University of British Columbia
Electrical and Computer Engineering
Electrical and Biomedical Engineering Design Studio
ELEC291/ELEC292

Adding WAV Sound to the AT89LP51RC2

Introduction

Occasionally a microcontroller project needs to produce a reasonable good quality sound output. One example of such a project could be an electronic clock for the visually impaired that would speak out the time after a push button is pressed. Other examples include devices that must provide feedback or warnings to users that may not be looking or paying continuous attention to the operating device. Fortunately, adding sound to a microcontroller project is not too difficult when the microcontroller system includes a Digital to Analog Converter (DAC). Since the storage of encoded sound requires significant amounts of memory, an external memory is often used. The idea is to store the encoded sound into the external memory. That stored sound can be reproduced by means of an interrupt service routine and the DAC with very little computing overhead for the microcontroller system. This document describes how to add sound to an 8051 microcontroller (AT89LP51RC2), but the same technique should be portable to other architectures as well.

Generating and Editing a Sound WAV File

The Waveform Audio File Format (WAV) was developed by Microsoft and IBM to play sound in personal computers. It is the main audio format used on Microsoft Windows operating systems for raw and usually uncompressed audio. The typical encoding used with WAV files is the linear pulse-code modulation (LPCM) format. In a LPCM audio stream, the amplitude of the analog signal is sampled at fixed intervals, and each sample is quantized within a range of a given digital resolution. The WAV file supports both mono and stereo sound. In this application, the WAV file sample rate is assumed to be 22050 Hz, the digital resolution is assumed to be 8-bits, and the number of sound tracks stored is assumed to be one (mono sound). A WAV file stored using this format will optimally use an 8-bit flash memory, and it is the most convenient format to use for an 8-bit microcontroller like the 8051. The microcontroller used for playback should be able to accurately reproduce the sampling rate of the WAV file using an Interrupt Service Routine (ISR).

The WAV file can be generated by recording a person's voice using a personal computer or phone while running an appropriate APP. For personal computers, a popular program to create and edit audio files is Audacity. Audacity is a free program that can be downloaded from:

https://www.audacityteam.org/download/

Alternatively, the audio file can be generated using the speech synthesizer software of a personal computer. In the Microsoft Windows operating system, this functionality can be accessed programmatically, for example, by using Visual Basic Scripts (VSB) such as the one below (talk.vbs) which outputs the words "Hello, world!" using the computer speakers.

```
Dim  oVoice
Set oVoice = CreateObject("SAPI.SpVoice")
Set oVoice.Voice = oVoice.GetVoices.Item(0)
oVoice.Rate = -1
oVoice.Volume = 100
oVoice.Speak "Hello, world!"
```

The output of a VBS can be redirected to a WAV file instead of the speakers as shown in the next script (Wav_from_Text.vbs):

```
Dim WavFileName, StringToConvert
Dim oFileStream, oVoice

'NORMAL for values between 0 and 39
'stereo = add 1
'16-bit = add 2
'8KHz = 4
'11KHz = 8
'12KHz = 12
'16KHz = 16
'22KHz = 20
'24KHz = 24
'32KHz = 28
'44KHz = 32
'48KHz = 36

StringToConvert = InputBox("Enter String to Convert:", "Convert String to WAV")
WavFileName = InputBox("Enter Filename to save to:", "Convert String to WAV")

Const WavFormat= 20 ' 22KHz sampling rate, 8-bit, mono
Const SSFMCreateForWrite = 3 ' Creates file even if file exists and so destroys or
overwrites the existing file

Set oFileStream = CreateObject("SAPI.SpFileStream")
oFileStream.Format.Type = WavFormat
oFileStream.Open WavFileName, SSFMCreateForWrite

Set oVoice = CreateObject("SAPI.SpVoice")
oVoice.Rate = 0
oVoice.Volume = 100
Set oVoice.AudioOutputStream = oFileStream
oVoice.Speak StringToConvert

oFileStream.Close
```

In the preceding script, the WAV file is generated using the format we require (22050Hz, 8-bit, mono) and no further processing is required. In some computers it is possible to select the talking voice (male or female for example) and the spoken language (English, Spanish, etc.). A quick web search will show you how to do that.

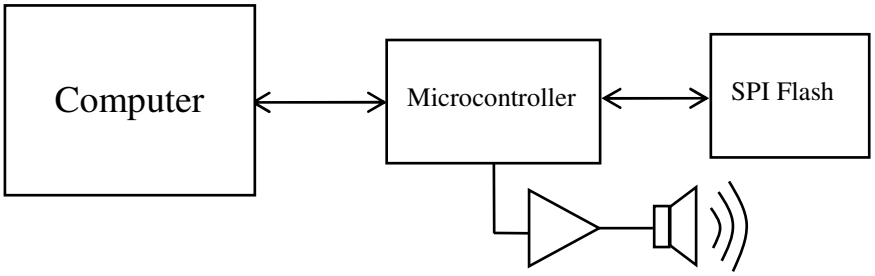
Another way of generating the sound file need for your project is by using some dedicated web engines. To create and save a sound voice file you can visit IBM's Watson Text to Speech at https://text-to-speech-demo.ng.bluemix.net/ or 'Text 2 Speech' at https://www.text2speech.org/. In both these web sites, the 'save file' option is available after right clicking the play button. Both sites generate MP3 compressed files. To convert the MP3 file to a WAV file with the desired format, you can either use Audacity as mentioned above, or visit an on-line audio format converter such as https://audio.online-convert.com/convert-to-wav.

Storing the Sound WAV File into an External SPI Flash Memory

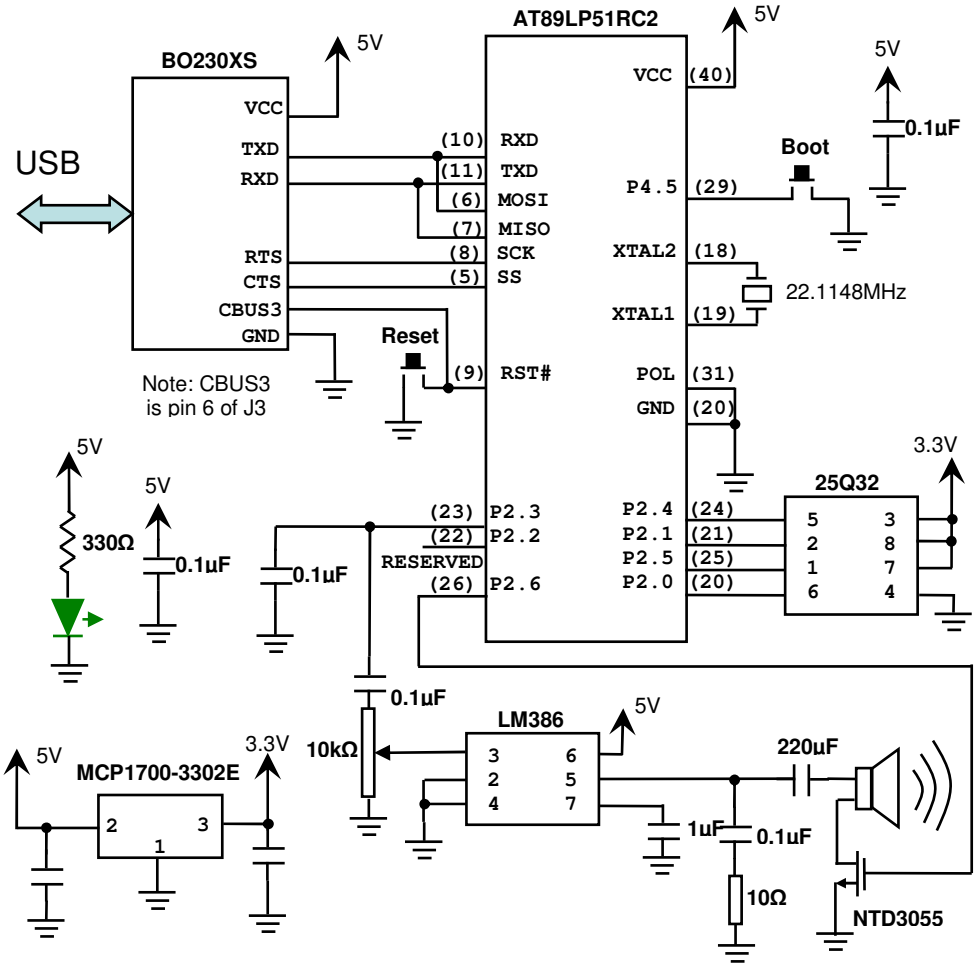
In general, a few seconds of audio stored using the WAV format will require several hundred kilo-bytes of memory storage. If the sampling rate is 22050Hz, each second of LPCM audio takes 22050 bytes when sampled with a resolution of 8-bits. Small general purpose microcontrollers generally have just a few kilobytes of flash memory available for program storage without significant storage space available to

accommodate these large WAV files. Fortunately it is very simple and inexpensive to add extra flash memory storage to a microcontroller system. For this application we will be using SPI flash memory to store the LPCM audio contained in a WAV file.

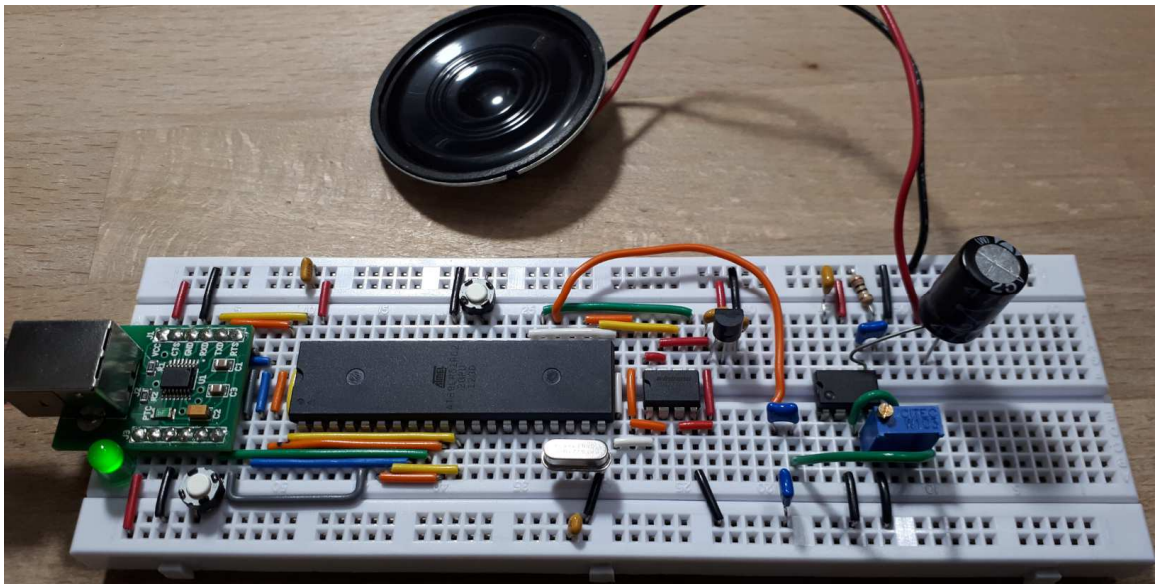
A simple approach to store the WAV file into the SPI flash memory is by using the target hardware directly. Under this scheme we use a dedicated microcontroller program (firmware) to communicate with the computer, obtain the WAV file via the serial port, and store the received file into the flash memory using SPI communication. This process is illustrated in the figure below.



The figure also shows a power amplifier connected to one of the DAC outputs of the microcontroller. The amplifier is connected to a speaker which produces the output sound. The DAC/amplifier/speaker combination is used in the final application program, but can be also used with the flash programming software to test the stored WAV file. A more detailed block diagram using the AT89LP51RC2 microcontroller is shown below.



In the diagram above, the SPI memory is the 25Q32 with a capacity of 32 Mega bits or 4 Mega bytes. The power amplifier is the LM386. Pressing the BOOT push-button will play back the content of the whole flash memory. The AT89LP51RC2 includes a differential DAC which is used in this application. For simplicity, only one of the differential outputs is used (P2.3, pin 23) and the other output should be left unconnected and not used for anything else (P2.2, pin 22). The picture below shows the AT89LP51RC2 setup that matches the diagram above (except for the NTD3055 MOSFET, which is used to remove speaker noise when sound is not being played).



The firmware (program that runs in microcontroller) used to program the SPI flash memory using the AT89LP51RC2 microcontroller is written in C language and named 'AT89LP51RC2_Receiver.c'. It should be available in the course web page. Compile and load the program as usual into the microcontroller.

To send the WAV file to the microcontroller, a program that runs on Microsoft Windows named 'Computer_Sender.c' (together with the executable 'Computer_Sender.exe') should be also available in the course web page. If you want to recompile this program you can use Visual C from a command prompt:

```
C:\Source>cl Computer_Sender.c
```

or similarly, if GCC is installed in your computer:

```
C:\Source>gcc Computer_Sender.c -o Computer_Sender.exe
```

The program can be also compiled and run from both Linux and macOS. Instructions are included at the end of this document.

The program 'Computer_Sender.exe' is run from a command prompt. To load the flash memory the following command is executed:

```
C:\Source>Computer_Sender -DCOM9 -W -V Clock22KHz_Mono.wav
```

You should replace 'COM9' with the number assigned by the operating system to the USB adapter in your system. Also, instead of typing 'Clock22KHz_Mono.wav' use your own WAV file. It may take several minutes to load the flash memory depending on the size of the WAV file. The output generated by the program looks as it follows:

```
C:\Source>Computer_Sender -DCOM9 -W -V Clock22KHz_Mono.wav
SPI flash memory programmer using serial protocol. (C) Jesus Calvino-Fraga (2012-2019)
Sample rate: 22050Hz
Loaded 508360 bytes from file 'Clock22KHz_Mono.wav'.
Manufacturer: 0xef, Type: 0x40, Size: 0x16
Memory has 4194304 bytes
Erasing flash memory...
 Done.
Writing flash memory...
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.. Done.
Actions completed in 56.7 seconds.
Verifying SPI flash content......
The CRC of the file and the SPI flash memory is the same: 0x05af.
Actions completed in 2.7 seconds.
```

At this point the SPI flash memory is loaded with the WAV file. If we wish to play the whole WAV file we type the following:

```
C:\Source>Computer_Sender -DCOM9 -P
```

If we want to generate an approximate index file (in c for the 8051) of sounds separated by silences, we can type the following command:

```
C:\Source>Computer_Sender -Cmyindex.c Clock22KHz_Mono.wav
```

The index file (in this case 'myindex.c') can be used in the final application to locate and play specific sounds. The program 'Computer_Sender.exe' has more options. To see the options available add to the command line '-?':

```
C:\Source>Computer_Sender -?
SPI flash memory programmer using serial protocol. (C) Jesus Calvino-Fraga (2012-2019)
Usage examples:
Computer_Sender -DCOM3 -w somefile.wav (write 'somefile.wav' to flash via COM3)
Computer_Sender -DCOM3 -w -I somefile.wav (write 'somefile.wav' to flash via COM3, do not
check for valid WAV)
Computer_Sender -DCOM3 -v somefile.wav (compare 'somefile.wav' and flash)
Computer_Sender -DCOM3 -Rotherfile.wav (save content of SPI flash to 'otherfile.wav')
```

```
Computer_Sender -DCOM3 -P (play the content of the flash memory)
Computer_Sender -DCOM3 -P0x20000,12540 (play the content of the flash memory starting at
address 0x20000 for 12540 bytes)
Computer_Sender  -Amyindex.asm  somefile.wav  (generate  asm  index  file  'myindex.asm'  for
'somefile.wav')
Computer_Sender  -Cmyindex.c  somefile.wav  (generate  C  index  file  'myindex.c'  for
'somefile.wav'.)
Computer_Sender -Cmyindex.c -S2000 somefile.wav (same as above but check for 2000 silence
bytes.  Default is 512.)
```

Playing the WAV File in the Project

Once the SPI flash memory is loaded with the WAV file, the firmware 'AT89LP51RC2_Receiver.asm' can be replaced with the application program. To reproduce sound, you can follow the method used in 'AT89LP51RC2_Receiver.c'. It is very simple and takes just a few lines of assembly code. What you need is to set a timer ISR routine to run at 22050Hz. In 'AT89LP51RC2_Receiver.c' Timer 1 interrupt is used for this purpose. Every time the ISR is executed (after the timer overflows) a byte is read from the flash memory and copied into the DAC.

Support for Linux and macOS

'Computer_Sender.c' compiles and runs also on Linux and macOS. To compile use GCC on a terminal shell:

```
gcc Computer_Sender.c -o Computer_Sender
```

The command line for all three OSs is the same, but the name of the serial port is different:

For **Windows** use:

-DCOM3 (or whatever port number is assigned to the USB adapter)

For **Linux** use:

-D/dev/ttyUSB0

You have to be a member of group 'dialout' for this to work. To join 'dialout' on Linux type:

```
sudo adduser $USER dialout
```

You have to log out and log back in for the changes to take effect.

For **macOS** use:

-D/dev/cu.usbserial-DN05FVT8

To find the name of the device you have use in macOS, type in a terminal shell:

```
ls -l /dev | grep "usb"
```

When executing the program in Linux type something like this:

```
./Computer_Sender -D/dev/ttyUSB0 -W -V Clock22KHz_Mono.wav
```

When executing the program in macOS type something like this:

```
./Computer_Sender -D/dev/cu.usbserial-DN05FVT8 -W -V Clock22KHz_Mono.wav
```