AVR Freaks

DESIGN NOTE #039

AUTHOR: ANTHONY BARRETT

KEYWORDS: PCM, SOUND, WAV, CONVERTION

Sound Sample Play Back

Ever wanted to add sound effects or voice sample play back to your AVR project? Well here is a simple interrupt driven sound sample play back routine for the 8515. It uses minimal external hardware i.e., just connect an Audio Transducer to PB1. The sound sample data is compressed into 1-bit samples (On or Off) which allows you to fit 1 second of 16 kHz sample play back data into less than 2K bytes of Program Memory. That is approximately a total of four seconds worth of play back time for an 8515. The sound quality is quite reasonable considering its 1-bit PCM. This routine also supports variable sample rates, i.e., 8 kHz, 11 kHz, 16 kHz, 22 kHz, 32 kHz, 44 kHz, and 48 kHz. So how does it work? Well, the process is divided into four easy steps:

1. First step is to create an 8-bit PCM mono wave file. Either record your voice and save it as an 8-bit PCM mono wave file or convert existing samples to 8-bit PCM mono wave format. You can use a sound editor application to trim the leading and lagging silence from your samples to keep the sample size to a minimum. After much experimenting a sample rate of 16kHz appears to be the best. You can use lower rates if you wish but it will compromise the sound quality significantly. Using sample rates above 16 kHz only makes a slight improvement but the trade off is a larger sample data size and hence less play back time. The maximum sample size is 65,535 bytes (0xFFFF) due to the 16-bit byte counter limit. This gives you a maximum play back per sample of 32.5 seconds at 16 kHz. But if that's not enough you can spilt larger samples into several smaller samples and play them back to back, assuming you have got a big enough AVR.
2. Once you have your samples ready you need to convert them into compressed 1-bit PCM AVR assembler source files which you can included straight into the Sound Sample Play Back routine source file. To do this I have written a simple console program in C called "Wave2AVR.exe". Just specify the input and output filenames and it does all the work for you.

   For example: Wave2AVR Sample.wav Sample.asm [t]

   Where "t" is an optional threshold setting. The threshold value is how much above and below the zero amplitude level will define a "1" (On) or "0" (Off) state. After some testing the default value of three seems to give the best results. But you can experiment by supplying your own threshold value if you wish. The output is a text assembler source file made up of Word defines (you can edit this file if you wish). This allows you to include the sample data directly into the Sound Sample Play Back source file and assemble it. Note that the conversion program does not check the Input File format so if you try to convert any file other than 8-bit PCM mono wave format you will get a garbage output file.

3. Now you need to edit the Sound Sample Play Back assembler source file “Play-Back.asm” to include your newly generated sample data file(s). You can include as many different sample data files as you can fit into the AVR’s Program Memory. They are included at the end of the source file. See the Play Back source file comments for details. Now assemble and program your AVR.
4. The final step is to connect an Audio Transducer to the output pin PB1. You can connect it straight to the AVR if you wish, i.e., one wire to pin PB1 and the other to GND. But it is good practise to put a series resistor in with the Transducer for protection. A 1k resistor works well. You could also try using a speaker instead. A speaker will give you better sound quality. When connecting a speaker you will need a coupling capacitor and resistor both connected in series with the speaker. A 270Ω resistor and approx. 100 uf capacitor works well. The 270Ω resistor prevents you from over loading the AVR’s output pin and the 100 uf capacitor decouples the DC component from the AVR. Okay, now that you are all connected you can hear your sample by either resetting the AVR or connecting a “Start” button to PB0.

All files including the C source file for the “Wave2AVR.exe” conversion program have been given to “www.AVRFreaks.com” for you to download in a file called “PlayBack.zip”. Feel free to make improvements or expand on the conversion program. All I ask is that all improvements be returned for free distribution to all. Below is the source file for the AVR 8515 but can be modified for other AVR's.

Code

```
;         *********************************
;         *                               *
;         *     Sound Sample Play Back    *
;         *                               *
;         *   AVR AT90S8515, 8MHz Clock   *
;         *                               *
;         *          28-Oct-2002          *
;         *    Version 1.00, AVRStudio4   *
;         *                               *
;         *       By Anthony Barrett      *
;         *                               *
;         *********************************
;
;
; This program will play back sound sample data generated by
; "Wave2AVR.exe".  See application note for details.
;
; Port pin descriptions:
;
;  PB0 -> Replay button
;  PB1 -> Transducer/Speaker output


;Includes:

.include "8515def.inc"
```

```
;Define SRAM variables:

.equ SRAMStart = $0060
.equ SamplePtrL = SRAMStart
.equ SamplePtrH = SamplePtrL + 1
.equ SampleCountL = SamplePtrH + 1
.equ SampleCountH = SampleCountL + 1
.equ SampleBitCount = SampleCountH + 1
.equ SampleByte = SampleBitCount + 1
.equ SampleRate = SampleByte + 1
.equ SampleComplete = SampleRate + 1


;Vector table:
           rjmp  RESET               ;Reset handler
           rjmp  EXT_INT0            ;IRQ0 handler
           rjmp  EXT_INT1            ;IRQ1 handler
           rjmp  TIM1_CAPT           ;Timer1 capture handler
           rjmp  TIM1_COMPA          ;Timer1 compareA handler
           rjmp  TIM1_COMPB          ;Timer1 compareB handler
           rjmp  TIM1_OVF            ;Timer1 overflow handler
           rjmp  TIM0_OVF            ;Timer0 overflow handler
           rjmp  SPI_STC             ;SPI transfer Complete handler
           rjmp  UART_RXC            ;UART RX complete handler
           rjmp  UART_DRE            ;UART UDR empty handler
           rjmp  UART_TXC            ;UART TX complete handler
           rjmp  ANA_COMP            ;Analog comparator handler


;Initalize:

RESET:     ldi   r16,high(RAMEND)    ;Set stack pointer
           out   SPH,r16
           ldi   r16,low(RAMEND)
           out   SPL,r16

           ldi   r16,0b00000001      ;Setup PORTB
           out   PORTB,r16           ;PB0 pullup on
           ldi   r16,0b00000010      ;PB1 output
           out   DDRB,r16

           ldi   r16,0b00000010      ;Set Timer0 pre-scale to Clk/8
           out   TCCR0,r16


;Start interupts:
```

```
            sei                         ;Start ints


;Main Progarm loop:

Main:       ldi   ZL,low(2 * Sample)    ;Z points to sample data
            ldi   ZH,high(2 * Sample)
            rcall PlaySample            ;Start play back

WaitLoop:   lds   r16,SampleComplete    ;Wait for sample to finish
            tst   r16
            breq  WaitLoop

WaitLoop2:  sbic  PINB,0                ;Wait for button press
            rjmp  WaitLoop2
            rjmp  Main


;Handlers:

      EXT_INT0:   reti
      EXT_INT1:   reti
      TIM1_CAPT:  reti
      TIM1_COMPA: reti
      TIM1_COMPB: reti
      TIM1_OVF:   reti

TIM0_OVF:   push  r16                   ;Save r16
            in    r16,SREG              ;Save status reg
            push  r16

            lds   r16,SampleRate        ;Restart Timer0 count
            out   TCNT0,r16

            lds   r16,SampleByte        ;Ouput next bit
            sbrc  r16,0
            sbi   PORTB,1
            sbrs  r16,0
            cbi   PORTB,1

            ror   r16                   ;Setup next bit
            sts   SampleByte,r16

            lds   r16,SampleBitCount    ;Check bit count
            inc   r16
            cpi   r16,8
            brne  T0Exit
```

```
                push  ZL                ;Saved used regs
                push  ZH

                lds   ZL,SampleCountL   ;Check sample count
                lds   ZH,SampleCountH
                sbiw  ZL,1
                sts   SampleCountL,ZL
                sts   SampleCountH,ZH
                brne  T0GetNextByte

                ldi   r16,0b00000000    ;Disable Timer0 ints (stop)
                out   TIMSK,r16

                ser   r16               ;Signal sample complete
                sts   SampleComplete,r16
                rjmp  T0Done

T0GetNextByte:
                push  r0                ;Save r0

                lds   ZL,SamplePtrL     ;Get next byte
                lds   ZH,SamplePtrH
                adiw  ZL,1
                sts   SamplePtrL,ZL
                sts   SamplePtrH,ZH
                lpm
                sts   SampleByte,r0

                pop   r0                ;Restore r0

T0Done:         pop   ZH                ;Restore used regs
                pop   ZL

                clr   r16               ;Restart bit count

T0Exit:         sts   SampleBitCount,r16

                pop   r16               ;Restore stat reg
                out   SREG,r16
                pop   r16               ;Restore r16
                reti


          SPI_STC:    reti
          UART_RXC:   reti
          UART_DRE:   reti
          UART_TXC:   reti
```

```
ANA_COMP:  reti


;Subroutines:

PlaySample: lpm                         ;Z -> Sample data
            sts   SampleCountL,r0       ;Get sample size
            adiw  ZL,1
            lpm
            sts   SampleCountH,r0
            adiw  ZL,1

            lpm                         ;Get sample rate (TCNT0 value)
            sts   SampleRate,r0
            adiw  ZL,1
            adiw  ZL,1                  ;Skip high byte

            sts   SamplePtrL,ZL         ;Save sample pointer
            sts   SampleptrH,ZH

            clr   r16                   ;Restart bit count
            sts   SampleBitCount,r16

            sts   SampleComplete,r16    ;Clear complete flag

            lpm                         ;Get first byte
            sts SampleByte,r0

            ser   r16                   ;Restart Timer0 count
            out   TCNT0,r16

            ldi   r16,0b00000010        ;Enable Timer0 ints (start)
            out   TIMSK,r16
            ret

;Include sample data file(s) generated by "Wave2AVR.exe":

Sample:

.include "D:\Sample.asm"

;Sample2:

;.include "D:\Sample2.asm"
```