Усовершенствованный двухканальный индикатор уровня звука на базе цветного 1,3″ TFT-дисплея и микроконтроллера EFM8LB10F16

Алексей Кузьминов

В статье приведены принципиальная схема, разводка и внешний вид платы, а также программные средства двухканального индикатора уровня звука на базе цветного 1,3″ TFT-дисплея с разрешением 240×240 пикселей (с контроллером ST7789), сопряжённого с микроконтроллером EFM8LB10F16 по параллельному интерфейсу. Показаны результаты работы устройства в составе УМЗЧ.

Дополнительные материалы к этой статье вы можете скачать с нашего сайта по этой ссылке

Введение

Двухканальный индикатор уровня звука на базе дисплея OLED 1306 и микроконтроллера (МК) EFM8LB12, описанный автором в [1], показал хорошую работу, однако его недостатками являются: миниатюрный размер (видимая область экрана составляет всего 22×12 мм), малое разрешение (80×160 пикселей) и всего два цвета (жёлтый и синий) на чёрном фоне. Эти недостатки не позволяют в полной мере комфортно оценить уровни звука двух каналов усилителя, и, кроме того, такой дисплей смотрится в составе усилителя не особенно эстетично, а на большом расстоянии, например 2-3 метра, его трудно разглядеть, если специально не приглядываться. У автора возникла идея: а нельзя ли вместо дисплея OLED 1306 использовать цветной дисплей бóльшего размера, например, TFT 1,3″ с видимой областью экрана 25×25 мм и разрешением 240×240 пикселей? По стоимости (около 150 руб.) такой дисплей не дороже дисплея OLED 1306. Но использование такого TFT-дисплея связано с одной проблемой (к счастью, решаемой). Дело в том, что каждая строка OLED-дисплея состоит из 8 бит (или одного байта), и если такой бит равен 1, то он светится, а если нет, то он чёрный. Поэтому, если для передачи информации в такой дисплей использовать достаточно высокоскоростной интерфейс SPI (на скорости, например, 36 Мбод), то частота обновления экрана этого дисплея будет не менее 5 кГц [1]. В цветном же TFT-дисплее каждый пиксель состоит из трёх цветов (красный, зелёный и синий – RGB), и для передачи цвета пикселя требуется не бит (как в OLED-дисплее), а два байта, которые и определяют цвет пикселя. Другими словами, объём информации, передаваемой в каждый пиксель TFT-дисплея, в 16 раз превышает аналогичный объём, передаваемый в OLED-дисплей. Кроме того, поскольку разрешение TFT 1,3″ дисплея составляет 240×240 пикселей (против 80×160 пикселей у OLED-дисплея), объём передаваемой информации в TF-дисплей многократно возрастает. Пусть каждая гистограмма, отражающая уровень звука в TFT-дисплее, имеет, например, ширину 40 пикселей и высоту 240 пикселей и, таким образом, состоит из 40×240 = 9600 пикселей, а две гистограммы соответственно из 19 200 пикселей. Тогда для передачи двух байт в каждый пиксель потребуется передать 19 200×2 = 38 400 байт для двух гистограмм. Простой расчёт показывает, что при использовании интерфейса SPI даже на скорости 36 Мбод частота обновления экрана TFT-дисплея будет составлять не более двух десятков Гц, что, понятно, неприемлемо, поскольку дисплей будет мерцать с частотой, заметной для глаза. Выход из создавшегося положения может быть найден, если в качестве канала передачи информации в TFT-дисплей использовать параллельный однобайтный интерфейс, в котором за один такт МК передаётся сразу весь байт, а при частоте тактового генератора МК в 72 МГц один такт составляет не более 14 наносекунд, что нетрудно подсчитать. А раз так, то, как будет видно из дальнейшего изложения, частота обновления экрана TFT-дисплея будет составлять около 370 Гц, что в 2–4 раза превышает частоту обновления экрана современных телевизоров и мониторов (это, как правило, около 100 Гц, реже 200 Гц). Цветные 1,3″ TFT-дисплеи с параллельным интерфейсом также выпускаются, и их стоимость не превышает стоимость дисплеев с интерфейсом SPI (контроллер ST7789 допускает оба типа интерфейсов – параллельный и последовательный SPI).

Преимущества цветного 1,3″ TFT-дисплея перед дисплеем OLED 1306 очевидны: это бóльший размер и бóльшее разрешение, а возможность использовать разные цвета (например, цвет фона – синий, цвет гистограммы при уровне звука ниже перегрузки – зелёный и цвет гистограммы при уровне звука выше перегрузки – красный) создаёт более комфортное восприятие уровней звука.

Здесь необходимо добавить, что цветные OLED-дисплеи соответствующего разрешения и размера также выпускаются, но они стоят в несколько раз дороже цветного 1,3″ TFT-дисплея.

Таким образом, с использованием параллельного интерфейса сопряжения МК с TFT-дисплеем задача построения индикатора уровня звука вполне разрешима. Что и является предметом настоящей статьи.

Схема платы устройства

Принципиальная схема платы устройства (рис. 1) ненамного отличается от аналогичной схемы, приведённой в [1].

Отличия аналоговой части следующие. Во-первых, вместо двух сдвоенных

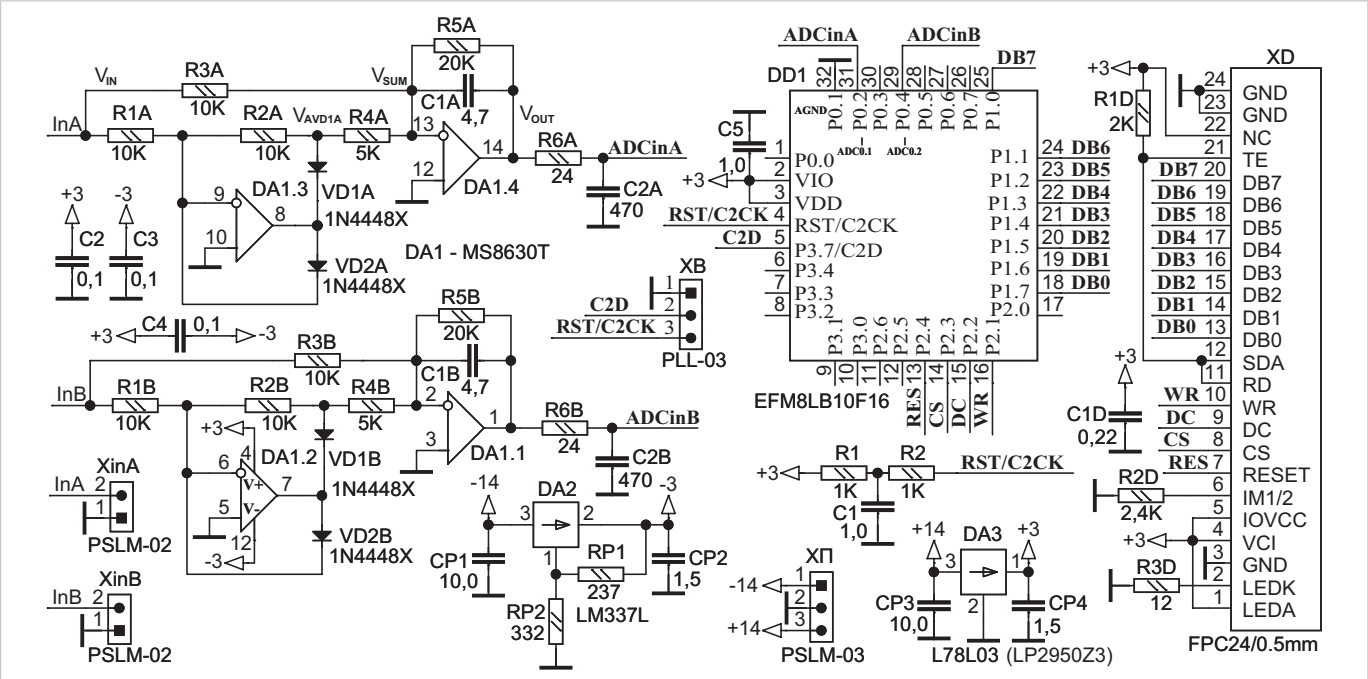

Рис. 1. Принципиальная схема платы устройства

ОУ MCP6002 в [1] применён счетверённый прецизионный и более высокоскоростной ОУ MS8630 (DA1) с «нулевым» смещением (zero drift), который по стоимости даже чуть меньше стоимости двух ОУ MCP6002. Напряжение смещения (Offset Voltage) MS8630 составляет всего 2 мкВ против ±7 мВ у MCP6002, а типовая скорость MS8630 составляет 1,25 В/мкс (3,8 МГц) против 0,6 В/мкс (1 МГц) у MCP6002. Кроме того, ОУ MS8630 с обвязкой занимает немного меньше места на плате, чем 2 ОУ MCP6002, в связи с чем размер платы несколько уменьшился (см. далее). Во-вторых, вместо диодов 1N4148 применены более современные диоды 1N4448 (VD1A–VD2B) с меньшим прямым падением напряжения. Всё это позволило поднять частоту выпрямленного напряжения с 10 кГц (в [1]) до 20 кГц (если не включать конденсаторы C1A и C1B, которые совместно с ОУ образуют интегратор или ФНЧ).

Принцип работы аналоговой части аналогичен принципу работы, описанному в [1], поэтому, чтобы не повторяться, он не приводится.

В цифровой части отличия следующие.

Вместо МК EFM8LB12F64 в корпусе QFN24 с программной памятью 64 кБ (в [1]) применён более дешёвый МК EFM8LB10F16 (DD1) в корпусе QFN32 с программной памятью 16 кБ (как будет видно из дальнейшего изложения, программа для МК занимает около 1,7 кБ).

Интерфейс с дисплеем – параллельный однобайтный. Данные (байт) выводятся с порта P1 (P1.0–P1.7) – сигналы DB7–DB0. В дисплей также передаются сигналы: WR (запись данных), DC (данные/команда), CS (выбор кристалла – Chip Select) и RES (RESET – сброс дисплея).

Шлейф дисплея подключён к 24-контактному разъёму FPC0.5-24-03 (XD) с верхним расположением контактов.

Как следует из справочного листка (datasheet) ST7789, на неиспользуемые выводы (контакты XD) дисплея требуется подавать лог. 1. В связи с этим выводы XD TE, SDA и RD подключены к резистору R1D, который соединён с питанием (+3 В). Вывод IM1/2 определяет, какой тип интерфейса (SPI или параллельный) используется. При подаче лог. 0 на вывод IM1/2 используется параллельный интерфейс, поэтому этот вывод подключён к резистору R2D, соединённому с «землёй». Выводы IOVCC и VCI (питание дисплея) и анод светодиода подсветки (LEDA) подключены к питанию (+3 В). Катод светодиода подсветки (LEDK) подключён к «земле» через резистор R3D; в этом случае ток через светодиод подсветки составляет около 13 мА, что соответствует типовому значению. В некоторых случаях в подобных дисплеях устанавливают транзистор с соответствующей обвязкой, на базу которого подают сигнал BLK, позволяющий использовать ШИМ для уменьшения потребления тока (как правило, в устройствах с батарейным питанием). Но в данном случае необходимости в таком транзисторе нет, и подсветка включена постоянно.

Как можно заметить из схемы, биты данных (сигналы DB0–DB7) подключены к порту P1 таким образом, что бит P1.0 соответствует биту DB7, P1.1 – DB6, … , P1.7 – DB0, или, другими словами, байт, выводимый через порт P1, имеет обратный порядок расположения бит по сравнению с порядком бит, требующимся для правильной передачи информации в дисплей. Эта особенность связана исключительно с оптимальностью разводки платы (см. далее), а обратный порядок бит может быть легко скомпенсирован программными средствами, при этом, как будет видно из дальнейшего изложения, на скорость обмена информацией МК с дисплеем эта особенность абсолютно не повлияет.

Питание платы (DA2, DA3, разъём ХП) и программирование МК (разъём XB, цепочка R1R2C1) организованы аналогично устройству, описанному в [1].

Разводка и внешний вид платы

Разводка (рис. 2а, б) сделана автором с помощью программы Sprinl LayOut V.6. Размер платы по сравнению с платой в [1] уменьшился на 1 мм (25×25 мм против 25×26 мм в [1]). Сама плата (рис. 2в) приклеена пористой лентой с двусторонним липким слоем к стеклотекстолитовой пласти-

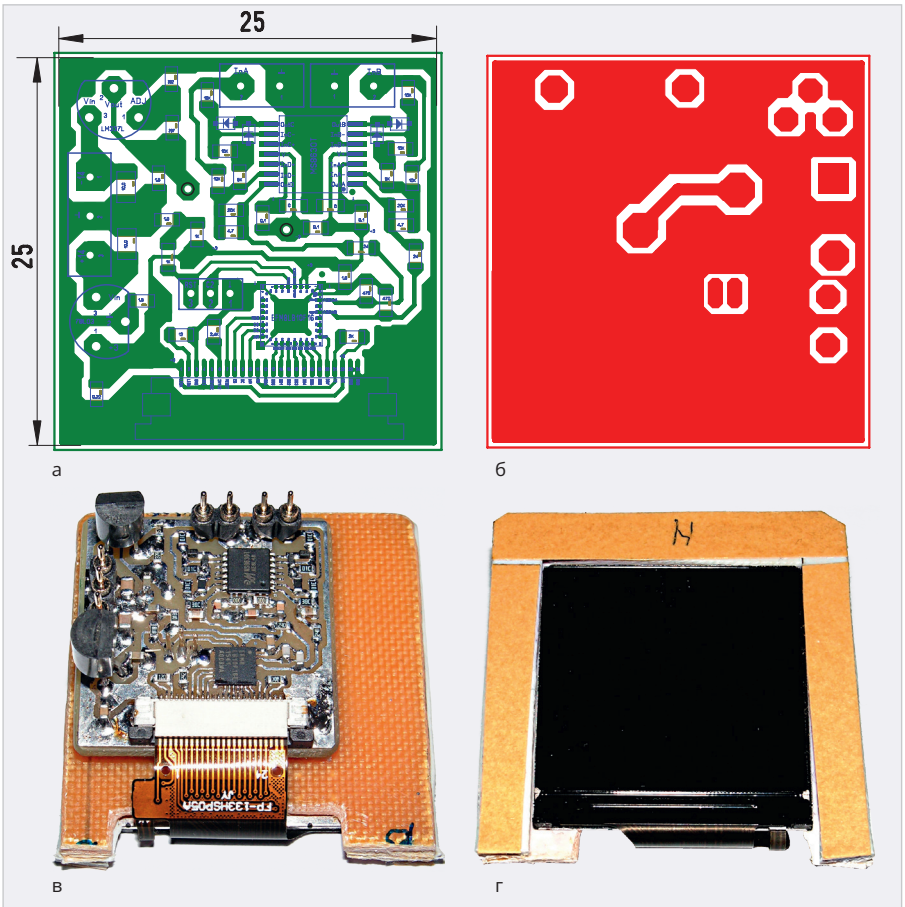

Рис. 2. Разводка и внешний вид платы: а, в – вид со стороны расположения компонент; б, г – вид с обратной стороны; д – укрупнённый вид МК

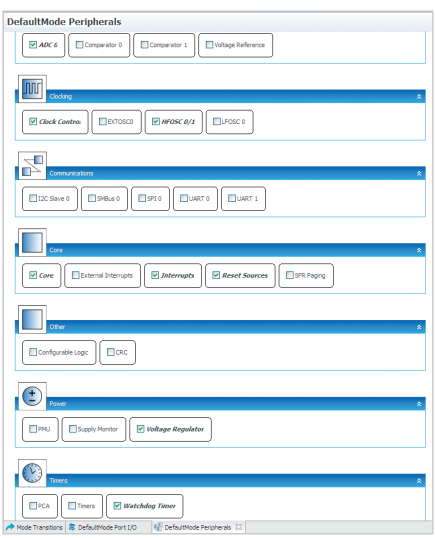

Рис. 3. Общее меню настроек

не, размерами на 5 мм больше размеров дисплея. Дисплей (рис. 2г) приклеен с обратной стороны к этой же пластине тонкой лентой с двусторонним липким слоем. Его шлейф перегнут через вырез в пластине и вставлен в соответствующий разъём на плате. По трём сторонам пластины к ней также тонкой лентой с двусторонним липким слоем приклеены три пластиковые пластины толщиной 1,5 мм, что приблизительно на 0,15–0,2 мм больше толщины дисплея. Это сделано для того, чтобы после приклейки этих пластин к внутренней поверхности лицевой панели корпуса усилителя дисплей не касался этой поверхности во избежание повреждения. Как можно заметить из рис. 2г, с ленты ещё не снят защитный слой вощёной бумаги (жёлтого цвета). Она будет снята непосредственно перед приклейкой устройства.

Как можно заметить из разводки (рис. 2а), внешнего вида платы (рис. 2в) и укрупнённого вида МК (рис. 2д), контакты дисплея (DB0–DB7) и контакты порта P1 (P1.0–P1.7) МК расположены в обратном порядке (об этом уже упоминалось выше), что позволило сделать дорожки, соединяющие эти контакты, наиболее короткими. Это же касается и всех остальных дорожек для сигналов (RES, DC, CS и WR). При тактовой частоте МК 72 МГц длительность некоторых импульсов составляет не более 14 нс, и, чтобы эти импульсы подавались без искажений, длина дорожек должна быть как можно меньше, что и сделано. А обратный порядок расположения бит, как уже упоминалось выше, скомпенсирован программными средствами (см. далее).

Здесь необходимо добавить следующее. При изготовлении платы в контактные площадки, отмеченные чёрными кружками на разводке (рис. 2а), необходимо вставить тонкий (0,2–0,3 мм) лужёный медный провод и пропаять его с двух сторон платы. Контактные площадки всех разъёмов и выводов микросхем стабилизаторов также необходимо пропаять с двух сторон платы. Кроме того, на торцевых поверхностях корпуса МК по углам расположены небольшие контактные площадки (по 2 в каждом углу), которые соединены с «землёй» МК. Эти контактные площадки необходимо соединить припоем с контактными площадками на плате, расположенными с трёх сторон корпуса МК и отмеченными на разводке как «земляные» («GND» и перевёрнутая буква «Т») – см. рис. 2д.

Программные средства

Программирование МК проводилось в среде Simplisity Studio v.4.0 на языке C51 (Keil 8051 v.9.54.0).

Общее меню настроек (рис. 3) отличается от аналогичного меню в [1] только тем, что из него исключён интерфейс SPI. Все остальные настройки (кроме портов МК) – те же, что и в [1], поэтому, чтобы не повторяться, они не приводятся.

Порты (рис. 4) настраиваются следующим образом. P0.1, P0.2 и P0.4 настраиваются как аналоговые входы (Analog I/O, пример – на рис. 4б) и, кроме того, должны быть пропущены командами skip (они отмечены красными крестиками).

Порты P1.0–P1.7 и P2.1–P2.4 настраиваются как цифровые выходы (Digital Push-Pull Output, пример – на рис. 4в).

Все остальные неиспользуемые порты настраиваются как цифровые входы (Digital OpenDrain I/O – пример на рис. 4г).

Таким образом, порты корпуса рис. 4а как раз и соответствуют разводке – рис. 2д.

Теперь по поводу подпрограмм (п/п) и основной программы.

Вначале об обратном порядке бит в байте. В Интернете автор обнаружил массу п/п, которые изменяют порядок бит в байте на обратный. Однако все они состоят из нескольких операторов, включая операторы сдвига и некоторые другие. Естественно, такие п/п выполняются достаточно долго и поэ-

тому автора не устроили. Единственно, что автору понравилось, – это интересная п/п, основанная на табличном преобразовании. Она приведена ниже. Её суть в том, что входной байт (uint8_t x) является индексом соответствующего элемента массива (code uint8_t table[]), расположенного в памяти программ (code) и состоящего из 256 байт. Например, если x = 0x00, то в нулевом элементе массива расположен искомый байт, который в данном случае также равен 0x00. Если x = 0x01 = 00000001_2, то искомый байт должен быть равен 10000000_2 = 0x80, чему и равен 1-й элемент массива. Если x = 0x02 = = 00000010_2, то искомый байт должен быть равен 01000000_2 = 0x40, чему и равен 2-й элемент массива, и так далее, вплоть до последнего элемента массива, который равен 0xff.

Обращение к этой п/п очень простое: если требуется получить байт (byte) с обратным порядком бит, то исходный байт (byte_in) требуется подставить в качестве аргумента в эту функцию: byte = = reverse_byte(byte_in);. Естественно, такая таблица потребует дополнительно 256 байт программной памяти, однако, как будет видно из дальнейшего изложения, это абсолютно не критично, так как вся программа занимает около 1,7 кБ программной памяти МК, составляющей 16 кБ. С другой стороны, эта п/п выполняется очень быстро, так как состоит всего из одного оператора (листинг 1).

Для вывода байта по параллельному интерфейсу используется следующая п/п (листинг 2), которая состоит всего из трёх команд: в порт P1 выводится исходный байт (byte), далее бит записи WR устанавливается в высокий уровень, а затем в низкий.

П/п (листинг 3) используется для вывода однобайтных команд и данных в дисплей при его инициализации и установки области экрана, в которую необходимо записать информацию. С использованием п/п reverse_byte(byte_in) и outbyte(uint8_t byte), описанных выше, обе п/п представлены ниже. Обращение к этим двум п/п, естественно, занимает некоторое время, однако, поскольку п/п вывода команды и данных используются только один раз, это время некритично. Это же касается п/п установки области экрана (см. далее), так как она используется не более 3 раз при построении каждой гистограммы.

Другое дело, если требуется вывести большой массив 2-байтных данных в каждый пиксел гистограммы. В этом случае обращение как к п/п вывода байта (outbyte(uint8_t byte)), так и к п/п reverse_byte(byte_in) будет занимать существенное время, поскольку указанный массив данных насчитывает несколько тысяч байт (см. далее). Поэтому, во-первых, поскольку про-

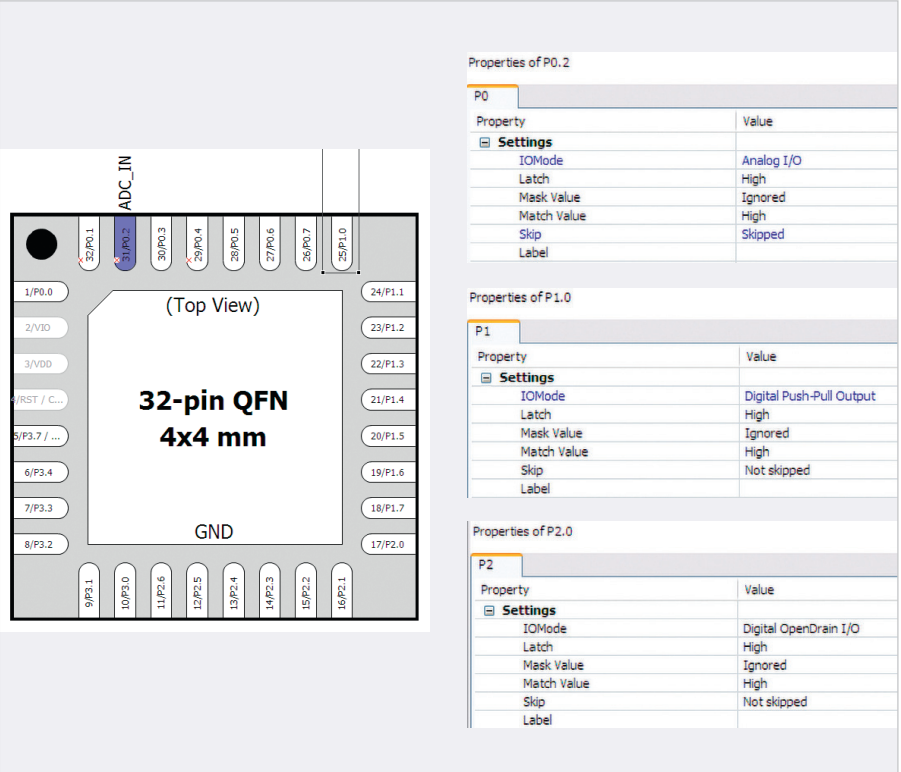

Рис. 4. Настройка портов МК: а – корпус МК с портами, б – аналоговые входы, в – цифровые выходы, г – цифровые входы

Листинг 1. Подпрограмма инвертирования байта

```
//---------------------------
uint8_t reverse_byte(uint8_t x) {
 code uint8_t table[] = {
  0x00, 0x80, 0x40, 0xc0, 0x20, 0xa0, 0x60, 0xe0,
  0x10, 0x90, 0x50, 0xd0, 0x30, 0xb0, 0x70, 0xf0,
  0x08, 0x88, 0x48, 0xc8, 0x28, 0xa8, 0x68, 0xe8,
  0x18, 0x98, 0x58, 0xd8, 0x38, 0xb8, 0x78, 0xf8,
  0x04, 0x84, 0x44, 0xc4, 0x24, 0xa4, 0x64, 0xe4,
  0x14, 0x94, 0x54, 0xd4, 0x34, 0xb4, 0x74, 0xf4,
  0x0c, 0x8c, 0x4c, 0xcc, 0x2c, 0xac, 0x6c, 0xec,
  0x1c, 0x9c, 0x5c, 0xdc, 0x3c, 0xbc, 0x7c, 0xfc,
  0x02, 0x82, 0x42, 0xc2, 0x22, 0xa2, 0x62, 0xe2,
  0x12, 0x92, 0x52, 0xd2, 0x32, 0xb2, 0x72, 0xf2,
  0x0a, 0x8a, 0x4a, 0xca, 0x2a, 0xaa, 0x6a, 0xea,
  0x1a, 0x9a, 0x5a, 0xda, 0x3a, 0xba, 0x7a, 0xfa,
  0x06, 0x86, 0x46, 0xc6, 0x26, 0xa6, 0x66, 0xe6,
  0x16, 0x96, 0x56, 0xd6, 0x36, 0xb6, 0x76, 0xf6,
  0x0e, 0x8e, 0x4e, 0xce, 0x2e, 0xae, 0x6e, 0xee,
  0x1e, 0x9e, 0x5e, 0xde, 0x3e, 0xbe, 0x7e, 0xfe,
  0x01, 0x81, 0x41, 0xc1, 0x21, 0xa1, 0x61, 0xe1,
  0x11, 0x91, 0x51, 0xd1, 0x31, 0xb1, 0x71, 0xf1,
  0x09, 0x89, 0x49, 0xc9, 0x29, 0xa9, 0x69, 0xe9,
  0x19, 0x99, 0x59, 0xd9, 0x39, 0xb9, 0x79, 0xf9,
  0x05, 0x85, 0x45, 0xc5, 0x25, 0xa5, 0x65, 0xe5,
  0x15, 0x95, 0x55, 0xd5, 0x35, 0xb5, 0x75, 0xf5,
  0x0d, 0x8d, 0x4d, 0xcd, 0x2d, 0xad, 0x6d, 0xed,
  0x1d, 0x9d, 0x5d, 0xdd, 0x3d, 0xbd, 0x7d, 0xfd,
  0x03, 0x83, 0x43, 0xc3, 0x23, 0xa3, 0x63, 0xe3,
  0x13, 0x93, 0x53, 0xd3, 0x33, 0xb3, 0x73, 0xf3,
  0x0b, 0x8b, 0x4b, 0xcb, 0x2b, 0xab, 0x6b, 0xeb,
  0x1b, 0x9b, 0x5b, 0xdb, 0x3b, 0xbb, 0x7b, 0xfb,
  0x07, 0x87, 0x47, 0xc7, 0x27, 0xa7, 0x67, 0xe7,
  0x17, 0x97, 0x57, 0xd7, 0x37, 0xb7, 0x77, 0xf7,
  0x0f, 0x8f, 0x4f, 0xcf, 0x2f, 0xaf, 0x6f, 0xef,
  0x1f, 0x9f, 0x5f, 0xdf, 0x3f, 0xbf, 0x7f, 0xff
 };
 return table[x];
}
```

Листинг 2. Подпрограмма вывода байта по параллельному интерфейсу

```
void outbyte(uint8_t byte) {
 P1 = byte;
 WR = 1;
 WR = 0;
}
```

Листинг 3. Подпрограммы вывода однобайтных команд и данных в дисплей

```
//----------------------
void outcmd(uint8_t byte_in) {//вывод команды
 uint8_t byte;
 byte = reverse_byte(byte_in);
 DC = 0;
 CS = 0;
 outbyte(byte);
 CS = 1;
}

//--------------------------------
void outdat(uint8_t byte_in) {//вывод данных
 uint8_t byte;
 byte = reverse_byte(byte_in);
 DC = 1;
 CS = 0;
 outbyte(byte);
 CS = 1;
}
```

Листинг 4. Подпрограмма записи двух байтов цвета

```
//----outdat16-------------
void outdat16(uint16_t wor) {
 U.US = wor;
 CS = 0;
 P1 = U.UB[0]; //Ст.б.
 WR = 1;
 WR = 0;
 CS = 1;
 CS = 0;
 P1 = U.UB[1];          //Мл.б.
 WR = 1;
 WR = 0;
 CS = 1;
}
//------------------
```

Листинг 5. Подпрограммы работы с дисплеем

```
//------------------
// определение области экрана для заполнения
void SetWindow(uint8_t startX, uint8_t startY, uint8_t stopX, uint8_t
stopY) {
 outcmd(0x2A);
 outdat(0x00);
 outdat(startX);
 outdat(0x00);
 outdat(stopX);
 outcmd(0x2B);
 outdat(0x00);
 outdat(startY);
 outdat(0x00);
 outdat(stopY);
}
//-----------------------

//-----------------------
// заливка всего дисплея цветом
void LCD_Fill(uint16_t color) {
 uint16_t i; //16
 SetWindow(0, 0, 239, 239); outcmd(0x2C);
 DC = 1;
 for (i = 0; i < 57600; i++) {//240*240=57600
  outdat16(color);
 }
 outcmd(0x2C);
 outcmd(0x29);
}

//----------------------------

// заливка области экрана цветом
void FillWin(uint8_t startX, uint8_t startY,

uint8_t stopX, uint8_t stopY, uint16_t color) {
 uint16_t i, j;
 SetWindow(startX, startY, stopX, stopY);
 outcmd(0x2C);
 j = ((stopX - startX) + 1) * ((stopY - startY) + 1);
 DC = 1;
 for (i = 0; i < j; i++) {
  outdat16(color);
 }
 outcmd(0x2C);   // запись в память дисплея
}
```

грамма использует всего три цвета: синий (BLUE), зелёный (GREEN) и красный (RED), имеет смысл заранее перевернуть все байты этих цветов (их всего 6 штук). Автором это сделано вручную и представлено ниже. В этом случае обращение к п/п reverse_byte(byte_in) уже не потребуется, и указанные двухбайтные слова побайтно будут выводиться напрямую в порт P1.

```
//#define BLUE  0x000f //синий
#define BLUE 0x00f0 //синий инверсный
//#define RED  0xF880 //красный
#define RED  0x1F01 //красный инверсный
//#define GREEN 0x07E0 //зелёный
#define GREEN 0xE007 //зелёный инверсный
```

Во-вторых, чтобы не обращаться к п/п outbyte(uint8_t byte), процедура записи байта в дисплей (манипуляции битами CS, WR и вывода байта в порт P1) внесена в само тело п/п вывода двухбайтного кода цвета. Для этого вначале установлено совмещение двухбайтного слова с массивом из двух однобайтных элементов:

```
 / / - - - - - - - - - -
union {
 uint16_t US;      // U.UB[0]
- Ст.б.
 uint8_t UB[1];    // U.UB[1]
- Мл.б.
 } U; // U.US - 2-байтное
uint16_t число.
 //----------
```

С использованием этого совмещения п/п записи двух байтов цвета в каждый пиксель представлена на листинге 4.

Как видно из этой п/п, запись каждого из двух байтов происходит за 5 тактов (начиная с CS = 0; по CS = 1;), каждый из которых при тактовой частоте 72 МГц занимает около 14 нс, т.е. всего 14 нс × 5 = 70 нс. Если каждая гистограмма имеет ширину 40 пикселей и высоту 240 пикселей, т.е. состоит из 9600 пикселей, то для передачи двух байтов в каждый пиксел двух гистограмм потребуется передать 9600 × 2 × × 2 = 38 400 байтов (см. выше). Умножив это количество байтов на 70 нс, получим: 38 400 × 70 нс = 2 688 000 нс = = 2,688 мс ≈ 2,7 мс, что соответствует частоте обновления экрана дисплея F = 1/2,7 мс = 0,3704 кГц ≈ 370 Гц, что, как уже говорилось, в 2–4 раза превышает частоту обновления экранов современных телевизоров и мониторов (100–200 Гц).

Следующие три п/п (листинг 5) определяют область экрана для заполнения данными, заливку всего дисплея определённым цветом (она входит в п/п инициализации дисплея) и заливку определённой области экрана определённым цветом.

П/п АЦП в настоящей программе та же самая, что и в [1]; она подробно

описана, поэтому, чтобы не повторяться, не приводится. Здесь необходимо только напомнить, что максимальное значение напряжения, измеренного АЦП, составляет 2,2 В, что соответствует реальному напряжению в 1,1 В, поскольку коэффициент усиления выпрямителя на основе DA1 (рис. 1) равен 2.

По п/п вывода в дисплей гистограммы в зависимости от показания АЦП и номера канала необходимо сделать следующие пояснения.

Поскольку разрешение дисплея составляет 240×240 пикселей, ширина каждой из двух гистограмм принята равной 40 пикселям, а для того чтобы обе гистограммы располагались на дисплее симметрично относительно его центра, расстояние от левого и правого края области экрана дисплея до начала горизонтального расположения гистограмм принято также равным 40 пикселям. Тогда, поскольку отсчёт количества пикселей относительно начала области экрана по горизонтали начинается с нуля (и идёт справа налево), правая гистограмма должна располагаться с 39 по 79 пиксел, а левая – со 159 по 199 пиксел.

Теперь по поводу высоты гистограмм. Поскольку максимальное значение показания АЦП по каждому каналу составляет 2,2 В или 2200 мВ (см. выше), а максимальное разрешение дисплея по вертикали составляет 240 пикселей, имеет смысл принять максимальную высоту гистограмм равной 220 пикселей. В этом случае каждый пиксель будет соответствовать 2200 мВ/220 = 10 мВ. А для того чтобы получить реальное значение высоты гистограммы в пикселях, NXP, требуется напряжение UL, измеренное АЦП, разделить на 10: NXP = = UL/10. Кроме того, как было принято в [1], порог напряжения, измеренного АЦП, при котором начинается область перегрузки, принят равным 0,9 В, или, учитывая коэффициент усиления выпрямителя, равный 2, это напряжение составляет 1,8 В или 1800 мВ, что составляет 180 пикселей (см. выше). В этом случае, если NXP < 179, то область гистограммы с нуля по NXP закрашивается зелёным цветом, а с NXP по 220-й пиксель – синим, т.е. как фон дисплея. В противном случае область ниже порога, т.е. с 0 по 179-й пиксель (так как отсчёт начинается с нуля) закрашивается зелёным цветом, выше, т.е. со 180-го пикселя по NXP – красным, а с NXP по максимальное значение 239-й пиксель – синим, т.е. также фоном дисплея.

Листинг 6. Подпрограмма вывода гистограмм

```
//---------------------------------
void outUL(uint16_t UL, uint8_t NK) { //UL - показание АЦП в мВ
 uint8_t XS, XE, YS, YE, NPX;
 if (NK == 1) { // левая гистограмма
  YS = 159;

  YE = 199;
 }

 else {
  YS = 39;  // правая гистограмма
  YE = 79;
 }
//-------------------
 NPX = UL / 10;
 if (NPX < 179){
  XS = 0;
  XE = NPX;
  FillWin(XS, YS, XE, YE, GREEN);
  XS = NPX + 1;
  XE = 220;
  FillWin(XS, YS, XE, YE, BLUE);
   }

 else {
  XS = 0;

  XE = 179;
  FillWin(XS, YS, XE, YE, GREEN);
  XS = 180;
  XE = NPX + 1;
  FillWin(XS, YS, XE, YE, RED);
  XS = NPX + 2;
  XE = 239;
  FillWin(XS, YS, XE, YE, BLUE);
 }
//----------------------------
} // конец п/п
//-----------------------------------
```

П/п вывода двух гистограмм приведена на листинге 6, и, с учётом вышеприведённого объяснения, её несложно понять.

Основная программа

Эта программа (листинг 7) не отличается от программы в [1] и приведена ниже для сведения. Как можно заметить, она зациклена (последний её оператор goto A).

После трансляции всей программы в среде Simplisity Studio внизу экрана появляется следующее сообщение:

```
  Program  Size:  data=34.1
xdata=0 const=256 code=1490
  LX51  RUN  COMPLETE.  0
WARNING(S), 0 ERROR(S)
  Finished building target:
EFM8LB10F16E-C-QFN32.omf
```

Из этого сообщения следует, что программа использует всего 34 байта внутренней оперативной памяти с прямой адресацией (data=34.1), размер которой 128 байт, внешняя дополнительная оперативная память с косвенной адресацией размером 1280 байт не используется вообще (xdata=0), размер кодовой части программы составляет 1490 байт (code=1490) плюс константы 256 байт (const=256) – это та самая таблица для получения обратного расположения бит в байте, итого вся область программной памяти составляет 1490 + 256 = 1746 байт, т.е. около 1,7 кБ, что на порядок меньше всей программной памяти МК (16 кБ). Программа использует так называемую small model, где все данные располагаются в памяти с прямой адресацией (data) и которая работает наиболее быстро.

Программа в уже готовом загрузочном *.hex-формате (EFM8LB10F16E-C-QFN32.hex) приведена в дополнительных материалах к статье на сайте журнала.

Конструкция и результаты работы устройства

Как было упомянуто выше, стеклотекстолитовая пластина с дисплеем и платой устройства (рис. 2г) приклеена к внутренней поверхности лицевой части корпуса усилителя. Для этого в ней прорезано прямоугольное окно по размеру видимой области экрана дисплея (рис. 5). Кабели питания и двух сигналов вставлены в соответствующие контакты платы так же, как и в [1] (в связи с простотой фотография не приводится).

По общему виду лицевой поверхности корпуса работающего усилите-

Листинг 7. Основная программа

```
// main() Routine
// ---------------------------------------------------------------------
----------
int main(void) {
// Call hardware initialization routine
 enter_DefaultMode_from_RESET();
//------------------
 WR = 0;
 CLS(); // Инициализация ЖКИ
 A:

 U1 = ACP(1);
 DEL2US();
 U2 = ACP(2);
 U1 = U1 * 3; //2-Ку ИУ;2-Ку ОУ выпрямителя; 0,75-Ку АЦП МК
 U2 = U2 * 3; //2*2*0,75=3.
 outUL(U1, 1); //левый канал
 outUL(U2, 2); //правый канал
 goto A;
}
```

ля о реальной работе устройства можно судить лишь косвенно по фотографии, т.е. в статике. Поэтому, для того чтобы реально убедиться в работе устройства, в дополнительных материалах к статье приведены два видео *.mov-файла длительностью 10 (PICT0001.MOV) и 14 (PICT0003.MOV) секунд, снятые фотокамерой с двух разных ракурсов. При просмотре файла PICT0001.MOV обратите внимание, что в некоторые моменты зажигаются светодиоды перегрузки, расположенные справа (над ручками регулировки громкости (см. рис. 5)). Эти два видеофайла – фрагменты композиции «Embrace» Armin van Buuren'a из музыкальной категории в стиле «trance», снятые фотокамерой при работе усилителя, когда к нему был подключён телефон, который воспроизводил реальный звуковой *.mp3-файл. При этом регулятор громкости на усилителе и движок громкости на телефоне были установлены почти на максимум, чтобы показать работу устройства при перегрузке, когда верхняя часть гистограмм окрашивается в красный цвет.

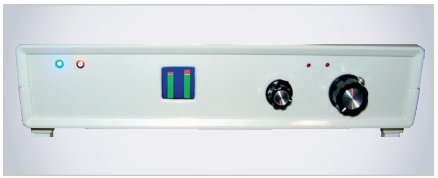

Рис. 5. Внешний вид работающего усилителя с индикатором уровней звука

Заключение

Применение МК EFM8LB10, счетверённого относительно высокоскоростного ОУ с «нулевым» смещением MS8630 и цветного 1,3" TFT дисплея с разрешением 240×240 пикселей с контроллером ST7789 позволило сконструировать малогабаритный и малоинерционный индикатор уровня звука, который показал отличную работу в составе аудиоусилителя. По стоимости он в несколько раз дешевле аналогичных готовых покупных устройств. Автор рекомендует его для повторения.

Литература

1. *Кузьминов А.* Двухканальный индикатор уровня звука на базе микроконтроллера EFM8LB12 и дисплея OLED 1306 // Современная электроника. 2024. № 4.