reasonable accuracy it is useful to use fixed point arithmetic with 16bit values. For example to store 20.12 Volts you would store and manipulate the number 2012. ie the number is scaled by 100 to effectively give to decimal points of accuracy. When working with low bit count, integer arithmetic you have to be very careful of over and underflows in calculations. So for each calculation you write you should consider what the minimum and maximum values are and if necessary pre-scale them before performing the calculation. Note that the mathematical divide operation is the most difficult for a processor to do and should be avoided where possible. Sometimes a multiplication with fixed point numbers. Or appropriate scaling can be performed on others. Also the processor is good at dividing by 2,4,8 etc.

The example sources contains a basic Car Data logging program named carLog1.c. This will log the data from all of the standard sources and provide the ability for a PC to download the data. This uses the piclib.c routines to interface to the various special PIC processor hardware modules such as ADC and I2C devices. It also manages a microsecond accurate timer to general timing use. This program forms a good basis for further development as it interfaces to most of the PIC hardware and implements the main "hard stuff". Note that this program uses interrupts. This can be a difficult concept to get the hang off, but it allows the PIC to perform many tasks simultaneously.

The example sources directory also contains some simple test examples to play with including LED drive code. Obviously it is worth looking at some teaching aids for the 'C' language, either in book form or on the Web.

Downloading The Data Log

We have developed a simple command line program, **carmon**, and a **GUI** program called **carmonGui** that is able to download the data log from the car computer. The **carmonGui** program can display the data in live graphs and save the data to an ASCII file. At the moment these programs only run under the Linux operating system, but they should be readily portable to Microsoft Windows. With the standard example code, the Car Computer responds to a simple set of one character commands sent through the main RS232 interface to perform this. The command set is fairly primitive at the moment, but could be readily extended.

The RS232 commands available on the Rotary Racers Car Computer are:

Command	Description
R	Resets the computer. This sets the charge calculation back to 100% and all settable parameters to default.
Z	Factory Reset. This does all of the standard reset plus it resets the log data pointers back to the start of memory.
m	Toggles the display mode
c	Send live data on RS232
r	Stop sending data and return to normal logging mode
t <s,m,h,day,month,year>	Sets the date and time in the real time clock device
i	Returns information. The number of directory entries, and the current directory pointer and data pointers in the log memory.
d <blockAddress,number>	Dumps the given number of logging data blocks starting from the given block number out on the RS232.

Driver Display and Control

The car computer provides a simple interface to a separate driver display/control unit. This is implemented through a standard RJ45 connector as used for computer networking although the connections are not compatible with computer networking standards.. This has 5Volt power lines, a serial I2C interface as well as 4 digital I/O lines to drive LED's and get input from switches.

This allows a simple panel with just LED's and push buttons to be used, or more extensive LCD panels. There is a circuit diagram for the Rotary Racers display panel on the Car Computer website. A description of the Rotary Racer display panel follows:

There is a small display panel for the driver to see information and control the computer. The display panel has a 2 line by 16 character LCD display for information display, 3 LED's (Red, Yellow and Green). And 4 push button switches under the LCD.

The car computer can be in one of a number of modes. The LCD will display different information depending on the mode of the computer. In normal race mode it will display the following:

- Car Speed
- Distance Traveled
- Motor Current
- Battery number
- Battery Charge

The push buttons do the following:

Mode	Button 1	Button 2	Button 3	Button4
Race	Turbo			Display mode
PitCrew	Turbo	Down	Up	Display mode

If all buttons are pressed for 1 second the CarComputer is reset to start of race conditions.

The LED's display the following information:

Mode	Red	Yellow	Green
Race	Battery Low or other warning		Turbo on

Pit Crew Display and Control

During operation the car computer will continuously store the main parameters measured into a FLASH EEPROM or SDCARD. The period of this is programmable from 1 per second to 1 per minute. There is room for 1024 sets of 32Byte measurement sets to be stored in a single 24LC256 EEPROM. With two such EEPROMS you can store the entire data for a 4 hour race sampled once every 8 seconds. A large SDCARD can store over a years worth of data sampled once per second. A laptop computer can be connected to the car computers serial port to retrieve the data.

It is also possible to get the data live through the serial port. This could be used with a wireless radio link to allow the pit-crew to view the data live from the car during a race. The wireless link could be bi-directional to allow the pit crew to control car parameters and send messages to the driver.

The actual data available is listed in the Data Storage section.

Car Computer Connections

This is a table of the PIC microprocessors connections and their use in the Rotary racers Greenpower car computer. Pins are numbered from the top left of the board.

Board Pin	Related CPU Pin	CPU Use	Usage
P1			**This is the two way power connector on the board. If a motor speed controller board is connected via P4, the power may be obtained from P4 and this connector may not be used**
P1.1			+12 to +24Volts power
P1.2			0V Power Ground
P2			**Analogue inputs and PWM outputs or digital I/O. Note that if P4 is used for current sensing and the battery voltage is sensed via the power input, there may not be any use for this connector.**
P2.1			+5Volt output
P2.2	2	ANA0/RA0	Battery Voltage input. This can be internally connected to p1.1 using a small jumper, JP1, on the board
P2.3			0V Ground
P2.4			+5Volt output
P2.5	3	ANA1/RA1	Battery Current. Voltage from Hall effect. Also available on P4.
P2.6			0V Ground
P2.7	12	RC1	PWM Motor speed control out. Also goes out on P4
P2.8			0V Ground
P2.9	13	RC2	PWM Fan speed control out or Timer Feedback pin. Also goes out on P4.
P2.10			0V Ground
P3			**Analogue inputs and digital I/O**
P3.1			+5Volt output
P3.2	4	ANA2/RA2	Temperature Input voltage.
P3.3			0V Ground
P3.4			+5Volt output
P3.5	5	ANA3/RA3	Throttle Input voltage.
P3.6			0V Ground
P3.7			+5Volt output

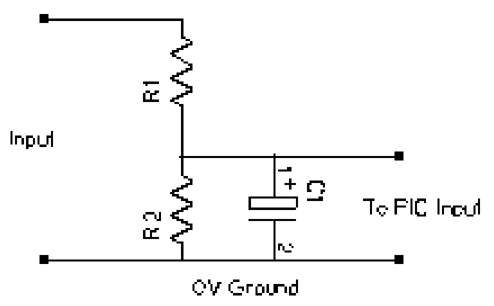
P3.8	21	RB0/INT/ANA12	Car speed in (magnetic reed relay) connected between +5 and this line.
P3.9			+5Volt
P3.10	11	RC0	Battery Number switch input. Connected between +5 and this line.
P4			**This is the IDC connector on the board. It is designed to connect to a motor speed controller or hall effect current sensor. It can be used to connect to other extension boards.**
P4.1			0V
P4.2	2		+12 to +24Volts power (Battery Voltage Input)
P4.3			0V
P4.4			+8 Volts Output (From first on-board regulator)
P4.5			0V
P4.6			+5 Volts Output
P4.7			0V
P4.8	3	ANA1/RA1	Battery Current Input from Hall effect
P4.9			0V
P4.10	12	RC1	PWM Motor speed control out
P4.11			0V
P4.12	13	RC2	PWM Fan speed control out (Timer feedback pin ??)
P4.13			0V
P4.14	15		I2C SDA
P4.15			0V
P4.16	14		I2C SDC
P5			**PIC programming port. This is used for low level programming of a PIC processor and matches the standard connector that is used for this purpose. It can also be used for debugging code.**
P6			**This provides a high current digital switch output to drive a relay or similar. A maximum of about 0.5 amp can be drawn using the on-board power supply. If the +ve voltage is obtained from elsewhere and the P6.2 connector is use as a switch to ground then up to 16 Amps can be switched. Note that level of current this may cause power glitches on the board however ... It is**

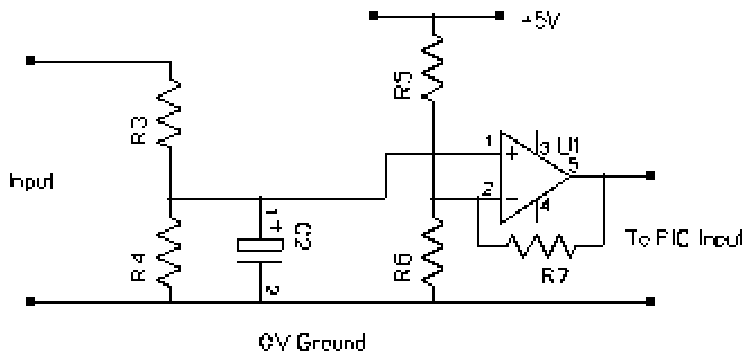
			also possible to not fit the MOSFET and wire across from the PIC's I/O pin giving another digital or analogue I/O line.
P6.1			+8V output of first switch mode regulator. This could be re-wired to the input volage. Also the first regulator could be modified for 12Volt operation.
P6.2	25	RB4/ANA11	Via IRFR024N MOSFET 16 Amp 55Volt
SW1	11	RC0	On board switch 1
J1			**This is designed for connecting a display panel with switches. However it could be used for other purposes.**
J1.1			0V
J1.2	15		I2C SDA
J1.3	26	RB5/ANA12	GREEN LED (1K resistor in series to 0V)
J1.4	28	RB7	RED LED (1K resistor in series to 0V)
J1.5	27	RB6	YELLOW LED (1K resistor in series to 0V)
J1.6	24	RB3/ANA9	Switch Input
J1.7	14		I2C SDC
J1.8			+5V
J2			**Serial port for PicAxe programmer. This is a “bit bashed” RS232 port for programming the PIC when used in the PicAxe environment. It has level reduction resistors on its input so that it can be directly connected to an RS232 port on a computer. It could also be used for two general purpose I/O lines. Note its pin-out matches the PicAxe serial port cables and is a bit strange.**
J2.1			TX
J2.2			RX
J2.3			Ground
J3			**Serial port for 'C'programming and general usage. This is a full RS232 voltage RS232 port connected to the PIC processors RS232 hardware. It could also be used for two general purpose I/O lines. Note its pin-out matches the PicAxe serial port cables and is a bit strange.**
J3.1			TX
J3.2			RX

J3.3			Ground

Connecting Things

Most of the Car Computers inputs are fed through a simple passive signal conditioning circuit. This consists of a simple resistor voltage divider, R1 and R2, followed by a capacitor C1 to filler the incoming signal and remove spikes. By default, the inputs have: R1 = 4K7, R2 = 220K and C1 = 10uF. This gives a near 0 – 5V input range with a high degree of low pass filtering. The voltage input has suitable resistors to handle a 0 to 34 Volt range. If it is desired to use these as outputs the capacitor should be removed and the value of the input resistor lowered to about 100 Ohms giving some degree of protection to the processors I/O pin.





Two of the inputs, designed for the current and temperature inputs, have the same basic input circuit but have a following operational amplifier that can scale and offset the voltage as required. R5, R6 and R6 can be adjusted for voltage offset and gain. By default the operational amplifier is set for a gain of 1 and no offset.

The input terminal blocks have 0V and 5V connections available to allow two and three wire sensors to be easily connected. See the Circuit Diagram for more details.

- **Voltage:** Assuming the car computer is connected to the batteries, then it is configured using JP1, to directly measure the battery voltage without any additional wiring. The resistor dividers and the software scaling has been calibrated to read the right voltage level. If desired the software calibration values can be changed to tweek the reading. The voltage can be read from 0 to about 34 Volts. If more accuracy around the 24Volts is desired, a zener diode of say 16 Volts could be put in series with the input attenuation resistor and the software scaling adjusted to suit.
- **Current:** We have used a hall effect sensor to measure the current. The one we have used is RS part number: 532-9126. This uses 5Volts. We have put this component on the Motor Speed board where it is connected in line with the main +ve supply from the battery. It would also be possible to use one of the “wire through” hall effect devices. The first regulator on the board could be set to supply the necessary voltage for this if it is not 5Volts. The current input is fed through a resistor divider chain and then into a filter capacitor. Following this is an operational amplifier in unity gain mode. This can be changed to provide the appropriate gain with perhaps a voltage offset as desired. The software scales the resulting ADC output to give a correct reading. If a different hall effect device is used from the one we used then this scaling will need to be calibrated. It would also be possible to use a small value resistor to sense the current and use a differential operational amplifier to measure the voltage and send this into the current

input.

- **Temperature**: The temperature input also has an operational amplifier. This allows a simple thermistor or perhaps diode temperature sensor to be used. This would be connected between 5V and the input line. The resistor attenuator and operational amplifiers scaling components would be set as required and the software scaling adjusted to calibrate the result.
- **Speed:** The speed input is a simple digital input. A reed, hall effect or optical switch can be connected between this and the 5V line. It expects one pulse per wheel rotation. The software has the diameter of the wheel configured into it. The software counts and accurately times the period between pulses. It then calculates the distance and the speed of the car.
- **Throttle**: The throttle is normally connected to a simple resistive potentiometer. We have use a linear slider type. The ends of this are connected to +5V and 0V with the sliders center tap connected to the input.
- **Battery Number:** The battery input is a simple digital input. We have battery connectors that have additional pins that allow us to define the two sets of batteries.
- **Display and Switches:** The display and switches are connected to the RJ45 connector. This provides +5V power, and I2C serial interface and 4 I/O lines. We use the I2C interace to connect directly to an 2x16 character LCD display. The 4 I/O lines are multiplexed between driving the 3 LED's and reading the state of the 4 push button switches. The push button switch state is read using an analogue input so that the state of all switches can be detected simultaneously.
- **PWM Outputs:** These are available on the screw terminals and on the daughter board connector P4. We use one of them on the Motor Speed controller to drive the power MOSFETS through a MOSFET driver chip to control the motors speed. The second PWM we use to vary the speed of the motors cooling fan via a MOSFET transistor.
- **RS232 Interfaces:** We use the main RS232 port for programming the PIC and for downloading the data log. We don't use the secondary RS232 interface at the moment although this is used as the main interface when programming a PicAxe processor.

Data Packets

The standard software stores data in the non-volatile EEPROM or SDCARD. This can then be transmitted on the serial port in the form of packets of data. The format of these data packets is as follows:

Location	Type	Description
0	16bit	Magic value: 0x4141 (“AA”)
2	8bit	Seconds (BCD)
3	8bit	Minutes (BCD)
4	8bit	Hours (BCD)
5	8bit	Day (BCD)
6	8bit	Month (BCD)
7	8bit	Year (BCD)
8	16bit	Voltage (Volts*100)
10	16bit	Current (Amps*100)

12	16bit	Charge1 (Percent*10)
14	16bit	Charge2 (Percent*10)
16	16bit	Speed (Km/Hour * 100)
18	16bit	Distance (Km * 100)
20	16bit	Throttle (0 - 1023)
22	8bit	Temperature
23	8bit	Battery number (0, 1)
24	16bit	MotorSpeed as set by Car Computer (0 - 1024)
26		
27		
28		
29		
30		
31	8bit	Checksum (Sum of all packet bytes except this one)

More Information

There is more information on the Car Computers website at:
http://portal.beam.ltd.uk/greenpower/carComputer