VirtualEngineIO Specification: A physical emulation of ECU commands for AVC

ES/Infotainment Division – ALTEN Sverige AB

Last Edit: February 20th 2019 – Ali Nakisai

VirtualEngineIO

For the development of a demo of the Instrument Cluster for the "ALTEN Virtual Cockpit" vehicle, we needed some sort of physical device, that could generate events and messages from an ECU. We came up with an Arduino Mega board that emulates some of the commands from an ECU. The piece of firmware is called **VirtualEngineIO**.

These commands have a simple customized format and are sent from the virtual ECU to the AGL system, using Serial port. These packets will be first received by a piece of software called the **VirtualEngine**. This software will convert the messages to equivalent CAN messages. It will as well generate fake "speed" and "RPM" values based on the status of the brake and gas 'pedals'.

This piece of documentation, describes the specifications that were in mind for the operation of the VirtualEngineIO.

What is emulated

The VirtualEngineIO firmware, emulates following four categories of signals:

(A) Signals regarding activation/deactivation of warning lights in the instrument cluster, including: (1) Left Blinker (2) Engine State Light (3) Oil Warning (4) ABS Brake Warning (5) Battery On (6) Seatbelt Warning (7) Doors Open Warning (8) Front Projector Indicator (9) Hand Brake Warning (10) Right Blinker

There are 10 physical buttons declared for each of these lights. Whenever a button is pressed (clicked), the MCU has to send a command to the host system (which is based on AGL).

(B) Acceleration and Brake 'pedals': There are two buttons defined for these two pedals. Whenever a button is pressed OR released, the MCU must inform the host system.

(C) Analog Gauges: There are three gauges defined: the fuel, the oil and the temperature gauge. These three gauges are simulated by three analog potentiometers. Their value is scaled in range of 0~100. Whenever a change occurs, the MCU has to inform the host system.

(D) Miscellaneous Buttons: That includes the buttons for engine ignition and hazard lights. The logic is same as **(A)**.

MCU Logic

It is evident that we have three types of Input signals:

- Digital signals which whenever a key is “pressed”, an event is sent.

We have 12 signals of this kind.

- Digital signals which whenever a key is pressed or released, an event is sent.

We have 2 signals of this kind

- Analog signals which must be taken in a scale of 0~100, which any change is reported by the MCU. This is done instead of reporting the values periodically, so to maximize performance and reduce overhead.

The commands are defined in the following section. The commands are send over a UART link (serial port), which for the Arduino Mega that we have, is carried over an FTDI USB-Serial converter.

Hardware for the digital signal: Each digital signal represents a button. The I/O pin of Arduino is internally pulled high and the button connects the pin to system GND (0V) when pressed. Therefore the key is **Active Low**. A logic NOT is placed to reverse the logic, so whenever the key is pressed, the memory values are set to **1** (and vice versa). The pin signal must be low-pass filtered in order to avoid ‘button bouncing effect’.

Hardware for the analog signal: Each analog signal represents an analog potentiometer (pot). The pot’s center leg is connected to the MCU. Two side legs are connected to VCC (5V) and GND (0V) respectively. This allows generation of a variable voltage in range of 0~5V on the analog pin. Capacitors could be added for better noise immunity. The voltage is read in range of 0~1023 by the MCU and is converted to range of 0~100. There is a simple digital ‘averaging’ mechanism in place to overcome and reduce the noise effects.

The processing loop: The input processing routine (IPR) occurs at least every 1 millisecond. So the maximum number of times the IPR is run is 1000 per second. The IPR basically has very small overhead, but in some iterations it may take longer and the number of times the IPR is run per second maybe less than 1000 (because of reasons such as UART message composition overhead, or IRQ handling delays).

Digital Signals: Last ‘stable’ status of each digital signal is stored. Whenever a change is detected (e.g. transition from 0 to 1 or vice versa), the change must be retain for a fixed amount of time called **Debouncing** time. The idea is to filter short-term noises or button bouncing effects. If the change is retained for the span of the **Debouncing** time, the change is recorded as a the new stable status of the key. An event is generated and send over the

UART link. For first type of digital signals, only transitions to 1 are reported (when the key is pressed).

The debouncing time is currently 20ms. It is evident that is it takes 20 IPRs or less before the change is considered stable. It is the actual time (in ms) that matters not the number of IPR iterations.

Analog Signals: Last status of each analog signal is stored. There is an averaging system in place to reduce the noise on analog signals. There is a global counter called the 'filter counter'. On every IPR iteration, the value of each analog signal is read by the Analog-Digital converter (ADC) and added to an accumulating register. After a fixed number of iterations (**Averaging Iterations**), the accumulator is divided by the number of samples and a 'low-pass filtered' value is calculated. This value is in range of 0~1023 and will be converted linearly to range of 0~100. For every analog signal, if there is a change in comparison to last status, the change is reported.

Current value for **Averaging Iterations** is 256. This number allows for low-level arithmetic substitution of division operator to right shift operator.

Contrary to the digital signals, it is the number of IPRs that matters in counting the averaging iterations, not the time that actually passes. Therefore, the updates may occur at least every 256 x 1ms = 256ms or slightly higher.

<NOTE> When the MCU is reset, all values are reset to 0 and respective events are generated.

Command Format

Commands are sent over UART. The baud rate is 115200.

For each category of the four aforementioned categories the command is listed as follows.

A) When a button related to one of the 10 warning lights is 'pressed':

```
ICON $N<\n>
```

Where $N is index of the icon (0~9) and <'\n'> is the line feed (character code 10).

B) When a button related to the acceleration/brake pedal is pressed.

```
ACCL $V<\n>

BREK $V<\n>
```

Where $V is a single character 'P' for 'pressed' and 'R' for 'released'.

C) When an analog signal value is changed:

```
GAUG $C $V<\n>
```

Where $C is type of the gauge: ‘F’ for fuel, ‘O’ for oil and ‘T’ for temperature.

Also $V is the value from 0 to 100.

D) When the ignite or hazard light buttons are pressed.

```
IGNT<\n>
HZRD<\n>
```

Wiring List

In current code, the pinout is as follows. The hardware for each type of pin is specified above and also in the following diagrams.

```
For digital pins which are connected to a button.

                                     /---\
  Arduino Digital Pin ---------------| O |-------------| GND
                                     \---/           -----
                                                      ---
                                                       -

For analog pins which are connected to a potentiometer.
                                          _
                                      ___|_|___
                                     |         |
  Arduino Analog Pin ------------    |_2__1__3_|
                                |       |  |  |
                                -------|  |  |---------| GND
                                         VCC         -----
                                                      ---
                                                       -

List of pins:
     Name                 Arduino Digital/Analog Pin    Event Name
     Left Blinker         22                            ICON 0
     Engine Warning       24                            ICON 1
     Oil Warning          26                            ICON 2
     ABS Brake Warn.      28                            ICON 3
     Battery On           30                            ICON 4
     Seatbelt             32                            ICON 5
     Doors                34                            ICON 6
     Front Projector      36                            ICON 7
     Hand Brake           38                            ICON 8
     Right Blinker        40                            ICON 9

     Accelerator          51                            ACCL
     Brake                53                            BREK

     Fuel Gauge           0                             GAUG F
     Oil Gauge            1                             GAUG O
     Thermo. Gauge        2                             GAUG T

     Ignition Key         50                            IGNT
     Hazard Light Key     52                            HZRD
```