Audio Workshop V1.0 User's Guide

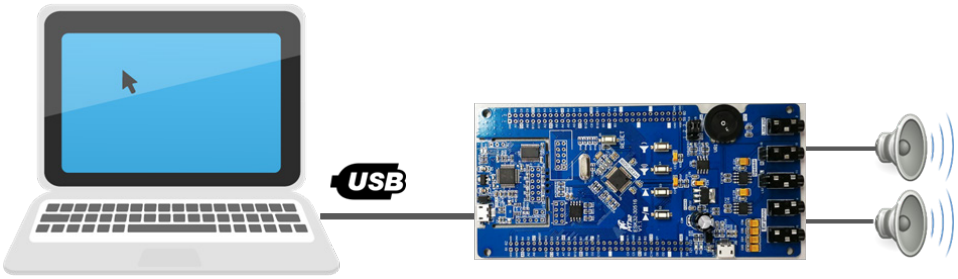
Revision: V1.20 Date: August 10, 2022

www.holtek.com

Table of Contents

1. Development Environment

1.1 Overall Environment



I. Audio Workshop

II. Keil™ MDK-ARM

Development Board

Speaker

1.2 Software

This includes the Audio Workshop and the Keil™ MDK-ARM µVision5.

1.2.1 Audio Workshop

- The Audio Workshop contains multiple functions, such as MCU selection, external SPI Flash Memory capacity selection, loading and editing MIDI/Sound WAV/Effect WAV files, sentence editing (combined by sound and effects), audio output configuration and MIDI interface configuration.
- MCU program compiling, SPI Flash tone data compiling and MCU/SPI Flash data downloading function.
- The tone editing functions contain tone library (SF2) loading, tone parameter editing, and tone replacing.

1.2.2 Keil™ MDK-ARM µVision5

- The Keil™ MDK-ARM µVision5 is used to edit and view the source code which can be downloaded to the development board.
- The Audio Workshop is required with the Keil™ MDK-ARM µVision5. Ensure that the Keil™ MDK-ARM µVision5 and Keil™ HT32 Support Packages have been installed before using the Audio Workshop.

1.3 Hardware

The Development Board is used together with the Audio Workshop for development.

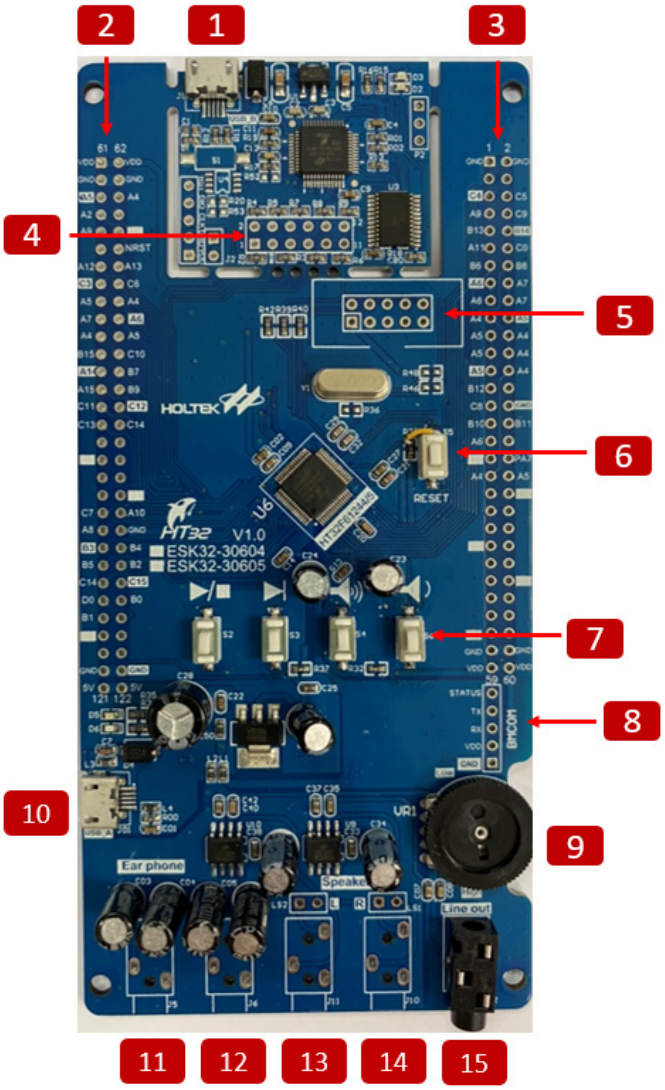
Note: The HT32F0006 project uses the ESK32-30617 development board for emulation development. Contact the Holtek technical department for actual HT32F0006 application development.

- The development board ESK32-30615 is for the HT32F61355
  - ♦ MCU with 32Mbits Data Flash ROM for data storage.
- The development board ESK32-30616 is for the HT32F61356
  - ♦ MCU with 64Mbits Data Flash ROM for data storage.
- The development board ESK32-30617 is for the HT32F61357/HT32F0006
  - ♦ MCU with 128Mbits Data Flash ROM for data storage.

- The development board ESK32-30605 is for the HT32F61244/ HT32F61245
  - ♦ MCU with 16/32Mbits Data Flash ROM for data storage.

1.3.1 Development Board Introduction

ESK32-30605



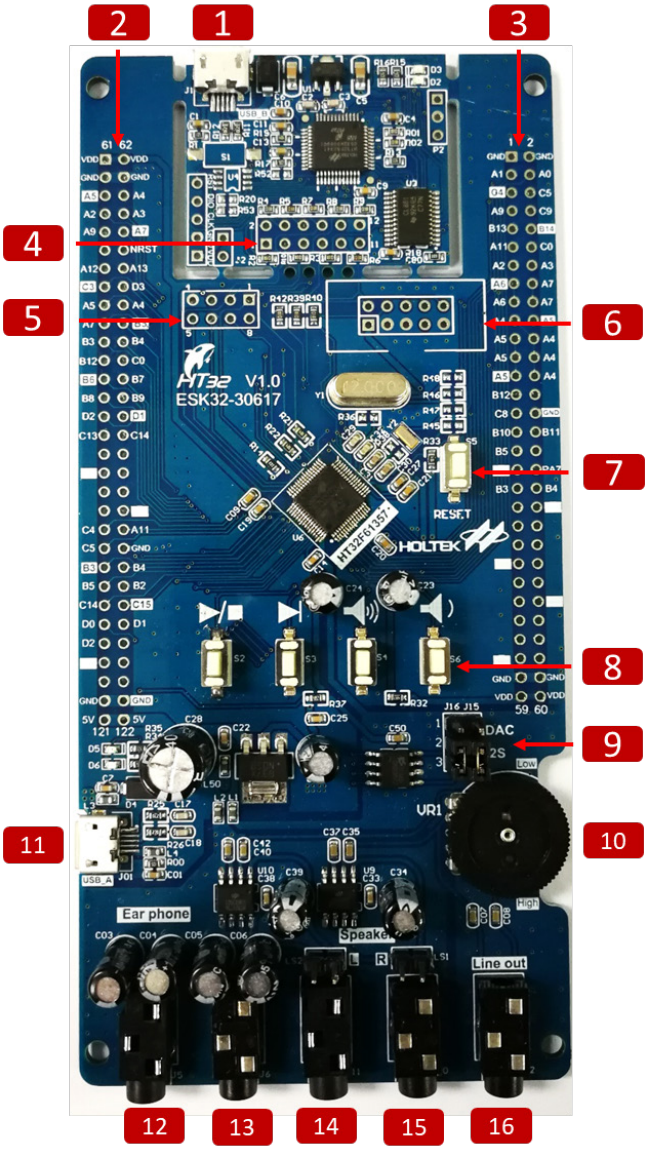
1. USB_B: e-Link32 Lite - for downloading program/data, emulating
2. GPIO/Expansion connector
3. GPIO/Expansion connector
4. Download connector
5. SWD-10P connector
6. Reset key
7. Functional keys:
   S2: Play/Stop
   S3: Next

S4: Sound +

S6: Sound -

8. UART connector
9. Volume adjustment: The volume increases when turned clockwise and decreases when turned counterclockwise
10. USB_A: HT32F6124x USB power supply
11. Earphone jack
12. Earphone jack
13. Left channel speaker jack
14. Right channel speaker jack
15. Line out jack

ESK32-30617 (it has the same configuration as ESK32-30615/ESK32-30616)



1. USB_B: e-Link32 Lite - for downloading program/data, emulating
2. GPIO/Expansion connector
3. GPIO/Expansion connector
4. Download connector
5. QSPI Flash connector
6. SWD-10P connector
7. Reset key
8. Functional keys:

   S2: Play/Stop

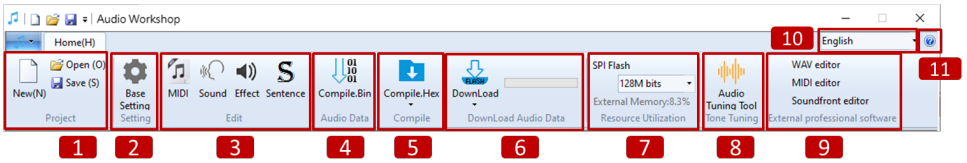
   S3: Next

   S4: Sound +

   S6: Sound -
9. DAC/I2S DAC jumper
10. Volume adjustment: The volume increases when turned clockwise and decreases when turned counterclockwise
11. USB_A: HT32F6135x USB interface
12. Ear phone jack
13. Ear phone jack
14. Left channel speaker jack
15. Right channel speaker jack
16. Line out jack

2. Audio Workshop Main Interface

2.1 Software Functional Introduction



1. New project/Open project/Save project
2. Base Setting: set the MIDI related functional parameters
3. Edit function (MIDI/Sound/Effect/Sentence)
4. Compile Audio Data (.bin)
5. Compile program files (.hex)
6. Download function:

   Download project program file (.hex) with audio data (.bin)
7. SPI Flash Memory capacity:

   Display the capacity and usage of SPI Flash memory
8. Enable the tone adjustment function

9. Quickly open external software menu:

   WAV editor: enable Audacity® by default

   MIDI editor: enable Cakewalk by BandLab© by default

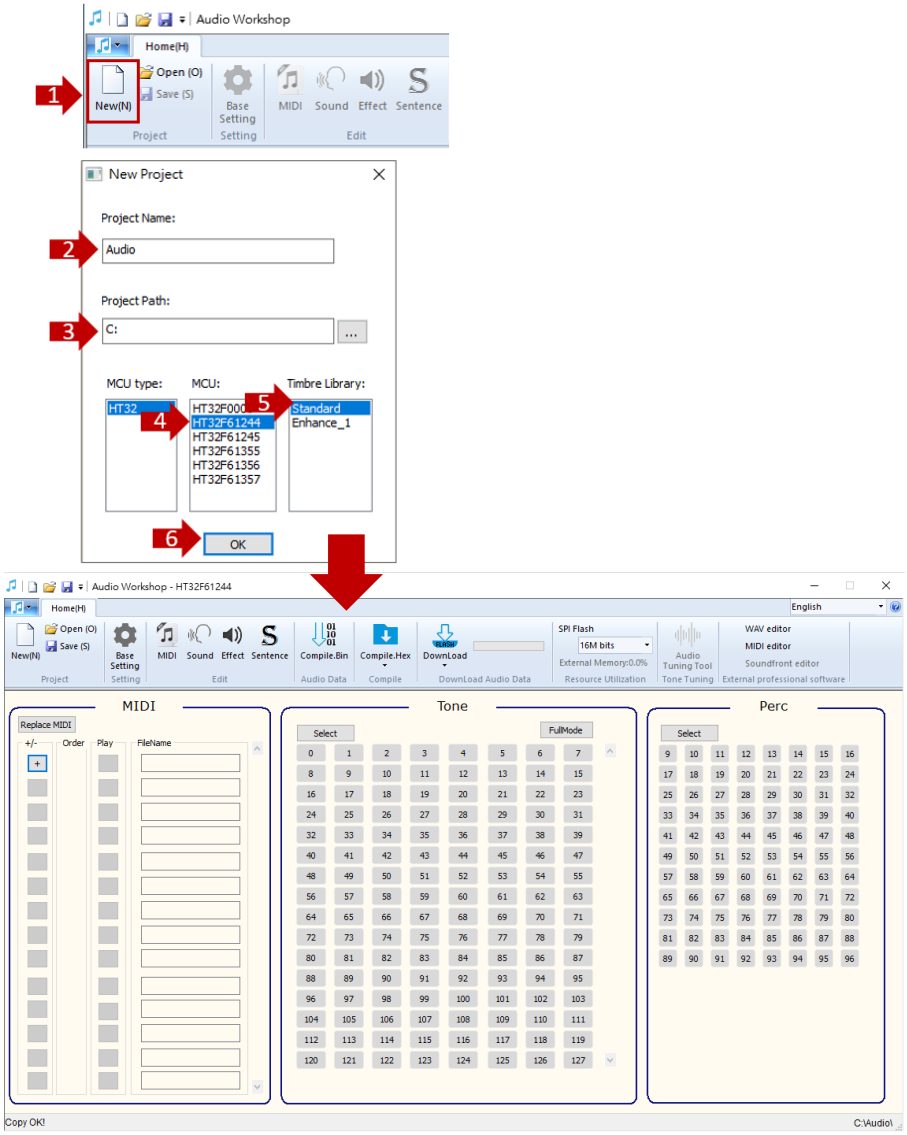
   Soundfront editor: enable Polyphone© by default

10. Interface language switching: supports three language options which are English, Simplified Chinese and Traditional Chinese

11. "About": Display software version information

3. New Project

3.1 New Project

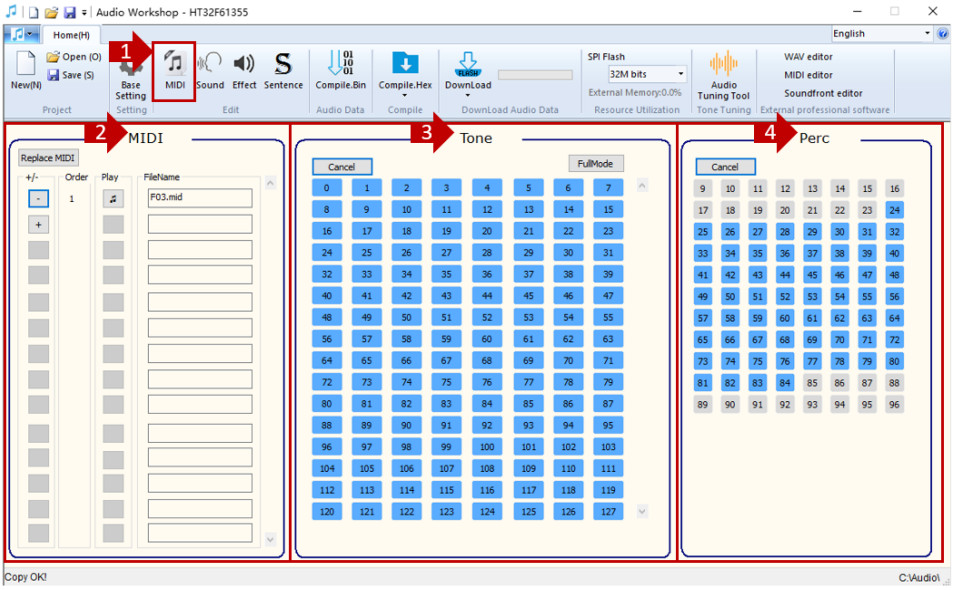


- Step 1: Click "New".
- Step 2: Set the project name.

- Step 3: Set the project path.
- Step 4: MCU selection.
- Step 5: Select the tone library.
- Step 6: Click "OK" and enter the audio main interface.

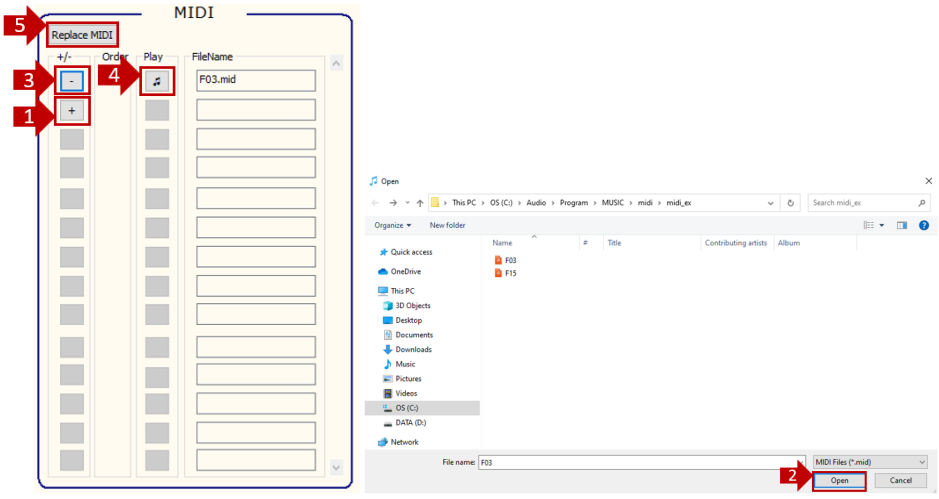
3.2 MIDI Configuration

3.2.1 Operating Process



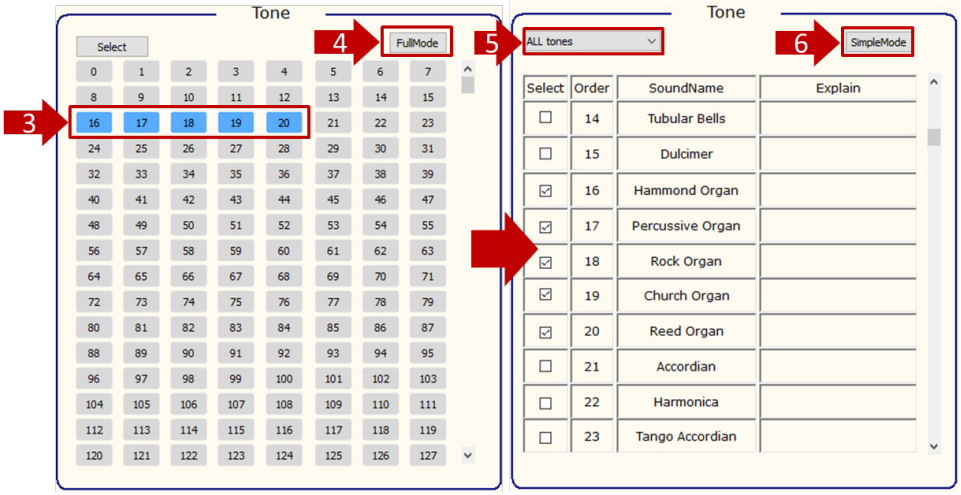
- Step 1: Click "MIDI".
- Step 2: Load/edit MIDI.
- Step 3: Set the tone.
- Step 4: Set the percussion.

3.2.2 Functional Introduction

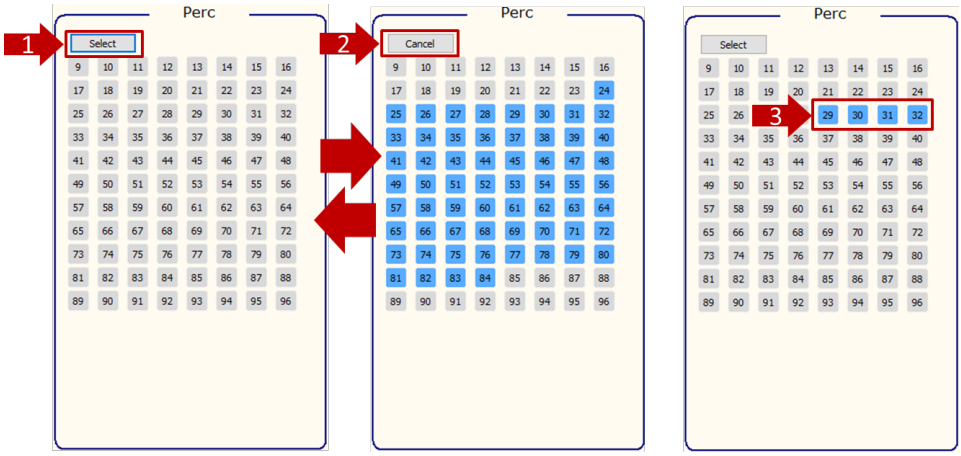


- Step 1: Click "+" to load MIDI.
- Step 2: After selecting the desired MIDI file, click the "Open" icon, then the MIDI will be loaded into the project.
- Step 3: Click "-" to delete MIDI.
- Step 4: Click the icon to play MIDI through the computer.
- Step 5: Click "Replace MIDI" to replace the MIDI file in the project.

- Step 1: Click "Select" to allow all tones (0~127) to be selected as shown on the right of the figure above.
- Step 2: Click "Cancel" to cancel all the selected tones as shown on the left of the figure above.



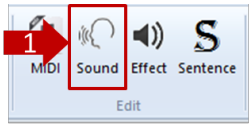
- Step 3: Click the tone number to select and cancel.
  For example: select the tone NO.16/17/18/19/20.
- Step 4: Click "FullMode" to display the full sound name corresponding to the tone as shown on the right of the figure above.
- Step 5: The selectable instrument types are displayed by their type.
- Step 6: Switch to the "SimpleMode" interface as shown on the left of the figure above.

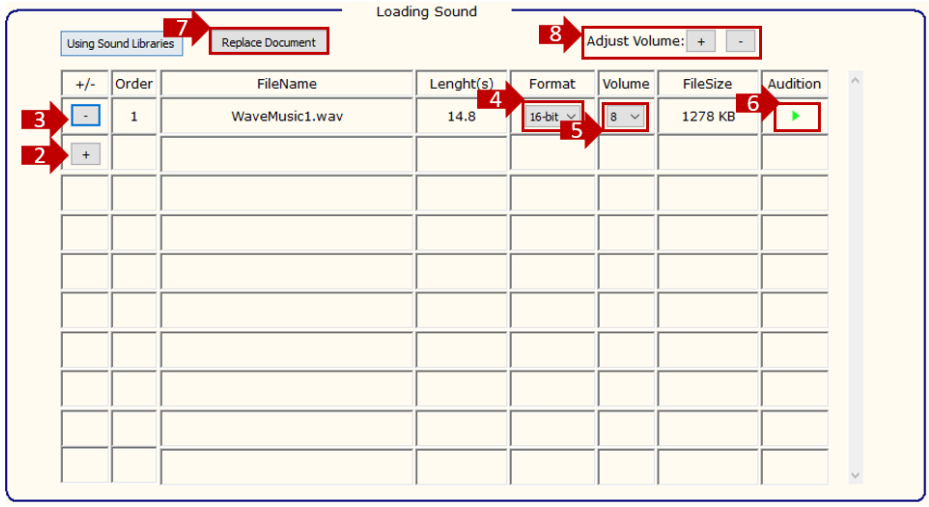


- Step 1: Click "Select" to select all percussions (24~84).
- Step 2: Click "Cancel" to cancel all the selected percussions.
- Step 3: Click percussion number to select and cancel.
  For example: Select the percussion No.29/30/31/32.

3.3 Sound Configuration

3.3.1 Functional Introduction



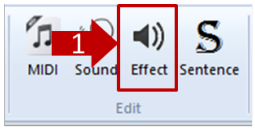


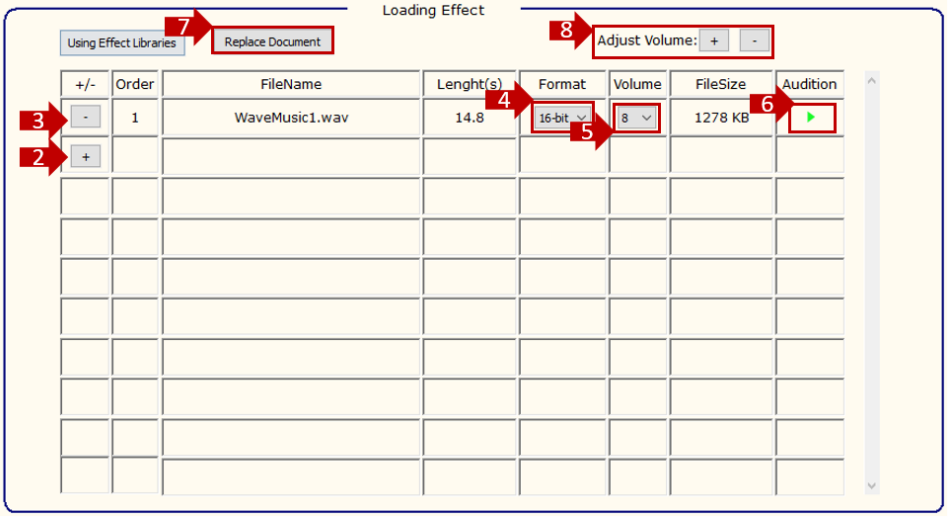
- Step 1: Click "Sound".
- Step 2: Click "+" to load the WAV file into the project.
- Step 3: Click "-" to delete the WAV file.
- Step 4: Set the compression format (PCM 8-bit/12-bit/16-bit).
- Step 5: Adjust the volume (1~15).
- Step 6: Click the icon to play the voice WAV through the computer.

- Step 7: Click "Replace Document" to replace the WAV file in the project.
- Step 8: Click "+" or "-" to adjust the volume of all WAV files.

3.4 Sound Effect Configuration

3.4.1 Functional Introduction

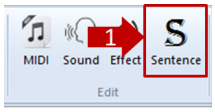


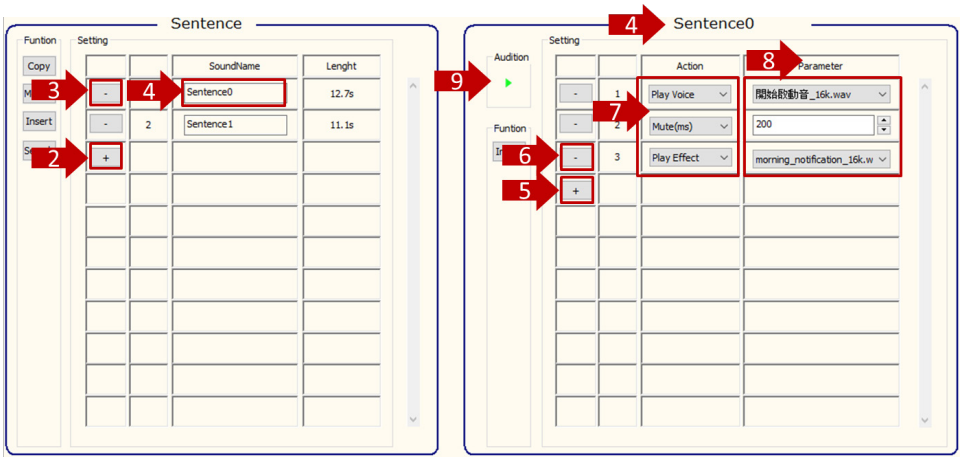


- Step 1: Click "Effect".
- Step 2: Click "+" to load the WAV file into the project.
- Step 3: Click "-" to delete the WAV file.
- Step 4: Set the compression format (PCM 8-bit/12-bit/16-bit).
- Step 5: Adjust the volume (1~15).
- Step 6: Click the icon to play the voice WAV through the computer.
- Step 7: Click "Replace Document" to replace the WAV file in the project.
- Step 8: Click "+" or "-" to adjust the volume of all WAV files.

3.5 Sentence Configuration

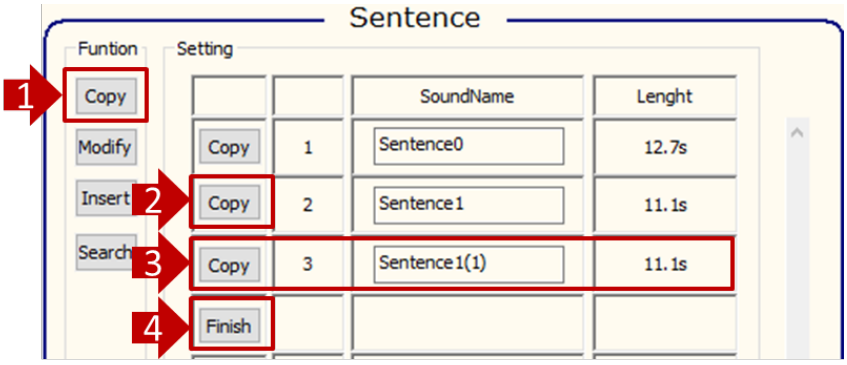
3.5.1 Functional Introduction 1



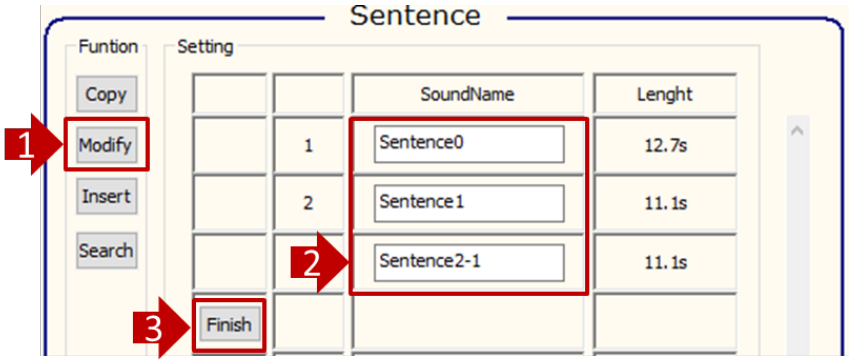


- Step 1: Click "Sentence".
- Step 2: Click "+" to add a new sentence.
- Step 3: Click "-" to delete the sentence.
- Step 4: Select the Sentence0 to edit the playing contents of Sentence0.
- Step 5: Add an New action.
- Step 6: Delete an action.
- Step 7: Edit the action and select "Play Voice/Play Effect/Mute".
- Step 8: Set the voice, effect sound or mute time.
- Step 9: Click the icon to play the sentence sound through the computer.

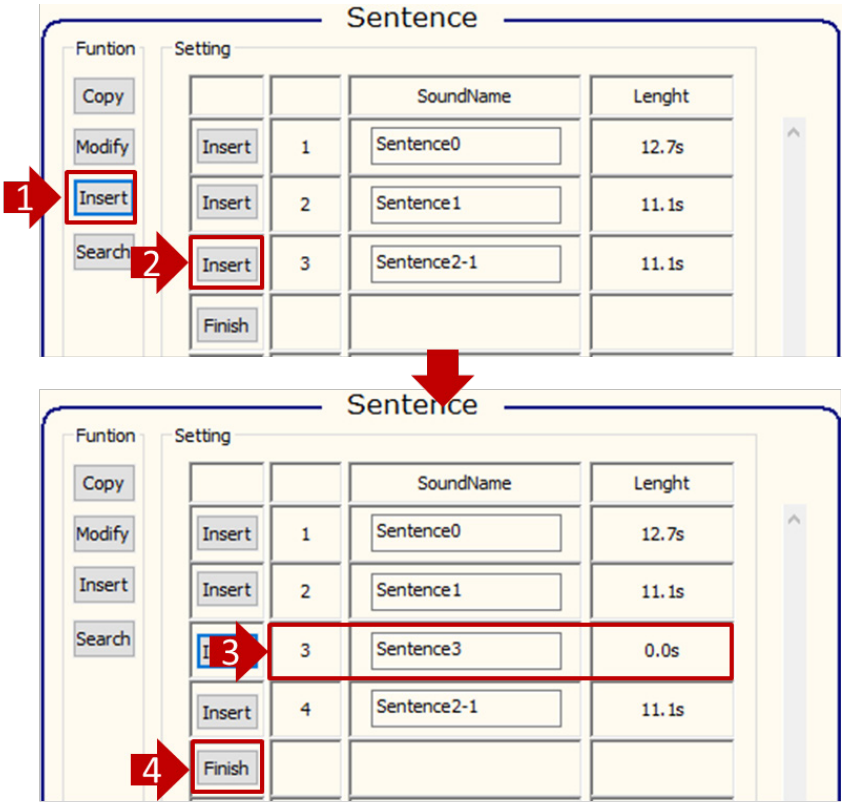
3.5.2 Functional Introduction 2



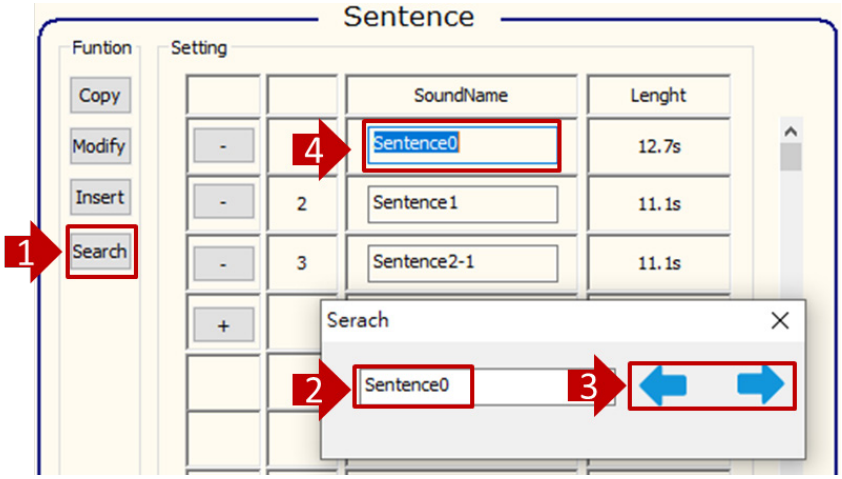
- Step 1: Copy sentence.
- Step 2: Click the button to copy Sentence1.
- Step 3: Automatically insert the copied Sentence (Sentence1(1)).
- Step 4: Click "Finish" to finish editing.



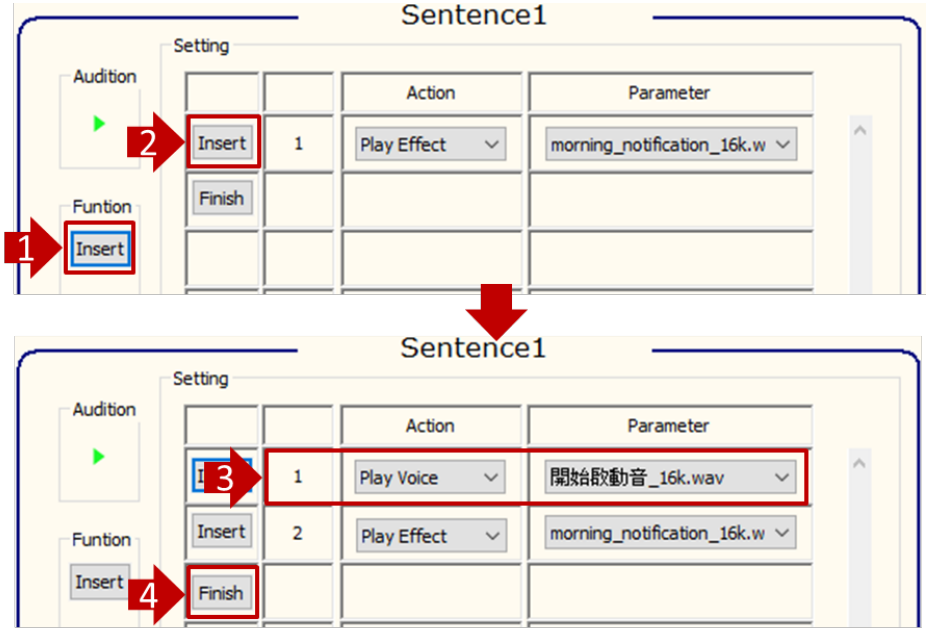
- Step 1: Modify the sentence name.
- Step 2: Modify the sentence name.
- Step 3: Click "Finish" to finish editing.



- Step 1: Insert sentence.
- Step 2: Click the button to insert the sentence.
- Step 3: Automatically insert an empty sentence (Sentence3).
- Step 4: Click "Finish" to finish editing.

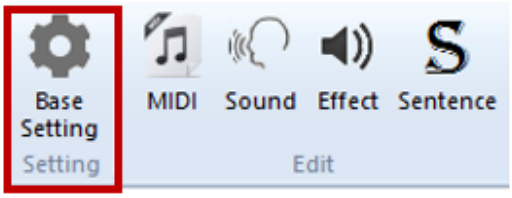


- Step 1: Search sentence.
- Step 2: Enter the sentence name.
- Step 3: Click to search for the Sentence0.
- Step 4: Find the Sentence0.

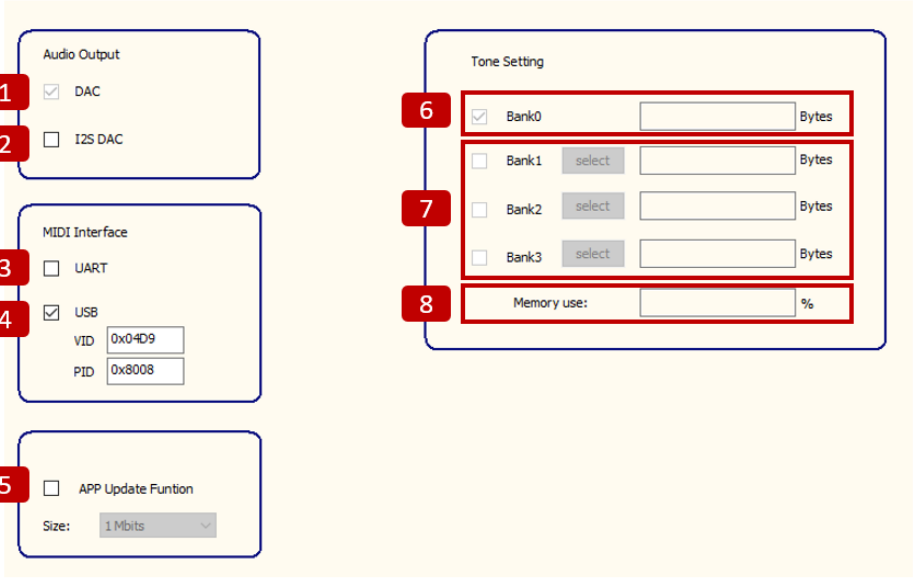


- Step 1: Insert sentence action.
- Step 2: Click the button to insert a sentence action.
- Step 3: Automatically insert a "Play Voice" action.
- Step 4: Click "Finish" to finish editing.

3.6 Base Settings



When the project uses the HT32F0006/HT32F61355/HT32F61356/HT32F61357 MCU, the basic setting page is as follows:



1. DAC: The default function.
2. Select "I2S DAC" to enable I2S DAC function.

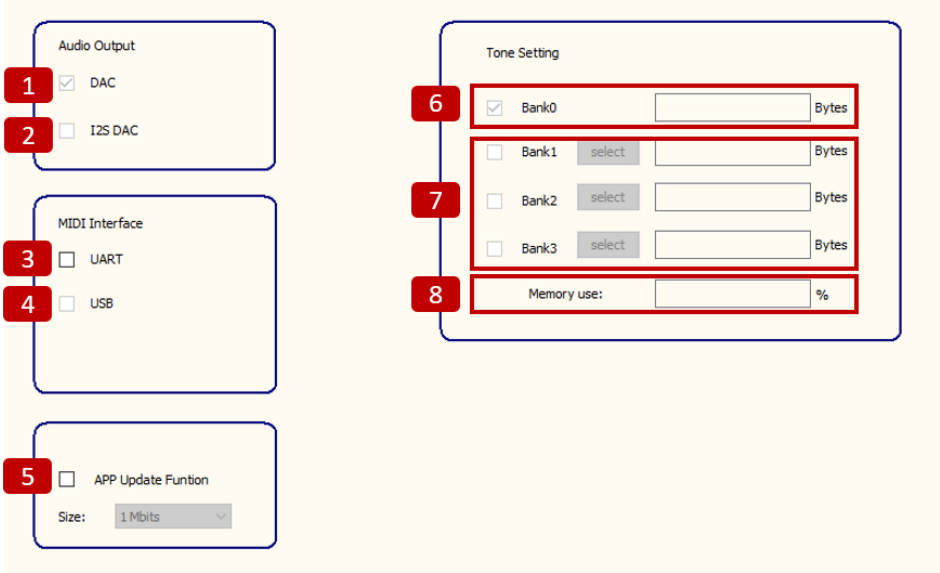
   The jumper on the development board can be switched to I2S to play through the I2S DAC.
3. UART: Select "UART" to receive MIDI information via UART. Cancel it to disable this function.
4. USB: Select "USB" to receive MIDI information via USB. Cancel it to disable this function.

   Provide the VID and PID for user modification.
5. APP Update Function: Select it to enable this function and cancel it to disable the function.

   Size: Set the size of the APP's updatable user MIDI music capacity.
6. Tone setting: select tone in Bank0 by default.
7. Bank1-3: Select it to add tone data. The tone source is determined by the __R_Bank value in the program file.

   Note: The Audio data (.bin) must first be compiled to enable this function. Refer to Section 3.7 for the detailed steps for Audio data compilation.
8. Memory use: Show how much Data Flash ROM has been used.

When the project uses the HT32F61244/HT32F61245 MCU, the basic setting page is as follows:

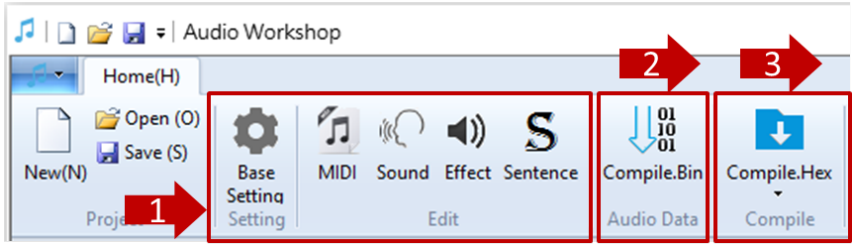


1. DAC: The default function.
2. I2S DAC: Not selected by default.
3. UART: Select "UART" to receive MIDI information via UART. Cancel it to disable this function.
4. USB: Not selected by default.
5. APP Update Function: Select it to enable this function and cancel it to disable the function.

   Size: Set the size of the APP's updatable user MIDI music capacity.
6. Tone setting: select tone in Bank0 by default.
7. Bank1-3: Select it to add tone data. The tone source is determined by the __R_Bank value in the program file.

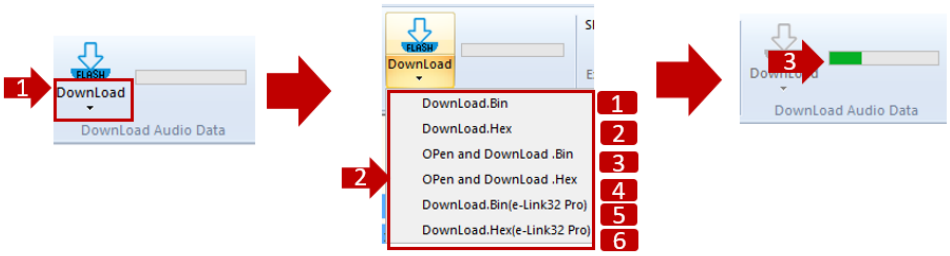
   Note: The Audio data (.bin) must first be compiled to enable this function. Refer to Section 3.7 for the detailed steps for Audio data compilation.
8. Memory use: Show how much Data Flash ROM has been used.

3.7 Compile the program and Audio data



- Step 1: Ensure that the "Base Setting/MIDI/Sound/Effect/Sentence" configurations have all been completed.
- Step 2: Click the icon to compile Audio data (.bin).
- Step 3: Click the icon to compile the program file (.hex) and the Keil project.

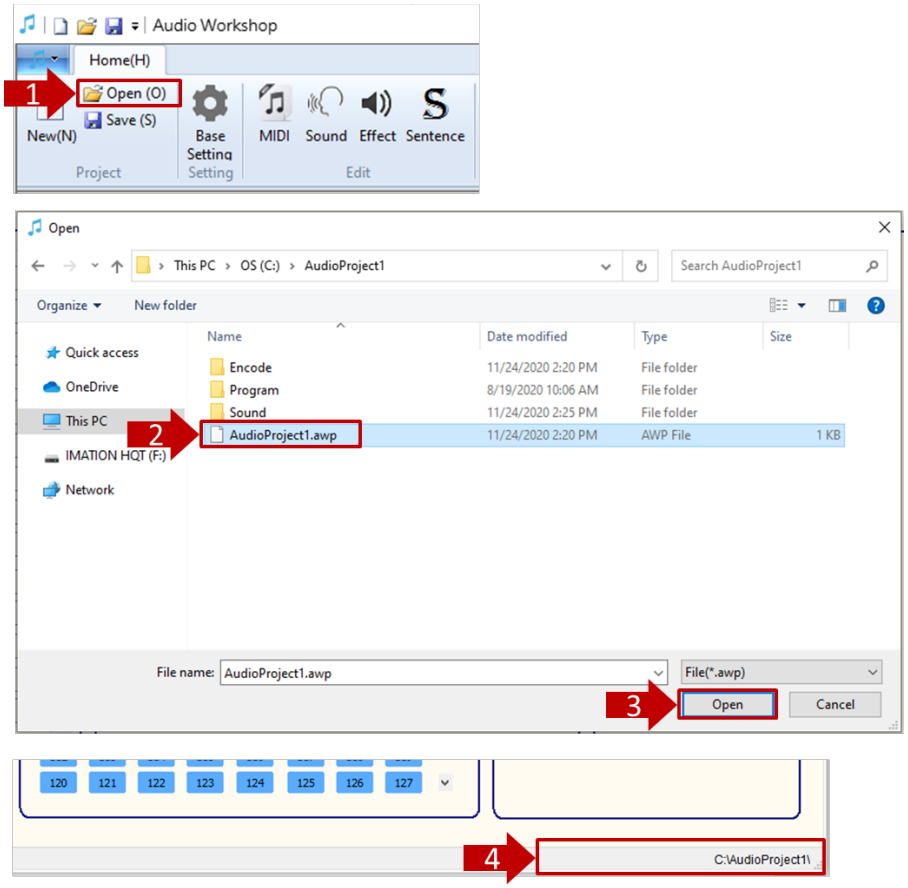
3.8 Download Function



- Step 1: Click "Download".
- Step 2: Select the download action.
  1: Download .Bin in the project.
  2: Download .Hex in the project.
  3: Open and download .Bin.
  4: Open and download .Hex.
  5: Use e-link32 Pro to download the bin file in the project.
  6: Use e-link32 Pro to download the hex file in the project.
  Note: Refer to Appendix 6.2 section for e-link32 Pro connection method.
- Step 3: When the download progress reaches 100%, the download has completed.

3.9 Open a Project

Users can open or switch to the Audio project by clicking "Open".



- Step 1: Click "Open".
- Step 2: Select the project "xxxx.awp" to be opened (the last saved project path is used automatically by default), such as AudioProject1.awp.
- Step 3: Click "Open" to enter the operating interface, which means the project has been opened successfully.
- Step 4: When opening the project, the project path is shown in the information field at the bottom of the interface.

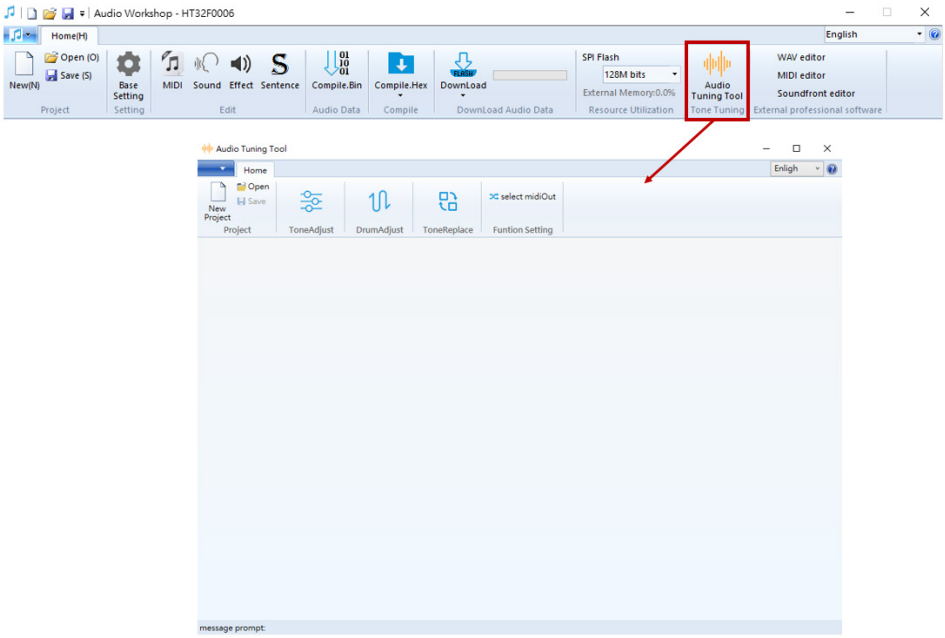
4. Tone Editing Function

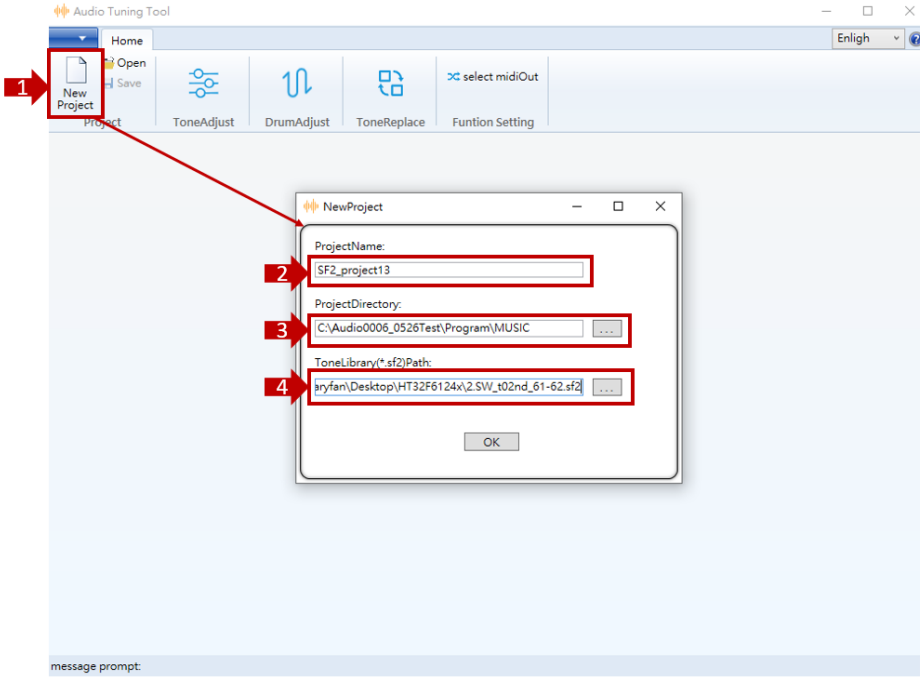
For the HT32F61355/HT32F61356/HT32F61357 and HT32F0006 devices, the Audio Workshop provides a tone editing function, using which tone parameters can be modified or replaced with customized tones.

Multiple tone projects can be created under an Audio Workshop project. The following shows the correct steps.

4.1 Create a New Project

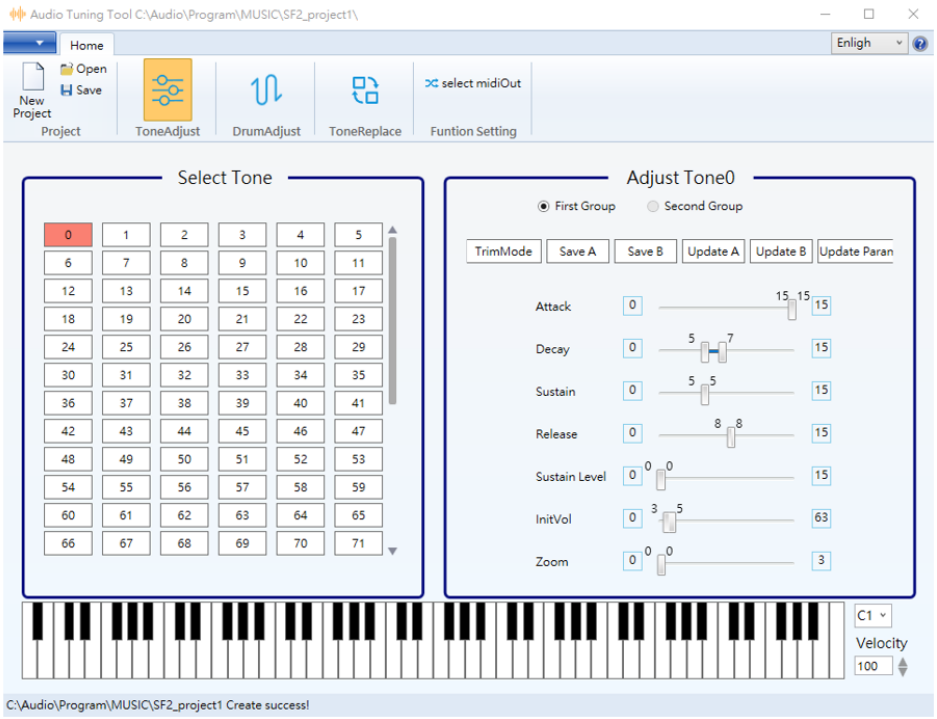
Click "Audio Tuning Tool" on the main interface of Audio Workshop to start the tone editing function interface, as shown in the following figure.



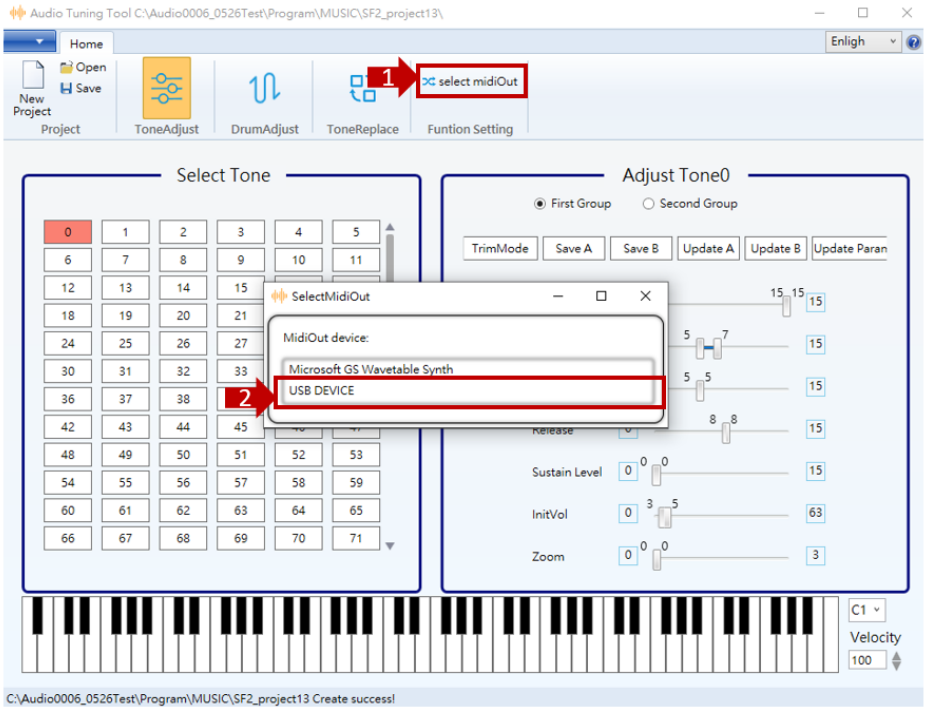


- Step 1: Click "New Project".
- Step 2: Set the tone project name.
- Step 3: Set the tone project path by using the default path.
- Step 4: Set the tone library (*.sf2) path to load.

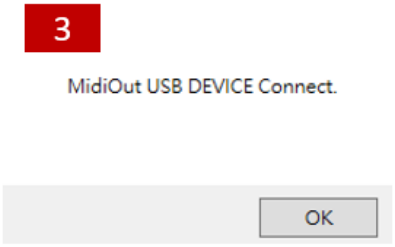
After the tone library has been loaded, it will enter the tone editing interface as shown in the following figure.



4.2 Select MidiOut Device



- Step 1: Select MidiOut device.
- Step 2: Select "USB DEVICE".
  First ensure that the PC USB interface is connected to the USB_A interface on the development board.
- Step 3: When the setting has completed, a "MidiOut USB DEVICE Connect" message will appear and then click "OK".

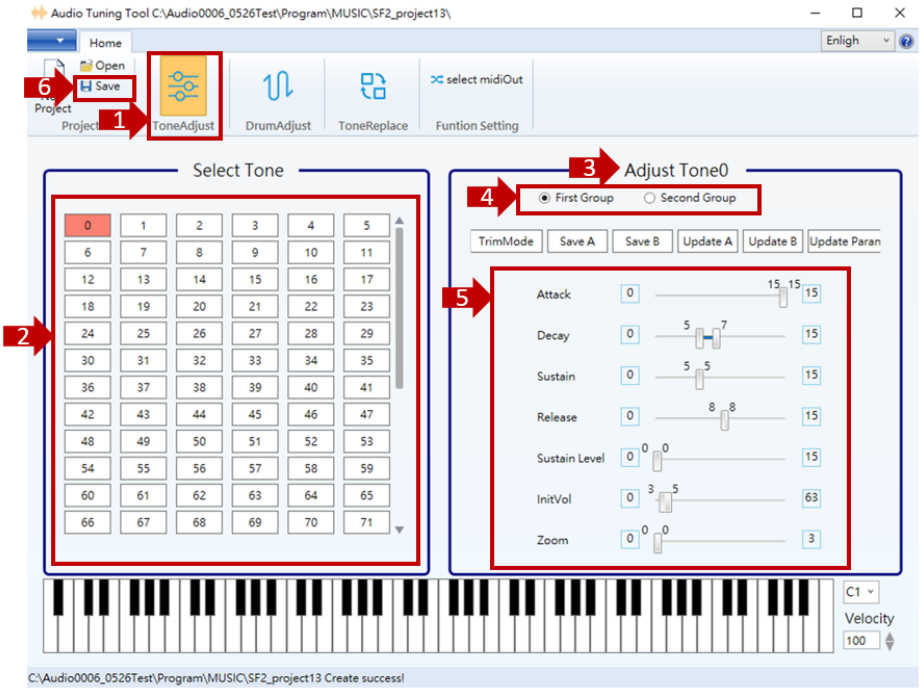


4.3 Tone Parameter Adjustment

This function provides tone parameter "coarse" and "trim" adjustment modes. The tone parameters being adjusted can be transmitted to the development board without programming and the adjusted effect can be heard immediately.

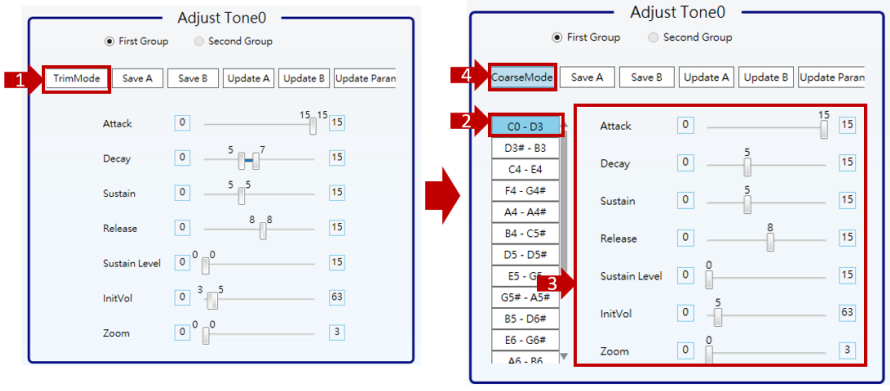
- Tone coarse adjustment: The parameters of all tone samples within the tone are adjusted together
- Tone trim adjustment: The parameters can be adjusted for individual tone samples within the tone

4.3.1 Tone Coarse Adjustment



- Step 1: Click "ToneAdjust".
- Step 2: Select the tone number (0~127).
- Step 3: The interface displays the selected tone number.
- Step 4: Select the first or second group of tones The loaded tone library (*.sf2) requires the second group of tones before this group is selected to modify the parameters. If it does not exist, the Second Group option cannot be selected.
- Step 5: Set the tone parameters:
  Provide parameters (Attack/Decay/Sustain/Release/Sustain Level/InitVol/Zoom) to be adjusted.
- Step 6: Save the tone parameters.

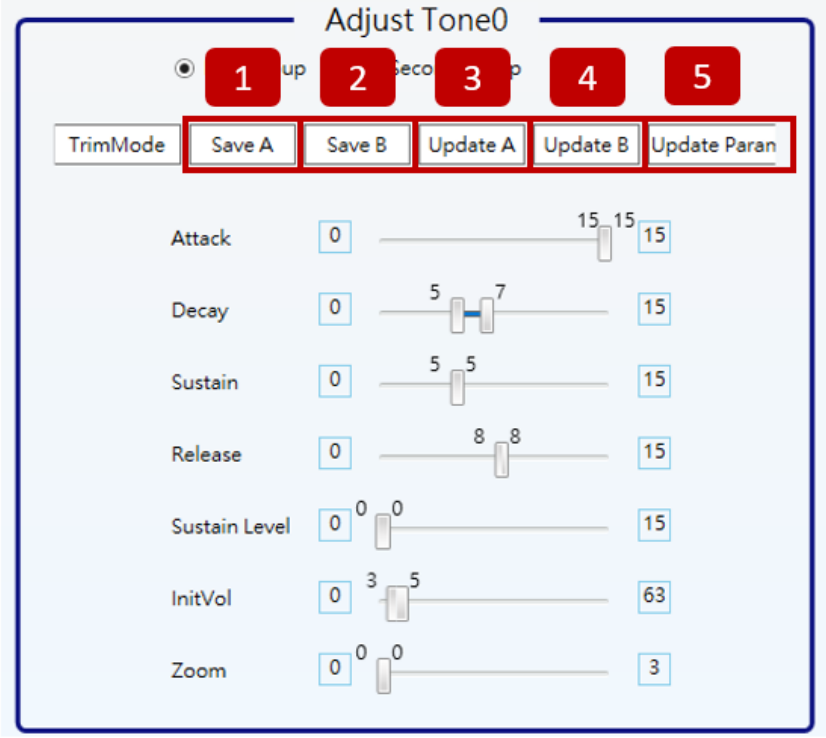
4.3.2 Tone Trim Adjustment

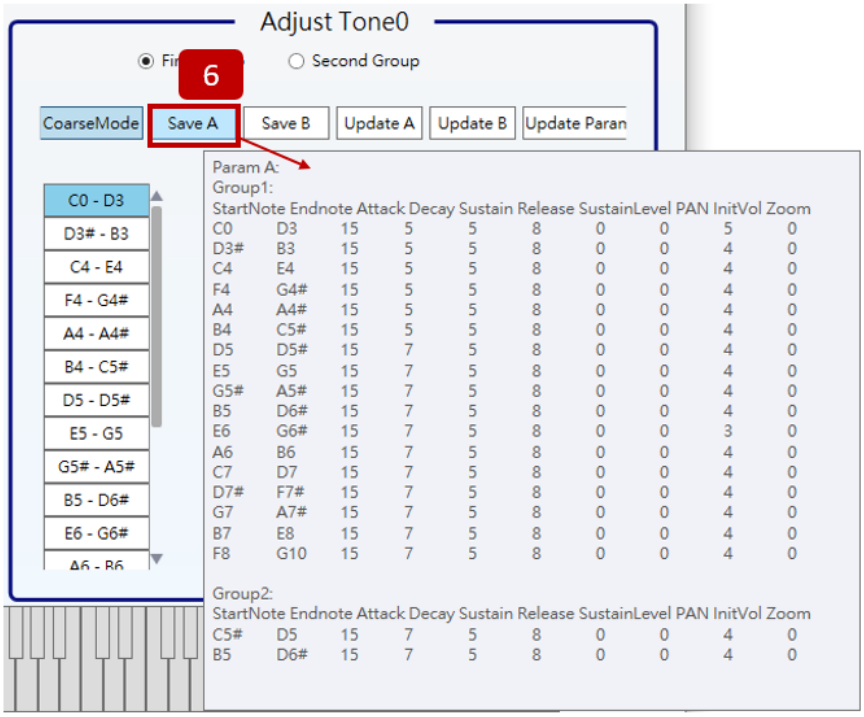


- Step 1: Click "TrimMode" to switch to the tone trim adjustment mode.
- Step 2: Display and select a tone sample (WAV) within the tone.
  For example, select the "C0-D3" sample within the tone 0.

- Step 3: Set the tone parameters (Attack/Decay/Sustain/Release/Sustain Level/InitVol/Zoom). For example, set the tone parameters of the tone 0 "C0-D3" sample.
- Step 4: Switch back to tone coarse adjustment mode.

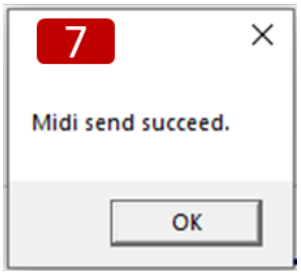
4.3.3 Debug ADSR Parameters





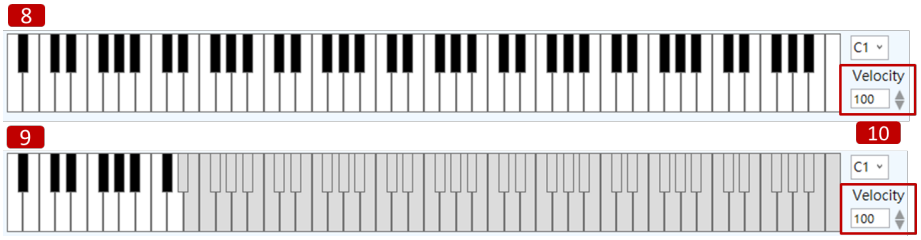
1. Save A: Store the current ADSR parameter to parameter A.
2. Save B: Store the current ADSR parameter to parameter B.

   The saved the parameters A/B can be used to quickly compare the parameter differences.
3. Update A: Send parameter A to the development board.
4. Update B: Send parameter B to the development board.
5. Update Param: Send the current ADSR parameters to the development board.
6. Display the parameters saved by parameter A/B (hover the cursor over "Save A/B").



7. When "Update Param/Update A/Update B" are updated successfully.

   Note: Confirm the PC USB connection to the development board USB_A before updating the parameters.



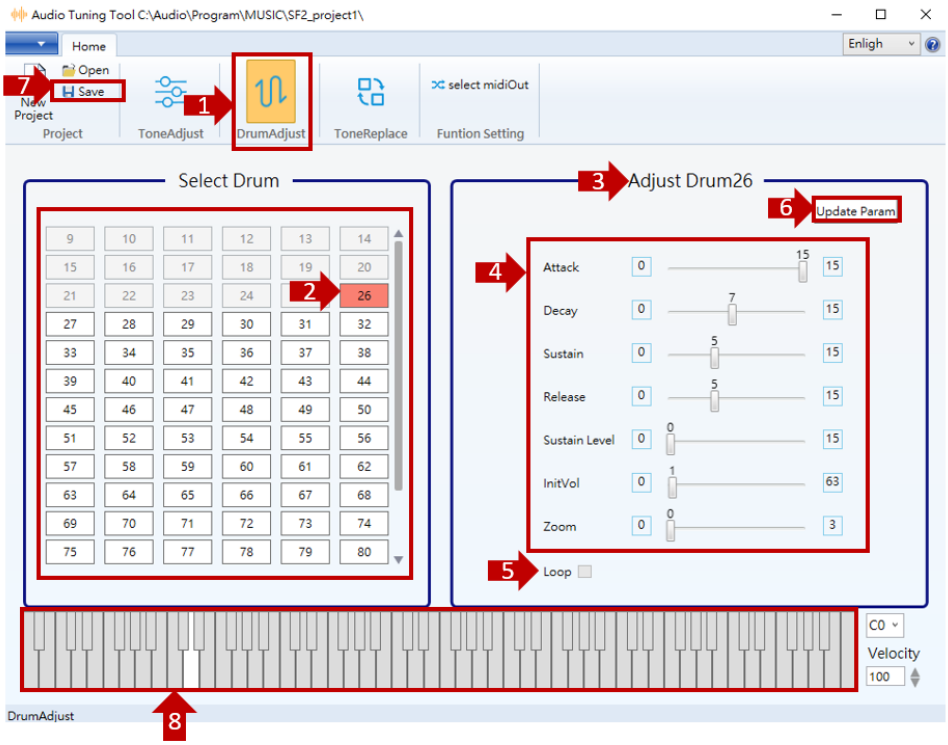
8. Keyboard (Tone coarse adjustment): Display the 88Key range and click the keyboard to test the tone sound effect.
9. Keyboard (Tone trim adjustment): Display the musical note range of the wav samples of the tone. Click the keyboard on the screen to test the tone sound effect.
10. Adjust keyboard pronunciation velocity (0~127).

Note: 1. After debugging the tone parameter effects, the adjusted tones should be replaced into the system tone library using the "Replace" function. Use the "Compile.Bin" and "Compile.Hex" functions of the Audio Workshop to generate the bin and hex files, which are downloaded to the development board using the "Download" function. After this, the adjusted tones can be called normally.

2. Confirm the PC USB connection to the development board USB_A before updating the parameters.

4.4 Drum Parameter Adjustment

4.4.1 Debug the ADSR Parameters



- Step 1: Click the "DrumAdjust" icon on the interface.
- Step 2: Select the drum number with the range of 9 to 96.
- Step 3: Display the selected drum number using the interface.
- Step 4: Set the drum parameters:
  Provide the parameters (Attack/Decay/Sustain/Release/Sustain Level/InitVol/Zoom) to adjust.
- Step 5: Show whether the drum has a Loop:
  A tick indicates that there is a Loop, in this case there is no Loop.
- Step 6: Send the current ADSR parameters to the development board.
- Step 7: Save the drum parameters.
- Step 8: Click the keyboard on the screen to test the current drum sound effect.

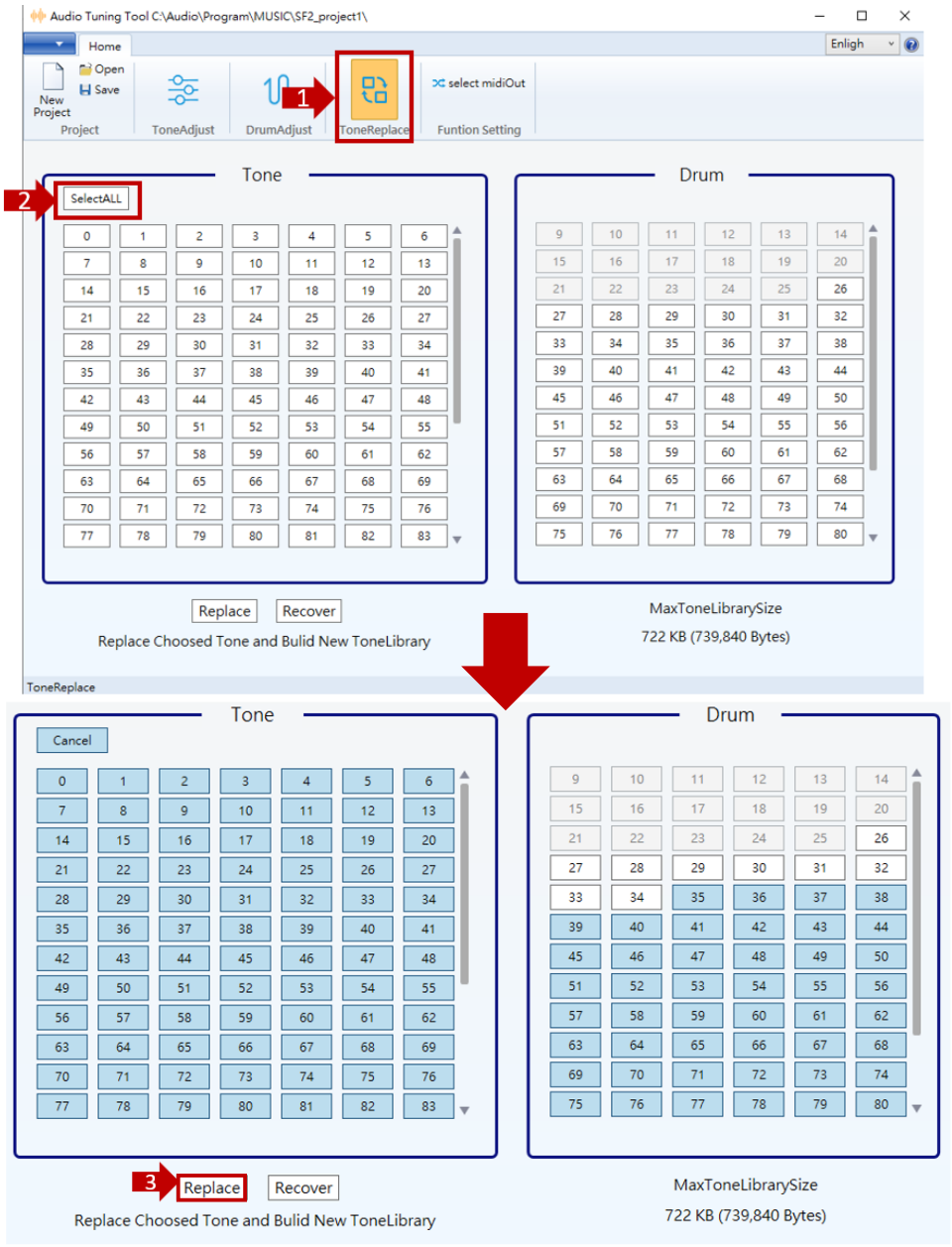
Note: 1. After debugging the drum parameter effects, the adjusted drums should be replaced into the system tone library using the "Replace" function. Use the "Compile.Bin" and "Compile.Hex" functions of the Audio Workshop to generate the bin and hex files, which are downloaded to the development board using the "Download" function. After this, the adjusted drums can be called normally.

2. Confirm the PC USB connection to the development board USB_A before updating the parameters.

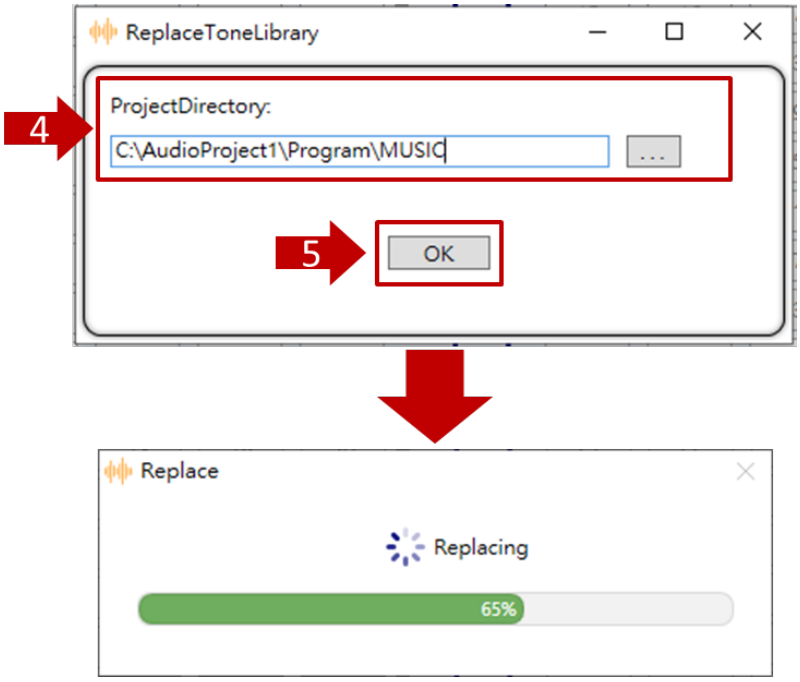
4.5 Tone Replacement

- Tone library replacement: Replace the system tone library (all tones) with the user tone library (all tones).
- Individual tone replacement: replace the individual user tone or drum.

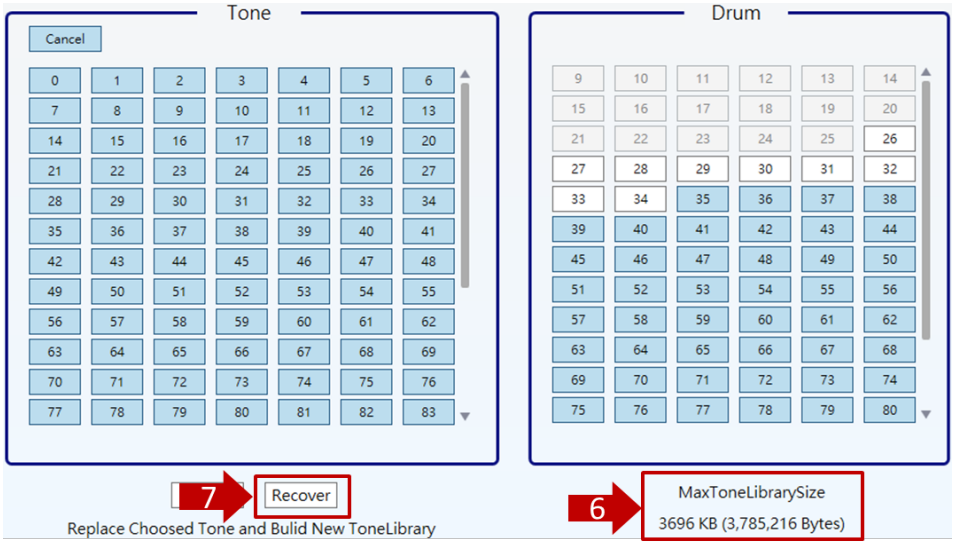
4.5.1 Tone Library Replacement



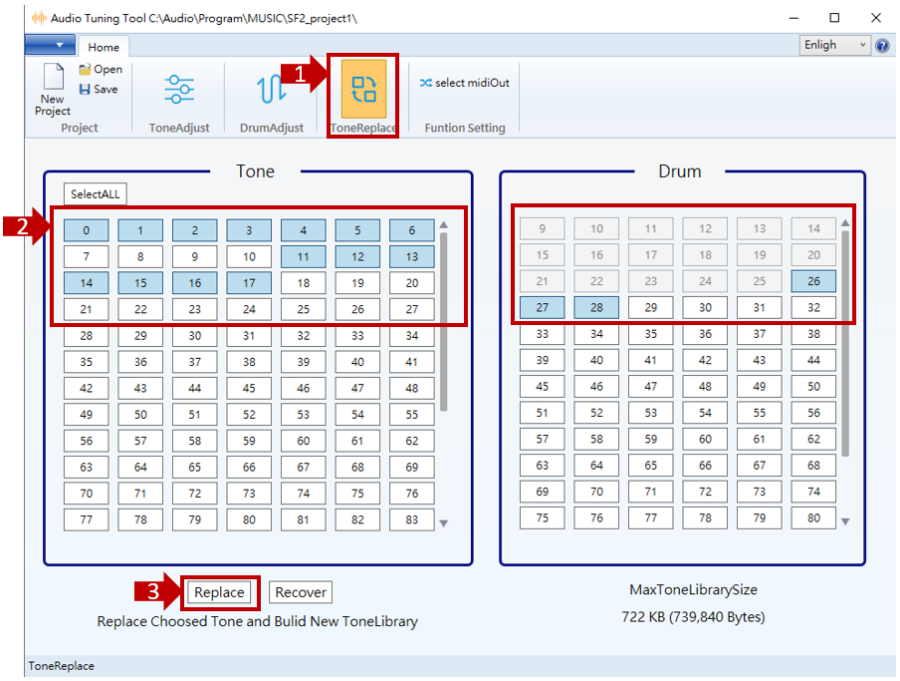
- Step 1: Click "ToneReplace" on the interface.
- Step 2: Click "SelectAll" for all tones and percussions to be selected (shown in blue).
- Step 3: Click "Replace", then a window as shown below will appear.



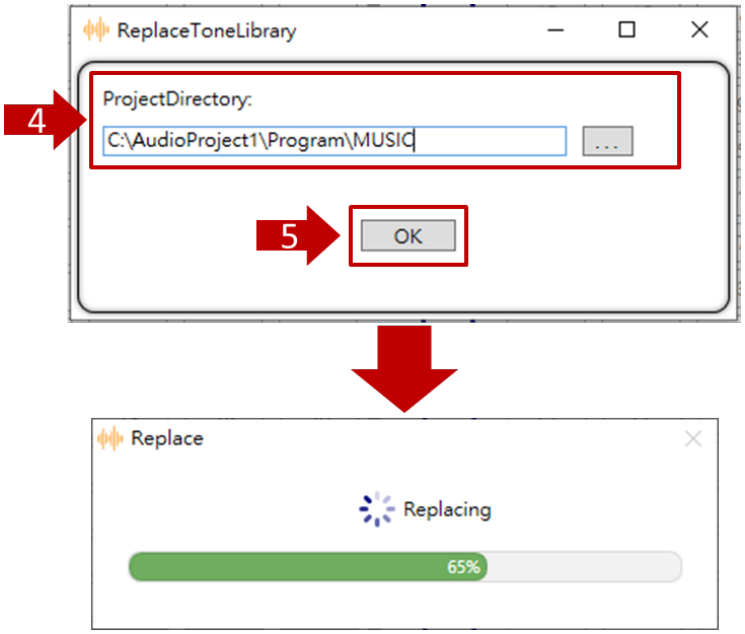
- Step 4: Set the replaceable Audio Workshop project path and use the default path.
- Step 5: Click "OK" to start the tone and drum data replacement. A 100% indication means the replacement has completed.
- Step 6: When the replacement has completed, the interface will estimate the maximum resource (SPI Flash ROM capacity) used after the tone replacement.
- Step 7: Click "Recover" to restore the default system tone library.



4.5.2 Individual Tone Replacement

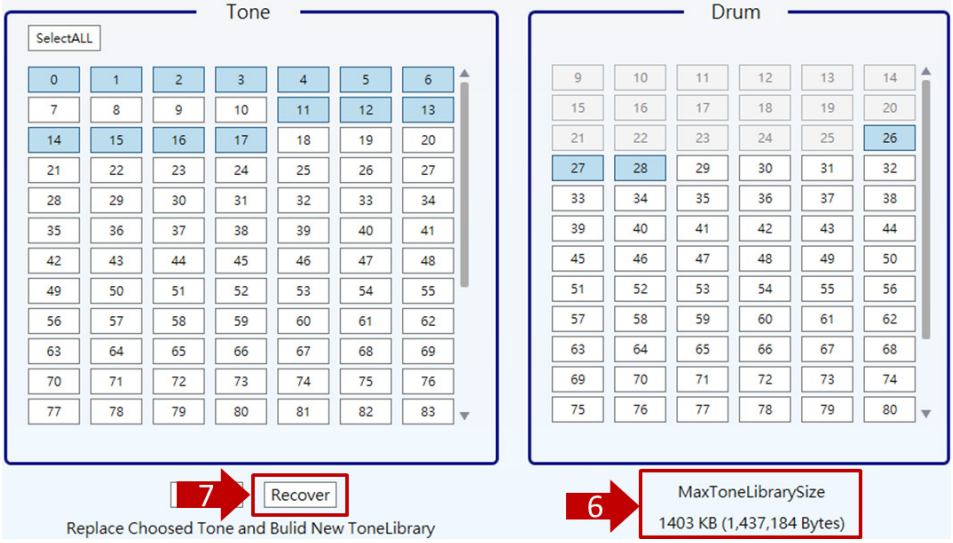


- Step 1: Click "ToneReplace" on the interface.
- Step 2: Select the tone and drum number to replace (If selected, it appears in blue).
  For example: select the tones No.0/1/2/3/4/5/6/11/12/13/14/15/16/17, and the drums No.26/27/28.
  In addition to single-clicking the tone number, the "Shift" key can be used to select more than one tone at a time.
  For example: tone (0 "Shift" 6, 11 "Shift" 13, 14 "Shift" 17), drum (26 "Shift" 28).
- Step 3: Click "Replace", then a window as shown below will appear.

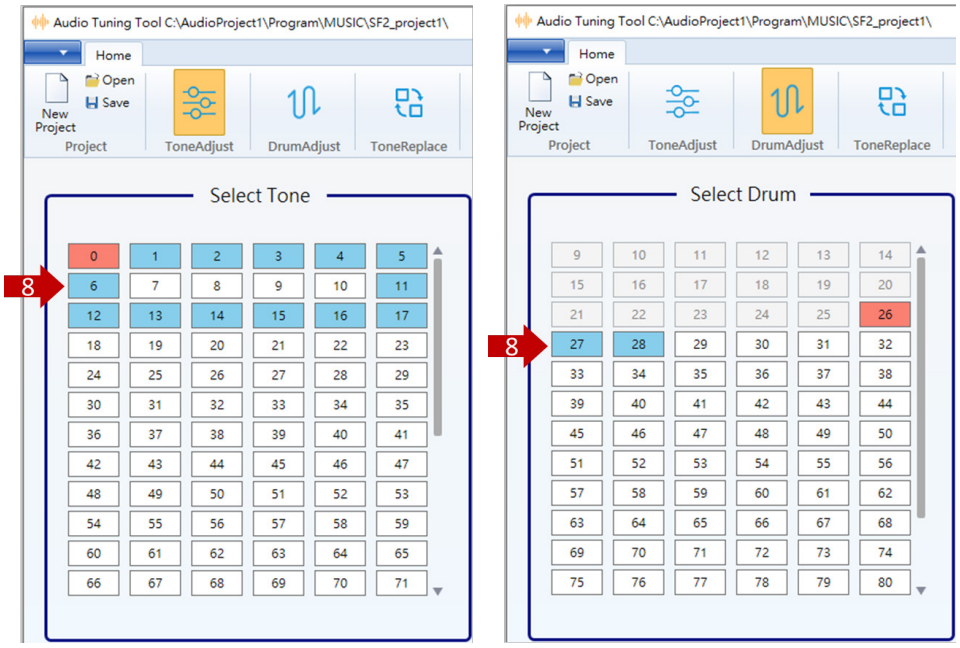


- Step 4: Set the replaceable Audio Workshop project path and just use the default path.

- Step 5: Click "OK" to start the tone and drum data replacement. An indication of 100% means the replacement has completed.
- Step 6: When the replacement has completed, the interface will estimate the maximum resource (SPI Flash ROM capacity) used after the tone replacement.
- Step 7: Click "Recover" to restore the default system tone library.

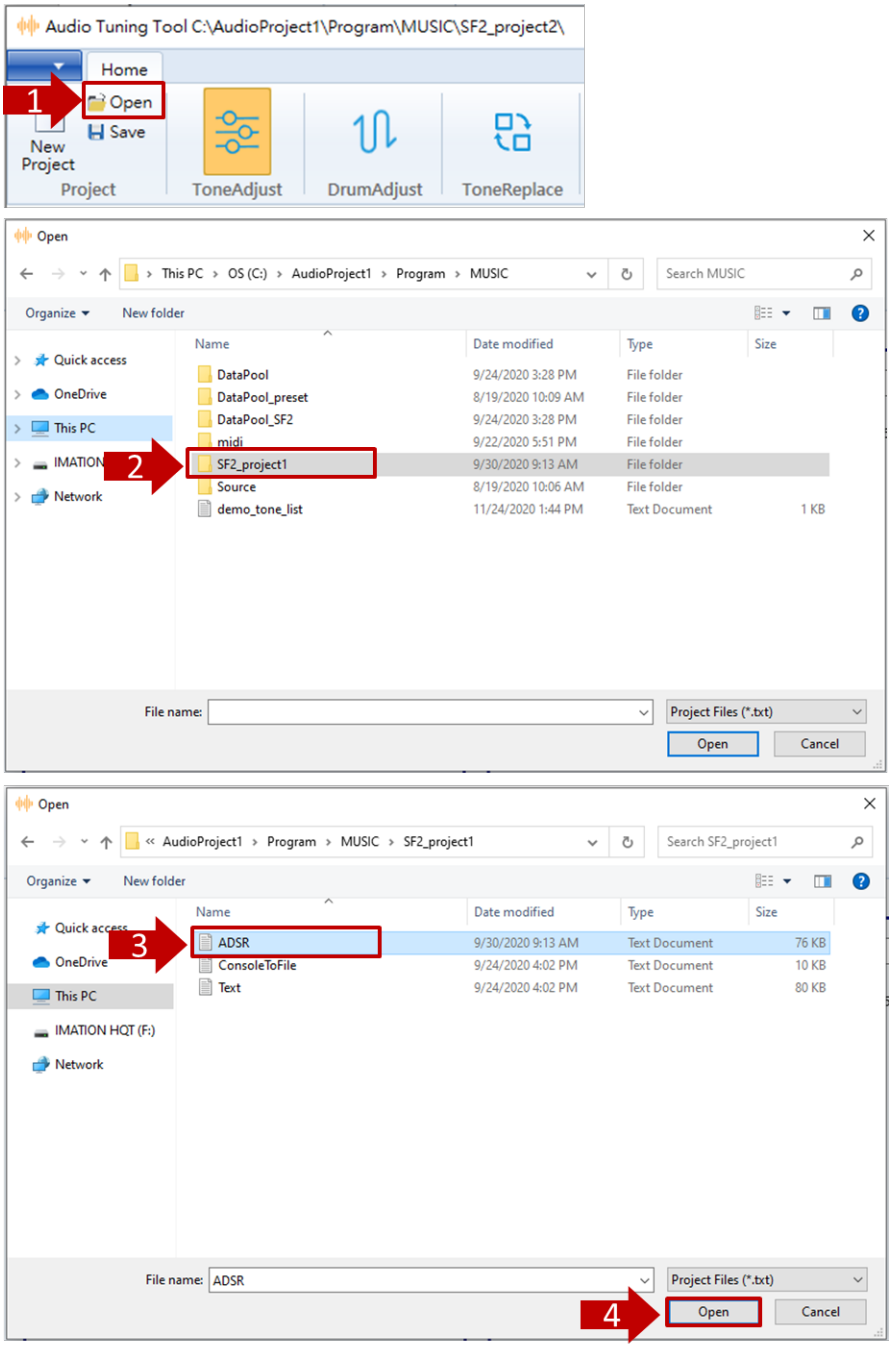


- Step 8: The replaced tones will be displayed on the "ToneAdjust" and "DrumAdjust" interfaces synchronously.

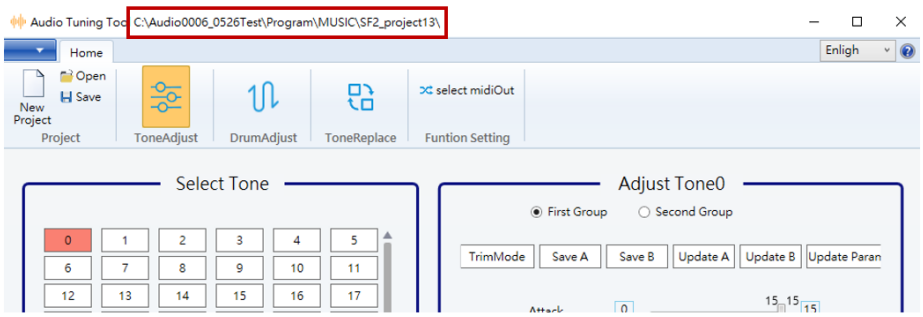


4.6 Open a Project

Users can open or switch to a tone project by clicking "Open".



- Step 1: Click "Open" on the interface.
- Step 2: Select the tone project to be opened.
- Step 3: Select "ADSR.txt" in this directory.
- Step 4: Click "Open" to enter the operating interface, as shown below, which indicates the project opening has completed.



- After opening the project, the project path will be displayed in the title bar as shown above
- The operating interface will show: the tone parameters, drum parameters, previous tone replacement information

5. Function Library Description

5.1 Play Functions

Functions	Description
void L_Audio_Init(void)	Initialisation setting
void L_Audio_Play(u8 audio_type, u32 index)	Play (MIDI/Sound/Effect)
void L_Audio_Stop(void)	Stop playing all sounds
void L_Midi_Stop(void)	Stop playing MIDI
void L_Sntnc_Main_Loop(void)	Main loop sentence processing
void L_Sntnc_Play(u32 index)	Play sentence
void L_Sntnc_Stop(void)	Stop playing sentence
EventStatus L_Midi_Finish(void)	Determine if the MIDI has finished playing
EventStatus L_Audio_Finish(void)	Determine if the sound/effect have finished playing
void L_Audio_Vol(u8 volume)	Volume setting

5.1.1 Initialisation Function

L_Audio_Init()

Description: Initialise the parameters Clock/Timer/QSPI/MIDI Engine/DAC/Volume and so on.

5.1.2 Play Functions (User function)

L_Audio_Play(audio_type, index)

Description: Play MIDI/Sound/Effect

Parameter:

audio type:

c_midi_type: MIDI
c_voice_type: Sound
c_sound_type: Effect

index: What track (starting with 1)

For example:

```
L_Audio_Play(c_midi_type, 2);  // Play the second midi
```

L_Audio_Stop()

Description: Stop playing the MIDI and stop all sounds that are playing

L_Midi_Stop()

Description: Stop playing the MIDI and stop all MIDI sounds that are playing

L_Sntnc_Main_Loop()

Description: To use the play sentence function, add this function to the while(1) in the main loop main()

L_Sntnc_Play(index)

Description: Play sentence

Parameter:

Index: Which sentence (starting with 1)

For example:

```
L_Sntnc_Play(2);  // Play the second sentence
```

L_Sntnc_Stop()

Description: stop playing the sentence

5.1.3 Volume Setting Function - User function

L_Audio_Vol(volume)

Parameter:

Volume: 0~15.

0 represents mute, 1~15 represents the volume from small to large.

Example:

```
L_Audio_Vol(8);  // Set the volume to value 8
```

5.1.4 Play Status Function

EventStatus L_Audio_Finish()

The return value:

ENABLE: Sound/effect play has been implemented

DISABLE: Sound/effect play has not completed

EventStatus L_Midi_Finish()

The return value:

ENABLE: MIDI play has been implemented

DISABLE: MIDI play has not completed

5.2 Effect Functions

Functions	Description
void __L_Set_PadelSlop(u8 sloppadel)	Sustain Pedal
void __L_Set_Pitch_Bend(u8 r_level)	Pitch Band
void __L_Set_Vib(u8 r_level)	Vibrato
void __L_dec_Tempo()	MIDI tempo decrease
void __L_inc_Tempo()	MIDI tempo increase
void __L_normal_Tempo()	MIDI returns to normal tempo

Functions	Description
void __L_Set_Tempo1(u16 r_tempo)	Set MIDI tempo
void __L_Limit_Relase(u8 level)	Quickly turn off the sound

5.2.1 Effect Functions

__L_Set_PadelSlop(u8 sloppadel)

Parameter:

Sloppadel: 0~8.

8 represents the sustain pedal off. The smaller this value, the longer the sustain continues

For example:

```
__L_Set_PadelSlop(8);  // turn off the sustain pedal
```

__L_Set_Pitch_Bend(r_level)

Parameter:

r_level: 0~255

0~100: The pitch goes up, 100 for 2 semitones

255~156: The pitch goes down, 156 for low 2 semitones.

For example:

```
__L_Set_Pitch_Bend (-50);  // go down a semitone
__L_Set_Pitch_Bend (50);  // go up a semitone
```

__L_Set_Vib(r_level)

Parameter:

r_level: 0~4

Vibrato sets the r_level value to control the dither range, the r-level is in units of 5 cents.

For example: If r_level=1, the vibrato jitters 5 cents. If r_level=2, the vibrato jitters 10 cents.

For example:

```
__L_Set_Vib (4);  // the vibrato jitters 20 cents
```

__L_dec_Tempo()

Description: MIDI tempo decreases by 1 beat, the range is 30~280 beats per minute.

__L_inc_Tempo()

Description: MIDI tempo increases by 1 beat, the range is 30~280 beats per minute.

__L_normal_Tempo()

Description: MIDI returns to normal tempo, the range is 30~280 beats per minute.

__L_Set_Tempo1(u16 r_tempo)

Description: Set MIDI tempo, the r_tempo range is from 30 to 280.

For example:

```
__L_Set_Tempo1 (80);  // MIDI tempo of 80
```

__L_Limit_Relase(u8 level)

Description: Quickly turn off the sound which is effective for the sound being played.

Parameter:

Level: 1~7. Here 7 represents the residual sound that closes the quickest while 1 represents that the residual sound that closes the slowest.

For example:

__L_Limit_Relase (4);

5.3 System Functions

Functions	Description
void __L_Func_Midi_Mute_Control(bool onoff)	MIDI play mute control
void __L_Func_Midi_Pause_Control(bool onoff)	MIDI play pause control
void __L_User_Measure_Finish(void)	Measure function
void __L_User_Beat_Finish(void)	Beat function
void __L_User_Half_Beat_Finish(void)	Half-beat function
void __L_Note_On(u8 r_type, u8 r_def_num, u8 r_keycode, u8 r_volume, u16 r_id_delay, u8 r_pan, u8 r_track)	Note on
void __L_Note_Off(u8 r_id)	Note off
void __L_Midi_NRPN(u32 r_NRPN)	NRPN event output
bool __L_Check_Wave_End()	Check if the channel sound play has been finished
void __L_Play_Midi(u16 number, E_Midi_Play_Begin_Middle_Typedf _e_play_begin_middle, E_Midi_Pause_Typedf r_e_pause)	Play MIDI

5.3.1 System Functions

__L_Func_Midi_Mute_Control(bool onoff)

Description: MIDI play mute control. MIDI event time continues.

Parameter:

Bool type:

TRUE mute

FALSE normal

For example:

__L_Func_Midi_Mute (TRUE); // Play mute

__L_Func_Midi_Pause_Control (bool onoff)

Description: MIDI play pause control. The MIDI event time is paused and the sound currently being played is stopped.

Parameter:

Bool type:

TRUE Pause

FALSE normal

For example:

__L_Func_Midi_Pause_Control(TRUE); // Play pause

__L_User_Measure_Finish ()

Description: Function call interface. This allows designers to know that Measure has finished.

__L_User_Beat_Finish ()

Description: Function call interface. This allows designers to know that Beat has finished.

__L_User_Half_Beat_Finish(void)

Description: Function call interface. This allows designers to know that Half Beat has finished.

__L_Note_On(u8 r_type, u8 r_def_num, u8 r_keycode, u8 r_volume, u16 r_id_delay, u8 r_pan, u8 r_track)

Description: Note on

Parameter:

r_type: Sound type(c_notekey_play, c_midi_play)

If the sound type is c_notekey_play, it requires the corresponding __L_Note_Off to turn off the sound.

If the sound type is c_midi_play, the sound length is r_id_delay×1/48beat.

r_def_num: Tone number

r_keycode: Pitch (0~127)

r_volume: Volume (0~63)

r_id_delay: Sound ID and duration

r_pan=0; By default

r_track=0; By default

For example:

The c_notekey_play represents the key sound but also needs the corresponding close note sound. If c_midi_play, just provide the sound length.

__L_Note_On(c_notekey_play, 20, 12, 0, 1, 0, 0)

The function indicates that the key sound is on and the tone number is 20. The sound pitch keycode is 12, the maximum volume is 0, and the r_id_delay value of 1 is used to turn off notes. If the user wants to turn off the sound, turn off the key sound using this function with __L_Note_Off(1) when the r_id_delay=1

__L_Note_On(c_midi_play, 20, 12, 0, 1, 0, 0)

This function indicates that this sound will be released after a 1/48 beat. A r_id_delay=1 gives a sound length of 1/48 beat. If it is 48, then the sound length is 1 beat. Note that the off function is not required.

__L_Note_Off(u8 r_id)

Description: Note off

Parameter:

r_id: Correspond to the r_id_delay in the __L_Note_On()

__L_Midi_NRPN(u32 r_NRPN)

Description: Function call interface. NRPN is converted by the transfer converter program.

The complete midi NRPN event should contain 0x6/x26/0x62/0x63.

__L_Check_Wave_End()

Description: Check if the channel sound play has been finished.

Return value:

TRUE: Finished.

FALSE: There still exists a sound in the channel.

void __L_Play_Midi(u16 number, E_Midi_Play_Begin_Middle_Typedf r_e_play_begin_middle, E_Midi_Pause_Typedf r_e_pause)

Description: play midi

Parameter:

Number: midi number

r_e_play_begin_middle=c_enum_midi_play_begin; By default

r_e_pause=c_enum_midi_not_pause; Return value by default

For example:

__L_Play_Midi(0,c_enum_midi_play_begin,c_enum_midi_not_pause)

This sentence plays the first MIDI which has been added to the workshop.

5.4 Other Descriptions

- MIDI IN Event

  The USB MIDI and workshop can be selected simultaneously by default.

  #define USB_MIDI_ENABLE (1)

Support usb midi standard input. USB midi support for command and transformation.

The support command table is as follows.

		USB midi in	Convert
Note off(8X)		V	V
Note on(9X)		V	V
Aftertouch(AX)		X	X
Controller(BX)	Volume(7)	V	V
	PAN(A)	V	V
	Padel(40)	V	V
	NRPN(6/26/62/63)	X	V
	All soundoff(78)	V	V
Program Change(Cx)		V	V
Channel Pressure(Dx)		X	X
Pitch Wheel(Ex)		V	V
Meta message(Fx)	End of Track(2F)	X	V
	Time Signature(58)	X	V
	Tempo(51)	X	V

- MIDI total ticks played and the tick currently played.

  __midi_total_tick: total ticks

  __midi_current_tick: the current tick

- When using the remaining space

  When the user needs to use the remaining external Flash memory capacity, check the "APP Update" function. The capacity is 1Mbits by default, which is the amount of user MIDI music to be stored.

  unsigned long const c_addr_user_midi_table=0xcf000;

  // The starting address of MIDI music is in word format

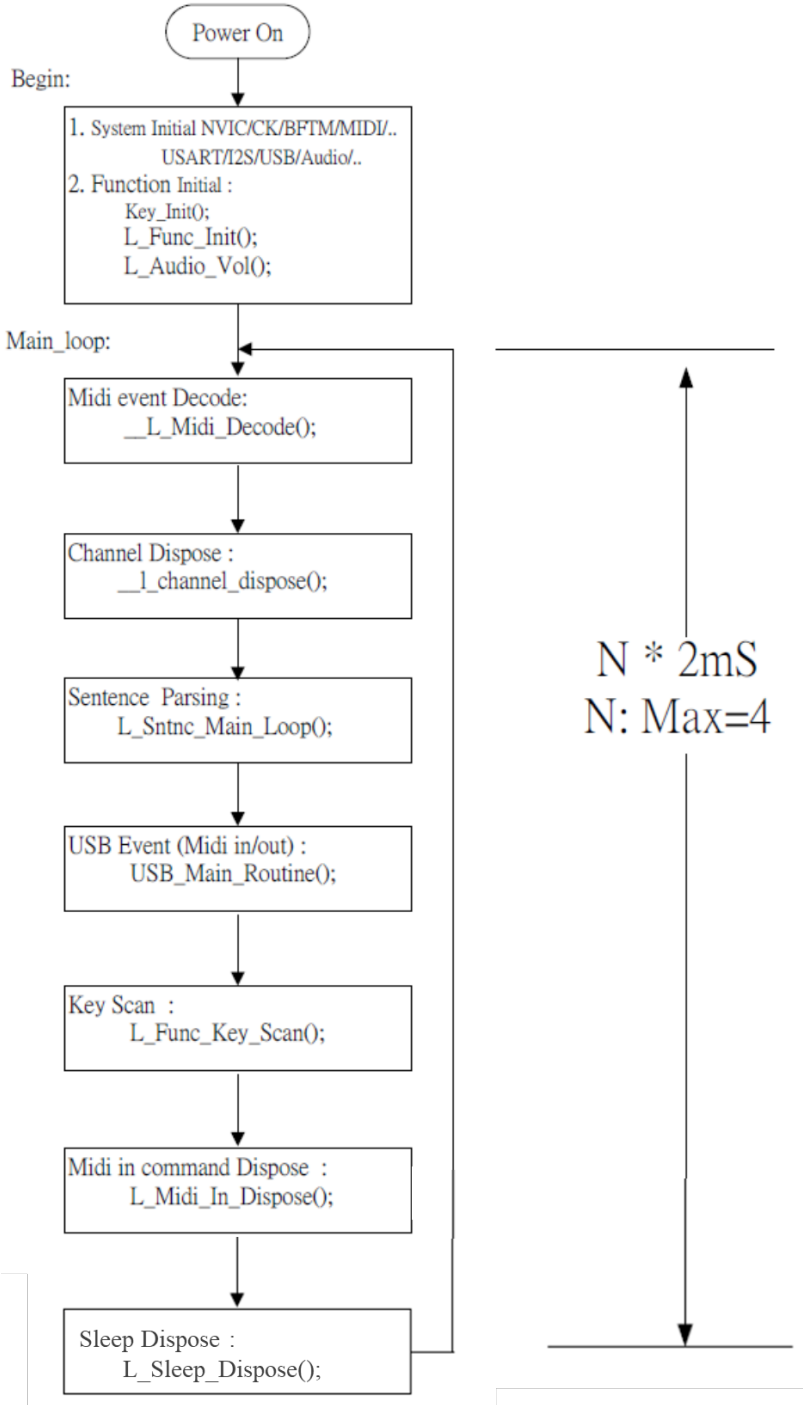
  unsigned long const c_addr_user_flash=0xdf000;

  // The starting address of data storage is in word format

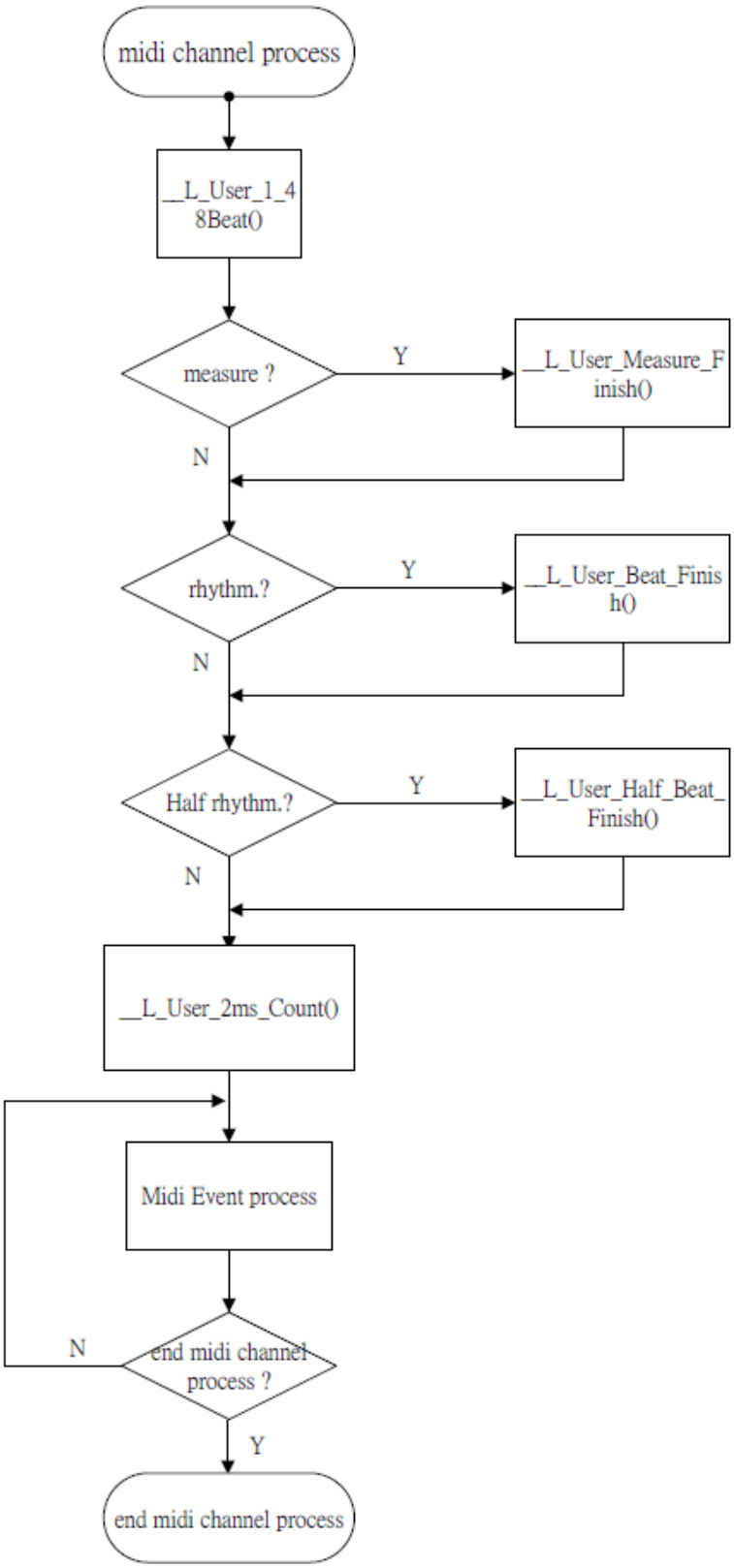
6. Appendix

6.1 Program Flow

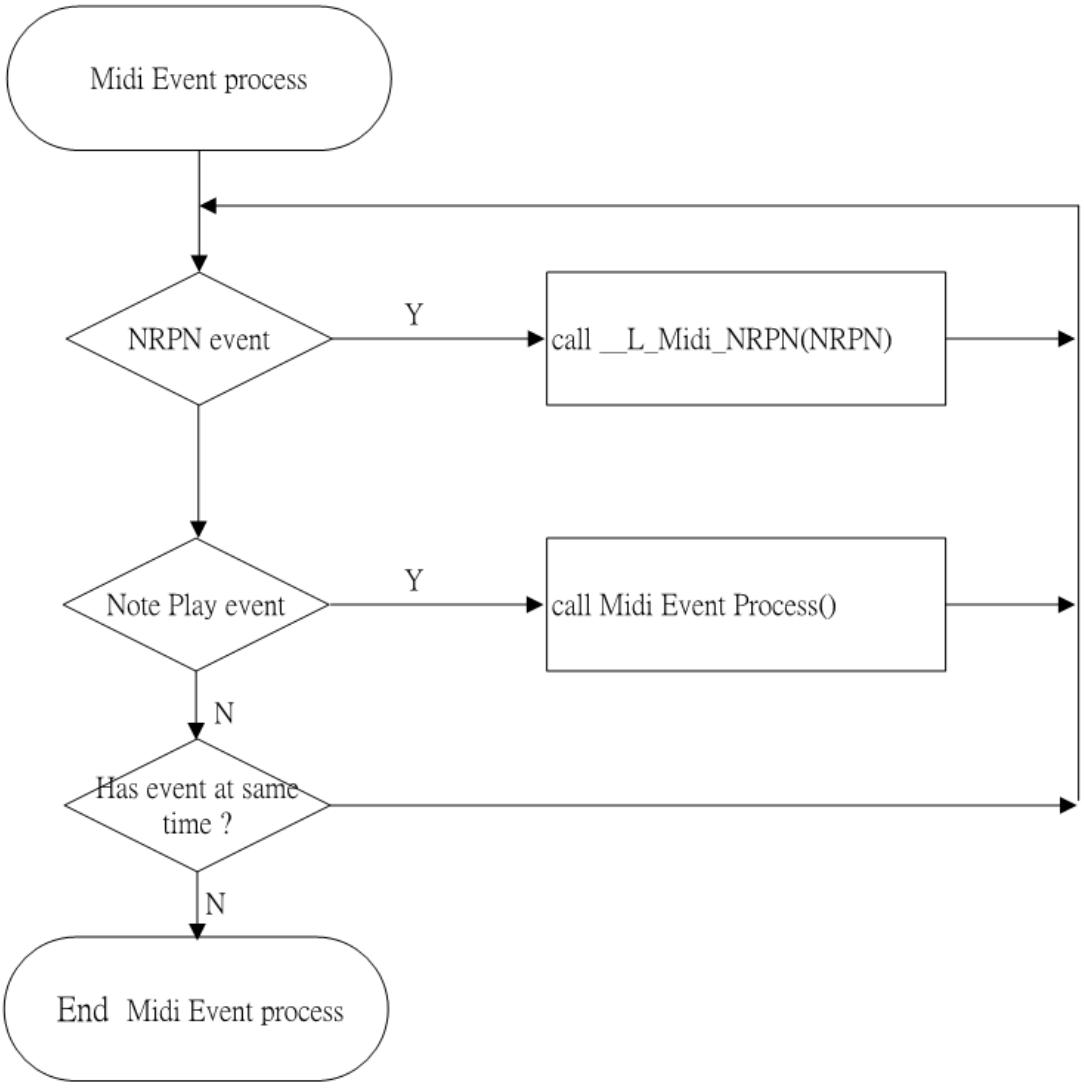
Program Main Flow

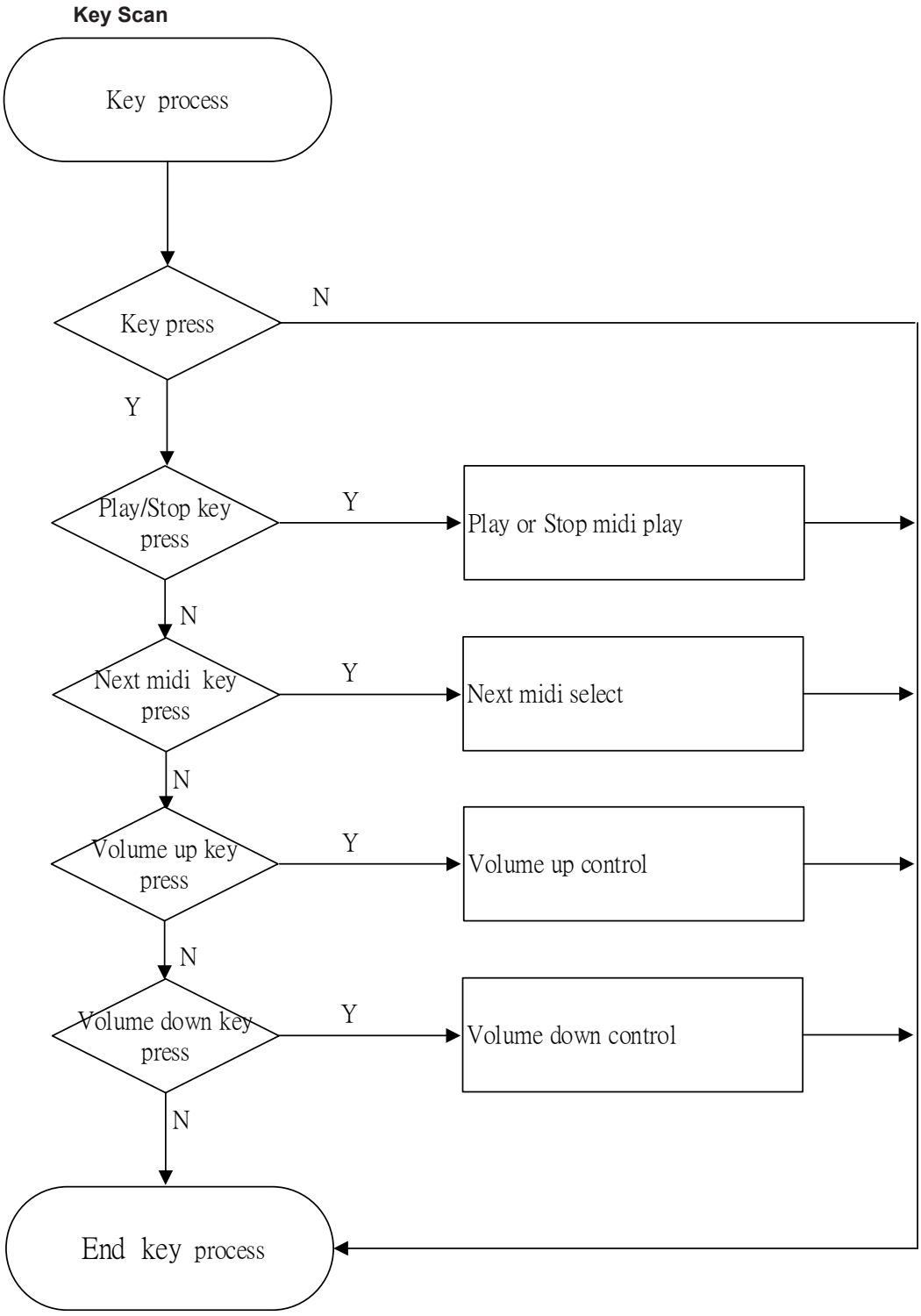


Midi Decode

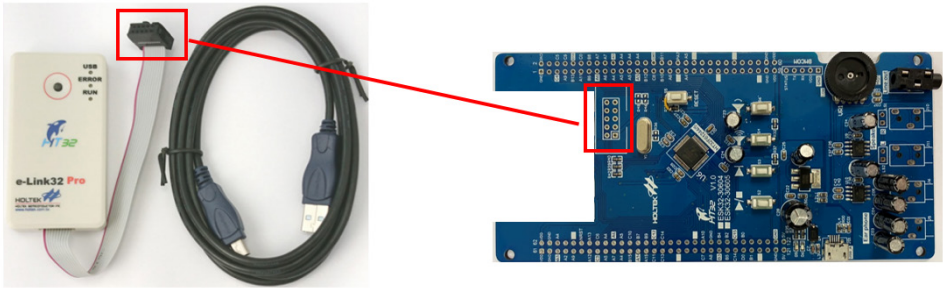

midi channel process
__L_User_1_4 8Beat()
measure ?
Y
__L_User_Measure_F inish()
N
rhythm.?
Y
__L_User_Beat_Finis h()
N
Half rhythm.?
Y
__L_User_Half_Beat_ Finish()
N
__L_User_2ms_Count()
Midi Event process
N
end midi channel process ?
Y
end midi channel process

Midi Event Process

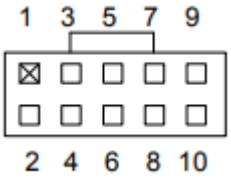



Key Scan
Key process
Key press
N
Y
Play/Stop key press
Y
Play or Stop midi play
N
Next midi key press
Y
Next midi select
N
Volume up key press
Y
Volume up control
N
Volume down key press
Y
Volume down control
N
End key process

6.2 e-link32 Pro Connection



The SWD-10P connector pins on the development board shown in Section 1.3.1 should be connected to an e-link32 Pro interface. Refer to Appendix 6.4 for detailed SWD-10P connector pin description. The e-link32 Pro interface description is as follows.

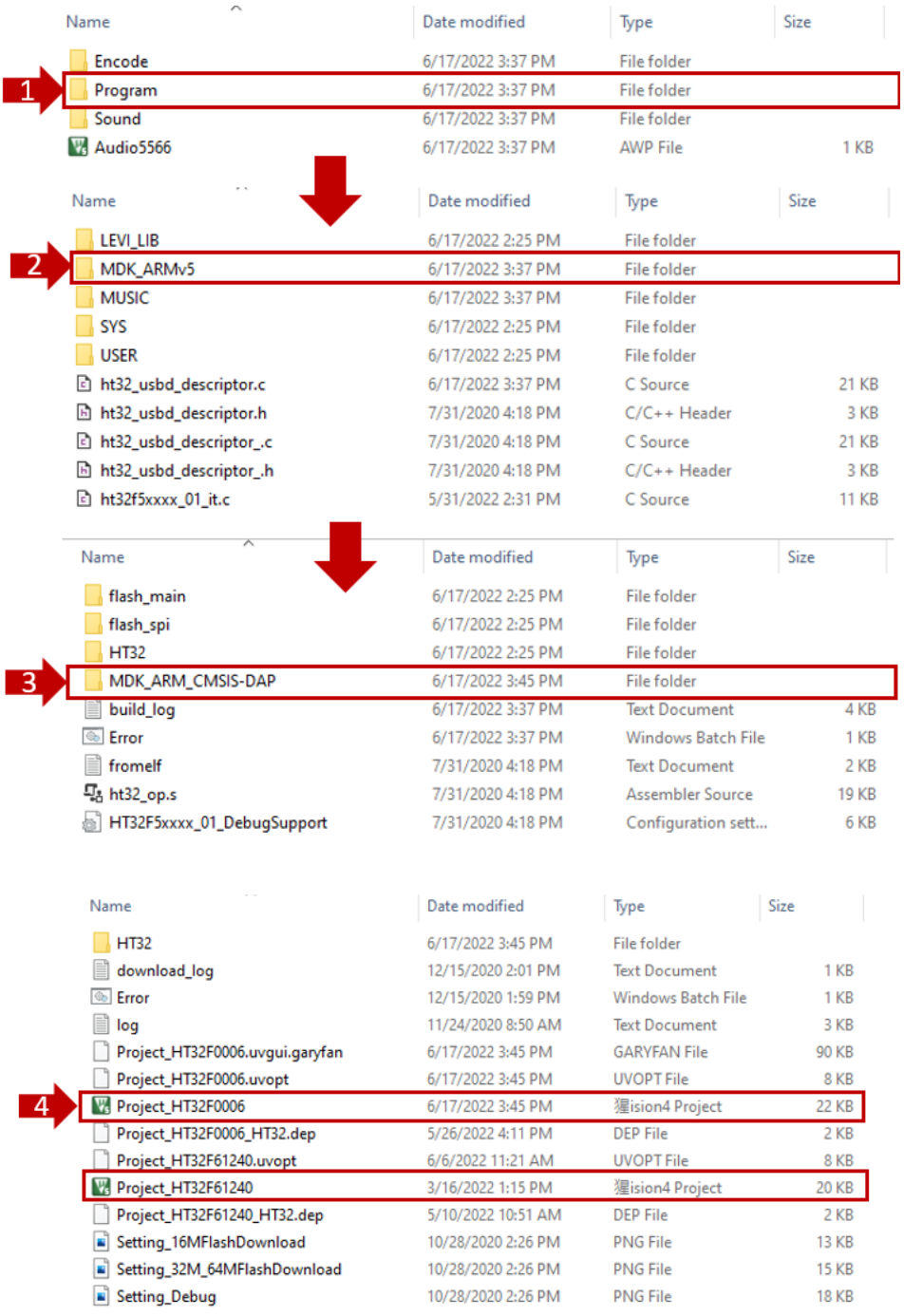


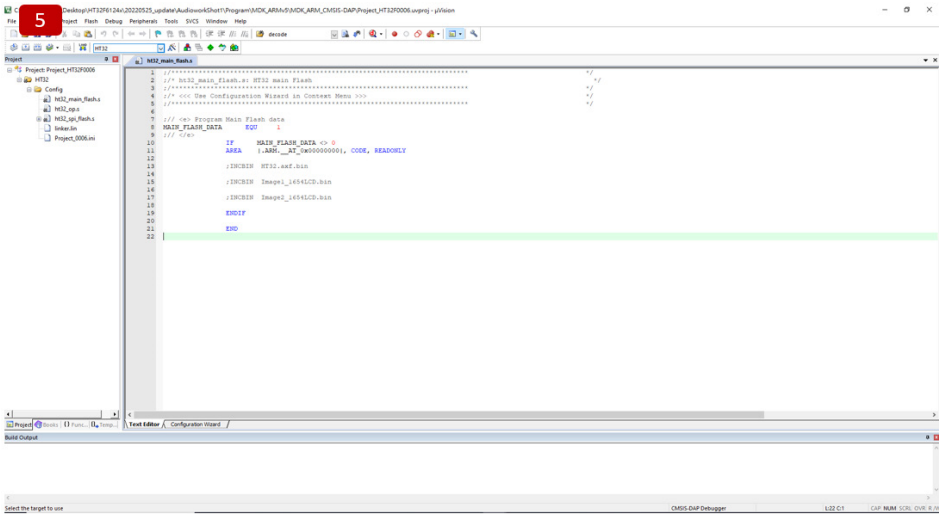
Pin#	Description	Pin#	Description
1	3.3V	2	SWDIO
3	GND	4	SWCLK
5	GND	6	Reserved
7	NC (VCOM_RXD(Note))	8	NC (VCOM_TXD(Note))
9	GND	10	Reset

Note: The Pin7 and Pin8 are VCOM_RXD and VCOM_TXD pins for the e-Link32 Pro, while these two pins are NC pins for the e-Link32.

6.3 Data Flash ROM Programming

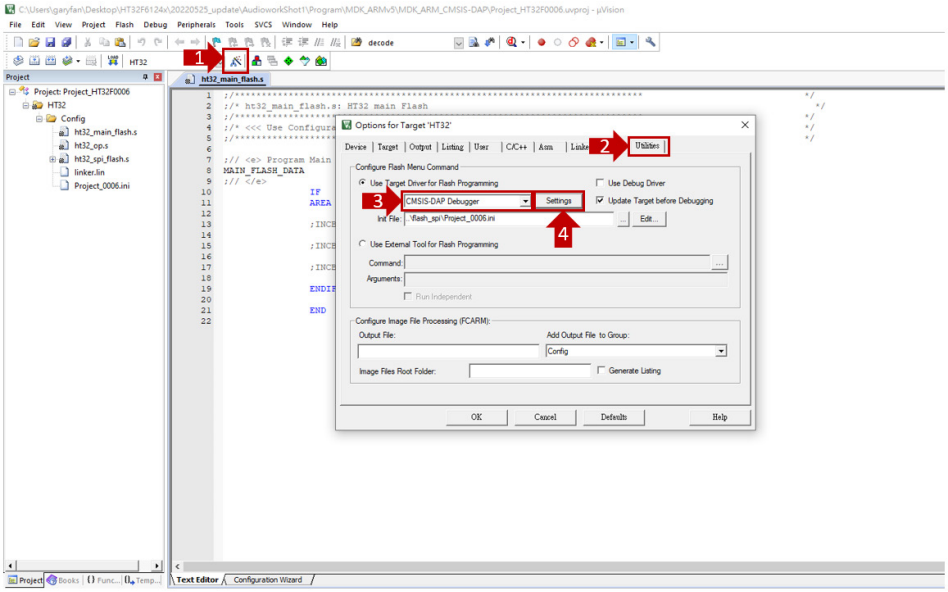
6.3.1 Open a Programming Project

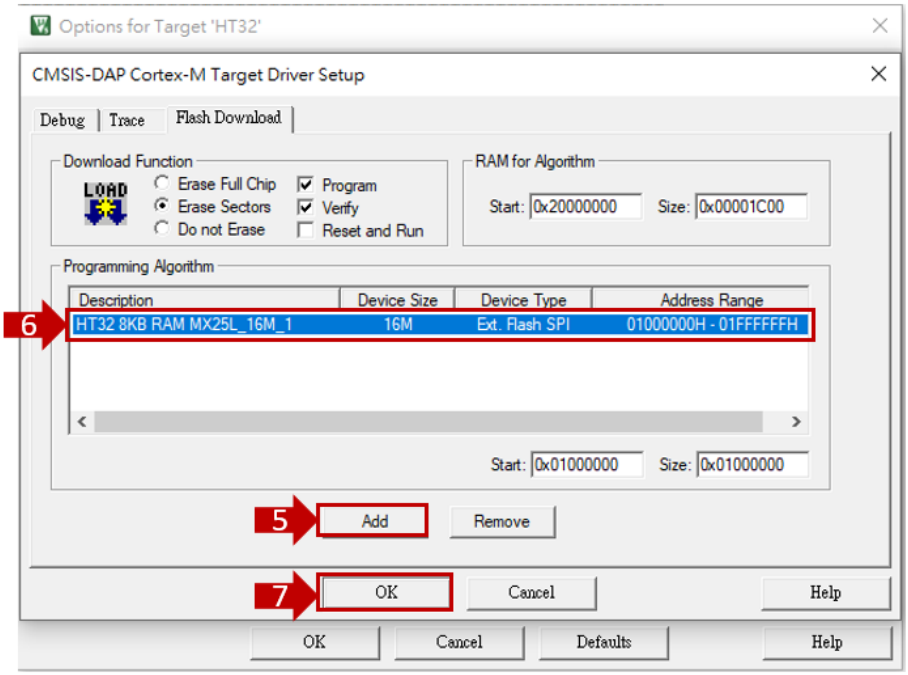


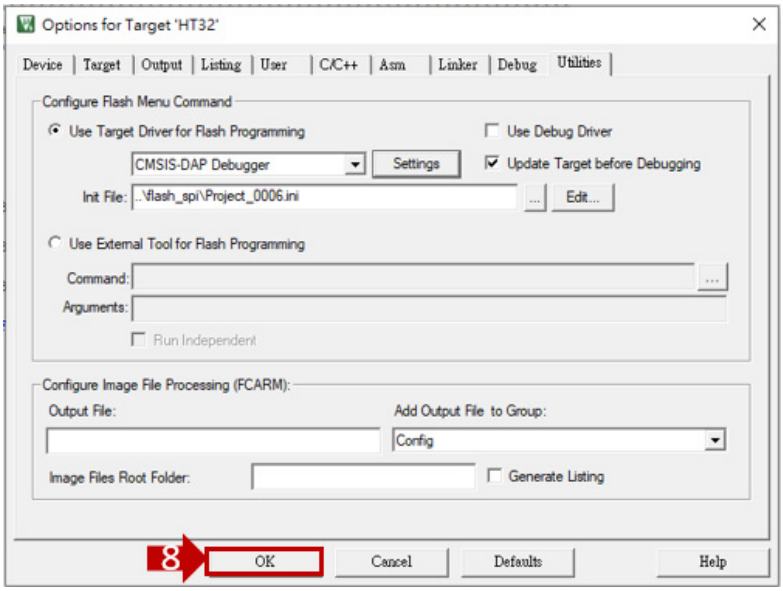


- Step 1: Open "Program" folder in the project folder.
- Step 2: Click "MDK_ARMv5" folder to open it.
- Step 3: Click "MDK_ARM_CMSIS-DAP" folder to open it.
- Step 4: Open a programming project.
  The HT32F0006 and the HT32F61355/HT32F61356/HT32F61357 select the Project_HT32F0006 for programming while the HT32F61244/HT32F61245 select the Project_HT32F61240 for programming.
- Step 5: Enter the Keil main page.

6.3.2 Loader Setting

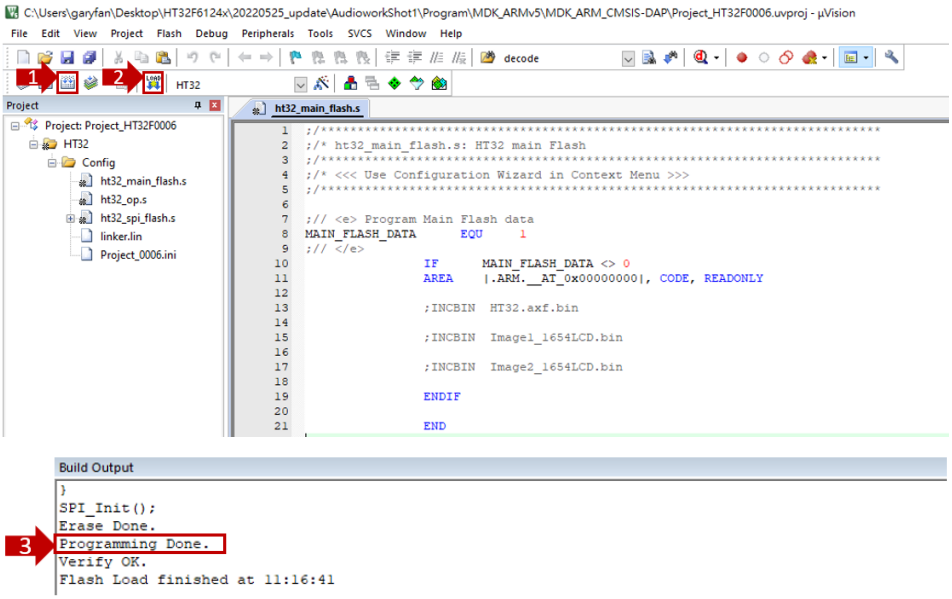






- Step 1: Click "Options for Target 'HT32'".
- Step 2: Click "Utilities".
- Step 3: Click "Use Target Driver for Flash Programming" to select CMSIS-DAP Debugger.
- Step 4: Click "Settings" to set the Loader.
- Step 5: Click "Add" to add the required Loader.
- Step 6: Select the required Loader.
- Step 7: Click 'OK' to complete the setting. Then close the "Settings" dialog box.
- Step 8: Click 'OK' to complete the setting. Then close the "Options for Target 'HT32'" dialog box.

6.3.3 Tone Library Data (.bin) Programming



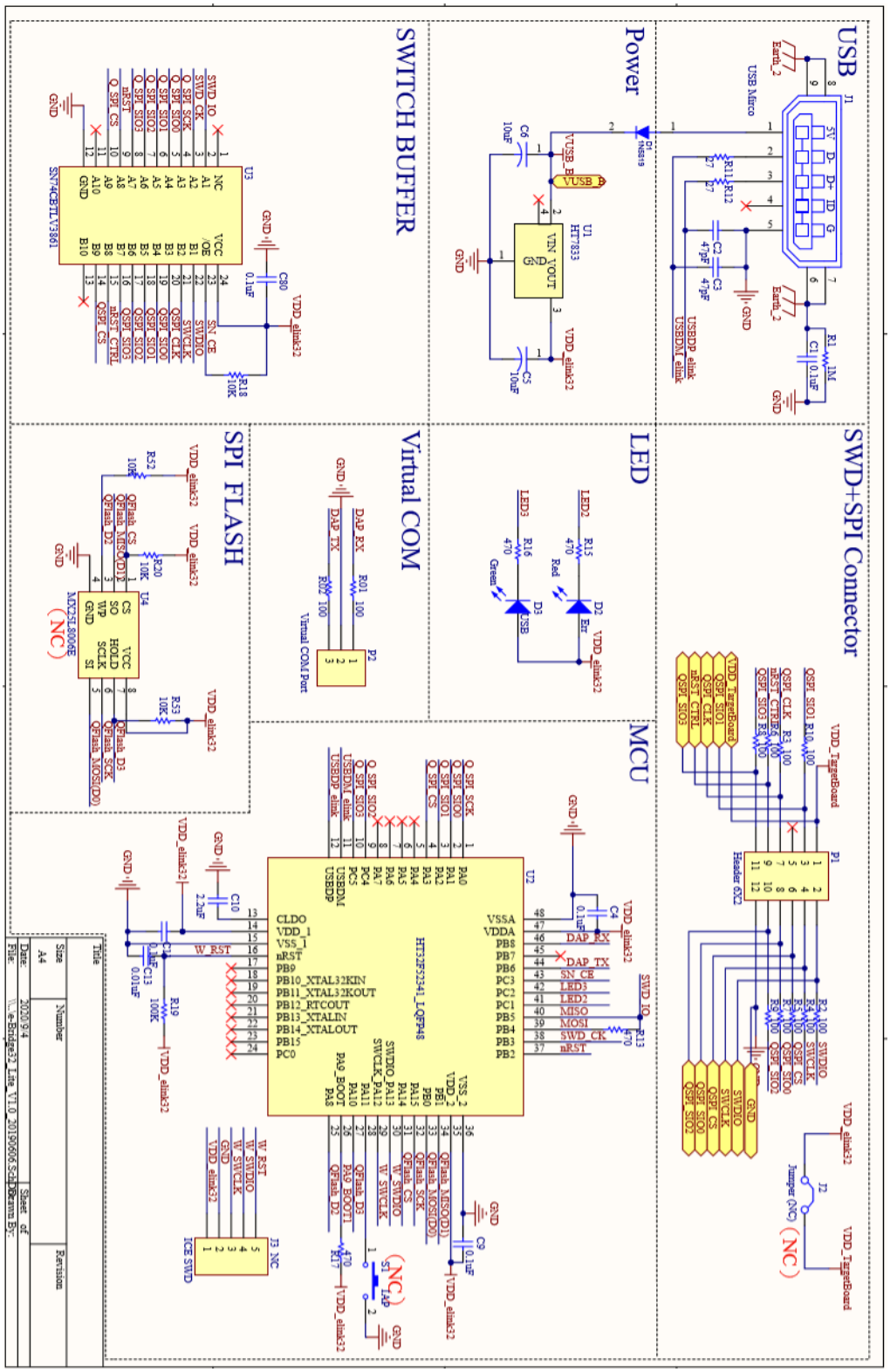
- Step 1: Click "Rebuild" to compile.
- Step 2: Click "Download" to start programming.
- Step 3: Confirm that a "Programming Done" message appears in the Build Output Window, indicating that the programming is successful.

6.4 Development Board Schematic Diagram

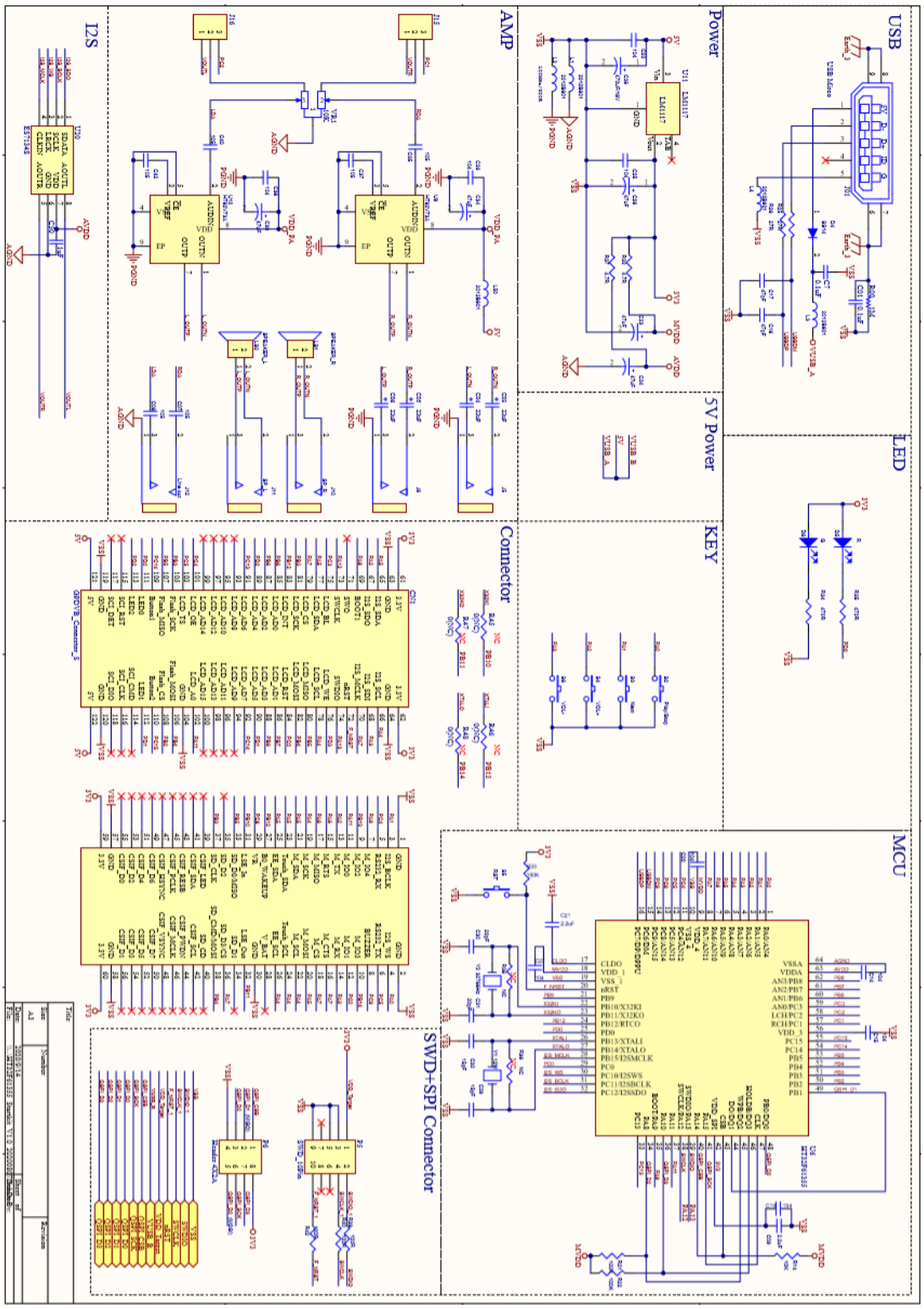
- Development Board ESK32-30615 for HT32F61355
- Development Board ESK32-30616 for HT32F61356
- Development Board ESK32-30617 for HT32F61357/HT32F0006
- Development Board ESK32-30605 for HT32F61244/HT32F61245

Development Board ESK32-30615 for HT32F61355.

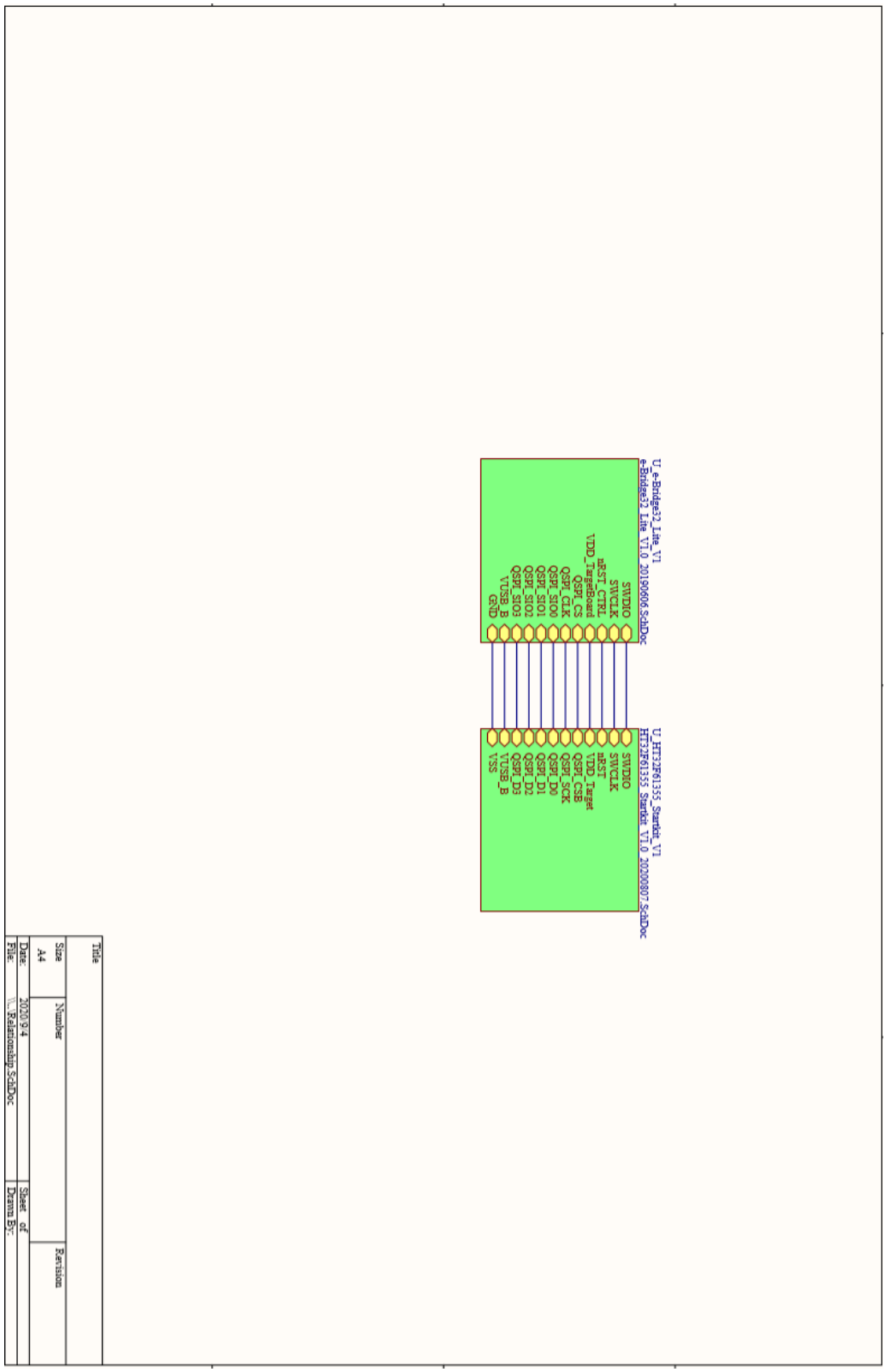
ESK32-30615 (1/3)



ESK32-30615 (2/3)

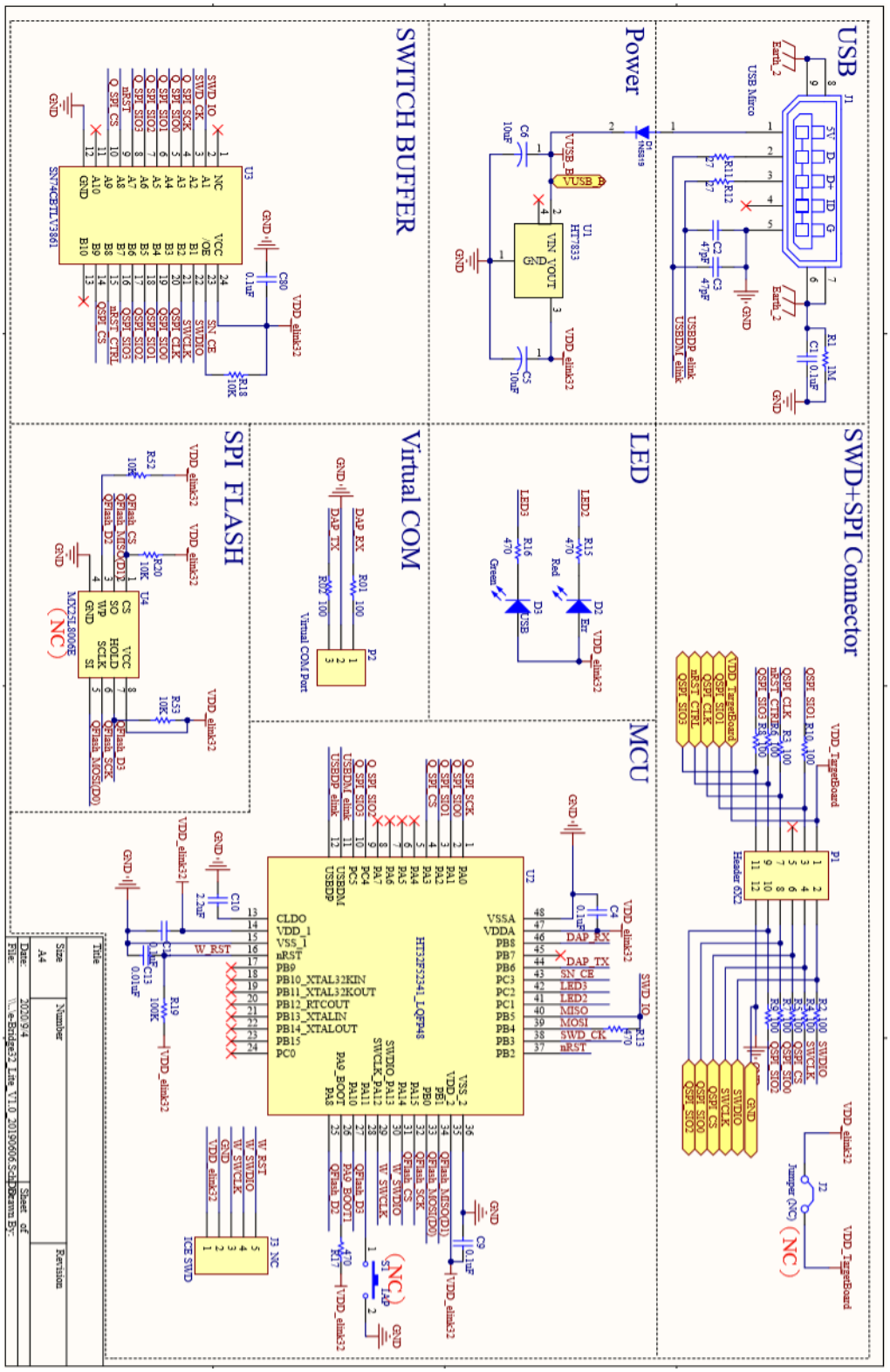


ESK32-30615 (3/3)

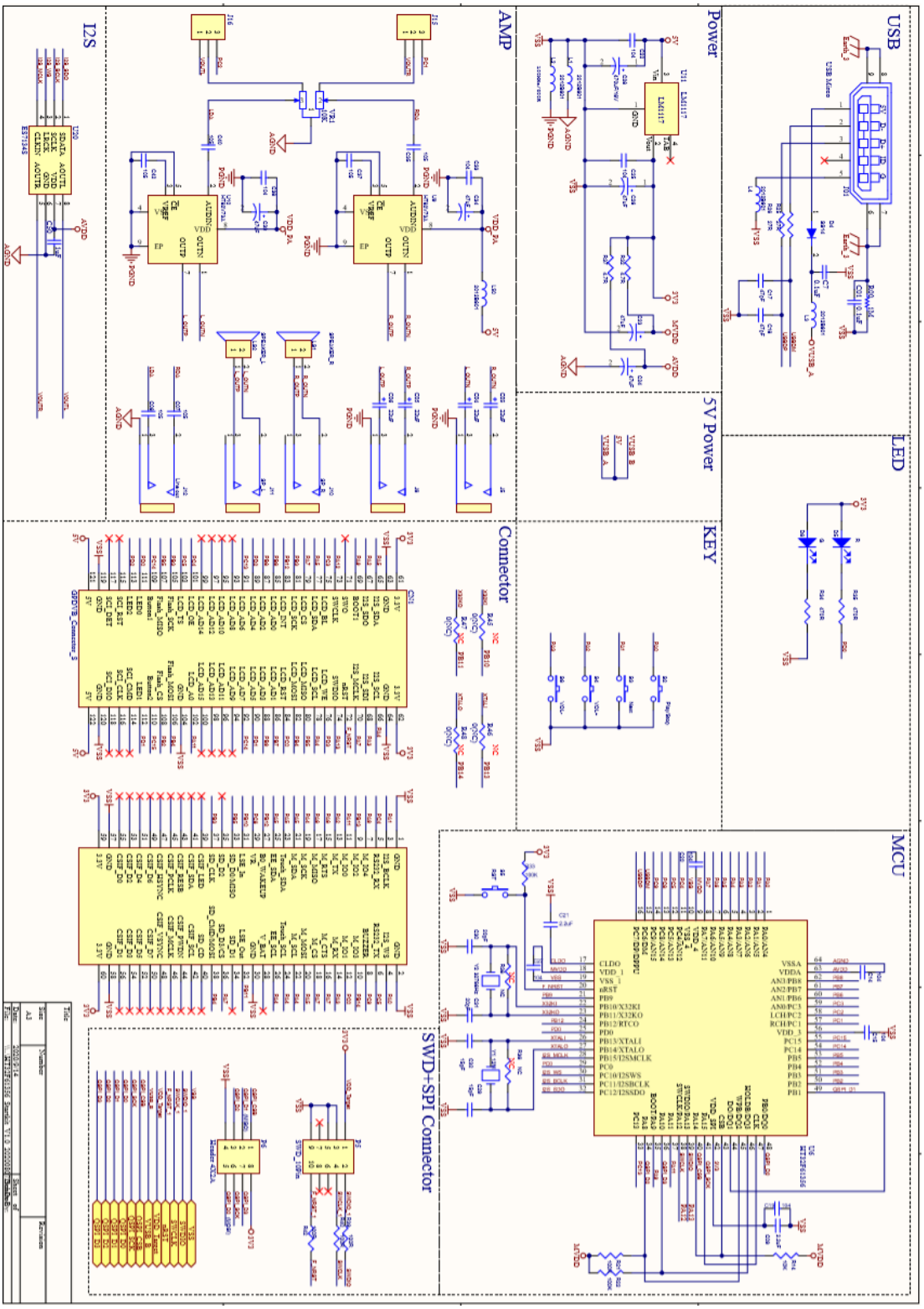


Development Board ESK32-30616 for HT32F61356.

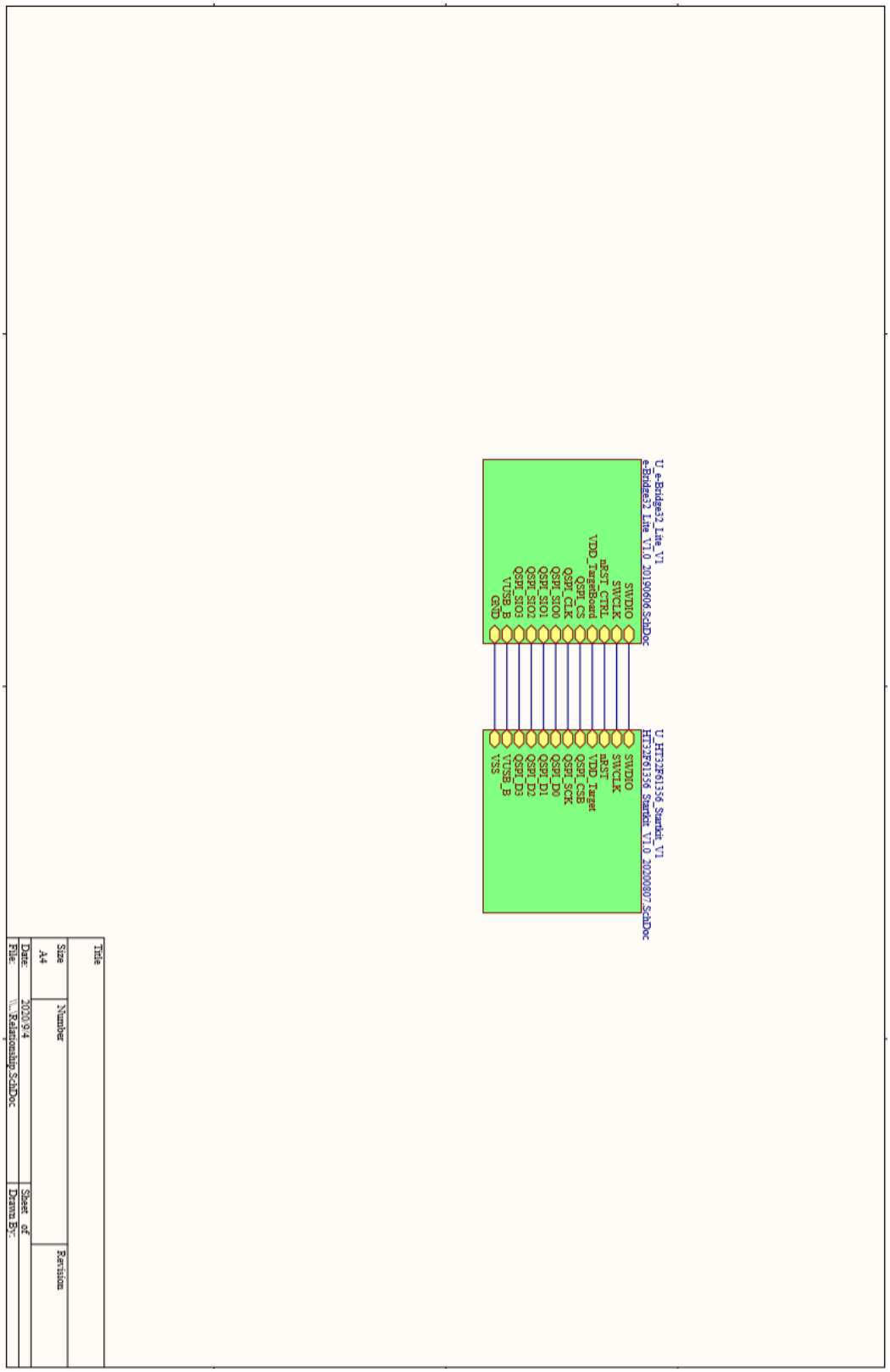
ESK32-30616 (1/3)



ESK32-30616 (2/3)

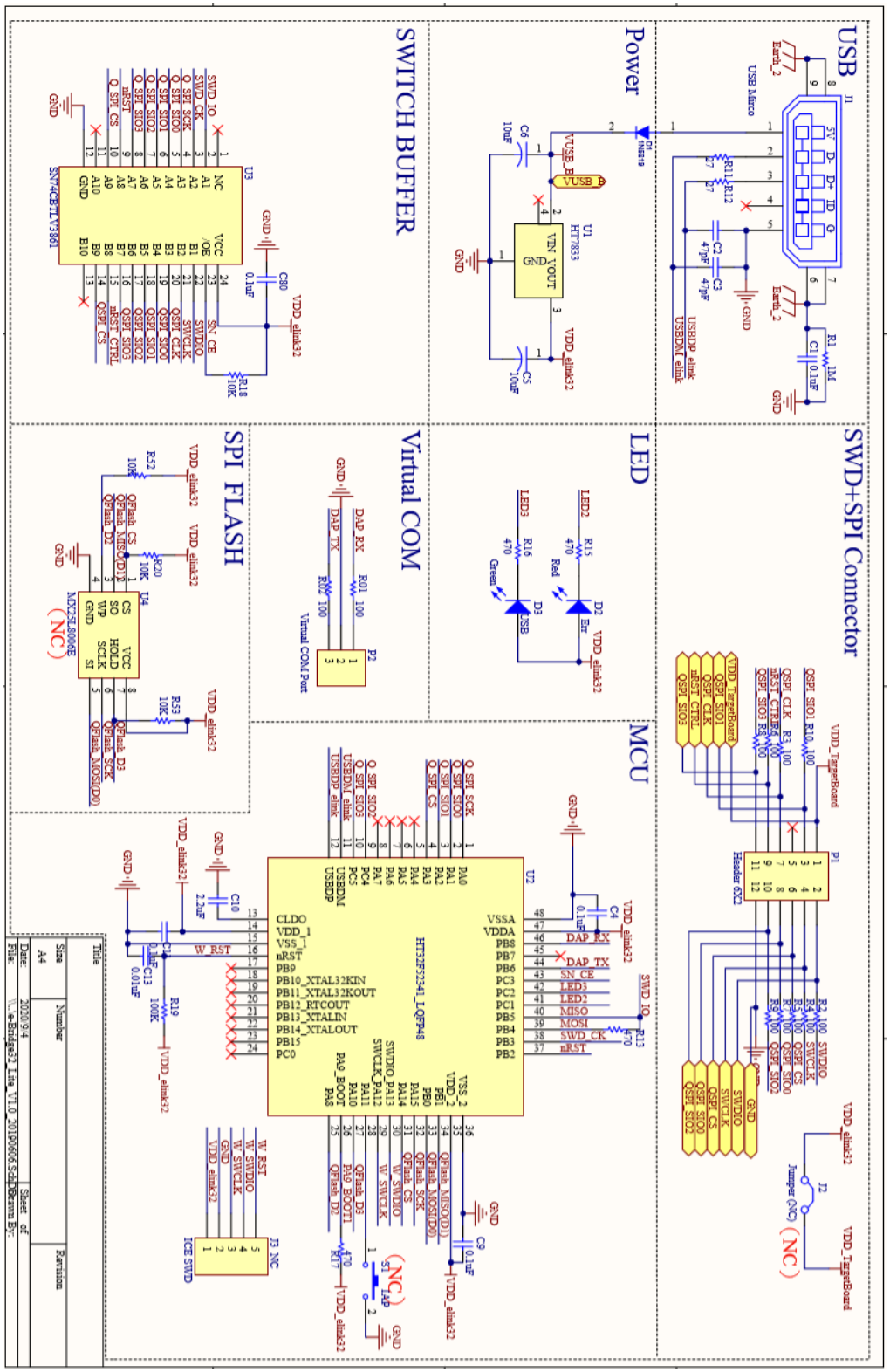


ESK32-30616 (3/3)

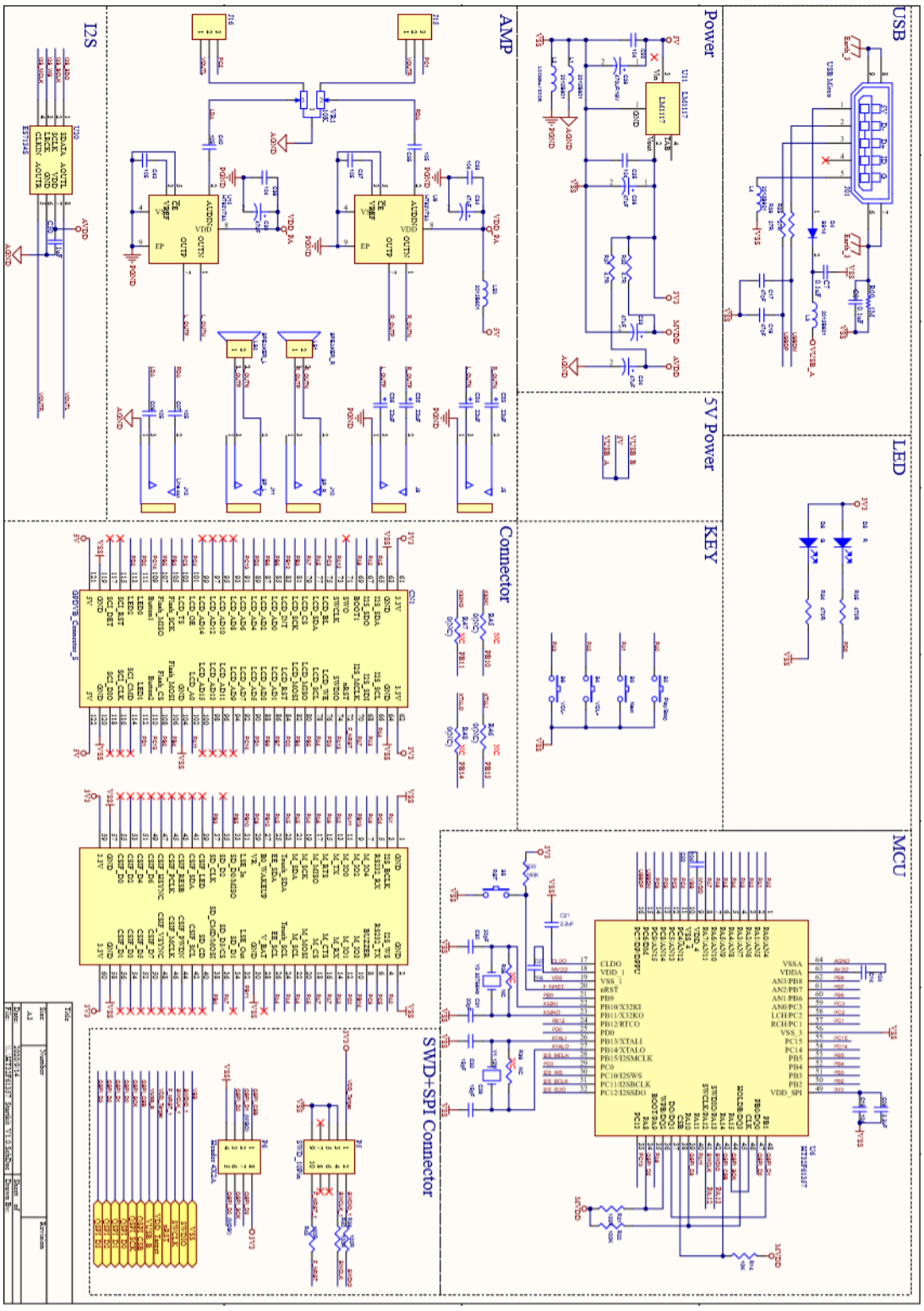


Development Board ESK32-30617 for HT32F61357/HT32F0006.

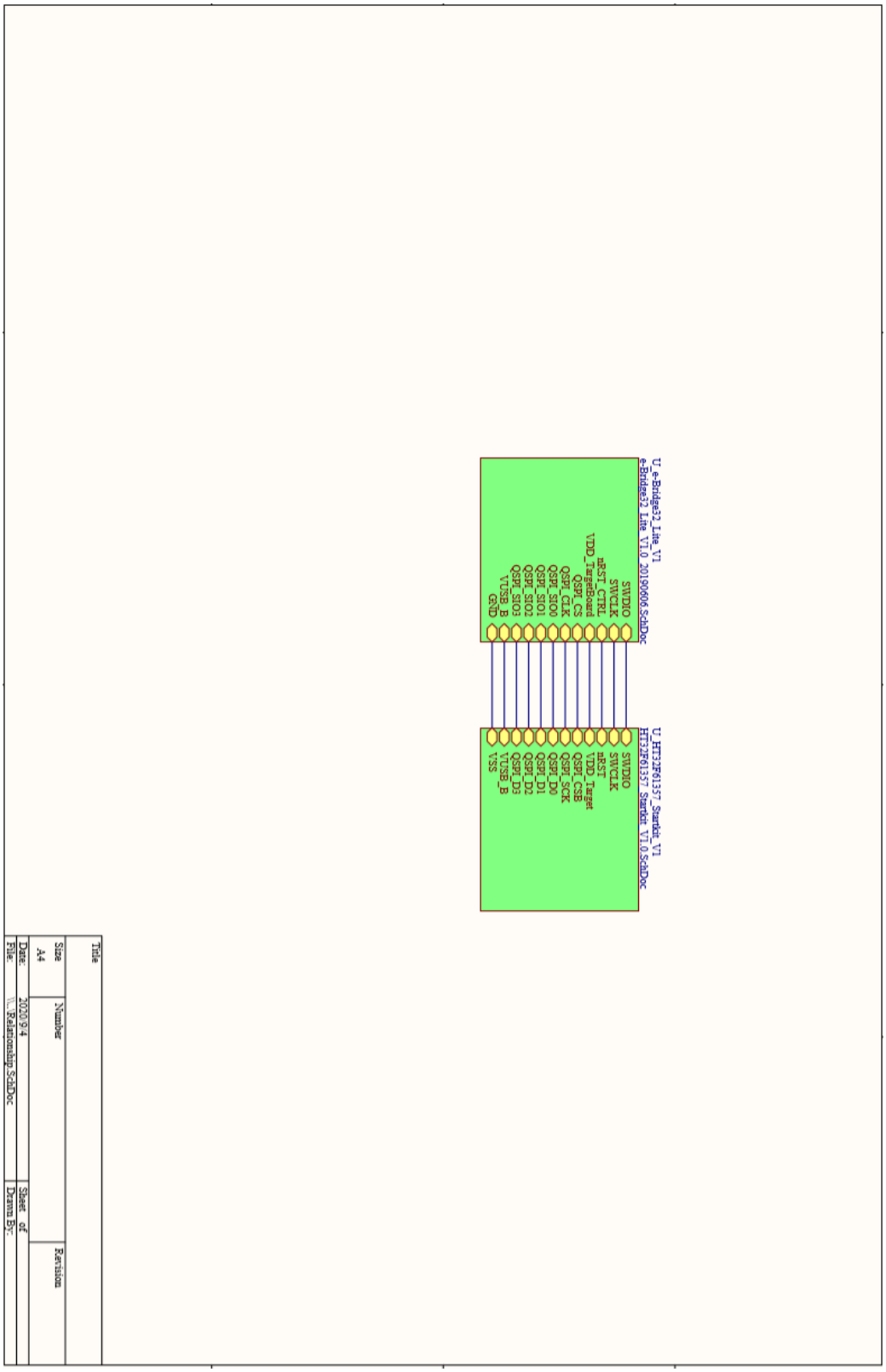
ESK32-30617 (1/3)



ESK32-30617 (2/3)

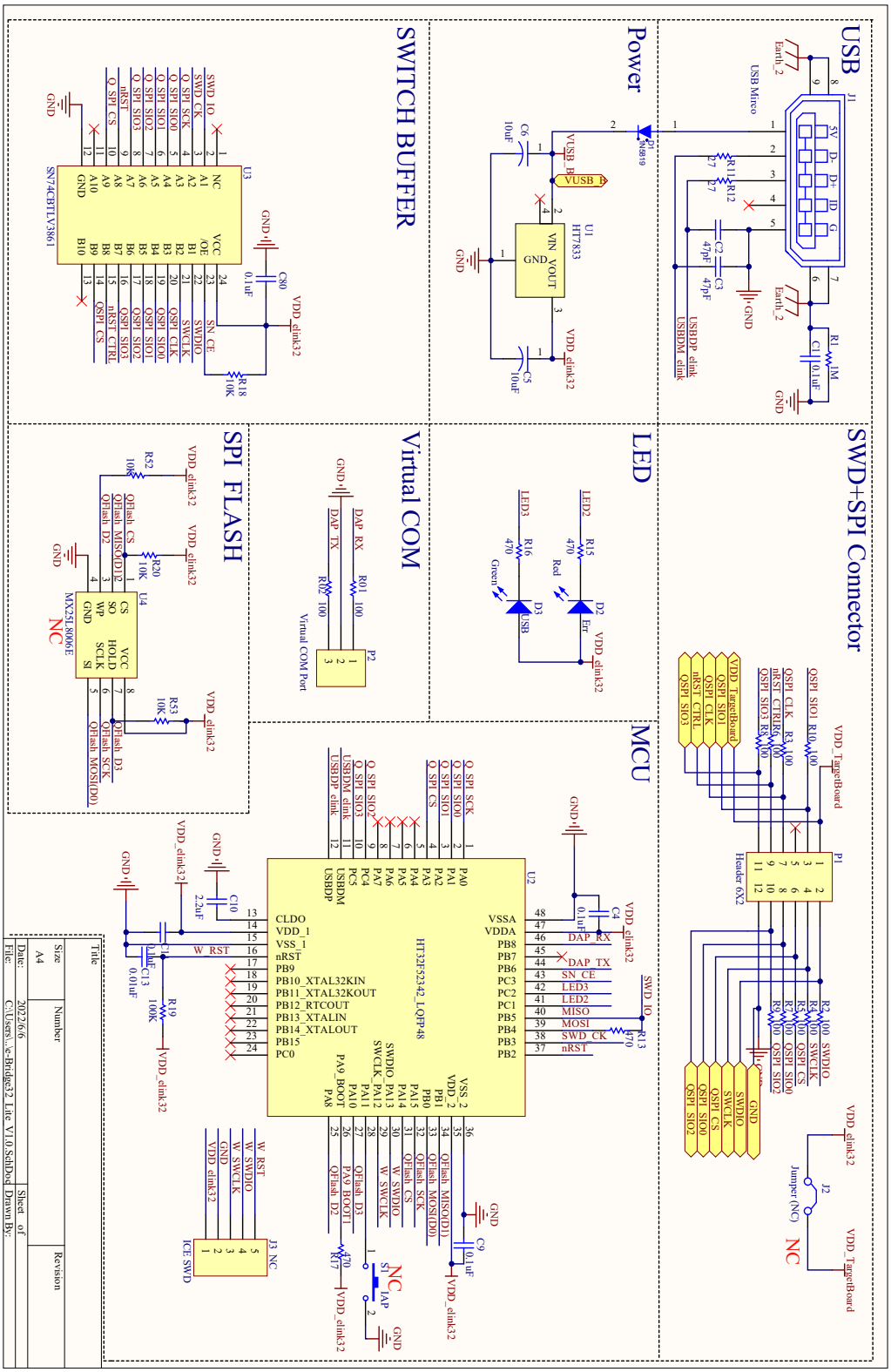


ESK32-30617 (3/3)

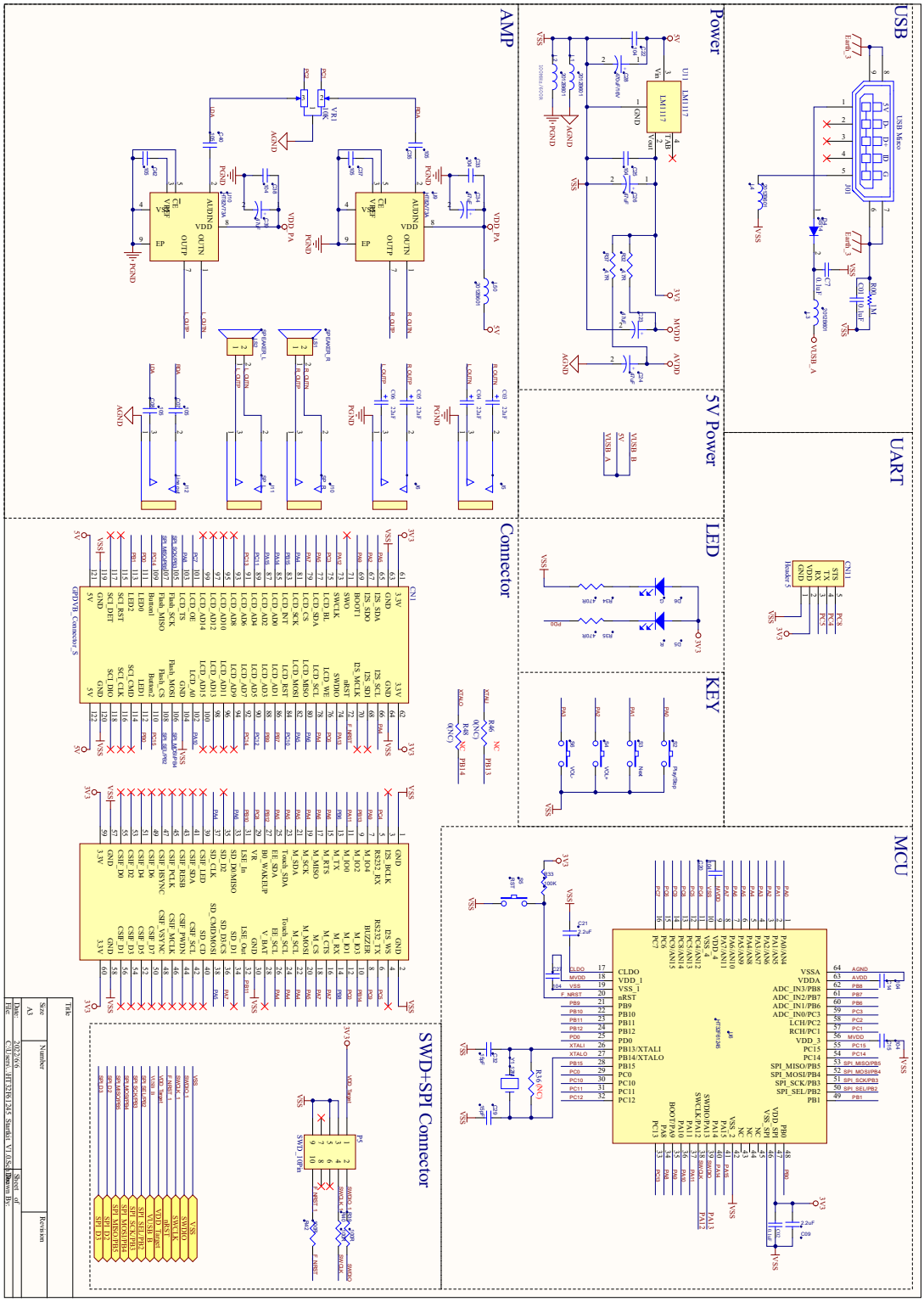


Development Board ESK32-30605 for HT32F61244/HT32F61245

ESK32-30605 (1/3)



ESK32-30605 (2/3)



ESK32-30605 (3/3)