

BICYCLE COMPUTER USING THE MC146805G2()1 MICROCOMPUTER

INTRODUCTION

The MC146805G2()1 (MC146805G2L1 or MC146805G2P1) is a fully static single-chip CMOS microcomputer. It has 112 bytes of RAM, 2106 bytes of user ROM, four 8-bit input/output ports, a timer, and an on-chip oscillator. The MC146805G2L1 ROM contains three distinct routines: self-check, monitor, and bicycle computer.

The self-check routine is included in all MC146805G2 MCU devices. The self-check feature is fully described in the MC146805G2()1 data sheet and it can be used to verify operation of the MCU.

The monitor routine, which is included in all MC146805G2()1 MCUs, is fully described in application note AN-852. The monitor allows the user to evaluate the MCU using a standard RS-232 terminal interface.

The bicycle computer routine is included in all MC146805G2()1 MCUs and provides a realistic application of the MCU. It is this application that is described here. It is intended only as an example of the type of application that can be accomplished with only 1300 of the 2100 bytes available on the MC146805G2. Furthermore, the bicycle computer is used for test sampling the part. A copy of the bicycle computer program listing is included in this application note.

BICYCLE COMPUTER FEATURES

In the configuration shown in Figure 1, the MC146805G2L1 can be used for a bicycle computer. Features provided by the bicycle computer include: (1) instantaneous speed, (2) average speed, (3) resettable trip odometer, (4) resettable long distance odometer, (5) cadence

(pedal crank revolutions per minute), (6) selection of English or metric units, and (7) calibration for wheel size.

HARDWARE CONFIGURATION

A schematic diagram for the bicycle computer is shown in Figure 1 and Figure 2 shows a parts layout diagram. As shown on the schematic diagram, the MC146805G2()1 and the liquid-crystal display (LCD) are the only major components required for the bicycle computer. All necessary drive signals for the LCD are contained in firmware. Two pushbutton switches (function and set) are required to furnish two momentary ground inputs, and two sensor inputs (one from the wheel and one from the pedal crank) are required as an interrupt and to pulse certain counters. Each sensor is a normally open switch which is activated by a magnet mounted on the wheel and pedal crank.

Figure 2 shows the layout of a PCB that may be used when assembling the bicycle computer. The printed circuit board (PCB) is designed to fit in a Wonder-Lite case. The Wonder-Lite is designed to mount on a bicycle and provides nighttime illumination. Dimensions for this board are 4.5 inches x 2.5 inches and could require some tailoring before fitting into the mounting case. However, an equivalent size wire-wrap type board, using wire-wrap connections and mounting sockets, could be used with an equivalently sized case (not a Wonder-Lite case).

The plastic which was previously used to close off the Wonder-Lite lens opening must be cemented in place. An epoxy type glue can be used. This plastic window can now be used for viewing the LCD. The lens area and connector entry

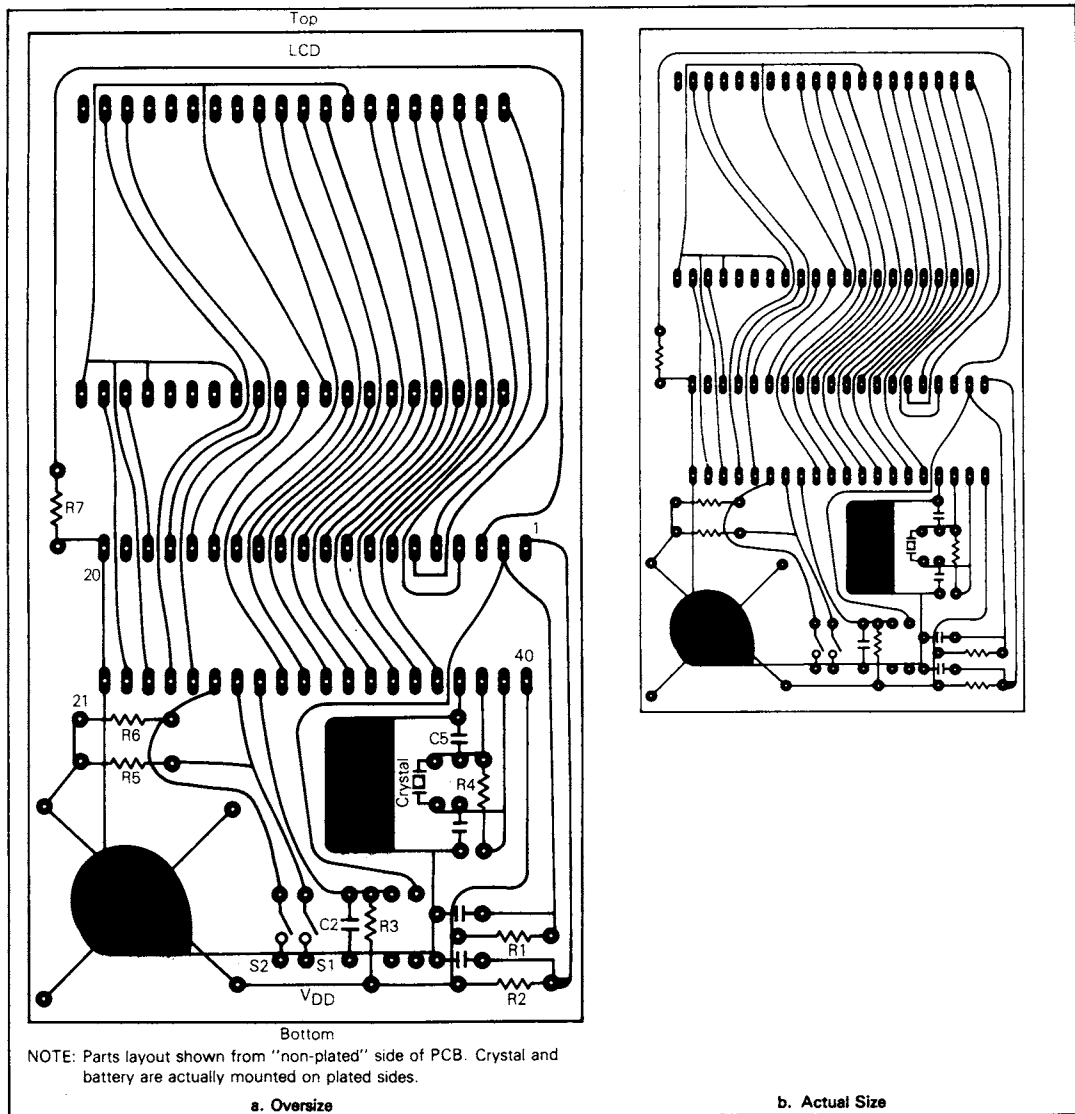


FIGURE 2 — Bicycle Computer Parts Layout Diagram

hole can be reasonably weather proofed by using a silicone rubber sealant around the edges.

Once the connector and two pushbutton switches are mounted, insert the PCB into the portion of the case that contains the viewing window. If the PCB requires tailoring in order to fit into the case, be sure to remove equal amounts from each side or each end of the PCB. To accomplish this, lay a flat file on a level surface and slide the printed circuit board edge over the file. Do not clamp the PCB and run the

file over it. Also, to prevent the tab that opens the Wonder-Lite case from interfering with the PCB, a notch should be cut in the top of the PCB where the tab is located.

Once the PCB fits into the case, components can be mounted onto the board. All components except the crystal and battery should be mounted on the side of the board that is not printed. A suggested method is to mount the resistors first followed by the capacitors, making sure that they lay as

flat as possible. This ensures that the height of these components does not interfere with the PCB mounting. Next, the crystal should be mounted on the printed side of the PCB. Before mounting the crystal, enclose the outside of the crystal case with transparent tape; however, leave an untaped area on one side of the crystal case to allow for soldering a ground wire.

NOTE

The transparent tape ensures that the crystal case does not short circuit any of the PCB plating.

Solder one end of a three-inch wire to the plated ground plane area where the crystal is to be mounted. Mount the crystal so that its fully taped side is flat against the PCB. The crystal leads can now be bent downward and soldered to the printed wiring which connects to pins 38 and 39. Solder the other end of the three-inch wire (one end soldered to ground plane) to the area left untaped on the crystal case. Be sure and trim this wire and make it taut along the case of the crystal. (Grounding the crystal case in this manner ensures that the crystal will provide oscillations at its fundamental frequency.)

Mount the LCD and then the MCU. Because of space limitations, do not use a mounting socket for the LCD or MCU when using the Wonder-Lite case.

The last component to be mounted is the battery. This battery could be a 3-volt, flat button, lithium battery; however, up to a 6-volt, flat button battery could be used provided it is small enough to allow for case closure. Mount the lithium, flat button battery as follows:

- (1) On the printed wiring side of the PCB, solder one end of a 1½-inch, noninsulated wire to the printed wiring which connects to R5 and R6.
- (2) Lay the - (negative) side of the button type battery on the large ground plane area shown in Figure 2.
- (3) Pass the 1½-inch wire, connected in (1) above, over the battery + (positive) side and solder it to the VDD bus.
- (4) Solder the uninsulated wire to the + side of the battery.
- (5) To hold the battery more securely, place an insulated, 3½-inch wire across the battery, perpendicular to the first one, and pass the ends of this wire through the two holes provided. Strip about 3/8-inch of insulation from each end of the insulated wire. Twist the two ends together until the insulated wire is reasonably tight against the battery case. Solder the twisted ends to prevent unraveling and then trim them if necessary.

NOTE

The overall effect is that the button battery is held in place with "criss-crossed" wires.

Once the battery is connected, the LCD should indicate "0" in the least significant digit; however, if it does not, momentarily short VDD to ground. This activates the power-reset circuitry and the zero should appear in about five seconds.

To mount the two pushbutton switches, two holes must be drilled in the case. A convenient point is the lower half side of the case which is opposite to the hinges. The slot left by the Wonder-Lite on-off switch should be closed using a piece of plastic (cut to size). The connector for the sensor wires

(wheel and pedal crank switches) can be mounted at the bottom of the case after drilling appropriate mounting holes. Some type of quick-disconnect connector (such as a Molex or telephone module connector) should be used to allow for easy removal of the bicycle computer. A putty-like waterproof sealant may be used to seal the two switches, the on-off switch cover, and the quick disconnect connector.

Once the switches and connector are mounted in the case, the necessary sensor and switch wires can then be soldered to the PCB. These solder points are shown as S1, S2, CAD (cadence sensor), and WHL (wheel sensor).

Once the wiring is complete and the PCB is in place, a type of nonconducting stuffing material should be inserted inside the case to prevent movement of the PCB. When the case is closed, the bicycle computer is complete except for mounting the sensors.

The wheel and cadence sensors each consist of a switch (reed or mercury film) which is activated by a permanent magnet. The wheel sensor switch should be mounted on the front wheel fork and the magnet should be attached to the wheel spokes as near the hub as possible. It is advisable to attach the magnet on the hub side which is opposite to the valve stem side in order to balance the wheel. The sensor switch can be mounted on the front fork so that the magnet passes within 1/4 to 3/8 inch in order to activate the sensor. Do not mount the switch and magnet so close that they touch under adverse (bumpy) conditions. The final placement will probably result after several tries. An epoxy glue can be used to hold the magnet to the spokes and the switch can be held to the fork for many months with duct tape. (If the magnet has a hole in the center, a small bolt may also be used to attach the magnet where two spokes cross; however, epoxy glue should still be used.) Wires from the wheel sensor switch can be routed via the fork to the mating connector. Be sure that the wheel sensor input is applied to the IRQ interrupt pin of the MC146805G2()1. By using one of the recommended sensor switches and the rc circuit, the remaining debounce problems are solved in the firmware.

The cadence sensor input is similar to the wheel sensor; however, mounting is done at the pedal or crank arm. The magnet can be attached to either pedal crank arm and the sensor switch attached to the frame. Again, experiment with the distance between the sensor and switch; however, try and mount the sensor so that the switch and magnet are not aligned when not pedaling. If they are aligned at this time, bicycle movement might cause the switch to open and close many times, producing very high cadence values. Since most people stop pedaling at 0, 90, 180, or 270 degrees in the pedal arc, it is best to mount the sensor so that it is activated at some point between these. Again, as in the wheel sensor, route the wires along the frame to the mating connector.

BICYCLE COMPUTER OPERATION

When power is initially applied to the circuit or when the MC146805G2()1 is reset, the bicycle computer program is selected and the bicycle computer displays the current instantaneous speed on the display (function 1). If the function button (S1) is then pushed, the bicycle computer will step to function 2. The functions are:

1. Instantaneous Speed
2. Average Speed
3. Resettable Trip Odometer
4. Resettable Long Distance Odometer
5. Cadence
6. English or Metric Units Selection
7. Wheel Size Calibration

Each time the function switch is pushed, the program steps to the next function; however, after function 7 it goes back to function 1. Some functions may require resetting. For example, at the beginning of each bicycle trip it may be desirable to reset the trip odometer to zero miles or kilometers. The set pushbutton (S2) is provided to perform this task. If the set button is pushed while in function 3, the trip odometer is reset to zero. However, it is not desirable to have the set button enabled at all times. For example, if the set button were accidentally pushed during a trip, the trip odometer would be reset to zero. Therefore, the set button is only enabled for the first five seconds after a new function is selected. Pushing the set button after five seconds will not affect the function. During the five seconds that the set button is enabled, the bicycle computer displays a fixed function identification display. For example, the trip odometer will display **odo** during this function 3 time. After five seconds, the selected function value is displayed and remains displayed until the function button is again pushed, stepping to the next function.

The following paragraphs will discuss each function in detail. They describe what will be displayed during the five second set enable period for each function and what effect, if any, pushing the set button will have when it is enabled.

Instantaneous Speed

When selected, the instantaneous speed function displays the current speed to the nearest mile or kilometer per hour. The speed is determined by measuring the period between wheel interrupts. By measuring this period and knowing the wheel circumference, the speed can be determined. During the set enable period, **SP** will be displayed. Depressing the set button during this period has no effect on instantaneous speed.

Average Speed

When selected, the average speed function is displayed to the nearest mile or kilometer per hour. Average speed is calculated by dividing the distance traveled by the time. During the set enable period, **ASP** is displayed. Depressing the set button during this period resets the average speed, distance, and time counters. Note that these counters are independent of the other distance and speed counters; therefore, resetting the average speed does not affect other functions such as the trip and long distance odometers.

In order to provide an accurate average speed over a long time interval, the average speed is biased toward long term (greater than 6 minutes) operation. In fact, the average speed is not guaranteed accurate until after 6 minutes but will remain accurate for several days and up to 409.6 total miles or kilometers. The average speed display is updated every 22.5 seconds.

Resettable Trip Odometer

When this function is selected, the distance since the last trip odometer reset (or power-on reset) is displayed to the nearest tenth of a mile or kilometer. The trip distance is calculated by counting the wheel interrupts and multiplying by the wheel circumference to obtain the distance. During this set enable period **odo** is displayed. Depressing the set button during this period resets the trip odometer to zero. Maximum trip distance is 199.9 miles or kilometers. If this distance is exceeded, the display wraps around to 0.0 and continues. Note that resetting the trip odometer has no effect on the long distance odometer or average speed.

Resettable Long Distance Odometer

When this function is selected, the distance since the last long distance odometer reset (or power-on reset) is displayed to the nearest mile or kilometer. The total distance is calculated by counting wheel interrupts and multiplying by the wheel circumference. During this set enable period, **dis** is displayed. Depressing the set button during this period resets the long distance odometer to zero. Maximum long distance is 1999 miles or kilometers. (Even though the display is limited to 1999, the internal counters will roll over much like a mechanical car odometer.) Resetting the long distance odometer has no effect on the trip odometer or average speed.

Instantaneous Cadence

Cadence is the number of times the pedal crank undergoes a complete revolution in one minute. An experienced bicyclist tries to keep his cadence constant at all times by adjusting his gears for varying conditions such as hills. An occasional cyclist might have an ideal cadence around 50-60, an experienced cyclist 70-80, and a racer up to 120. When this function is selected, the instantaneous cadence is displayed. The cadence is calculated by measuring the period between successive switch closures from the cadence sensor.

Since the cadence can be quite low (or even zero if coasting), the period between two cadence closures is highly variable. For this reason, the program will zero the cadence display if no closure occurs after 6 seconds. This means either the cadence is less than 10 rpm or the rider has stopped pedaling. Also, this means that if you are pedaling at a fast rate and stop instantly, it will be six seconds before the cadence returns to zero. The cadence display may also display odd values if the cyclist happens to stop pedaling while the magnet is directly over the switch. This might cause the switch to open and close many times, producing very high cadence values. Since most people stop pedaling at 0, 90, 180, or 270 degrees in the pedal arc, this problem might be minimized by placing the switch at some other angle.

During this set enable period, **Cad** is displayed. Depressing the set button during this period has no effect on the instantaneous cadence function.

English or Metric Units Selection

When selected, this function displays 1888 (all segments on) and will also display an arrow if the current units are metric. If the current units are English, then there is no arrow on the display. Depressing the set button while set is enabled causes the units to be changed (i.e., if they were English they will change to metric and vice versa). Changing the units will cause future instantaneous and average speeds to be displayed in the new units as well as the trip and long distance odometers. The units can be changed at any time by this function without destroying any distances or speeds. This is possible because the program actually updates all functions in both units at all times.

Wheel Size Calibration

When this function is selected, the current wheel circumference, to the nearest 1/2 inch, is displayed. This circumference is critical in the calculation of speed and distance. Setting a new wheel size is somewhat more complex than other set procedures. If the set button is pushed while this function is enabled, then a 39.5 inch wheel circumference appears on the display. The program then goes into a special mode where it increments this wheel size by 1/2 inch about

every second. If the set button is pushed the second time while in this incrementing mode, the value currently being displayed becomes the new wheel circumference. If the set button is not pushed a second time before the function button is pushed again, the original wheel circumference will be reestablished as the active circumference. Valid circumferences are between 39.5 and 99.5 and are always in inches regardless of the units (English or metric) that are in effect. If the circumference reaches 99.5 before the set button is pushed the second time, the display will wrap around to 39.5 and continue. See the section on Accuracy for hints on how to measure your wheel circumference.

DEFAULT VALUES

When the MC146805G2()1 is reset or after a power-on reset, the program clears all distance and average speed values and picks defaults for the wheel circumference and units. They are:

1. Wheel Circumference = 84.5 inches
2. English Units

The circumference of 84.5 inches is the average for the standard 27 inch front bicycle wheel found on most ten speed bicycles in the United States. Some European and Japanese models use 700 cm wheels which have an average circumference of about 83.0 inches. If you have 700 cm wheels or if you plan to use the bicycle computer on other size wheels, such as children's bicycles, then you will have to reset your wheel circumference every time the MC146805G2()1 is reset. This usually occurs only when a new battery is installed.

POWER SAVING FEATURES

The schematic diagram of Figure 1 indicates that the bicycle computer has no on-off switch. This is because the program on the MC146805G2()1 controls its power consumption by placing the MC146805G2()1 into either the wait or stop state when appropriate. The wait state places the device into a low power mode where all processing stops. Only the on-chip timer and oscillator remain active. In the stop state, even the timer and oscillator are stopped; however, the on-chip RAM remains valid. If an $\overline{\text{IRQ}}$ interrupt occurs ($\overline{\text{IRQ}}$ is connected to the wheel switch), the MC146805G2()1 will restart execution. For more details on the wait and stop states, refer to the MC146805G2 data sheet and the "M6805 HMOS/M146805 CMOS Family Users Manual."

The bicycle computer firmware takes advantage of these built-in M146805 CMOS features by considering the program to be in 1 of 3 modes. These are:

1. Normal Operating Mode
2. Standby Mode
3. Stop Mode

Normal operating mode is defined as the mode when the bicycle computer is performing its normal functions. The program enters into normal mode on power-on reset and remains in this mode as long as there are wheel interrupts, cadence inputs, or either pushbutton is depressed. While in the normal mode, the program can be considered as executing a large loop. The length of time necessary to complete the loop is variable depending on the rate of the inputs, the current function, etc. When the program completes the loop, it goes into the wait state until an on-chip timer interrupt occurs. The timer is set up to generate an interrupt every millisecond. Our measurements have shown that in practice the MC146805G2()1 spends about 30% of its time in the wait state during normal operation.

If the program senses no activity on the wheel interrupt input, the cadence input, or either pushbutton for 2.5 minutes, it goes into the standby mode. In this mode, the display is blanked and the internal processing necessary to calculate values for the displays ceases. In this mode, the MC146805G2()1 spends about 70% of its time in the wait state.

Note that while the program is in the standby mode, all necessary internal calculations are continued. If a wheel interrupt or cadence input occurs or if either button is pushed, the program immediately returns to the normal mode as if it had never been in standby.

If the program remains in standby mode for two hours it will then go into stop mode. In stop mode all processing ceases. All values are maintained in RAM. However, since the timer stops, it is impossible to keep track of time. Therefore, the average speed value will be incorrect when the program returns to normal mode. The program will return to normal mode only after receiving a wheel interrupt ($\overline{\text{IRQ}}$). Instantaneous speed, trip distance, long distance, and cadence will all be correct when returning to normal mode. The selected units and wheel circumference will also be returned to the value they had before going into standby mode.

To summarize, when the MC146805G2() initially comes out of reset, the program goes into the normal mode. If at any time there is no activity on any of the input lines for 2½ minutes, then the program enters standby mode. Any input will put the program back into normal mode, but if no input occurs for two hours, then stop mode will be entered. This would most likely occur after the completion of the trip or during an overnight stop. The next time a wheel interrupt occurs, the program will return to normal mode. Only the average speed counters will have to be reset. Two hours was selected as the time out for entering the stop mode because breaks or lunches during a trip normally would not last over two hours.

ACCURACY

The internal inaccuracy of the bicycle computer program is less than 0.5% for most functions. Any other inaccuracies stem from the inaccuracy of the wheel circumference. Therefore, it is important to measure your wheel circumference as accurately as possible to the nearest ½ inch.

This can easily be done by first drawing a straight line, on a garage floor or sidewalk, approximately 90 inches (2.3 meters) long for an adult ten speed bicycle. Next, place the front wheel of the upright bicycle on one end of the line. Make a mark on the tire and on the line at the point where the tire contacts the line. Roll the bicycle and tire along the line one revolution and make a mark on the line. Then, measure the distance between the two marks on the line; this is the circumference. Round this to the nearest ½ inch and if it is not equal to 84.5 inches (the default), input the new wheel circumference as described above under Wheel Size Calibration. If you wish to be very accurate, you might sit on the bicycle while rolling down the line. This will take into consideration any variation in the circumference due to deformation of the tire under load. No matter how careful you are, the inputted circumference might be as much as ¼ inch from the actual circumference, adding additional inaccuracies. Our calculations show that the total of all possible inaccuracies should remain less than 1%. This is considerably more accurate than the average automobile odometer.

ALTERNATE APPLICATIONS FOR THE BICYCLE COMPUTER

Although we have never tried it, we feel certain that this bicycle computer could be used for other types of vehicles. Car wheels, for example, fall within the 39.5 to 99.5 inch wheel circumference and the program has enough accuracy and throughput to handle speeds up to 255 miles or kilometers per hour. It has also been suggested that the cadence input could be used to display engine rpm. Since the maximum displayable cadence value is 255, it would be necessary to divide the engine rpm by 100 to obtain rpm/100. Actually the measured rpm would be from 1000 to 25,500 since any cadence below 10 is assumed to be 0.

When using the computer in alternate applications, only

two things must be remembered. First, the wheel input is connected to the $\overline{\text{IRQ}}$ interrupt input. For this reason it must be debounced. An rc time constant of 10 milliseconds will suffice. Since the wheel input is a low-going edge-sensitive interrupt, the period of a wheel revolution is measured from successive falling edges of the $\overline{\text{IRQ}}$ interrupt pin.

Secondly, the cadence input is not an interrupt, but is debounced by software. The software debounce time is three milliseconds. If the cadence input does not remain low for three milliseconds, the input will not be recognized. However, if an invalid input results in a low input which lasts longer than three milliseconds, it will be mistaken for a valid input. The cadence period is measured between successive falling edges of the cadence input line.

Parts List for Bicycle Computer

Battery 3.5 V		
C1, C2	0.01 μF	10 V
C3	0.1 μF	10 V
C3, C4	30 pF	
Crystal	1 MHz	
Display, Liquid-Crystal		FE0201 AND Company or Similar
IC1	MC146805G2L1/P1	Motorola
Magnets		2 Reasonably Powerful
Switches		2 spst Push-Button Switches Normally Open
Reed Switch		2 Miniature Reed Switches, Radio Shack or 2 Mercury Film Switches (Fifth Dimension, Inc. 801 New York Avenue, Trenton, NJ 08638 609-393-8350) or Similar
R1-R3	1 M Ω	$\frac{1}{4}$ Watt
R4-R6	10 M Ω	$\frac{1}{4}$ Watt
R7	10 k Ω	$\frac{1}{4}$ Watt

TABLE 1 — Summary of Operations

Function	Normal Display	Units	Set Enable Display	Effect of Set Push
Instantaneous Speed	XXX	mph/kph	SP	No Effect
Average Speed	XXX	mph/kph	ASP	Reset Time and Distance Counters
Trip Odometer	XXX.X	mi./km		Reset Trip Mileage
Long Distance Odometer	XXXX	mi./km	IS	Reset Long Distance Mileage
Instantaneous Cadence	XXX	rev./min.	CA	No Effect
Unit Select	-1888*	metric/Eng.	-1888*	Switch Units
Wheel Size	XX.X	inches	XX.X	First Push Starts Recalibration Second Push Locks In New Value

* Arrow only set if units are metric.

TTL MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982
OPT CMOS, LLE=95, ABS, S, CRE

MC146805G2 ROM PATTERN

THE MC6805G2 SINGLE-CHIP MICROCOMPUTER IS A 40-PIN CMOS DEVICE WITH 2096 BYTES OF ROM, 112 BYTES OF RAM, FOUR 8-BIT I/O PORTS, A TIMER AND AN EXTERNAL INTERRUPT INPUT. THE ROM CONTAINS TWO SEPARATE PROGRAMS. EITHER OF THESE PROGRAMS MAY BE SELECTED ON RESET BY WIRING PORT C AS FOLLOWS:

C7	C1	C0	FUNCTION
1	0	0	MONITOR (300 BAUD)
1	0	1	MONITOR (1200 BAUD)
1	1	0	MONITOR (4800 BAUD)
1	1	1	MONITOR (9600 BAUD)
0	X	X	BICYCLE ODOMETER

THE MONITOR IS SUBSTANTIALLY THE SAME AS ALL PREVIOUS MONITORS FOR THE 6805. THE MONITOR USES SERIAL I/O FOR ITS COMMUNICATION WITH THE OPERATOR. SERIAL INPUT IS C2 AND SERIAL OUTPUT IS C3.

I/O REGISTER ADDRESSES

ADDRESS	NAME	VALUE	DESCRIPTION
0000	A PORTA EQU	\$000	I/O PORT 0
0001	A PORTB EQU	\$001	I/O PORT 1
0002	A PORTC EQU	\$002	I/O PORT 2
0003	A PORTD EQU	\$003	I/O PORT 3
0004	A DOR EQU	4	DATA DIRECTION REGISTER OFFSET (E.G. PORTA+DOR)
0006	A TIMER EQU	\$008	8-BIT TIMER REGISTER
0007	A TCR EQU	\$009	TIMER CONTROL REGISTER
0010	A RAM EQU	\$010	START OF ON-CHIP RAM
0008	A ZROM EQU	\$080	START OF PAGE ZERO ROM
0100	A ROM EQU	\$100	START OF MAIN ROM
2000	A MEMSIZ EQU	\$2000	MEMORY ADDRESS SPACE SIZE

CHARACTER CONSTANTS

ADDRESS	NAME	VALUE	DESCRIPTION
000D	A CR EQU	\$0D	CARRIAGE RETURN
000A	A LF EQU	\$0A	LINE FEED
0020	A BL EQU	\$20	BLANK
0000	A EOS EQU	\$00	END OF STRING

BICYCLE ODOMETER

TEST PATTERN FOR THE MC146805G2

COPYRIGHT (C) MOTOROLA INC. 1980

SOFTWARE WRITTEN BY JOEL BONEY SEPT. 1980
HARDWARE BY VERNON GOLER

THE SOFTWARE FOR THE BICYCLE ODOMETER IS DRIVEN BY A SEQUENCE OF EVENTS. SOME EVENTS ARE INTERRUPTS, SOME ARE INPUTS AND SOME ARE SCHEDULED BY AN INTERRUPT OR INPUT. THERE IS A SEPARATE SOFTWARE TASK (ROUTINE) FOR EACH EVENT. ADDITIONALLY THERE IS A TASK CALLED THE MAIN TASK (OR LOOP) THAT RUNS CONSTANTLY AT THE LOWEST PRIORITY. ADVANTAGE IS MADE OF THE 146805'S 'WAIT' INSTRUCTION WHENEVER POSSIBLE TO REDUCE TOTAL POWER CONSUMPTION. THE MAIN ROUTINES OR TASKS ARE:

NAME	VALUE
MAIN LOOP	
1 MSEC INTERRUPT	
WHEEL INTERRUPT	
30 MSEC ROUTINE	
600 MSEC ROUTINE	

EQUATES AND VARIABLES

EQUATES

ADDRESS	NAME	VALUE
0001	A BIT0 EQU	1
0002	A BIT1 EQU	2
0004	A BIT2 EQU	4
0008	A BIT3 EQU	8
0010	A BIT4 EQU	16
0020	A BIT5 EQU	32
0040	A BIT6 EQU	64
0080	A BIT7 EQU	128

EQUATES FOR STATUS WORD 1

ADDRESS	NAME	VALUE
0007	A STNDBY EQU	7
0006	A DECPY EQU	6
0005	A FUNDNM EQU	5
00100	A WFUN EQU	4
0003	A SETDWN EQU	3
0002	A WTSET EQU	2
0001	A SETENB EQU	1
0000	A STOPIT EQU	0

EQUATES FOR STATUS WORD 2

ADDRESS	NAME	VALUE
0007	A LDZERO EQU	7


```

00117 0006 A KMHILF EQU 6
00118 0005 A ROST30 EQU 5
00119 0004 A WTSET2 EQU 4
00120 0003 A ROST60 EQU 3
00121 0002 A CADDOWN EQU 2
00122 0001 A WTCAD EQU 1
00123 0000 A DOST EQU 0
00124
00125 * I / O EQUATES
00126
00127 * THE LCD DIGITS ARE NUMERED FROM THE HUNDREDTH'S DIGIT (1) TO ONE'S
00128 * FOR EXAMPLE G3 IS THE G SEGMENT IN THE LS (ONE'S) DIGIT.
00129 * THE THOUSANDTH'S DIGIT IS CALLED 'K'.
00130
00131
00132 * THE I/O PORTS SEEM TO BE NEEDLESSLY SCRAMBLED. TRUE, THEY ARE
00133 * SCRAMBLED, BUT THE REASON IS THIS SCRAMBLING ALLOWS THE
00134 * BIKE ODO TO BE LAID OUT ON A SINGLE SIDED PC BOARD.
00135
00136 * PORTA:
00137 * 7
00138 * -----
00139 * | G3 | F3 | A3 | B3 | C3 | D3 | E3 | DP4 |
00140 * -----
00141
00142 *
00143 * PORTB:
00144 * 7
00145 * -----
00146 * | UN | K | E1 | D1 | C1 | E2 | D2 | C2 |
00147 * -----
00148
00149 * UN = UNDEFINED
00150 * K = THOUSANDTHS DIGIT
00151 0007 A UN EQU 7
00152 0006 A K EQU 6
00153
00154 * PORTC:
00155 * 7
00156 * -----
00157 * | SEL | YY | COM | XX | W | RST | FUN | CAD |
00158 * -----
00159 * $ $ $ $ $ $ $ $ = INPUT PIN
00160
00161 * SEL = SELECTS BETWEEN BIKE ODO AND MONITOR (0 = BIKE ODO)
00162 * YY = MINUS SIGN
00163 * COM = BACKPLANE VOLTAGE
00164 * XX = TOP AND BOTTOM OF '+'
00165 * W = ARROW (SET FOR METRIC UNITS)
00166 * RST = RESET KEY INPUT
00167 * FUN = FUNCTION KEY INPUT
00168 * CAD = CADENCE INPUT
00169
00170 0007 A SEL EQU 7
00171 0006 A YY EQU 6
00172 0005 A COM EQU 5
00173 0004 A XX EQU 4
00174 0003 A W EQU 3

```

```

A
+---+
F |
+---+
G
+---+
E |
+---+
D

```

```

00175 0002 A RSTKEY EQU 2
00176 0001 A FNKEY EQU 1
00177 0000 A CAD EQU 0
00178
00179 * PORTD:
00180 * 7
00181 * -----
00182 * | B2 | A2 | F2 | G2 | B1 | A1 | F1 | G1 |
00183 * -----
00184
00185 *
00186 * EQUATES FOR ABOVE
00187
00188 * ALIAS FOR PORTC (PORT CONTAINING INPUTS)
00189 0002 A IOX EQU PORTC
00190
00191 *
00192 * FUNCTION NUMBER EQUATES
00193
00194
00195 0000 A SP EQU 0
00196 0001 A ASP EQU 1
00197 0002 A TRIP EQU 2
00198 0003 A DIS EQU 3
00199 0004 A CADEN EQU 4
00200 0005 A KMH1 EQU 5
00201 0006 A WHLSIZ EQU 6
00202
00203 0006 A MAXFUN EQU 6 MAXIMUM FUNCTION NUMBER
00204
00205 *
00206 *
00207 * R A M V A R I A B L E S
00208 *
00209 *
00210 *
00211 *
00212 * MUCH OF THE ALGORITHM FOR THIS BICYCLE ODOMETER IS GIVEN HERE IN
00213 * THE EXPLANATION OF THE VARIABLES. HOW THE VARIABLES ARE SET, CHANGE
00214 * AND MANIPULATED TELLS MUCH OF WHAT IS GOING ON. THEREFORE, THERE IS
00215 * DETAILED INFORMATION ABOUT EACH VARIABLE GIVEN HERE.
00216
00217
00218A 0010 ORG $10 START OF RAM
00219
00220 *
00221 * NOTE: EVERYTHING FROM HERE DOWNT0 'FUNCT' SHOULD NOT BE SEPARATED
00222 * BECAUSE THE INITIALIZATION ROUTINE DOES A BLOCK CLEAR
00223 * OF THIS AREA.
00224 *
00225 *
00226 * STATUS BYTE 1
00227
00228 * THE BITS IN THIS STATUS BYTE INDICATE THE STATE OF THE PROGRAM
00229
00230 *
00231 * BIT 7 STANDBY MODE (STNDRY)
00232 * BIT 6 DECIMAL POINT (DECP1)
00233 * BIT 5 FUNCTION KEY DOWN (FUNDOWN)

```

```

00233      *      BIT 4   WAITING FOR FUNCTION KEY UP (WTFUN)
00234      *      BIT 3   RESET KEY DOWN (SETDWN)
00235      *      BIT 2   WAITING FOR RESET KEY UP (WTSET)
00236      *      BIT 1   "RESET" ENABLED MODE (SETEND)
00237      *      BIT 0   STOP MODE (STOPIT)
00238
00239A 0010 0001 A STAT1 RMB 1
00240      *
00241      * STATUS BYTE 2
00242
00243      * THE BITS IN THIS BYTE INDICATE ADDITIONAL PROGRAM STATE.
00244
00245      *      BIT 7   1 = PAST LEADING ZEROS IN DISPLAY OUTPUT (LDZERO)
00246      *      BIT 6   KM MODE (KMHILE)
00247      *      BIT 5   30 MSEC REQUEST (RQST30)
00248      *      BIT 4   WAITING FOR 2ND RESET IN WHEEL SIZE ENTRY (WTSET2)
00249      *      BIT 3   600 MSEC REQUEST (RQS600)
00250      *      BIT 2   CADENCE INPUT PRESENT (CADDWN)
00251      *      BIT 1   WAITING FOR CADENCE TO CLEAR (WTCAD)
00252      *      BIT 0   DO FUNCTION RESET (DOSET)
00253
00254A 0011 0001 A STAT2 RMB 1
00255      *
00256      * TRIP MILES COUNTER
00257
00258      *      UPDATED WHEN MILES REV. CTR (MRCTR) UNDERFLOWS
00259
00260A 0012 0002 A TMCTR RMB 2
00261      *
00262      * TRIP KILOMETER COUNTER
00263
00264      *      UPDATED WHEN KM REV. CTR. (KRCTR) UNDERFLOWS
00265
00266A 0014 0002 A TKCTR RMB 2
00267      *
00268      * TOTAL MILES COUNTER
00269
00270      *      UPDATED WHEN MILES UPDATE CTR (MUCTR) UNDERFLOWS
00271
00272A 0016 0002 A TOTMI RMB 2
00273      *
00274      * TOTAL KMS COUNTER
00275
00276      *      UPDATED WHEN KM UPDATE CTR (KUCTR) UNDERFLOWS
00277
00278A 0018 0002 A TOTKM RMB 2
00279      *
00280      * AVERAGE SPEED DISTANCE - MILES
00281
00282      *      CONTAINS .1MI*16
00283      *      UPDATED BY MILES REV (MRCTR) UNDERFLOW
00284
00285A 001A 0002 A ASDMI RMB 2
00286      *
00287      * AVERAGE SPEED DISTANCE - KILOMETERS
00288
00289      *      CONTAINS .1KM*16
00290      *      UPDATED BY MILES REV (KRCTR) UNDERFLOW

```

```

00291
00292A 001C 0002 A ASDKM RMB 2
00293      *
00294      * AVERAGE SPEED TIME
00295
00296      *      UPDATED BY AVERAGE SPEED TIME UPDATE CTR (ASTUPD) UNDERFLOW
00297
00298A 001E 0002 A ASTIME RMB 2
00299      *
00300      * VALUE OF ASDMI AT LAST ASTIME INCREMENT
00301
00302      *      UPDATED FROM ASDMI EVERY ASTIME UPDATE
00303
00304A 0020 0002 A LASDMI RMB 2
00305      *
00306      * VALUE OF ASDKM AT LAST ASTIME INCREMENT
00307
00308      *      UPDATED FROM ASDKM EVERY ASTIME UPDATE
00309
00310A 0022 0002 A LASDKM RMB 2
00311      *
00312      * STANDBY COUNTER
00313
00314      *      INITIALIZED TO 256 (0). APPROXIMATELY 2.5 MINUTES
00315      *      RE-INITIALIZED ON EVERY WHEEL INTERRUPT OR BUTTON PUSH
00316      *      DECREMENTED BY SPEED/CADENCE UPDATE CTR (SCUPD) UNDERFLOW
00317      *      WHEN UNDERFLOWED
00318      *      SET STANDBY BIT
00319      *      RESET TO 256 (0)
00320
00321A 0024 0001 A STBCTR RMB 1
00322      *
00323      * CURRENT FUNCTION
00324
00325      *      SET BY MODE CHANGE
00326
00327A 0025 0001 A FUNCT RMB 1      MODE = 0 TO MAXFUN
00328      *
00329      * .....
00330
00331 0026 A ENDCLR EQU *
00332      *
00333      * NOTE: EVERYTHING FROM HERE UP TO STATUS BYTE 1 SHOULD NOT BE
00334      * MOVED BECAUSE THE INITIALIZATION ROUTINE DOES A BLOCK CLEAR OF
00335      * THIS AREA.
00336      * .....
00337      * .....
00338      * .....
00339      * .....
00340      * .....
00341      * .....
00342      * EVERYTHING FROM HERE DOWN TO TSCAD IS INITIALIZED DURING STARTUP
00343      * BY MOVING A BLOCK OF DATA FROM ROM INTO RAM. DON'T REARRANGE ANY
00344      * OF THIS DATA WITHOUT REARRANGING THOSE ROM TABLES TOO.
00345
00346 0026 A FIRSTI EQU *
00347      *
00348      * .....

```

```

00349      *
00350      * .1 MILE CONSTANT
00351      *
00352      * CONTAINS NUMBER OF REVOLUTIONS/ .1 MILE.
00353      * INITIALIZED TO 27" DEFAULT BY INIT CODE
00354      * RECALCULATED BY WHEEL SIZE ROUTINE
00355      *
00356A 0026 0001 A TENTHM RMB 1
00357      *
00358      * MILES REVOLUTION COUNTER - TRIP ODO
00359      *
00360      *
00361      * INITIALIZED TO .1 MILE CONSTANT (TENTHM)
00362      * DECREMENTED ON EACH WHEEL INTERRUPT
00363      * ON UNDERFLOW RESET TO .1 MILE CONSTANT AND
00364      * INCR. TRIP MILES CTR (TMCTR)
00365      * DECR MILES UPDATE CTR (MUCTR)
00366      * ADDS 16 TO AVERAGE SPEED DISTANCE (ASDIST)
00367A 0027 0001 A MRCRTR RMB 1
00368      *
00369      * .1 MILE CONSTANT
00370      *
00371      * CONTAINS NUMBER OF REVOLUTIONS/MILE
00372      * INITIALIZED TO 27" DEFAULT BY INIT CODE
00373      * RECALCULATED BY WHEEL SIZE ROUTINE
00374A 0028 0002 A ONEMI RMB 2
00375      *
00376      * .1 KM CONSTANT
00377      *
00378      * CONTAINS NUMBER OF REVOLUTIONS/ .1 KM.
00379      * INITIALIZED TO DEFAULT (27" WHEEL) ON INIT
00380      * RECALCULATED BY WHEEL SIZE ROUTINE
00381      *
00382A 002A 0001 A TENTHK RMB 1
00383      *
00384      * KM REVOLUTION COUNTER - TRIP ODO
00385      *
00386      * INITIALIZED TO .1 KM CONSTANT (TENTHK)
00387      * DECREMENTED ON EACH WHEEL INTERRUPT
00388      * ON UNDERFLOW, RESET TO .1 KM CONSTANT AND
00389      * INCR TRIP KM CTR. (TKCTR)
00390      * DECR KM UPDATE CTR. (XUCTR)
00391      *
00392A 002B 0001 A KRCTR RMB 1
00393      *
00394      * ONE KM CONSTANT
00395      *
00396      * CONTAINS NUMBER OF REVOLUTIONS KM.
00397      * INITIALIZED TO DEFAULT VALUE (27" WHEEL) ON INIT
00398      * RECALCULATED BY WHEEL SIZE ROUTINE
00399      *
00400A 002C 0002 A ONEKM RMB 2
00401      *
00402      * CIRCUMFERENCE * 2 CONSTANT - ENGLISH UNITS
00403      *
00404      * INITIALIZED BY MAIN ROUTINE (DEFAULT = 27")
00405      * RECALCULATED BY WHEEL SIZE ROUTINE
00406      *

```

```

00407A 002E 0001 A CIR2MI RMB 1
00408      *
00409      * TEMPORARY CIR2MI
00410      *
00411      * USED DURING WHEEL SIZE UPDATE TO HOLD CIR2MI UNTIL
00412      * THE SET BUTTON IS PUSHED THE SECOND TIME
00413      *
00414A 002F 0001 A TCIR2 RMB 1
00415      *
00416      * CIRCUMFERENCE - METRIC UNITS
00417      *
00418      * INITIALIZED BY MAIN ROUTINE (DEFAULT = 27")
00419      * RECALCULATED BY WHEEL SIZE ROUTINE
00420      *
00421A 0030 0001 A CIRKM RMB 1
00422      *
00423      * SPEED CONSTANT - MPH
00424      *
00425      * CONSTANT USED IN THE INSTANTANEOUS SPEED CALCULATION
00426      * INITIALIZED BY MAIN ROUTINE (DEFAULT = 27")
00427      * RECALCULATED BY WHEEL SIZE FUNCTION
00428      *
00429A 0031 0002 A CMI RMB 2
00430      *
00431      * SPEED CONSTANT - KPH
00432      *
00433      * CONSTANT USED IN THE INSTANTANEOUS SPEED CALCULATION
00434      * INITIALIZED BY MAIN ROUTINE (DEFAULT = 27")
00435      * RECALCULATED BY WHEEL SIZE FUNCTION
00436      *
00437A 0033 0002 A CKM RMB 2
00438      *
00439      * CIRCUMFERENCE DISPLAY VALUE
00440      *
00441      * USED FOR DISPLAYING THE WHEEL CIRCUMFERENCE
00442      * INITED BY MAIN ROUTINE
00443      * RECALCULATED BY WHEEL SIZE FUNCTION
00444      *
00445A 0035 0002 A DISCIR RMB 2
00446      *
00447      * TEMPORARY DISPLAY CIRCUMFERENCE
00448      *
00449      * USED DURING SELECTION OF NEW WHEEL SIZE TO HOLD DISCIR
00450      * UNTIL THE SECOND SET IS PUSHED
00451      *
00452A 0037 0002 A TDISC RMB 2
00453      *
00454      * TOTAL MILES UPDATE CTR
00455      *
00456      * INITIALIZED TO ONEMI AT INIT AND ON UNDERFLOW
00457      * RESET TO ONEMI ON NEW WHEEL SIZE
00458      * DECREMENTED BY WHEEL INTERRUPT
00459      * ON UNDERFLOW REINITIALIZE TO ONEMI AND
00460      * INCREMENT TOTAL MILES CTR.
00461      *
00462A 0039 0002 A TOTMUC RMB 2
00463      *
00464      * TOTAL KMS UPDATE CTR

```

```

00465      *
00466      *      INITIALIZED TO ONEKM AT INIT AND ON UNDERFLOW
00467      *      RESET TO ONEKM ON NEW WHEEL SIZE
00468      *      DECREMENTED BY WHEEL INTERRUPT
00469      *      ON UNDERFLOW REINITIALIZE TO ONEKM AND
00470      *      INCREMENT TOTAL KMS CTR.
00471      *
00472A 003B 0002 A TOTKUC RMB 2
00473      *
00474      *      AVERAGE SPEED DISTANCE UPDATE CTR - MILES
00475      *
00476      *      INITIALIZED TO TENTHM AT INIT
00477      *      RESET TO TENTHM ON NEW WHEEL SIZE
00478      *      DECREMENTED BY WHEEL INTERRUPT
00479      *      ON UNDERFLOW REINITIALIZE TO TENTHM
00480      *      INCREMENT ASDMI
00481      *
00482A 003D 0001 A ASMUCT RMB 1
00483      *
00484      *      AVERAGE SPEED DISTANCE UPDATE CTR - KMS
00485      *
00486      *      INITIALIZED TO TENTHK AT INIT
00487      *      RESET TO TENTHK ON NEW WHEEL SIZE
00488      *      DECREMENTED BY WHEEL INTERRUPT
00489      *      ON UNDERFLOW REINITIALIZE TO TENTHK
00490      *      INCREMENT ASDKM
00491      *
00492A 003E 0001 A ASKUCT RMB 1
00493      *
00494      *      *
00495      *
00496      *      *
00497      *      *
00498      *      *
00499      *      *
00500      *      *
00501      *      *
00502      *      *
00503      *      *
00504      *      *
00505      *      *
00506      *      *
00507      *      *
00508      *      *
00509      *      *
00510      *      *
00511      *      *
00512      *      *
00513      *      *
00514      *      *
00515A 003F 0001 A SCUPD RMB 1
00516      *
00517      *      *
00518      *      *
00519      *      *
00520      *      *
00521      *      *
00522      *      *

```

```

00523      *
00524      *      WHEN UNDERFLOW
00525      *      DECREMENTS FLIP30
00526      *      RE-INITIALIZED TO 15
00527A 0040 0001 A DSPUPD RMB 1
00528      *
00529      *      *
00530      *      *
00531      *      *
00532      *      *
00533      *      *
00534A 0041 0001 A FLIP30 RMB 1
00535      *
00536      *      *
00537      *      *
00538      *      *
00539      *      *
00540      *      *
00541      *      *
00542      *      *
00543A 0042 0002 A ASTUPD RMB 2
00544      *
00545      *      *
00546      *      *
00547      *      *
00548      *      *
00549      *      *
00550      *      *
00551      *      *
00552      *      *
00553      *      *
00554A 0044 0001 A SECTR RMB 1
00555      *
00556      *      *
00557      *      *
00558      *      *
00559      *      *
00560      *      *
00561A 0045 0001 A WHCTR RMB 1
00562      *
00563      *      *
00564      *      *
00565      *      *
00566      *      *
00567      *      *
00568      *      *
00569      *      *
00570A 0046 0003 A STPCTR RMB 3
00571      *
00572      *      *
00573      *      *
00574      *      *
00575      *      *
00576      *      *
00577      *      *
00578      *      *
00579      *      *
00580A 0049 0001 A SPDZER RMB 1

```

```

00581 *
00582 * CADENCE EQUAL ZERO FLAG
00583 *
00584 *
00585 * IF ZERO THEN CADENCE IS NOT ZERO
00586 * IF NON ZERO THEN CADENCE IS ZERO
00587 * INITIALIZED TO ZERO
00588 * SET TO 1 AFTER 6 SECS OF NO CADENCE INTERRUPTS
00589 * RESET TO ZERO ON ANY CADENCE INPUT
00590A 004A 0001 A CADZER RMB 1
00591 *
00592 * TIME BETWEEN LAST 2 WHEEL INTERRUPTS IN MSEC (TBWHL)
00593 *
00594 *
00595 * UPDATED EVERY INTERRUPT FROM LAST TIME SINCE LAST WHEEL INTER
00596 * MAXIMUM VALUE ALLOWED = 0 MPH
00597A 004B 0002 A TBWHL RMB 2
00598 *
00599 * TIME SINCE LAST WHEEL INTERRUPT IN MSEC
00600 *
00601 *
00602 * UPDATED EACH MSEC INTERRUPT
00603 * MAX VALUE ALLOWED = 32K
00604 * MOVED TO TIME BETWEEN LAST 2 WHEEL INTERRUPTS (TBWHL) ON WHEEL
00605A 004D 0002 A TSWHL RMB 2
00606 *
00607 * TIME BETWEEN LAST 2 CADENCE INPUTS IN MSEC
00608 *
00609 *
00610 * UPDATED EACH TIME A NEW CADENCE INPUT OCCURS FROM THE TIME
00611 * SINCE THE LAST CADENCE INPUT (TSCAD)
00612 * MAX VALUE ALLOWED = 32K = 8 REV/MIN
00613A 004F 0002 A TBCAD RMB 2
00614 *
00615 * TIME SINCE LAST CADENCE INPUT
00616 *
00617 *
00618 * UPDATED EACH MSEC
00619 * MAX VALUE ALLOWED = 32K MSEC
00620 * MOVED TO TIME BETWEEN LAST 2 CADENCE INPUTS (TBCAD) ON NEW CA
00621A 0051 0002 A TSCAD RMB 2
00622 *
00623 *
00624 *
00625 * EVERYTHING FROM HERE UP TO SCUPD IS REINITIALIZED BEFORE EACH STOP
00626 * EVERYTHING FROM HERE UP TO TENTHM IS INITIALIZED ON START UP
00627 *
00628 * DON'T REARRANGE ANY OF THIS CODE!!!!!!!!!!!!!!
00629 *
00630 *
00631 *
00632 * TEMPORARY DISPLAY VALUES.
00633 *
00634 *
00635 *
00636 *
00637 * DISPLAY VALUE
00638 *

```

```

00639 *
00640 * CONTAINS A BINARY NUMBER TO BE CONVERTED TO BCD AND DISPLAYED
00641 * UPDATED IN 600 MSEC ROUTINE
00642A 0053 0002 A DISPLY RMB 2
00643 *
00644 *
00645 * D1-D4
00646 *
00647 *
00648 * USED TO STORE DATA FOR LCD DIGITS. BY ALGORITHM, THIS IS
00649 * CONVERTED TO THE TRUE PORT VALUES AND STORED IN PA-PD (BELOW)
00650A 0055 0001 A D1 RMB 1
00651A 0056 0001 A D2 RMB 1
00652A 0057 0001 A D3 RMB 1
00653A 0058 0001 A D4 RMB 1
00654 *
00655 * PA-PD
00656 *
00657 *
00658 * TEMPORARY LOCATIONS THAT CONTAIN A COPY OF THE DATA IN THE
00659 * I/O PORTS (PORTA - PORTD).
00660A 0059 0001 A PA RMB 1
00661A 005A 0001 A PB RMB 1
00662A 005B 0001 A PC RMB 1
00663A 005C 0001 A PD RMB 1
00664 *
00665 *
00666 *
00667 * MULTIPLY / DIVIDE VARIABLES
00668 *
00669 *
00670 *
00671 * DIVISOR, MTOTAL
00672 *
00673 *
00674 * DIVISOR FOR 16 BIT / 16 BIT DIVIDE ROUTINE
00675 * ALSO USED AS A TEMP IN MULTIPLY
00676 *
00677A 005D 0001 A MTOTAL EQU *
00678A 005E 0002 A DIVSR RMB 2
00679 *
00680 *
00681 * DIVIDEND
00682 *
00683A 005F 0002 A DIVDND RMB 2
00684 *
00685 * TEMPORARY BYTE
00686 *
00687A 0061 0001 A TEMP RMB 1
00688 *
00689 *
00690 *
00691 *
00692 * TEMPORARY STORAGE FOR X REGISTER IN DIVIDE
00693 * ALSO USED FOR COUNTER IN MULTIPLY
00694 *
00695A 0062 0001 A MOUNT EQU *
00696A 0063 0001 A SAVEX RMB 1
00697 *

```

```

00697 *
00698 *
00699 *
00700 *
00701 *
00702 *
00703 *
00704 * BASE PAGE ROM
00705 *
00706 *
00707 *
00708 *
00709 *
00710 *
00711 * INITIALIZATION ROUTINES
00712 *
00713 * BELOW ARE SEVERAL INITIALIZATION SUBROUTINES THAT ARE USED
00714 * THRUOUT THE PROGRAM.
00715 *
00716 *
00717A 0080 *
00718 * ORG SBC PAGE ZERO ROM
00719 *
00720 * INITIALIZE AVERAGE SPEED VALUES
00721 *
00722 *
00723A 0080 3F 1A A CLRAS EQU *
00724A 0082 3F 1B A CLR ASDMI 0 --> AVERAGE SPEED DISTANCE
00725A 0084 3F 1C A CLR ASDKM
00726A 0086 3F 1D A CLR ASDKM+1
00727A 0088 3F 1E A CLR ASTIME 0 --> AVERAGE SPEED TIME
00728A 008A 3F 1F A CLR ASTIME+1
00729A 008C 3F 20 A CLR LASDMI 0 --> LAST ASDMI
00730A 008E 3F 21 A CLR LASDMI+1
00731A 0090 3F 22 A CLR LASDKM 0 --> LAST ASDKM
00732A 0092 3F 23 A CLR LASDKM+1
00733A 0094 AD 09 009F BSR CLRASU 750 --> AVERAGE SPEED UPDATE CTR
00734 *
00735A 0096 B6 26 A LDA TENTHM TENTHM --> AVERAGE SPEED UPDATE CTR - MI
00736A 0098 B7 3D A STA ASMUCT
00737A 009A B6 2A A LDA TENTHK TENTHK --> AVERAGE SPEED UPDATE CTR - KM
00738A 009C B7 3E A STA ASKUCT
00739A 009E B1 *
00740 *
00741 * SET AVERAGE SPEED UPDATE TO 750
00742 * CLRASU EQU * ENTER HERE TO ONLY DO AVG. SPEED UPDATE CTR.
00743A 009F A6 02 A LDA #750/256 750 --> AVERAGE SPEED UPDATE CTR.
00744A 00A1 B7 42 A STA ASTUPD
00745A 00A3 A6 0E A LDA #750.5FF
00746A 00A5 B7 43 A STA ASTUPD+1
00747A 00A7 B1 RTS EXIT BOTH CLRAS OR CLRASU
00748 *
00749 * INITIALIZE DISTANCE COUNTERS
00750 *
00751 *
00752A 00AB 3F 16 A CLRDIS EQU *
00753A 00AA 3F 17 A CLR TOTMI 0 --> TOTAL MILES
00754A 00AC 3F 18 A CLR TOTMI+1
00755A 00AC 3F 18 A CLR TOTKM 0 --> TOTAL KMS

```

```

00755A 00A2 3F 19 A CLR TOTKM+1
00756 *
00757 * HERE TO JUST INITIALIZE MILES AND KM UPDATE CTR.
00758 * CLRUP EQU *
00759A 00B0 AD 09 00BB BSR INITTM ONEMI --> TOTMUC
00760 *
00761 *
00762A 00B2 3F 2C A LDA ONEKM ONEKM --> TOTKUC
00763A 00B4 B7 3B A STA TOTKUC
00764A 00B6 B6 2D A LDA ONEKM+1
00765A 00B8 B7 3C A STA TOTKUC+1
00766A 00BA B1 RTS
00767 *
00768 *
00769A 00BB 3F 28 A INITTM EQU *
00770A 00BD B7 39 A LDA ONEMI ONEMI --> TOTMUC
00771A 00BF B6 29 A STA TOTMUC
00772A 00C1 B7 3A A LDA ONEMI+1
00773A 00C3 B1 A STA TOTMUC+1
00774 *
00775 *
00776 *
00777 * DINC - 16 BIT INCREMENT SUBROUTINE
00778 * ON ENTRY X POINTS TO 2 BYTE VALUE TO BE INCREMENTED
00779 *
00780 * NORMAL EXECUTION = 15 CYCLES
00781 * WORSE CASE = 20 CYCLES
00782 *
00783 *
00784A 00C4 6C 01 A DINC EQU *
00785A 00C6 26 01 00C9 INC I,X INCR LS BYTE
00786A 00C8 7C BNE D11 BRANCH IF NO CARRY
00787A 00C9 B1 INC ,X INCR MS BYTE
00788 *
00789 *
00790 *
00791 *
00792 * EXECUTION TIME = 16 CYCLES
00793 *
00794 *
00795A 00CA 3F 24 A NOSTBY EQU *
00796A 00CC 1F 10 A CLR STRCTR RESET CTR (256)
00797A 00CE B1 A BCLR STNDBY,STAT1 CLEAR STANDBY MODE
00798 *
00799 *
00800 *
00801 *
00802 * BRANCH VECTOR HERE SO CALLS TO MUL WILL BE SHORT
00803A 00CF 20 5B 012C MUL BRA MULST
00804 *
00805 *
00806 *
00807 *
00808 *
00809 *
00810 *
00811 *
00812 *
00813A 00D1 00D1 A DDEC EQU *

```

```

00813A 00D1 6D 01 A TST 1,X WILL WE CAUSE A BORROW?
00814A 00D3 26 01 BNE DDI BRANCH IF NOT
00815A 00D5 7A 00 DEC ,X YES, DO BORROW FIRST (DECR MS BYTE)
00816A 00D6 6A 01 A DD1 DEC 1,X DECR LS BYTE
00817A 00D8 26 01 BNE DDXIT SET Z BIT CORRECTLY
00818A 00DA 7D TST ,X
00819A 00DB 01 DDXIT RTS
00820 *
00821 *
00822 *
00823 * CLEAR THE LCD DISPLAYS
00824 *
00825 *
00826 00DC A CLRCD EQU *
00827A 00DC 3F 00 A CLR PORTA
00828A 00DE 3F 01 A CLR PORTB
00829A 00E0 3F 02 A CLR PORTC
00830A 00E2 3F 03 A CLR PORTD
00831A 00E4 01 RTS
00832 *
00833 *
00834 * DIVIDE ROUTINE
00835 *
00836 *
00837 *
00838 *
00839 * 16 BIT / 16 BIT --> 8 BIT RESULT DIVIDE
00840 *
00841 * ON ENTRY:
00842 * DIVSR CONTAINS THE DIVISOR
00843 * DIVDND CONTAINS THE DIVIDEND
00844 *
00845 * ON EXIT:
00846 * A CONTAINS THE ROUNDED QUOTIENT
00847 * DIVSR AND DIVDND ARE DESTROYED
00848 * TEMP IS DESTROYED
00849 *
00850 * IF DIVISION BY ZERO, 255 IS RETURNED.
00851 *
00852 * ADAPTED FROM A 6801 DIVIDE ALGORITHM FOUND IN THE 6801
00853 * USER'S MANUAL WRITTEN BY BILL BRUCE.
00854 *
00855 * AVERAGE EXECUTION SPEED = 644 CYCLES
00856 * WORST CASE SPEED = 1376 CYCLES
00857 *
00858 *
00859 *
00860 *
00861 00E5 A DIV EQU *
00862A 00E5 A6 02 A LDA #2 SET SHIFT COUNT TO GENERATE 9 BITS
00863A 00E7 87 61 A STA TEMP 8 FOR RESULT PLUS 1 TO ROUND
00864A 00E9 B6 5D A LDA DIVSR
00865A 00EB BE 5E A LDA DIVSR+1
00866A 00ED 26 07 00F6 BNE NOZERO CHECK FOR DIVISION BY ZERO
00867A 00EF 4D TSTA
00868A 00F0 26 05 00F7 BNE DIV01 BRANCH IF NOT ZERO
00869A 00F2 A6 FF A LDA #255 DIVISION BY ZERO
00870A 00F4 20 35 012B BRA DIVOUT GO EXIT

```

```

00871 *
00872 00F6 A NOZERO EQU *
00873A 00F6 4D TSTA SHIFT DIVISOR LEFT UNTIL SIGN BIT =1
00874A 00F7 2B 06 00FF DIV01 BMI OUTD
00875A 00F9 3C 61 A LOOP2 INC TEMP
00876A 00F8 58 LSLX
00877A 00FC 49 ROLA INCR SHIFT COUNT
00878A 00FD 2A FA 00F9 BPL LOOP2
00879A 00FF B7 5D A OUTD STA DIVSR RESTORE DIVISOR
00880A 0101 BF 5E A STX DIVSR+1
00881A 0103 5F CLRX CLEAR PLACE FOR QUOTIENT
00882 * MAIN LOOP
00883A 0104 B6 60 A LOOP LDA DIVDND+1 DIVIDEND-DIVISOR --> DIVIDEND
00884A 0106 B8 5E A SUB DIVSR+1
00885A 0108 B7 60 A STA DIVDND+1
00886A 010A B6 5F A LDA DIVDND
00887A 010C B2 5D A SBC DIVSR
00888A 010E 24 09 0119 BCC ZOT BRANCH IF CARRY SET (BEFORE SAVE OF DIVDND)
00889A 0110 B6 60 A LDA DIVDND+1 ADD IT BACK
00890A 0112 B8 5E A ADD DIVSR+1 NOTE, MSB WAS NEVER STORED SO WE ONLY
00891A 0114 B7 60 A STA DIVDND+1 HAVE TO ADD TO THE LS BYTE
00892A 0116 58 LSLX SHIFT IN ZERO
00893A 0117 20 04 011D BRA OVER
00894A 0119 B7 5F A ZOT STA DIVDND SAVE MS BYTE OF NEW DIVIDEND
00895A 011B 99 SEC
00896A 011C 59 ROLX SHIFT 1 BIT INTO QUOTIENT
00897A 011D 49 OVER ROLA CARRY INTO QUOTIENT
00898A 011E 34 5D A LSR DIVSR SHIFT DIVISOR RIGHT BY 1
00899A 0120 36 5E A ROR DIVSR+1
00900A 0122 3A 61 A DEC TEMP DONE
00901A 0124 26 DE 0104 BNE LOOP BRANCH IF NOT
00902A 0126 44 LSRA GET CORRECT QUOTIENT
00903A 0127 56 RORX C = ROUND BIT
00904A 0128 9F TXA
00905A 0129 A9 00 A ADC #0 AND ROUND
00906 012B A DIVOUT EQU *
00907A 012B 01 RTS EXIT
00908 *
00909 *
00910 *
00911 * MULTIPLY
00912 *
00913 * 8 BIT BY 8 BIT UNSIGNED MULTIPLY
00914 * OPERANDS IN A AND X ON ENTRY
00915 * 16 BIT RESULT IN X:A ON EXIT; X HAS MSB.
00916 *
00917 * AVERAGE EXECUTION = 323 CYCLES
00918 * WORST CASE = 425 CYCLES
00919 *
00920 012C A MULST EQU *
00921A 012C 3F 5D A CLR MTOTAL INITIALIZE RESULT TEMP
00922A 012E 3F 5E A CLR MTOTAL+1
00923A 0130 B7 61 A STA TEMP SAVE ONE ARGUMENT
00924A 0132 A6 09 A LDA #9
00925A 0134 B7 62 A STA MOUNT BYTE LENGTH = 8
00926 *
00927 * THE ALGORITHM IS A PLAIN SHIFT AND ADD
00928 *

```

```

00929      0136      A BIGLOP EQU *
00930A 0136 B6 61      A LDA TEMP GET BACK ARGUMENT
00931      0138      A SLOOP EQU *
00932A 0138 3A 62      A DEC MCOUNT WHILE COUNT IS NOT ZERO
00933A 013A 27 13      014F BEQ DONE
00934A 013C 38 5E      A LSL MTOTAL+1 SHIT TOTAL LEFT BY 1
00935A 013E 39 5D      A ROL MTOTAL
00936A 0140 58          A LSLX GET NEXT BIT FROM X
00937A 0141 24 F5      0138 BCC SLOOP NO ADD IF C=0
00938          *
00939          * C=1: ADD A TO TOTAL
00940          *
00941A 0143 B7 61      A STA TEMP
00942A 0145 B8 5E      A ADD MTOTAL+1
00943A 0147 24 02      0148 BCC NOCARY
00944A 0149 3C 5D      A INC MTOTAL
00945A 014B B7 5E      A NOCARY STA MTOTAL+1
00946A 014D 20 E7      0136 BRA BIGLOP
00947          *
00948          * HERE TO EXIT
00949          *
00950      014F      A DONE EQU *
00951A 014F BE 5D      A LDX MTOTAL
00952A 0151 B6 5E      A LDA MTOTAL+1 RETURN RESULT IN A:X
00953A 0153 01          A RTS
00954          *
00955          *
00956          *
00957          * MAIN ROUTINE
00958          *
00959          *
00960          *
00961          *
00962          * THE MAIN LOOP INITIALIZES ALL VARIABLES ON POWER-ON-RESET AND
00963          * THEN GOES INTO AN ENDLESS LOOP. THE LOOP LOOKS TO SEE IF THE
00964          * ROUTINES HAVE SCHEDULED A 30 OR 600 MSEC ROUTINE REQUEST. IF
00965          * THEY ARE CALLED, AT THE END OF EACH LOOP THE PROCESSOR ENTERS
00966          * THE 'WAIT' STATE UNTIL A IMSEC OR WHEEL INTERRUPT OCCURS.
00967          * WHILE IN STANDBY MODE THE 600MSEC ROUTINE WILL NOT BE EXECUTE
00968          *
00969          * TIMING ANALYSIS --
00970          *
00971          * SINCE IT IS CRITICAL THAT A REAL TIME PROGRAM OF THIS NATURE
00972          * NEVER RUN OUT OF PROCESSING TIME, I DID AN ANALYSIS OF THE
00973          * TIMING. THE PROGRAM ESSENTIALLY REPEATS ITSELF EVERY 600MSEC,
00974          * HENCE, WE CAN CONSIDER ONE 600MSEC CYCLE. ON THE AVERAGE, IN
00975          * EACH 600MSEC, THE FOLLOWING ROUTINES WILL BE ACTIVE
00976          *
00977          * 228 MSEC MAIN ROUTINE AND IMSEC INTERRUPT
00978          * 3 MSEC 30 MSEC ROUTINE
00979          * 7 MSEC WHEEL INTERRUPT
00980          * 10 MSEC 600MSEC ROUTINE
00981          *
00982          * ---
00983          * 248 MSEC
00984          * THIS IS A 41% UTILIZATION. (59% OF THE TIME IS SPENT IN WAIT)
00985          *
00986          * IN ORDER TO INSURE OPERATION I ALSO DID A WORST CASE TIMING.
00987          * THE SITUATION TO ACHIEVE THE WORST CASE IS B1*ARRE. (GOING

```

```

00987          *
00988          * 90MPH WHILE RESETTNG THE WHEEL SIZE WITH ALL MILEAGE
00989          * VALUES OVERFLOWING SIMULTANEOUSLY, ETC.)
00990          *
00991          * 360 MSEC MAIN ROUTINE AND IMSEC INTERRUPT
00992          * 34 MSEC 30 MSEC ROUTINE
00993          * 28 MSEC WHEEL INTERRUPT
00994          * 32 MSEC 600 MSEC ROUTINE
00995          *
00996          * ---
00997          * 454
00998          * THIS IS A UTILIZATION OF 75%.
00999          *
01000          *
01001          * EMPIRACALLY MEASURED UTILIZATIONS WERE --
01002          *
01003          * NORMAL OPERATION 45%
01004          * STANDBY 30%
01005          *
01006          *
01007          *
01008          *
01009      0154      A ODO EQU *
01010A 0154 9B          A SEI SET MASK JUST IN CASE
01011A 0155 9C          A RSP ALSO JUST IN CASE
01012A 0156 AE 18      A LDX #STAT1 CLEAR ALL VARIABLES BETWEEN STATUS BYTE 1
01013A 0158 7F          CLRIT CLR ,X AND FUNCTION INCLUSIVE.
01014A 0159 5C          A INCX
01015A 015A A3 26      A CMPLX #ENDCLR
01016A 015C 26 FA      0158 BNE CLRIT
01017          *
01018          * MOVE A TABLE OF INITIALIZATION VALUES FROM ROM TO RAM. THE
01019          * DEFAULT VALUES ASSUME A STANDARD 27" BICYCLE WHEEL
01020          * WHICH IS ABOUT 84.5" IN CIRCUMFERENCE.
01021          *
01022A 015E AE 2C          A LDX #ENDTAB-INITAB GET NUMBER OF VALUES TO XFER
01023A 0160 D6 01B3      A INITIT LDA INITAB,X GET INIT VALUE
01024A 0163 E7 26          A STA FIRSTI,X PUT IN RAM
01025A 0165 5A          A DECX
01026A 0166 2A F8      0160 BPL INITIT
01027          *
01028          *
01029          * SETUP THE I/O PORTS. INIT THE I/O PORT DATA AND THE DATA
01030          * DIRECTION REGISTER.
01031          *
01032          *
01033A 0168 BD DC          A JSR CLRLCD CLEAR LCD DATA
01034A 016A A6 FF          A LDA #1
01035A 016C B7 04          A STA PORTA+DDR SET DDR'S
01036A 016E B7 05          A STA PORTB+DDR PORTA,B,D ARE ALL OUTPUTS
01037A 0170 B7 07          A STA PORTD+DDR
01038A 0172 A6 78          A LDA #578 PORTC (BITS 7,2,1,0 ARE INPUTS)
01039A 0174 B7 06          A STA PORTC+DDR
01040          *
01041          * RESTART ENTRY - RESTART HERE AFTER STOP.
01042          *
01043      0176      A RESTRT EQU *
01044          *

```



```

01045      *
01046      *
01047      *
01048      *
01049A 0176 A6 41      A      LDA      #BIT0+BIT6 PRESCALER=/2; DISABLE INTERRUPTS
01050A 0178 B7 09      A      STA      TCR
01051A 017A A6 7D      A      LDA      #125      250,000HZ/125*2*1 = 1 MSEC
01052A 017C B7 08      A      STA      TIMER
01053A 017E 1D 09      A      BCLR      6,TCR      ENABLE INTERRUPTS
01054      *
01055      *
01056      *
01057      *
01058A 0180 9A          *      CLI
01059      *
01060      *
01061      *
01062      *
01063      *
01064      *
01065      *
01066      *
01067      *
01068      *
01069      *
01070A 0181 07 11 03 0187      A BACK      EQU      *
01071A 0184 CD 0329      A      JSR      RQS600,STAT2,BACK2 BRANCH IF NO 600MSEC REQUESTED
01072      *
01073A 0187 0B 11 03 018D      A BACK2      EQU      *
01074A 018A CD 02A2      A      JSR      RQST30,STAT2,BACK3 BRANCH IF NO 30MSEC REQUEST
01075      *
01076A 018D 0E 10 03 0193      A BACK3      EQU      *
01077A 0190 8F          *      BRSET   STNDBY,STAT1,BACK4 BRANCH IF STANDBY MODE
01078A 0191 20 EE 0181      A      WAIT      GO TO SLEEP
01079      *
01080A 0193 00 10 03 0199      A BACK4      EQU      *
01081A 0196 8F          *      BRA      BACK      WHEN AWAKENED GO TO START
01082A 0197 20 EE 0187      A      EQU      *
01083      *
01084      *
01085      *
01086      *
01087      *
01088      *
01089      *
01090A 0199 9B          *      BRSET   STOPIT,STAT1,BACK5 BRANCH IF STOP MODE TOO
01091A 019A AE 13          *      WAIT      GO TO SLEEP
01092A 019C D6 01CC      A BACK5      EQU      *
01093A 019F E7 3F          *      BRA      BACK2      IN STANDBY, SKIP 600MSEC CHECK
01094A 01A1 5A          *
01095A 01A2 26 FB 019C      A BACK6      EQU      *
01096      *
01097A 01A4 BD 00          *      JSR      CLRAS      RESET AVERAGE SPEED VARIABLES
01098      *
01099A 01A6 A6 01          *      LDA      #BIT7+BIT0
01100A 01A8 B7 10          *      STA      STAT1      SET STANDBY AND STOP MODE BITS (CLEAR REST)
01101A 01AA B6 11          *      LDA      STAT2
01102A 01AC A4 40          *      AND      #BIT6      SAVE KM/MI FLAG

```

PAGE 020 BICYCLE .SA:1

MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

```

01103A 01AE B7 11      A      STA      STAT2
01104      *
01105A 01B0 0E          *      STOP      STOP, ENABLE IRQ
01106      *
01107A 01B1 20 C3 0176      A      BRA      RESTRT
01108      *
01109      *
01110      *
01111      *
01112      *
01113      *
01114      *
01115      *
01116      *
01117      *
01118      *
01119      *
01120      *
01121A 01B3 4B          *      THESE VALUES ARE WRITTEN FROM ROM INTO RAM AT STRATUP
01122A 01B5 02EE          *      TO INITIALIZE THE DEFAULT VARIABLES. DEFAULT IS 27"
01123A 01B7 2F          *      WHEELS THAT HAVE A CIRCUMFERENCE OF ABOUT 84.5 INCHES
01124A 01B9 01D6          *
01125A 01BB A9          *
01126A 01BE 12BB          *
01127A 01C2 034D          *
01128A 01C6 02EE          *
01129A 01CA 48          *
01130      *
01131      *
01132      *
01133      *
01134      *
01135A 01CC 14          *      DON'T CHANGE THE ORDER OF THIS DATA WITHOUT ALSO CHANGING
01136A 01CF 02EE          *      THE ORDER OF THE RAM DATA INTO WHICH IT WILL BE WRITTEN.
01137A 01D1 08          *
01138A 01D2 02          *
01139A 01D3 03          *
01140A 01D6 01          *
01141      *
01142A 01D8 0000          *      THESE VALUES ARE ALSO USED TO RESTART THE SOFTWARE AFTER
01143      *      A STOP OPERATION.
01144      *
01145      *
01146      *
01147      *
01148      *
01149      *
01150      *
01151      *
01152      *
01153      *
01154      *
01155      *
01156      *
01157      *
01158      *
01159      *
01160      *

```

```

01161 * DEBOUNCES THE CADENCE INPUT AND STORES THE CADENCE PERIOD WHEN A
01162 * CADENCE INPUT OCCURS.
01163 *
01164 * THE LCD'S ARE COMPLEMENTED HERE EACH 15MSEC. UNLESS A WHEEL INTERRU
01165 * OCCURS JUST BEFORE A 1MSEC INTERRUPT, THERE IS NO AVERAGE DC VOLTAG
01166 * BY THE LCD'S (I.E., THEY ARE SWITCHED EXACTLY EVERY 15MSEC). IF THE
01167 * WHEEL INTERRUPT BEATS THE 1MSEC INTERRUPT BY 1 CYCLE AND THE BATTER
01168 * IS AT 9V (WORST CASE), THEN THE AVERAGE DC VOLTAGE IS 42MV. THIS IS
01169 * BELOW THE 50MV THAT CAUSES LCD FATIGUE.
01170 *
01171 * EXAMPLE: WHEEL INTERRUPT RUNS 36 CYCLES BEFORE RE-ENABLING INTERRUPT
01172 *
01173 *      AVG DC = (36 CYCLES * 40SEC * 9V) / 30,000 USEC = 42MV
01174 *
01175 *
01176 *
01177 *      NORMAL EXECUTION = 77 CYCLES (288 USEC)
01178 *      WORSE CASE      = 207 CYCLES (828 USEC)
01179 *
01180 *
01181 *
01182 *
01183 *
01184 *
01185 *
01186 *
01187 *
01188 *      ALL INTERRUPTS ARE DISABLED FOR WHOLE ROUTINE
01189 *
01190 *
01191 *      RESET TIMER VALUE
01192 *
01193A 01E8 A6 7D A LDA #123 2 TIMER DATA = 125 + (CURRENT TIMER) -2
01194A 01E2 B8 08 A ADD TIMER 3 THE CURRENT TIMER VALUE IS NEGATIVE
01195A 01E4 B7 08 A STA TIMER 4 THE -2 IS TO FIX FOR TIME BETWEEN ADD AND S
01196A 01E6 1F 09 A BCLR 7,TCR 5 CLEAR TIMER INTERRUPT
01197A 01E8 3A 40 A DEC DSPUPD 5 HAS 15 MSEC PASSED?
01198A 01EA 26 16 0202 BNE ONE9 3 BRANCH IF NOT
01199 *
01200 * COMPLEMENT THE LCD DISPLAYS
01201 *
01202A 01EC 33 00 A COM PORTA
01203A 01EE 33 01 A COM PORTB
01204A 01F0 33 02 A COM PORTC
01205A 01F2 33 03 A COM PORTD
01206 *
01207A 01F4 A6 0F A LDA #15 15 --> DISPLAY UPDATE CTR.
01208A 01F6 B7 40 A STA DSPUPD
01209A 01F8 3A 41 A DEC FLIP30 SEE IF ITS TIME TO DO THE 30MSEC ROUTINE
01210A 01FA 26 06 0202 BNE ONE9
01211A 01FC A6 02 A LDA #2
01212A 01FE B7 41 A STA FLIP30
01213A 0200 1A 11 A BSET ROST30,STAT2 REQUEST 30MSEC ROUTINE
01214 *
01215 *
01216A 0202 0E 10 40 0245 A ONE9 EQU *
01217A 0204 0E 10 40 0245 A BRSET STNDBY,STAT1,ONE8 5 SKIP REST IF IN STANDBY
01218A 0206 3C 4E A INC TSLWHL+1 5 INCREMENT TSLWHL

```

```

01219A 0207 26 09 0212 BNE ONE1 3
01220A 0209 3C 4D A INC TSLWHL
01221A 020B 0D 4D 04 0212 BRCLR 6,TSLWHL,ONE1 IF NO NEW WHEEL INTERRUPT IN 16.384 SECS
01222A 020E 3A 4D A DEC TSLWHL THEN WE MUST BE COMPLETELY STOPPED
01223A 0210 18 49 A BSET 8,SPDZER SET SPEED ZERO FLAG
01224 *
01225 *
01226A 0212 3C 52 A ONE1 EQU *
01227A 0214 26 0C 0222 INC TSCAD+1 5 INCREMENT TSCAD
01228A 0216 3C 51 A BNE ONE2 3
01229A 0218 06 51 A INC TSCAD
01230A 021A 11 18 A LDA TSCAD IF TIME = 6 SECS THEN THIS PERSON
01231A 021C 26 04 0222 CNP #24 18 NOT PEDDLING
01232A 021E 3A 51 A BNE ONE2
01233A 0220 18 4A A DEC TSCAD KEEP IT IN RANGE
01234A 0222 18 4A A BSET 8,CADZER CADENCE NOT ZERO
01235 *
01236A 0222 00 02 08 0236 A ONE2 EQU *
01237A 0224 03 11 04 023C BRSET CAD,10X,ONE5 5 BRANCH IF CADENCE NOT CLOSED
01238A 0226 12 11 04 023C BRCLR CADWN,STAT2,ONE4 BRANCH IF CADENCE PRESENT STATUS NOT
01239A 0228 20 19 0245 BSET WTCAD,STAT2 SET WAITING FOR CADENCE TO CLEAR STATUS
01240A 022A 14 11 A BRA ONE5
01241A 022C 14 11 A EQU *
01242A 022E 20 15 0245 BSET CADWN,STAT2 SET CADENCE PRESENT STATUS
01243A 0230 14 11 A BRA ONE5
01244A 0232 13 11 A EQU *
01245A 0234 03 11 18 0245 BRCLR WTCAD,STAT2,ONE5 5 BRANCH IF NOT WAITING FOR CADENCE T
01246A 0236 13 11 A BRSET WTCAD,STAT2 CLEAR WAITING FOR CADENCE TO CLEAR STATUS
01247A 0238 3F 4A A CLR CADZER CADENCE NOT ZERO
01248A 023A 06 51 A LDA TSCAD TIME SINCE LAST CADENCE --> TIME BETWEEN
01249A 023C 07 4F A STA TSCAD LAST TWO CADENCE CLOSURES
01250A 023E 06 52 A LDA TSCAD+1
01251A 0240 07 50 A STA TSCAD+1
01252A 0242 3F 51 A CLR TSCAD 8 --> TSCAD
01253A 0244 3F 52 A CLR TSCAD+1
01254A 0246 00 0245 A ONE2 EQU *
01255A 0248 00 0245 A RTI 9
01256 *
01257 *
01258 *
01259 *
01260 *
01261 *
01262 *
01263 *
01264 *
01265 *
01266 *
01267 *
01268 *
01269 *
01270 *
01271 *
01272 *
01273 *
01274 *
01275 *
01276 *

```

W H E E L I N T E R R U P T

EACH TIME THE WHEEL HAS 1 COMPLETE REVOLUTION A WHEEL INTERRU
OCCURS. IF .1 MILE OR KM HAS OCCURRED, THE APPROPRIATE TRIP
DISTANCES ARE UPDATED AND IF 1 MILE HAS OCCURRED THE TOTAL
DISTANCES ARE UPDATED. ALSO THE AVERAGE SPEED DISTANCE IS
UPDATED EACH MILE AND THE TIME SINCE THE LAST WHEEL INTERRUPT
IS SAVED FOR SPEED CALCULATIONS.

ABOUT THE QUICKEST THIS INTERRUPT COULD OCCUR IS ABOUT EVERY
45 MSEC. (SMALL WHEEL AT 90MPH)

NORMAL EXECUTION = 207 CYCLES - 828 USEC
WORSE CASE = 513 CYCLES - 2052 USEC

```

01277 *
01278 *
01279 *
01280 *
01281 *
01282 *
01283 *
01284 *
01285 *
01286 *
01287 *
01288 *
01289 *
01290 *
01291 *
01292 *
01293 *
01294 *
01295 *
01296 *
01297 *
01298A *
01299A *
01300 *
01301A *
01302A *
01303A *
01304A *
01305 *
01306A *
01307A *
01308A *
01309 *
01310A *
01311A *
01312A *
01313 *
01314A *
01315A *
01316A *
01317A *
01318 *
01319A *
01320A *
01321 *
01322 *
01323 *
01324 *
01325A *
01326A *
01327 *
01328A *
01329A *
01330A *
01331A *
01332A *
01333A *
01334A *
01335 *
01336A *
01337A *
01338A *
01339 *
01340A *
01341A *
01342A *
01343A *
01344A *
01345A *
01346A *
01347 *
01348A *
01349 *
01350 *
01351 *
01352 *
01353 *
01354 *
01355 *
01356 *
01357 *
01358 *
01359 *
01360 *
01361 *
01362 *
01363 *
01364 *
01365 *
01366 *
01367 *
01368 *
01369 *
01370 *
01371 *
01372 *
01373A *
01374A *
01375 *
01376 *
01377 *
01378 *
01379A *
01380A *
01381A *
01382A *
01383A *
01384A *
01385A *
01386A *
01387A *
01388A *
01389A *
01390 *
01391 *
01392 *

```

LEAVE OTHER INTERRUPTS DISABLED FOR AWHILE

18 CYCLES TO GET HERE
3 TIME SINCE LAST WHEEL INTERRUPT -->
4 TIME BETWEEN INTERRUPTS
3
4
5 CLEAR TIME SINCE LAST INTERRUPT
5
5 CLEAR ZERO SPEED FLAG
2 ENABLE 1 MSEC INTERRUPTS

36 CYCLES BEFORE INTERRUPTS ARE RE-ENA

NOSTBY CLEAR STANDBY
STOPTI,STAT1 CLEAR STOP MODE

DO MILEAGE FOR TRIP AND TOTAL MILES. ALSO DO AVERAGE SPEED DISTANCE

DEC MRCTR DECR MILES REVOLUTION CTR
BNE WH07 SKIP IF NOT ZERO

NEW 1.1 MILE FOR TRIP ODO
LDA TENTHNM RE-INIT MILES REVOLUTION CTR
STA MRCTR
LDX #TMCTR INCR TRIP MILES CTR
JSR DINC

WH07 LDX #TOTMUC DECREMENT TOTAL MILES UPDATE CTR
JSR DDEC
BNE WH08

NEW 1.0 MILE FOR TOTAL MILES
JSR INITTM OKEMI --> TOTMUC
LDX #TOTMI INCR TOTAL MILES CTR
JSR DINC

WH08 DEC ASMUCT DECR AVERAGE SPEED UPDATE CTR
BNE WH01
LDA TENTHNM REINIT
STA ASMUCT

ADD 1.1 MILE TO AVERAGE SPEED DISTANCE
LDX #ASDMI GO INCREMENT AVG. SPEED DISTANCE
JSR DINC

DO METRIC UNITS NOW

WH01 EQU *
DEC KRCTR DECR KM REVOLUTION CTR.
BNE WH10 BRANCH IF NOT ZERO

NEW 1.1 KM FOR TRIP ODO
LDA TENTHNM .1KM CONSTANT --> KM REV. CTR.
STA KRCTR
LDX #TKCTR INCR. TRIP KM CTR.
JSR DINC

WH10 LDX #TOTKUC DECREMENT TOTAL KM UPDATE CTR
JSR DDEC
DNE WH04

```

01335 *
01336A *
01337A *
01338A *
01339 *
01340A *
01341A *
01342A *
01343A *
01344A *
01345A *
01346A *
01347 *
01348A *
01349 *
01350 *
01351 *
01352 *
01353 *
01354 *
01355 *
01356 *
01357 *
01358 *
01359 *
01360 *
01361 *
01362 *
01363 *
01364 *
01365 *
01366 *
01367 *
01368 *
01369 *
01370 *
01371 *
01372 *
01373A *
01374A *
01375 *
01376 *
01377 *
01378 *
01379A *
01380A *
01381A *
01382A *
01383A *
01384A *
01385A *
01386A *
01387A *
01388A *
01389A *
01390 *
01391 *
01392 *

```

NEW 1.0 KM FOR TOTAL MILES
JSR INITTK OKEMK --> TOTKUC
LDX #TOTKM INCREMENT TOTAL KM CTR.
JSR DINC

WH04 DEC ASKUCT DECR AVG. SPEED KM UPDATE CTR
BNE WH02
LDA TENTHNM
STA ASKUCT

INCREMENT AVERAGE SPEED KM TOTAL
LDX #ASDKM GO INCREMENT AVG. SPEED DISTANCE
JSR DINC

WH02 EQU *
RTI ADIOS

3 B MSEC REQUEST ROUTINE

THIS IS NOT AN INTERRUPT; BUT IS SCHEDULED BY THE 1 MSEC INTE
EVERY 3B INVOCATIONS. THE
3B MSEC ROUTINE DEBOUNCES THE 'FUNCTION' AND 'RESET' BUTTONS.
CHANGES THE FUNCTION WHEN APPROPRIATE AND ENABLES THE WINDOW
RESETS WILL BE ALLOWED. IT DOES NOT DO ANYTHING WITH THE
FUNCTION OR RESET INPUTS -- IT PASSES THIS INFO ON TO THE 600
ROUTINE. THIS ROUTINE ALSO HANDLES THE TIME SAVED FOR AVERAGE
SPEED CALCULATIONS. LASTLY, EVERY 20 INVOCATIONS IT SCHEDULES
600 MSEC ROUTINE.

AVERAGE EXECUTION TIME = 87 CYCLES (348USEC)
WORST CASE TIME = 279 CYCLES (1116 USEC)

T30MS EQU *
BCLR RQST30,STAT2 CLEAR REQUEST BIT
BCLR STNDBY,STAT1,T3014 SKIP IF NOT IN STANDBY MODE

IN STANDBY, CHECK FOR STOP MODE
(TRIPLE DECREMENT STPCTR)

TST STPCTR+2 WILL LSB CAUSE BORROW
BNE T3020 BRANCH IF NOT
LDX #STPCTR GO DECREMENT 2 MSB'S
JSR DDEC

T3020 DEC STPCTR+2 DECR LSB
BNE T3014 BRANCH IF NOT NOW ZERO
LDA STPCTR+1 TEST FOR ALL 3 BYTES = ZERO
ORA STPCTR
BNE T3014

GO INTO STOP MODE
BSET STOPTI,STAT1

DO DEBOUNCE OF FUNCTION KEY

```

01393      02B8      A T3014 EQU *
01394A 02B8 02 00 02C9 BRSET FNKEY,IOX,T3081 BRANCH IF FUNCTION KEY NOT DOWN
01395A 02B8 00 10 04 02C5 BRCLR FUNDWN,STAT1,T3080 BRANCH IF KEYDOWN STATUS NOT SET
01396A 02C1 10 10 A BSET WTFUN,STAT1 SET WAITING FOR FUNCTION UP
01397A 02C3 20 1F 02E4 BRA T3084
01398      02C5      A T3080 EQU * HERE WHEN FUNCTION KEY JUST WENT DOWN
01399      02C5      A T3080 EQU *
01400A 02C5 1A 10 A BSET FUNDWN,STAT1
01401A 02C7 20 1B 02E4 BRA T3084
01402      02C9      A T3081 EQU * HERE WHEN FUNCTION KEY NOT DOWN
01403      02C9      A T3081 EQU *
01404A 02C9 1B 10 A BCLR FUNDWN,STAT1 CLEAR KEY DOWN STATUS
01405A 02C9 09 10 16 02E4 BRCLR WTFUN,STAT1,T3083 BRANCH IF NOT WAITING FOR KEY UP
01406      02C8 19 10 A * KEY WAS DOWN AND IS NOW BACK UP. WE HAVE A NEW INPUT
01407A 02C8 19 10 A BCLR WTFUN,STAT1 CLEAR WAITING FOR FUNCTION KEY UP
01408A 02D0 19 11 A BCLR WTSET2,STAT2 CLEAR WAITING FOR NEW WHEEL SIZE
01409A 02D2 8D CA A JSR MOSTBY CLEAR STANDBY
01410A 02D4 86 25 A LDA FUNCT INCREMENT THE FUNCTION NUMBER
01411A 02D6 4C A INCA
01412A 02D7 A1 06 A CMP #MAXFUN FUNCTION WRAP AROUND?
01413A 02D9 23 01 02DC BLS T3082 BRANCH IF NOT
01414A 02D8 4F A CLRA
01415      02DC      A T3082 EQU *
01416A 02DC 87 25 A STA FUNCT STORE NEW FUNCTION
01417A 02DE 12 10 A BSET SETENB,STAT1 ENABLE RESETS FOR ABOUT 5 SEC
01418A 02E0 A6 00 A LDA #8 8 --> RESET ENABLE CTR.
01419A 02E2 87 44 A STA SECTR
01420      02E4      A T3083 EQU *
01421      *
01422      * DEBOUNCE RESET KEY
01423      *
01424      02E4      A T3084 EQU *
01425A 02E4 04 02 00 02F2 BRSET RSTKEY,IOX,T3087 BRANCH IF RESET KEY UP
01426A 02E7 07 10 04 02EE BRCLR SETDWN,STAT1,T3085 BRANCH IF KEY DOWN STATUS NOT SET
01427A 02E8 14 10 A BSET WTSET,STAT1 SET WAITING FOR RESET KEY UP
01428A 02EC 20 12 0300 BRA T3089
01429      02EE      A T3085 EQU * RESET KEY JUST WENT DOWN
01430      02EE      A T3085 EQU *
01431A 02EE 16 10 A BSET SETDWN,STAT1 SET RESET KEY DOWN
01432A 02F0 20 0E 0300 BRA T3089 WAIT UNTIL IT COMES UP TO DO ANYTHING
01433      02F2      A T3087 EQU * HERE ON RESET KEY NOT DOWN
01434      02F2      A T3087 EQU *
01435A 02F2 17 10 A BCLR SETDWN,STAT1 CLEAR RESET KEY DOWN STATUS
01436A 02F4 05 10 09 0300 BRCLR WTSET,STAT1,T3088 BRANCH IF NOT WAITING FOR KEY UP
01437A 02F7 15 10 A BCLR WTSET,STAT1 CLEAR WAITING FOR KEY UP
01438A 02F9 8D CA A JSR MOSTBY CLEAR STANDBY
01439A 02FB 03 10 02 0300 BRCLR SETENB,STAT1,T3088 SKIP IF RESET NOT ENABLED
01440A 02FE 10 11 A BSET DOSET,STAT2 SET BIT SO 600MSEC WILL DO RESET
01441      0300      A T3088 EQU *
01442      *
01443      * UPDATE THE TIME FOR AVERAGE SPEED CALCULATIONS
01444      *
01445      0300      A T3089 EQU *
01446A 0300 A2 42 A LDX #ASTUPD DECR. AVG. SPEED UPDTE CTR.
01447A 0302 8D D1 A JSR DDEC
01448A 0304 26 10 031E BNE T3010 BRANCH IF NOT ZERO
01449A 0306 8D 9F A JSR CLRASR 750 --> AVG. SPEED UPDATE CTR.
01450A 0308 A2 1E A LDX #ASTIME INCR AVERAGE SPEED TIME

```

```

01451A 030A 8D C4 A JSR DINC
01452      * SNAPSHOT THE CURRENT AVERAGE SPEED DISTANCES
01453A 030C 9B A SEI
01454A 030D 86 1A A LDA ASDMI STORE CURRENT VALUES IN LAST VALUES
01455A 030F 87 20 A STA LASDMI
01456A 0311 86 1B A LDA ASDMI+1
01457A 0313 87 21 A STA LASDMI+1
01458A 0315 86 1C A LDA ASDKM
01459A 0317 87 22 A STA LASDKM
01460A 0319 86 1D A LDA ASDKM+1
01461A 031B 87 23 A STA LASDKM+1
01462A 031D 9A A CLI
01463      031E      A T3010 EQU *
01464      *
01465      * SEE IF WE NEED TO SCHEDULE THE 600MSEC ROUTINE
01466      *
01467A 031E 3A 3F A DEC SCUPD DECR SPEED/CADENCE UPDATE CTR.
01468A 0320 26 06 0320 BNE T3012 BRANCH IF NOT 600 MSEC YET
01469A 0322 16 11 A BSET ROS600,STAT2 REQUEST 600 MSEC ROUTINE
01470A 0324 A6 14 A LDA #20 20 --> SPEED/CADENCE UPDATE CTR.
01471A 0326 87 3F A STA SCUPD
01472      0328      A T3012 EQU *
01473A 0328 81 A RTS
01474      *
01475      *
01476      *
01477      * 6 0 0 M S E C R O U T I N E
01478      *
01479      *
01480      *
01481      * THIS ROUTINE IS SCHEDULED EVERY 600MSEC BY THE 30MSEC ROUTINE
01482      * ITS MAIN FUNCTION IS TO CALCULATE NEW VALUES TO DISPLAY OR
01483      * TO EXECUTE A "RESET" FOR THE CURRENT FUNCTION. THE ROUTINE
01484      * USES VALUES STORED AND UPDATED BY THE 1MSEC, 30MSEC AND
01485      * WHEEL INTERRUPT ROUTINES TO CALCULATE THE CURRENTLY SELECTED
01486      * OUTPUT FUNCTION. IF RESET IS ENABLED, THE FUNCTION ABBREVIATI
01487      * IS OUTPUT TO THE DISPLAYS. IF RESET IS NOT ENABLED, THE ROUTI
01488      * DETERMINES THE ACTIVE FUNCTION AND DISPLAYS THE CORRECT VALUE
01489      *
01490      *
01491      * IT IS DIFFICULT TO CALCULATE ALL THE POSSIBLE TIMINGS THRU
01492      * THIS PACKAGE SO, FIRST, I CALCULATED SOME APPROXIMATE VALUES
01493      * FOR NORMAL SORT OF DISPLAY OPTIONS (NO RESET BUTTON PUSHED).
01494      * THEN I FIGURED THE VERY WORSE CASE WHICH IS WHEN A NEW WHEEL
01495      * CIRCUMFERENCE IS ENTERED AND THAT ENTRY CAUSES BOTH MULTIPLY
01496      * AND DIVIDE TO GO THRU EVERY POSSIBLE LOOP (HIGHLY UNLIKELY).
01497      * THE RESULTS ARE:
01498      *
01499      * APPROXIMATE EXECUTION SPEEDS FOR NORMAL DISPLAYS --
01500      *
01501      * DISPLAY SPEED,AVERAGE SPEED, OF CADENCE -- 2000 CYCLES AVG.
01502      * DISPLAY TRIP OR TOTAL DISTANCE -- 3100 CYCLES MAX
01503      * 1400 CYCLES AVG
01504      * 1800 CYCLES MAX
01505      *
01506      * APPROXIMATE EXECUTION TIME FOR WORST CASE -- 10000 CYCLES
01507      *
01508      *

```

PAGE 020 BICYCLE .SA:1

MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

150

```

01625      *      TABLE OF LCD VAULES TO OUTPUT WHILE RESET IS ENABLED
01626      *      ORDER IS: PORTD, PORTC, PORTB, PORTA
01627      *      LCD4 CONTAINS COM AND THE DECIMAL POINT TOO
01628      *
01629      *
01630      A SETTAB EQU      *      ORDERED BY FUNCTION NUMBER
01631A 03B3 70      A      FCB $78,$80,$81,$F2 SPEED 'SP'
01632A 03B7 7F      A      FCB $77,$80,$2B,$F2 AVERAGE SPEED 'ASP'
01633A 03BB 91      A      FCB $91,$80,$3F,$8E TRIP ODOMETER 'ODO'
01634A 03BF 89      A      FCB $89,$80,$39,$EC TOTAL DISTANCE 'DIS'
01635A 03C3 F6      A      FCB $F6,$80,$35,$9E CADENCE 'CAD'
01636      * NEW FUNCTIONS SHOULD BE ADDED BEFORE HERE
01637A 03C7 FF      A KMTAB FCB $FF,$80,$7F,$FE UNITS=KM '--- 1888'
01638A 03CB FF      A MITAB FCB $FF,$80,$7F,$FE UNITS=MILE '1888'
01639      *
01640      *
01641      *
01642      *
01643      * DO RESET ACTION
01644      *
01645      *
01646      *
01647      *
01648A 03CF 03CF      A T600MX EQU      *
01649A 03CF 11 11      A      BCLR DOSET,STAT2 CLEAR BIT
01650A 03D1 B6 25      A      LDA FUNCT GET FUNCTION NUMBER
01651A 03D3 48          A      LSLA          *3
01652A 03D4 B8 25      A      ADD FUNCT
01653A 03D6 87          A      TAX
01654A 03D7 9B          A      SEI          BEST TO DISABLE INTERRUPTS
01655A 03DB DD 03DF      A      JSR T600RS,X JUMP TO RESET
01656A 03DB 9A          A      CLI          RE-ENABLE INTERRUPTS
01657A 03DC CC 0557      A      JMP D600X GO EXIT
01658      *
01659      *
01660      *
01661      *      RESET SUBROUTINE CALLS
01662      *
01663      *
01664      *
01665      * INSTANTANEOUS SPEED
01666A 03DF 03DF      A T600RS EQU      *
01667A 03DF 81          A      RTS
01668A 03E0 9D          A      NOP          PAD NOP'S TO MAKE EACH SET OF
01669A 03E1 9D          A      NOP          RESET CODE 3 BYTES LONG
01670      * AVERAGE SPEED
01671A 03E2 BD 00      A      JSR CLRAS
01672A 03E4 81          A      RTS
01673      * TRIP DISTANCE
01674A 03E5 AD 14      03FB      BSR CLRTRP
01675A 03E7 81          A      RTS
01676      * TOTAL DISTANCE
01677A 03E8 BD A8      A      JSR CLRDIS
01678A 03EA 81          A      RTS
01679      * CADENCE
01680A 03EB 81          A      RTS
01681A 03EC 9D          A      NOP
01682A 03ED 9D          A      NOP

```

```

01683      * KM/MI SET
01684A 03EE AD 04      03F4      BSR FK0MI
01685A 03F0 9D          A      NOP
01686      * WHEEL SIZE
01687A 03F1 AD 19      040C      BSR NEWSWS
01688A 03F3 81          A      RTS
01689      *
01690      *
01691      *
01692      * RESET SUBROUTINES
01693      *
01694      *
01695      *
01696      * FLIP KILOMETER/ MILE FLAG
01697      *
01698      A FK0MI EQU      *
01699A 03F4 B6 11      A      LDA STAT2 FLIP KIOMILE BIT
01700A 03F6 A8 40      A      EOR $BIT6
01701A 03F8 B7 11      A      STA STAT2
01702A 03FA 81          A      RTS
01703      *
01704      * INITIALIZE TRIP ODO
01705      *
01706      A CLRTRP EQU      *
01707A 03FB 3F 12      A      CLR TMCTR $ --- TRIP MILES CTR.
01708A 03FD 3F 13      A      CLR TMCTR+1
01709A 03FF 3F 14      A      CLR TKCTR $ --- TRIP KM CTR.
01710A 0401 3F 15      A      CLR TKCTR+1
01711A 0403 B6 26      A      LDA TENTHMK RESET REV/.1 MILE CONSTANT
01712A 0405 B7 27      A      STA MKCTR
01713A 0407 B6 2A      A      LDA TENTHKT RESET REV/.1 KM CONSTANT
01714A 0409 B7 2B      A      STA KRCTR
01715A 040B 81          A      RTS
01716      *
01717      * DO THE 'SET' FUNCTION FOR WHEEL SIZE
01718      *
01719      *
01720      * TWO FLAGS DETERMINE WHAT THIS ROUTINE DOES
01721      *
01722      *      DOSET WTSET2
01723      *      0 0 DISPLAY CIRCUMFERENCE (DONE ABOVE)
01724      *      0 1 WAITING FOR NEW WHEEL SIZE (DONE ABOVE)
01725      *      1 0 INITIALIZE FOR NEW WHEEL SIZE INPUT
01726      *      1 1 HAVE NEW CIRCUMFERENCE. CALCULATE NEW VALUES
01727      *
01728      * ONLY THE CASES WITH DOSET=1 ARE HANDLED IN THIS ROUTINE. THE OTHER
01729      * CASES WERE HANDLED ABOVE DURING THE DISPLAY CODE.
01730      *
01731      A NEWSWS EQU      *
01732A 040C 80 11 14 0423      BSET WTSET2,STAT2,WS10 SKIP IF NEW VALUE
01733      * INIT FOR NEW WHEEL SIZE INPUT
01734A 040F 18 11      A      BSET WTSET2,STAT2 SET FLAG (WAITING)
01735A 0411 A6 4F      A      LDA $79 START WITH CIRCUMFERENCE = 39.5 INCHES
01736A 0413 B7 2P      A      STA TCIR2
01737A 0415 A6 81      A      LDA $395/$256 INIT DISPLAY
01738A 0417 A6 08      A      LDY $395/$FF
01739A 0419 B7 37      A      STA TDISC
01740A 041B BF 30      A      STX TDISC+1
01741A 041D A6 02      A      LDA $2 INIT EVERY OTHER CTR

```

```

PAGE 031 BICYCLE.SA:1 MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

01741A 041F B7 45 A STA WHCTR
01742A 0421 20 7C 049F BRA WSOUT
01743 * HERE ON A NEW CIRCUMFERENCE. CALCULATE THE FOLLOWING VALUES:
01744 * CIRKM
01745 * TENTHM
01746 * ONEMI
01747 * TENTHK
01748 * ONEKMK
01749 * CMI
01750 * CKM
01751 *
01752 A WS10 EQU *
01753A 0423 19 11 A BCLR WTSET2,STAT2 RESET WAITING BIT
01754 *
01755 * CIRKM = (CIR2MI * 128) / 101
01756 *
01757A 0425 9A CLRI RE-ENABLE INTERRUPTS
01758A 0426 B6 37 A LDA TDISC MOVE THE TEMPORARY VALUES INTO THE NEW VALUES
01759A 0428 B7 35 A STA DISCIR
01760A 042A B6 38 A LDA TDISC+1
01761A 042C B7 36 A STA DISCIR+1
01762A 042E BE 2F A LDX TCIR2
01763A 0430 BF 2E A STX CIR2MI
01764A 0432 4F CLRA
01765A 0433 54 LSRX
01766A 0434 46 RORA *128
01767A 0435 BF 5F A STX DIVDND
01768A 0437 B7 60 A STX DIVDND+1
01769A 0439 3F 5D A CLR DIVSR
01770A 043B A6 65 A LDA #101 /101
01771A 043D B7 5E A STA DIVSR+1
01772A 043F BD E5 A JSR DIV
01773A 0441 B7 30 A STA CIRKM
01774 *
01775 * TENTHM = 12672 / CIR2MI
01776 *
01777A 0443 AE 31 A LDX #12672/256
01778A 0445 A6 80 A LDA #126721.5FF
01779A 0447 BF 5F A STX DIVDND
01780A 0449 B7 60 A STA DIVDND+1
01781A 044B 3F 5D A CLR DIVSR
01782A 044D B6 2E A LDA CIR2MI
01783A 044F B7 5E A STA DIVSR+1
01784A 0451 BD E5 A JSR DIV
01785A 0453 B7 26 A STA TENTHM
01786 *
01787 * ONEMI = TENTHM * 10
01788 *
01789A 0455 AE 0A A LDX #10
01790A 0457 BD CF A JSR MUL
01791A 0459 BF 28 A STX ONEMI
01792A 045B B7 29 A STA ONEMI+1
01793 *
01794 * TENTHK = 10000 / CIRKM
01795 *
01796A 045D AE 27 A LDX #10000/256
01797A 045F A6 10 A LDA #100001.5FF
01798A 0461 BF 5F A STX DIVDND

```

```

PAGE 032 BICYCLE.SA:1 MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

01799A 0463 B7 60 A STA DIVDND+1
01800A 0465 3F 5D A CLR DIVSR
01801A 0467 B6 30 A LDA CIRKM
01802A 0469 B7 5E A STA DIVSR+1
01803A 046B BD E5 A JSR DIV
01804A 046D B7 2A A STA TENTHK
01805 *
01806 * ONEKMK = TENTHK * 10
01807 *
01808A 046F AE 0A A LDX #10
01809A 0471 BD CF A JSR MUL
01810A 0473 BF 2C A STX ONEKMK
01811A 0475 B7 2D A STA ONEKMK+1
01812 *
01813 * CMI = (227*CIR2MI) / 8
01814 *
01815A 0477 A6 E3 A LDA #227
01816A 0479 BE 2E A LDX CIR2MI
01817A 047B BD CF A JSR MUL
01818A 047D 54 LSRX /8
01819A 047E 46 RORA
01820A 047F 54 LSRX
01821A 0480 46 RORA
01822A 0481 54 LSRX
01823A 0482 46 RORA
01824A 0483 24 04 0489 BCC WS01 ROUND RESULT
01825A 0485 4C 0489 INCA
01826A 0486 26 01 0489 BNE WS01
01827A 0488 5C INCX
01828A 0489 BF 31 A WS01 STX CMI
01829A 048B B7 32 A STA CMI+1
01830 *
01831 * CKM = 36 * CIRKM
01832 *
01833A 048D A6 24 A LDA #36
01834A 048F BE 30 A LDX CIRKM
01835A 0491 BD CF A JSR MUL
01836A 0493 BF 33 A STX CKM
01837A 0495 B7 34 A STA CKM+1
01838 *
01839 * SINCE WE CHANGED THE WHEEL SIZE, IT ONLY MAKES SENSE TO RESET ALL
01840 * DISTANCE AND AVERAGE SPEED COUNTERS
01841 *
01842A 0497 9B SEI
01843A 0498 BD 00 A JSR CLRAS AVERAGE SPEED
01844A 049A CD 03FB A JSR CLRTRP TRIP ODO
01845A 049D BD A8 A JSR CLDIS TOTAL DISTANCE
01846 A WSOUT EQU *
01847A 049F 01 RTS
01848 *
01849 *
01850 *
01851 * HERE TO DO THE NORMAL FUNCTION FOR THE REQUESTED DISPLAY
01852 *
01853 * IN MOST CALCULATION X CONTAINS THE MS BYTE
01854 *
01855 *
01856 *

```

```

01857      04A0      A DOFUNC EQU *
01858A 04A0 1D 10      A BCLR DECP,STAT1 CLEAR DECIMAL POINT
01859A 04A2 B6 25      A LDA FUNCT GET FUNCTION NUMBER
01860A 04A4 26 20      04C6 BNE DOF10 BRANCH IF NOT ZERO (SPEED)
01861      *
01862      * CASE FUNCTION OF:
01863      *
01864      *
01865      *
01866      * INSTANTANEOUS SPEED
01867      *
01868      *
01869      *
01870A 04A6 3D 49      A TST SPDZER IS SPEED ZERO
01871A 04A8 27 04      04AE BEQ DOF02 BRANCH IF NOT
01872A 04AA 5F          CLRX CLRA SET NUMERATOR TO ZERO
01873A 04AB 4F          BRA DOFSPD
01874A 04AC 2D 0D      04BB *
01875      *
01876      A DOF02 EQU *
01877A 04AE 0C 11 06 04B7 BRSET KMMILE,STAT2,DOF01 BRANCH IF KM
01878      * MILES S = CM1/T
01879A 04B1 8E 31      A LDX CM1 GET MILE SPEED
01880A 04B3 B6 32      A LDA CM1+1
01881A 04B5 28 04      04BB BRA DOFSPD
01882      * KILOMETER S = CKM/T
01883A 04B7 8E 33      A DOF01 LDX CKM GET KM SPEED CONSTANT
01884A 04B9 B6 34      A LDA CKM+1
01885      * ALL SPEED CODE MERGES HERE TO DO CALCULATION
01886      A DOFSPD EQU *
01887A 04BB BF 5F      A STX DIVDND STORE THE CONSTANT IN THE DIVIDEND
01888A 04BD B7 60      A STA DIVDND+1
01889A 04BF 9B          SEI DISABLE INTERRUPTS A USEC OR TWO
01890A 04C0 BE 4B          LDX GET DIVISOR
01891A 04C2 B6 4C          A LDA TBWHL+1
01892A 04C4 20 6C      0532 BRA DOF01V GO DO DIVIDE AND DISPLAY RESULT
01893      *
01894      * CASE OF AVERAGE SPEED?
01895      *
01896      *
01897      *
01898A 04C6 4A          DOF10 DECA FUNCTION = AVERAGE SPEED?
01899A 04C7 26 21      04EA BNE DOF20
01900      * AVERAGE SPEED
01901A 04C9 9B          SEI TURN OFF INTERRUPTS
01902A 04CA 0C 11 06 04D3 BRSET KMMILE,STAT2,DOF11 BRANCH IF KM
01903      * MILES
01904A 04CD BE 20      A LDX LASDMI ASDMI*16/ASTIME
01905A 04CF B6 21      A LDA LASDMI+1
01906A 04D1 20 04      04D7 BRA DOF12
01907      * KM
01908A 04D3 BE 22      A DOF11 LDX LASDKM ASDKM*16/ASTIME
01909A 04D5 B6 23      A LDA LASDKM+1
01910      *
01911      A DOF12 EQU *
01912A 04D7 99          SEC ADD .05 MILES TO DISTANCE (AVERAGE ERROR)
01913A 04D8 49          ROLA *16
01914A 04D9 59          ROLX

```

```

01915A 04DA 48          LSLA
01916A 04DB 59          ROLX
01917A 04DC 48          LSLA
01918A 04DD 59          ROLX
01919A 04DE 48          LSLA
01920A 04DF 59          ROLX
01921A 04E0 BF 5F      A STX DIVDND
01922A 04E2 B7 60      A STA DIVDND+1
01923A 04E4 BE 1E      A LDX ASTIME GET AVERAGE SPEED TIME
01924A 04E6 B6 1F      A LDA ASTIME+1
01925A 04E8 20 48      0532 BRA DOF01V GO DO DIVIDE AND DISPLAY RESULT
01926      *
01927      * CASE OF TRIP DISTANCE (ODO) ?
01928      *
01929      *
01930      *
01931A 04EA 4A          DOF20 DECA FUNCTION = TRIP DISTANCE
01932A 04EB 26 12      04FF BNE DOF30
01933      * TRIP DISTANCE
01934A 04ED 1C 10      A BSET DECP,STAT1 SET DECIMAL POINT
01935A 04EF 9B          SEI DISABLE INTERRUPTS
01936A 04F0 0C 11 06 04F9 BRSET KMMILE,STAT2,DOF25 BRANCH IF KM MODE
01937      * MILES
01938A 04F3 BE 12      A LDX TMCTR
01939A 04F5 B6 13      A LDA TMCTR+1
01940A 04F7 20 57      0550 BRA DOF39 GO DISPLAY VALUE
01941      * KM
01942A 04F9 BE 14      A DOF25 LDX TKCTR
01943A 04FB B6 15      A LDA TKCTR+1
01944A 04FD 20 51      0550 BRA DOF39
01945      *
01946      * CASE FUNCTION OF TOTAL DISTANCE?
01947      *
01948      *
01949      *
01950A 04FF 4A          DOF30 DECA
01951A 0500 26 10      0512 BNE DOF40
01952      * TOTAL DISTANCE
01953A 0502 9B          SEI
01954A 0503 0C 11 06 050C BRSET KMMILE,STAT2,DOF35 BRANCH IF KM MODE
01955      * MILES
01956A 0506 BE 16      A LDX TOTM1
01957A 0508 B6 17      A LDA TOTM1+1
01958A 050A 20 44      0550 BRA DOF39
01959      * KM
01960A 050C BE 18      A DOF35 LDX TOTKM
01961A 050E B6 19      A LDA TOTKM+1
01962A 0510 20 3E      0550 BRA DOF39
01963      *
01964      * CASE FUNCTION OF CADENCE
01965      *
01966      *
01967      *
01968A 0512 4A          DOF40 DECA FUNCTION=CADENCE?
01969A 0513 26 2C      0541 BNE DOF50
01970A 0515 3D 4A      A TST CADZER IS CADENCE ZERO
01971A 0517 27 04      051D BEQ DOF41 BRANCH IF NOT
01972A 0519 5F          CLRX

```



```

PAGE 035 BICYCLE .SA:1          MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

01973A 051A 4F          CLRA          *
01974A 051B 28 84      0521 A DOP41 EQU          *
01975          051D          *
01976A 051D AE EA      A LDX          $68000/256 DIVIDEND = 68,000
01977A 051F A6 60      A LDA          $68000+1.SFF
01978          0521          *
01979A 0521 BF 5F      A DOP42 EQU          *
01980A 0523 B7 60      A STX          DIVDND
01981A 0525 98          A STA          DIVDND+1
01982A 0526 BF 4F      A LDA          TBCAD+1
01983A 0528 D6 58      A LDX          TBCAD          DIVISOR = CADENCE PERIOD
01984          * KLUDGE TO MAKE SURE CADENCE = ZERO IF TBCAD IS TOO BIG
01985A 052A A3 17      0532 A CPX          #23
01986A 053C 25 84      A BLO          DOP44
01987A 053D 3F 57      A CLR          DIVDND
01988A 0538 3F 60      A CLR          DIVDND+1
01989          0532 A DOP44 EQU          *
01990          *
01991          *
01992          * HERE TO STORE A,X INTO THE DIVISOR AND CALL THE DIVIDE ROUTINE
01993          *
01994          *
01995          0532 A DOPDIV EQU          *
01996A 0532 9A          CLI          RE-ENABLE INTERRUPTS
01997A 0533 BF 5D      A STX          DIVSR          STORE THE DIVISOR
01998A 0535 B7 5E      A STA          DIVSR+1
01999A 0537 BF 5E      A JSR          DIV          GO DO 16/16 BIT DIVIDE WITH ROUND 8 BIT ANS.
02000A 0539 3F 53      A CLR          DISPLY          CLEAR MSB OF DISPLAY VALUE
02001A 053B B7 54      A STA          DISPLY+1          STORE DIVIDE RESULT IN LSB
02002A 053D AD 19      0558 BSR          DOPDSP          GO DO THE DISPLAY
02003A 053F 20 16      0557 BRA          D680X
02004          *****
02005          *
02006          * CASE FUNCTION OF KMMILE?
02007          *
02008          *****
02009          0541 A DOP50 EQU          *
02010A 0541 4A          DECA
02011A 0542 26 86      054A BNE          DOP60
02012          * KM/MILE
02013A 0544 AE 07      A LDX          $1888/256 DISPLAY = 188R
02014A 0546 A6 60      A LDA          $1888+1.SFF
02015A 0548 20 06      0558 BRA          DOP39
02016          *****
02017          *
02018          * CASE FUNCTION OF WHEEL SIZE
02019          * ALSO JUMPED TO BY THE SET CODE
02020          *
02021          *****
02022          054A A DOP60 EQU          *
02023A 054A BE 35      A LDX          DISCR
02024A 054C B6 36      A LDA          DISCR+1
02025A 054E 1C 10      A BSET          DECP,STAT1 SET DECIMAL POINT
02026          *
02027          *
02028          * STORE A,X INTO THE DISPLAY VALUE
02029          * ALSO CALLED AS A SUBROUTINE FROM THE RESET CODE
02030          *

PAGE 036 BICYCLE .SA:1          MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

02031          *
02032          0558 A DOP39 EQU          *
02033A 0558 9A          CLI          RE-ENABLE INTERRUPTS
02034A 0551 BF 53      A STX          DISPLY
02035A 0553 B7 54      A STA          DISPLY+1
02036A 0555 AD 01      0558 BSR          DOPDSP          GO DISPLAY SPEED
02037          *
02038          * FINAL EXIT OF 600 MSEC ROUTINE
02039          *
02040          0557 A D680X EQU          *
02041A 0557 01          RTS          ADIOS
02042          *
02043          *****
02044          *
02045          * MAIN DISPLAY ROUTINE
02046          *
02047          * CONVERTS THE BINARY VALUE IN DSPLY,DSPLY+1 TO A 7
02048          * SEGMENT BCD DISPLAY FOR THE OUTPUT PORTS. USES LOCATIONS
02049          * TEMP,DIGCTR, DI-D4, PA-PD AND SAVEX.
02050          *
02051          *
02052          * AVERAGE EXECUTION = 777 CYCLES          OLD TIME WAS 717
02053          * WORST CASE = 7777 CYCLES          OLD TIME WAS 1266
02054          *****
02055          *
02056          0558 A DOPDSP EQU          *
02057A 0558 1F 11      A BCLR          LDZERO,STAT2 SET TO SUPPRESS LEADING ZEROES
02058A 055A 5F          CLRX          INIT INDEX INTO BCD DIVIDERS
02059          *
02060A 055B A6 FF      A DLOOP2 EQU          *
02061A 055D B7 61      A LDA          #-1          INIT CTR
02062          055F A DLOOP EQU          *
02063A 055F 3C 61      A INC          TEMP          INCR. COUNTER
02064A 0561 B6 54      A LDA          DISPLY+1          DISPLY - CONSTANT FROM DIVTAB
02065A 0563 D0 5FDF      A SUB          DIVTAB+1,X
02066A 0566 B7 54      A STA          DISPLY+1
02067A 0568 B6 53      A LDA          DISPLY
02068A 056A D2 5FDF      A SBC          DIVTAB,X
02069A 056D B7 53      A STA          DISPLY
02070A 056F 2A EE      055F BPL          DLOOP          STILL POSITIVE?
02071          * NOW TEMP = BCD DIGIT; BUT WE NEED TO ADD BACK TO DISPLY
02072A 0571 B6 54      A LDA          DISPLY+1
02073A 0573 D8 5FDF      A ADD          DIVTAB+1,X
02074A 0576 B7 54      A STA          DISPLY+1
02075A 0578 B6 53      A LDA          DISPLY
02076A 057A D9 5FDF      A ADC          DIVTAB,X
02077A 057D B7 53      A STA          DISPLY
02078A 057F B6 61      A LDA          TEMP
02079A 0581 5D          TSTX          PICK UP THE BCD DIGIT
02080A 0582 26 02      0586 BNE          DOPD1          IF THOUSANDS DIGIT THEN SET TO 0 OR 1
02081A 0584 A1 01      A AND          #1
02082A 0586 AD 58      0580 DOPD1 BSR          CVTBCD          GO CONVERT DIGIT TO LCD FORMAT
02083A 0588 54          LSRX          (MAKE BYTE INDEX)
02084A 0589 E7 55      A STA          DI,X          STORE IN TEMP TABLE
02085A 058B 58          LSLX
02086A 058C 5C          INCX          BUMP PTR TO NEXT TABLE ENTRY
02087A 058D 5C          INCX
02088A 058E A3 86      A CMPX          #6          DONE 3 DIGITS?

```

PAGE 037 BICYCLE .SA:1

MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

```

02089A 0598 26 C9 0558 BNE DLOOP2
02090A 0592 B6 54 A LDA DISPLY+1 IF SO DIGIT 4 IS IN THE REMAINDER
02091A 0594 1E 11 A BSET LDZERO,STAT2,DON'T SUPPRESS LAST ZERO IF PRESENT
02092A 0596 AD 48 05E9 BSR CVTBCD GO CONVERT DIGIT 4
02093A 0598 B7 58 A STA D4
02094
02095
02096
02097
02098
02099
02100
02101
02102
02103
02104A 059A B6 58 A LDA D4 GET DIGIT 3 DATA
02105A 059C 48 A LSLA POSITION IT
02106A 059D BD 10 01 05A1 BRCLR DECP,STAT1,DOFD3 ADD IN DECIMAL POINT IF ANY
02107A 05A0 4C INCA
02108A 05A1 B7 59 A DOFD3 STA PA PORTA IS DONE
02109
02110A 05A3 B6 56 A * WORK ON PORT B
02111A 05A5 5F A LDA D2 GET DIGIT 1 DATA
02112A 05A6 44 CLRX SHIFT 3 LSB'S OF DIGIT 1 INTO BOTTOM OF X
02113A 05A7 59 LSRA
02114A 05A8 44 ROLX
02115A 05A9 59 LSRA
02116A 05AA 44 ROLX
02117A 05AB 59 LSRA
02118A 05AC B6 57 A ROLX
02119A 05AE 44 A LDA D3 GET DIGIT 2 DATA
02120A 05AF 59 LSRA SHIFT 3 LSB'S OF DIGIT 2 INTO 3 LSB'S OF X
02121A 05B0 44 ROLX WHILE SHIFTING DIGIT 1'S BITS UP TOO.
02122A 05B1 59 LSRA
02123A 05B2 44 ROLX
02124A 05B3 59 LSRA
02125A 05B4 BF 5A A ROLX
02126A 05B6 B7 57 A STX PB SAVE DIGIT 1 AND DIGIT 2 INFO
02127A 05B8 B6 55 A STA D3 SAVE REST OF DIGIT TWO'S BITS
02128A 05BA A1 0C A LDA D1 SET X IN PB IF THOUSANDTHS DIGIT = 1
02129A 05BC 26 02 05C8 A CMP #50C (THIS IS WHAT A ONE LOOKS LIKE)
02130A 05BE 1C 5A A BNE DOFD4
02131
02132 A DOFD4 EQU *PB
02133
02134
02135
02136
02137A 05C0 B6 56 A
02138A 05C2 48 A LDA D2
02139A 05C3 A4 F0 A LSLA GET G1 F1 A1 B1 IN MSB'S
02140A 05C5 BB 57 A AND #5F0
02141A 05C7 AE 08 A ADD D3 THIS IS NOW THE INVERSE OF WHAT WE WANT
02142A 05C9 48 A LDX #8 FLIP IT
02143A 05CA 36 5C A LOFPX LSLA
02144A 05CC 5A A ROR PD AND PUT IT IN PD
02145A 05CD 26 FA 05C9 A DECX
02146
02147
02148
02149
02150
02151
02152A 05D7 AE 03 A BPL DOFD5
02153A 05D9 E6 59 A DOFD7 LDA PA,X
02154A 05DB F7 A STA PORTA,X
02155A 05DC 5A A BPL DOFD7
02156A 05DD 2A FA 05D9 A DOFD7
02157A 05DF B1 A RTS END OF DOFDSF
02158
02159
02160
02161
02162A 05E8 BF 62 A CVTBCD STX SAVEX SAVE CURRENT X
02163A 05E2 4D A TSTA IS DIGIT ZERO?
02164A 05E3 26 04 05E9 A BNE CVTB81 BRANCH IF NOT
02165A 05E5 0E 11 01 05E9 A BRSET LDZERO,STAT2,CVTB81 GO CONVERT IF NOT LEADING ZERO
02166A 05E8 01 A RTS LEADING ZERO. EXIT WITH A = 0
02167A 05E9 1E 11 A CVTB81 LDZERO,STAT2 NO MORE LEADING ZEROS
02168A 05E9 97 A TAX MAKE RCD DIGIT THE INDEX
02169A 05EC D6 05F2 A LDA SEVSEG,X GET SEVEN SEGMENT VALUE
02170A 05EF BE 62 A LDX SAVEX RESTORE X
02171A 05F1 B1 A RTS
02172
02173
02174
02175
02176
02177A 05F2 3F A SEVSEG FCB $3F 0
02178A 05F3 0C A FCB $0C 1
02179A 05F4 58 A FCB $58 2
02180A 05F5 5E A FCB $5E 3
02181A 05F6 6C A FCB $6C 4
02182A 05F7 76 A FCB $76 5
02183A 05F8 77 A FCB $77 6
02184A 05F9 1C A FCB $1C 7
02185A 05FA 7F A FCB $7F 8
02186A 05FB 7E A FCB $7E 9
02187
02188
02189
02190
02191
02192A 05FC 03E8 A DIVTAB FDB 1000
02193A 05FE 0064 A FDB 100
02194A 0600 000A A FDB 10
02195

```

PAGE 038 BICYCLE .SA:1

MC146805G2L1 EVALUATION ROM (C) MOTOROLA 1982

```

02147A 05CF 4F CLRA SEE IF W SET
02148A 05D0 BD 11 02 05D5 BRCLR KMMILE,STAT2,DOFD6 BRANCH IF ENGLISH UNITS
02149A 05D3 AA 00 A ORA #80 SET W
02150A 05D5 B7 58 A DOFD6 STA PC
02151
02152A 05D7 AE 03 A * WRITE PA-PD TO PORTA - PORTD
02153A 05D9 E6 59 A DOFD7 LDA PA,X
02154A 05DB F7 A STA PORTA,X
02155A 05DC 5A A BPL DOFD7
02156A 05DD 2A FA 05D9 A DOFD7
02157A 05DF B1 A RTS END OF DOFDSF
02158
02159
02160
02161
02162A 05E8 BF 62 A CVTBCD STX SAVEX SAVE CURRENT X
02163A 05E2 4D A TSTA IS DIGIT ZERO?
02164A 05E3 26 04 05E9 A BNE CVTB81 BRANCH IF NOT
02165A 05E5 0E 11 01 05E9 A BRSET LDZERO,STAT2,CVTB81 GO CONVERT IF NOT LEADING ZERO
02166A 05E8 01 A RTS LEADING ZERO. EXIT WITH A = 0
02167A 05E9 1E 11 A CVTB81 LDZERO,STAT2 NO MORE LEADING ZEROS
02168A 05E9 97 A TAX MAKE RCD DIGIT THE INDEX
02169A 05EC D6 05F2 A LDA SEVSEG,X GET SEVEN SEGMENT VALUE
02170A 05EF BE 62 A LDX SAVEX RESTORE X
02171A 05F1 B1 A RTS
02172
02173
02174
02175
02176
02177A 05F2 3F A SEVSEG FCB $3F 0
02178A 05F3 0C A FCB $0C 1
02179A 05F4 58 A FCB $58 2
02180A 05F5 5E A FCB $5E 3
02181A 05F6 6C A FCB $6C 4
02182A 05F7 76 A FCB $76 5
02183A 05F8 77 A FCB $77 6
02184A 05F9 1C A FCB $1C 7
02185A 05FA 7F A FCB $7F 8
02186A 05FB 7E A FCB $7E 9
02187
02188
02189
02190
02191
02192A 05FC 03E8 A DIVTAB FDB 1000
02193A 05FE 0064 A FDB 100
02194A 0600 000A A FDB 10
02195

```