Университет ИТМО

Курсовая работа

по дисциплине: «Информационно-управляющие системы»

«Контроллер датчика уровня топлива»

Выполнили:
студенты IV курса
группы Р3415
Припадчев Артём
Кунцова Анастасия

Проверил:
Ключев А.О.

Санкт-Петербург
2015

Оглавление

Этап 1

Контроллер уровня топлива в баке автомобиля

Функциональность: имеется два режима вывода информации об уровне топлива:

- Текущий уровень в литрах
- Прогноз расстояния в км или милях

Значение уровня топлива генерируется эмулятором. Информации о текущем уровне выводится на ЖКИ в двух режимах:

- Графическом ([:::::])
- Числовом (123)

При опускании уровня бензина ниже критического срабатывает индикатор (светодиод) и звуковое оповещение.

Ограничения на применение: рабочее напряжение учебного стенда SDK-1.1-9B

Перспективные возможности: изменение мер счисления (км на мили и т.п.). Установление нижнего порога уровня топлива для различных категорий автомобиля.

ПО и инструментальные средства

Компиляторы sdcc

Используются для компиляции исходных текстов программ. Для компиляции программ для учебного стенда SDK-1.1 используется компилятор sdcc.

SDCC (Small Device C Compiler) – бесплатный с открытым исходным кодом, переносимый, оптимизирующий ANSI C компилятор для 8-разрядных микроконтроллеров на базе Intel MCS51, Maxim 80DS390, Zilog Z80, Motorola 68HC08. Распространяется под лицензией GNU GPL.

В результате компиляции на выходе получаются два файла формата .hex, готовые для записи в память контроллера.

Формат Intel HEX

Intel HEX – формат файла. Основным отличием этого формата от таких монстров, как ELF и COFF является крайняя простота. Формат позволяет хранить только образ памяти. Ни о каком перемещаемом коде и возможности хранения объектных файлов в этом формате речи не идет.

В настоящий момент этот формат в основном используется при программировании встроенных систем. Большинство компиляторов и линкеров умеют выдавать загрузочный модуль в этом формате. Строки файла представляют собой текстовые записи, в которых закодированы адреса расположения, команды и данные в шестнадцатеричной системе счисления. Изначально, HEX формат использовался для работы с перфоленточными загрузчиками. В настоящее время HEX формат используют для программирования различных контроллеров и связи с программаторами ППЗУ.

Язык программирования C51

C - это язык программирования общего назначения, предназначенный для написания программ, эффективных по исполняемому коду, с элементами структурного программирования и богатым набором операторов. Язык программирования C практически не имеет ограничений, что позволяет использовать его для эффективного решения широкого круга задач. Однако при написании программ для микроконтроллеров, принадлежащих к семейству MCS-51, необходимо учитывать особенности построения аппаратуры этих микросхем, поэтому был создан диалект этого языка.

В состав языка программирования C-51 введён ряд изменений, отображающих особенности построения памяти микроконтроллеров семейства MCS-51. Кроме того, эти изменения позволяют непосредственно обращаться к

встроенным портам, таймерам и другим устройствам микроконтроллеров указанного семейства. Особенности микроконтроллеров этого семейства в основном отображаются через описания переменных.

Язык программирования C-51 удовлетворяет стандарту ANSI-C и предназначен для получения компактных и быстродействующих программ, предназначенных для микроконтроллеров семейства MCS-51. Язык C-51 обеспечивает гибкость программирования на широко известном языке C, при скорости работы и компактности, сравнимой с программами, написанными на языке программирования ассемблер.

Утилиты m3p и make

M3P – кроссплатформенная инструментальная система со встроенным интерпретатором языка FORTH. Система M3P предназначена для решения следующего ряда задач:

- Отладки, тестирования и внутрисистемного программирования встроенных систем;
- Интеграции инструментальных средств в единую систему;
- Связывания разнородных инструментальных средств посредством языка сценариев.

Утилита make используется для автоматизации процесса сборки исходных текстов программ в .hex файл по созданному makefile. Далее скомпилированная программа записывается в память учебного стенда SDK-1.1 с помощью утилиты m3p и скрипта на языке Forth. С ее помощью открывается соединение со стендом SDK-1.1 по последовательному каналу.

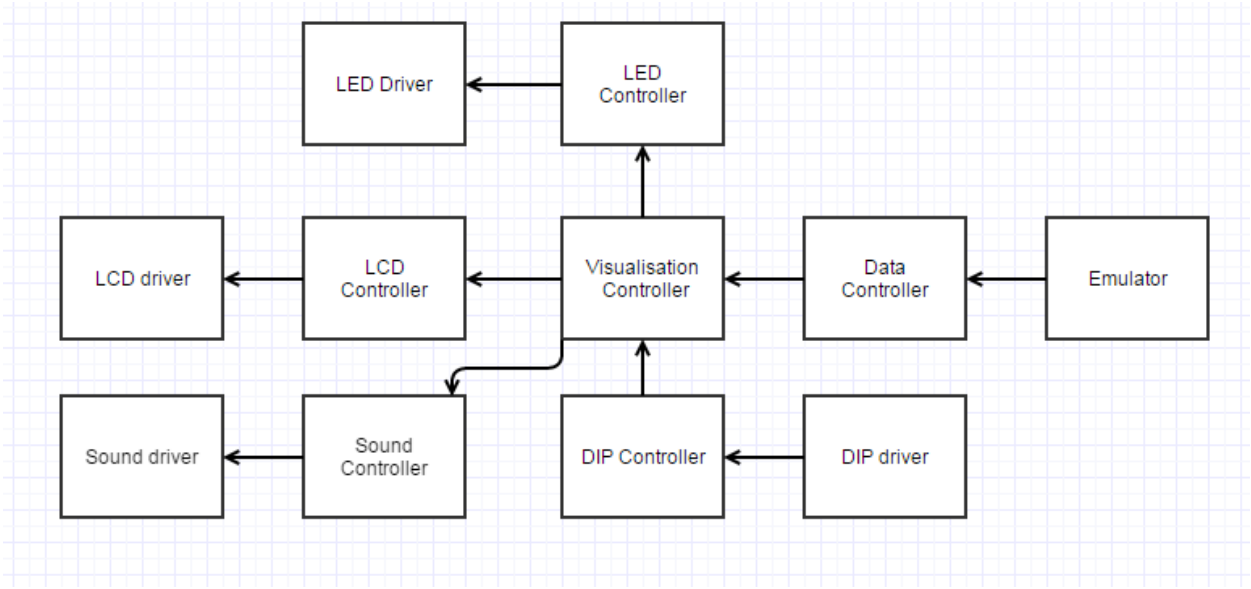
Этап 2

Архитектура устройства

Программная архитектура ИУС

Система состоит из двух основных компонентов: датчик уровня бензина и учебного стенда SDK1.1 для визуализирования показаний счетчика. Датчик уровня бензина имитируется дополнительным программным компонентом для передачи значений непосредственно с персонального компьютера. По последовательному каналу он подключается к учебному стенду SDK1.1. Ниже приведены структуры программного обеспечения на этих модулей:

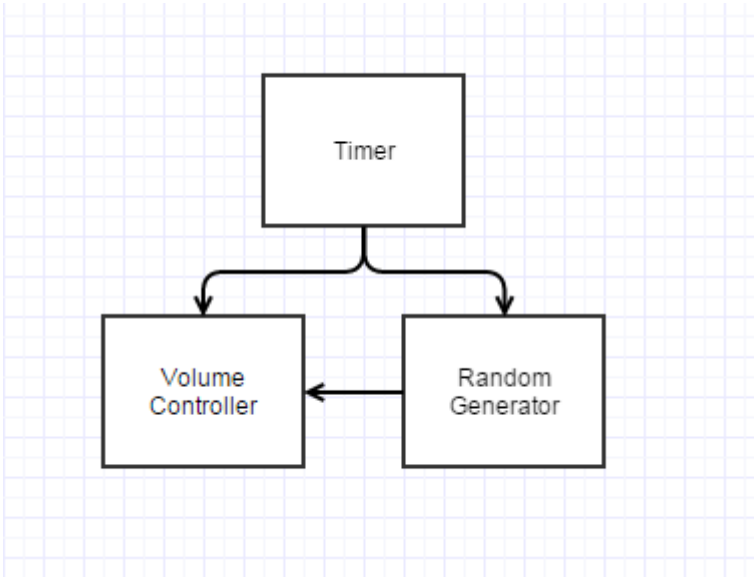
1. **Учебный стенд SDK – 1.1**



Задача учебного стенда SDK – 1.1 – визуализировать данные, приходящие от датчика уровня топлива. Data Controller принимает, проверяет их корректность и направляет данные на визуализацию. Visualisation Controller в зависимости от установленного режима вывода информации (за предоставление данных о режиме вывода информации отвечает DIP Controller) формирует данные для вывода на устройства визуализации (LED и LCD). LED Controller формирует данные для вывода на на LED. LCD формирует набор данных для вывода на LCD.

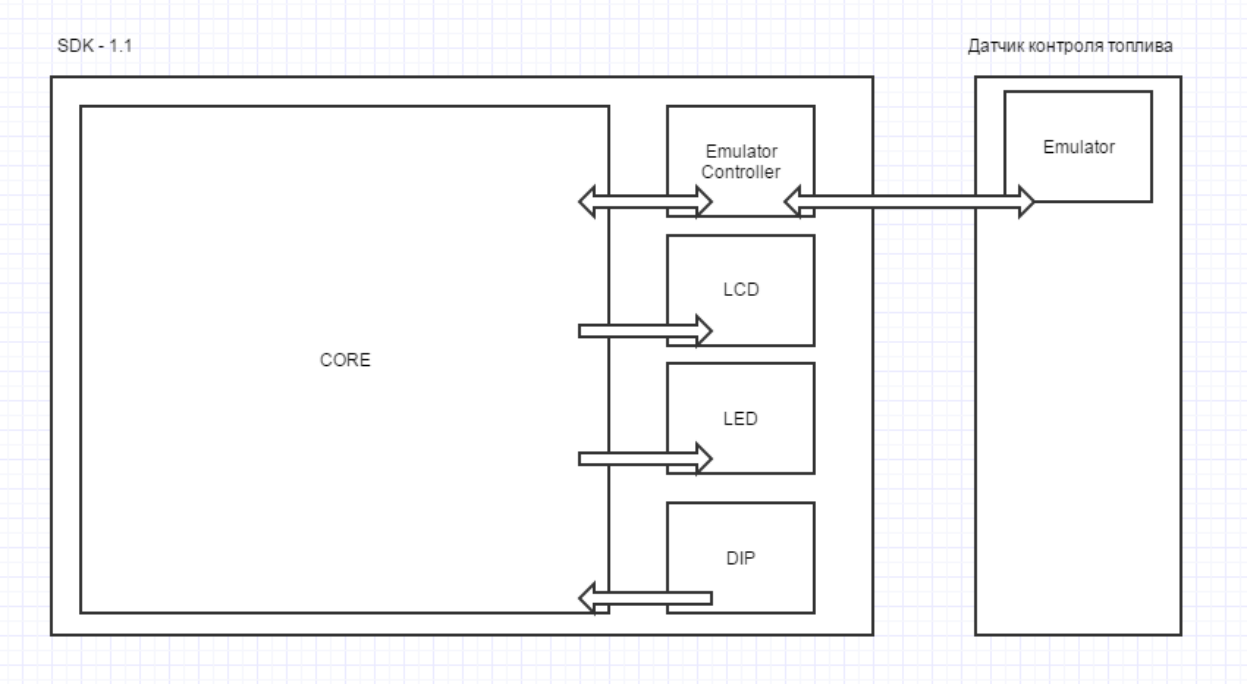
Датчик уровня бензина (эмулятор)

Посылает данные о текущем уровне топлива. Эмуляция осуществляется за счет таймера. С каждым срабатыванием таймера уровень бензина опускается на случайную величину. При опускании уровня ниже критического, происходит “дозаправка” на случайную величину.



Аппаратная архитектура

Основным оборудованием является стенд SDK – 1.1. Так как датчик уровня топлива имитируется программным средством, то его аппаратная архитектура не приведена. На приведенном ниже рисунке можно увидеть, как взаимодействуют между собой различные части SDK – 1.1 и как SDK – 1.1 взаимодействует с датчиком контроля топлива.



На SDK – 1.1 используется контроллер эмулятора для обмена данными с датчиком контроля топлива, LED и LCD используются для визуализации полученных данных. С помощью DIP переключателей ведется управление режимами вывода информации на LCD.

Особенности контроллера уровня топлива с точки зрения пользователя

Имеются несколько режимов вывода информации о текущем уровне топлива: графический и текстовый. Пользователь выбирает более подходящий ему путем переключения DIP. При снижении уровня ниже критической отметки пользователь моментально оповещается путем зажигания LED. Контроллер не очень эргономичен, в следствии того, что SDK – 1.1 является монолитной конструкцией и элементы на нем не передвигаются.

ЭТАП 3

Особенности реализации

Эмулятор:

Эмулятор реализован на основе таймера. Изменения срабатываю по таймеру каждый N времени. N задается программно (так как это только эмулятор для тестирования). Бензин расходуется не равномерно, чтобы приблизить эмулятор к реальному датчику.

```
void emulator()
{
	if (time == 254)
	{
		del++;
		if (del == 10)
		{
			del = 0;
			if (emulated_tank_volume > 1)
			emulated_tank_volume = emulated_tank_volume - next_rand() % dif;
		}
	}
	if (emulated_tank_volume < threshold) emulated_tank_volume = next_rand() % medium_volume +
low_add_volume;
}
```

Обмен данными с визуализатором осуществляется через интерфейс:

```
unsigned char get_tank_value();
```

Генератор случайных чисел реализован на основе таймера:

```
unsigned char pre_rand = 17;
unsigned char next_rand(void)
{
    unsigned char timerValue = get_current_millis();
    unsigned char newRandom = (timerValue * pre_rand + timerValue) % 256;
    pre_rand = newRandom;
    return newRandom;
}
```

Визуализатор:

Визуализатор работает по принципу наименьших действий (меняет показания только при изменении значения уровня топлива в баке или при изменении режима визуализации).

Контроллер данных:

```
unsigned char data_controller();
```

Считывает новые данные с датчика уровня топлива и передает их в визуализатор:

```
value = data_controller();
visualise(value);
```

Контроллер визуализации:

Получает на входе значение уровня топлива в баке. Считывает данные с DIP, и на основе выбранных DIP формирует данные для вывода на LCD. Так же, если уровень топлива опускается ниже порога, выводит информацию на LED.

```
void visualise(unsigned char value)
{
dip = read_max(2);
if (value == old_value && dip == old_dip) return;

metrica = (dip & 0x1) ? MILES : KILOMETERS;
mode = (dip & 0x2) ? TANK_VALUE : PREDICTION;
consumption = (metrica == MILES) ? consumption100miles : consumption100km;
sub_info = (mode == PREDICTION) ? value * consumption : value;
lcd_controller(value, sub_info);

if (value < fuel_threshold)
{
make_sound();
write_led(0xff);
}
else write_led(0x00);
old_value = value;
old_dip = dip;
}
```

Этим выражением мы проверяем актуальность данных отображенных на экране и решаем, требуется ли перерисовка.

```
if (value == old_value && dip == old_dip) return;
```

Контроллер LCD:

Преобразует данные, полученные от визуализатора в представление, которое может отобразить LCD:

```
void lcd_controller(unsigned char volume, unsigned char additional_info)
{
format_postfix();
format_row1(volume);
format_row2(additional_info);
clear_lcd();
for (i = 0; i < 16; i++)
{
    print_char_lcd(row1[i]);
}
goto_xy_lcd(1,2);
for (i = 0; i < 16; i++)
{
    print_char_lcd(row2[i]);
}
}
```

Контроллер звука:

При опускании уровня топлива ниже заданного порога срабатывает звуковой сигнал.

```
void make_sound()
{
unsigned int sound = 0;
int i;
for (i = 0; i < 100; i++) {
    sound = (sound == 0x00) ? 0x10 : 0x00;
    write_max(4, sound);
}
}
```

Контроллер LED и DIP:

Устроены по принципы записи/чтения значений в требуемый регистр через расширитель портов ввода-вывода.

```
void write_led(unsigned char value);
unsigned char read_dip();
```

Драйверы устройств:

Для работы с LED, LCD, звуком, DIP и расширителем портов ввода-вывода были разработаны драйверы этих устройств.

Этап 4

Тестирование

Тестирование было разделено на 3 этапа

Этап 1. Разработка эмулятора датчика уровня топлива

Для тестирования визуализатора датчика уровня топлива в баке был разработан эмулятор, генерирующий показания датчика. Эмулятор можно сконфигурировать для генерации значений с различным разбросом.

```
void emulator()
{
	if (time == 254)
	{
		del++;
		if (del == 10)
		{
			del = 0;
			if (emulated_tank_volume > 1)
			emulated_tank_volume = emulated_tank_volume - next_rand() % dif;
		}
	}
	if (emulated_tank_volume < threshold) emulated_tank_volume = next_rand() % medium_volume + low_add_volume;
}
```

Разброс можно регулировать переменными:

- dif – для увеличения/уменьшения величины случайного расхода топлива
- medium_volume – для увеличения/уменьшения величины пополняемого топлива

Также изменяя условия цикла можно изменять скорость расхода/пополнения топлива.

Этап 2. Тестирование пользовательского интерфейса

Были протестированы все возможные варианты режимов вывода данных:

1. Мера длины – километры, режим вывода – текущее значение уровня топлива.
2. Мера длины – километры, режим вывода – прогнозируемое расстояние на остаток топлива.
3. Мера длины – миля, режим вывода – текущее значение уровня топлива.
4. Мера длины – миля, режим вывода – прогнозируемое расстояние на остаток топлива.
5. Звуковое и визуальное оповещение при снижении уровня топлива ниже порогового значения.

На начальном этапе вывод информации производился в одну строку. То есть на экран выводилась либо графическая информация, либо значение в цифрах. В ходе тестирования правильным был посчитан подход, когда отображение сразу ведется в двух режимах: графическом и цифровом. Причем мера цифрового значения может регулироваться dip-переключателями.

Отображение информации реализовано на двух строках. Для перехода на вторую строку используется метод goto_xy_lcd(1,2).

Этап 3. Тестирование на гибкость.

Получение текущего уровня топлива реализовано интерфейсом get_tank_value(), что позволяет заменять блок получения данных без вмешательства в код основной программы.

Устанавливать параметры для различных категорий автомобиля можно с помощью переменных:

- tank_max_volume – максимальный объем бака в литрах

- fuel_threshold – значение уровня топлива при котором подается звуковой и визуальный сигнал
- consumption100km – расход в литрах на 100 километров
- consumption100miles – расход в милях на 100 километров

Тестирование производилось при различных значениях параметров.

Для того, чтобы всесторонне оценить работу контроллера уровня топлива использовался простейший генератор псевдослучайных чисел:

```
unsigned char pre_rand = 17;
unsigned char next_rand(void)
{
	unsigned char timerValue = get_current_millis();
	unsigned char newRandom = (timerValue * pre_rand + timerValue) % 256;
	pre_rand = newRandom;
	return newRandom;
}
```

Литература

1. Ключев, А.О., Ковязина Д.Р., Кустарев, П.В., Платунов, А.Е. Аппаратные и программные средства встраиваемых систем. Учебное пособие – СПб.: СПбГУ ИТМО, 2010. – 290 с.
2. Ключев А.О., Ковязина Д.Р., Петров Е.В., Платунов А.Е. Интерфейсы периферийных устройств. – СПб.: СПбГУ ИТМО, 2010. – 290 с.