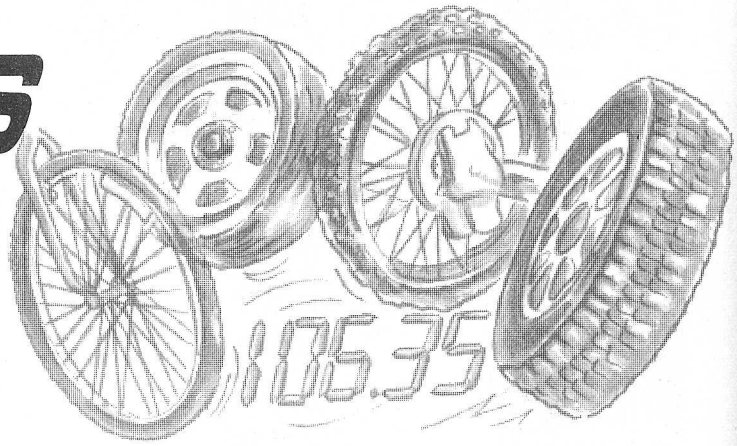
PIC-AGORAS WHEELIE METER



JOHN BECKER

Part 1

Flux-gate sensing and PIC microcontrolling combine to bring you the (nearly) ultimate in wheeled-distance measurement and display.

INSPIRED by the recently introduced FGM-3 magnetic sensor, and the author's desire to update an existing 6502-based bicycle computer, this design offers the opportunity to measure the distance covered by almost any wheeled vehicle, from a golf trolley to a bicycle and beyond. Wheel diameters up to about three metres can be catered for (. . . want to electronify your Penny-Farthing or Traction Engine?) at speeds of up to about 100 m.p.h.

The controlling heart of the design is the now-familiar PIC16C84 microcontroller, and its readout is on a readily available intelligent liquid crystal display (l.c.d.). Whilst the software is complex (though that needn't concern you), the design is simple to build and install.

Wheel rotation sensing is done using a magnetic field sensor in conjunction with a small magnet. The sensor is attached to the frame of the bike or other vehicle; the magnet can be secured to any convenient part of the wheel.

A further option is open to adventurous experimenters – you could, perhaps, be inspired to try modifing the design for use with sail-boards and the like!

As to the Title? There's a tale to tell, but we'll keep it short, at the end!

MAGNETIC SENSOR

Any type of magnetic sensor could have been the starting point for the design. All have their merits; all can have their problems. When contemplating the design of PIC-Agoras, though, it seemed an interesting idea to use the new FGM-3 magnetic field sensor from Speake & Co.

Its response has proved to be excellent and obviously has many other applications in which it can be used. Its only problem in this application was the need to convert the modulations of its frequency output to decoded single pulses.

The FGM-3 device is a very high sensitivity magnetic field sensor operating in the ±0·5 Oersted (±50 microtesla) range. Its operation is based on the flux-gate principle in which the strength of a magnetic field determines the frequency response of one or more coils.

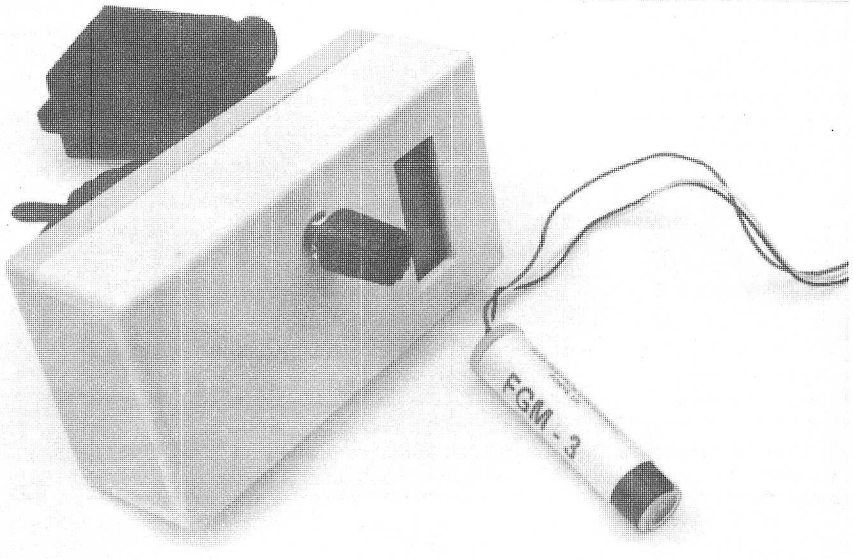
Flux-gates are commonly found in the miniature electronic compasses which are now in widespread use amongst the small-boat (and large) community. Go to any yachting chandler and you will probably see many examples.

In the past, flux-gates as ready-made individual components have not been available to the hobbyist electronic market. Regrettably, the precision with which the coils need to be wound does not allow them to be easily constructed in the average workshop. Certainly not in the author's, and he's tried! Thus it was with considerable interest that the FGM-3 was received.

SELF-CONTAINED

This device requires no adjustment on the part of the user. It is totally self-contained. The induction coil and all the frequency-generating electronics are enclosed in a tube about 65mm long by 16mm in diameter. The tube is transparent and looks like glass, though scraping it reveals that it is a plastic material. Its strength has not been put to the test (!) but it is probably wise to treat it with respect.

Three wires are the only connections that need to be made, two for power at 5V (7V absolute maximum), and one for the signal output. The output signal is a robust 5V rectangular pulse train whose frequency varies inversely with the magnetic field strength detected (giving a pulse period which is directly proportional to the field strength). This signal is ideally



Completed Wheelie Meter with FGM-3 magnetic field sensor.

suited to interfacing to a computer, or microcontroller, or other digital electronic circuit.

Typically, the frequency swing is from 50kHz to 120kHz for a field strength ranging between ±0·5 Oersted. You do not need to be concerned about field strengths in this application, however. Suffice to say that the frequency swing caused by moving a variety of small magnets past the sensor at several tens of millimetres distance is very detectable.

The sensitivity of the FGM-3 is such that even the earth's magnetic field can be detected, making the sensor usable in electronic compass applications. Be aware, though, that actually *designing* an electronic compass involves good knowledge of some quite high mathematics and the use of a microcontroller or similar which can cope with all the code involved!

The sensor's temperature sensitivity is excellent, at 0·003%/°C at 25°C.

The magnet chosen for use with the sensor was a small disc type removed from a plastic "fridge magnet" that typically holds kitchen notes to the side of a domestic fridge.

SENSOR INTERFACE CIRCUIT

First thoughts may suggest that since the FGM-3 outputs a well-shaped 5V pulse train, the changes in frequency when a magnet passes close by could be readily analysed by a PIC microcontroller. In principle, the PIC *can* be programmed to analyse them. The drawback is that it would need to spend most of its time analysing and not have much time to process the results.

There is *a lot* of processing to be done, as will be seen later. Consequently, an electronic interface is needed to do a bit of real-time pre-processing. We are not interested in the *actual* frequency of the pulse train. It is the *change* in frequency when the magnet passes that is of interest. All we need to know is whether or not the magnet is close or distant, a simple binary logic situation; Logic 1, the magnet is near; Logic 0, the magnet is distant, or vice-versa.

This conversion from frequency change to logic level could be done in several ways. Here it is done using a phase locked loop (p.l.l.) chip, the familiar CMOS 4046 device. The details in Fig. 1 show the frequency-to-logic conversion circuit diagram, the Sensor circuit.

The FGM-3 magnetic field sensor is shown as component X1, and the p.l.l. is IC1. The output from the sensor is fed into IC1's first phase comparator input, PCA IN at pin 14. The second phase comparator input is PCB IN at pin 3, which is coupled to the chip's voltage controlled oscillator output, VCO OUT at pin 4.

The basic oscillator frequency of the VCO is determined by the values of capacitor C1 and the joint action of resistors R2 and R3, plus preset potentiometer VR1. Variation in the voltage level on IC1 pin 9, VCO IN, causes the VCO output frequency to change accordingly.

Within IC1, circuitry compares the difference in phase between the two signals

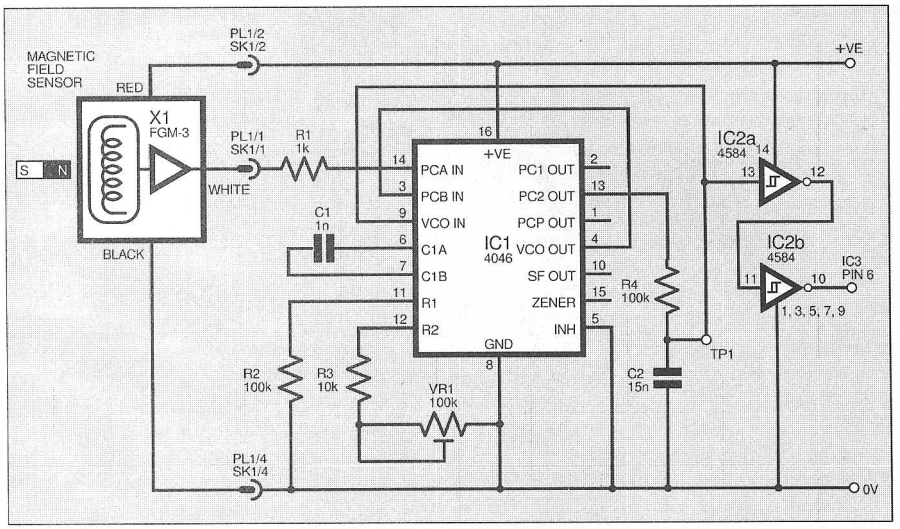


Fig. 1. Circuit diagram for the Sensor Interface frequency-to-logic section.

on the phase comparator inputs. Any difference in the phase causes a change in the voltage output at PC2 OUT, pin 13. Between them, resistor R4 and capacitor C2 smooth this output voltage and feed it back to the VCO IN input, pin 9. IC1 then does its best to change its VCO frequency and phase so that it equals that coming in from the sensor X1.

In this application, the VCO frequency range is set so that the sensor frequency and the VCO frequency never do need to match. Instead, the output voltage at PC2 OUT (pin 13) is forced to be at maximum when the sensor frequency is fast, and at minimum when it is slow. These maximum and minimum frequencies/voltages correspond to the magnet being close or distant to the sensor.

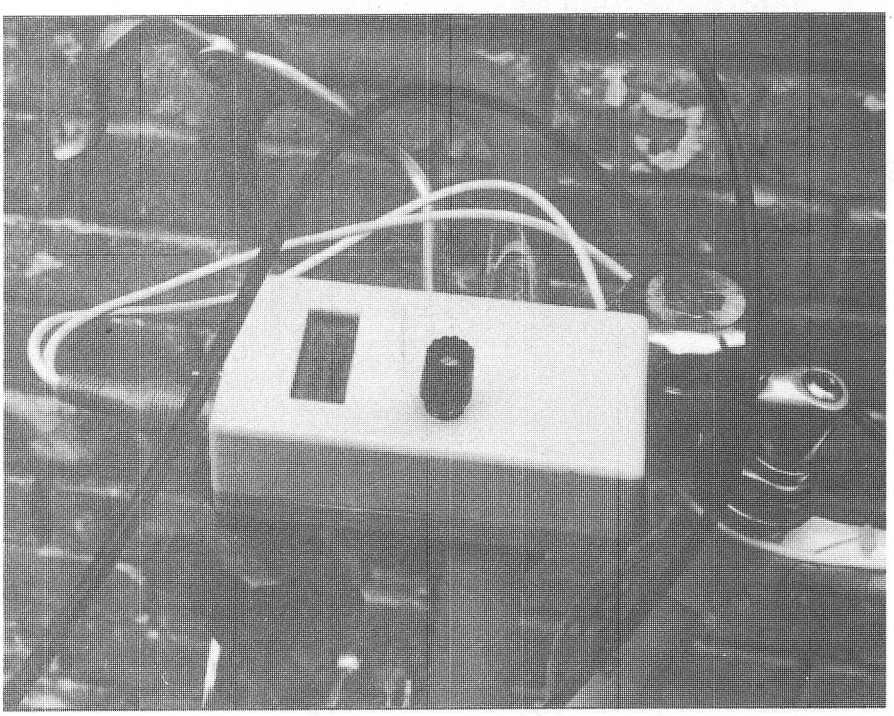
Converting the maximum and minimum voltages to cleanly changing Logic 1 and Logic 0 signals is done by the twin Schmitt inverter gates IC2a and IC2b. It is the output from IC2b pin 10 which provides the sharp logic level change which the microcontroller then needs to detect.

MICROCONTROLLER CIRCUIT

How the microcontroller processes the logic signal from the Sensor Interface will be discussed presently. Before it is, though, let's consider the general processing and display circuit in Fig. 2.

There are two active components in Fig. 2, the PIC16C84 microcontroller, IC3, and the intelligent liquid crystal display (l.c.d.), X3, which is based on the Hitachi HD44780 control chip. In-depth discussions of both these devices have appeared in previous issues of *EPE*; the l.c.d. module in February and March '97 issues (*How to Use Intelligent L.C.D.s),* and the PIC in several issues over the last year or so, notably February '96 (*Simple PIC16C84 Programmer*).

To briefly recap on the PIC16C84, it has one kilobyte (1KB) of EEPROM (electrically programmable read only memory) program memory, 64 bytes of EEPROM long-term data memory, and 36 volatile general-purpose data registers (SRAM – static random access memory). There are



PIC-Agoras Wheelie Meter installed on the handle-bars of a bicycle.

two bidirectional ports, the 5-bit Port RA and the 8-bit Port RB. The device includes its own master clock generator (oscillator) whose frequency is set externally either by a quartz crystal network or an *RC* (resistor/capacitor) network. There are four modes of clock operation:

RC Resistor/capacitor oscillator

XT Standard crystal oscillator

HS High speed crystal/resonator

LP Power saving, low frequency crystal

For this application, the XT mode is used with a 3·2768MHz crystal, X2, in conjunction with capacitors C3, C4 and resistor R8.

Also included in this microcontroller are a Watchdog Timer (WDT), and an edge-sensitive interrupt pin at Port INT/RB0 (pin 6). Program memory can be loaded serially while in-circuit, Port CLK/RB6 (pin 12) being the clock input, and Port DIO/RB7 (pin 13) the serial data input.

When not in use for loading program data, both the latter pins are available for normal data input/output.

To set the device into program loading mode, input $\overline{\text{MCLR}}$ (pin 4) has to be set to a voltage of between +12V and +14V. This input also serves as the Reset line, being taken to 0V to activate the in-chip reset routines. Normally, it is held at about the same positive voltage as supplies the chip's power, between +4V and +6V.

Switches S2, S3 and S5, resistors R6, R16, R17 and R9, and diode D2 are all associated with the programming mode and will be discussed when programming is described.

MODE SELECTION

Switch S4 is a panel mounted rotary 4-bit binary switch of which only the first three bits are used, providing eight modes (16 lines of display) that can be set via Port pins RA0, RA1 and RA2. The software program constantly monitors these pins and takes the appropriate action depending on the mode selected.

The modes are:

0 Show trip elapsed time and distance in kilometres

1 Show current speed and trip average speed in kilometres

2 Show trip peak speed and absolute distance since unit's first use, in kilometres

3 Show trip elapsed time and distance in miles

4 Show current speed and trip average speed in miles

5 Show trip peak speed and absolute distance since unit's first use, in miles

6 Show number of wheel rotations detected per second and average per 10 seconds

7 Reset all trip counters to zero

Switch S4 has only one pole (P), consequently only the selected "way" (pin 1, 2 or 4) can be held at a specific logic level via the pole. To keep the unselected pins at a known logic level requires them to be held at that level via biasing resistors, in this case R13 to R15.

The pole of the switch is put under program control, via Port pin DIO/RB7 (13). Only when the switch needs to be read is this line taken high. At all other times it is held low.

DISPLAY MODULE

The l.c.d. module X3 has two lines each of eight groups of display pixels (a 2-line 8-character unit). As discussed in *EPE* Feb '97, any of the module's internal library of alphanumeric characters and other symbols can be routed to any of the 16 display positions under software control.

Routing the data to be displayed from the microcontroller IC3 to the l.c.d. X3 requires the data code to be set up on the l.c.d. data pins and the appropriate toggling of two control lines, RS and E (pins 4 and 6). The l.c.d. can be operated in 8-bit or 4-bit mode. In the latter mode, data is only applied to data pins D4 to D7 (pins 11 to 14), pins D0 to D3 (pins 7 to 10) being left unconnected.

Data can be sent to the module either as Control commands or Character display data. The Control commands available are numerous, and can include such things as which line and pixel area the characters are to be displayed at; where the cursor sits; whether data is to be scrolled and in what direction; etc. To set Control command mode, line RS is taken to Logic 0. Setting RS to Logic 1 puts the display into Character transfer mode.

Control and Character data transfer are actioned when line E is taken low, to Logic 0.

There is a third control line, R/W (pin 5), This controls whether the l.c.d. is to receive data (Write mode, Logic 0), or supply it back to the outside world (Read mode, Logic 1). In this application, only Write mode is required, so the R/W pin is tied to the 0V line.

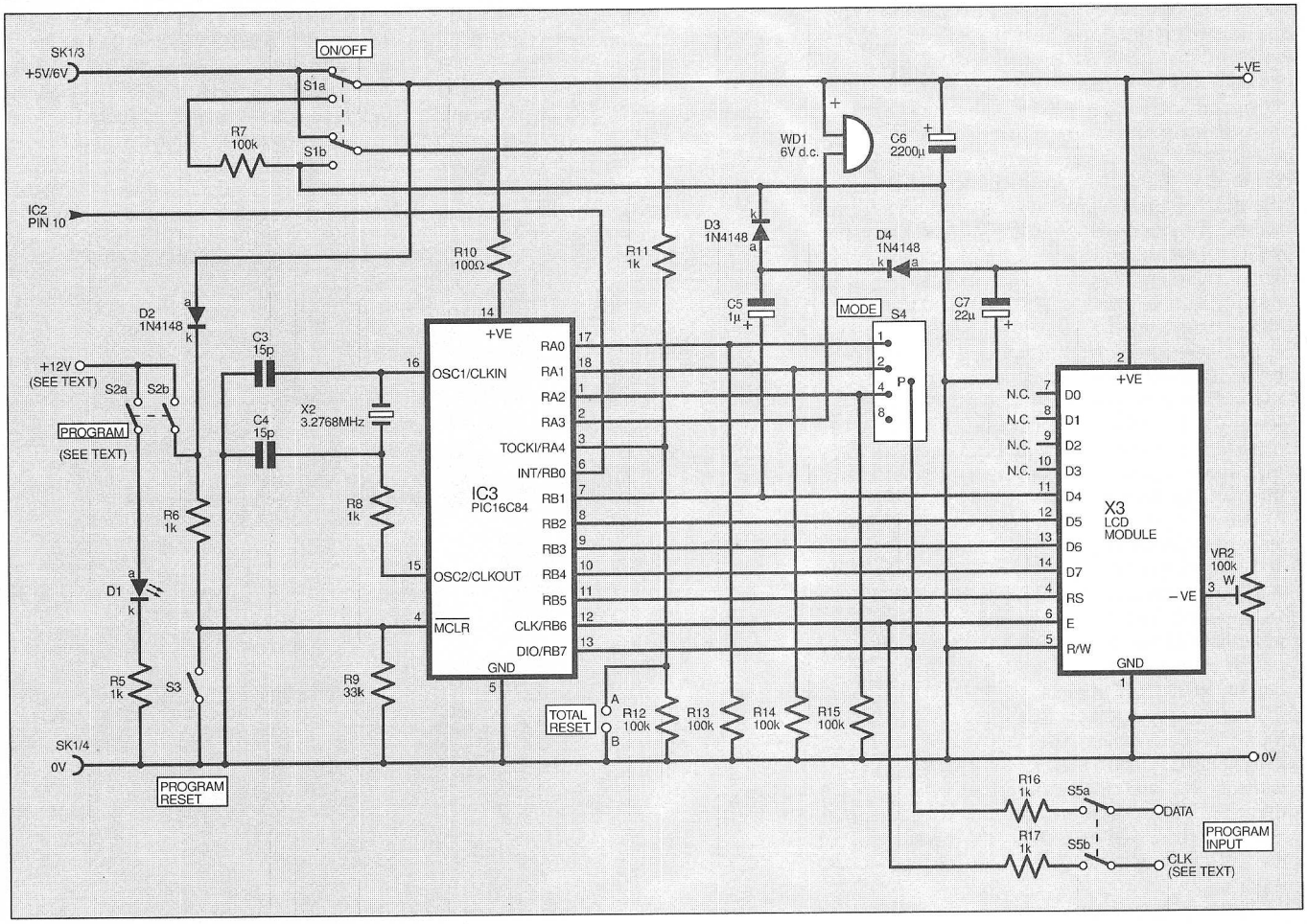


Fig. 2. Circuit diagram for the general Processing and Display stages of PIC-Agoras.

The l.c.d. module needs a negative voltage for screen contrast biasing. A simple negative inverter circuit is used, comprising capacitors C5, C7, diodes D3, D4, and preset potentiometer VR2. When IC3 is not actually transferring data to the l.c.d., Port RB1 (pin 7) is programmed to constantly toggle this line up and down. The resulting waveform is a.c. coupled by C5, negatively rectified by D3 and D4, and smoothed by C7. Preset VR2 sets the negative bias voltage for the desired screen contrast.

(The need for this negative bias is much bemoaned! Why couldn't the manufacturer allow this version of the l.c.d. to be biased to the 0V line instead, as is the case for other versions?)

AUTOMATIC SAVING

The PIC-Agoras system has been designed and programmed so that at the moment of switching off its power (by switch S1), the program automatically stores the current trip data – time elapsed, distance covered and peak speed – into the chip's EEPROM data memory. Next time the unit is switched on, this data is automatically recalled. You can thus go off for refreshments during a trip without wasting battery power!

While the unit is switched on, capacitor C6 (having a fairly high value) is held charged at close to full battery voltage level. Port pin TOCKI/RA4 (pin 3) is held high via resistor R11, and IC3 itself is powered via R10.

When S1 switches off the battery, IC3 and the display are powered by the charge held in C6. Port TOCKI/RA4, though, is immediately biased to a Logic 0 level via switch S1b. The program responds to this logic change and immediately jumps into the data-save routine. This routine is speedy and is completed before the charge on C6 has decayed below the voltage at which IC3 can continue to function.

Resistor R12 prevents TOCKI/RA4 from being indeterminately biased at the moment when the pole of S1b shifts between its other two contacts (it is a break-before-make switch, a make-before-break switch would cause the battery to be shorted to 0V by S1b during this transition).

NO CONFUSION

Resistor R7 continues to discharge C6 to a zero volts level after IC3 has ceased to function. A couple of minutes or so should be allowed to elapse before switching the unit on again. Failure to do this could confuse the microcontroller if its supply voltage from C6 has not fallen sufficiently and the chip is still being powered at an acceptable level. Such confusion could interfere with the EEPROM data storage values, and prevent the program restarting when power is switched on again.

The purpose of resistor R10 in IC3's positive power line is to limit the current it tries to draw from C6 during power-down.

Much thought and experimentation were put into the automatic saving problem, but no better solution could be envisaged. The reason for wanting automatic storage at switch-off was to overcome human memory loss!

Working data totals are held in volatile memory since there is a finite limit of times that the EEPROM data memory can be written to (typically 1,000,000 times). Consequently, it cannot be used as an active writable working memory during normal high-speed processing.

Obviously, then, a mere human would have to remember to switch the unit to another mode in order to store the trip data prior to switching off; an action all too easy to forget after a lengthy, tiring ride!

RESETTING BY CHOICE

It *is* possible to store the trip data by a switched action if you prefer. Switch S4 Mode 7 is the Reset mode. Switching to this mode triggers a 10-second countdown timer routine, during which Port RA3 activates the warning buzzer, WD1, accompanied by a display of the countdown seconds remaining. (The buzzer also "beeps" when power is switched off.)

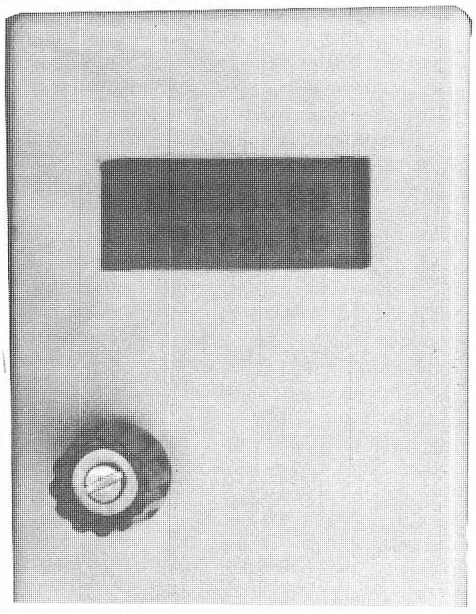
At the end of the countdown, the EEPROM trip-data storage routine is actioned; current trip data is stored into the EEPROM memory and running totals (distance covered since first-ever use of the unit) are updated. The ordinary working memory areas for the trip are then reset to zero. Normally, this routine is selected at the start of a trip.

During the countdown, Reset mode can be aborted so that if you inadvertently switch to it you can switch to another mode without updating the EEPROM and resetting trip data.

The buzzer is included to give adequate warning that Reset mode has been selected. Switching out of Reset mode resets the countdown timer so that next time this mode is selected, the count period is again set to start from 10 seconds.

TOTAL RESET

There is a further Reset facility provided, the ability to totally reset *all* the totals in the EEPROM memory back to zero. For data safety reasons, this facility is not accessible from outside the unit, which needs to be opened. In the circuit diagram of Fig. 2 are two points alongside



Completed display unit showing readout window and Mode switch.

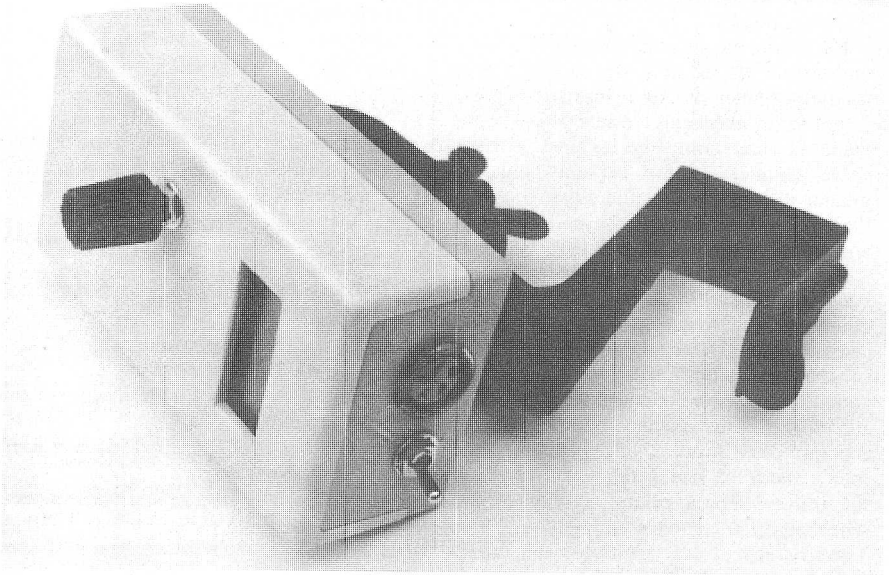
resistor R12 marked *A* and *B*. With the unit's power switched off, these two points should have a link wire soldered across them, connecting Port TOCKI/RA4 to the 0V line.

The power is then switched on and during the program's initialisation routine, the fact that Port TOCKI/RA4 is held low will be detected by the software. The program then jumps to the Total Reset routine, zeroing all EEPROM data memory locations. The program then goes into a catatonic state, refusing to do anything further.

Following Total Reset, power should be switched off and the *A* – *B* link removed. The unit can now be closed up, switched on again and used in the normal way.

SOFTWARE

The controlling software program is *complex*! Indeed, it's too complex to describe in detail, so only a brief run-down can be given. It is also lengthy, occupying nearly all available program memory locations (three bytes to spare!).



Finished control unit showing power/signal socket, on/off switch and cycle mounting clamp.

and that one or more functions would have to be dropped. However, programmers love challenges and ways were found!

The program source code was written for assembly by TASM software, the program discussed in the earlier mentioned *Simple PIC16C84 Programmer* of Feb '96. That code is available as advised in *Shop Talk*.

Readers having assemblers other than TASM (MPASM, for example), should be able to translate the code without too much difficulty.

The program starts off with two initialisation routines. The first sets up various parameters in memory and on the Ports, and retrieves stored data from the EEPROM memory. The other is for the l.c.d. module, setting it into 4-bit 2-line control mode. During this sequence of events, the need for Total Reset is examined and actioned accordingly.

The main program now starts, from within an interrupt detection loop. This looped routine examines the status of an internal timer and that of the internal register on Port INT/RB0 (pin 6). This is the line that is connected to the final output of the Sensor circuit at IC2b pin 10.

If a wheel rotation has been detected (an internal register flag is automatically triggered without program intervention), two wheel-count counters are incremented. The maximum pulse count rate is 25Hz (25 wheel revolutions per second).

TIME - OUT

If the time-out flag of the timer has not been set, the program remains in the interrupt loop. If the flag has been set, which automatically occurs every 1/25th of a second, the first elapsed-time clock counter is incremented and the program jumps to the main calculation and display routine.

The first procedure in this routine is to read the value of the first elapsed-time counter. If it has received 25 pulses (= one second), the elapsed seconds, minutes and hours counters are incremented appropriately. If the time reaches 24 hours, all four elapsed-time counters are reset to zero. (An actual time of day clock facility is not included.)

Each time the seconds counter reaches a multiple of 10 seconds, the second wheel rotation counter has its information transferred to an averaging counter. The count value is also compared against a peak value counter; if the present count is greater than the peak value already stored, the peak value becomes the present count and stored in EEPROM memory.

When these 10-second procedures have been completed, the second wheel rotation counter is reset to zero.

Every *one* second, though, the contents of the first wheel rotation counter are added to a distance counter and then reset. Incrementing the distance counter is quite a complicated process since it has to take into account the size of the wheel to which the rotation count refers. Setting of the wheel size value is covered later.

Wheel size is stored in two bytes as part of the program itself. The data is retrieved by two sub-routine calls, one for each byte.

together and stored in three dedicated registers as a 24-bit binary number. This value represents the distance travelled during the present trip.

MODE STATUS

The program next reads the status of Mode switch S4. The route that the program now takes depends on the Mode detected, but all routes are accompanied by an updating of the display screen in some manner.

In any Mode, though, only one line of the l.c.d. is updated at a time. The program alternates between the two lines, and this alternation occurs every 25th of a second, so the data is effectively always seen as a real-time value. Each line of the display has eight bytes of data sent to it, which comprise an appropriate mixture of alphanumeric characters and blanks. These bytes are sent as a consecutive sequence of 16 nibbles (2 nibbles = 1 byte) as required by the 4-bit l.c.d. control mode.

The Mode procedures and displays are as follows:

MODE 0

Display Line 1: Trip elapsed time routine. Converts seconds, minutes and hours data from its 3-byte decimal storage format to 8-byte l.c.d. display format with colon and decimal point inserted; 24-hour clock.

Display Line 2: Trip distance in kilometres routine. Converts distance data from 3-byte binary format to 8-byte l.c.d. display format, to two decimal places of accuracy, inserting decimal point and identity letters KD (Kilometres/Distance).

Typical display: 01:23.05
KD031.53

MODE 1

Display Line 1: Current speed in kilometres routine. Evaluates speed from 10-second wheel rotation counter average data, converting answer to 8-byte l.c.d. display format, to two decimal places of accuracy, inserting decimal point and identity letters KV (Kilometres/Velocity). The 10-second averaging of the rotation count helps to smooth out small distracting variations in the displayed answer caused by the rate of the sampling process.

Display Line 2: Average speed in kilometres routine. Evaluates average speed from distance and elapsed time data, converting answer to 8-byte l.c.d. display format, to two decimal places of accuracy, inserting decimal point and identity letters KA (Kilometres/Average).

Typical display: KV047.31
KA022.79

MODE 2

Display Line 1: Peak speed in kilometres routine. Evaluates peak speed from peak wheel rotation count data, converting answer to 8-byte l.c.d. display format, to two decimal places of accuracy, inserting decimal point and identity letters KP (Kilometres/Peak).

Display Line 2: Total distance (ever covered) in kilometres routine. Evaluates total distance covered from current distance travelled and previous absolute distance travelled as held in EEPROM, converting answer to 8-byte l.c.d. display format, to

Typical display: KP053.73
K0431.53

MODE 3

Display Line 1: Trip elapsed time routine. Identical to Mode 0, Line 1.

Display Line 2: Trip distance in miles routine. Follows kilometre conversion routine of Mode 0 Line 2, then converts answer to miles equivalent in 8-byte l.c.d. format, to two decimal places of accuracy, inserting decimal point and identity letters MD (Miles/Distance).

Typical display: 01:23.05
MD014.24

MODE 4

Display Line 1: Current speed in miles routine. Follows kilometres conversion routine of Mode 1 Line 1 then converts answer to equivalent miles value in 8-byte l.c.d. format, to two decimal places of accuracy, inserting decimal point and identity letters MV (Miles/Velocity). One kilometre is taken as five-eighths (5/8) of a mile (in all Modes 4, 5, 6 calculations).

Display Line 2: Average speed in miles routine. Follows kilometres routine of Mode 1 Line 2 then converts answer to equivalent miles value in 8-byte l.c.d. format, to two decimal places of accuracy, inserting decimal point and identity letters MA (Miles/Average).

Typical display: MV029.56
MA014.24

MODE 5

Display Line 1: Peak speed in miles routine. Follows kilometres routine of Mode 2 Line 1 then converts answer to equivalent miles value in 8-byte l.c.d. format, to two decimal places of accuracy, inserting decimal point and identity letters MP (Miles/Peak).

Display Line 2: Total distance (ever covered) in miles routine. Follows kilometre routine of Mode 2 Line 2 then converts answer to equivalent miles value in 8-byte l.c.d. format, to two decimal places of accuracy, inserting decimal point and identity letter M.

Typical display: MP033.58
M0269.70

MODE 6

Display Line 1: Current wheel rotation count during one second routine. Converts immediate wheel turns count to 8-byte l.c.d. format, inserting letters TL (Turns/totaL), three blanks and three digits. The count value is that since the last reset of the counter and so is seen to increment from zero to total one-second count, and then back to zero, and so on. A useful routine when aligning sensor and magnet on the vehicle.

Display Line 2: As for Mode 6 Line 1, but is an averaged value for ten second period, with prefix letters TA (Turns/Average).

Typical display: TL 005
TA 018

MODE 7

Display Line 1: Visual warning of RESET countdown in progress with seconds remaining count displayed; buzzer sounds. Remains at zero count and buzzer

continues to sound until Mode is changed.

Display Line 2: As for Mode 7 Line 1.

Typical display: RESET! 7
RESET! 7

ARITHMETIC

Most of the above Mode routines use repeated accesses to multiplication and division routines, which not only are too difficult to explain briefly, but also *extremely* lengthy, consuming much of the 1KByte program space available. Regrettably, PIC microcontrollers do not have these routines built into their command structure and so they have had to be written as blow-by-blow sub-routines. What a hassle!

Writing the routines proved how accustomed one can become to programming languages which *do* have the routines built in – with the 80486 and even the much earlier 8086 microprocessors, for example, programmers have a much easier time of it in this respect, though without some of the other benefits that a PIC microcontroller can offer.

POWER SUPPLY

It is intended that this design should be powered by a "heavy duty" 6V battery; a supply between 4V and 6V is acceptable. The *absolute maximum* is 7V and this must NOT be sustained or exceeded.

Current consumption at 5V is approximately 14·5mA.

CONSTRUCTION

Full details of the printed circuit board (p.c.b.) assembly and wiring to other components are shown in Fig. 3, as is the track layout for the copper foil pattern. This board is available from the *EPE PCB Service*, code 141.

If you do not intend to program your own PIC16C84 (see later), or if you already have a PIC programmer, then resistors R5, R16 and R17, l.e.d. D1, switches S2, S3 and S5, and computer connector PL2 can be omitted. Similarly, you can ignore further references to any aspect relating to on-board programming of IC3.

Experienced constructors will, of course, assemble the p.c.b. in whatever order they like. Other readers may prefer to follow the order in which the author assembled his prototype:

First, insert the link wire shown above IC3, and then the i.c. sockets. Do not try to save money by omitting the latter; it is much easier to just pull out an i.c. rather than have to unsolder it, should you ever need to replace it.

Next, insert the resistors and diodes, followed by the preset potentiometers and the capacitors. Note that capacitor C6 and preset VR2 are mounted on the rear of the board (trackside). Mount crystal X2 so that it lies horizontally above capacitor C3 and resistor R14. Now insert 1mm terminal pins where needed for connecting wires to off-board parts. Finally, solder in switch S4.

Make doubly-sure that the polarity of the diodes and electrolytic capacitors are observed, as well as the orientation of switch S4 (check this with a meter if you are not sure, though it should be marked underneath).

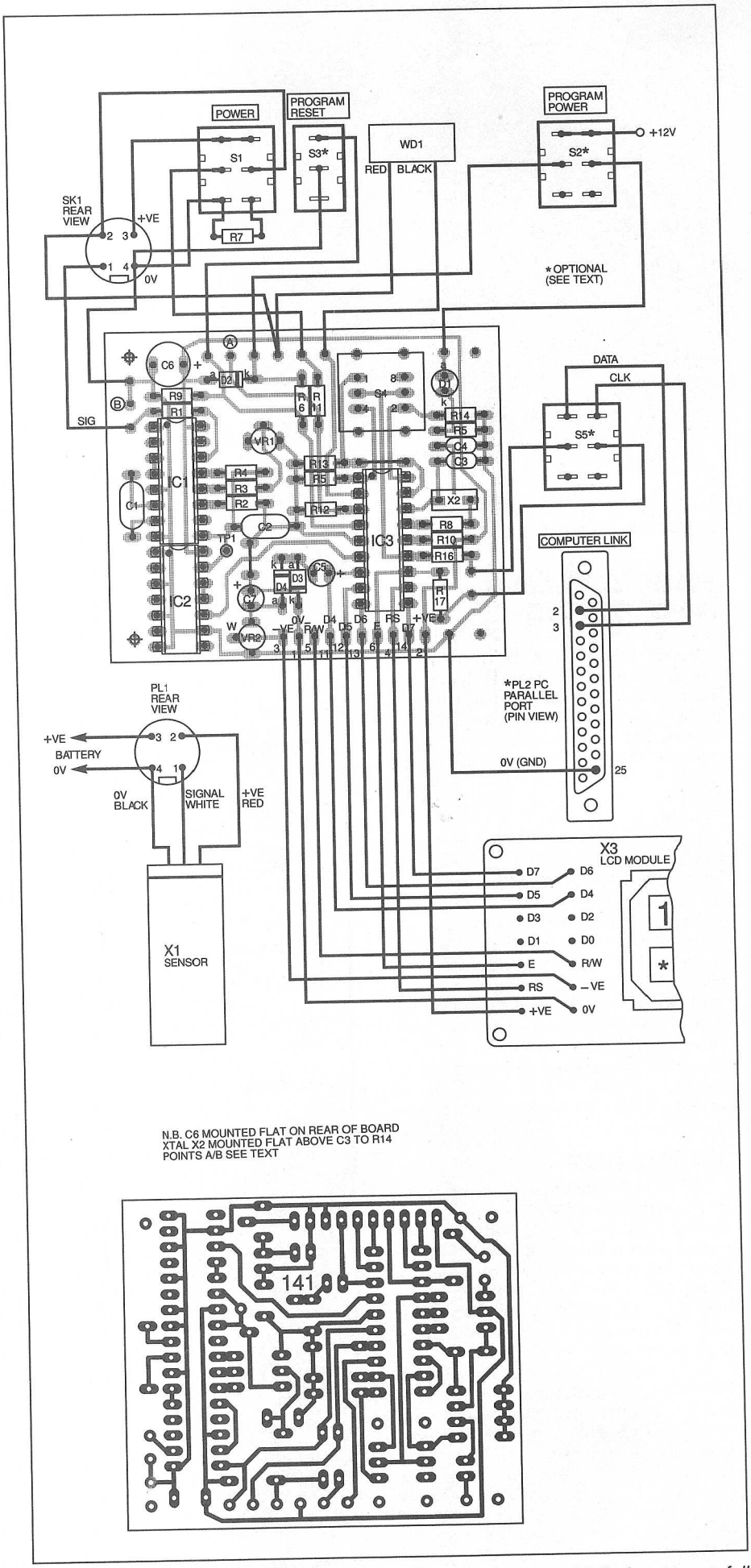
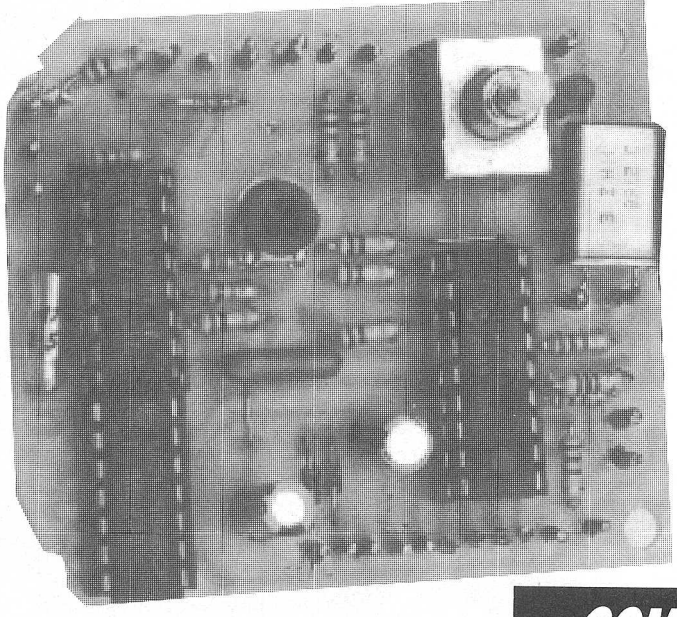


Fig. 3. Printed circuit board component layout, interwiring and full size copper foil master for PIC-Agoras. Note that C6 and VR1 are mounted on the track side of the p.c.b.



Layout of components on the prototype printed circuit board.

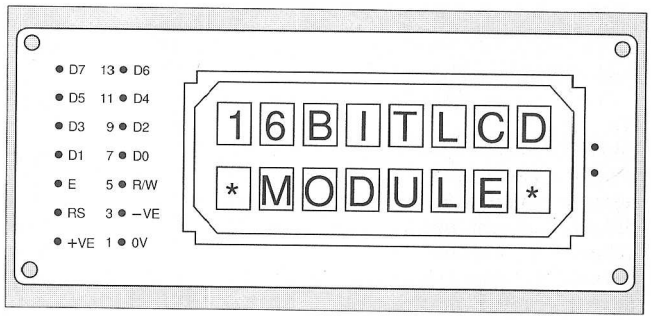


Fig. 4. Pinout information for the l.c.d. 2-line 8-character module.

CASE DRILLING AND FITTING

Before wiring the p.c.b. to anything else, drill out the lid of the recommended case to accept the mounting bush of switch S4. The l.c.d. requires an oblong cut out in the lid, cut so that only the display area is visible. The time-honoured technique of multiple perimeter hole drilling is suggested for this, pushing out the unwanted section and filing down the edges to full smoothness.

Drill out one end of the other part of the case to accept socket SK1 and switch S1. Ultimately, this end of the case will be at the same end as the display.

The l.c.d. module in the prototype was not bolted to the case. Fortuitously, its width made it a snug fit between the side walls of the case. The buzzer was then stuck to one wall with double-sided adhesive tape immediately above the l.c.d., so holding the latter in position.

The p.c.b. is secured inside the box by means of switch S4's bush and nut assembly. It may be necessary to trim the board's corners.

WIRING-UP

Before you mount the l.c.d., buzzer and p.c.b., carry out all the inter-component wiring. Keep leads fairly short but long enough so that you can manipulate things during the process. Connect the wires to their terminal pins at the *rear* of the p.c.b.; with the l.c.d., push the tinned ends of the wires through its p.c.b. holes and solder them on the display side.

If you are going to program your own PIC16C84, now wire-up switches S2, S3 and S5, and the three wires to computer connector PL2. The latter wires should be long enough so that PL2 can be plugged into the parallel printer port of your PC-compatible computer. The author's wires were about two metres long and no problems were experienced with data transfer.

Temporarily wire the FGM-3 sensor to DIN plug PL1, using the sensor's own leads. Also connect the battery-clip wires to this plug. Connect switch S2 to a +12V (or up to +14V) power supply. From this supply also make a 0V connection to SK1 pin 4. Observe the correct polarity of all these connections.

COMPONENTS

Resistors

R1, R5, R6, R8, R11, R16, R17	1k (7 off)	
R2, R4, R7, R12 to R15	100k (7 off)	
R3	10k	
R9	33k	
R10	100Ω	

All 0·25W 5% carbon film

See SHOP TALK Page

Potentiometers

VR1, VR2	100k min. round cermet preset (2 off)

Capacitors

C1	1n polyester
C2	15n polyester
C3, C4	15p polystyrene (2 off)
C5	1μ radial elect. 63V
C6	2200μ radial elect. 16V
C7	22μ radial elect. 16V

Semiconductors

D1	red l.e.d. (see text)
D2 to D4	1N4148 signal diode (3 off)
IC1	4046 phase locked loop
IC2	4584 hex Schmitt inverter
IC3	PIC16C84, pre-programmed, see text

Miscellaneous

X1	FGM-3 magnetic field sensor
X2	3·2768MHz crystal
X3	2-line 8-character l.c.d. module (HD44780 compatible)
S1	min d.p.d.t. toggle switch
S2, S5	min. d.p.s.t. toggle switch (3 off)
S3	min. s.p.s.t. toggle switch
S4	4-bit binary, rotary switch
PL1	4-pin DIN line plug
PL2	25-pin parallel printer port connector
SK1	4-pin DIN chassis socket
WD1	6V active buzzer

Printed circuit board, available from the *EPE PCB Service*, code 141; 14-pin d.i.l. socket (2 off); 16-pin d.i.l. socket; collet knob for S4 (3·2mm shaft) and coloured insert; min. 2-way terminal block; small disc magnet; heavy-duty 6V battery and clip; 1mm terminal pins; 2-tone plastic case, 120mm × 66mm × 30mm (l × w × h); cable ties; connecting wire; solder, etc.

Approx Cost Guidance Only £39 excl. programming parts

FIRST CHECKS

Leave all the i.c.s out of their sockets for the moment and *do not* plug PL2 into the computer.

Thoroughly check all your soldered connections, both on the p.c.b. and on all other parts, preferably using a close-up magnifying glass.

When satisfied, plug PL1 into SK1, switch off programing switches S2, S3 and S5. Switch on both the normal 5V/6V supply and the 12V programming supply.

Referring to the circuit diagrams in Fig. 1 and Fig. 2, take a few readings around the p.c.b. to check that the positive supply is reaching the parts that it should, e.g. IC1 pin 16, IC2 pin 14, IC3 pin 14, l.c.d. (X3) pin 2, and IC3 pin 4. At the latter pin, the voltage will be about half a volt below the 5V/6V supply line voltage due to the presence of diode D2 and resistors R6 and R9.

If you have an oscilloscope, check that there is an output frequency signal from the Sensor at IC1 pin 14.

Switch off the main power supply switch S1 and switch on Program switch S2. Immediately check that the +12V supply *is not* reaching IC3 pin 14. (If it is, diode D2 is in the wrong way round; switch off and change it.) The l.e.d. D1 should light up and IC3 pin 4 should now be at the 12V level. Switch on Reset switch S3 and check that IC3 pin 4 is now at the 0V level.

Switch off both power supplies, insert all three i.c.s., observing the usual static-prevention precautions. Perform the above tests again from the beginning. An oscilloscope check at IC3 pins 15 and 16 should confirm that the clock generator around crystal X2 is functioning (only a scope can confirm this easily). At present, unless you are using a pre-programmed IC3, it is unlikely that the l.c.d. screen will show any detail, remaining totally blank.

SENSOR CHECKING

The Sensor circuit can be checked without IC3 having been programmed. Tape the sensor X1 to the workbench and connect a multimeter to test point TP1. Set the meter to a 5V d.c. (or greater) range.

Pass a small magnet back and forth past the sensor and observe the meter response as you do so. It will be found that a greater response is likely to occur nearer to one end of the sensor than the other, depending on which side of the

magnet faces the sensor. Note these facts in some way on both items (e.g. sticky label or Chinagraph pencil).

While passing the magnet, experiment with various settings of preset VR1 and jot down the resulting observed voltage differences. Don't be too fussy about these tests since more accurate setting up will be done once everything is mounted on the vehicle and its wheel.

However, you should establish that there are magnet positions and settings of VR1 which cause the output from IC2b pin 10 to swing between logic level extremes (0V and +5V).

SOFTWARE DIFFERENCES

There is a slight difference in the software for those who are doing their own programming and that supplied in the pre-programmed chips. The difference allows home-programmers to preset the wheel size from within the software. Pre-programmed chips can only have the wheel diameter set externally, by the user, via a signal generator. This is due to the complexities of having chips pre-programmed commercially to meet individual wheel diameter requirements.

In the software for the pre-programmed chips, diameter parameters can be set according to the frequency fed into the unit from the signal generator. These parameters are stored in the EEPROM data memory, and not in the program memory.

Home-programmers may use either technique. For the latter method, the software is used without modification. For the preset internal method (which, it has to be said, is the simplest way if you have the equipment), the following changes must be made:

Delete source code lines 173 to 175:

```
PAGE1
BCF TRISA,3
PAGE0
```

Delete source code lines 191 to 195:

```
btfss PORTA,3
GOTO WHEEL
PAGE1
BCF TRISA,3
PAGE0
```

Delete all source code lines from 1301 to 1317 inclusive. You MUST retain the final line of the software (.end)

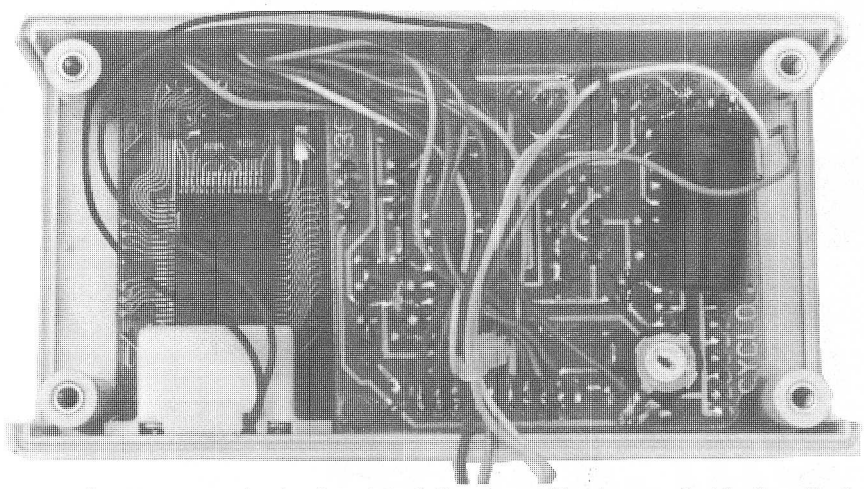
All 'comments' following the colon (;) in all the above deletion lines (not shown here) should also be deleted.

Source code lines 295 and 296 must be reinstated by deleting the colon (;) immediately in front of them (they are inoperative in the unmodified software):

```
SIZE1: retlw 45
SIZE2: retlw 56
```

There must be no blanks in front of these commands.

Having modified the lines via a text editor, such as the MS-DOS 'EDIT' software found on most recent PCs, save the software as an ASCII file (NO formatting commands specific to many word processors must be contained in this file). The software is then ready for assembly.



Layout of components in the interior of the case. The buzzer holds the display module in place.

SOFTWARE PROGRAMMING

Assuming all is well, IC3 can be programmed on-board by the computer. Load the software into the computer via an appropriate PIC16C84 software assembly program (e.g. TASM). The software must have been assembled as a binary file with extension *.com*. Plug connector PL2 into the parallel printer port.

The PIC16C84 first needs to be initialised with its basic operational parameters. Switch on Reset switch S3 (Reset mode). Switch on S2 (Program mode – l.e.d. on). Switch on Program Input switch S5, connecting the PIC to the computer's printer port. Switch off Reset switch S3. Run the initialisation routine provided with the PIC programming software package, and at the computer screen prompts:

Set the Clock for crystal XT mode (up to 4MHz)
Set Watchdog Timer OFF
Set Power-on-reset ON

The programming software may advise different orders of switching the Reset and Program switches, if so, follow its instructions. Note, though, that any switch numbers referred to in the programming software are unlikely to correspond to the switch numbers of PIC-Agoras.

Having completed the initialisation, the main software can be downloaded to the PIC16C84. Assume for the moment that the wheel size data has already been set to the required value. If it has not, don't worry, it can be done later and the PIC can be reprogrammed with the revised software. You can reprogram the PIC at least 100,000 times if you wish, and probably more. The default wheel size values are those used by the author, for a 27·5 inch diameter wheel.

Leaving Program switch S2 on, again set Reset switch S3 on, and then off. Run the downloading software package and follow screen prompts (also following any switch setting requirements, if different).

When downloading is complete, as advised on-screen, switch on S3, switch off S2, switch off S5, in that order.

Set Mode switch S4 to Mode 1 (fully anti-clockwise if the switch rotation is lugged; if it's not, just guess, no harm will come).

Next month: First Display Run; Downloading Failure; Setting Wheel Size and Alternative Sensing etc.