Литература

1. Зимелев, Г. В. Теория автомобиля / 2-е изд., перераб. – Москва : военное издательство министерства обороны СССР, 1959. – 454с.

2. Сафиуллин, Р. Н. Эксплуатация автомобилей : учебник для вузов / Р. Н. Сафиуллин, А. Г. Башкардин – 2-е изд., испр. и доп. – Москва : Издательство Юрайт, 2019. – 204 с.

3. Тарг, С. М. Краткий курс теоретической механики: учеб. для втузов. –10-е изд., перераб. и доп. – М. : Высш. шк., 1986. – 416 с.

4. Журавлев, В. Ф. Основы теоретической механики : учебник / В. Ф. Журавлев. – 3-е изд., перераб. – Москва : Физматлит, 2008. – 304 с.

5. Хусаинов, А. Ш. Теория автомобиля. Конспект лекций / А. Ш. Хусаинов, В. В. Селифонов – Ульяновск: УлГТУ, 2008. – 121 с.

УДК 007.681.5

СРЕДСТВА ИЗУЧЕНИЯ CAN ШИНЫ АВТОМОБИЛЯ

Магистрант гр. 501140-20 Седяко П. В.

Научный руководитель – канд. техн. наук, доц. Гурский А. С.

Работа автомобиля постоянно контролируется по физическим параметрам (температура охлаждающей жидкости, расход воздуха, частота вращения коленчатого вала и др.); такие измерения производятся с помощью электронных датчиков, преобразующих измеряемую величину в другую величину удобную для передачи и обработки главным блоком управления.

Во всех современных автомобилях каждая система (система питания, тормозная система, система освещения и др.) может иметь свой собственный блок управления, что приводит к их большому количеству в автомобиле в связи с чем применяется CAN шина для их связи между собой. Так CAN шина связывает приборные панели, блоки управления коробкой передач, блоки климат контроля и блоки прочих систем, что делает ее наиболее подходящим источником получения информации об автомобиле во время его работы.

CAN-шина. CAN (*Controller Area Network* – сеть контроллеров) представляет собой стандарт промышленной сети, ориентированный, прежде всего, на объединение в единую сеть различных

устройств. Режим передачи – последовательный, широковещательный, пакетный.

Протокол CAN разработан компанией Robert Bosch GmbH в середине 1980-х и в настоящее время широко распространён в системах промышленной автоматизации, домашней автоматизации, автомобильной промышленности и многих других сферах. На практике под CAN-сетью обычно подразумевается сеть топологии «шина», представленная в виде двух проводов витой пары по которым предаются симметричные импульсы. Один провод имеет высокое состояние CAN High, а другой – низкое CAN Low [1].

В общем виде протокол передачи данных по CAN-шине состоит из кадров, которые принимаются всеми узлами сети (рисунок 1).

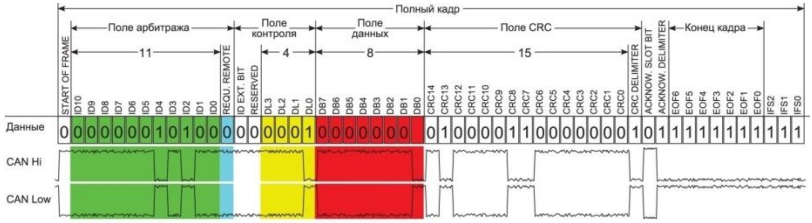

Рисунок 1 – Кадр данных

Существует два формата кадров данных: базовый (с 11-битными идентификаторами) и расширенный (с 29 битными идентификаторами).

Базовый кадр состоит из стартового бита (Start of frame) который представляет начало кадра сообщения CAN, далее идет 11-битный идентификатор, который устанавливает приоритет сообщения CAN. Чем меньше идентификатор, тем выше приоритет сообщения. Поле запроса на передачу (Remote transmission request) обычно является доминантным, но он становится рецессивным, когда один узел запрашивает данные у другого. Поле IDE (определяет длину идентификатора) является доминантным, когда отправляется стандартный кадр, и рецессивным, когда расширенный кадр. Бит r0 зарезервирован и не используется. DLC показывает, сколько байтов данных находится в сообщении. Таким образом, каждый получатель может проверить, получил ли он информацию в полном объеме. Далее идут сами дан-

ные 0-8 байт. Защитное поле (Cyclic redundancy check CRC) - это контрольная сумма всего кадра, предназначена для обнаружения ошибок передаваемых данных. Конец кадра (EOF) означает конец сообщения CAN и имеет ширину 7 бит для обнаружения ошибок. Последняя часть сообщения CAN – это межкадровое пространство (IFS), используемое в качестве временной задержки необходимой контроллеру CAN, чтобы переместить полученное сообщение в буфер для дальнейшей обработки.

Расширенный CAN использует 29-битный идентификатор вместе с несколькими дополнительными битами (рисунок 2). Расширенное сообщение имеет заменяющий бит (SRR) после 11-битного идентификатора, который действует как заполнитель для сохранения той же структуры, что и стандартный CAN. Поле IDE должно быть рецессивным, указывая, что за ним следует расширенный идентификатор. Далее идет 18-битный идентификатор за которым бит RTR, а за ним следует второй резервный бит r1. Остальная часть сообщения остается прежней.

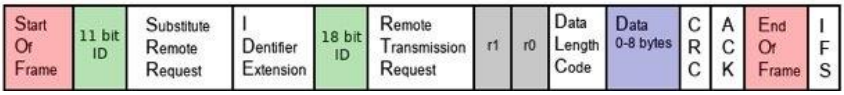

Рисунок 2 – Расширенный кадр данных

CAN-регистратор. Сборка CAN-регистратора началась с выбора способа и устройства для получения данных из CAN шины. После анализа устройств различных производителей и моделей, была выбрана схема (рисунок 3) на основе модуля MCP2515 и микроконтроллера Arduino Nano из-за простоты их использования, доступного для бесплатной загрузки программного обеспечения, а также возможности программирования микроконтроллера под различные задачи [2].

Платформа Arduino. Arduino представляет собой готовую аппаратно-программную платформу программная часть, которой состоит из бесплатной программной оболочки Arduino IDE для написания программ, их компиляции и программирования аппаратуры. В этой оболочке имеется текстовый редактор, менеджер проектов, препроцессор, компилятор и инструменты для загрузки программы в микроконтроллер. Оболочка написана на Java на основе проекта Processing, работает под Windows, Mac OS X и Linux.

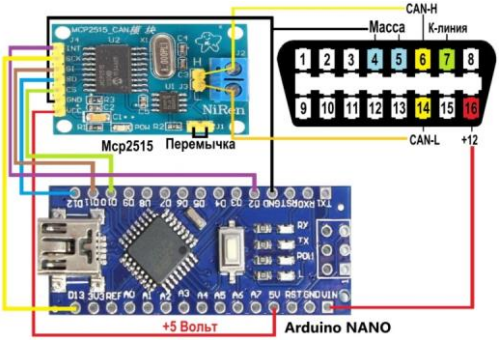

Рисунок 3 – Схема подключения микросхемы MCP2515 и микроконтроллера Arduino Nano к OBD разъему

Язык программирования Arduino называется Arduino C и представляет собой язык C++ с фреймворком Wiring.

Аппаратная часть представляет собой набор смонтированных печатных плат, продающихся как официальным производителем, так и сторонними производителями [3], [4].

К основным особенностям платформы Arduino можно отнести следующее:

– платформа имеет полностью открытую архитектуру;

– программирование производится через USB разъем, без использования программаторов и низкоуровневого программирования;

– низкая стоимость;

– стандартные размеры плат и типов разъемов позволяют собирать различные устройства без пайки используя платы расширения.

Дополнительные платы, ставятся подобно слоям бутерброда поверх платы Arduino.

Микросхема MCP 2515. Модуль MCP2515 (рисунок 4) представляет собой специализированный контроллер сети CAN, реализующий спецификацию CAN версии 2.0B. Модуль MCP2515 может передавать и принимать сообщения как базового, так и расширенного сообщения. У MCP2515 имеется 2 маски разрешения (acceptance mask) и 6 фильтров (acceptance filter), которые используются для отбрасывания нежелательных сообщений, что уменьшает нагрузку на управляющий микроконтроллер. MCP2515 обеспечивает реализацию протокола CAN по SPI шине [5].

Рисунок 4 – Микросхема MCP 2515

MCP2515 разработана таким образом, чтобы максимально упростить приложения, требующие подключения к CAN-шине. Устройство состоит из 3 основных блоков:

1) модуль CAN, который включает в себя систему обработки протокола CAN, маски, фильтры, буферы передачи и приема;

2) логика управления и регистры, которые используются для конфигурирования устройства и работы с ним;

3) блок поддержки протокола SPI.

Краткие характеристики микросхемы:

– соответствует стандарту CAN V2.0B;

– стандартные и расширенные ID;

– шесть 29 битных фильтра и две маски для приема сообщений;

– высокоскоростной SPI, до 10 МГц;

– напряжение питания от 2,7В до 5,5В.

Расшифровка сообщений. В автомобиле может применяться CAN шина разных типов, а порой она может быть и не единственной так как, одни системы более требовательны к скорости передачи данных (блок управления коробкой передач, блок управления двигателем и др.) а другие менее требовательны (блок системы климат контроля, блоками управления в дверях автомобиля и др.). В связи с этим скорость передачи данных в выбранной CAN шине может быть разной, но не произвольной. Чаще всего используются скорости 100 кбит/с, 125 кбит/с, 200 кбит/с, 250 кбит/с, 500 кбит/с, 1000 кбит/с.

Как правило, идентификаторы сообщений и сами данные для удобства принято представлять в шестнадцатеричном виде, а не двоичном. Такое представление облегчает анализ сообщений за счет меньшего количества символов. Так, например, сообщение в байте D2 которого содержится уровень топлива в баке выглядит следующим образом (рисунок 5).

ID	DLC	D1	D2	D3	D4	D5	D6	D7	D8
98FEFC65	8	FF	4E	FF	FF	FF	FF	FF	FF

Рисунок 5 – Сообщение
ID – идентификатор; DLC – количество байт в сообщении;
D1…D8 – байты сообщений.

Arduino поддерживает четыре классических системы исчисления: двоичную, восьмеричную, десятичную и шестнадцатеричную. 16-ричная система имеет 16 значений на один разряд, первые 10 как у десятичной, остальные – первые буквы латинского алфавита: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f.

При записи двоичная (Binary) имеет префикс 0b или B, то есть двоичное число 101 запишется как 0b101 или B101. Восьмеричная (Octal) имеет префикс 0, например, 012. Шестнадцатеричная (hexadecimal) имеет префикс 0x, FF19 запишется как 0xFF19.

Основная идея 16-ричной системы в том, что она позволяет записывать длинные десятеричные числа короче, например, один байт (255) запишется как 0xFF, два байта (65 535) как 0xFFFF, а три байта (16 777 215) как 0xFFFFFF [6].

Стоит отметить что иногда данные записываются в 2 байта и более, это обусловлено тем что один байт состоит из 8 бит каждый из которых может принимать значения лог. 0 или лог. 1, что дает 2^8 =256 комбинаций, что в свою очередь позволяет запомнить от 0 до 255 целых чисел или же меньший/больший диапазон чисел, используя переводные коэффициенты тем самым увеличивая/уменьшая шаг между рядом стоящими запоминаемыми числами. Так, например, зная, что какой-либо параметр принимает значения от 0 до 100, можно использовать один байт и переводной коэффициент 0,5, что даст точность в 0,5 ед. Когда же параметр принимает отрицательные, значения (температура, ускорение и др.) необходимо использовать

смещение. В связи с вышеизложенным, перевод данных из шестнадцатеричной формы в форму понятную для понимания требует корректировки с помощью переводных коэффициентов и смещения. Для каждого транспортного средства эти коэффициенты свои.

Так же при расшифровке сообщения, в котором параметр представлен в виде двух и более байтов, число записывается в позиционной системе счисления по основанию 256:

$$M = \sum_{n}^{n-1} A_i \cdot 256^i = A_0 \cdot 256^0 + A_1 \cdot 256^1 + \ldots + A_{n-1} \cdot 256^{n-1},$$

где A_i – набор целых чисел, каждое из которых лежит в интервале от 0 до 255, являющийся последовательностью байтов, составляющих M. При этом A_0 называется младшим байтом, а A_{n-1} – старшим байтом числа M.

Пример 1. Сообщение (рисунок 6) содержит в байте D2 информацию об отношении объема топлива в баке к объему бака, переводной коэффициент, согласно [7], равен 0,4%, смещение равно 0.

ID	DLC	D1	D2	D3	D4	D5	D6	D7	D8
98FEFC65	8	FF	4E	FF	FF	FF	FF	FF	FF

D2	
4E(HEX)	$L = 78 \cdot 0,4 = 31,2\ \%$
78(DEC)	

Рисунок 6 – Уровень топлива

Пример 2. сообщение, содержащее в байтах D7 и D8 (рисунок 7) скорость автомобиля будет расшифровываться следующим образом.

ID	DLC	D1	D2	D3	D4	D5	D6	D7	D8
98FE6CEE	8	FF	FF	FF	FF	FF	FF	76	E

Рисунок 7 Скорость автомобиля

D7 является младшим байтом, а D8 старшим байтом.

Тогда с учетом переводного коэффициента равного $\frac{1}{256}$, и смещения равного 0, согласно [7], скорость будет определяться следующим образом:

D7	D8	$Re = \frac{v \cdot d}{\nu}$
76(HEX)	E(HEX)	
118(DEC)	14(DEC)	

Чтение сообщений. В используемом CAN-регистраторе для чтения сообщений из CAN шины применялось две программы. Первая программа, написанная и дополненная множеством энтузиастов [8], она выводит все сообщения, идущие по шине (рисунок 8).

Рисунок 8 – Поток сообщений телематической шины S6

Вторая программа выводит лишь наиболее полезные сообщения: в которых находятся данные о скорости автомобиля, расходе топлива и температуре охлаждающей жидкости, а также через определенный промежуток времени выводит расчет средних значений скорости и расхода.

Рисунок 9 – Стенд «Транспортная телематика»

Программа для чтения всего потока сообщений выглядит следующим образом (рисунок 10).

```
#include <SPI.h> // подключение библиотеки шины SPI
#include <mcp2515.h>// библиотека для работы с CAN модулем MCP2515
struct can_frame canMsg;
MCP2515 mcp2515(10); // установка пина Chip Select
void setup() {
  Serial.begin(115200);//настройка скорости последовательного порта
  SPI.begin();
  mcp2515.reset(); // перезагрузка CAN модулем MCP2515
  mcp2515.setBitrate (CAN_500KBPS,  MCP_8MHZ); //скорость передачи данных
  mcp2515.setNormalMode(); // выбор режима работы CAN модулем MCP2515
  Serial.println("------- CAN Read ----------");//шапка
  Serial.println("ID    DLC     DATA ");//шапка
}
void loop() {
    if (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK) {
    Serial.print(canMsg.can_id, HEX); // печать ID
    Serial.print("  ");
    Serial.print(canMsg.can_dlc, HEX); // печать DLC
    Serial.print(" ");
        for (int i = 0; i<canMsg.can_dlc; i++) {
        Serial.print(canMsg.data[i],HEX); //печать данных в HEX формате
        Serial.print(" ");
        }
    Serial.println();
   }
}
```

Рисунок 10 – Программа для чтения всего потока сообщений

Подключался CAN-регистратор к телематической шине S6 стенда «Транспортная телематика» (рисунок 9) предназначенному для

функциональной имитации работы автомобиля и телематической системы.

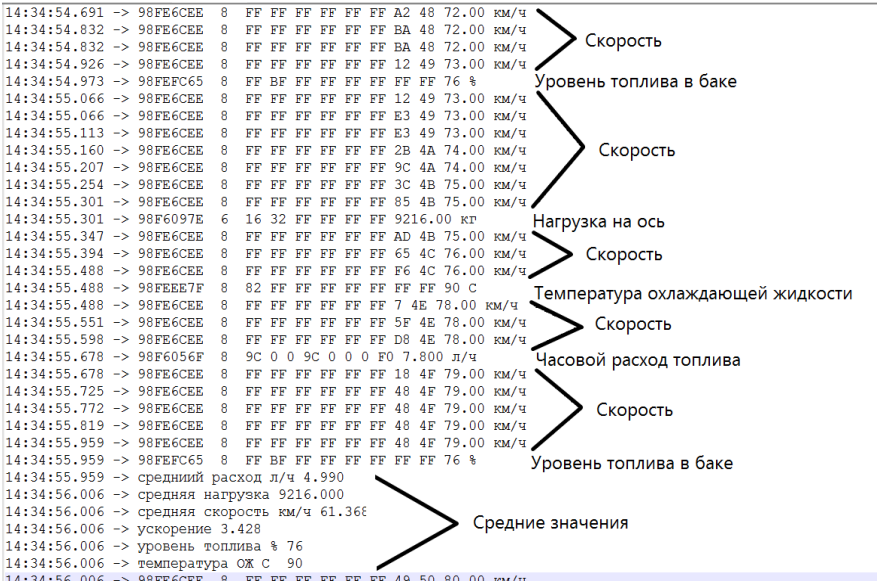

Рисунок 11 – Результаты работы второй программы

В данной работе были извлечены и расшифрованы значения ключевых параметров, связанных с работой транспортного средства. Проверка результатов проводилась с помощью приборной панели и тахографа установленного на стенде, а также профессионального прибора CAN Master.

Результаты проделанной работы могут найти применение при выполнении работ по диагностике и ремонту автомобилей, при обучении студентов, или при разработке систем и устройств на базе CAN шины.

Дальнейшая работа предполагает написание программы для ПК позволяющая записывать полученные данные в файл, для получения возможности строить графики изменения параметров по полученным данным, а также представлять поток сообщений в режиме «обновления» (отображение только изменений в полученных сообщениях).

Литература

1. WIKI2 [Электронный ресурс]. – Электронные данные. – Режим доступа: //wiki2.org/ru/Controller_Area_Network.

2. DRIVE2 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.drive2.ru/l/469794106709639308/.

3. ARDUINO.UA [Электронный ресурс]. – Электронные данные. – Режим доступа: https://https://arduino.ua/art2-istoriya-sozdaniya-arduino.

4. MICPIC [Электронный ресурс]. – Электронные данные. – Режим доступа: http://micpic.ru/home/proekty-na-arduino/186-chto-takoe-arduino.html.

5. MICROSIN.NET [Электронный ресурс]. – Электронные данные. – Режим доступа: http://microsin.net/adminstuff/hardware/mcp2515-stand-alone-can-controller-with-spi-interface.html.

6. Alex Gyver Technologies[Электронный ресурс]. – Электронные данные. – Режим доступа: https://alexgyver.ru/lessons/variables-types/.

7. PDF4PRO [Электронный ресурс]. – Электронные данные. – Режим доступа: https://pdf4pro.com/amp/view/fms-standard-description-version-03-14-09-6c7bc.html.

8. GITHUB [Электронный ресурс]. – Электронные данные. – Режим доступа: https://github.com/autowp/arduino-mcp2515.