Home made comprehensive engine control, EFIS and unlimited engine data acquisition.

Micro computer are getting cheaper and cheaper smaller and smaller and faster and faster. One can buy a Maximite as little as $60 with 2 gig of solid state memory in the form of an SD card so it will work in an aircraft at high latitude unlike a hard drive.

The idea here is to use some of this low cost high tech computer power to cut the time and cost of programming engine control computers in the BASIC language. High level languages are not as efficient as asm but that can be compensated for with a fast enough computer.

I started looking around for a computer board that could be programmed in BASIC that did not have any of the drawback of Windows computers and dynamic memory. It needed static memory.

I picked up the PC board from CicuitGizmo in Idaho.

It was designed by Geof in Perth Australia

He answered a couple of questions I had.

You can buy it here.

http://www.circuitgizmos.com/products/cgcolormax2/cgcolormax2.shtml



It has a very powerful BASIC which appears to be a superset of QuickBasic. Geof wrote the basic interpreter himself and designed the board so he is the guy that also answers all the questions.

Obviously a very small company and very easy and quick to give support. A huge advantage for you. I had the free Integrated Development Environment installed and running in less than five minutes. It is as easy as QB. I wrote and compiled this program in another minute. It is a time delay routine. I sent this jpg screen shot off to Geof with a question about the error. This is one 1/100th of the time I needed to get the Motorola 68xx IDE up and running.

In the BASIC computer language there are two kinds of parsers. A parser reads the plain language text file you write and converts the words into codes. The first kind generates intermediate codes or tokens. These tokens are then sent to a interpreter in ROM or firmware that converts them into bits and bytes the computer can

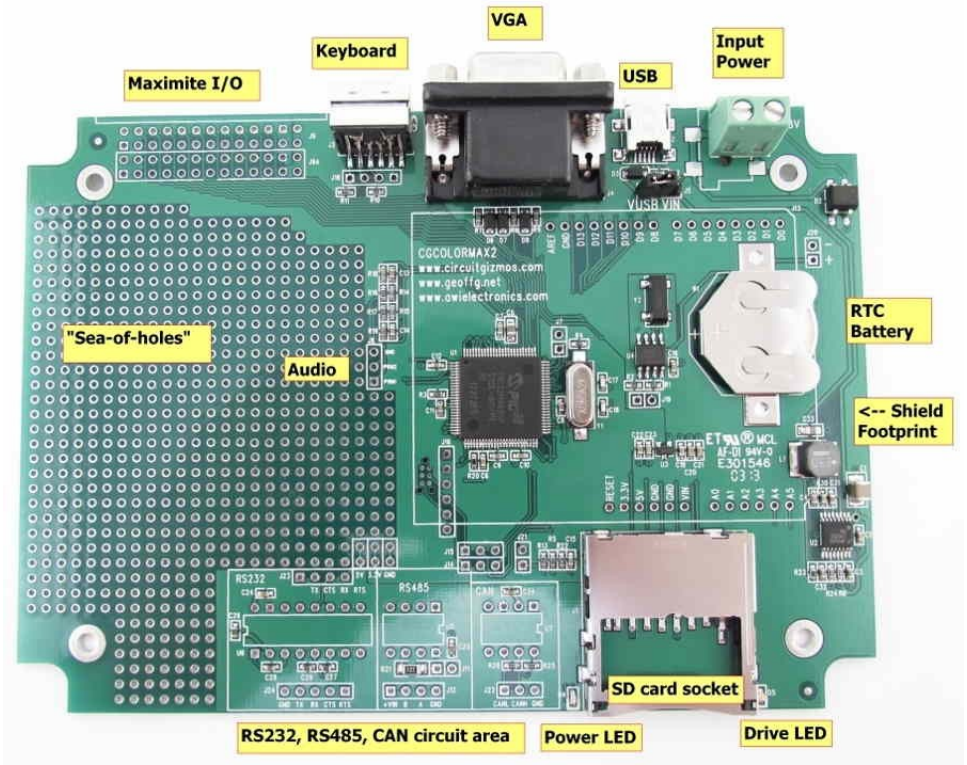
understand called machine code. In that case the BASIC is called a Interpreter.

The second kind of BASIC computer language is called a compiler that generates machine code directly. Generally speaking compilers generate programs that run faster and use less memory space.

Overkill for an EFI/EFIS system.

Computers are very dumb machines and they must be told in detail exactly what to do and how to do it.

For example; to command a computer to walk in ASM or C you must tell it to move its left leg forward. Lower left foot. Shift weight over left foot. Move right leg forward. Lower right foot. Shift weight over right foot. Etc., etc., etc.. In a Basic interpreter you merely tell it to walk. Now if you want to tell it to modify the WAY it walks you need to see the assembler listing so you can change the BASIC walk command. Some times this becomes necessary due to differences in the hard ware you are trying to control. The best way to understand this is download the free Basic compiler and start using it.



It comes with a simple DOS OS for file I/O to the SD card so it can be used in real time. It is at least as fast as the average run of the mill single chip computers

found in most EFI systems. Cost was $60 with the processor. Some other brands are also available at about the same price. The video is built and one USB port is built in. This board looked very promising so I bought one and experimented with it.

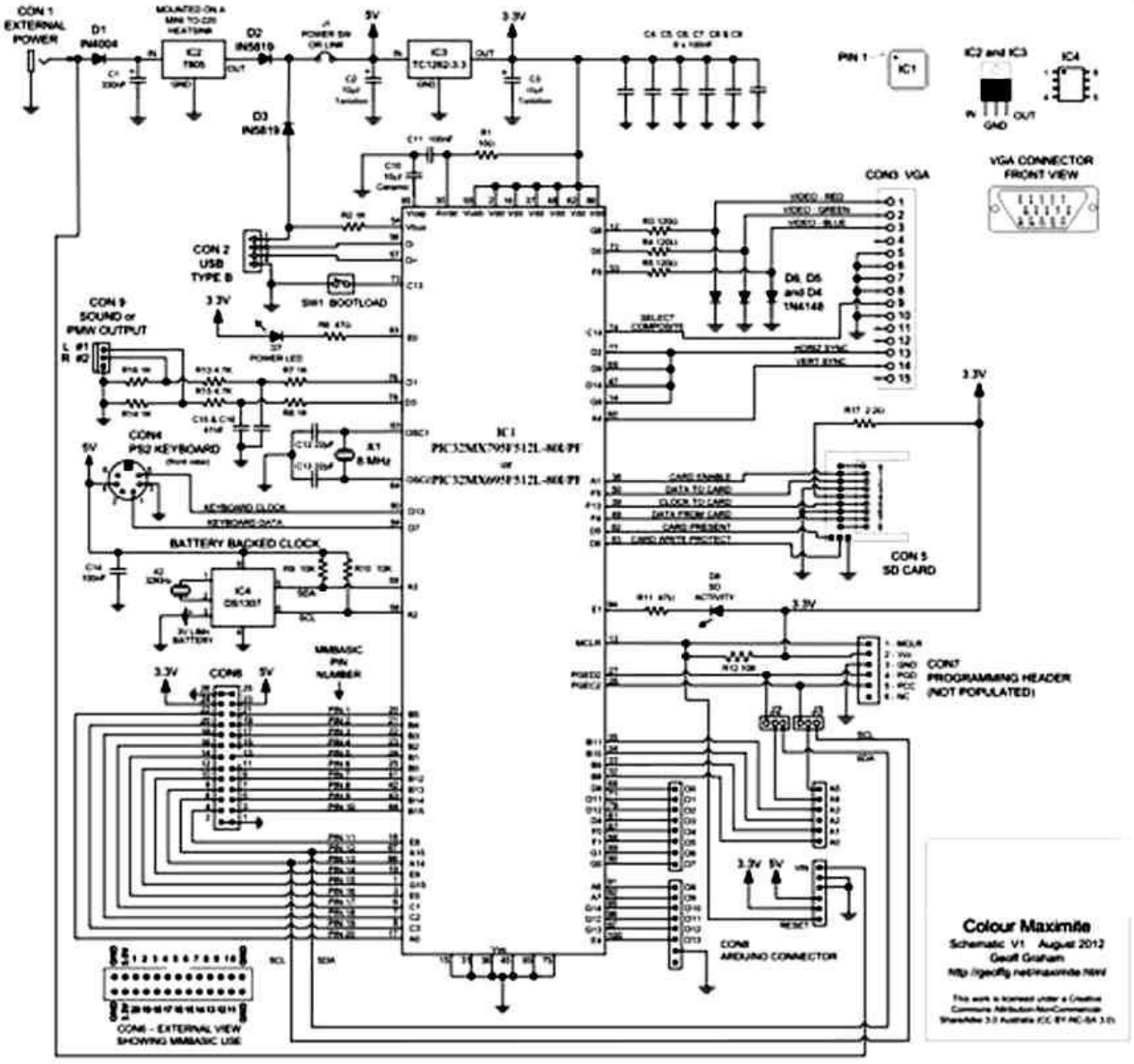
No need for a fragile hard drive.

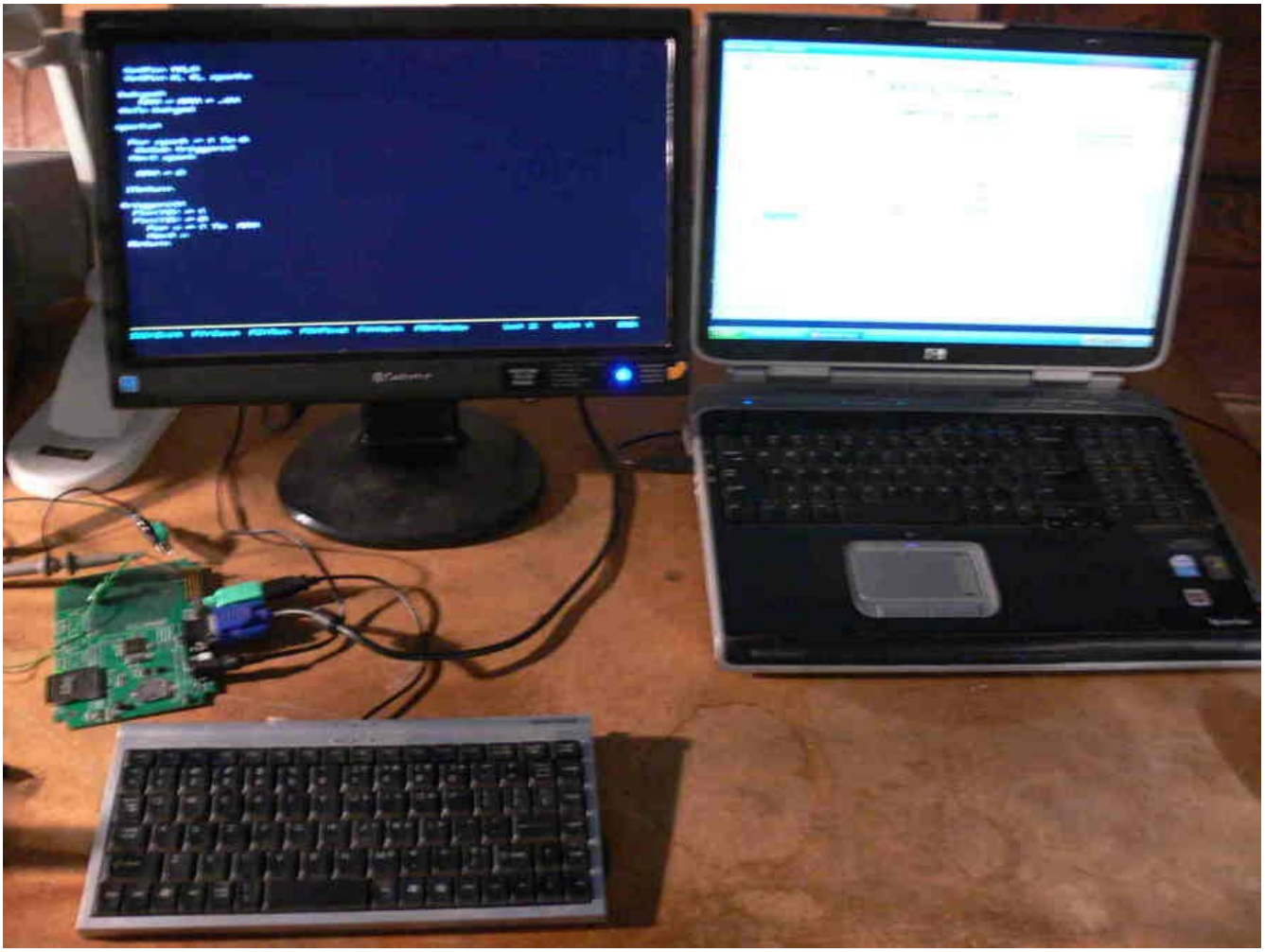
Why DOS and not Windows or Linux? DOS is far more real time than Windows and Linux ever could be. There is too much stuff going on in the back ground to support the Windows and Linux multi-tasking GUI.

This board has built in high speed 10 bit analog to digital converters.

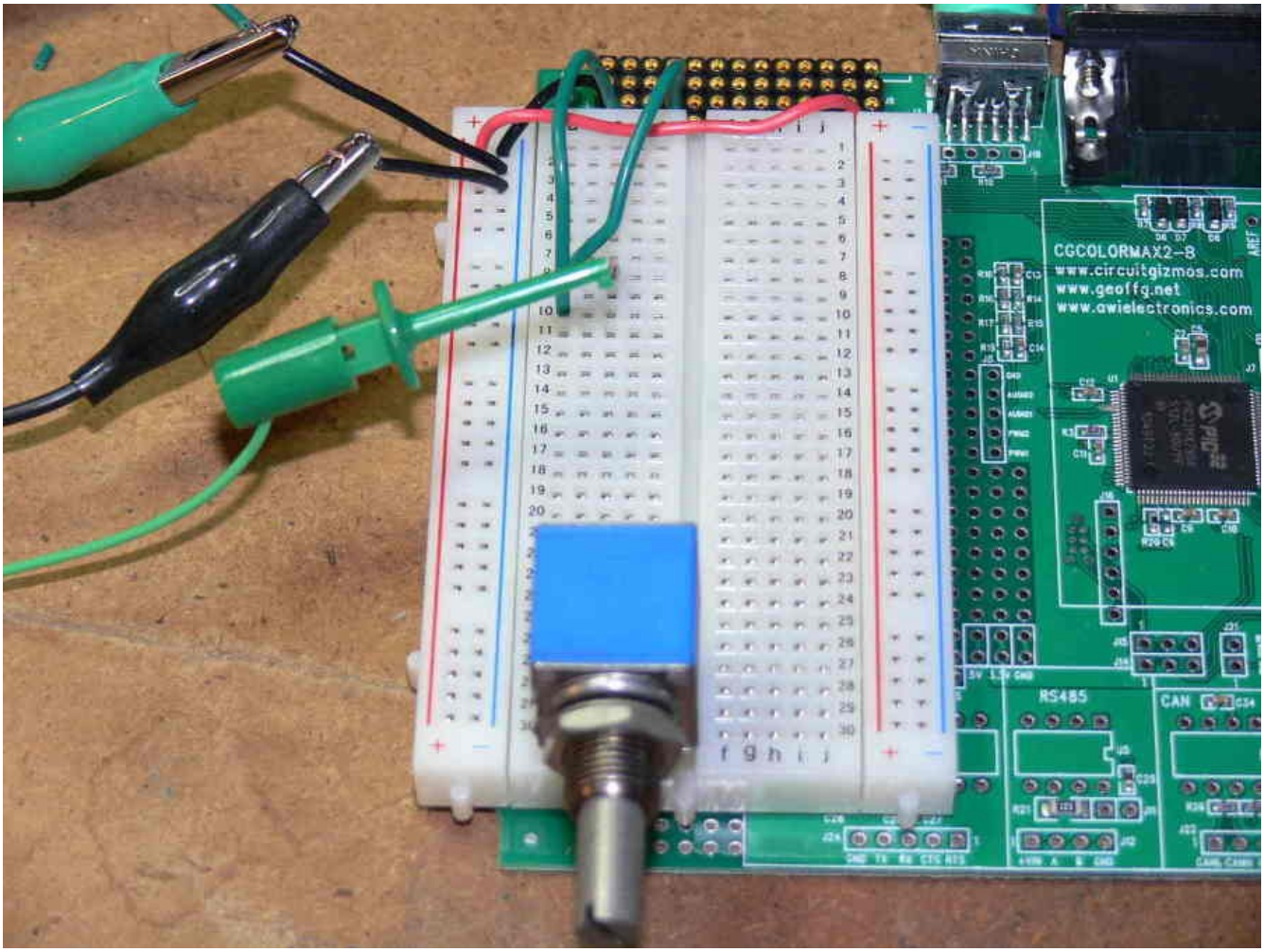
This board is far easier to program than a Megasquirt . All sorts of other things can be done with the full screen of course in terms of displaying data. For example the screen could read out the real time value of the pulse width being used and the RPM. It could also total the pulse widths and read out the GPH. It could also run my DOS-QB based moving map software as I only need that data out of the GPS once per second.

The C programming language is unnecessarily cryptic IMHO and I refuse to waste time learning it. PC computers are very powerful these days and you should be able to program them in easily understood plain language and let the computer do all the work converting the program to bits and bytes. C places too much tedious housing keeping work on the programmer. IMHO one might as well program in ASM as one has better control over the exact time it takes to complete any instruction. C was the one thing I did not like about the Megasquirt or the Arduino approach.

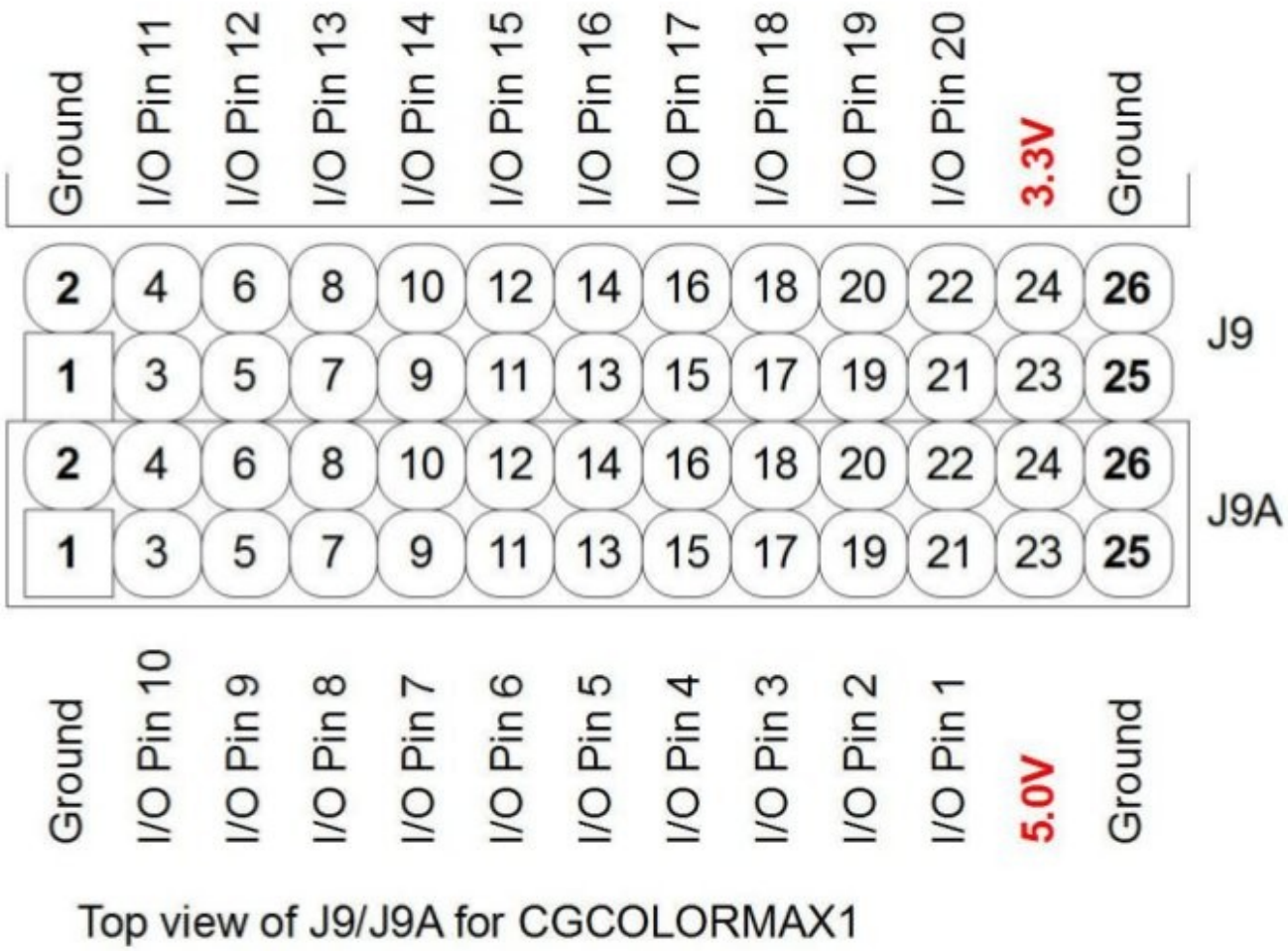

CON 1
EXTERNAL
POWER
D1
IN4004
D2
IN5819
D3
IN5819
IC2
7805
IC3
TC1262-3.3
5V
3.3V
PIN 1
IC1
IC2 and IC3
IC4
GND
OUT
VGA CONNECTOR
FRONT VIEW
CON3 VGA
VIDEO RED
VIDEO GREEN
VIDEO BLUE
D6, D5
and D4
1N4148
HORIZ SYNC
VERT SYNC
CON 2
USB
TYPE B
SW1 BOOTLOAD
CON 9
SOUND or
PWM OUTPUT
L #1
R #2
POWER LED
CON4
PS2 KEYBOARD
KEYBOARD CLOCK
KEYBOARD DATA
X1
8 MHz
IC1
PIC32MX795F512L-80I/PT
BATTERY BACKED CLOCK
IC4
DS1307
SDA
SCL
CARD ENABLE
DATA TO CARD
CLOCK TO CARD
DATA FROM CARD
CARD PRESENT
CARD WRITE PROTECT
CON 5
SD CARD
D8
SD
ACTIVITY
MCLR
CON7
PROGRAMMING HEADER
(NOT POPULATED)
MMBASIC
PIN
NUMBER
CON6
CON8
ARDUINO CONNECTOR
RESET
CON6 - EXTERNAL VIEW
SHOWING MMBASIC USE
Colour Maximite
Schematic V1 August 2012
Geoff Graham
http://geoffg.net/maximite.html



The note book on the right of the MM screen holds the BASIC documentation in pdf for so you can easily scroll up and down in the docs to refresh your memory of the commands.

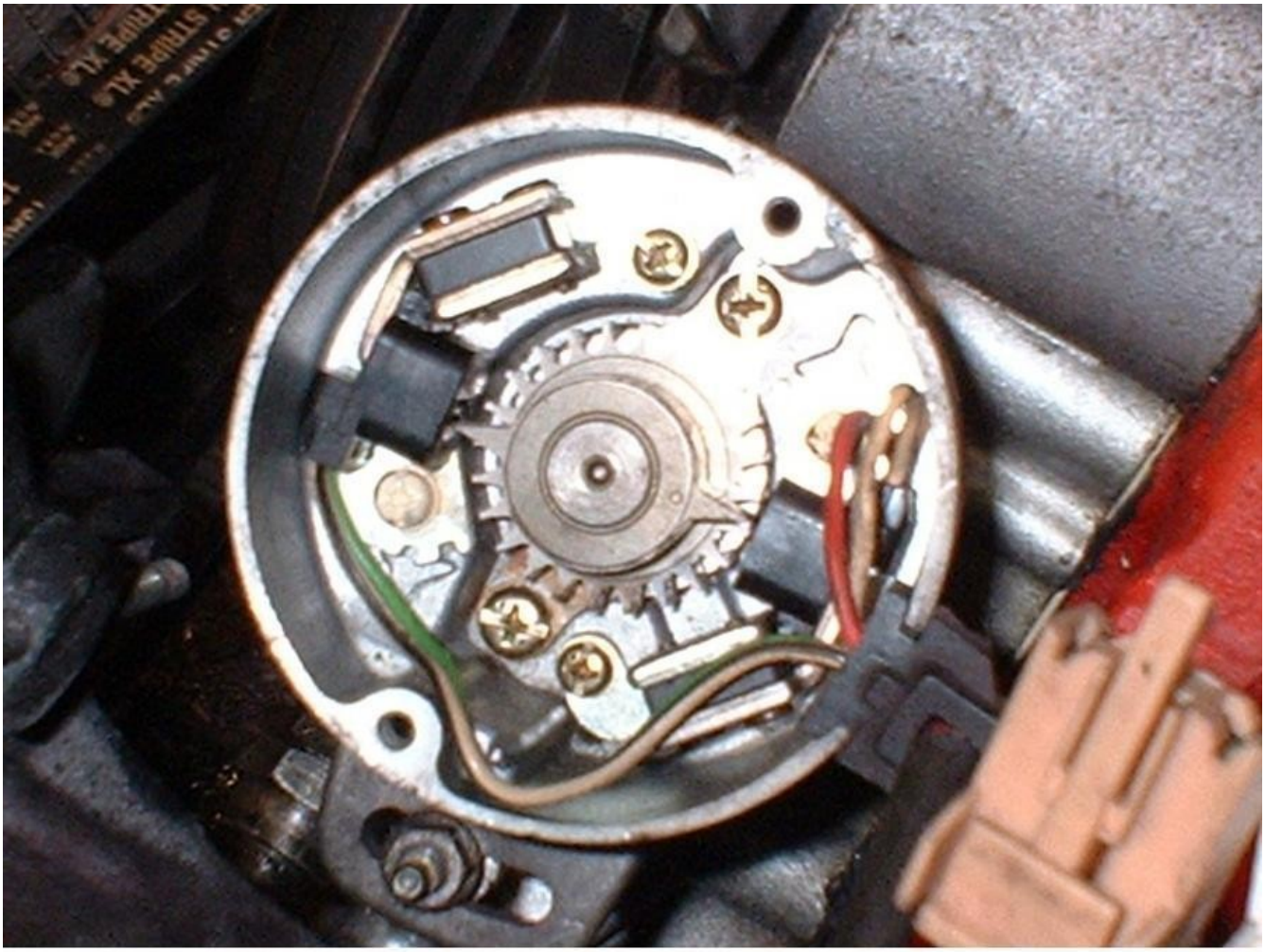


Proto plug in Board for development of EFI/EFIS.



Top view of J9/J9A for CGCOLORMAX1

Here are the pin outs for the MM. J9 and J9A are identical.

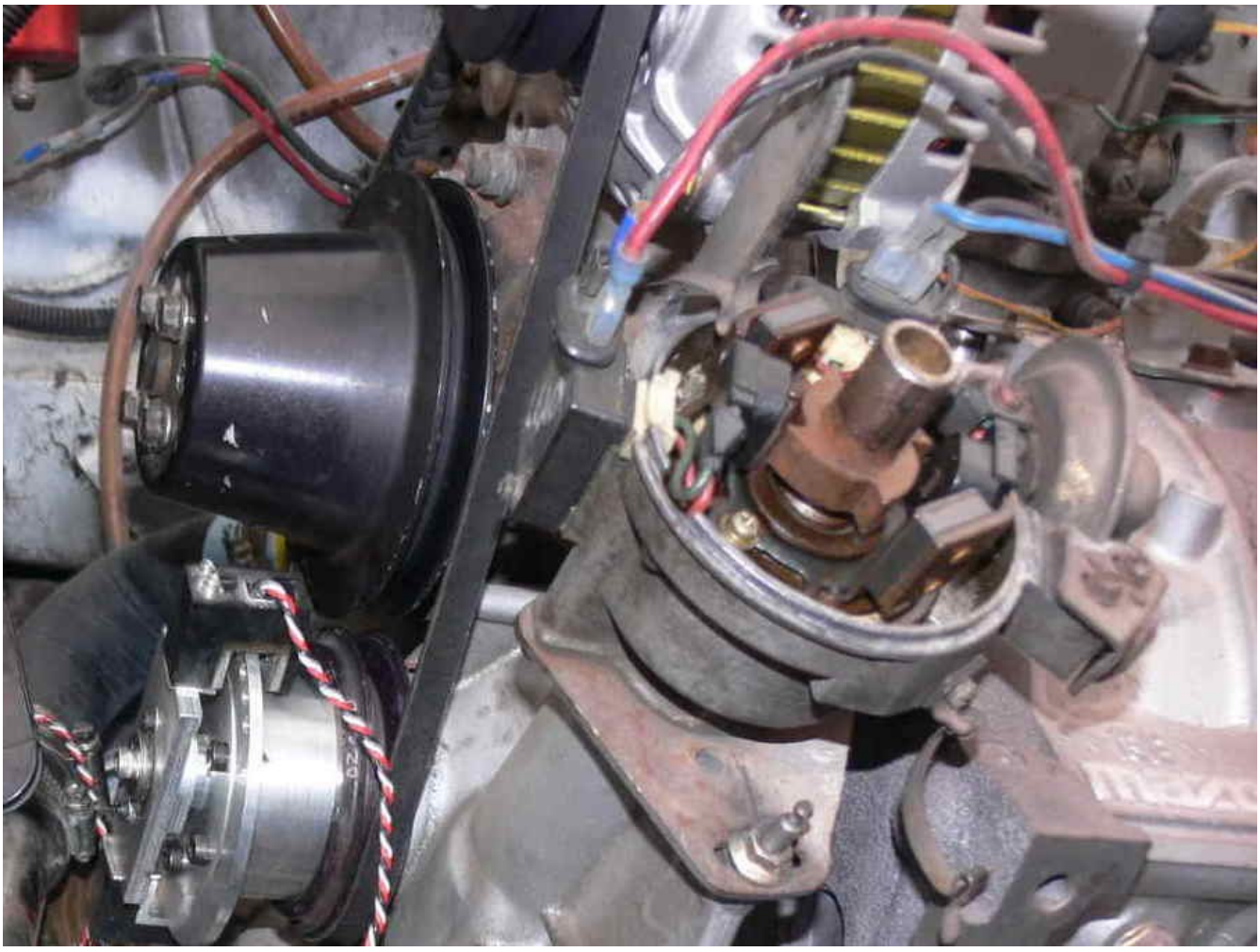


2nd gen Mazda Rx7 Crank Angle Sensor II

This is a driver's side fender view of the Crank Angle Sensor for the 2nd gen Rx7 and Mazda Cosmo 13B engines. The home and trigger pickups are located approximately 2:30 and 8:30 o'clock respectively. This is also presuming that the engine is rotated from TDC to 5*ATDC (yellow mark on factory crankshaft pulley).

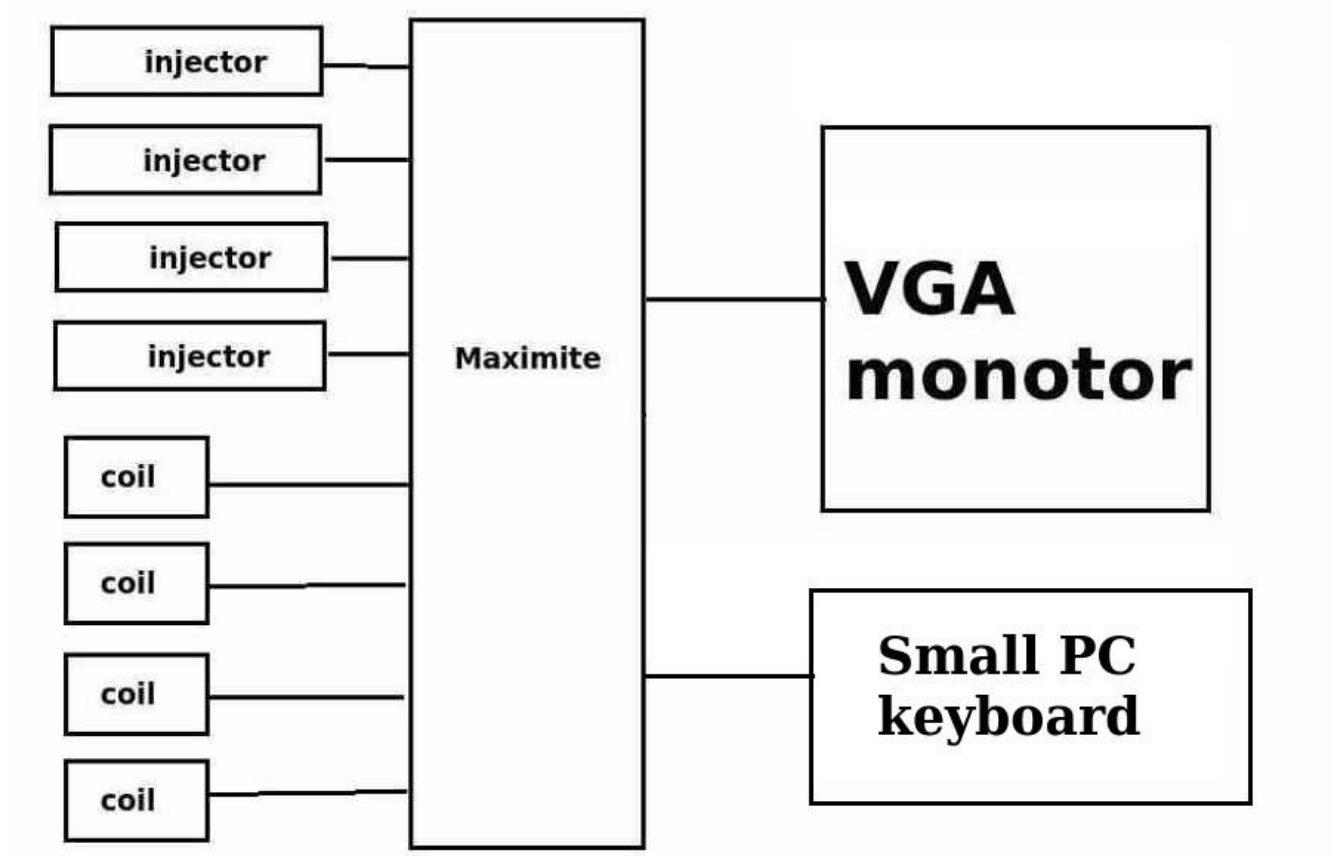
CAS_____	__Purpose
Red_____	___Main Trigger +ve
Green___	____Home Trigger +ve
White___	___Main Trigger -ve
White/Black_Blue	___Home Trigger -ve

We will only use the top sensor to provide RPM in to the EFI program and the EFIS out to the display monitor. This is only one of of several schemes used by Mazda to control the EFI and the ignition.



This is the racing CAS. It has different signal characteristics that can be handled by jumpers on the EFI printed circuit board.

I set up a BASIC program that would test the speed of the EFI program running under Maximite. Frankly I was stunned. The Mximite executed 40 million steps in one second Namely a rotary engine electronic fuel injection.


injector
injector
injector
injector
coil
coil
coil
coil
Maximite
VGA
monotor
Small PC
keyboard

```
Mode 4
 'Puts larger characters on the display.

' this character means that it is a comment and it is ignored by the MM.
For n = 11 To 20

SetPin n, 9   'set pins 11 to 20 as open col 5 v output

Next n



  Open "data.txt" For output As #1 'data file
 on the SD card.


  SetPin 8, 6, 10 'pin 8 low to high int gosub line 10

  SetPin 10, 1 'manifold press on pin 10

  SetTick 1000,20
   'set the clock interrupt to one second.


BakGnd:
   'Batch mode EFI
MAF1 = Pin(10)  'A to D on pin 10. 'Very simple.
MAF2 = MAF1 * 10

MAF3 = Int(MAF2)

PWM 20,MAF3,MAF3 'pulse width on primary injectors
 simple.
""""""""""""""""""and secondary  20 times a second

GoTo BakGnd
```

```
10 RPM:
  'count every time the CAS triggers
RPM = RPM + 1

 IReturn



20 RPMRoutine:
 'calculate the RPM and print it on the display.
RPM=RPM*30

Cls
              'clear screen
Locate 30,30
   'column 30 line 30
Print "RPM = ";RPM

RPM = 0
      'zero the RPM counter
Locate 30,60   'col 30 line 40

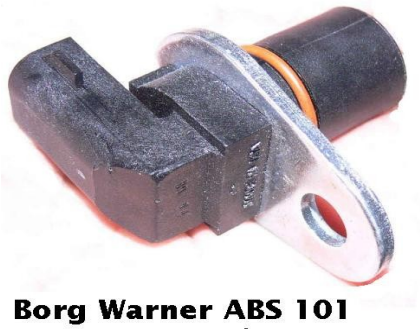
Print "Mass Air Flow = ";MAF 'EFIS function

IReturn
```

This system is so modular however that someone could implement CAN if they wished. Both the ignition system, injector system and RPM counter will work with any computer that has parallel port capability.

A timer interrupts the program from time to time and up dates a real time clock maintained in memory. To make a long story short I decide to not rely on the BASIC for accurate timing events like determining the RPM and firing the ignition.

Low cost ABS mag sensor that every car in the world with ABS has four of.



Borg Warner ABS 101 sensor about $25 at Pep Boys.

The other device that will be off loaded is the ignition timing. Here is a picture of a bench test set up.

For this project you won't need to learn to program but you will need to use the compiler. You can just download what I and others write posted on here and compile it into your rotary EFI system through a USB connection. The same thing as having the EFI computer in your car re-flashed. You won't need to ship your computer somewhere to have somebody else do it. If you do wish to program it is not as hard as it looks. Even though there are over 50 BASIC instructions typical programs use less than a dozen popular instructions.

The ZX-24s Atmega1280N $90 proto board has a 64K byte RAM on board that could store a 16 byte string of GPS time stamped location data. It is a simple program as you read in the GPS data from just about any aircraft panel GPS receiver through the RS232 port and store it in the net book every 30 seconds or so. If my arithmetic is correct that is hours of data so you can tell where you were and how fast you went.

Of course you could also send the data out the USB port and let a Netbook store it on a hard disk or solid state memory.

There is a USB interface so one can send MAP and fuel table display data to a cheap net book in the cockpit. There is also a RS232 device so real time GPS data can be input through a NEMA 0183 standard aircraft interface. Http://en.wikipedia.org/wiki/NMEA_0183 The processors have as many as 100 I/O pins that make designing a printed circuit EFI controller board much simpler.

I have no intension of doing a FADEC as there are situations that require pilot adjustments. Even though there will be auto lean the pilot will be able to over ride that.

Zbasic comes with a wealth of free source code to do different things. One of the reasons it is free is because BASIC code is so quick and easy learn and write people don’t value it highly so they tend to give it away. Our EFI system will follow that reasoning so you can understand exactly how it works, what it does and when it does it.

Hard ware interrupts rapidly gets to be a real can of worms. That is why I am doing the critical real time stuff like determining RPM with the flywheel teeth and firing the ignition with simple external hardware like two Hall Effect sensors for a two rotor and three for a 3 rotor. The ignition can be advanced with an old fashion push pull cable like the Ford Model T. Easy to understand and fix.

One of the jobs it must to do is read a mass airflow sensor and calculate an injector pulse width.
Here is the function you use to get the digital value of an analog voltage appearing on pins .

'SETPIN pin, cfg [,target]

'Will configure the external I/O pin according to cfg :

'The Color Maximite has 20 I/O pins.

'function PIN() will also return the output value.

'The starting line number of the interrupt routine is specified in the third 'parameter

' target . This mode also configures the pin As a digital input so the value of the 'pin

'can always be retrieved using the function PIN().

manpre = Pin(10) 'A to D on pin 10. may need to be scaled

Resource Usage

Only analog port pins may be used to perform an analog-to-digital conversion.

One of the drawbacks to mass air flow sensors is the voltage is more or less proportional to the cube of the mass air flow while engines need fuel directly proportional to the mass air flow. This is handled in most modern EFI's by the look up table. As far as I can tell the BASIC exponent function can get the cube root... so in theory no look up table is required. It is similar to a manifold pressure based EFI system were manifold pressure is more or less proportional to the mass air flow. Unfortunately therein lie the problems we see in tuning existing EFI systems as manifold pressure can be affected by a lot of things. Bottom line is we will be using the industry standard mass air flow sensor.

Here is the BASIC exponent function.

To translate from computer science language to Dick and Jane language .

In QB the form is (xxxx)^.5 is the square root of xxxx.
(xxxx)^.333 is the cube root of xxxx.
.5 and .3333 are the exponents.

So the exponent can be selected to linearize the mass air flow sensor voltage out put. I intend to use MAF as the basic means of determining the correct fuel flow adjusted by EGT based auto tune. This is going to make tuning simple if not entirely automatic if it works. To completely understand what is going on with these functions it is best to experiment with these functions and use debug.print to send the results to your desktop computer.

This is the companion instruction to set the pulse width of the injector.

INSTRUMENT PANEL DISPLAY

I am thinking of using this low cost 3 inch wide by 2.5 inches high 80 character display. It is 4 lines by 20 characters wide. That should be plenty of info for the pilot. The over all size of the box and panel cut out will be about 5 inches wide and 4 inches high with switches, netbook USB connector and a leaning knob below the display.

These are some of the display options that are easy to implement shown in the attached jpg.

The trade off is the amount of info you get compared to the ease of reading. The larger the display the better but it takes up panel space. I am leaning towards a 4 line 20 character wide display so we have more feedback info. We can get that 3 inches wide or six inches wide.

The display? NITS? sunlight readable? I am a proponent of OLED for sunlight viewing. Perhaps it is my age, but my LCD instruments in sunlight are worthless

even at full contrast.

other microcontroller) using 7, 3 or 2 I/O lines. Example code is included containing the necessary interface code plus a collection of useful higher level routines for managing the LCD and displaying characters. Although the example code is written for a 2x16 LCD (two rows, sixteen columns), it can be easily adapted to displays with other geometries.

Since we have 16 channels of A to D it would be real cheap to add a compression checker. All it takes is another FreeScale $10 pressure sensor. That means you turn the engine over with the starter, connect a hose to the pressure sensor and a phony spark plug pressure tap. The computer samples the pressure 1000 times a second so you get a plot of the compression pressure. A very low cost option.

This system will be ideal for your rotary engine as it is entirely modular. 1,2,3 or 4 rotors it does not care. Just add CD ignition modules and one of these RPM counters and you will have a complete engine control computer with all engine parameters sent out for real time display and recorded on a cheap net book computer. The same net book computer can be used to program the system. The software is programmed in BASIC so any child can learn to program it. It will be entirely open source so anybody can upload and download programs on here.