Siren7
Sample Code
Programming Guide

V1.00.001

Publication Release Date: Aug. 2011

Support Chips:

ISD9160

Support Platforms:

Nuvoton

The information in this document is subject to change without notice.

The Nuvoton Technology Corp. shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

This documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from the Nuvoton Technology Corp.

Table of Contents

1. Introduction

This document introduces how to compression/decompression voice data with Siren7 library API.

1.1. Feature

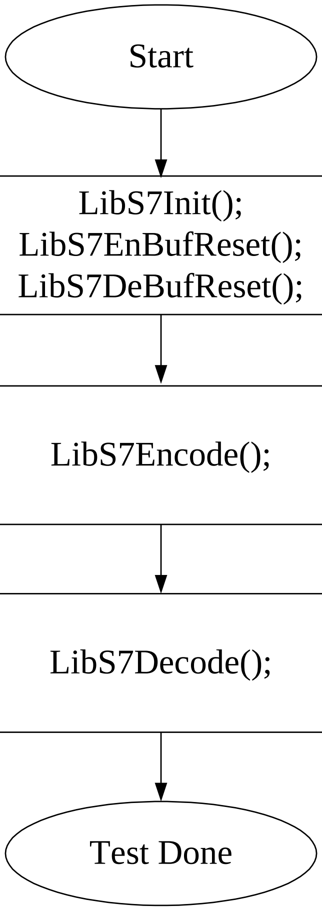
- Load a file which is pcm data format and compression as Siren7 format.
- Decompression Siren7 data as pcm data format then output to SPK.

1.2. Limitation

- Use Siren7 bandwidth 7K mode to do the compression.
- Support bit rate from 4k~32k bit per sample.
- The sample rate of test data is limited by input source file.

2. Calling Flow

2.1. Blocking APIs calling flow



2.2. API Usage Reference

- ITU-T Rec. G.722.1.pdf

3. Code Section – Smpl_LibSiren7EnDec.c

3.1. Main function

The main functions of Smpl_ LibSiren7EnDec.c are

- Configure system clock
- Configure GPIO pins to become UART interface pins
- Initial Siren7 parameters and reset buffer
- Initial PDMA channel and DPWM path
- Demo calling APIs
  - Compression pcm data as Siren7 format
  - Decompression Siren7 data to pcm format
  - Move pcm data to speaker output
- Test done

Configure system clock

The following codes are used to configure system clocks

```
     UNLOCKREG();
     SYSCLK->PWRCON.OSC49M_EN = 1;
     SYSCLK->CLKSEL0.HCLK_S = 0; /* Select HCLK source as 48MHz */
     SYSCLK->CLKDIV.HCLK_N  = 0;    /* Select no division         */
     SYSCLK->CLKSEL0.OSCFSel = 0;  /* 1= 32MHz, 0=48MHz */
```

System clock register is normally write-protected. Before system clock setting, it needs "UNLOCKREG()" to unlock the protected register. After system clock configured, we must call "LOCKREG()" to lock these protected register.

Configure GPIO pins to become UART interface pins

UART interface is shared with GPIO. Programmer should enable the GPIO pins as the UART interface pins. These codes are in the function Smpl_LibS7UartInit ().

```
     /* Set UART Pin */
     DrvGPIO_InitFunction(FUNC_UART0);
          /* UART Setting */
    sParam.u32BaudRate        = 115200;
    sParam.u8cDataBits          = DRVUART_DATABITS_8;
    sParam.u8cStopBits          = DRVUART_STOPBITS_1;
    sParam.u8cParity            = DRVUART_PARITY_NONE;
    sParam.u8cRxTriggerLevel= DRVUART_FIFO_1BYTES;

    DrvUART_Open(UART_PORT0,&sParam);
```

Initial Siren7 library

Users must initial the Siren7 library for resetting parameters and buffers. The initial procedures are doing in the “LibS7Init ()”. The following API is called in Smpl_LibS7APIinit()

```
     LibS7Init ();
     LibS7EnBufReset ();
     LibS7DeBufReset ();
```

Initial PDMA channel and DPWM path

Users must initial the PDMA for auto move the data from source address to destination address. And initial DPWM path for output audio data to speak. The initial procedures are doing in the “Smpl_LibS7PlaySoundInit ()”.

```
     Smpl_LibS7InitDPWM(DPWMSAMPLERATE);
     u32TotalPCMcount= ((uint32_t)&u32AudioDataEnd-(uint32_t)&u32AudioDataBegin)/2;
     boPCMplaying=TRUE;
     u32AudioSampleCount=0;
     u32PDMA1CallBackCount=0;

     u32BufferEmptyAddr= (uint32_t) &i16AudioBuffer[0][0];
     Smpl_LibS7CompEngine();
     u32BufferReadyAddr= (uint32_t) &i16AudioBuffer[0][0];
     Smpl_LibS7PDMA1forDPWM();

     u32BufferEmptyAddr= (uint32_t) &i16AudioBuffer[1][0];
     boBufferEmpty=TRUE;
```

Demo calling APIs

After initialize the functions, the device starts to move the first frame of pcm data. And the data has been compressed by calling the compression engine.

- Compression data
  Calling “LibS7Encode()” can compress the pcm data to Siren7.

```
     LibS7Encode(&sEnDeCtl, &sS7Enc_Ctx, (signed short *)u32BufferEmptyAddr,
s16Out_words);
```

- Decompression data
  Decompressing Siren7 data follow by compression had been done.

```
    LibS7Decode(&sEnDeCtl, &sS7Dec_Ctx, s16Out_words, (signed short
*)u32BufferEmptyAddr);
```

The pcm data is different with original one after through compression/decompression procedure.

- Output audio data to speaker
  Although the audio data passed through Siren7 compression/decompression steps and was different with original. It can be outputted as normal voice that human being can’t discover.

```
    STR_PDMA_T sPDMA;

    sPDMA.sSrcAddr.u32Addr   = u32BufferReadyAddr;
    sPDMA.sDestAddr.u32Addr  = (uint32_t)&DPWM->FIFO;
    sPDMA.u8Mode             = eDRVPDMA_MODE_MEM2APB;;
    sPDMA.u8TransWidth       = eDRVPDMA_WIDTH_16BITS;
    sPDMA.sSrcAddr.eAddrDirection = eDRVPDMA_DIRECTION_INCREMENTED;
    sPDMA.sDestAddr.eAddrDirection     = eDRVPDMA_DIRECTION_FIXED;
   sPDMA.i32ByteCnt = AUDIOBUFFERSIZE * 2;       //Full MIC buffer length (byte)
   DrvPDMA_Open(eDRVPDMA_CHANNEL_1, &sPDMA);

    // PDMA Setting
    DrvPDMA_SetCHForAPBDevice(
    eDRVPDMA_CHANNEL_1,
    eDRVPDMA_DPWM,
    eDRVPDMA_WRITE_APB
    );

    // Enable DPWM DMA
    DrvDPWM_EnablePDMA();
    // Enable INT
    DrvPDMA_EnableInt(eDRVPDMA_CHANNEL_1, eDRVPDMA_BLKD );
    // Install Callback function
    DrvPDMA_InstallCallBack(eDRVPDMA_CHANNEL_1, eDRVPDMA_BLKD,
                        (PFN_DRVPDMA_CALLBACK) Smpl_LibS7PDMA1_Callback );
    DrvPDMA_CHEnablelTransfer(eDRVPDMA_CHANNEL_1);

    boPDMA1Done=FALSE;
```

- Move data in circles
  The audio source data has been segmented in many pieces of frames. Programmer should move all of the frames to speaker that could hear the formerly voice after translation.

```
    while (boPCMplaying == TRUE)
```

```
     {
          if (boBufferEmpty==TRUE)
          {
               Smpl_LibS7CompEngine();
               boBufferEmpty=FALSE;
          }

          if ((boPDMA1Done==TRUE) && (boBufferEmpty==FALSE))
               Smpl_LibS7PDMA1forDPWM();

     }
```

4. Revision History

Version	Date	Description
V1.00.001	Aug. 30, 2011	● Created

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in equipment or systems intended for surgical implantation, atomic energy control instruments, aircraft or spacecraft instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for any other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications whereby failure could result or lead to personal injury, death or severe property or environmental damage.

Nuvoton customers using or selling these products for such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from their improper use or sales.