Motorcycle Trip Computer

Functional Description
11/23/02
ET471
Reuben Brown


Introduction:

Electronics in the automobile industry take on all manner of forms, some much more useful than others. One of the most useful innovations I have seen installed on many cars is a trip computer. These computers vary in function, but most have one or two odometers, an average speed display, a trip time display, and also a clock. The best feature on most of these computers is a range function. This function tells the driver approximately how far they can drive before they will run out of gas.

Being an avid motorcyclist, I have thought that these computers would be ideal for motorcycle riding. I would be less likely to have to make an emergency switch to my reserve tank and hope that the next gas station was downhill. Also having odometers that can be used to track ride distances and service intervals would help to plan future excursions, keep up on maintenance and enhance the overall motorcycling experience.

Hardware Description:

The motorcycle trip computer will be implemented using the Motorola M68HC12B32 microcontroller. The microcontroller will process all user inputs and display data collected to the user on an LCD screen. Components can be broken down into five parts: the microcontroller, wheel speed sensor, 4x20 LCD display, 3 button keypad, and real time clock. The arrangement of these parts is shown in figure 1.2. Since there are only three buttons the user interface will be intuitive and very easy to navigate. When riding in traffic the user will want to be able to quickly choose which function to use, while maintaining all concentration on the task at hand - riding. A sketch of the final product with maximum dimensions is shown in figure 1.1.
The Motorola M68HC12 will be used to integrate all peripherals (as shown in figure 1.2) and execute the code that powers the motorcycle computer. The main function of the microcontroller will be processing user inputs from the keypad, and displaying information to the user on the LCD. The user interface will be described in detail later. The other function of the microcontroller will be calculating distance, speed, miles per gallon and range based on the data collected from the wheel speed sensor.

The M86HC912B32 is a very versatile MCU. This microcontroller meets the speed and memory requirements demanded by this project. The Motorola HCS12 series of microcontrollers are faster and have more onboard memory; however since the HC12 line fulfills my needs I don’t need to spend extra money for resources I won’t need. The 68HC11 line could also be used to implement this project; however since I have experience working with the HC12 I will be using it.
My program will be stored in the 32 Kbytes of flash EEPROM. I will be using both the 1 Kbyte of RAM for storing variables such as speed, distance, fuel tank size and gas used. I will also have to transfer these variables to the 768 Bytes of byte-erasable EEPROM. This will allow the system to be turned off without losing vital information.

I will be using a unipolar Hall Effect switch to implement wheel speed sensing. From the wheel speed and tire circumference, I will easily be able to calculate distance traveled, average trip speed and with more difficulty gas mileage, and range.

The sensor will interface with the microcontroller through Port T. as shown in figure 1.2. A sensor such as the Allegro A3240 is quite convenient because with a 5 volt supply it is directly compatible with digital logic, and no signal conditioning, or shaping will be necessary. The rise and fall time for this sensor are a maximum of 2 microseconds.

To keep accurate time, even when the computer is shut off, I will be using a real time clock. The clock will interface to the MCU through Port S; the serial peripheral interface. The clock I use will have a battery backup system. When the main power is connected, the battery is not used. As soon as the main power is turned off or lost, the RTC will run off of the battery. This system will keep the clock accurate even when the unit is shut off. With a high grade crystal this device will be able to keep accurate time so that the user will not have to constantly reset their clock.

The user will communicate with the computer from the three pushbuttons shown in figure 1.1. These buttons are interfaced with the MCU on Port B PB0 through PB3. The function of these buttons will be discussed at length in the user interface section to follow.
The display for the computer will be a 4x20 display it is connected to the MCU on Port A and PDLC4-PDLC6. This display shows to the user speed, distance traveled, menu options and all other user accessible functions. These are shown in figures 1.3 through 1.7. The display functions will be discussed in the user interface section.

All power needs for the computer system will be delivered from the 12 volt motorcycle battery. Since this is an enormous supply no special consideration for power consumption of parts will be an issue. The 12 volts will be regulated to 5 volts with regulator such as National’s LM340. This regulator is made for the automotive industry,
and is claimed to be practically indestructible. The regulator will have to be able to put
out current of at least 150 mA, according to my initial power consumption estimate. The
LM340 puts out 1 A of current which will easily meet power requirements for this
project. The computer will be wired with the ignition of the motorcycle. When the key is
on the computer will be on, likewise when the key is turned off the computer will turn
off.

**Software Description:**

This project entails building a portable computer for the motorcycle user. I plan
on writing most of the software for the computer system in C. Depending on specific
applications certain tasks or modules perhaps will be implemented with assembly
language. The program will be implemented with a preemptive kernel to switch between
tasks. This kernel will be implemented using the µCos operating system.

The programming will be broken down into eight modules. They are: MAIN,
COMPUTE_FUNCTIONS, RTC, WHEEL_SENSOR, LCD, USER_INPUT, EESTORE
and KERNEL. These eight modules should be able to handle all needs of processing
sensor information, calculations and user interfacing.

The MAIN will control what is being displayed to the user, and what state a
button press will result in. This module will control all aspects of the computer. It will be
in charge of setting and resetting functions. All of the various states discussed in the user
interface section are implemented through this module. Figures 1.8 through 1.10 show
the state diagrams for processing inputs.

The COMPUTE_FUNCTIONS module contains all functions having to deal with
computing speed, distance, range, miles per gallon, and trip time. I will have to write this
module. I anticipate that this will be a fairly tricky bit of code because there is a lot of data manipulation involved in computing range.

The RTC module controls communications between the MCU and the RTC. It will allow reading the clock, and writing to the clock. I will be adapting a module written by Todd Morton to implement this.

The WHEEL_SENSOR module will consist of a task to detect the pulses sent by the speed sensor. The module will then calculate speed based on time elapsed between pulses. I will have to write this module.

The LCD module contains functions that control the display of information to the user. I will be using a module written by Andrew Pace, and revised by Todd Morton. I will probably have to revise this code to meet the specific needs of this project.

The USER_INPUT module is going to detect user inputs. This module will debounce the pushbuttons and process which button has been pressed. I will modify a module, key.c, written by Todd Morton, this module is designed for a 4x4 keypad matrix. My three pushbuttons will be much simpler to implement, so less programming will be necessary.

The EESTORE module consists of writing vital variable to the byte erasable EEPROM. The variable must be stored in non-volatile memory to ensure that the variables will be accessible even if power to the unit is shut off or lost.

The KERNEL will be in charge of switching between tasks. I will be learning how to write a μCos in ETEC 454 this winter, and will write this module after I have an understanding of how to implement it.
**User Interface:**

The user interface for this project consists of a backlit 4x20 LCD and the 3 button keypad shown in figure 1.1. Since this system is designed to be used in the field by someone riding a motorcycle all visual display must be easily read. Any input required must be easily input so that the user can pay attention to riding, not navigating a complicated computer system.

The display shows to the user all information of the system. It consists of two menus. The first menu is pictured in figure 1.3. This is the default screen of the system, and is displayed on power up.

```
888 MPH  88:88 AM
88 TRIP SPD  88:88 TRIP
888 RANGE  8888.8 TRP
88 MPG     8888.8 ODO
```

Figure 1.3

The left column of the display shows: speed, average speed, range indicates miles remaining before running out of gas and miles per gallon is the average gas mileage the motorcycle is getting. The right column shows: the real time, trip time, trip odometer and an independent odometer.

The next menu, shown in figure 1.4, allows the user to reset functions and input data into the computer. This menu is reached by pressing the menu key. This display is a scrolling menu, so the two pictures are different portions of the same screen. To select an item to modify the user presses the select key. The state diagram in figure 1.9 shows how this menu is scrolled through. The cursors moves line by line down the menu, indicating which item will be modified. To modify an item the user merely presses input.
The first option, RESET TRIP, resets the trip time, trip speed and the trip odometer. The next three options set the selected item to zero. The last three options set the tank size, wheel size or time.

The clock set, wheel size and tank size are at the bottom of the list because they should rarely be needed. The state diagram for all these function is shown in figure 1.10. The set time function sets the system clock. This function will rarely be used, at most twice a year for daylight savings time. When this function is selected the display shifts to what is shown in figure 1.5 the user then enters the correct time by pressing input to increment the selected digit by one. Select will move the cursor to the next digit. When the proper time is set pressing menu returns to the settings menu.

The set fuel tank selection must be set for the computer to compute gas mileage and range. This screen is shown in figure 1.6. The user sets the tank size the same as the
clock, input increments the digit shown by the cursor, and select moves the cursor right. If the cursor is at the right end of the number and select is pressed the cursor wraps around to the first digit.

![SET TANK SIZE](image)

Figure 1.6

The last user screen is accessed from the main screen by pressing input. The display then shows what is picture in figure 1.7. The user can then input fuel consumption. This information is vital to the accuracy of the gas mileage and range calculations. If the user does not enter the amount of fuel dispensed at every fuel up the gas mileage and range displayed will be completely inaccurate. For this reason I removed this option from the settings menu, and made it directly accessible from the main display. When input is pressed the LCD displays what is shown in figure 1.7. Gas is entered the same way as time and tank size.

![INPUT FUEL](image)

Figure 1.7

Wheel size is entered the same as fuel, and tank size. The display will look the same as figure 1.7 except the units are feet, and it will have 3 significant digits. Figures 1.9 and 1.10 show how these states are reached and operated. Figure 1.8 shows the overall state behavior of the system.
Figure 1.8

Menu State Diagram

Figure 1.9
The development of this project is split into two main categories, hardware development and software development. The much larger task for this project will be writing software to make this device function. The major obstacle to the success of this project is that I don’t know the µCos platform. I will be taking ETEC 454 this winter to help prepare me for implementing the software for this project. I hope to have the hardware implemented by no later than the middle of winter quarter, and spend the rest of the year writing and modifying software.
Getting the items I need for the project should pose no problems, I already have several of the necessary parts. The major part I don’t have yet is the M68HC912B32. I will order this shortly. The lead time on this part shouldn’t pose any problems in the development plan.

**Winter Break:**

Week 1 – Ski Vacation, to help with sanity

Week 2 – Implement speed sensor

Week 3 – Begin power supply, Christmas time

**Winter Quarter:**

Week 1 – Finish power supply Design keypad

Week 2 – Finish hardware design

Week 3 – Modify RTC serial communications module.

Week 4 – Use and modify LCD interface module

Week 5 – Finish LCD module

Week 6 – Modify INPUT module for my four button keypad

Week 7 – Open for unseen setbacks

Week 8 – Begin SPEED_SENSOR Module

Week 9 – Finish SPEED_SENSOR Module

Week 10 – Modify STORE module for specific project needs

Week 11 – Wrap up all software worked on so far

Week 12 – Finals week prepare for death

**Spring Quarter**

Week 1 – Begin KERNEL Module
Week 2 – Finish KERNEL Module

Week 3 - 7 - Implement USER_INTERFACE Module

Week 8 – Open – fix all software bugs

Week 9 – Test and troubleshoot software and hardware interfacing

Week 10 - Field Test unit – nice long ride to LaConner

Week 11 – Prepare Demonstration – work out final bugs.

Week 12 – Demonstrate

Development of this project will take place in ET340. I will be using the CodeWrite, and Introl-Code compiler for programming. To debug software, I will be using the Noral 68HC12 Flex debugging system. I will use the Hewlett Packard 54645D Digital Oscilloscope for timing analysis of this project. For field testing, I will strap the prototype onto my Bike and head out on the road.

For demonstration purposes, I will simulate the motorcycle wheel with a motor. Unfortunately administration won’t let me ride around the third floor of the ETEC building, so I will have to simulate the project. I will then demonstrate the various user input functions of the computer. I plan on also having a presentation of the actual field use of the computer.

The demonstration will be of a prototype of the project. The computer will still be on the HC12 evaluation board. The prototype will be completely functional and enclosed in a handlebar mounted case.
Electrical Specifications:

- **Accuracy**
  - Speed – 1 MPH
  - Range – 10 Miles (After 3 tanks to build up Gas Mileage data)
  - Gas Mileage – 3 MPG
  - Distance – 0.1 Miles
  - Clock – 5 minutes (over 6 months)
  - Tank Size – .1 Gallon (user must accurately input tank size)
  - Wheel Size – 1/8” (user must accurately input wheel size)
  - Time – 5 minutes (over 6 months)
  - Fuel - .001 Gal

- **Range**
  - Speed 0 – 80 MPH
  - Range 0 – 999 Miles
  - Gas Mileage 0 – 99 MPG
  - Distance 0 – 9999.9 Miles
  - Tank Size 0 – 9.99 Gallons
  - Wheel Size – 2 -2.75 ft.
  - Trip Speed 0 – 99 MPH
  - Trip Time 0 – 99:59 hours
  - Time – 12 Hours AM; 12 Hours PM
  - Fuel – 0 – 9.999 Gallons

- **Resolution**
  - Speed 1 MPH
  - Range 1 Miles
  - Gas Mileage 1 MPG
  - Distance 0 – .1 Miles
  - Tank Size 0 – .01 Gallons
  - Wheel Size .01 ft.
  - Trip Speed 1 MPH
  - Trip Time – 1 minute
  - Time – 1 minute
  - Fuel – .001 Gallons

- **Wheel Speed Sampling Rate**
  - 1 – 20 Hz

- **Power Requirements**
  - Source – 12 Volt Motorcycle battery
  - Worst case Power Dissipation 562 mW

- **Environmental Considerations**
  - Operating Temp – 0 to 50 degrees C
  - Shock Proof case (final product)
  - Water Proof case (final product)
  - Vibration Proof
  - Sensor Assembly – Dirt Proof (final product)
• Size Limits
  • Main board 3” x 5” x 2”
  • Sensor Board 1” x 1”

**Preliminary Parts List:**

<table>
<thead>
<tr>
<th>Part</th>
<th>Part #</th>
<th>Manufacturer</th>
<th>Vendor</th>
<th>Current</th>
<th>Price</th>
<th>Lead Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall Sensor</td>
<td>A3249E</td>
<td>Allegro</td>
<td>Newark</td>
<td>8 mA</td>
<td>$0.90</td>
<td>1 Month</td>
</tr>
<tr>
<td>Regulator</td>
<td>LM1085</td>
<td>National</td>
<td>Digi-Key</td>
<td>6 mA</td>
<td>$0.61</td>
<td>3 Weeks</td>
</tr>
<tr>
<td>Clock</td>
<td>DS1302</td>
<td>Dallas</td>
<td>Digi-Key</td>
<td>0.45 mA</td>
<td>$1.66</td>
<td>1 Week</td>
</tr>
<tr>
<td>Crystal (RTC)</td>
<td>CFS206</td>
<td>Citizen</td>
<td>Digi-Key</td>
<td>NA</td>
<td>$0.24</td>
<td>1 Week</td>
</tr>
<tr>
<td>Crystal (MCU)</td>
<td>CSA309</td>
<td>Citizen</td>
<td>Digi-Key</td>
<td>NA</td>
<td>$0.65</td>
<td>2 Week</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>M68HC912B32</td>
<td>Motorola</td>
<td>Future Elect.</td>
<td>45 mA</td>
<td>$16.00</td>
<td>2 Weeks</td>
</tr>
<tr>
<td>Resistors</td>
<td>various</td>
<td>Mouser</td>
<td>NA</td>
<td>50 mA</td>
<td>$0.15</td>
<td>1 Week</td>
</tr>
<tr>
<td>Capacitors</td>
<td>various</td>
<td>Mouser</td>
<td>NA</td>
<td>NA</td>
<td>$1.00</td>
<td>1 Week</td>
</tr>
<tr>
<td>LCD</td>
<td>LCM-S02004</td>
<td>Lumex</td>
<td>Digi-Key</td>
<td>3 mA</td>
<td>$17.98</td>
<td>1 Week</td>
</tr>
<tr>
<td>Pushbuttons</td>
<td>8161</td>
<td>ITT</td>
<td>Mouser</td>
<td>NA</td>
<td>$4.03</td>
<td>1 Week</td>
</tr>
<tr>
<td>Magnet</td>
<td>R012</td>
<td>Magnet Factory</td>
<td>NA</td>
<td>NA</td>
<td>$0.57</td>
<td>1 Week</td>
</tr>
<tr>
<td>Case</td>
<td>K-HPL</td>
<td>Pac-Tec</td>
<td>Mouser</td>
<td>NA</td>
<td>$8.54</td>
<td>1 Week</td>
</tr>
<tr>
<td>PC Boards</td>
<td>574-169P44</td>
<td>Vector</td>
<td>Mouser</td>
<td>NA</td>
<td>$8.32</td>
<td>1 Week</td>
</tr>
<tr>
<td>Low V. Detect</td>
<td>MAX6314</td>
<td>Maxim</td>
<td>Digi-Key</td>
<td>0.005 mA</td>
<td>$0.99</td>
<td>1 Week</td>
</tr>
</tbody>
</table>

**Totals:**

112.45 mA $61.64