Convolution example Guideline

This guideline provide the general tutorial about how to call a function to do convolution with our technical package. The example also transmits the result and display it in your host computer.

First example: Convolve when data buffer is full. (Not a real time convolution, no recommend to test)

INLINE function:

/*calculate the size of your array*/

```
#define ELEMENT_COUNT(X) (sizeof(X) / sizeof((X)[0]))
```

Software convolution functions:

```
/*Function for convolution.*/
void convolve(alt_16 Signal[/* SignalLen */], size_t SignalLen,
              alt_16 hfunction[/* KernelLen */], size_t hfunctionLength,
              alt_16 Result[/* SignalLen + KernelLen - 1 */])
{
  size_t n;
  for (n = 0; n < SignalLen + hfunctionLength - 1; n++){
        size_t kmin, kmax, k;
      Result[n] = 0;
      kmin = (n >= hfunctionLength - 1) ? n - (hfunctionLength - 1) : 0;
      kmax = (n < SignalLen - 1) ? n : SignalLen - 1;
      for (k = kmin; k <= kmax; k++){
          Result[n] += Signal[k] * hfunction[n - k];
      }
  }
}
```

This function takes 5 arguments:

1. Signal ring buffer

2. Ring buffer size

3. h function array

4. h function array size

5. Result buffer

Second Example: Convolve when a new data arrive (real time).

Software convolution functions:

```
/*Real time convolution: Do a convolution calculation when a new data comming in.*/
void convolve(int dataindex, float hfunction[]){
    int index;
    int count;
    int convResult = 0;
    index = dataindex;
    for(count = 0; count < BUFFERSIZE; count++){
       convResult += hfunction[(calBuffersize - count)] * leftChannelData[index %
BUFFERSIZE];
       index++;
    }
    convResultBuffer[convIndex] = convResult;
    convIndex = (convIndex + 1) % (CONVBUFFSIZE);
}
```

This function is called inside each “ready” ISR. Once every new sample is arrived, we need to do a convolution and give us a new convolution result. The size of your h(n) functions affect the running time of the convolution. In our package we set the h(n) function size to 32 which is the maximum size to work on 48k sampling rate (32k sampling rate will have the perfect performance in both left and right channel). Sorry for this limitation, if we make a larger h(n) function size the calculation will not be done when a new data is arrived.

Main function for testing: (For second example)

```
/*If a user request to send data back to the host computer*/
if(uartStartSendFlag){
       uart_SendConvBuff();
       uart_SendLeftBuff();
       uartStartSendFlag = 0;
       alt_irq_enable(leftready_id);
       alt_irq_enable(rightready_id);
}
```

The uartStartSendFlag will be TRUE if a user presses Key0 to trigger an interrupt which allow CPU to send data to the host computer. (Please see Key0 ISR). The function “uart_SendConvBuff()” will send the whole result buffer through RS232 to your host computer and reset the interrupt flag.

Convolution example:

In our package we have the following two examples to compare our convolution function with Matlab's inner conv(data,h) function. In this example we used an extra testing purpose data buffer to store and count 1000 data point. Once this testing buffer is full, you are ready to press Key0 to send both 1000 raw data and convolution result to Matlab. Please look into YOURISR.h file to understand both convolve function and leftReady ISR.

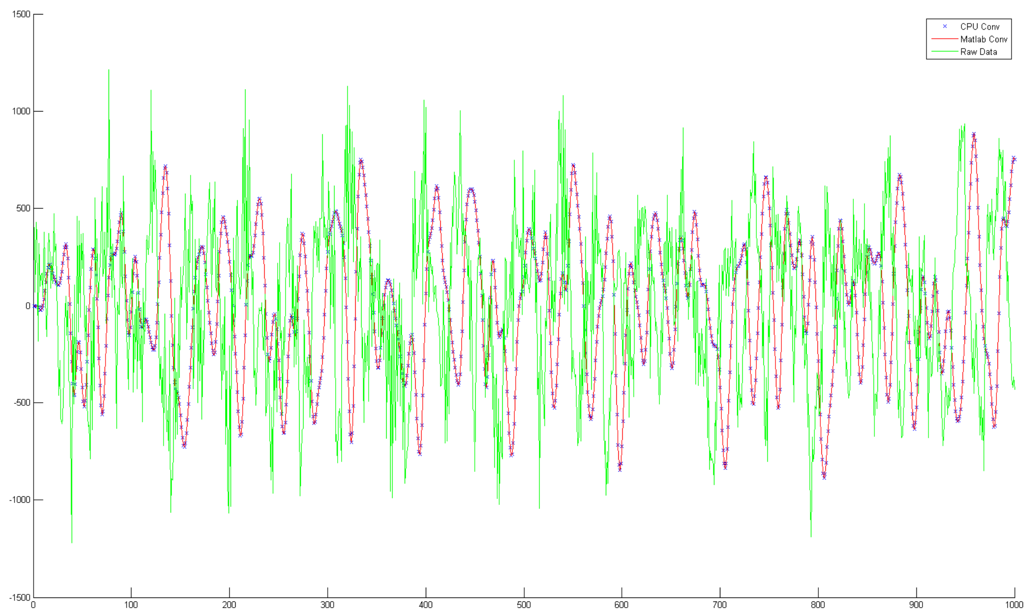
- h(n): A low pass filter and has size of 27.
- Data buffer has size of 27.
- Signal: Real music audio signal.
- Sampling rate: 32KHz
- IMPORTANT NOTE: Please know that the data type we collect is a signed 16 bit integer value, which means the data value has the range from:
  ||||||||||-32768 (-2^15) to 32768 (2^15) ||||||||||

The last bit represents the sign of the value (1->negative, 0->positive).

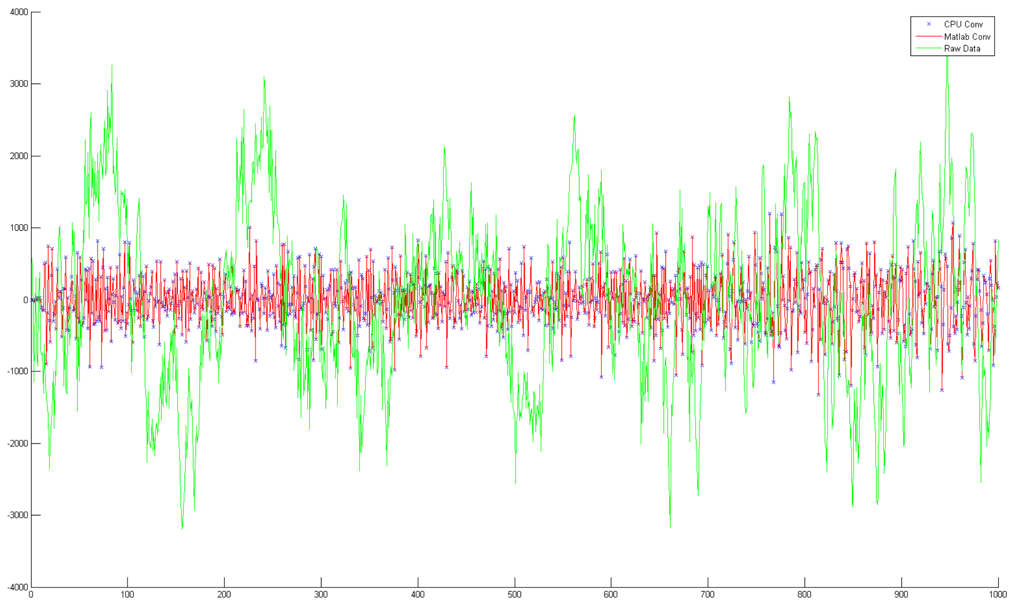
Please also realize that if the digital value of your signal is too large, the final result of your convolution may pass the above limit and the display will be messy.

For this example, the low-pass filter is generated by Matlab, we duplicated the filter to our C code and use it to do the convolution. In our matlab side, we send out 1000 music data and convolution result back to Matlab and compare the convolution result with Matlab's conv(data,h) function.

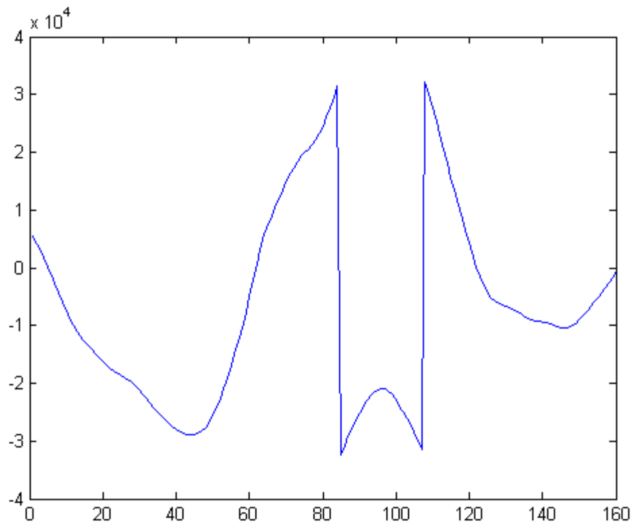
Matlab plot result:



Now we change to a high-pass filter and the result is:



PLEASE NOTE: A messy result may caused by overlimit:



The values between 82 to 108 are all greater than 32768 so the 16th bit becomes ‘1’ and your code treats them as a negative value (2’s complement).

Please follow the steps to transmit data and display the result:

Step 0: Choose which example you want to use and properly adjust the code.

Step 1: Properly configure and program the board with the package.

Step 2: Play a music and input the audio signal to AIC23 LINEIN.

Step 3: Input a headphone or speaker to LINEOUT and make sure you hear back your music.

Step 4: Open Matlab software and make sure you have the correct "COM" port setting.

Step 5: If there is no error so far, you can modify and run the "test.m" function and press Key0. The convolution result should be display on the screen.

```
*note: For this example, when you press key0 to send the result, data collection will
be disable for full data transmission. You will hear a very short pulse but no worry
it will automatically resume when finish sending.
```