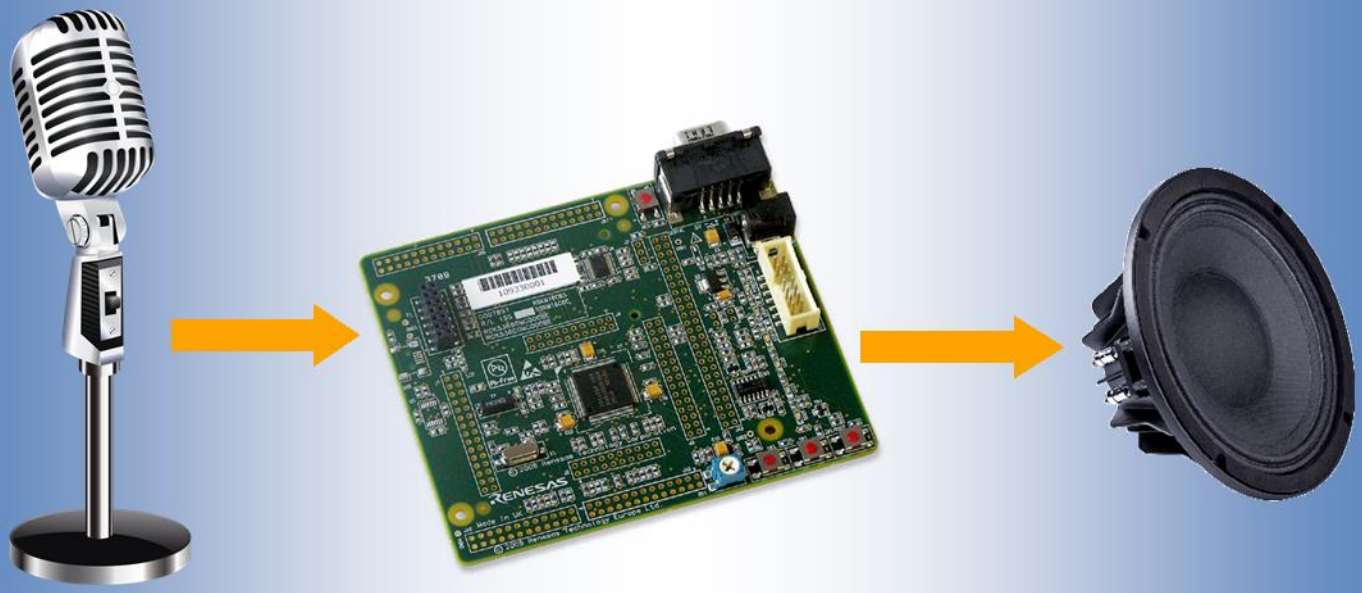
SILK encoder implementation on the RX210 microcontroller

Application Note



Sebastien Bouiller

January 2015

Summary

Illustration Table

Introduction

This application note will describe the different steps and the main points to perform the porting of the audio encoder SILK on the RX210 microcontroller, maintaining good audio quality and as less CPU load as possible. First of all, the SILK encoder will be introduced. Then the tests of the encoder on a PC will be described. Last of all, how to port the encoder on the RX210 microcontroller will be explained.

It should be noticed that following this application note is also valid for the porting of the SILK decoder. However, it won't describe anything about how to test and verify the user's application.

What is the SILK encoder?

This section quickly introduces the SILK codec, especially the input parameters.

SILK is a free and open source speech codec (encoder-decoder) for real-time voice communications. It was developed by Skype, standardized by the IETF, and is part of the universal audio codec Opus. The SILK encoder is based on linear prediction (LP) and is able to compress a speech audio file by more than 10, supporting different input parameters:

- Sampling rate: 8, 12, 16 or 24 kHz
- Packet rate: 20, 40, 60, 80 or 100ms
- Bitrate: 6-40 kbps
- Packet loss resilience: 0-100%
- Forward Error Correction (FEC)
- Complexity
- Discontinuous Transmission (DTX)

This flexibility makes this codec very scalable and adaptive, especially concerning network application through control of bitrate, packet rate, packet loss resilience and DTX. More information about the SILK encoder are provided in the website http://tools.ietf.org/id/draft-vos-silk-02.txt

Test and validation on PC

This section explains how to download the SILK's source code and the preliminary modification in the library which can be validated on PC, before the porting on the microcontroller.

As the SILK codec is fully free and open source, it is possible to download its source code online. The fixed point version of the library is available on this website: http://tools.ietf.org/id/draft-vos-silk-02.txt. Be sure to download the fixed point version of the library to reduce the execution time of your microcontroller application, especially for a RX210 target which doesn't have internal FPU.

One of the main points at this step is to define the input parameters' values with the constraint of maintaining good audio quality, reducing the CPU load. With this purpose, FEC and DTX are disabled and packet loss resilience is set to 0%. For the application described in this application note, and after having performed tests on the library with different input values for the parameters, setting the sampling and the packet rate to respectively 8 kHz and 20 ms is suitable. According to the SILK documentation, the corresponding bitrate to reach the best quality must be set to have 1.5 bit/sample. With a 8 kHz sampling rate, it means at least 12kbps.

The SILK library already provides a beginning of embedded-oriented option. It can be activated by defining the following macro:

```
#define EMBEDDED_OPT   1
```

This set the complexity level to 0 and disables the bandwidth transition filtering for mode switching, allowing CPU load reduction. Here is a summary of the different settings for the input parameters:

- Sampling rate: 8 kHz
- Packet rate: 20 ms
- Bitrate: >= 12kbps
- Packet loss resilience: 0%
- No FEC
- No DTX
- Complexity: 0

Now the input parameters are fixed, all conditional tests on their values are removed. For example the following code:

```
/* Check sampling frequency first, to avoid divide by zero later */

if( ( encControl->sampleRate !=  8000 ) && ( encControl->sampleRate
!= 12000 ) &&
        (  encControl->sampleRate  !=  16000  )  &&  (  encControl-
>sampleRate != 24000 ) ) {
        ret = SKP_SILK_ENC_FS_NOT_SUPPORTED;
        SKP_assert( 0 );
        return( ret );
    }
```

is now useless as the sampling rate will always be 8 kHz. Some macros like the MAX_FS_KHZ are modified also because, in this case, the only sampling rate available is 8 kHz.

The encoder structure is dynamically allocated to memory using the malloc function, which shall not be used in embedded programming because of its slowness and high risks of memory leakage. Thus, the function SKP_Silk_SDK_Get_Encoder_Size shall be removed because it does not make any more sense and the following functions: SKP_Silk_SDK_QueryEncoder, SKP_Silk_SDK_InitEncoder, SKP_Silk_SDK_Encode shall be modified, especially concerning their prototype. In the main function, instead of using malloc, the encControl structure must be declared directly.

The library often uses macros which calls several functions provided within the standard stdlib C library which are memset, memmove and memcpy.

```
/* Useful Macros that can be adjusted to other platforms */
#define SKP_memcpy(a, b, c)                 memcpy((a), (b), (c))     /* Dest, Src, ByteCount */
#define SKP_memset(a, b, c)                 memset((a), (b), (c))     /* Dest, value, ByteCount */
#define SKP_memmove(a, b, c)                memmove((a), (b), (c))     /* Dest, Src, ByteCount */
```

Figure 1: stdlib functions definition

Including such a large library in an embedded application is not feasible, especially considering the fact that only three functions of this library are used by the SILK algorithm. So, those functions shall be re-written in the SKP_Silk_SigProc_FIX.h file as inlined functions in order to keep the same CPU load as before.

The float.h library is included in the SKP_Silk_typedef.h file as a remaining of the original floating point library. It can be removed without any issue, including the following conditional compiling instructions:

```
#if SKP_USE_DOUBLE_PRECISION_FLOATS

# define SKP_float      double

# define SKP_float_MAX DBL_MAX

#else

# define SKP_float      float

# define SKP_float_MAX FLT_MAX

#endif
```

Porting on the RX210

This section will briefly introduce the RX210 microcontroller, and will explain what shall be done while porting the SILK's algorithm on this target.

The RX210 microcontroller has a 32 bits CISC Harvard CPU without a floating point unit. It has up to 1 Mbytes code Flash memory and 96 kbytes on-chip SRAM. It can also perform DSP instructions, which are useful for the SILK user's application.

The E2Studio project for the RX210 can be created now preliminary modifications were done in the SILK library. Some special operations needs to be done while creating the project to be able to efficiently perform the porting. The different operations which will be shown from now were made for the GNURXV14.20 compiler with the RX210 Starter Kit, and should be different depending on the compiler and/or the board used.

First, while creating the project, the red surrounded options should be checked:

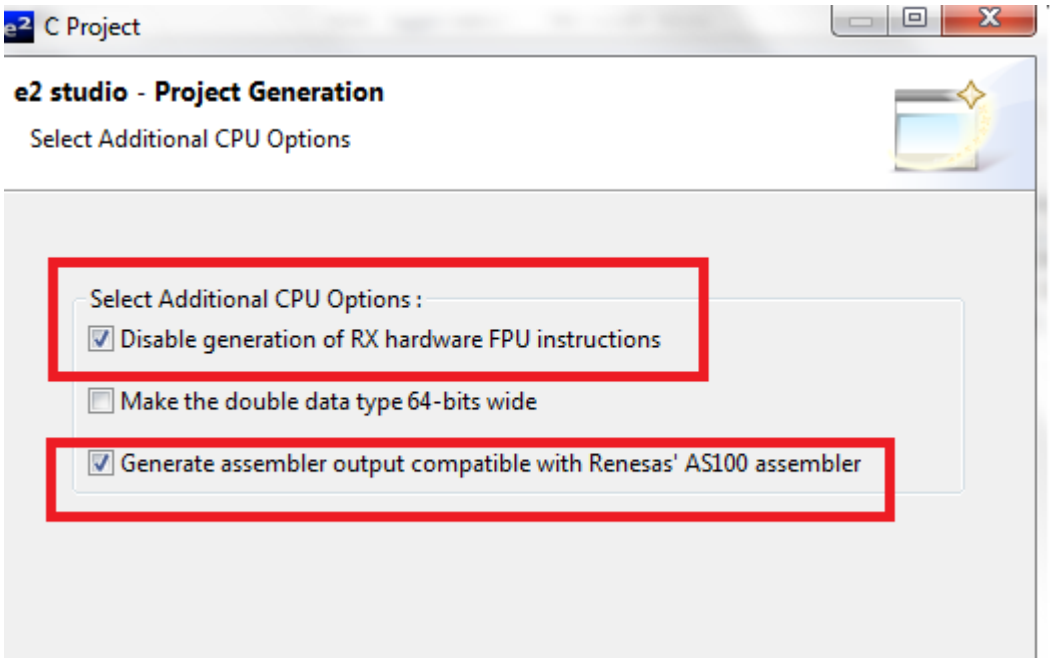


Figure 2: Options to check while creating the project

Then, in the project properties -> Run/Debug Settings, the hardware debug configuration should be edited as follows:

- In the Connection Settings tab, if an EXTAL is used its value should be checked depending on the board crystal oscillator.

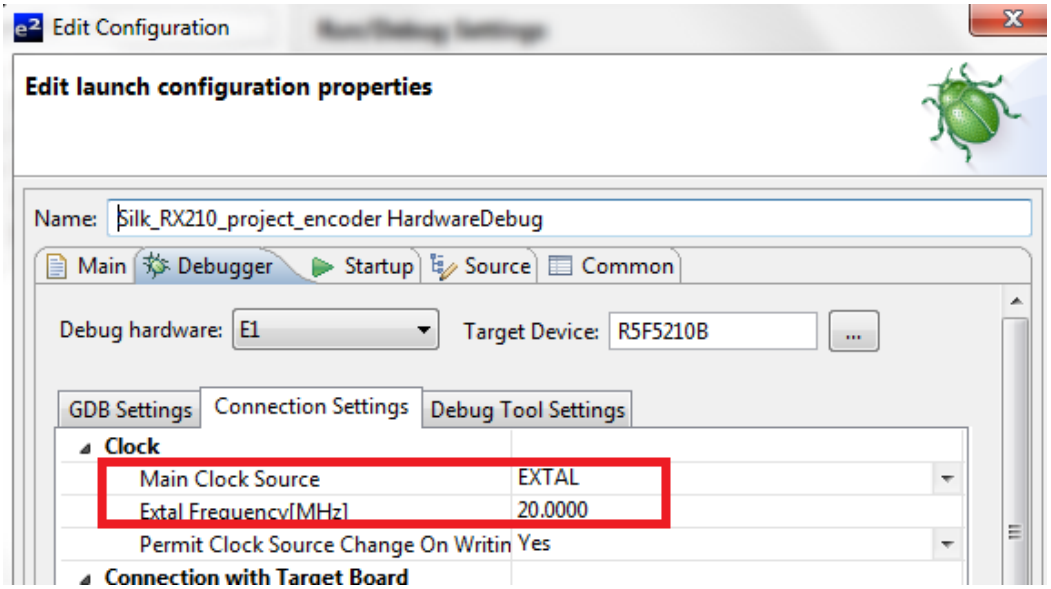


Figure 3: checking of the external clock

- In the Debug Tool Settings tab, the Work RAM start address should be set to at least 0x17000 in order to have as much SRAM available as possible.

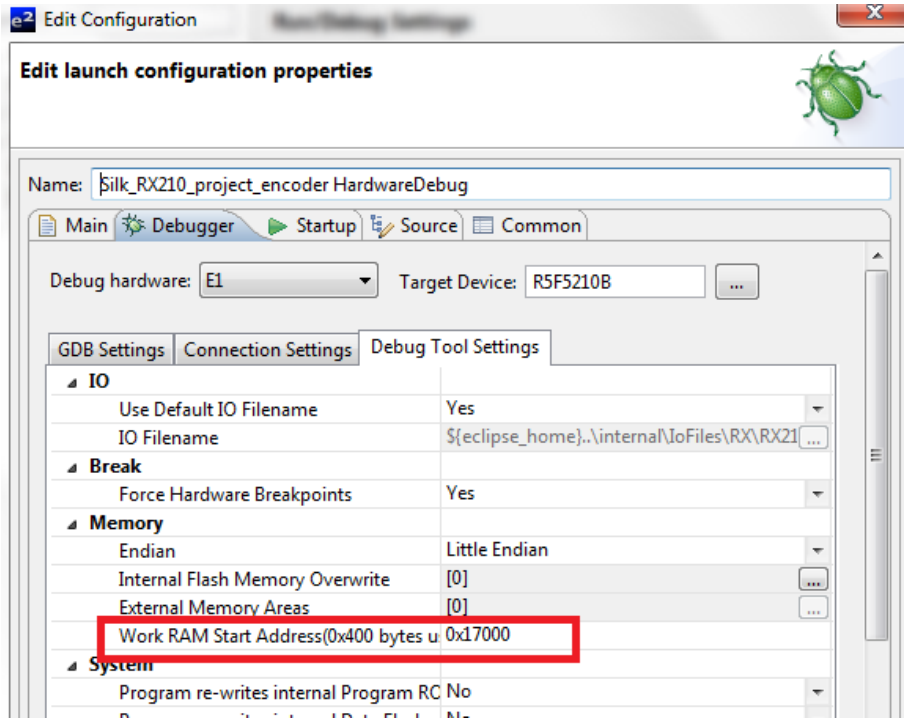


Figure 4: RAM start address

In the SILK library, some files and tables are dedicated for the decoder only, so that they are not useful for the porting of the encoder (and vice-versa). Thus, those files should not be included in the project.

The SILK library was originally designed to operate on audio files, especially pcm and wav files. For an embedded application, files are replaced by buffer and the fprintf, fread etc… functions can be removed. The buffer's size can be computed by multiplying the input sampling rate by the input packet rate (with 16 bits input samples and 8 bits output samples). For example, with an 8 kHz input sampling rate and a 20 ms input packet rate, the input buffer's size will be 160 words of 16 bits and the output buffer's size will be up to 160 words of 8 bits.

In SKP_Silk_typedef.h, the SKP_int_ptr_size should be redefined to match the RX210 architecture, and the SKP_INLINE macro should be redefined also to match the compiler's specifications.

By default, the stack size (256 bytes) is insufficient to make the application run because of the important size of the structures and buffers locally declared in the SILK functions. This makes the stack pointer sometimes points in the code Flash, which leads to unpredictable behavior from program crashing to infinite loop. It's possible to see how much each function consumes stack and which register is used as the stack pointer by having a look in the assembly files. Here is an example with GNURX compiler:

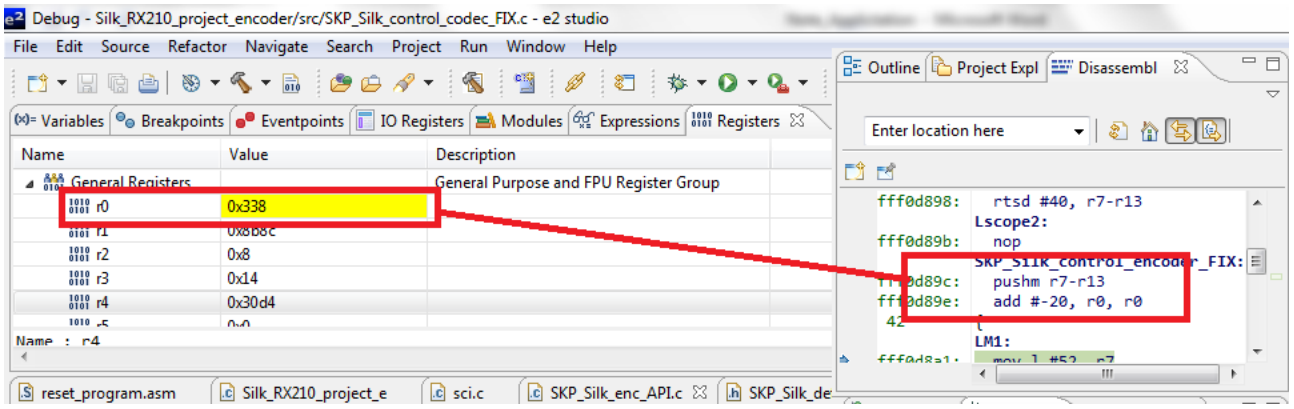


Figure 5: General register used as stack pointer

There are two complementary ways to solve this issue. First of all, the structures and buffers are declared in global in order to consume SRAM and not stack. Second of all, the size of the stack is increased as follows in the project properties window:

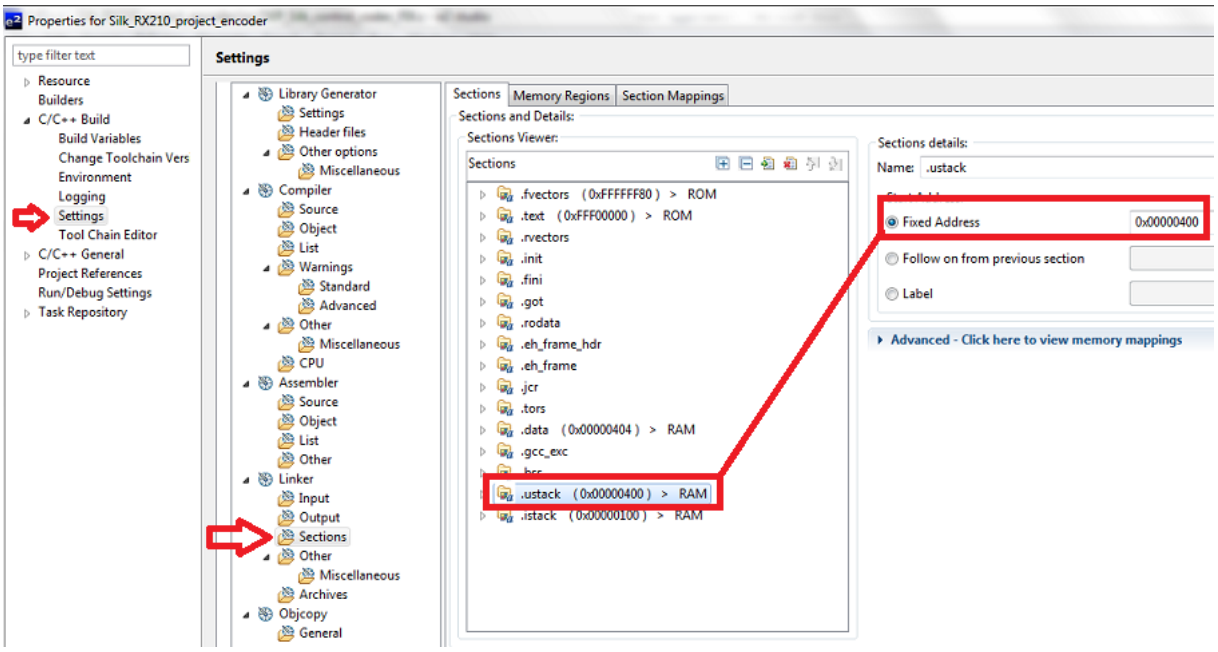


Figure 6: increasing the stack size

The data section start address must be also modified in order to avoid conflicts between the stack and the remaining SRAM.

To be sure that there's no more stack issues, the stack pointer's value must be less than or equal to the size of the stack during all the application execution.

Conclusion

The free audio codec SILK is currently the best one for speech applications, providing a good audio quality with a high compression level. Vocal synthesis and intercom applications can be performed by using this codec, for example the SKYPE software.