Синтез мелодий в микроконтроллерах фирмы SiLabs

Олег Николайчук
onic@ch.moldpac.md

Схемотехника

Целью настоящей статьи является ознакомление читателей с одним из путей формирования звуковых сигналов в микроконтроллерах фирмы Silicon Laboratories (SiLabs).

Любое устройство, выполненное на базе микроконтроллера фирмы SiLabs (Silicon Laboratories) [1],
Можно легко оснастить синтезатором мелодий, значительно расширяющим функциональные возможности прибора. Ведь возможности одно-тонального звукового излучателя – буззера, которыми очень часто оснащаются устройства на микроконтроллерах, достаточно ограничены только ритмическим рисунком звуковых сигналов. Использование же звукового синтезатора мелодий повышает узнаваемость сигналов и, безусловно, значительно увеличивает их разнообразие. Для сравнения вспомним скромные возможности встроенных динамиков персональных компьютеров, и сравним их с возможностями даже простейших звуковых карт.

Аппаратная реализация узла синтеза мелодий является весьма простой. Для этого узла требуется наличие всего одной свободной линии порта ввода/вывода. Возможны как минимум два варианта реализации этого узла: с прямым включением излучателя звука и с дополнительным внешним усилителем.

В первом случае свободный вывод линии порта ввода/вывода должен быть настроен на вывод в режиме с открытым истоком (Open Drain Mode), и к нему подключается один из выводов малогабаритного излучателя звука, например HC0905F (5 В) или HC0903F (3 В) фирмы JL-World Corp.[2]. Второй вывод излучателя звука подключается непосредственно к линии питания микроконтроллера. Следует отметить, что сопротивление этих излучателей звука довольно низкое – 40 и 25 Ом соответственно, поэтому последовательно с излучателем иногда включается резистор с номиналом в 30-56 Ом. Однако возможно и непосредственное (т.е. без дополнительного резистора) включение излучателя звука, т.к. линии портов ввода/вывода микроконтроллеров фирмы SiLabs имеют встроенные ограничители тока. Следует отметить, что громкость звука при таком включении очень небольшая, поэтому достаточно часто используется второй вариант включения – с внешним дополнительным усилителем.

Внешний дополнительный усилитель может быть выполнен как на транзисторе, так на специализированной микросхеме усилителя низкой частоты. При этом используемая линия порта ввода вывода должна быть настроена на работу в ключевом режиме (Push-Pull Mode).

Рассмотрим программную сторону реализации синтезатора мелодий. Суть описываемого варианта реализации синтезатора звуков заключается в том, что звуковая частота каждой ноты генерируется с помощью функций задержек, которые пересчитывают тактовую частоту микроконтроллера и переключают выбранный нами вывод линии ввода/вывода. Допустим, что мы назвали выбранную линию ввода/вывода P2.0 следующим образом:

```
sbit    MUZ   = P2 ^ 0;
```

Рассматриваемые ниже подпрограммы будут использовать функции задержек общего назначения, которые обычно используются в основной программе. Отметим также, что приведенные значения временных задержек рассчитаны на тактовую частоту генератора 22.1184 МГц.

```
//************************************
// Функция генерации короткой задержки
// Примерно 8+MKS микросекунд
//************************************
void    Time   (unsigned MKS)
{
```

```
        while (MKS--) {WDT();}
}
//************************************
// Функция генерации длинной задержки
// Примерно MS миллисекунд
//************************************
void    Delay  (unsigned MS)
{
        while(MS--) {Time(1000);}
}
//************************************
// Функция перезапуска охранного таймера
//************************************
void    WDT (void)
{
// последовательность команд, зависящая от
// семейства микроконтроллеров SiLabs
}
```

В нашем случае мы будем реализовать одноголосый синтезатор звуков, охватывающий диапазон две с лишним октавы музыкального ряда, - первую, вторую и две ноты третьей октавы. Известно, что частоты различных нот музыкального ряда имеют не целочисленные значения, что мы конечно реализовать не будем ввиду значительной сложности, поэтому некоторые ноты синтезируемого нами музыкального ряда будут несколько отличаться от стандартных, однако уловить на слух такое различие смогут лишь люди с абсолютным музыкальным слухом и соответствующим образованием.

Итак, как мы уже сказали, мы будем формировать ноты с округленными значениями частот:

Для первой октавы ноты будут иметь следующие частоты: До – 424 Гц, До # (диез) – 401 Гц, Ре – 377 Гц, Ре # - 356 Гц, Ми – 336 Гц, Фа – 317 Гц, Фа # - 299 Гц, Соль – 283 Гц, Соль # - 266 Гц, Ля – 251 Гц, Ля # - 237 Гц, Си – 224 Гц. Очевидно, что каждая нота имеет определенную длительность звучания, обозначенную как DL. Тогда, если функция, формирующая звук ноты, будет называться DNota, для первой октавы можно написать макроопределения нот следующим образом:

```
#define Do1(DL);      DNota(424,DL);
#define Do1D(DL);     DNota(401,DL);
#define Re1(DL);      DNota(377,DL);
#define Re1D(DL);     DNota(356,DL);
#define Me1(DL);      DNota(336,DL);
#define Fa1(DL);      DNota(317,DL);
#define Fa1D(DL);     DNota(299,DL);
#define Sol1(DL);     DNota(283,DL);
#define Sol1D(DL);    DNota(266,DL);
#define La1(DL);      DNota(251,DL);
#define La1D(DL);     DNota(237,DL);
#define Se1(DL);      DNota(224,DL);
```

Для второй октавы частоты отличаются примерно в два раза и макроопределения нот будут выглядеть следующим образом:

```
#define Do2(DL);      DNota(211,DL);
#define Do2D(DL);     DNota(199,DL);
#define Re2(DL);      DNota(188,DL);
#define Re2D(DL);     DNota(178,DL);
#define Me2(DL);      DNota(167,DL);
```

```
#define Fa2(DL);      DNota(158,DL);
#define Fa2D(DL);     DNota(149,DL);
#define Sol2(DL);     DNota(141,DL);
#define Sol2D(DL);    DNota(133,DL);
#define La2(DL);      DNota(126,DL);
#define La2D(DL);     DNota(119,DL);
#define Se2(DL);      DNota(113,DL);
```

Из третей октавы мы будем использовать только две ноты, которые наиболее часто встречаются в большинстве известных мелодий:

```
#define Do3(DL);      DNota(107,DL);
#define Re3(DL);      DNota(94,DL);
```

Теперь нам осталось рассмотреть что из себя представляет собственно подпрограмма, формирующая ноту:

```
//************************************
// Подпрограмма формирования звука ноты
// FR – частота ноты
// DL – длительность ноты
//************************************
void DNota (int FR, int DL)
{
        for(int i=0; i<DL; i++)
        {
                MUZ=0;        Time(FR);
                MUZ=1;        Time(FR);
        }
}
```

Длительности нот в музыкальной грамоте принято обозначать в условных единицах как «целые» - (мы будем обозначать их как 1), «половинки» - (2), «четвертинки» - (4), «восьмушки» - (8), «шестнадцатые» - (16) и «тридцать вторые» - (32). На самом деле «целые» ноты имеют наибольшую длительность, а остальные имеют длительность соответственно ½, ¼, 1/8, 1/16, и 1/32 от длительности целой. Обычно для мелодии среднего темпа длительность «тридцать второй» составляет примерно 500 мкс. Соответственно, легко посчитать длительность остальных нот.

Известно также, что при формировании мелодий используются паузы, или «пустые ноты», которые также имеют определенные длительности, как и ноты. Рассмотрим подпрограмму, которая формирует паузы:

```
//************************************
// Подпрограмма формирования пауз
// DL – длительность паузы (1,2,4,8,16,32)
//************************************
void Pause (int DL)
{
        switch(DL)
        {
                case 16: Delay(1);     break;
                case  8: Delay(2);     break;
```

```
                case  4: Delay(4);      break;
                case  2: Delay(8);      break;
                case  1: Delay(16);     break;
                case 32: Time(500);     break;
                default:                break;
        }
}
```

Ну вот практически и все, что необходимо для формирования мелодий. К сожалению, длительность нот нельзя указывать в условных единицах и приходится приводить в мкс.

Отметим также, что для облегчения читаемости программы мелодии также рекомендуется сперва записывать в виде макроопределений, а затем вызывать в основной программе. Для примера приведем примеры записи нескольких известных мелодий. Например так выглядит запись хроматической гаммы:

```
#define MUZ_Gamma1();
        EA=0;Do1(50);Do1D(50);Re1(50);Re1D(50);Me1(50);\
        Fa1(50);Fa1D(50);Sol1(50);Sol1D(50);La1(50);La1D(50);Se1(50);\
        Do2(50);Do2D(50);Re2(50);Re2D(50);Me2(50);\
        Fa2(50);Fa2D(50);Sol2(50);Sol2D(50);La2(50);La2D(50);Se2(50);\
        Do3(50);EA=1;
```

Отметим, что для того, чтобы обработка любых прерываний не искажала звучание мелодий, желательно на время звучания запрещать все прерывания.

А так выглядит запись начала известной мелодии Моцарта:

```
#define MUZ_Mozard1();
        EA=0;Sol1(150);Delay(32);Re1(50);Sol1(150);Pause(1);Re1(50);\
        Sol1(50);Re1(50);Sol1(50);Se1(50);Re2(200);EA=1;
```

И в заключении фрагмент известной мелодии «Тореадор»:

```
#define MUZ_Toreodor();
        EA=0;Sol2(200);Pause(8);La2(100);Pause(8);Sol2(100);Pause(8);\
        Me2(200);Pause(8);Me2(200);Pause(8); \
        Me2(100);Pause(8);Re2(100);Pause(8);Me2(100);Pause(8);Fa2(100);Pause(8);\
        Me2(400);Pause(2);EA=1;
```

Приведенных сведений вполне достаточно для того, чтобы любой специалист, немного знакомый с азами музыкальной грамоты, мог аналогично записать любую известную мелодию.

Одним из простейших применений описанного синтезатора мелодий является музыкальный звонок. На микроконтроллере C8051F300 фирмы Silicon Laboratories его можно сделать очень маленьким, с печатной платой, умещающейся в спичечном коробке, питанием от трех малогабаритных аккумуляторов (плоских GP или AAA) и что самое интересное, с множеством оригинальных мелодий и большими возможностями. В отличие от древнего отечественного семейства микросхем УМС07(08), выпускающегося уже более 20 лет, эта конструкция позволяет оперативно изменять мелодии (путем перепрограммирования). Кроме того, она свободна от ряда недостатков, присущих конструкциям на микросхемах серии УМС, главным из которых является сложность

организации автоматической смены мелодий, обусловленная отсутствием сигнала конца мелодии. Принципиальная схема музыкального звонка приведена на рис.1.

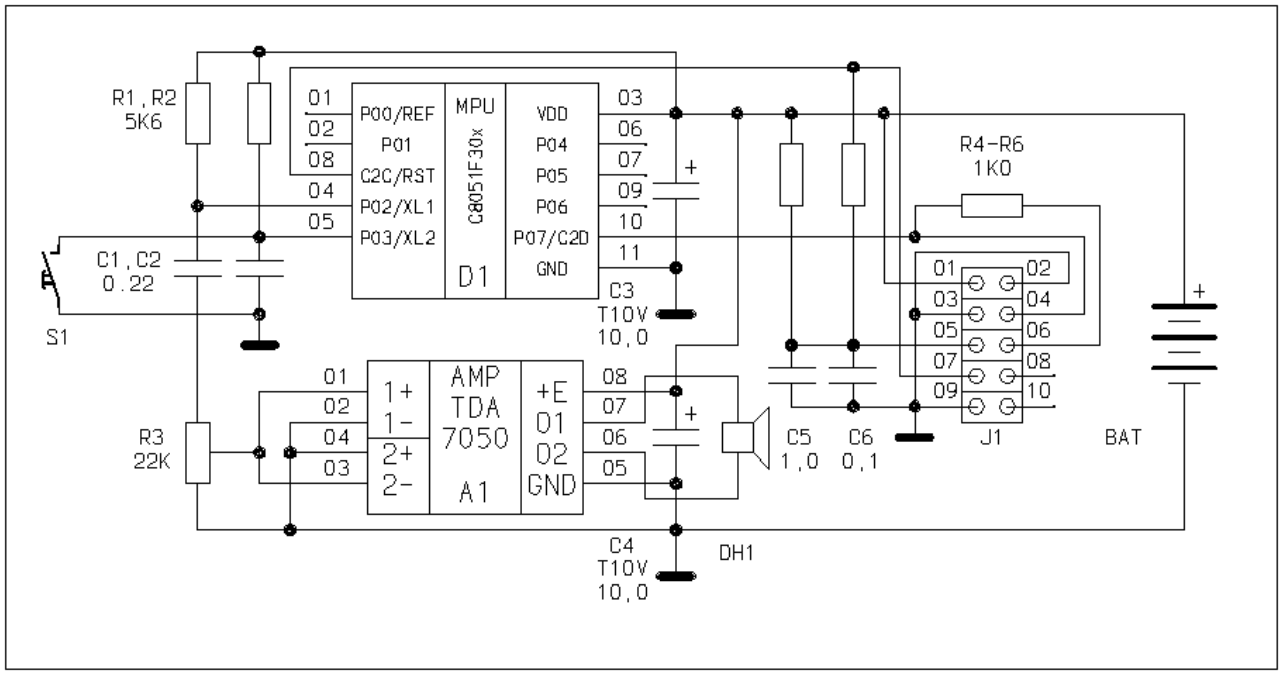

Рис.1. Музыкальный звонок на микроконтроллере C8051F300

Музыкальный звонок состоит из двух функциональных узлов: собственно микроконтроллера D1 (C8051F300) и усилителя низкой частоты A1 (TDA7050). Звонок питается от трех батареей или аккумуляторов AA с суммарным напряжением 3,6 В. Потребление в спящем режиме соответствует примерно 2,5 мА, в рабочем режиме – около 60 мА. Кнопка звонка S1 подключена на вход микроконтроллера P0.3. Программный синтезатор мелодий настроен на использование в качестве выхода – вывода P0.2. Музыкальная последовательность импульсов с выхода P0.2 поступает на регулятор громкости на резисторе R3 через разделительный конденсатор C1. С выхода регулятора громкости сигнал поступает на разнополярные входы усилителей A1, включенных по мостовой схеме. Резисторы R4-R6, конденсаторы C5,C5 и штыревой разъем J1 образуют цепи сброса и программирования/отладки интерфейса C2 (модифицированный JTAG интерфейс). Микроконтроллер D1 C8051F300 имеет объем программной Flash памяти – 8 Кбайт, что позволяет разместить в ней кроме самой программы до 150-200 мелодий. Наличие в музыкальном звонке микроконтроллера позволяет гибко изменять алгоритм работы звонка и переключение мелодий.

Приведенную схему можно еще улучшить за счет некоторого усложнения. Например, добавив два транзистора, образующих ключ питания, подаваемого на усилитель A1 и управляемого от микроконтроллера можно снизить потребление схемы в спящем режиме до десятков микроампер, что значительно увеличит срок службы батарей питания. Другие неиспользованные входы можно использовать для организации, например, световой индикации или для установки переключателя, выбирающего набор мелодий или длительность звучания.

Литература:

1. http://www.silabs.com/
2. http://www.jlworld.com