Winbond
Electronics Corp.

Vibrator Application Note

1. Introduction

This application note explains how to control the W569x vibrator. First, it introduces the five control modes, and then it provides program examples.

2. Control Modes

The W569x offers five modes for motor control. The mode is selected using the API function W569_MotorControlSource.

W569_MotorControlSource (BYTE bySource)

bySource =   0   Off

1   Forced control

2   Sequence synchronization

3   MIDI Event

4   Pattern

Each mode is described in more detail in the following sub-sections.

2.1 Mode 0

Mode 0 turns off the vibrator and powers down the motor control circuit. Usually, the W569_Stop command leaves one register open to control the vibrator, but Mode 0 powers down this port circuit.

2.2 Mode 1

Mode 1 is forced control, which forces the vibrator to run at a certain speed. The API function W569_SetMotorOnOff(value) controls the level of vibration. The "value" can vary from 0 to 255; 0 turns off the vibrator. For example,

W569_MotorControlSource(1)

W569_SetMotorOnOff(0)

………………..

W569_SetMotorOnOff(255)

Once turned on, the vibrator does not stop until the value is set to zero or the control mode changes.

Forced control is also effective when the IC is powered down, but it behaves a little differently. In power-down mode, it is possible to turn the vibrator on and off but not to control the level of vibration. Every value greater than zero (i.e., 1 ~ 255) only turns on the vibrator, and zero turns it off.

2.3 Mode 2

Mode 2 is sequence mode. In sequence mode, the vibrator turns on and off synchronized with the channel notes. This is illustrated below.
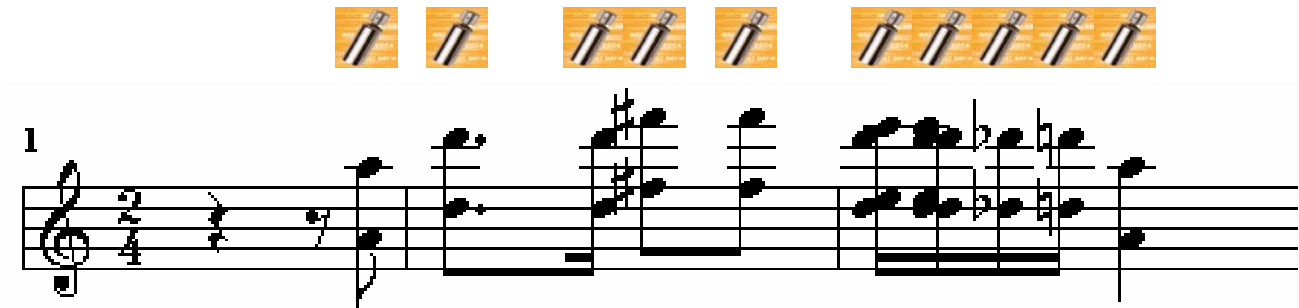
The API function W569_SetMotorLevel(value) controls the level of vibration. The "value" can vary from 1 to 255, and the default is 255.

The midi file must be setup correctly using the ringtone player tool for this mode to be effective. First, open the midi file. Then, select the "Sequencer" radio button, and select the appropriate channel(s). (The W5691 only allows one-channel synchronization, while the W56940 and W56964 support multiple channels.) Lastly, convert the midi file to XMF.

Then, by selecting Mode 2 and playing this midi file, the vibrator is synchronized with the notes.

2.4 Mode 3

Mode 3 is event mode, in which the vibrator is controlled by midi events. Midi events are otherwise-unused notes that represent motor activity. These notes must be prepared in advance using a emidi editor, like cakewalk. Create the notes at the desired times using note B7 (decade 95) on channel 10. The level of vibration is controlled by the velocity, which varies from 0 to 127. One example is shown below.
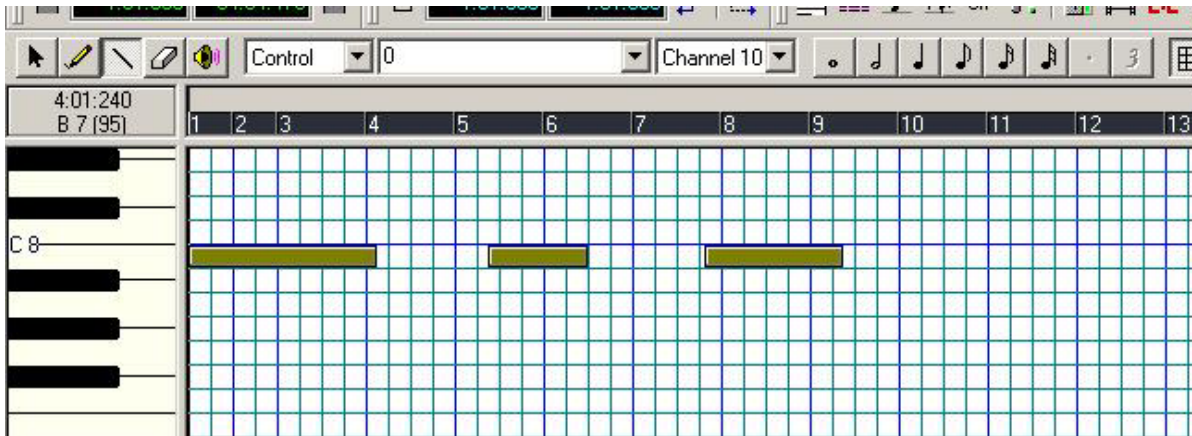
2.5 Mode 4

Mode 4 is pattern mode. The pattern is based on a table that contains the vibrator run-time and the level of vibration for each part of the pattern. (An example is provided in section 3.1.) Then, the location of this table is provided to API function W569_SetMotorPattern to build the epattern. Whenever midi is playing, the vibrator follows the specified pattern, repeating it until the midi sound stops. If no sound is playing, the vibrator does not run.

The pattern can be morphed. Morphing makes the changes in vibration level more gradual by increasing or decreasing the vibration level in steps. The morphing time is the duration of each step. For example, if one part of the pattern lasts 500 ms and the morphing time is 50 ms, the vibrator level changes (linearly) from the beginning level to the final level in ten (500 ms / 50 ms) steps. The shorter the morphing time, the more gradual the change in vibration level, but the effect diminishes as morphing time becomes really small. The morphing time is specified in W569_SetMotorPattern.

3. Implementation

This section provides program examples and frameworks for controlling the W569x vibrator in different command modes. It also includes a note about differences between the W5691BY and W5691CY.
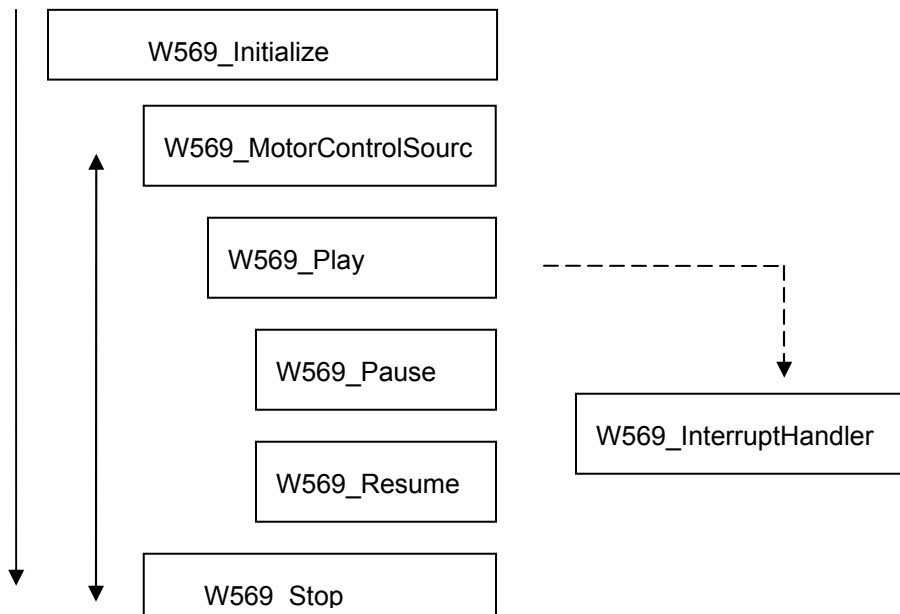
First, it begins with a reminder to set the control mode again for different channels. Consider this example:

W569_MotorControlSource(2);

W569_Play(midi 1);

…………..

W569_Play(midi 2);

…………..

W569_Play(midi 3);

…………..

W569_MotorControlSource(0);

W569_Play(midi 4);

…………..

This program sets the control mode and channel before midi 1 and not before midi 2 or midi 3. If midi 1 synchronizes with the vibrator on channel 5, the vibrator tries to synchronize with midi 2 on channel 5 as well. If midi 2 does not use channel 5, the W569x uC turns off the motor to save power and does not turn it on again when midi 2 finishes. As a result, if midi 3 uses channel 5, the vibrator will run synchronize with midi 3 again. This applies to event mode and pattern mode as well. If this behavior is not desired, change the control mode before midi 2 and midi 3; i.e., use the W569_MotorControlSource command again.

3.1   Source Coding

This section provides several examples of programs that control the vibrator in different situations. A general framework is provided below.

```
┌──────────────────────────────┐
│        W569_Initialize       │
└──────────────────────────────┘
    ┌──────────────────────────────┐
    │   W569_MotorControlSourc     │
    └──────────────────────────────┘
        ┌──────────────────────────────┐
        │         W569_Play            │ ─ ─ ─ ─ ─ ─ ─ ┐
        └──────────────────────────────┘               │
            ┌──────────────────────────────┐           │
            │        W569_Pause            │           ▼
            └──────────────────────────────┘   ┌──────────────────────────────┐
                                               │   W569_InterruptHandler      │
            ┌──────────────────────────────┐   └──────────────────────────────┘
            │        W569_Resume           │
            └──────────────────────────────┘
        ┌──────────────────────────────┐
        │         W569  Stop           │
        └──────────────────────────────┘
```

The rest of this section provides program examples, separated by remark or comment statements. Please note that W569_initialize is only shown once because all of these examples are shown as part of the same program. If the examples are divided into separate programs, W569_initialize should be included at the beginning of each one.

/*--*/

//To force control vibrator at power down mode

/*--*/

W569_initialize;

W569_MotorControlSource(1);

W569_SetMotorOnOff(1);

    ……

W569_SetMotorOnOff(0) ;

    ……..

```
/*------------------------------------------------------------------*/
//play sound and force control vibrator
/*------------------------------------------------------------------*/
W569_MotorControlSource(1);
ID=W569_Play(playdata1,sizeof(g_abysound2),2);
while(W569_IsPlaying(ID))
{
W569_SetMotorOnOff( 255 );
    …….
W569_SetMotorOnOff( 0) ;
    ……..
}
W569_Stop(ID)


/*------------------------------------------------------------------*/
//play midi with sequence synchronization
/*------------------------------------------------------------------*/

W569_SetMotorBlinking(1);
W569_SetMotorLevel(150);
W569_MotorControlSource(2);
ID=W569_Play(playdata2,sizeof(g_abysequence),1);
while(W569_IsPlaying(ID))
{
}
W569_Stop(ID)


/
```

```
*----------------------------------------------------------*/
//play midi with midi event
/*----------------------------------------------------------*/


W569_SetMotorBlinking(1);
W569_MotorControlSource(3);
ID=W569_Play(playdata3,sizeof(g_abyevent),1);
while(W569_IsPlaying(ID))
{
}
W569_Stop(ID)
/*----------------------------------------------------------*/
//play midi with vibrator pattern
/*----------------------------------------------------------*/
BYTE abyMotorPlayPattern[] =
{
0xF4, 0x01,              /* 500 ms (until vibrator level is 255) */
0xFF,                      /* vibrator level = 255 */
0xF4, 0x01,              /* 500 ms (until vibrator level is 0) */
0x00,                      /* vibrator level = 0 */
0xFA, 0x00,              /* 250 ms (until vibrator level is 100) */
0x64,                      /* vibrator level=100*/
0xFA, 0x00,              /* 250 ms (until vibrator level is 0) */
0x00,                       /* vibrator level=0 */
0x00, 0x00              /* end of pattern */
};
BYTE *Motorpattern=abyMotorPlayPattern;
W569_SetMotorBlinking(1);
W569_SetMotorPattern(Motorpattern,50);
W569_MotorControlSource(4);
ID=W569_Play(playdata4,sizeof(g_abypattern),1);
while(W569_IsPlaying(ID))
{
}
W569_Stop(ID)
```

3.2 W569CY and W5691BY Differences

The definition of high-active and low-active is reversed. This is a setting that affects control modes 1 and 4 and is maintained in MWstd.h.

Prototype	#define MOTOR_LOW_ACTIVE value (default 0)	
Description	Setup the motor output is high-active or low-active	
Value	0	Motor is high-active
	1	Motor is low-active

In the W569CY, high-active means higher values correspond to higher speeds; 0 turns off the motor, and 255 is the highest vibration level. In the W5691BY, however, high-active means higher values correspond to lower speeds; 0 is the highest vibration level, and 255 turns off the motor. Low-active is always the opposite of high-active. The examples in the rest of this document are based on W569CY motors that are high-active or W5691BY motors that are low-active—0 turns off the motor, and 255 is the highest vibration level.

High-active and low-active have no meaning in control modes 2 or 3. In control modes 2 and 3, higher values correspond to higher speeds, regardless of the version or setting in MWstd.h.

This reversal creates another difference right after power-on. The setting in MWstd.h is not enforced until W569_initialize is called. Until then, the defaults—high-active and vibration level 0—are used, but the combined effect of these settings is the opposite because the definition of high-active is reversed. As a result, the W569CY vibrator is initially turned off, while the W5691BY vibrator initially runs at the highest level. Before initialize there is a most difference, the W5691BY vibrator output pin is high but W5691CY is low.

4. Version History

Revision	Date	Modifications
A0	January 2005	• Preliminary release
A1	March 2005	• Add Disclaimer

Important Notice

Winbond products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Further more, Winbond products are not intended for applications wherein failure of Winbond products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

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