Implementing an analog speedometer in STISIM Drive using Parallax BSTAMP microcontroller

J-F Tessier[1,2] M. Kaszap[1] M. Lavallière[1,2]
M. Tremblay[1,2] N. Teasdale[1,2]

[1]*Université Laval, Faculté de médecine, Division de Kinésiologie, PEPS, Québec, Canada*
[2]*Unité de recherche sur le vieillissement,*
Centre de recherche FRSQ du CHA universitaire de Québec
email: Normand.Teasdale@kin.msp.ulaval.ca

Abstract

In a non-instrumented cab, STISIM Drive software normally projects the speed of the vehicle through a dashboard presented on the simulation screen. This simulated dashboard can be displayed with several graphical options. In all cases, there is a loss of information arising from the road. A solution is to integrate the speedometer into a dashboard and to disable the simulated projection. This solution increases the virtual immersion of the driver and presents speed in a more realistic way. We are proposing a simple solution based on Parallax Inc. Basic Stamp microcontroller. In addition to its low cost and simplicity, this solution allows integration of other technical elements of the driving experience (e.g., activation of turn signals, horn, etc.).

Keywords – Simulator environment, odometer, microcontroller, visual immersion

1. Introduction

A perennial issue when using driving simulators involves the validity and fidelity of the simulator. Modern PCs and associated technologies (for instance, graphics card) have allowed to improve these aspects and to produce realistic scenes and sound effects to immerse the driver in the roadway environment [1]. When using a non-instrumented cab most software simulators projects the speed of the vehicle through a dashboard presented on the simulation screen. In STISIM Drive, several graphical options are available.

Figure 1 shows four different possible displays. In all cases there is a loss of information arising from the road since an analog or a digital speedometer presented with or without a cab structure is displayed in the bottom of the scenario projection. A solution is to integrate the speedometer into a genuine dashboard and to disable the projected dashboard and speedometer. This solution increases the virtual immersion of the driver and presents speed in a more realistic way.

There are different methods to achieve this goal. Mostly for standardization purposes, the method proposed by Systems Technology Inc. requires a digital to analog board, a voltage to frequency converter and a speedometer using a square wave frequency signal as input. The digital to analog board outputs a voltage signal proportional to the speed.

Fig. 1 - Four different displays of the odometer in STISIM Drive

This voltage signal is then converted into a square-wave frequency signal read by the speedometer. Although functional, this solution can be costly and does not offer additional options for increasing the virtual immersion of the driver. We are proposing a cost-effective alternative to implement an analog or a digital speedometer using a microcontroller. This solution offers the possibility to instrument other features such as turn indicators, horn or dashboard lighting.

2. Microcontroller

Microcontrollers are embedded computer systems and can be found in most electronic devices such as alarm clock, cell phones, Mp3 players. A microcontroller is a functional computer system on a single chip. It normally includes a processor, memory and most importantly programmable input/output (I/O) peripherals.

For this simulator project we adopted Parallax Inc. (www.parallax.com) BASIC Stamp microcontroller. There are several versions of the BASIC Stamp module. They differ with respect to the size of their memory and their execution speed. The BS2px24 module was selected for our project (4,000 instructions, 19,000 instructions per second). The microcontroller is programmed using a BASIC-like language (PBASIC) that includes a specific set of instructions to control the I/O pins and the serial port.

3. Step by step implementation

Figure 2 presents a flow-chart diagram of the processes for displaying speed on a genuine speedometer. Essentially, the microcontroller reads the speed of the vehicle from the serial port of the simulator computer and converts the serial data into a format readable by the microcontroller (3.1), transforms this value into a frequency signal that can be read by a servo motor and moves the needle of the odometer to a desired angular position (3.2). Additionally, two control loops are embedded to generate the proper light and sound signals when the turn indicator lever is activated (3.3). A detailed and commented PBASIC code is freely available upon request from the authors. Each specific process is further explained and commented below.

3.1. Serial port reading

This is the most complex process. It consists of reading the speed of the vehicle generated by STISIM Drive. Initially, the STISIM Drive serial communication protocol must be set to communicate with the microcontroller. Within STISIM Drive, this is done by selecting the «Options menu, Configuration window, Other panel tab and Serial Data Output». The following settings are needed for communicating with the microcontroller: Port (com1), Baud Rate (2400), Parity (None), Data Bits (8), Stop Bits (1). Also, the STISIM scenario must include a serial out command to output the speed through the serial port of the main STISIM Drive computer. In STISIM Drive, this command must be inserted at the beginning of the scenario (0, SOUT, 23 where 0 indicates to start the serial out command SOUT at distance 0 and 23 indicates to output variable number 23 which is speed in the unit defined in the scenario). The speed provided by STISIM Drive through the PC serial port cannot be read directly by the microcontroller. This difficulty arises because of a mismatch in the data format. STISIM Drive outputs data using a 32-bit floating point representation (Float 32) whereas the BSTAMP microcontroller uses a 16-bit integer representation.

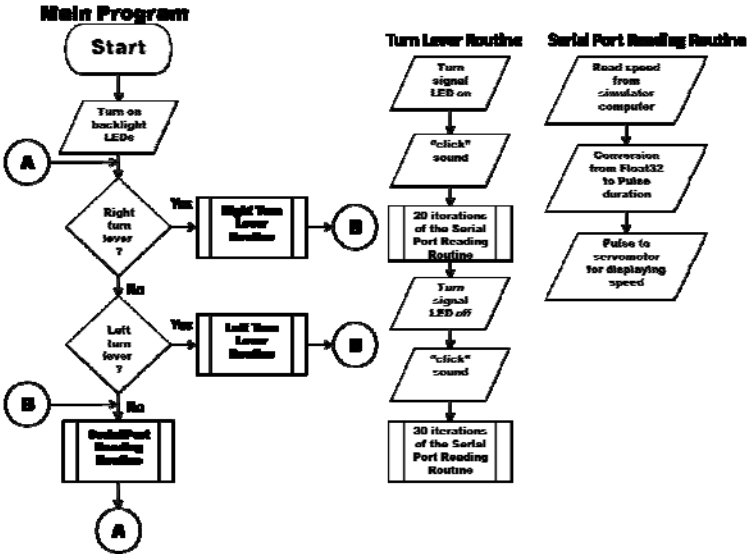

Fig. 2 - Flow-chart diagram of the microcontroller program. A and B are entrance and exit points. The turn lever and the serial port reading routines are further developed on the right portion of the figure

In STISIM Drive, up to 50 variables can be output through the serial port. In our particular case, we only selected to output the speed (km/h) (variable number 23). An explanation of the floating point to integer representation follows.Each speed value is preceded by 4 identical bytes (255) and an integer value to indicate the number of variables that are output. Hence, because only one variable is output in our project, a 5-byte header containing «255, 255, 255, 255, 1» is expected before each 32-bit speed value. The Institute of Electrical and Electronics Engineers (IEEE) has produced a standard for floating point arithmetic. A complete review is beyond the scope of this paper. Nevertheless, a brief description is necessary for understanding how speed values provided by STISIM Drive were converted into a 16-bit integer representation. Figure 3a shows that each 32-bit value received by the microcontroller is numbered from 0 to 31, right to left. Bit 31 is the sign bit 'S', bit 23 to 30 are the exponent bits 'E', and bit 0 to 22 are the fraction 'F' (also called the mantissa or significand). Because of the values expected from STISIM Drive (speed values varying from 0 to 256 km/hr), bit 0 to 15 can be neglected leaving us with bit 16 to 31 (Figure 3b). These 16 bits are read and stored in a 16-bit integer variable that needs to be deciphered to obtain the speed value. This process is now described. The last bit (bit 31) is the sign value and is always positive (0) for our project (speed from 0 to 256). The bits 23 to 30 represent the exponent (E) while the remaining 7 bits (bit 16 to bit 22) are the fraction (F). The most simple case is when E=0 and F is =0, then the speed is 0. For all other values, some arithmetic must be done and the speed = $2^{(E-127)}$ * (1.F). The necessary arithmetic for computing the speed can be best illustrated using examples.

Figure 3c illustrates for a selection of speeds, the speed value and the corresponding 32-bit floating point binary representation in memory and the 16-bit decimal representation. Let us start with a speed value of 1 km/hr (when the metric system is selected in STISIM Drive). The E value for 1 km/hr is 127 (or $2^0+2^1+2^2+2^3+2^4+2^5+2^6$) which leads to a value of 1 (that is, $2^{127-127}$ or 2^0). The fraction is 0 which leaves us with a value of 1.0. The speed is thus 1 km/hr (1*1.0 = 1.0). Let us now consider an example when the fraction is different than 0. For illustration purposes, we will find the E and F values for 110 km/hr. For this case, the exponent equals 133 (or $2^0+2^2+2^7$) which leads to an E value of 64 (that is, $2^{133-127}$ or 2^6). The fraction is represented by the following 7 bits (bit 16 to bit 22). The bits 18, 19, 20 and 22 are high. The value of each bit is defined as the fraction $1/2^n$ (starting with bit 22, in a reversed order). Hence, bit 22 represents a value of $1/2^1$, bit 20 is $1/2^3$, bit 19 is $1/2^4$ and bit 18 is $1/2^5$. F is computed by considering the sum of all 7 bits. To ease this task, it is best to find a common denominator (32 in this particular case, or 2^5). The sum of the fractions is thus ($1/2^1+1/2^3+1/2^4+1/2^5$, or 16/32+4/32+2/32+1/32 or 23/32). The speed value thus equals 110 km/hr (1.F*E or (1+23/32)*64).

The calculation process can be tedious. Figure 4 shows the relationship between speed and its 16-bit representation. A series of seven linear sections forming an irregular curved line covers the range 0 to 255 km/hr. For illustration purposes, only four sections are presented.

Fig. 3a - Illustration of the 32-bit floating point representation

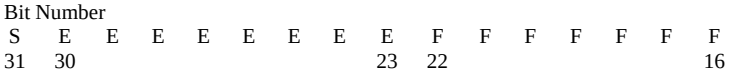

Fig. 3b - Illustration of the 16 bits considered for calculating speed after truncating the first 16 bits (bit 0 to bit 15)

	Bit Number																														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	DR
Speed	S	E								F																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16256
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16384
3	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16448
4	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16512
5	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16544
6	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16576
7	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16608
8	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16640
9	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16656
10	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16672
…																																	
100	0	1	0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17096
101	0	1	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17098
102	0	1	0	0	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17100
103	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17102
104	0	1	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17104
105	0	1	0	0	0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17106
106	0	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17108
107	0	1	0	0	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17110
108	0	1	0	0	0	0	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17112
109	0	1	0	0	0	0	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17114
110	0	1	0	0	0	0	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17116
…																																	
255	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17280

Fig. 3c - Table for the speed, 32-bit representation in memory and the corresponding 16-bit decimal representation (DR, rightmost column)

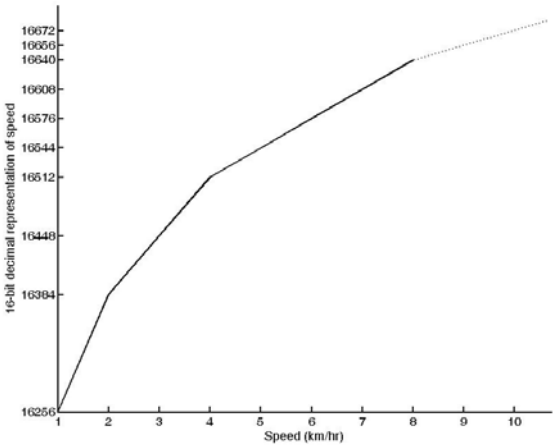

Figure 4: Relationship between speed (decimal system in km/hr) and the 16-bit decimal representation. Values for 1 to 8 km/hr only are presented

Note that the decimal value 16256 simply represents the 16-bit integer value for 1 km/hr (after truncating bit 0 to bit 15; 17280 represents a speed of 255 km/hr). To facilitate the calculation process we used seven first-order equations and the slope and origin of each of the seven sections served to calculate speed with the following formula (Speed = 16-bit decimal representation – origin of section/slope of section).

3.2. Speed conversion and Servo motor control

Once the speed is read by the microcontroller it can be displayed in a cockpit through an analog speedometer or through a LCD display. Our cockpit includes an analog speedometer and a needle moves to indicate the current speed. The speedometer we have covers 180 deg with each

degree representing one km/hr. To move the needle we used a Futaba standard servo motor allowing a range of 0 to 180 degrees. The needle of the speedometer is mounted on the rotating spindle of the motor. The motor is controlled by sending pulses of various durations through one I/O pin of the microcontroller repeatedly (PULSOUT command in PBASIC) until the desired position is achieved. Each position of the 180 degrees range of the motor is associated with a specific pulse duration.

3.3. Turn indicators and lighting

When a driving scenario is displayed with a projector, the room must be dark enough to enhance the quality of the projection. Therefore, the dashboard must be illuminated to be fully visible (similar to night vision conditions in a real car). This is achieved through a simple command line at the beginning of the microcontroller code which sets a number of I/O pins as output pins with a high value. These pins must be connected with a 220 Ohms resistor in series with the cathode of LEDs properly positioned into the dashboard to provide backlighting.

To increase the driver's immersion, we also implemented turn indicator signals. The turn indicator lever is instrumented with switches (for left and right turn signals). An upward or downward movement of the lever closes a circuit and activates a flashing light emitting diode (LED) and sound routine. During this routine, the LED (right or left turn indicator) oscillates at a low frequency and the speed value is adjusted whenever the flasher signal and the sound are in their off state.

4. Other possibilities

As mentioned in section 3.2, it is possible to present speed using a liquid crystal display (LCD) instead of an analog speedometer. This option (similar to a digital odometer) is easier to implement as once the speed is decoded it is simply a matter of displaying it on a LCD. It simply requires one line of code and there is no need to match the speed with a position value readable by a servo motor. Other sensors and signals can also be implemented and tailored for specific purposes. For instance, the activation of a radio button could serve to trigger a specific STISIM event by sending a digital signal to the I/O board of the STISIM computer. This could allow to examine distraction effects on the capacity to react to an event. Also, in STISIM Drive the temporal resolution is 30 Hz (33.33 ms) or less. Depending of the hardware configuration (graphics card), more detailed graphics and computations will slow down the temporal resolution. A microcontroller provides more precise timing routines (ms accuracy) that are not affected by the speed of the display. This may prove important for studies where precise timing is required (for instance, reaction time measurements).

Acknowledgement

Supported by grants from AUTO21 and the Natural Science and Engineering Research Council (NSERC) of Canada. Jean-François Tessier and Martin Lavallière were supported by graduate scholarships from the Centre d'excellence sur le vieillissement de Québec (CHA, CEVQ) and the Fonds de la recherche en santé du Québec (FRSQ), respectively.

References

1. Allen RW, Rosenthal TJ, Harmsen A, Markham S (1999) Low cost, PC based techniques for driving simulation implementation. In: Driving Simulation Conference (DSC) conference proceedings, Paris, France.