

Project Title: My Z8 Can Speak! Z4229 - Continuous Variable Slope Delta modulation vocoder implementation using the Z8Encore micro controller

This project is a proof of concept that the Z8Encore ADC and PWM peripherals can be used for voice processing with good results.

Delta modulation has evolved into a simple, efficient method of digitizing voice for secure, reliable communications and for voice I/O in data processing.

Companded PCM, for telephone quality transmission, requires about 64K bits/sec data rate per channel.

CVSD produces at 16K bits/sec an acceptable reconstructed voice but the sound is reminiscent of a damaged loudspeaker.

Because of the simplicity of the algorithm is easy to implement CVSD on a small CPU like Z8 and get positive results. Implementing the digital version of CVSD has some advantages over the analog counterpart: no analog integrator, the time constants are set by the clock frequency and do not drift with time or temperature.

The integrators are initialized at 0 each time the process starts.

The Delta encoder / decoder is model is the one proposed by Greefkes and Riemens in 1970 and known as CVSD.

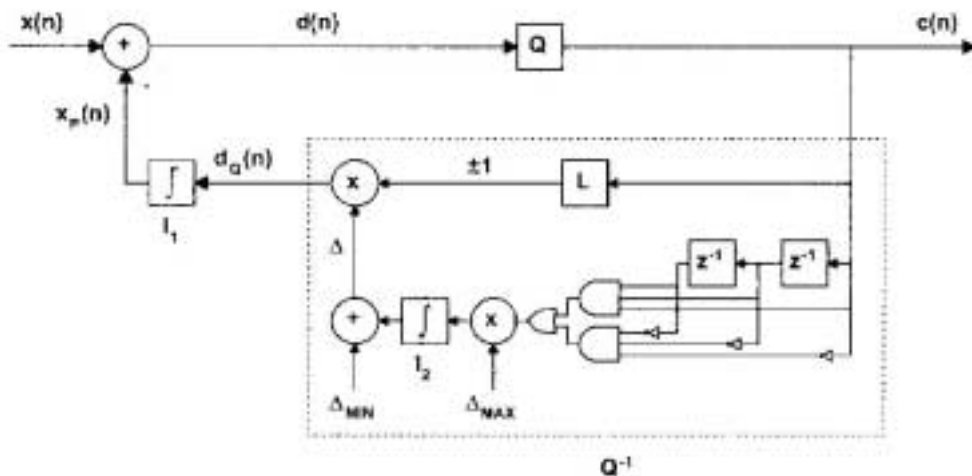
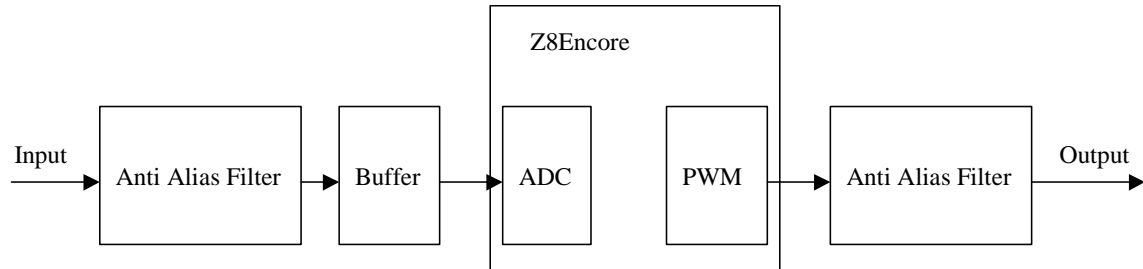


Figure 1: CVSD Encoder

Hardware

The hardware consists of the Anti Alias Filters and an impedance matching circuit for the ADC. The Z8Encore development kit is used to support the Z8Encore CPU.



Anti Alias Filter

Parameters:

Cutoff Frequency: 3.4Khz (-3dB)

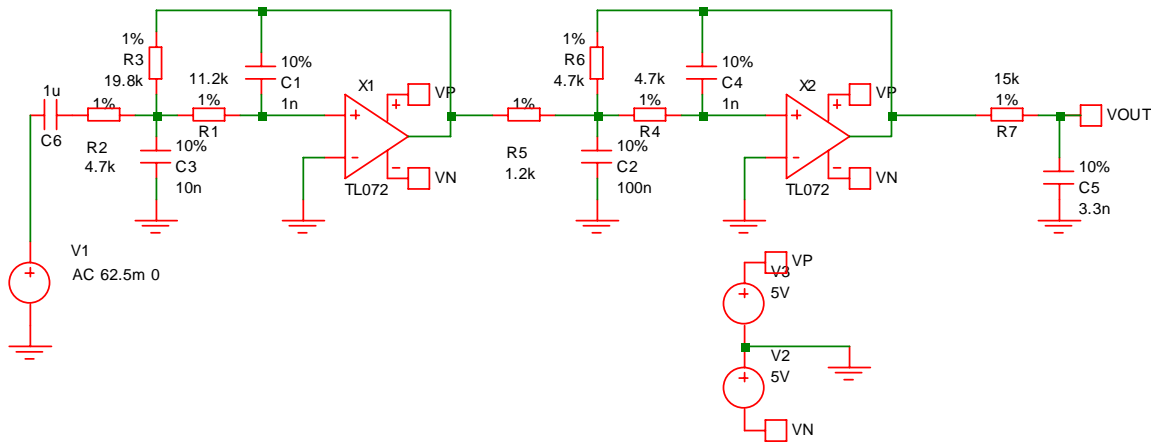
Filter Type: Butterworth (no ripple)

Filter Order: 5

Multiple Feedback Topology Filter

Gain = 16

TL072 and MCP6021 with single 5Volt Supply used for simulation



Software

The software is interrupt driven and is using the ADC and PWM to acquire and output the signal.

The PWM module is using Timer1. Because the PWM unit does not have buffer registers a small trick is used. A 9 bit PWM is used but the values are only 8 bit. The 8 bit values are centered on $\frac{1}{2}$ of the 9 bit maximum value (512). Therefore the PWM value will be

greater than 128 at any moment. Loading the value immediately when the reload interrupt occur (in less than 128 system clock cycles) will keep the PWM on the safe side.

```

PCADDR = 0x01;          // Port C data direction
PCCTL  &= 0xF8;        // PC1 = Timer1_out is output. PC0=out,PC2=out, Rest all
inputs.
PCADDR = 0x02;          // Port C alternate function,
PCCTL  |= 0x02;         // Enabled Timer1(PWM output)
PCADDR = 0x03;          // Port C Open Drain control
PCCTL  &= 0xFD;         // Timer1(PWM output) NOT open drain
PCADDR = 0x04;          // Port C HIGH DRIVE control
PCCTL  |= 0x02;         // Timer1(PWM output) High Drive Enable
PCADDR = 0x00;          // protect port control

/*****
* Initialize Timer 1 in continuous PWM mode with prescale = 1
* Initialize the PWM width 0x0100 - 1/2 scale.
* Initialize the PWM period to 27.77us / 36khz.
*****/
T1CTL  = 0x43;          // PWM mode, Prescale 1, TPOL=0, not enabled.
T1PWMH = 0x01;          // PWM High
T1PWL  = 0x00;          // PWM Low    1/2 scale

T1CPH  = 0x02;          // RELOAD Hi/ Low = 0x0200 with a prescaler of 1
T1CPL  = 0x00;          // results 36KHz PWM cycle at 18.432MHz sysclk

T1CTL  |= 0x80;          // Enable timer PWM

```

The ADC is used in continuous mode. A new conversion will end every 256-clock cycles.

```

PBADDR = 0x02;          // Port B alternate function,
PBCTL  |= 0x01;        // Enabled PB0=ANA0=ADC INPUT 0
PBADDR = 0x03;          // Port C OPEN DRAIN control
PBCTL  &= 0xFE;         // PB0=ANA0 NOT open drain
PBADDR = 0x04;          // Port C HIGH DRIVE control
PBCTL  &= 0xFE;         // PB0=ANA0 NOT High Drive
PBADDR = 0x00;

ADCCTL = 0xB0;          // CEN=1, 0, VRED=1, CONT=1, ANAIN=0000
// CEN will be 0 at first complete conversion result 5129+40 system clock cycles

```

while (ADCCTL & 0x80)//wait the end of the first A/D conversion

The PWM interrupt will occur 36000 times per second. Because the Sampling Frequency is only 18.000Hz only one of two interrupts will be used to change the PWM value.

Port C out 0 is used to output the Sample Frequency; Port C output 1 is the PWM output.

```

#pragma interrupt
void interrupt dac_isr(void)
{
    _aaa--;
}

```

```

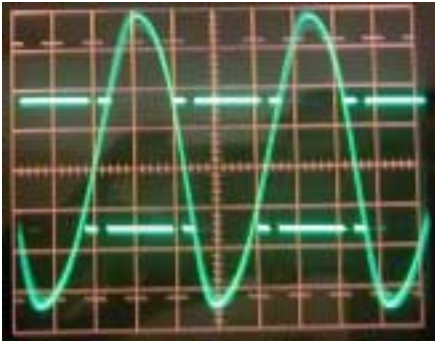
if (_aaa == 0)
    {
        if (DAC_READY == FALSE)
            {
                T1PWMH = _pwmhi; // Write first the high byte to avoid an early PWM
trigger
                T1PWML = _pwmlo;
                DAC_READY = TRUE;
            }
        _aaa = 2; // 36/2=18khz FSample
        PCOUT = PCOUT ^0x1; // toggle PC0
    }
}

```

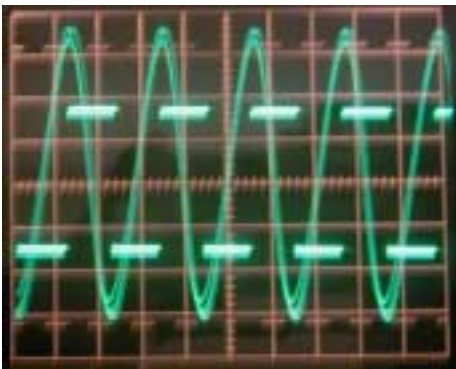
Finally the main loop is implementing the Delta vocoder. The value of the encoded bit can be found at Port C bit 2 for each coding cycle.

RESULTS

The results were obtained by applying a sine wave to the vocoder input and watching the output on a scope.



1125 Hz Sine Wave



2250 Hz Sine wave