Freescale Semiconductor
Application Note

Document Number: AN4435
Rev. 0, 01/2012

Sound Generation Logic (SGL) Module

Monophonic sound generation software examples demonstrating the Sound Generation Logic module

by: Steve Mihalik, Field Applications Engineering, Detroit
Daniel McKenna, Applications Engineering, East Kilbride

Contents

1 Introduction 1
2 SGL module configuration and use 3
2.1 SGL prescaler 3
2.2 Sound_ctrl mode and SDCIF flag behavior. 3
2.3 Recommended application sequences 4
2.4 Mapping eMIOS channels to MUXA, MUXB of SGL 5
3 Example implementation 6
3.1 Clocks and eMIOS. 6
3.2 Mode entry and peripheral configuration. 6
3.3 Waveform of SOUND output for PWM inputs 7
4 Design steps 8
5 Code 12
5.1 SGL project: code documented in application note 12
5.2 INTC_SW_SGL project: sample interrupt functions14

1 Introduction

The sounds demonstrated by software in this application note are:

- Monophonic[1], continuous sound which is terminated by software
- Monophonic, continuous sounds which are terminated after a fixed duration with cases of
  - Initial sound
  - Changed audio frequency
  - Changed amplitude
- Monophonic, continuous sound using on/off time periods for a fixed duration

Source code is available for these two examples:

1. SGL: code using SGL flag without interrupts. This is documented in the application note.
2. INTC_SW_SGL: sample functions that use SGL interrupt.

1. Some reference manuals use monotonic as a complement to polyphonic, but the correct term is monophonic. Monotonic refers to a single frequency or pitch being used. The SGL can generate this but also can change the frequency, producing a series of notes, which is monophonic.

The SGL produces monophonic sound using tone amplitude modulation, based on two different mixed signals. The first signal (MUXA input) has a variable frequency and fixed duty cycle. The second signal (MUXB input) has a fixed frequency and variable pulse width (duty cycle) for generation of the amplitude signal. The sound generator is generally a PWM that generates two different signals and a mixer to generate a tone at the output. It is passed through a first-order low-pass filter to produce a signal that can be fed to a speaker interface.[1] For testing, a speaker can be connected directly to the SOUND output.

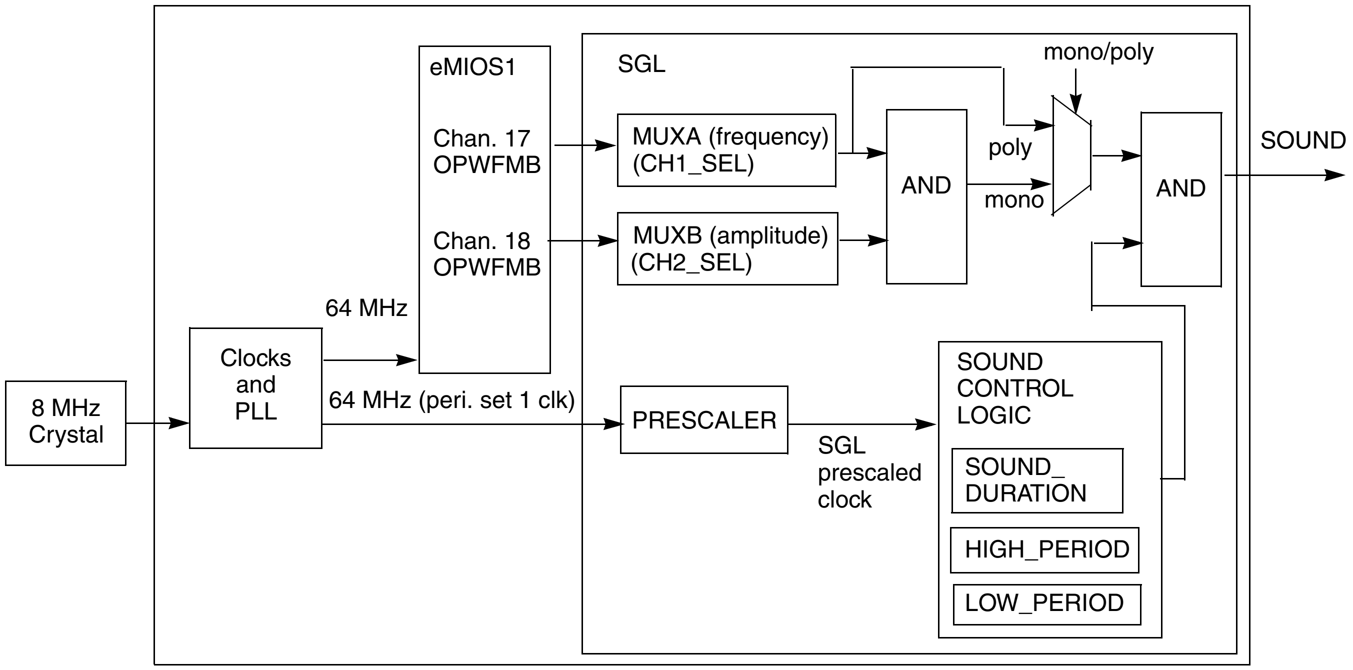


Figure 1. SGL module configured for monophonic example

Table 1. Microcontroller signals for Sound Generation Logic example

Signal	Port	SIU pad configuration and selection registers[1]	Package pin number			Freescale XPC56xx EVB motherboard pins
			144 QFP	176 QFP	208 BGA	
Sound	PC10	SIU_PCR40 = 0x0A00	60	76	T9	PJ4–11
eMIOS_1[17]	PJ12	SIU_PCR117 = 0x0A00	—	135	C6	PJ2–13
eMIOS_1[18]	PD5	SIU_PCR51 = 0x0E00	80	96	M15	PJ3–6
GPIO[43][2]	PC13	SIU_PCR43 = 0x0200	57	73	N9	Not required

[1] eMIOS channels do not need pad configuration in SIU for SGL use. They are configured as outputs in this example to observe sound output relative to PWM inputs.

[2] Required to power speaker on Freescale evaluation board MPC5606S_DEMO_V2.

1. MPC5606SRM, *MPC5606S Microcontroller Reference Manual,* Rev. 6, 11/2011, section 34.4.

2 SGL module configuration and use

2.1 SGL prescaler

The SGL prescaler, MODE_SEL[PRE], divides peripheral set 1 clock frequency. This post-prescaler clock is used for:

- Clock input to Sound_Duration, High_Period, Low_Period registers
- Clock for changing MODE_SEL bits
  Be careful to ensure one of these post-prescaler clocks passes before attempting any further mode changes. (Not an issue when prescaler is divide by 1.)

2.2 Sound_ctrl mode and SDCIF flag behavior

The SOUND_CTRL field value determines the mode for the SGL to operate per Table 2.

Table 2. MODE_SEL[SOUND_CTRL] field summary

Sound_ctrl mode	Sound_duration register	High_period, Low_period registers	Result
0	—	—	No sound generation
1	Applicable	—	Continuous sound until DURATION expires
2 or 3	—	—	Continuous sound until mode is exited
5	Applicable	Applicable	Periodic sound until DURATION expires
6 or 7	—	Applicable	Periodic sound until mode is exited

2.2.1 No sound generation (mode 0)

If SDCIF is set, then moving to mode 0 will stop sound generation and reset the duration count.

If SDCIF is not set, and sound is currently being generated, moving to mode 0 will not stop sound generation — sound generation will remain in its current state. For example, if sound is on, then moving to mode 0 will keep the sound on indefinitely. The duration count will freeze. If moving back into a duration mode, then the count will resume from where it has been left. (SeeSection 2.3, “Recommended application sequences,” for handling this case.)

2.2.2 Using Sound_duration register and SDCIF flag (modes 1, 5)

The SDCIF flag sets when duration expires. To continue with a subsequent duration (for example with new PWM values to change frequency or duty cycle) you must:

1. Move to mode 0 (resets state machine and enables clearing SDCIF flag).
2. Clear SDCIF flag.
3. Move back to mode 1 or mode 5. This re-arms the duration mode for sound generation.

Do not use duration values of zero or one, or else the SDCIF flag will not set. If using duration modes, use a value of two or greater for duration.

A change to the duration value before the duration expires (SDCIF flag sets) has the following effects:

- Mode 1: The value of the duration is stored when you enter the mode. Therefore any change to the value in the duration register will not take effect until the mode is re-entered.
- Mode 5: Any change to the duration register prior to the count reaching zero restarts the sound immediately with the new duration value.

2.2.3 Using High_period and Low_period registers (modes 5, 6, 7)

Do not use High_period or Low_period register values of zero when using periodic modes.

Changing the High_period or Low_period values will take effect as soon as the current high period and low period have completed their current cycle. In other words, changes take effect at the next full period.

2.3 Recommended application sequences

2.3.1 Entering mode from mode 0 (default)

1. Clear SDCIF flag (if set).
2. Configure Sound_duration, High_period, and Low_period registers.
3. Write desired mode value to SOUND_CTRL field.

2.3.2 Duration complete (modes 1, 5)

When SDCIF flag is set, the sequence to clear the flag and optionally start or re-start a mode is:

1. Move to mode 0 (enables clearing SDCIF flag).
2. Clear SDCIF flag.
3. If necessary, configure Sound_duration, High_period, and Low_period registers for next sound.
4. If necesary, start next sound by writing the same mode or any new mode value to SOUND_CTRL.

2.3.3 Exiting modes that do not use duration (modes 2, 3, 6, 7)

To stop sound generation when duration is not used, load a minimum duration value and temporarily enter a mode that uses duration. This will enable setting the SDCIF flag. The sequence is then similar to that in Section 2.3.2, “Duration complete (modes 1, 5).”

1. Set Sound_duration = 2.
2. Move to mode 1. After two clock cycles, the SDCIF flag will set and the sound will stop.
3. Move to mode 0.
4. Clear SDCIF flag.
5. If necessary, configure Sound_duration, High_period, and Low_period registers for next sound.
6. If necessary, start next sound by writing the same mode or any new mode value to SOUND_CTRL.

2.3.4 Exiting mode 1 before duration expires

1. Set Sound_duration = 2. This will hasten duration expiring with the minimum value.
2. Move to mode 2.
3. After one clock, move to mode 1.
4. After three clocks (one to clock mode, two to allow duration to expire) the sound will stop and the flag will set. Now move to mode 0.
5. Clear SDCIF flag.
6. If necessary, configure Sound_duration, High_period, and Low_period registers for next sound.
7. If necessary, start next sound by writing the same mode or any new mode value to SOUND_CTRL.

2.3.5 Exiting mode 5 before duration expires

This uses the same sequence as in the prior section, but moves back to mode 5 instead of mode 1 in step 3.

2.4 Mapping eMIOS channels to MUXA, MUXB of SGL

The mapping of the two channels for SGL module use is done in the MODE_SEL register per Table 3.

Table 3. Mapping of eMIOS channels for MODE_SEL fields CH1_SEL and CH2_SEL

CH1_SEL, CH2_SEL field value		eMIOS channel selected
0	0x0	eMIOS_0 Channel 16
1	0x1	eMIOS_0 Channel 17
2	0x2	eMIOS_0 Channel 18
3	0x3	eMIOS_0 Channel 19
4	0x4	eMIOS_0 Channel 20
5	0x5	eMIOS_0 Channel 21
6	0x6	eMIOS_0 Channel 22
7	0x7	eMIOS_0 Channel 23
8	0x8	eMIOS_1 Channel 16
9	0x9	eMIOS_1 Channel 17
10	0xA	eMIOS_1 Channel 18
11	0xB	eMIOS_1 Channel 19

Table 3. Mapping of eMIOS channels for MODE_SEL fields CH1_SEL and CH2_SEL (continued)

CH1_SEL, CH2_SEL field value		eMIOS channel selected
12	0xC	eMIOS_1 Channel 20
13	0xD	eMIOS_1 Channel 21
14	0xE	eMIOS_1 Channel 22
15	0xF	eMIOS_1 Channel 23

3 Example implementation

3.1 Clocks and eMIOS

Clocks and prescalers used are summarized below. Additional eMIOS information can be found in Freescale document AN2865, “MPC5500 & MPC5600 Simple Cookbook,” Rev. 4, 04/2010.

- 64 MHz clock to eMIOS_0 module
- eMIOS_1 module: 1 MHz internal clock (1 µs period)
  - — prescale 64 MHz clock to eMIOS module by 64
- eMIOS_1 channel 17 (audio frequency): 1 kHz PWM
  - — prescale emios_1 module clock by 1, count to 1k (1 ms), 50% duty cycle
- eMIOS_1 channel 18 (amplitude): 100 kHz PWM
  - — prescale emios_1 module clock by 1, count to 10 (10 µs), 50% duty cycle

3.2 Mode entry and peripheral configuration

Mode transition is required in changing Mode Entry (ME) module register values. For example, enabling the crystal oscillator to be active in the default mode (DRUN) requires enabling the crystal oscillator in appropriate mode configuration register (ME_DRUN_MC for DRUN), then initiating a mode transition to that mode. This example transitions from the default mode after reset (DRUN) to RUN0 mode.

Table 4. Mode configurations for Sound Generation Logic example[1]

Mode[2]	Mode configuration register	Settings									
		Mode configuration register value	Sysclk selection	Clock sources				Memory power mode		Main voltage reg.	I/O power down ctrl
				16 MHz IRC	XOSC0	PLL0	PLL1	Data flash	Code flash		
DRUN	ME_DRUN_MC	0x001F 0010 (default)	16 MHz IRC	**On**	Off	Off	Off	Normal	Normal	On	Off
RUN0	ME_RUN0_MC	0x001F 007D	PLL0	**On**	**On**	**On**	Off	Normal	Normal	On	Off

[1] Modes are enabled in ME_ME register.
[2] Other modes are not used in example.

Peripherals also have configurations to gate clocks on or off for different modes, enabling low power. Table 5 summarizes the peripheral configurations implemented to clock the required peripherals.

Table 5. Peripheral configurations for sound generation logic example[1]

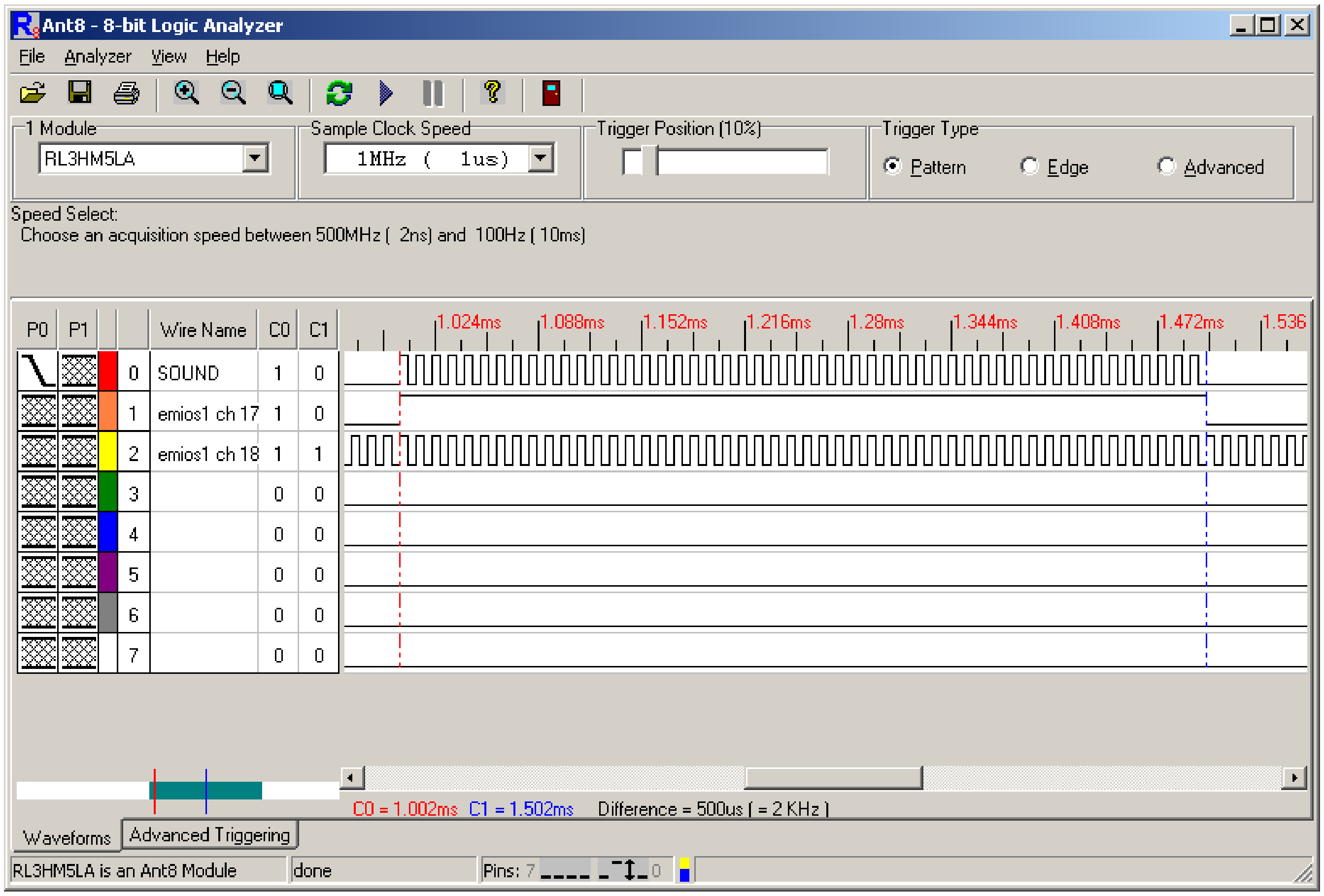
Peripheral Config.[2]	Peripheral Config. Register	Enabled modes								Peripherals selecting configuration	
		RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET	Peripheral	PCTL Reg. #
PC1	ME_ RUNPC_1	0	0	0	1	0	0	0	0	SIUL Sound Gen. eMIOS 1	68 62 72

[1] Low power modes are not used in example.

[2] Other peripheral configurations are not used in example.

3.3 Waveform of SOUND output for PWM inputs

The screenshot below shows waveforms for the monophonic continuous mode. The SOUND output is simply an "AND" of the two signals. The higher frequency component is used as a volume control and the low frequency is in the audio range to produce the actual heard audio frequency.



1 Module: RL3HM5LA

Sample Clock Speed: 1MHz (1us)

Trigger Position (10%)

Trigger Type: [● Pattern] ○ Edge ○ Advanced

Speed Select:
Choose an acquisition speed between 500MHz (2ns) and 100Hz (10ms)

	Wire Name	C0	C1
0	SOUND	1	0
1	emios1 ch 17	1	0
2	emios1 ch 18	1	1
3		0	0
4		0	0
5		0	0
6		0	0
7		0	0

C0 = 1.002ms C1 = 1.502ms Difference = 500us (= 2 KHz)

RL3HM5LA is an Ant8 Module | done | Pins: 7 0

Figure 2. Screenshot of SOUND output generated for PWM inputs

4 Design steps

Table 6. Steps for Sound Generation Logic example

Step		Relevant Field	Pseudo-code

Init modes and clock	Enable desired modes	DRUN=1,
		RUN0 = 1
			ME_ME = 0x0000 001D

	Initialize PLL0 dividers to provide 64 MHz for input crystal before mode configuration: • 8 MHz xtal: FMPLL[0]_CR=0x02400100		CGM_F MPLL[0]_CR = 0x02400100

	Configure Run0 Mode:
	• I/O output power-down: no safe gating
	• Main voltage regulator is on (default)
	• Data, code flash in normal mode (default)
	• PLL0 is switched on
	• Crystal oscillator is switched on
	• 16 MHz IRC is switched on (default)
	• Select PLL0 (system pll) as sysclk
		PDO=0
		MVRON=1
		DFLAON,
		CFLAON= 3
		PLL0ON=1
		XOSC0ON=1
		16MHz_IRCON=1
		SYSCLK=0x4
			ME_RUN0_MC = 0x001F 0074

	Peripheral configurations: • Peri. Config. 1: run in RUN0 mode only	RUN0=1	ME_RUN_PC1 = 0000 0010

	Assign peripheral configuration to peripherals:
	• SGL: select ME_RUN_PC1
	• SIUL: select ME_RUN_PC1
	• Sound Gen.: select ME_RUN_PC1
		RUN_CFG = 1
		RUN_CFG = 1
		RUN_CFG = 1
			MC_PCTL62 = 0x01
			MC_PCTL68 = 0x01
			MC_PCTL72 = 0x01

	Initiate software mode transition to RUN0 mode
	• Mode & key, then mode & inverted key
	• Wait for transition to complete
	• Verify current mode is RUN0
		TARGET_MODE = RUN0
		S_TRANS
		CURRENTMODE
			ME_MCTL = 0x4000 5AF0
			ME_MCTL = 0x4000 A50F
			wait ME_GS [S_TRANS] = 0
			verify ME_GS [CURRENTMODE]

Init Peri Clk Gen	Initialize peripheral clock generation
	• peripheral set 1 clock div cfg 0: ena peri set 1 (SGM) = sysclk/1
	• Aux Clk2 (eMIOS1): Select PLL for clock source
	• Aux Clk2 (eMIOS1): Enable input clk = AuxClk2/1
			:
			CGM_SC_DC0=0x80
			CGM_AC2_SC=0x03000000
			CGM_AC2_DC=0x80000000

disable Watchdog	Disable watchdog by writing keys to Status register, then clearing WEN		SWT_SR = 0x0000 C520
			SWT_SR = 0x0000 D928
			SWT_CR = 0x8000 010A

init EMIOS_1 ch 17 (1K Hz OPWFMB)	Set duty cycle = 500 eMIOS clocks (1 µs each)		EMIOS_1_CH[17]CADR = 500
	Set period = 1000 eMIOS clocks (1 µs each)		EMIOS_1_CH[17]CBDR = 1000

	Set up Channel Control:
	• Channel counter's prescaler = 1
	• Enable counter's prescaler
	• Set polarity to be active low
	• Mode = OPWFMB, next period update, flag at B
		UCPRE = 0
		UCPREN = 1
		EDPOL = 0
		MODE = 0x58
			EMIOS_1_CH[17] CCR = 0x0200 0058

Table 6. Steps for Sound Generation Logic example (continued)

Step		Relevant Field	Pseudo-code
init EMIOS ch 18 (100K Hz OPWFMB)	Set duty cycle = 5 eMIOS clocks (1 µs each)		EMIOS_1_CH[17]CADR = 5
	Set period = 10 eMIOS clocks (1 µs each)		EMIOS_1_CH[17]CBDR = 10

	Set up channel control:
	• Channel counter's prescaler = 1
	• Enable counter's prescaler
	• Set polarity to be active low
	• Mode = OPWFM, B next period update, flag at B
		
		UCPRE = 0
		UCPREN = 1
		EDPOL = 0
		MODE = 0x58
			EMIOS_1_CH[18] CCR = 0x0200 0058

initEMIOS1	Config eMIOS1: MHz internal clock & start timers:
	• Set global prescaler = div. by 64 (63 + 1) 1 µs clk
	• Enable global prescaler & internal clock
		
		GPRE = 63(0x3F)
		GPREN = 1
			
			EMIOS_1_MCR[GPRE] = 0x3F
			EMIOS_1_MCR[GPREN] = 1

initSGL for monophonic	Configure SGL:
	• Select monophonic sound
	• Sound control: no sound (default value)
	• No interrupt at duration complete
	• Prescale 64 MHz peripheral set 1 clock by 1
	• Mux A: use eMIOS_1 ch 17
	• Mux B: use eMIOS_1 ch 18
		
		M_P = 1
		SOUND_CTRL=0
		SDCIE = 0
		PRE =0
		CH1_SEL = 9
		CH2_SEL = 0xA
			SGL_MODE_SEL = 0x1009 000A

	Port PC10: SGL (option 2), enable as output	PA=2, OBE = 1	SIU_PCR[40] = 0x0A00

	If using Freescale MPC5606S_DEMO_V2 EVB, configure output to control BATT signal:
	• Enable port PC13 as GPIO output
	• Enable BATT switch to speaker IC
		
		OBE=1
		GPDO=1
			
			SIU_PCR[43] = 0x0200
			SIU_GPDO[43] = 1

SGL monophonic continuous - no duration (mode 2)	Configure SGL for continuous, no duration	SOUND_CTRL = 2	SGL_MODE_SEL[SOUND_CTRL] =2
	Wait for some delay before continuing on		Delay
	Set minimum duration to enable setting SDCIF		SGL_SOUND_DURATION = 2

	Turn off sound generation
	• Temporarily move to mode with duration
	• Move to mode 0: no sound generation and enables clearing flag
		
		SOUND_CTRL = 1
		SOUND_CTRL = 0
			
			SGL_MODE_SEL[SOUND_CTRL] = 1
			SGL_MODE_SEL[SOUND_CTRL] = 0

	Clear flag	SDCIF = 1	SGL_SGL_STATUS = 0x0000 0001

Table 6. Steps for Sound Generation Logic example (continued)

Step		Relevant Field	Pseudo-code
SGL monophonic continuous with duration (mode 1 — provides 3 durations of different sounds)	*1st Duration:*		

	Configure SGL for duration mode:
	• Set Duration = 1 sec (using 64 MHz clocks)
	• Sound control: continuous, duration mode
		SOUND_CTRL=1	SGL_SOUND_DURATION = 64 M
			SGL_MODE_SEL[SOUND_CTRL] =1

	Wait for sound duration to complete	SDCIF	wait for SGL_SGL_STATUS[SDCIF]=1
	Mode 0: no sound gen. & enables clearing flag	SOUND_CTRL=0	SGL_MODE_SEL[SOUND_CTRL] = 0
	Clear sound duration complete flag	SDCIF = 1	SGL_SGL_STATUS = 0x0000 0001
	2nd Duration:		

	Change frequency (Mux A PWM): 2 KHz, 50%:
	• Disable channel for update
	• Period = 2000 eMIOS clocks (1 µs each)
	• Duty = 1000 eMIOS clocks (1 µs each)
	• Re-enable updating channel
			EMIOS_1_OUDR[OU17] = 1
			EMIOS_1_CH[17]CBDR = 2000
			EMIOS_1_CH[17]CADR = 1000
			EMIOS_1_OUDR[OU17] = 0

	Re-configure SGL for continuous, duration mode	SOUND_CTRL=1	SGL_MODE_SEL[SOUND_CTRL] = 1
	Wait for sound duration to complete	SDCIF	wait for SGL_SGL_STATUS[SDCIF] = 1
	Mode 0: no sound gen. & enables clearing flag	SOUND_CTRL=0	SGL_MODE_SEL[SOUND_CTRL] = 0
	Clear sound duration complete flag	SDCIF = 1	SGL_SGL_STATUS = 0x0000 0001
	3rd Duration:		
	Change amplitude (Mux B PWM): • Duty = 30% (Mux B PWM period is 10 clocks)		EMIOS_1_CH[18]CBDR = 3
	Reconfigure SGL for continuous, duration mode	SOUND_CTRL=1	SGL_MODE_SEL[SOUND_CTRL] = 1
	Wait for sound duration to complete	SDCIF	wait for SGL_SGL_STATUS[SDCIF] = 1
	Mode 0: no sound gen. & enables clearing flag	SOUND_CTRL=0	SGL_MODE_SEL[SOUND_CTRL] = 0
	Clear sound duration complete flag	SDCIF = 1	SGL_SGL_STATUS = 0x0000 0001
SGL monophonic periodic with duration (mode 5)	Duration = 2 second (64 MHz clocks)		SGL_SOUND_DURATION = 128M
	Initialize high period		SGL_HIGH_PERIOD = 8M
	Initialize low period		SGL_LOW_PERIOD = 24M
	Configure SGL for continuous, duration mode	SOUND_CTRL=5	SGL_MODE_SEL[SOUND_CTRL] = 5
	Wait for sound duration to complete	SDCIF	wait for SGL_SGL_STATUS[SDCIF] = 1
	Mode 0: no sound gen. & enables clearing flag	SOUND_CTRL=0	SGL_MODE_SEL[SOUND_CTRL] = 0
	Clear sound duration complete flag	SDCIF = 1	SGL_SGL_STATUS = 0x0000 0001

Table 6. Steps for Sound Generation Logic example (continued)

Step		Relevant Field	Pseudo-code
SGL monophonic periodic, no duration (mode 6)	Initialize high period		SGL_HIGH_PERIOD = 4M
	Initialize low period		SGL_LOW_PERIOD = 12M
	Configure SGL for continuous, no duration mode	SOUND_CTRL = 6	SGL_MODE_SEL[SOUND_CTRL] = 6
	Wait for some delay before continuing on		Delay
	Set minimum duration to enable setting SDCIF		SGL_SOUND_DURATION = 2

	Turn off sound generation
	• Temporarily move to mode with duration
	• Move to mode 0: no sound generation and enables clearing flag
		SOUND_CTRL = 1
		SOUND_CTRL = 0
			SGL_MODE_SEL[SOUND_CTRL] = 1
			SGL_MODE_SEL[SOUND_CTRL] = 0

	Clear sound duration complete flag	SDCIF = 1	SGL_SGL_STATUS = 0x0000 0001

5 Code

5.1 SGL project: code documented in application note

```
/* main.c - SGL example to generate monophonic sounds */
/* Jan 11 2012 S Mihalik - Initial version */
/* Copyright Freescale Semiconductor, Inc 2012 All rights reserved. */
#include "MPC5606S_M07N.h"          /* Use proper include file */
  vuint32_t i = 0;                 /* Dummy idle counter */
  vuint32_t j = 0;                 /* Dummy timeout counter */

void initModesAndClock(void) {
  ME.MER.R = 0x0000001D;           /* Enable DRUN, RUN0, SAFE, RESET modes */
                                   /* Initialize PLL before turning it on: */
  CGM.FMPLL[0].CR.R = 0x02400100; /* 8 MHz xtal: Set PLL0 to 64 MHz */
  ME.RUN[0].R = 0x001F0074;        /* RUN0 cfg: 16MHzIRCON,OSC0ON,PLL0ON,syclk=PLL0*/
  ME.RUNPC[1].R = 0x00000010;      /* Peri. Cfg. 1 settings: only run in RUN0 mode */
  ME.PCTL[62].R = 0x01;            /* SGL   : select ME.RUNPC[1] */
  ME.PCTL[68].R = 0x01;            /* SIUL  : select ME.RUNPC[1] */
  ME.PCTL[73].R = 0x01;            /* EMIOS1: select ME.RUNPC[1] */
  ME.MCTL.R = 0x40005AF0;          /* Enter RUN0 Mode & Key */
  ME.MCTL.R = 0x4000A50F;          /* Enter RUN0 Mode & Inverted Key */
  while (ME.GS.B.S_MTRANS == 1) {} /* Wait for mode transition to complete */
  while(ME.GS.B.S_CURRENTMODE != 4) {} /* Verify RUN0 is the current mode */
}

void initPeriClkGen(void) {
  CGM.SC_DC[0].R = 0x80;           /* Enable peri set 1 (SGM) with sysclk/1 */
  CGM.AC2_SC.R   = 0x03000000;     /* Aux Clk2 (EMIOS_1): select PLL for clock source */
  CGM.AC2_DC.R   = 0x80000000;     /* Aux Clk2 (EMIOS_1): enable input clk=AuxClk2 / 1*/
}

void disableWatchdog(void) {
  SWT.SR.R = 0x0000c520;           /* Write keys to clear soft lock bit */
  SWT.SR.R = 0x0000d928;
  SWT.CR.R = 0x8000010A;           /* Clear watchdog enable (WEN) */
}

void initEMIOS_1(void) {
  /* eMIOS 1 chan 17:  Provide audio frequency to SGL Mux A */
  EMIOS_1.CH[17].CADR.R = 500;       /* Count n eMIOS clocks for OWPFMB duty cycle */
  EMIOS_1.CH[17].CBDR.R = 1000;      /* Count n eMIOS clocks for OWPFMB period */
  EMIOS_1.CH[17].CCR.R= 0x02000058; /* UCPRE==0, UCPREN=1, MODE=OPWFM */
  /* Next line done for test purposes to observe PWM signal */
  SIU.PCR[117].R = 0x0A00;           /* Port PJ12: eMIOS (opt.2), output */

  /* eMIOS 1 chan 18:  Provide amplitude to SGL Mux B */
  EMIOS_1.CH[18].CADR.R = 5;       /* Count n eMIOS clocks for OWPFMB duty cycle */
  EMIOS_1.CH[18].CBDR.R = 10;      /* Count n eMIOS clocks for OWPFMB period */
  EMIOS_1.CH[18].CCR.R= 0x02000058; /* UCPRE==0, UCPREN=1, MODE=OPWFM */
  /* Next line done for test purposes to observe PWM signal */
  SIU.PCR[51].R = 0x0E00;          /* Port PD5: eMIOS (opt.e), output */
                                   /* Start eMIOS_1 */
  EMIOS_1.MCR.B.GPRE = 63;         /* eMIOS clk = 64 MHz / (n+1) = 1 MHz */
  EMIOS_1.MCR.B.GPREN = 1;         /* enable prescaler and eMIOS clock */
}
```

```
void initSGL_mono(void) {
  SGL.MODE_SEL.B.M_P = 1;            /* Select monophonic sound */
  SGL.MODE_SEL.B.PRE = 0;            /* Prescale 64 MHz clock by 1 */
  SGL.MODE_SEL.B.CH1_SEL = 0x9;      /* Select EMIOS_1 ch 17: 1KHz, 50% duty */
  SGL.MODE_SEL.B.CH2_SEL = 0xA;      /* Select EMIOS_1 ch 18: 2KHz, 50% duty */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;     /* Sound control: No sound (reset default) */
  SIU.PCR[40].R = 0x0A00;            /* Port PC10: SGL (opt. 2) output */
  /* The following two lines apply if using Freescale MPC5606S_DEMO_V2 EVB: */
  SIU.PCR[43].R = 0x0200;            /* Port PC13: GPIO output for BATT switch */
  SIU.GPDO[43].R = 1;                /* Enable BATT switch to speaker circuit */
}
void SGL_mono_cont(void){            /* MODE 2 */
  SGL.MODE_SEL.B.SOUND_CTRL = 2;     /* Sound control: enable continuous sound */
  for (j=0; j<10000000; j++)  {}     /* Delay, then exit mode 2 */
  SGL.SOUND_DURATION.R = 2;          /* Duration: set >1 to enable SDCIF flag*/
  SGL.MODE_SEL.B.SOUND_CTRL = 1;     /* Sound ctrl: expire DURATION to set flag */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;     /* Sound ctrl: no sound, enable clearing flag */
  SGL.SGL_STATUS.R = 0x00000001;     /* Clear flag */
}

void SGL_mono_cont_duration(void){     /* MODE 1 multiple cases */
/* 1st duration:  1 second at current frequency, duty cycle */
  SGL.SOUND_DURATION.R = 64000000;     /* Duration: # of SGLclks (1 sec) */
  SGL.MODE_SEL.B.SOUND_CTRL = 1;       /* Sound ctrl: continuous sound with duration*/
  while (!SGL.SGL_STATUS.B.SDCIF) {}   /* Wait for flag */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;       /* Sound ctrl: no sound, enable clearing flag*/
  SGL.SGL_STATUS.R = 0x00000001;       /* Clear flag */
/* 2nd duration: change audio frequency(Mux A PWM frequency with fixed duty) */
  EMIOS_1.OUDR.B.OU17 = 1;             /* Disable emios1 ch 17 for updating data regs */
  EMIOS_1.CH[17].CBDR.R = 2000;        /* Mux A period:  set to 2000 eMIOS clks */
  EMIOS_1.CH[17].CADR.R = 1000;        /* Mux A duty cycle: keep at 50% */
  EMIOS_1.OUDR.B.OU17 = 0;             /* Re-enable emios 1 chan 17  */
  SGL.MODE_SEL.B.SOUND_CTRL = 1;       /* Sound ctrl: continuous sound with duration*/
  while (!SGL.SGL_STATUS.B.SDCIF)  {}  /* Wait for flag */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;       /* Sound ctrl: no sound, enable clearing flag*/
  SGL.SGL_STATUS.R = 0x00000001;       /* Clear flag */
/* 3rd duration: change amplitude (Mux B PWM duty cycle) */
  EMIOS_1.CH[18].CADR.R = 3;           /* Mux B duty: change from 5 to 3 */
  SGL.MODE_SEL.B.SOUND_CTRL = 1;       /* Sound ctrl: continuous sound with duration*/
  while (!SGL.SGL_STATUS.B.SDCIF) {}   /* Wait for flag */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;       /* Sound ctrl: no sound, enable clearing flag*/
  SGL.SGL_STATUS.R = 0x00000001;       /* Clear flag */
}

void SGL_mono_periodic_duration(void){ /* MODE 5 */
  SGL.SOUND_DURATION.R = 128000000;    /* Duration:    # of SGL clks (2 sec) */
  SGL.HIGH_PERIOD.R     =   8000000;   /* High period: # of SGL clks (1/8 sec) */
  SGL.LOW_PERIOD.R      =  24000000;   /* High period: # of SGL clks (3/8 sec) */
  SGL.MODE_SEL.B.SOUND_CTRL = 5;       /* Sound ctrl: cont sound w duration & periods*/
  while (!SGL.SGL_STATUS.B.SDCIF) {}   /* Wait for flag */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;       /* Sound ctrl: no sound, enable clearing flag*/
  SGL.SGL_STATUS.R = 0x00000001;       /* Clear flag */
}

void SGL_mono_periodic(void){          /* MODE 6 */
  SGL.HIGH_PERIOD.R     =   4000000;   /* High period: # of SGL clks (1/16 sec) */
  SGL.LOW_PERIOD.R     =  12000000;  /* High period: # of SGL clks (3/16 sec) */
  SGL.MODE_SEL.B.SOUND_CTRL = 6;     /* Sound ctrl: cont sound with periods*/
  for (j=0; j<10000000; j++)  {}     /* Delay then exit mode 6 */
  SGL.SOUND_DURATION.R = 2;          /* Duration: set >1 to enable SDCIF flag */
  SGL.MODE_SEL.B.SOUND_CTRL = 1;     /* Sound ctrl: expire DURATION to set flag */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;     /* Sound ctrl: no sound, enable clearing flag*/
  SGL.SGL_STATUS.R = 0x00000001;     /* Clear flag */
}
void main (void) {
  initModesAndClock();         /* Initialize mode entries and system clock */
  initPeriClkGen();            /* Generate peripheral clk to SGL, other set 1 peris */
  disableWatchdog();           /* Disable watchdog */
  initEMIOS_1();               /* Initialize eMIOS 1 channels 17 & 18 */
  initSGL_mono();              /* Init SGL: Mono, no sound, prescale by 1, emios1 ch 17, 18*/
  SGL_mono_cont();             /* Generate a mono sound, continuous */
  for (j=0; j<5000000; j++){}  /* Delay between sounds */
  SGL_mono_cont_duration();    /* Generate different mono sounds, cont w duration */
  for (j=0; j<5000000; j++){}  /* Delay between sounds */
  SGL_mono_periodic_duration();/* Generate mono sound, periodic with duration */
  for (j=0; j<5000000; j++){}  /* Delay between sounds */
  SGL_mono_periodic();         /* Generate mono sound, no duration */
  while (1) {
        i++;
  }
}
```

5.2 INTC_SW_SGL project: sample interrupt functions

```
void initSGL_mono_dur_irq(void) {
  SGL.MODE_SEL.B.M_P = 1;          /* Select monophonic sound */
  SGL.MODE_SEL.B.PRE = 0;          /* Prescale 64 MHz clock by 1 */
  SGL.MODE_SEL.B.CH1_SEL = 0x9;    /* Select EMIOS_1 ch 17: 1KHz, 50% duty */
  SGL.MODE_SEL.B.CH2_SEL = 0xA;    /* Select EMIOS_1 ch 18: 2KHz, 50% duty */
  SGL.MODE_SEL.B.SOUND_CTRL = 0;   /* Sound control: No sound (reset default) */
  SIU.PCR[40].R = 0x0A00;          /* Port PC10: SGL (opt. 2) output */
  /* The following two lines apply if using Freescale MPC5606S_DEMO_V2 EVB: */
  SIU.PCR[43].R = 0x0200;          /* Port PC13: GPIO output for BATT switch */
  SIU.GPDO[43].R = 1;              /* Enable BATT switch to speaker circuit */
  SGL.SOUND_DURATION.R = 64000000; /* Duration: # of SGLclks (1 sec) */
  SGL.MODE_SEL.B.SDCIE = 1;        /* Enable sound duration complete interrupt */
  INTC.PSR[183].R = 3;             /* Set interrupt priority */
  SGL.MODE_SEL.B.SOUND_CTRL = 5;   /* Sound ctrl: dummy write */
  SGL.MODE_SEL.B.SOUND_CTRL = 1;   /* Sound ctrl: continuous sound with duration*/
}

void SGL_ISR(void) {                /* Decrease duty cycle until 0 */
  if (EMIOS_1.CH[18].CADR.R > 0) {  /* If duty cycle >0 */
      EMIOS_1.CH[18].CADR.R = EMIOS_1.CH[18].CADR.R - 1;  /* decrease duty cycle*/
  }
  else {
    SGL.MODE_SEL.B.SDCIE = 0;         /* Disable further interrupts */
  }
  SGL.MODE_SEL.B.SOUND_CTRL = 0; /* Sound ctrl: first write */
  SGL.SGL_STATUS.R = 0x00000001; /* Clear flag done after writing SOUND_CTRL=0*/
  SGL.MODE_SEL.B.SOUND_CTRL = 1; /* Sound ctrl: re-arm cont. sound with dur. */
  }
}
```

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4435
Rev. 0
01/2012