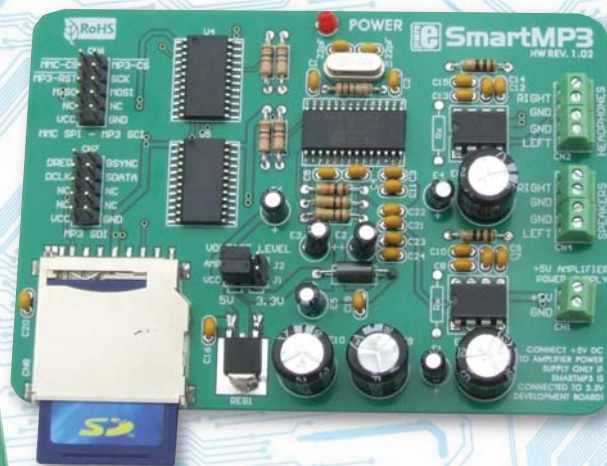


OK. Vama je potreban ... MP3 plejer



SmartMP3 modul povezan sa razvojnim sistemom LV24-33A

Milan Rajić
MikroElektronika - Sektor za razvoj softvera

Upotreba MP3 formata je dovela do revolucije u kompresiji digitalnog zvuka omogućujući da audio fajlovi budu i po nekoliko puta manji. Ako želite da u vaš projekat uključite zvučne poruke ili muziku, videćete da je to veoma lako. Potrebna vam je bilo koja standardna MMC ili SD memorijska kartica, nešto hardvera i malo vremena...

Za razvoj softvera i njegovo testiranje korišćeni su razvojni sistem LV24-33A i modul Smart MP3. Pre početka rada, potrebno je formatirati MMC karticu i na nju snimiti fajl *sound1.mp3* (kartica se formatira u FAT16, tj. FAT formatu).

Kvalitet reprodukcije zvuka u MP3 formatu zavisi od brzine semplovanja i bitrejt-a. Kao i audio CD, većina MP3 fajlova je semplovana frekvencijom 44.1 kHz. Bitrejt MP3 fajla nam označava koliko je verno kodiran originalni zvuk. Za reprodukciju govora je dovoljno 64 kbit/s dok se za muziku koristi bitrejt 128 kbit/s ili viši. U našem primeru smo koristili muzički fajl sa bitrejtom 128 kbit/s.

Hardver

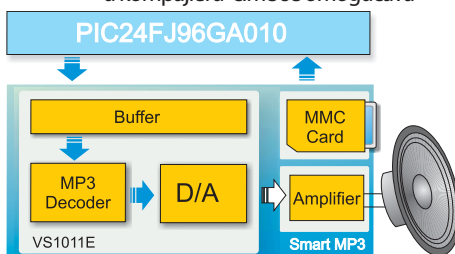
Zvuk u ovom fajlu je kodiran u MP3 formatu tako da nam je za njegovu reprodukciju potreban MP3 dekodera. U našem primeru, za ovu svrhu se koristi čip VS1011E. Ovo kolo dekoduje MP3 zapis i vrši digitalno-analognu konverziju signala tako da se nakon obrade dobija signal koji se preko malog audio pojačavača može dovesti na zvučnike.

Zbog činjenice da MMC/SD kartice koriste sektore sa 512 bajtova, za kontrolu rada MP3 dekodera je potreban mikrokontroler sa više od 512 bajtova RAM-a. Mi smo izabrali PIC24FJ96GA010 koji ima 1536 bajtova RAM memorije.

Softver

Rad programa koji upravlja ovim uređajem može se podeliti u pet koraka:

- Korak 1:** Inicijalizacija SPI modula mikrokontrolera.
- Korak 2:** Inicijalizacija Mmc_FAT16 biblioteke u kompajleru čime se omogućava



Slika 1. Blok dijagram Smart MP3 modula sa mikrokontrolerom PIC24FJ96GA010

čitanje MP3 fajlova sa MMC ili SD kartica.

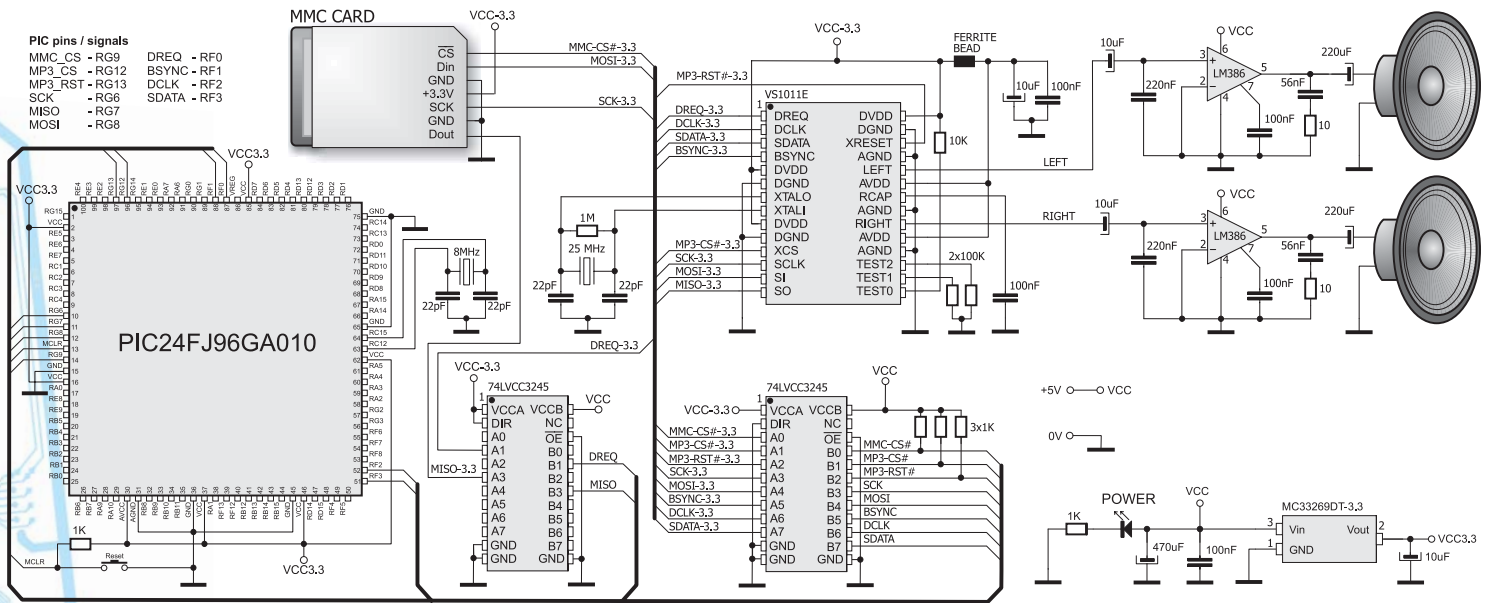
Korak 3: Čitanje dela fajla.

Korak 4: Slanje podataka u bafer MP3 dekodera.

Korak 5: Ako nije završena reprodukcija kompletnog fajla, vraćamo se na korak 3.

Testiranje

Testiranje rada uređaja je bolje početi sa manjim bitrejt faktorom pa ga zatim postepeno povećavati. Naime, bafer MP3 dekodera je veličine 2048 bajtova. Ako taj bafer popunimo delom MP3 fajla bitrejt 128 kbit/s, on će sadržati dva puta više odbiraka zvuka nego kada bismo u njega upisali deo fajla bitrejt 256 kbit/s. Samim tim, zvuk koji je u baferu će se reprodukovati duplo duže kod fajla sa manjim bitrejtom. Ako suviše povećamo bitrejt fajla može se desiti da se zvuk iz bafera reprodukuje pre nego što mikrokontroler stigne da pročita sledeći deo fajla sa kartice i upiše ga u bafer, zbog čega će zvuk biti isprekidan. Ako se to desi, možemo smanjiti bitrejt MP3 fajla



Šema 1. Povezivanje modula Smart MP3 sa mikrokontrolerom PIC24FJ96GA010

la koristeći neki od audio editora ili upotrebiti kristal frekvencije više od 8MHz (pogledajte šemu 1).

U svakom slučaju, o ovome ne morate previše brinuti jer je naš program testiran na nekoliko familija mikrokontrolera sa različitim vrednostima kristala i bio je u stanju da reprodukuje MP3 fajlove srednjeg i visokog kvaliteta.

Sa druge strane, mali bitrejt znači da bafer dekodera punimo zvukom dužeg trajanja. Može se desiti da dekodera ne reprodukuje zvuk iz bafera pre nego što mi pokušamo da ga ponovo napunimo. Da bismo to izbegli, pri slanju podataka uvek proveravamo da li je dekodera spreman za prijem novih podataka. Stoga u funkcijama za slanje podataka čekamo da data request signal dekodera (DREQ) bude na logičkoj jedinici (1).

Proširenja programa

Kada isprobamo ovaj primer možemo ga proširiti. DREQ signal se može periodično proveravati. Može se dodati rutina za kontrolu jačine zvuka, za kontrolu ugrađenog Bass/Treble enhancer-a itd. MMC biblioteka nam daje mogućnost da odaberemo fajl sa nekim drugim imenom. Na taj način možemo stvoriti bazu MP3 poruka, zvukova ili pesama koje ćemo koristiti u našoj aplikaciji i poslati odgovarajući MP3 fajl dekodera u zavisnosti od situacije.

Na donjoj slici je prikazana lista funkcija koje se nalaze u biblioteci *Mmc_FAT16*. Ova biblioteka je sastavni deo kompajlera *mikroPASCAL for dsPIC* compiler.

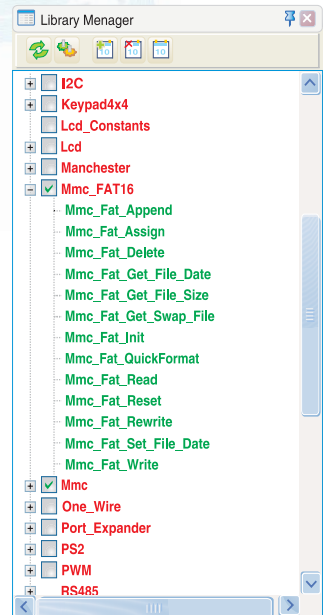
Primer 1: Program za demonstraciju rada modula Smart MP3

```

program MP3_Simple_Test;
const BUFFER_SIZE = 2048; // global variables
var filename : string(13);
    i, file_size : dword;
    data_buffer : array(32) of byte;
    BufferLarge : array(BUFFER_SIZE) of byte;
procedure SW_SPL_Write(data_ : byte); begin // Writes one byte to MP3 SDI
PORTF.1 := 1; // Set BSYN before sending the first bit
PORTF.2 := 0; PORTF.3 := data_0; PORTF.2 := 1; // bitorder is LSB first
PORTF.2 := 0; PORTF.3 := data_1; PORTF.2 := 1;
PORTF.1 := 0; // Clear BSYN after sending the second bit
PORTF.2 := 0; PORTF.3 := data_2; PORTF.2 := 1;
PORTF.2 := 0; PORTF.3 := data_3; PORTF.2 := 1;
PORTF.2 := 0; PORTF.3 := data_4; PORTF.2 := 1;
PORTF.2 := 0; PORTF.3 := data_5; PORTF.2 := 1;
PORTF.2 := 0; PORTF.3 := data_6; PORTF.2 := 1;
PORTF.2 := 0; PORTF.3 := data_7; PORTF.2 := 1;
PORTF.2 := 0;
end;
// Writes one word to MP3 SCI
procedure MP3_SCI_Write(address : byte; data_in : word); begin
PORTG.12 := 0; // select MP3 SCI
Spi2_Write(0x03); // Read command
Spi2_Write(address);
Spi2_Write(Hi(data_in));
Spi2_Write(Lo(data_in));
PORTG.12 := 1; // deselect MP3 SCI
while (PORTF.0 = 0) do nop; // wait until DREQ becomes 1, see MP3 codec
datasheet, Serial Protocol for SCI
end;
// Reads words_count words from MP3 SCI
procedure MP3_SCI_Read(start_address, words_count : byte; data_buffer : ^byte);
var i : byte;
begin
PORTG.12 := 0; // select MP3 SCI
Spi2_Write(0x03); // Read command
Spi2_Write(start_address);
for i := 1 to (2*words_count) do
begin
data_buffer[i] := Spi2_Read(0); // read and store a byte
Inc(data_buffer); // point to next byte
end;
PORTG.12 := 1; // deselect MP3 SCI
while (PORTF.0 = 0) do nop; // wait until DREQ becomes 1, see MP3 codec
datasheet, Serial Protocol for SCI
end;
procedure MP3_SDI_Write(data_ : byte); begin // Write one byte to MP3 SDI
while (PORTF.0 = 0) do nop; // wait until DREQ becomes 1, see MP3 codec
datasheet, Serial Protocol for SCI
SW_SPL_Write(data_);
end;
procedure MP3_SDI_Write_32(data_ : ^byte); // Write 32 bytes to MP3 SDI
var i : byte;
begin
while (PORTF.0 = 0) do nop; // wait until DREQ becomes 1, see MP3 codec
datasheet, Serial Protocol for SCI
for i := 1 to 32 do begin
SW_SPL_Write(data_[i]); // Write byte pointed by data
end;
end;
procedure Set_Clock(clock_khz : word; doubler : byte); begin // Set clock
clock_khz := clock_khz / 2; // calculate value
if (doubler > 0) then clock_khz := clock_khz * 2;
MP3_SCI_Write(0x03, clock_khz); // Write value to CLOCKF register
end;
procedure Soft_Reset(); begin // Software Reset
MP3_SCI_Write(0x00, 0x204); // Set SM_RESET bit and SM_BITORD bit(bitorder is LSB first)
Delay_us(2); // Required, see MP3 codec datasheet -> Software Reset
while (PORTF.0 = 0) do nop; // wait until DREQ becomes 1, see MP3 codec
datasheet, Serial Protocol for SCI
for i := 1 to 2048 do MP3_SDI_Write(0); // feed 2048 zeros to the MP3 SDI bus
end;
procedure Init(); begin // Software Reset
ADPCFG := 0x0FFF; // set all AN pins to digital
PORTF.2 := 0; TRISF.2 := 0; // Clear SW SPI SCK, configure pin as output
PORTF.3 := 0; TRISF.3 := 0; // Clear SW SPI SDA, configure pin as output
PORTG.12 := 1; TRISG.12 := 0; // Deselect MP3_CS, configure pin as output
PORTG.13 := 1; TRISG.13 := 0; // Set MP3_RST pin, configure pin as output
TRISF.1 := 1; PORTF.1 := 0; // Configure DREQ as input
PORTF.1 := 0; TRISF.1 := 0; // Clear BSYN, configure pin as output
end;
begin // main function
filename := 'sound1.mp3'; // Set File name
Init();
Spi2_Init_Advanced(SPI_MASTER, SPI_8_BIT, SPI_PRESCALE_SEC_1, SPI_PRESCALE_PRI_64,
_SPI_SS_DISABLE, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_HIGH, _SPI_ACTIVE_2);
IDLE;
Soft_Reset(); Set_Clock(25000, 0); // SW Reset, set clock to 25MHz
if (Mmc_Fat_Init(PORTG.9) = 0) then
if (Mmc_Fat_Assign(filename, 0) <> 0) then
begin
Mmc_Fat_Reset(file_size); // Call Reset before file reading,
// procedure returns size of the file
// send file blocks to MP3 SDI
while (file_size > BUFFER_SIZE) do begin
for i := 0 to BUFFER_SIZE - 1 do Mmc_Fat_Read(BufferLarge[i]);
for i := 0 to BUFFER_SIZE / 32 - 1 do MP3_SDI_Write_32(@BufferLarge + i * 32);
file_size := file_size - BUFFER_SIZE;
end;
// send the rest of the file to MP3 SDI
for i := 0 to file_size - 1 do Mmc_Fat_Read(BufferLarge[i]);
for i := 0 to file_size - 1 do MP3_SDI_Write(BufferLarge[i]);
end;
end;

```

Napisano u kompajleru mikroPASCAL for dsPIC



Mmc_Fat_Append()	Dodaj podatke na kraj fajla
Mmc_Fat_Assign()*	Pripremi fajl za FAT operacije
Mmc_Fat_Delete()	Obriši fajl
Mmc_Fat_Get_File_Date()	Vрати datum i vreme fajla
Mmc_Fat_Get_File_Size()	Vрати veličinu fajla
Mmc_Fat_Get_Swap_File()	Napravi swap fajl
Mmc_Fat_Init()*	Inicijalizuj karticu za FAT operacije
Mmc_Fat_QuickFormat()	Formatiraj karticu
Mmc_Fat_Read()*	Pročitaj podatke iz fajla
Mmc_Fat_Reset()*	Pripremi fajl za čitanje
Mmc_Fat_Rewrite()	Pripremi fajl za pisanje
Mmc_Fat_Set_File_Date()	Setuj datum i vreme fajla
Mmc_Fat_Write()	Upiši podatke u fajl

* Mmc_FAT16 funkcije korišćene u programu

Ostale funkcije kompajlera *mikroPASCAL for dsPIC* korišćene u programu:

Spi_Init_Advanced() Inicijalizuj SPI modul mikrokontrolera

Pored proširene verzije ovog programa koji je napisan za dsPIC® mikrokontrolere, sa našeg sajta možete preuzeti i verzije pisane za PIC® i AVR® mikrokontrolere www.mikroe.com/en/article/