1. Introduction

This document describes the APIs provided by the W569 middleware. These APIs make it simpler to control playback and to drive the LED and motor. While the middleware contains several files, this document focuses on the APIs and how to use them. The playegn.h file includes the function definitions for each of these APIs as well.

The APIs are divided into four sections: volume control, procedure control function, LED and motor control, and other settings.

2. Volume Control

The W569 middleware provides these APIs to control the volume of various outputs.

- Set Equalizer Volume
- Set Headphone Left Volume
- Set Headphone Right Volume
- Set Speaker Volume

All of these APIs use a volume index, which is defined in the following table, to control the volume.

Index	Vol(dB)	Index	Vol(dB)	Index	Vol(dB)	Index	Vol(dB)
00H	Mute	08H	-23	10H	-15	18H	-7
01H	-30	09H	-22	11H	-14	19H	-6
02H	-29	0AH	-21	12H	-13	1AH	-5
03H	-28	0BH	-20	13H	-12	1BH	-4
04H	-27	0CH	-19	14H	-11	1CH	-3
05H	-26	0DH	-18	15H	-10	1DH	-2
06H	-25	0EH	-17	16H	-9	1EH	-1
07H	-24	0FH	-16	17H	-8	1FH	-0

2.1 Set Equalizer Volume

This API changes the equalizer output volume any time.

Prototype	short W569_SetEqualizerVolume(BYTE byVolume)
Description	Set the equalizer output volume
Argument	byVolume: The range is 00h to 1Fh
Return value	=0 Success <0 Error

2.2 Set Headphone Right & Left Volume

This API only works when OP2 & OP3 are set to stereo mode.

Prototype	short W569_SetHeadphoneLeftVolume(BYTE byVol) short W569_SetHeadphoneRightVolume(BYTE byVol)
Description	To set up the stereo right and left output volume
Argument	byVol: The range is 00h to 1Fh.
Return value	=0 Success <0 Error

2.3 Set Speaker Volume

This API controls the volume when the line-in function is in use.

Prototype	short W569_SetSpeakerVolume(BYTE byVol)
Description	Set the speaker output volume
Argument	byVol: The range is 00h to 1Fh
Return value	=0 Success <0 Error

3. Special volume control

We provide some special API functions to let user can make special sound effect. User can use those functions to make your mobile voice changefully.

- Set Speech Pan
- Set Synthesize Volume
- Set Synthesize Speed
- Set Synthesize Pitch Shift

3.1 Set Speech Pan

This function controls the left-right distribution of stereo output. this function is only for when play speech. This distribution can range from 0 (hard left) to 127 (hard right). A value of 64 specifies sound that is centered between left and right.

Prototype	Short W569_SetSpeechPan(short nChannelID, BYTE byMode)
Description	In stereo mode, this sets the right and left channel volume
Argument	nChannelID: Choice channelID you want to setup byMode: The value is 0 to 127 . The balance is 64.
Return value	=0 Success <0 Error

3.2 Set Synthesize Volume

When you play two sounds, you can turn one of them volume down by this function. This function is different to section2 volume control. The section2 volume control is for hardware setting. It will change the total volume. But this is for firmware control, user can change volume which want channel. This is only Winbond has.

Prototype	Short W569_SetSynthesizeVolume(short nChannelID, BYTE byVolume);
Description	Setting sound volume
Argument	nChannelID: Choice channelID you want to setup byVolume: The value is 0(mute) to 1F(max)
Return value	=0 Success <0 Error

3.3 Set Synthesize Speed

You can use this function to increase or decrease midi sound tempo which you want channel. This is only effect to play midi.

Prototype	short W569_SetSynthesizeSpeed (short nChannelID, BYTE bySpeed)
Description	Setting midi tempo
Argument	nChannelID: Choice channelID you want to setup bySpeed: bySpeed = 1 ~ 100 ~ 255 Slowest Normal Fastest
Return value	=0 Success <0 Error

3.4 Set Synthesize Pitch Shift

You can use this function to increase or decrease midi sound pitch which you want channel. This is only effect to play midi.

Prototype	short W569_SetSynthesizePitchShift (short nChannelID, signed char chPitchShift)
Description	Setting midi pitch
Argument	nChannelID: Choice channelID you want to setup chPitchShift: chPitchShift = -24 ~ 0 ~ 24 -2 octaves Normal +2 octaves
Return value	=0 Success <0 Error

4. Procedure control function

Winbond W569 ringtone chips can play 2 MIDI, 1 MIDI and 1 speech, or 2 speech files at the same time. These APIs control playback.

- Play
- Stop
- Pause
- Resume
- Is playing
- Initialize
- Interrupt Handler
- Set Polyphony Mode
- Set FIFO Size
- Call Back Function
- Auto power down

4.1 Play

This API plays MIDI and speech. It performs the following tasks:

1. Decode the midi format.
2. Power up microprocessor and amplifier.
3. Set the LED and Motor mode.
4. Play MIDI or speech sound.

Prototype	short W569_Play(const BYTE* pbyData , DWORD dwDataSize , BYTE byRepeatNum)
Description	Sound Play
Argument	pbyData: pointer to sound data dwDataSize: size of sound MIDI file in bytes byRepeatNum: number of times to repeat playback.
Return value	0 or 1. The return value is the ChannelID used in other APIs. The W569 supports playback on two channels at the same time, so this value is required to control playback.

4.2 Stop

This API function stops playing MIDI or speech sound. It performs the following tasks:

1. Stop MIDI and speech playback
2. Stop LED and Motor .
3. Power down the ringtone chip.

Prototype	Short W569_Stop(short nChannelID)
Description	Stop playing MIDI or speech
Argument	nChannelID: Set what channel ID sound you want to stop
Return value	=0 Success <0 Error

4.3 Pause

This function pauses playback. It does not power down the ringtone chip or clear any registers or memory in the ringtone chip. It just pauses playback.

Prototype	short W569_ Pause(short nChannelID)
Description	Pause playback.
Argument	nChannelID: Set what channelID sound you want to pause.
Return value	=0 Success <0 Error

4.4 Resume

This function resumes playback that has been paused.

Prototype	short W569_Resume(short nChannelID)
Description	Resume playback after pausing.
Argument	nChannelID: Set what channelID sound you want to resume.
Return value	=0 Success <0 Error

4.5 IsPlaying

This function checks whether or not the W569 chip is playing sound on one of its channels.

Prototype	Short W569_IsPlaying(short nChannelID)
Description	Check whether or not the chip is playing sound on nChannelID.
Argument	nChannelID: Set what sound you want to check it state.
Return value	=0 not playing =1 playing

4.6 Initialize

This API must be called before other APIs can be used. This function initializes the middleware and W569 hardware.

Prototype	short W569_Initialize(void)
Description	Initial the W569 ringtone chip
Argument	
Return value	=0 Success <0 Error

4.7 Interrupt Handler

This API handles all W569 interrupts—for example, when the FIFO is empty, timer interrupt, etc.

Prototype	short W569_ InterruptHandler (void)
Description	Handle W569 interrupts
Argument	
Return value	=0 Success <0 Error

4.8 Set Polyphony Mode

This API controls the number of polyphones used to play sound. The W56964 is a 64- polyphony sound player, and it can use 32 or 64 polyphones to play sound. The W56940 can use 40 or 32 polyphones. The default value in both the W56940 and the W56964 is 32 polyphones.

Prototype	short W569_SetPolyphonyMode(BYTE byPolyphonyMode)

Description	Set up the polyphone mode to play sound.
Argument	byPolyphoneMode: 32 ,40 or 64
Return value	=0 Success <0 Error

4.9 Set FiFo size

User can change the fifo size by this API function. But this FiFo size setting must in power down mode. When you play two sounds, our chip will use two FiFo to play sound. If you only want to play one sound, user can use this function to enlarge the FiFo size. By use this the frequency of interrupt will became small.

Prototype	Short W569_SetFIFOSize(WORD wS_FIFOSize, WORD wP_FIFOSize, WORD wS_FIFOThreshold, WORD wP_FIFOThreshold);
Description	Set up the FIFO size
Argument	* wS_FIFOSize = 256 or 512 * wP_FIFOSize = 256 or 384
Return value	=0 Success <0 Error

The FiFo size setting can be below table. If the FIFO threshold setting is too small, the interrupt signal will frequent. If the FiFo threshold setting is bigger, the baseband CPU transfer data to FiFo will take more time. General we always set the FiFo threshold half of FIFo size. User must to pay attention if FiFo size setting is (512 ,0), it just can play only one midi or one speech sound at same time.

FiFO size setting

mode	wS_FIFOSize	wP_FIFOSize
1	512	0
2	256	256
3	256	384

FiFo threshold setting

If wS_FIFOSize=256	If wS_FIFOSize=512	If wP_FIFOSize=256	If wP_FIFOSize=384
32	64	32	64
64	128	64	128
96	192	96	192

128	256	128	256
160	320	160	320
192	384	192	Reserved
224	448	224	Reserved

4.10 Call back Function

When sound played and want to know end or not, general we will use the API function W569_IsPlaying() to check. Or user can use the call back function to check the sound playing end or not. You can set up what to do at what channel when sound end or sound repead. User can add the call back function in your program like follow.

```
void CPUHost_W569PlaybackCallback(short nChannel, BYTE byStatus)
{
    switch(byStatus)
    {
    case MW_PLAYBACK_CALLBACK_REPEAT:
        printf("Channel %d Repeat\n", nChannel);
        break;
    case MW_PLAYBACK_CALLBACK_END:
        printf("Channel %d Played End\n", nChannel);
        break;
    }
}
```

4.11 Auto power down

General when we get W569_Stop API or midi sound play end, the ringtone chip will stop and get into power down mode. If you just want to stop sound and not get ringtone chip into power down mode, user can use the API to setup. The default is true to auto power down.

Prototype	void W569_AutoPowerDown(BOOL bAuto)
Description	when sound play end to set power down or not
Argument	bAuto : True False

5. LED and Motor Control

The W569 provides API functions to control the LED and Motor. In addition, the W569 middleware provides API functions compatible with the Yamaha chip so that it is not necessary to rewrite baseband programs to drive Winbond's polyphone chip instead. **Please note that programs should not mix Yamaha-compatible APIs and Winbond APIs. Programs should use only one set of APIs.**

A. These functions are compatible with the Yamaha chip.

- LED Control Source
- Set LED Blinking
- Set LED RGB
- Set LED Pattern
- Motor Control Source
- Set Motor Blinking
- Set Motor Level
- Set Motor Pattern

B. These functions are provided by Winbond

- Set LED On Off
- Set Motor On Off
- Set LED Mode
- Set Motor Mode

5.1 LED Control Source

This API selects how the LED is controlled (forced control, sequence synchronization, MIDI event, or pattern) and also turns off the LED.

Prototype	short W569_LEDControlSource(BYTE bySource)		
Description	Set the LED control source		
Argument	bySource	0	Off
		1	Forced control
		2	Sequence synchronization
		3	MIDI Event
		4	Pattern
Return value	=0 Success <0 Error		

5.2 Set LED Blinking

This API sets the frequency at which the LED blinks.

Prototype	short W569_SetLEDBlinking(BYTE byFreqIndex)	
Description	Set frequency for blinking LED	
Argument	byFreqIndex	0 No blinking 1 14.75Hz 2 9.83Hz 3 7.37Hz 4 4.92Hz 5 3.28Hz
Return value	=0 Success <0 Error	

5.3 Set LED RGB

This function can only be used when the LED Control Source is "2" (sequence synchronization). This API sets the R, G, and B colors to get the desired color.

Prototype	short W569_SetLEDRGB(BYTE byRed , BYTE byGreen , BYTE byBlue)
Description	Set R, G, and B levels of brightness
Argument	byRed, : Set red LED brightness (0 to 255) byGreen: Set green LED brightness(0 to 255) byBlue : Set blue LED brightness (0 to 255)
Return value	=0 Success <0 Error

5.4 SetLEDPattern

This function can only be used when the LED Control Source is “4” (pattern). This API sets the pattern the LED will follow.

Prototype	void W569_SetLEDPattern(BYTE* pbyPattern, BYTE byMorphDeltaTime)
Description	Set the LED bright pattern
Argument	pbyPattern : point to LED pattern byMorphDeltaTime : Morphing color time in msec
Return value	

The LED pattern (pointed to by pbyPattern) must follow the format below.

LED pattern format:

((LED phase)+) 0x00 0x00

LED phase includes the timing and R, G and B value.

Byte No	Description
#1	Low byte of delta time
#2	High byte of delta time
#3	R value
#4	G value
#5	B value

The timing setting (delta time) has two vectors: the first is low-byte, the second high-byte. The unit of measure for delta time is milliseconds. The R/G/B LED brightness has a maximum value of 255, and 0 turns off the LED.

For example, the LED pattern is red for 500 ms, green for 500 ms, and blue for 500 ms. The turn on time, 500 ms, is 0x01F4 in hex. The brightnesses are set to 255, and the (R,G,B) value for red is (0xFF, 0x00, 0x00); for green (0x00, 0xFF, 0x00); and for blue (0x00, 0x00, 0xFF). At the end of the pattern, the two bytes (0x00, 0x00) are added to indicate the end of the pattern. As a result, the pattern is stored as follows:

```
BYTE abyLEDPlayPattern [] =
    {
```

```
    0xF4, 0x01,          /* 500 ms */
    0xFF, 0x00, 0x00,    /* (R,G,B) = (255, 0, 0) */
    0xF4, 0x01,          /* 500 ms */
    0x00, 0xFF, 0x00,    /* (R,G,B) = (0, 255, 0) */
    0xF4, 0x01,          /* 500 ms */
    0x00, 0x00, 0xFF,    /* (R,G,B) = (0, 0, 255) */
    0x00, 0x00
};
```

5.5 Motor Control Source

This API selects how the motor is controlled (forced control, sequence synchronization, MIDI event, or pattern) and also turns off the motor.

Prototype	Short W569_MotorControlSource(BYTE bySource)		
Description	Set the Motor control source		
Argument	bySource	0	Off
		1	Forced control
		2	Sequence synchronization
		3	MIDI Event
		4	Pattern
Return value	=0 Success <0 Error		

5.6 Set Motor Blinking

This API sets the frequency at which the motor blinks.

Prototype	short W569_SetMotorBlinking(BYTE byFreqIndex)		
Description	Set frequency index for motor		
Argument	byFreqIndex	0	No blinking
		1	2.27Hz
		2	1.97Hz

		3 1.48Hz 4 0.98Hz 5 0.48Hz
Return value	=0 Success <0 Error	

5.7 Set Motor Level

This function can only be used when the Motor Control Source is "2" (sequence synchronization). When sound is played in sequence mode, this API sets the motor level and turns the motor on and off with the channel notes.

Prototype	short W569_SetMotorLevel (BYTE byLevel)
Description	Set PWM level of motor
Argument	byLevel : Level of vibration(0~255). If equal to 0, the motor is turned off.
Return value	=0 Success <0 Error

5.8 Set Motor Pattern

This API sets the pattern the motor will follow.

Prototype	void W569_SetMotorPattern(BYTE* pbyPattern , BYTE byMorphDeltaTime)
Description	Setting the motor bright pattern
Argument	pbyPattern : point to motor pattern byMorphDeltaTime : Morphing motor time in msec
Return value	

The Motor pattern (pointed to by pbyPattern) must follow the format below.

Motor pattern format:

((Motor phase)+) 0x00 0x00

Motor phase includes the timing and the vibration level.

Byte No	Description
#1	Low byte of delta time
#2	High byte of delta time
#3	Vibration level

The timing setting (delta time) has two vectors:, the first is low-byte, the second high-byte. The unit of measure for delta time is milliseconds. The Motor level has a maximum value of 255, and 0 turn off the motor.

For example, the motor pattern is on for 500 ms, off for 500 ms, and on for 500 ms. The turn on time, 500 ms, is 0x01F4 in hex.. The motor level is the maximum 255 (0xFF). At the end of the pattern, the two byptes (0x00, 0x00) are added. As a result, the pattern is stored as follows:

```
BYTE abyMotorPlayPattern[] =
    {
        0xF4, 0x01,         /* 500 ms */
        0xFF,     /* (Motor) = (255) */
        0xF4, 0x01,         /* 500 ms */
        0x00,     /* (Motor) = (0) */
        0xF4, 0x01,         /* 500 ms */
        0xFF,     /* (Motor) = (255) */
        0x00, 0x00
    };
```

5.9 Set LED On Off

This function can only be used when the LED mode is 1, and it controls the LED even when the ringtone chip is in power down mode. This API controls the LED to get the desired color (R,G,B).

Prototype	short W569_SetLEDOnOff(BYTE byRed , BYTE byGreen , BYTE byBlue)
Description	Set R, G, and B levels of brightness
Argument	byRed, : Set red LED brightness (0 to 255) byGreen: Set green LED brightness(0 to 255) byBlue : Set blue LED brightness (0 to 255)
Return value	=0 Success <0 Error

5.10 Set Motor On Off

This function can only be used when the Motor mode is 1, and it controls the motor even when the ringtone chip is in power down mode. This API controls the motor PWM level.

Prototype	short W569_SetMotorOnOff(BYTE byLevel)
Description	Set PWM level of motor
Argument	byLevel : Level of vibration(0~255). If equal to 0, motor is turned off.
Return value	=0 Success <0 Error

5.11 Set LED Mode

This API controls the mode in which the LED operates.

Prototype	short W569_SetLEDMode(BYTE byMode, BYTE byType, BYTE byChannel, BYTE byFreqIndex, BYTE* pbyParam)
Description	Set LED Mode
Argument	byMode: LED mode bytype: Type setting for each mode byChannel: Synchronize midi channel byFreqIndex: Frequency index for blinking LED pbyParam: Pointer to additional parameter
Return value	=0 Success <0 Error

The values of LED mode, type, and channel are described in the following table:

byMode	byType	byChannel	pbyParam	Description

0	NA	NA	NA	Turn off LED	
1	NA	NA	NA	Directly control LED	
2	0	0 ~ 15	NA	Synchronize with "Note On/Off" and "LED On/Off" events. The default is LED On.	White color
	1				Brightness of white color is synchronized with note velocity
	2				Automatically change LED colors
	3				Automatically change LED color and the brightness is synchronized with velocity
3	0	0 ~ 15	NA	Synchronize with only "LED On/Off" events	White color
	1				Brightness of white color is synchronized with note velocity
	2				Automatically change LED colors
	3				Automatically change LED color and the brightness is synchronized with velocity
4	0		Pointer to LED pattern		LED is controlled according to LED pattern
	>0				LED is controlled according to LED pattern and the color transition is morphing. byType is the delta time for each transition.

The LED pattern (pointed to by pbyParam) follows the same format described in W569_SetLEDPattern above.

The frequency index for LED blinking is the same as described in W569_SetLEDBlinking above and is copied into the table below.

byFreqIndex	Description

0	No blinking
1	Blinking: 14.75 Hz
2	Blinking: 9.83Hz
3	Blinking: 7.37Hz
4	Blinking: 4.92Hz
5	Blinking: 3.28Hz

5.12 Set Motor Mode

This API controls the mode in which the motor operates.

Prototype	short W569_SetMotorMode(BYTE byMode , BYTE byType , BYTE byChannel , BYTE byFreqIndex , BYTE* pbyParam)
Description	Set Motor Mode
Argument	byMode: Motor mode bytype: Type setting for each mode byChannel: Synchronize midi channel byFreqIndex: Frequency index for blinking motor pbyParam: Pointer to additional parameter
Return value	=0 Success <0 Error

The values of Motor mode, tyhp, and channel are described in the following table:

byMode	byType	byChannel	pbyParam	Description	
0	NA	NA	NA	Turn off Motor	
1	NA	NA	NA	Directly control Motor	
2	0	0 ~ 15	NA	Synchronize with	Maximum vibration level

	1			"Note On/Off" and "Motor On/Off" events. The default is Motor On.	Vibration level is synchronized with note velocity
3	0	0 ~ 15	NA	Synchronize with only "Motor On/Off" events	Maximum vibration level
	1				Vibration level is synchronized with note velocity
4	0		Pointer to Motor pattern		Vibration level is controlled according to Motor pattern
	>0				Vibration level is controlled according to Motor pattern and the level transition is morphing. byType is the delta time for each transition.

The Motor pattern (pointed to by pbyParam) follows the same format described in W569_SetMotorPattern above.

The frequency index for Motor blinking is the same as described in W569_SetMotorBlinking above and is copied into the table below.

byFreqIndex	Description
0	No blinking
1	Blinking: 2.27Hz
2	Blinking: 1.97Hz
3	Blinking: 1.48Hz
4	Blinking: 0.98Hz
5	Blinking: 0.48Hz

6. Other settings

Users must setup some parameters based on their system environment. These parameters must be set correctly to play sound accurately. Unless stated otherwise, these parameters are maintained in the MWDefine.h file in the middleware.

The parameters are described in the following sub-sections:

- PLL
- Input Clock
- Clock Type
- Center Voltage
- Headphone Mode Settings
- Enable Louder Speech
- Disable Louder Speech
- Memory locate
- LED & Motor high or low active setting

6.1 PLL

The input clock may be different in each system, so the PLL value must be set to get the correct pitch when sound is played. If the PLL value is not set properly, the sound pitch and speed will be lost. Winbond provides the PLLOUT tool to compute the correct PLL_M and PLL_N value. After computing these values (N,M) ,the following parameters should be set in the MWDefine.h file.

40&64 polyphony	#define W5692_PLL_ADJUST_N N_value #define W5692_PLL_ADJUST_M M_value

6.2 Input Clock

The input clock may be different in each system. This parameter is the input clock frequency, which is required to play sound with the correct tempo.

Prototype	#define W56964_CLKI value
Description	This setup is required for correct tempo when sound is played.
Value	This value depends on your clock input. If you clock input is 15MHz , you must modify 15000000.

6.3 Clock Type

The output voltage level is different for different kinds of clock generators. This parameter indicates the clock input type in VDCK.

Prototype	#define VDCK value (default 0)
Description	Input mode selection for CLKI pin
Value	0: CMOS input 1: Oscillator mode.

6.4 Center Voltage

This value sets the center voltage in the amplifier to avoid signal distortion. This value is set in the MWConfig.h file.

Prototype	#define SPK_VOL_REG_VDS value (default 0)			
Description	Set the center voltage of speaker amplifier when it is on			
Argument	Value	Center voltage(v)	VDD	SPVDD
	0	0.6xVDD	3.0V	3.6V
	1	0.5xVDD	3.0V	3.0V
	2	0.67xVDD	2.7V	3.6V
	3	0.72xVDD	2.5V	3.6V

6.5 Headphone Mode Settings

In the W56940 and W56964, the headphones use the same two output pins as LED2/3. The parameter DOUT_LED indicates whether these pins support headphones or these pins support LEDs.

Prototype	#define DOUT_LED value (default 1)
Description	select digital output port or headphone output for pins 10 and 11
Value	0: headphone output 1: digital_output_port (LED)

If these pins support headphones, the following parameter selects stereo or mono mode.

Prototype	#define HEADPHONE_MONO value (default 0)
Description	set headphone to be mono or stereo
Value	0: stereo 1: mono

6.6 Louder speech

In W56940 and W56964, user can use API function to increase speech volume. But when you use this function, our chip only can play one speech at same time. User can set the parameter of MWDefine.h.

Prototype	#define SUPPORT_LOUDER_SPEECH value (default 0)
Description	To open the louder speech function.
Value	0: close louder speech 1: open louder speech

By fallow example will show how to use the louder speech function.

```
W569_EnableLouderSpeech();
ID=W569_Play(sound5,sizeof(100000),1);
while(W569_IsPlaying(ID))
……
W569_Stop(ID);
W569_DisableLouderSpeech();
```

6.7 Memory locate

If your system doesn't support the MALLOC command to distribution the memory, you can change the setting in MWDefine.h of middleware.

Prototype	#define NO_MALLOC_FREE value (default is 0)
Description	To setting the malloc
Value	0: use malloc 1: not use malloc

6.8 LED & Motor high or low active setting

This is for different application. User can set the LED and motor high or low active output. you can change the setting in MWStd.h. The detail you can refer to Application Note 002 for vibrator and Application Note 003 for LED. The default is high active.

Prototype	#define LED_LOW_ACTIVE value (default is 0) #define MOTOR_LOW_ACTIVE value (default is 0)
Description	To set high or low active
Value	0: high active 1: low active

7. Revision History

Revision	Date	Modifications
A0	May 2005	• Initial release.