Rally Computer Design Specifications

Eric Dinger

November 18, 2012

Contents

1 Market Research

Looking at the market, no widely used rally computers are made in America for the type of rallying done here. All the common ones are made in England. Another problem is that outside of Monit, no rally computers have been designed in the last decade to take advantage of advances of technology and interfaces (Alfa recently upgraded to a new display technology). In addition these are all closed source systems (even Alfa who will sell you their source), by making my rally computer open source I hope to encourage development of new features that may be too small for a manufacture to focus on, but

would be useful to some. In addition as part of the long term design I would like to feature integration with smartphones either for display or dataloging.

The following is a small sample of feelings I've encountered on the internet about existing rally computers and observations I've noticed broken down by brand.

- **TerraTrip**
  - Unreliable
  - Awkward interface
  - Display can be hard to read
  - Has clock
- **Brantz (2 S Pro)**
  - Simple
  - No clock
- **Coralba**
  - TSD focused
  - Expensive
  - Programable to an extent
  - Has clock
- **Monit**
  - Some version have clock
  - Graphical LCD based display, menu based interface
  - 2 or 4 button interface
  - Seems popular, mainly for it's ease of use.

2 High Level Overview

Looking at these points and what I want out of the first generation of my Rally computer I selected a few items

- Stage rally focused
- 3 Odometers (1 total, 2 interval)

- Switchable odometer directions (on 2 odo's)
- Current speed display
- No clock
- Modern menu based interface
- Automatic factor computation
- Factor = number of pulses per mile or kilometer (depending on unit preference)
- Character (not graphic) LCD display
- Keybad interface
- Open source software and hardware

3 Hardware overview

3.1 Display

In the first prototype the display will consist of one 20x4 hitachi HD47780 based character LCD display using an adafruit I^2C& SPI backpack. This option was chosen for several reasons, the hardware is on hand and easy/cheap to acquire if either is damaged during testing. The reason for going with the serial backpack was simple, running the display even in 4 bit mode requires many pins on the microcontroller, using the backpack requires only 4 (3 of which can be shared) when using SPI mode. SPI mode was chosen because of its high throughput compared to I^2C.

Future prototypes and production will feature a New Haven Display 20x4 OLED character display, this is the same type of display used by the refreshed Alfa rally computers. These displays are dimensionally equivalent to the 20x4 LCD display and features onboard SPI communications, eliminating the need for serial backpack. The instruction sets differ between the two displays though.

3.2 Processor

The design will be based on the Atmel ATMEGA 328 microcontroller. This chip is more than powerful enough for the application, has enough pins, and is throughly documented on the internet in various locations, which is useful

since I want this project to be added to by rally enthusiasts, who are not necessarily programmers.

3.3 Probe interface

The idea is to be compatible with existing probes and stock VSS signals as possible. Currently the design uses a 2 pole Low-pass filter followed by a Schmidt trigger. The low-pass filter and Schmidt combo has a ceiling frequency of approximately 1.3KHz. The input to this filter is expected to be a 0-5v square wave.

3.4 Power supply

Coming soon. Currently powered off a USB car charger. Automobile power environments are a nasty place, and although I haven't ran any experiments I assume that race cars are worse due to many reasons.

3.5 Keypad

Currently a 4x4 matrix keypad manufactured by Greyhill.

3.6 Misc.

Connections to the outside world will use twist-lock connectors. Not looking at mil-spec at this time due to cost (~$40 per connector).

4 Software overview

4.1 Design considerations

Since the factor will be the number of pulses per mile (or km), we can use some common tire sizes to get an idea of the range of pulses expected using a sensor that measures at the lug studs.

Tire size	lugs	PPM
205/50-15	4	3496.72
15/62-15	5	4131.21
185/75-13	4	3477.73

We can see that through a range of common tire sizes the pulse count doesn't differ over to large a range, say 3000 to 5000 ppm. If we take a car traveling at 120mph with 5000ppm we see that the input frequency will be $\frac{(120*5000)}{3600} = 166.66$Hz

This is well below the ceiling frequency of the hardware low-pass filter, so we run into no issues there, and so far below the processor speed (16MHz) that if the interrupt handler is anywhere near efficient there will be no issues.

4.2 Program flow

Since this computer is quite simple and has no external sensors/chips to configure and bring up, the program flow is fairly straight forward. The following verbal description will be replaced by a uml diagram.

Startup:
Load stored distances and calibration.
Configure the interrupts and LCD/OLED display
Enter main program flow

Main loop:
Look for key press
At 10Hz rate update LCD/OLED display. Depending on screen calculate distances, speed, or show current menu

Interrupt handler:
Update pulse count
Measure time since last pulse (For speed measurement)

4.3 Keypad mapping

Since we are using a 4x4 keypad, it will be layed out in the traditional 9 key arrangment, with the addition of some other movement and entry keys.

1	2	3	↑
4	5	6	↓
7	8	9	
P	0		E

The use of the numbers is obvious, but the ↑ and ↓ will move the cursor between selected odo's or menu entries. E switches into entry mode, and when in entry mode saves the entered data. P switches between pages (menus).