Таблица 1. Представление кода идентификации в прямом и обратном виде

1	2	3	4	5	6	7	8
D2	01	2F	62	90	04	0B	1F
1F	0B	04	90	62	2F	01	D2

идентификации протокола 1-Wire. Например, файл **hex_12f629_2409_v1_D2012F6290040B1F.hex** предназначен для схемы ветвителя с номером **D2012F6290040B1F**. Для работы программы термометрии этот номер записывается в файле инициализации к соответствующим датчикам DS1820, чтобы обращаться именно к данному ветвителю с установленными для него датчиками.

Полученные файлы нужно записать в МК любым программатором для данного типа МК. Если не требуется модернизация программы, например, изменение кодов для обращения к ветвителям, то сформированный каталог можно удалить. Если изменили коды обращения, то оставляете сформированный файл **silos_v3.exe** и удаляете все предыдущие файлы. После компиляции сформированный файл **silos_v3.exe** содержит все новые изменения для работы с новыми номерами идентификации ветвителя. В новом файле изменится контрольная сумма и дата создания, размер файла останется прежним.

При более «серьёзной» модификации программы изменится и размер исполняемого файла. Рассмотрим несколько поясняющих примеров функционирования программного обеспечения для схемы ветвителя на 12F629 на языке программирования FORTH. Текст программы находится в файле **silos_v3_2409_v1.f** и компилируется через выполнение общей программы **silos_v3.f** с помощью файла **100_spf4.exe.** Для уточнения, где описание и где программа, фрагменты текста программ, «вырезанные» из основного текста, размещены в цветных квадратах.

Пример 1. Изменение идентификационного кода

В стандарте интерфейса 1-Wire доступ к «ведомым» элементам осуществляется по их идентификационному коду, который состоит из 8 байт [1]. Первый байт определяет функцию или тип устройства, далее 6 байт каких-либо номеров, а последний байт – контрольная сумма первых 7 байт.

Таким образом, формируется стандартизация интерфейса 1-Wire и его устройств. Например, первый байт 0x28 определяет датчики температур DS18B20, первый байт 0x1F определяет ветвители DS2409 и так далее.

Требуется выполнять стандартизацию обращения к устройствам интерфейса 1-Wire, поскольку в этом случае программное обеспечение производителя будет работать нормально со схемой ветвителя на МК.

Следовательно, можно изменять со 2-го по 7-й байт и оставлять 1-й байт с кодом 0x1F. Последний байт формируется по первым 7 байтам.

Для формирования последовательности номеров задействованы 2-й и 3-й байт. Остальные 4, 5, 6, 7-й байты не меняются, и допустимо записывать любую информацию.

Рассмотрим распределение байт идентификационного номера:

D2012F6290040B1F – код идентификации, где:

- 1F – тип устройства 1-Wire, ветвитель DS2409, и он заменён на 12F629;
- 040B – номер, формируемый программой в количестве 10 штук, начиная с 0403 до 040C;
- 012F6290 – постоянные байты, где записан тип используемого микроконтроллера;
- D2 – контрольная сумма первых 7 байт.

Отсчёт нумерации байтов может выполняться в разных направлениях и зависит от представления в программе. На нижнем уровне передача начинается с байта кода информации об устройстве, а в программном обеспечении верхнего уровня отображается в обратном виде. Представление кодов идентификации с разным видом дано в табл. 1.

- В файле программы silos_v3_2409_v1.f текстовым редактором находим строки, определяющие начальный номер идентификации:

```
CREATE ZAPROS_ACP_OUT_RGN1 0x3 C,
\ мл. байт
CREATE ZAPROS_ACP_OUT_RGN2 0x4 C,
\ ст. байт и меняем, например, старший
байт на 0x1
```

Таким образом, будет формироваться последовательность 0103, 0104, ... 010C, всего 10 кодов.

Далее находим строку, определяющую количество файлов с идентификационными кодами:

```
CREATE KOLVO_PIC_UI 0xA, \ и изменяем
```

на другое требуемое количество файлов, отличное от 10. Следует обращать внимание на различные способы записи чисел с разными форматами счисления. **0xA** – это число 10 в шестнадцатеричной системе счисления. Запись и отображение значения чисел контролируется через переменную **BASE**.

- В файле программы находим форт-слово `NAME_PIC_HEX_RG1` и изменяем значение постоянных байтов.

Например, после `ks` на месте 12F629 пишем простой ряд чисел от 0...7:

`:NAME_PIC_HEX_RG1 S» hex_12f629_2409_v1_ks012F629000001F.hex»;` – был текст форт-слова,

`: NAME_PIC_HEX_RG1 S» hex_12f629_2409_v1_ks0123456700001F.hex»;` – стал текст форт-слова,

Необходимо отметить, что имя формируемого файла содержит уточнение функции работы. Это все байты до ks. При необходимости данные по названию тоже можно поменять. Например:

`: NAME_PIC_HEX_RG1 S» hex_12f629_2409_v1_ks012F629000001F.hex»;` – был текст форт-слова.

`: NAME_PIC_HEX_RG1 S» Novosibirsk_SibGUTIS_ks012F629000001F.hex»;` – стал текст форт-слова.

Чтобы не нарушать структуру имени файловой системы, размер текста не должен превышать 255 байт.

- Выполняем компиляцию программы через командный файл start_new_versii.cmd и получаем файлы для программирования МК. В процессе выполнения программа посчитает контрольную сумму ks и подставит её в нужное место. В байтах 2-м и 3-м формируются номера с увеличением на +1 для младшего байта. Все данные записываются в HEX-формате в шестнадцатеричной системе счисления.

Для понимания отличия значения числа и его отображения в различных системах счисления (сс) приведём пример отображения натуральных чисел в интервале 0...16 в сс от 16 до 2 на языке FORTH.

```
\ начало текста программы
: PKZ_DEMO BASE @ DUP
 DECIMAL CR
  S" Ncc=" TYPE . CR    BASE !
  2DUP DO I . LOOP ;
: FORMAT_CH_DEMO  HEX 0x10 0
  2 0x10 DO I BASE !  PKZ_DEMO
  -1 +LOOP  HEX 2DROP ;
 FORMAT_CH_DEMO
\ окончание текста программы.
```

Для работы программы нужно создать файл, например, **test.f** блокнотом, скопировать текст от комментария \ **начало** ... до комментария \ **окончание** ... и сохранить. Косая линия вле-