To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

Application Note

78K0 Family Sound Generator

8-bit Single-Chip Microcontroller

Hardware

µPD1615, µPD16F15, µPD1616
µPD780823, µPD780824, µPD780826
µPD780828, µPD78F0828
µPD780948, µPD78F0948
µPD780949, µPD78F0949
µPD780973, µPD78F0974

Document No. U14227EE1V0AN00
Date Published April 1999

FIP is a trademark of NEC Corporation
EEPROM and IEBus are trademarks of NEC Corporation.
MS-DOS and MS-Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
PC/AT and PC DOS are trademarks of IBM Corp.
IBM-DOS, PC/AT and PC DOS are trademarks of International Business Machines Corporation.
HP9000 Series 300, HP9000 Series 700, and HP-UX are trademarks of Hewlett-Packard Company.
SPARCstation is a trademark of SPARC International, Inc.
Sun OS is a trademark of Sun Microsystems, Inc.
Ethernet is a trademark of Xerox Corp.
NEWS and NEWS-OS are trademarks of Sony Corporation.
OSF/Motif is a trade mark of OpenSoftware Foundation, Inc..
TRON is an abbreviation of The Realtime Operating system Nucleus.
ITRON is an abbreviation of Industrial TRON.

The related documents in this publication may include preliminary versions. However, preliminary versions are not marked as such.

The export of this product from Japan is regulated by the Japanese government. To export this product may be prohibited without governmental license, the need for which must be judged by the customer. The export or re-export of this product from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

M4 94.11

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

Table of Contents

Introduction 7
1. What is “Sound”? 7
2. Digital solutions 8

Chapter 1 NEC Sound Generator 9
1.1 Possibilities 9
1.2 Block Diagram 9
1.3 Registers 10
1.3.1 Sound Generator Control Register (SGCR) 10
1.3.2 Sound Generator Buzzer Control Register (SGBR) 12
1.3.3 Sound Generator Amplitude Register (SGAM) 13
1.3.4 Sound Generator Settings 14
1.4 Output Signal 14

Chapter 2 Examples 16
2.1 Alarm & Gong Programs 16
2.1.1 Organisation 16
2.1.2 The Main 17
2.1.3 Function Called to Produce an Amplitude Variation 19
2.1.4 Table to Produce a Quick Amplitude Variation 22
2.1.5 Function Called to Produce a Frequency Variation 25
2.1.6 Table to Produce a Quick Frequency Variation 28
2.1.7 Interrupt Vectors Pointers 31
2.2 Program to Execute "Au Clair de la Lune" 33

Chapter 3 Hardware 37

Contents of Figures

Figure 1-1: Sound Generator Block Diagram 9
Figure 1-2: Sound Generator Control Register (SGCR) Format 10
Figure 1-3: TCE bit function 10
Figure 1-4: Setting the bits SGOB, SGCL2, SGCL1 and SGCL0 11
Figure 1-5: Maximum and minimum values of the buzzer output frequency 11
Figure 1-6: Sound Generator Buzzer Control Register (SGBR) Format 12
Figure 1-7: Sound Generator Amplitude Register (SGAM) Format 13
Figure 1-8: Sound Generator Output Signal 14

Figure 2-1: Organisation 16

Figure 3-1: Security Circuit by Filtering 37

Contents of Tables

2.1.4 Table to Produce a Quick Amplitude Variation 22

2.1.6 Table to Produce a Quick Frequency Variation 28

Introduction

The sound generator produces signals composed of frequency output and PWM signal for volume control. The generated frequency is in the range of 256 Hz to 7.7 kHz, it can be used either for simple alarm sounds, buzzer, gong or beeper.

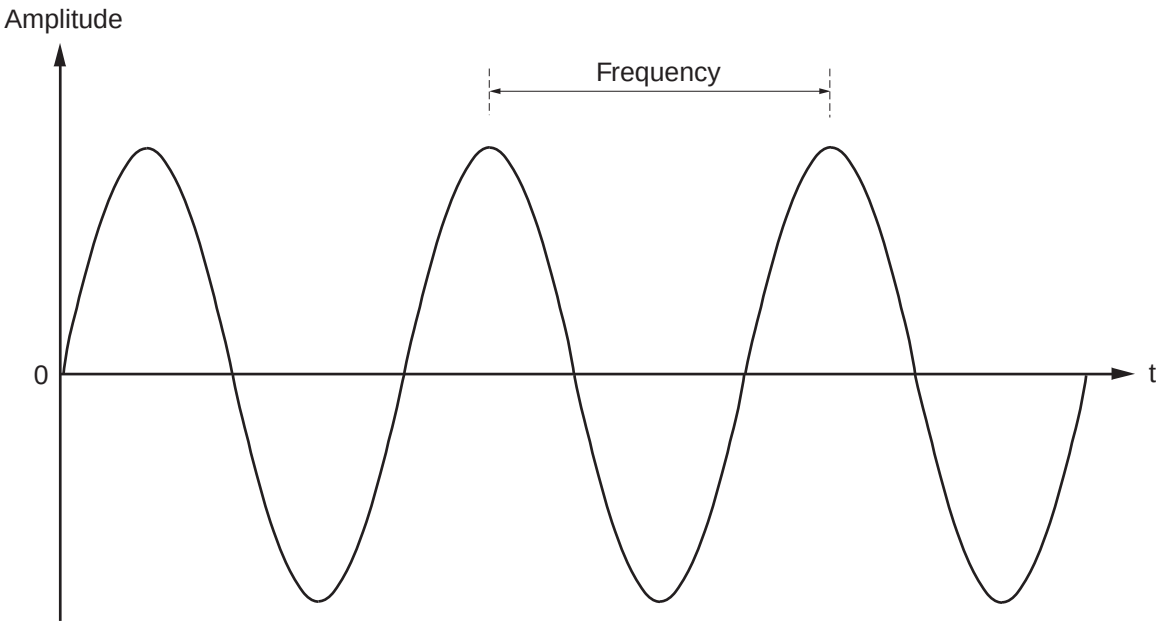
The sound generator is ideal for all the application which need events sounds or alarm, such as:
- Dashboard
- Body unit
- Security
- Multifunction display

A large panel of microcontrollers have sound generator on chip :
- µPD78(F)097X
- µPD78(F)082X
- µPD16(F)1X
- µPD78(F)094X

This application note has been developed for the 78K0(F)94X microcontroller. Therefore, all the programs are dedicated to the 78K0(F)94x microcontroller (especially for the interrupt pointers). But there is an easy migration path between all the 78K0 products. So it is very simple to transfer this code to the other 8 bits microcontrollers with sound generator on chip.

1. What is “Sound”?

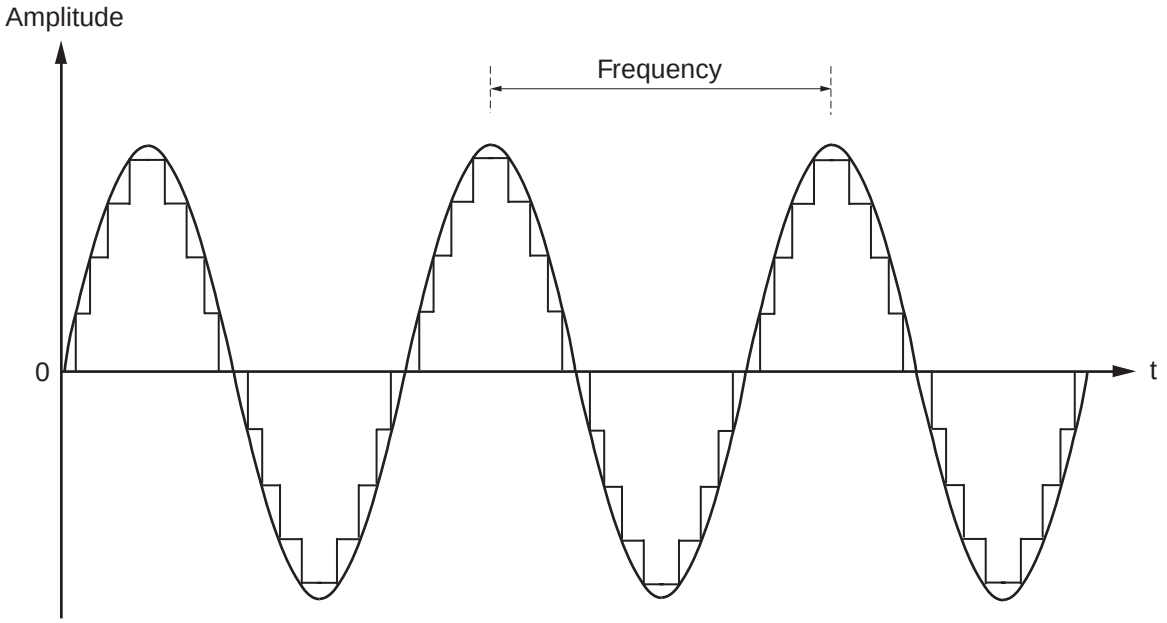
The simplest form of sound is a harmonic wave. So we can define a sound by a frequency and an amplitude :



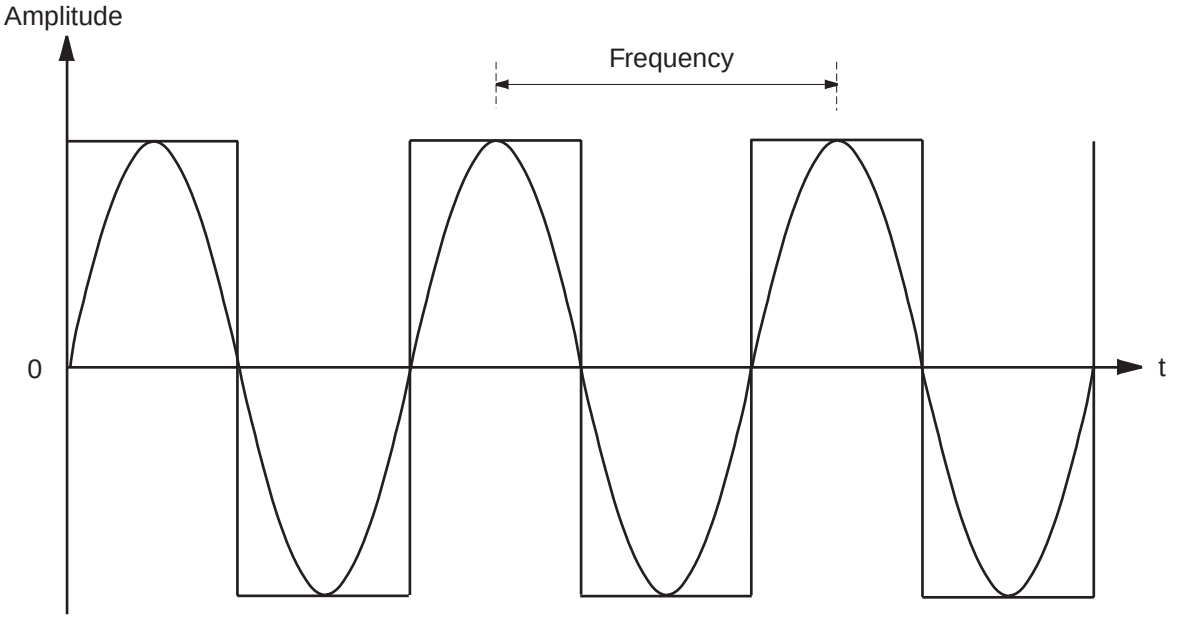
The amplitude defines the volume and the frequency defines the tone.
Now the target is to reproduce a sound.

2. Digital solutions

- Sine wave approach with a Digital to Analog converter :



- Sine wave approach by Pulse With Modulation (PWM) output and with filtering :



The NEC sound generator uses the second solution. The advantages are that we need to do only an amplitude selection and a frequency selection.

Chapter 1 NEC Sound Generator

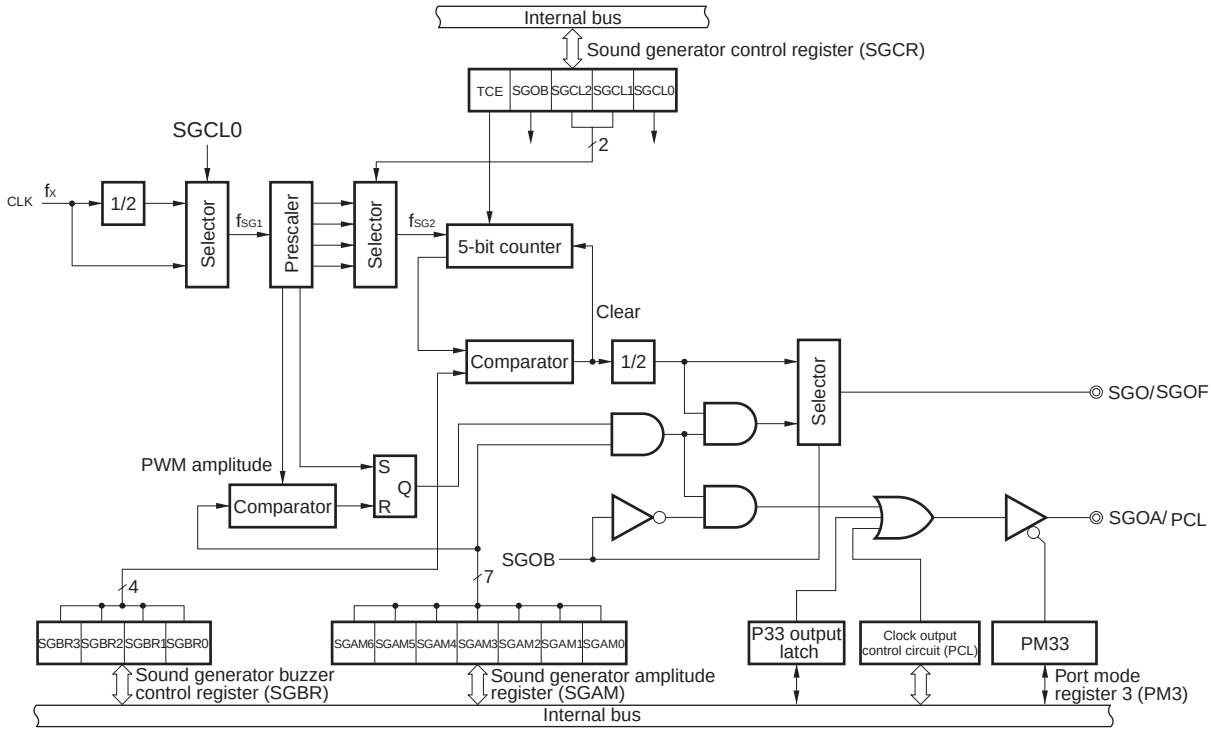
1.1 Possibilities

The control of different registers permits to choose:

- A frequency output signal between 250 Hz and 7,3 kHz,
- One of the 128 steps amplitude level,
- A composed or separated frequency/amplitude output.

1.2 Block Diagram

Figure 1-1: Sound Generator Block Diagram



1.3 Registers

The following three types of registers are used to control the sound generator :

- Sound generator control register (SGCR),
- Sound generator buzzer control register (SGBR),
- Sound generator amplitude control register (SGAM).

1.3.1 Sound Generator Control Register (SGCR)

SGCR is a register which sets up the followings four types :

- Controls sound generator output,
- Selects output of sound generator,
- Selects sound generator input frequency fSG1,
- Selects 5-bit counter input frequency fSG2.

SGCR is set with a 1-bit or 8-bits memory manipulation instruction.
RESET input SGCR to 00H.

Figure 1-2 shows the SGCR format.

Figure 1-2: Sound Generator Control Register (SGCR) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
SGCR	TCE	0	0	0	SGOB	SGCL2	SGCL1	SGCL0	FFC0H	04H	R/W

Figure 1-3 shows the TCE bit function.

Figure 1-3: TCE bit function

TCE	Sound Generator Output Selection
0	Timer operation stopped SGOF/SGO and SGOA for low-level output
1	Sound generator operation SGOF/SGO and SGOA for output

SGOF : Basic cycle signal (without amplitude),
SGO : Basic cycle signal (with amplitude),
SGOA : Amplitude signal

Caution : It's better before setting the TCE bit, to set all the other bits.

Figure 1-4 shows how to set bits SGOB, SGCL2, SGCL1 and SGCL0.

Figure 1-4: Setting the bits SGOB, SGCL2, SGCL1 and SGCL0

SGOB	Sound Generator Output Selection
0	Selects SGOF and SGOA outputs
1	Selects SGO and PCL outputs

SGCL2	SGCL1	5-Bit Counter Input Frequency f_{SG2} Selection
0	0	$f_{SG2} = f_{SG1}/2^5$
0	1	$f_{SG2} = f_{SG1}/2^6$
1	0	$f_{SG2} = f_{SG1}/2^7$
1	1	$f_{SG2} = f_{SG1}/2^8$

SGCL0	Sound Generator Input Frequency Selection
0	$f_{SG1} = f_X/2$
1	$f_{SG1} = f_X$

Cautions : 1. When rewriting SGCR to other data, stop the timer operation (TCE=0) before hand.
2. Bits 4 to 6 must be set to 0.

Maximum and minimum values of the buzzer output frequency are shown in the Figure 1-5.

Figure 1-5: Maximum and minimum values of the buzzer output frequency

SGCL2	SGCL1	SGCL0	Maximum and Minimum Values of Buzzer Output				
			f_{SG2}	f_X = 8 MHz		f_X = 8.38 MHz	
				Max. (kHz)	Min. (kHz)	Max. (kHz)	Min. (kHz)
0	0	0	$f_{SG1}/2^6$	3.677	1.953	3.851	2.046
0	0	1	$f_{SG1}/2^5$	7.354	3.906	7.702	4.092
0	1	0	$f_{SG1}/2^7$	1.838	0.976	1.926	1.024
0	1	1	$f_{SG1}/2^6$	3.677	1.953	0.481	2.046
1	0	0	$f_{SG1}/2^8$	0.919	0.488	0.963	0.512
1	0	1	$f_{SG1}/2^7$	1.838	0.976	1.926	1.024
1	1	0	$f_{SG1}/2^9$	0.460	0.244	0.481	0.256
1	1	1	$f_{SG1}/2^8$	0.919	0.488	0.963	0.512

1.3.2 Sound Generator Buzzer Control Register (SGBR)

SGBR is a register that sets the basic frequency of the sound generator output signal.
SGBR is set with a 1-bit or 8-bits memory manipulation instruction.
RESET input clears SGCR to 00H.

Figure 1-6 shows the SGBR format.

Figure 1-6: Sound Generator Buzzer Control Register (SGBR) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
SGBR	0	0	0	0	SGBR3	SGBR2	SGBR1	SGBR0	FFC2H	00H	R/W

SGBR3	SGBR2	SGBR1	SGBR0	Buzzer Output Frequency (kHz) Note	
				fx = 8 MHz)	fx = 8.38 MHz)
0	0	0	0	3.677	3.851
0	0	0	1	3.472	3.637
0	0	1	0	3.290	3.446
0	0	1	1	3.125	3.273
0	1	0	0	2.976	3.117
0	1	0	1	2.841	2.976
0	1	1	0	2.717	2.847
0	1	1	1	2.604	2.728
1	0	0	0	2.500	2.619
1	0	0	1	2.404	2.518
1	0	1	0	2.315	2.425
1	0	1	1	2.232	2.339
1	1	0	0	2.155	2.258
1	1	0	1	2.083	2.182
1	1	1	0	2.016	2.112
1	1	1	1	1.953	2.046

Remark : The values in the previous table exist when SGCL0, SGCL1 and SGCL2 are setting to 0,0 and 0.

Cautions : 1. When rewriting SGBR to other data, stop the timer operation (TCE=0) before hand.
2. Bits 4 to 7 must be set to 0.

1.3.3 Sound Generator Amplitude Register (SGAM)

SGAM is a register that sets the amplitude of the sound generator output signal.
SGAM is set with a 1-bit or 8-bits memory manipulation instruction.
RESET input clears SGAM register to 00H.

Figure 1-7 shows the SGAM format.

Figure 1-7: Sound Generator Amplitude Register (SGAM) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
SGAM	0	SGAM6	SGAM5	SGAM4	SGAM3	SGAM2	SGAM1	SGAM0	FFC1H	00H	R/W

SGAM6	SGAM5	SGAM4	SGAM3	SGAM2	SGAM1	SGAM0	Amplitude
0	0	0	0	0	0	0	0/128
0	0	0	0	0	0	1	2/128
0	0	0	0	0	1	0	3/128
0	0	0	0	0	1	1	4/128
0	0	0	0	1	0	0	5/128
0	0	0	0	1	0	1	6/128
0	0	0	0	1	1	0	7/128
0	0	0	0	1	1	1	8/128
0	0	0	1	0	0	0	9/128
0	0	0	1	0	0	1	10/128
0	0	0	1	0	1	0	11/128
0	0	0	1	0	1	1	12/128
0	0	0	1	1	0	0	13/128
0	0	0	1	1	0	1	14/128
0	0	0	1	1	1	0	15/128
0	0	0	1	1	1	1	16/128
0	0	1	0	0	0	0	17/128
0	0	1	0	0	0	1	18/128
0	0	1	0	0	1	0	19/128
0	0	1	0	0	1	1	20/128
0	0	1	0	1	0	0	21/128
0	0	1	0	1	0	1	22/128
0	0	1	0	1	1	0	23/128
0	0	1	0	1	1	1	24/128
0	0	1	1	0	0	0	25/128
0	0	1	1	0	0	1	26/128
0	0	1	1	0	1	0	27/128
0	0	1	1	0	1	1	28/128
0	0	1	1	1	0	0	29/128
0	0	1	1	1	0	1	30/128
0	0	1	1	1	1	0	31/128
			⋮				⋮
1	1	1	1	1	1	1	128/128

Cautions :
1. **When rewriting the contents of SGAM, the timer operation does not need to be stopped. However, note that a high level may be output for one period due to rewrite timing.**
2. **Bits 7 must be set to 0.**

1.3.4 Sound Generator Settings

<A>. Choose the sound generator output by SGOB bit.

<B>. Set port 3 in output mode by PM3 register.

<C>. Set the maximum and the minimum values of the buzzer output frequency through SGCL2, SGCL1 and SGCL1.

<D>. Choose the output frequency by SGBR register.
The sound generator output frequency f_{SG} can be calculated by the following expression :

$$f_{SG} = 2^{(SGCL0 - SGCL1 - 2 \times SGCL2 - 7)} \times [\, f_X / (SGBR + 17)]$$

Remarks :

- fx is the component frequency.
- SGCL2, SGCL1 and SGCL0 values must be replaced by decimals values in the previous expression.

Example :

When fx = 8MHz, SGCL0 to SGCL2 is (1,0,0), SGBR0 to SGBR3 is (1,1,1,1) and SGBR = 15. Therefore, the sound generator output frequency is fsg = 3,906 KHz.

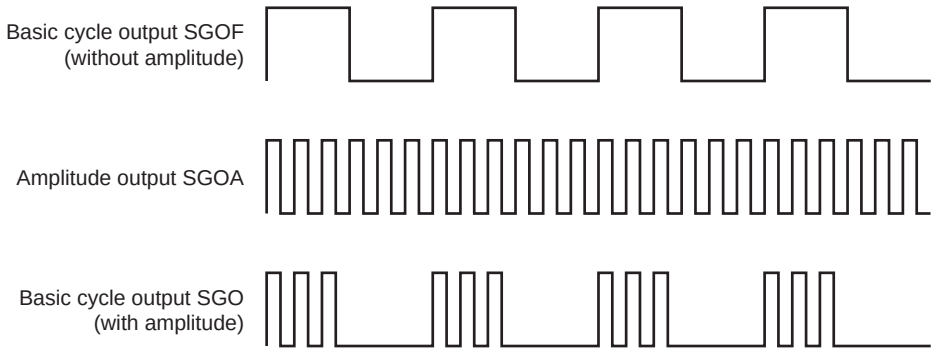
<E>. Select the amplitude of the output signal by SGAM register.

<F>. Set TCE to 1 to start the sound generator operation.

1.4 Output Signal

Select SGO or SGOF by setting bit 3 (SGOB) of the sound generator control register (SGCR) to “1”. If you choose to use the SGOF output, the amplitude specified by the SGAM0 to SGAM6 is output from the SGOA pin. When SGO output is selected, the SGOA pin can be used as a PCL output (clock output) or I/O port pin.

Figure 1-8: Sound Generator Output Signal



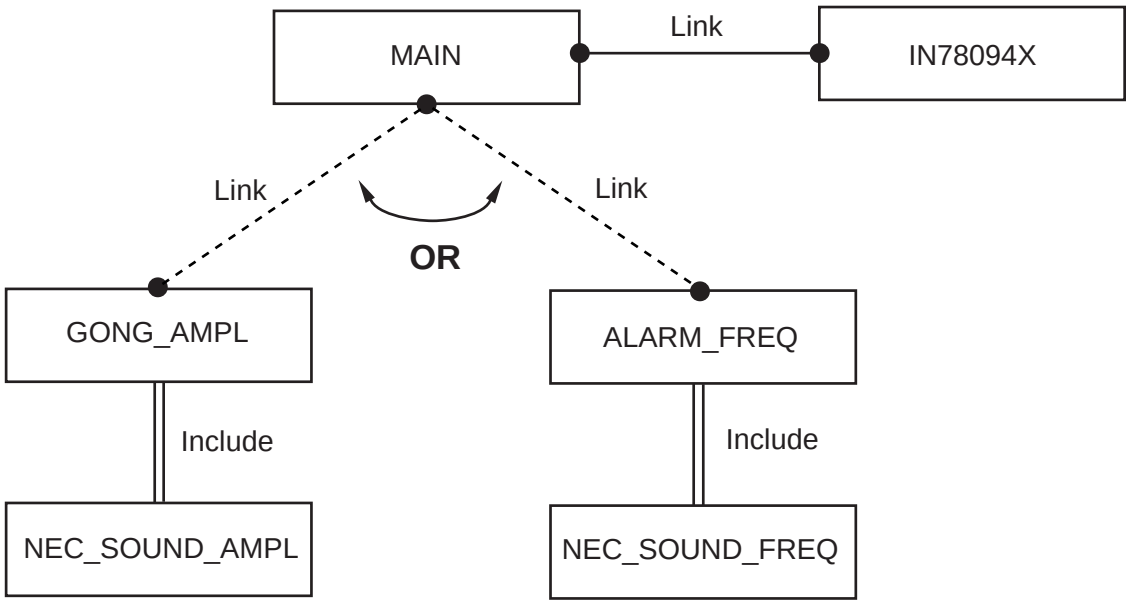
[Memo]

Chapter 2 Examples

2.1 Alarm & Gong Programs

2.1.1 Organisation

Figure 2-1: Organisation



Remarks : You must link the MAIN and IN78094X in all the cases.
If you want to produce a gong effect, link the MAIN (and IN78094x) and GONG_AMPL files.
If you want to produce an alarm effect link the MAIN (and IN78094x) and ALARM_FREQ files.

Caution : You must be careful with yours includes. For example, NEC_SOUND_AMPL and NEC-SOUND_FREQ must be in the current directory.

2.1.2 The Main

```
//
*************************************************************************
//********************** NEC Electronics ************************
//
*************************************************************************
//
// Name : MAIN.
//
// Date : 3/02/98 (d/m/y).
//
// Descritpion : This module consist of the function Main.
//                 After CPU settings, the function GONG_AMPL(or
//                  ALARM_FREQ) is called.
//                  Warning : This programme use the port 14 to
//                  drive the sound and the frequency,
//                 (you can choose between 2 frequencies).
//
//
// Inputs : None.
// Outputs : None.
// Functions needed : None
//
// Modules Needed : GONGON.C
//
*************************************************************************

// ***** Includes :

# pragma language  = extended

# include <c:\iar\inc\io78094X.H>

# include <c:\iar\inc\in78000.H>

// ***** Code :

//——Declarations :
extern unsigned char gongstatus ;

unsigned char soundselection = 0 ;

//—— Function Protypes :
extern void gong(unsigned char) ;

// Function Main:

void main(void)
{
// Init. CPU
PCC = 0x00 ; // Main clock (fx=8Mhz)
MEM = 0 ;     //
IMS = 0xCF ; //
```

```
// Init. port for gong ON/OFF selection :
PF14 =0 ;         //Port function
PM14 = 0x03 ; // Port 14.0 and 14.1 input mode.
                // Port 14.2 to 14.7 ouput mode.

// Loop

while (1)
   {
   soundselection = 0;

   if (gongstatus == 0)
      {
        if (P14.0 == 1)
          {
          soundselection = P14 & 0x02 ;
          gong(soundselection);
          }
      }
   } // End while

}
```

2.1.3 Function Called to Produce an Amplitude Variation

```
//
*********************************************************************
//************************ NEC ELectronics ************************
//
*********************************************************************
//
// Name : GONG_AMPL
//
// Date : 3/02/98 (d/m/y).
//
// Description : Function to test the sound generator of the
//               µPD78K0948. To use only the amplitude setting.
//
// Inputs : None.
// Outputs : None.
// Needed modules : necsound_Ampl.h       Amplitude table
//             in78094x.C       Interrupt definition
//
//
*********************************************************************

// ***** Includes :

# pragma language  = extended

# include <c:\iar\inc\io78094X.H>

# include <c:\iar\inc\in78000.H>

# include “necsound.h”   // To use the table of amplitude varia-
tions.

// ***** Code :

//—— Global declaration :
unsigned char gongstatus = 0;
extern void NoFunction(void) ;

//—— Prototyps :
void gongISR (void) ;

void gong (unsigned char soundselection)

{
extern void (*pISR_TM50) ();       // Pointer to ISR of Timer 50

// Init sound generator :
SGCR = SGCR | 0x08 ;  // Selects by SGOB bit SGO and PCL outputs

// You can choose the frequency value by the port P14.1
```

```
if (soundselection)
{
SGCR = (SGCR & 0xF8) | 0x04 ; //SGCL2 = 1, SGCL1=SGCL0=0
                                  // min freq = 0,512 KHz
                                  // max freq = 0,963 KHz

SGBR = 0x09 ;                   // SGBR = 9
                                  // => Fsg=600Hz
}
else
{
  SGCR = (SGCR & 0xF8) | 0x02 ;
  SGBR = 0x0E ;  // => Fx=1000Hz
}

SGAM = 0x00 ;  // Set Amplitude to 0.

PM3.4 = 0 ;     // Set SGO pin to ouput mode.

gongstatus = 1 ; // set gongstatus = ON ;

SGCR = SGCR | 0x80 ; // TCE = 1, Sound generator operation enable.


//—— Init gong interrupt :
TCL50 = 0x07 ;             // Timer freq = Fx/512 = 15,625 KHz
TMC50 = TMC50 & 0xBF ;    // To be sure to have TMC50.7 = 0
                             // To clear and start on match with
CR50

CR50 = 0x7A ;              // Interrupt on 15,625KHz/122 => all
7,8ms

pISR_TM50 = gongISR ;

_EI() ;                  // Enabled interrupt

MK1L.6 = 0 ;         // Enable timer 50 interrupt
PR1L.6 = 0 ;             // Enable timer 50 interrupt
TMC50 = TMC50 | 0x80 ;       // Enable timer 50

}
//************************ End Gong *****************************

//
*********************************************************************
// Function:     GongISR
// Parameter:    None
// Description: Transfers sound generator amplitude values from a
look-up
//                table to the respective SG SFR.
//                In a second time this function disable inter-
rupts.
//
*********************************************************************
```

```
void gongISR (void)
{
static unsigned char gongcounter = 0;

if (gongstatus)                              // Gong enabled?
   {
   SGAM = NEC_Sound_3[gongcounter++];         // Set new SG Amplitude

   if (gongcounter > 127)                     // Gong done?
      {
      gongcounter = 0;                       // Reset Gong Counter
      gongstatus  = 0;                       // Disable Gong
      SGCR = SGCR & 0x0f ;                   // Disable Sound Generator
      MK1L = MK1L | 0x40 ;                   // Disable Gong Interrupt
      TMC50 = TMC50 & 0x7F ;                 // Disable Gong Interrupt
Source
      pISR_TM50 = NoFunction ;               // Assign NoFunction as ISR
of TM50
      }
   }

}//************************* End GongISR ************************
```

2.1.4 Table to Produce a Quick Amplitude Variation

```
//
*************************************************************************
//************************* NEC Electromics ***********************
//
*************************************************************************
//
// Name : NEC_SOUND_3
//
// Date : 3/02/98 (d/m/y).
//
// Description : All the 7,8 ms a new value will be store in SGAM.
//       To produce a quick amplitude variation.
//
//
*************************************************************************
const unsigned char NEC_Sound_3[128] =
{
            0x2F,   //    1.  SG Amplitude after      0.0  ms
            0x3F,   //    2.  SG Amplitude after      7.8  ms
            0x4F,   //    3.  SG Amplitude after     15.6  ms
            0x57,   //    4.  SG Amplitude after     23.4  ms
            0x67,   //    5.  SG Amplitude after     31.2  ms
            0x77,   //    6.  SG Amplitude after     39.0  ms
            0x7F,   //    7.  SG Amplitude after     46.8  ms
            0x7F,   //    8.  SG Amplitude after     54.6  ms
            0x7F,   //    9.  SG Amplitude after     62.4  ms
            0x7F,   //   10.  SG Amplitude after     70.2  ms
            0x7F,   //   11.  SG Amplitude after     78.0  ms
            0x7F,   //   12.  SG Amplitude after     85.8  ms
            0x7F,   //   13.  SG Amplitude after     93.6  ms
            0x7F,   //   14.  SG Amplitude after    101.4  ms
            0x7F,   //   15.  SG Amplitude after    109.2  ms
            0x7F,   //   16.  SG Amplitude after    117.0  ms
            0x7F,   //   17.  SG Amplitude after    124.8  ms
            0x7F,   //   18.  SG Amplitude after    132.6  ms
            0x7F,   //   19.  SG Amplitude after    140.4  ms
            0x7F,   //   20.  SG Amplitude after    148.2  ms
            0x7F,   //   21.  SG Amplitude after    156.0  ms
            0x7F,   //   22.  SG Amplitude after    163.8  ms
            0x7F,   //   23.  SG Amplitude after    171.6  ms
            0x7F,   //   24.  SG Amplitude after    179.4  ms
            0x7F,   //   25.  SG Amplitude after    187.2  ms
            0x7F,   //   26.  SG Amplitude after    195.0  ms
            0x7F,   //   27.  SG Amplitude after    202.8  ms
            0x7F,   //   28.  SG Amplitude after    210.6  ms
            0x7F,   //   29.  SG Amplitude after    218.4  ms
            0x7F,   //   30.  SG Amplitude after    226.2  ms
            0x7F,   //   31.  SG Amplitude after    234.0  ms
            0x7F,   //   32.  SG Amplitude after    241.8  ms
            0x7F,   //   33.  SG Amplitude after    249.6  ms
            0x7F,   //   34.  SG Amplitude after    257.4  ms
            0x7F,   //   35.  SG Amplitude after    265.2  ms
            0x7F,   //   36.  SG Amplitude after    273.0  ms
            0x7F,   //   37.  SG Amplitude after    280.8  ms
            0x7F,   //   38.  SG Amplitude after    288.6  ms
            0x77,   //   39.  SG Amplitude after    296.4  ms
            0x73,   //   40.  SG Amplitude after    304.2  ms
```

```
0x6E,   //  41.  SG Amplitude after   312.0  ms
0x6A,   //  42.  SG Amplitude after   319.8  ms
0x66,   //  43.  SG Amplitude after   327.6  ms
0x62,   //  44.  SG Amplitude after   335.4  ms
0x5D,   //  45.  SG Amplitude after   343.2  ms
0x59,   //  46.  SG Amplitude after   351.0  ms
0x55,   //  47.  SG Amplitude after   358.8  ms
0x52,   //  48.  SG Amplitude after   366.6  ms
0x50,   //  49.  SG Amplitude after   374.4  ms
0x4C,   //  50.  SG Amplitude after   382.2  ms
0x4A,   //  51.  SG Amplitude after   390.0  ms
0x47,   //  52.  SG Amplitude after   397.8  ms
0x44,   //  53.  SG Amplitude after   405.6  ms
0x42,   //  54.  SG Amplitude after   413.4  ms
0x3F,   //  55.  SG Amplitude after   421.2  ms
0x3B,   //  56.  SG Amplitude after   429.0  ms
0x39,   //  57.  SG Amplitude after   436.8  ms
0x37,   //  58.  SG Amplitude after   444.6  ms
0x35,   //  59.  SG Amplitude after   452.4  ms
0x33,   //  60.  SG Amplitude after   460.2  ms
0x31,   //  61.  SG Amplitude after   468.0  ms
0x2F,   //  62.  SG Amplitude after   475.8  ms
0x2D,   //  63.  SG Amplitude after   483.6  ms
0x2A,   //  64.  SG Amplitude after   491.4  ms
0x29,   //  65.  SG Amplitude after   499.2  ms
0x27,   //  66.  SG Amplitude after   507.0  ms
0x26,   //  67.  SG Amplitude after   514.8  ms
0x24,   //  68.  SG Amplitude after   522.6  ms
0x22,   //  69.  SG Amplitude after   530.4  ms
0x21,   //  70.  SG Amplitude after   538.2  ms
0x20,   //  71.  SG Amplitude after   546.0  ms
0x1F,   //  72.  SG Amplitude after   553.8  ms
0x1E,   //  73.  SG Amplitude after   561.6  ms
0x1D,   //  74.  SG Amplitude after   569.4  ms
0x19,   //  75.  SG Amplitude after   577.2  ms
0x18,   //  76.  SG Amplitude after   585.0  ms
0x17,   //  77.  SG Amplitude after   592.8  ms
0x16,   //  78.  SG Amplitude after   600.6  ms
0x15,   //  79.  SG Amplitude after   608.4  ms
0x14,   //  80.  SG Amplitude after   616.2  ms
0x13,   //  81.  SG Amplitude after   624.0  ms
0x12,   //  82.  SG Amplitude after   631.8  ms
0x11,   //  83.  SG Amplitude after   639.6  ms
0x10,   //  84.  SG Amplitude after   647.4  ms
0xF,    //  85.  SG Amplitude after   655.2  ms
0xE,    //  86.  SG Amplitude after   663.0  ms
0xE,    //  87.  SG Amplitude after   670.8  ms
0xD,    //  88.  SG Amplitude after   678.6  ms
0xC,    //  89.  SG Amplitude after   686.4  ms
0xB,    //  90.  SG Amplitude after   694.2  ms
0xA,    //  91.  SG Amplitude after   702.0  ms
0x9,    //  92.  SG Amplitude after   709.8  ms
0x9,    //  93.  SG Amplitude after   717.6  ms
0x8,    //  94.  SG Amplitude after   725.4  ms
0x8,    //  95.  SG Amplitude after   733.2  ms
0x7,    //  96.  SG Amplitude after   741.0  ms
```

```
    0x7,    //   97.  SG Amplitude after   748.8  ms
    0x6,    //   98.  SG Amplitude after   756.6  ms
    0x6,    //   99.  SG Amplitude after   764.4  ms
    0x5,    //  100.  SG Amplitude after   772.2  ms
    0x5,    //  101.  SG Amplitude after   780.0  ms
    0x4,    //  102.  SG Amplitude after   787.8  ms
    0x4,    //  103.  SG Amplitude after   795.6  ms
    0x3,    //  104.  SG Amplitude after   803.4  ms
    0x3,    //  105.  SG Amplitude after   811.2  ms
    0x2,    //  106.  SG Amplitude after   819.0  ms
    0x2,    //  107.  SG Amplitude after   826.8  ms
    0x1,    //  108.  SG Amplitude after   834.6  ms
    0x1,    //  109.  SG Amplitude after   842.4  ms
    0x0,    //  110.  SG Amplitude after   850.2  ms
    0x0,    //  111.  SG Amplitude after   858.0  ms
    0x0,    //  112.  SG Amplitude after   865.8  ms
    0x0,    //  113.  SG Amplitude after   873.6  ms
    0x0,    //  114.  SG Amplitude after   881.4  ms
    0x0,    //  115.  SG Amplitude after   889.2  ms
    0x0,    //  116.  SG Amplitude after   897.0  ms
    0x0,    //  117.  SG Amplitude after   904.8  ms
    0x0,    //  118.  SG Amplitude after   912.6  ms
    0x0,    //  119.  SG Amplitude after   920.4  ms
    0x0,    //  120.  SG Amplitude after   928.2  ms
    0x0,    //  121.  SG Amplitude after   936.0  ms
    0x0,    //  122.  SG Amplitude after   943.8  ms
    0x0,    //  123.  SG Amplitude after   951.6  ms
    0x0,    //  124.  SG Amplitude after   959.4  ms
    0x0,    //  125.  SG Amplitude after   967.2  ms
    0x0,    //  126.  SG Amplitude after   975.0  ms
    0x0,    //  127.  SG Amplitude after   982.8  ms
    0x0     //  128.  SG Amplitude after   990.6  ms

};
```

2.1.5 Function Called to Produce a Frequency Variation

```
//
*********************************************************************
//************************ NEC Electronics ************************
//
*********************************************************************
//
// Name : ALARM_FREQ
//
// Date : 20/05/98 (d/m/y).
//
// Description : Function to test the sound generator of the
//             µPD78K0948. To use only the amplitude setting.
//
// Inputs : None.
// Outputs : None.
// Functions needed :    necsound_Freq.h       Amplitude table
//               in78094x.C       Interrupt definition

//
//
*********************************************************************

// ***** Includes :

# pragma language  = extended

# include <c:\iar\inc\io78094X.H>

# include <c:\iar\inc\in78000.H>

# include “losound.h” // to produce the frequency variation

// ***** Code :

//—— Global declaration :
unsigned char gongstatus = 0;
extern void NoFunction(void) ;

//—— Prototypes :
void gongISR (void) ;


void gong (unsigned char soundselection)

{
extern void (*pISR_TM50) ();       // Pointer to ISR of Timer 50

// Init sound generator :
SGCR = SGCR | 0x08 ;  // Selects by SGOB bit SGO and PCL outputs


// You can choose the frequency value by the port P14.1
```

```
if (soundselection)
{
SGCR = (SGCR & 0xF8) | 0x04 ; //SGCL2 = 1, SGCL1=SGCL0=0
                                  // min freq = 0,512 KHz
                                  // max freq = 0,963 Khz

//SGBR = 0x09 ;                   // SGBR = 9
                                  // => Fsg=600Hz
}

else
{
  SGCR = (SGCR & 0xF8) | 0x02 ;
//  SGBR = 0x0E ;  // => Fx=1000Hz
}
SGAM = 0x00 ;  // Set Amplitude to 0.

PM3.4 = 0 ;     // Set SGO pin to output mode.

gongstatus = 1 ; // set gongstatus = ON ;

SGCR = SGCR | 0x80 ; // TCE = 1, Sound generator operation enable.


//—— Init. gong interrupt :
TCL50 = 0x07 ;              // Timer freq = Fx/512 = 15,625 KHz
TMC50 = TMC50 & 0xBF ;    // To be sure to have TMC50.7 = 0
                             // To clear and start on match with
CR50

CR50 = 0x7A ;               // Interrupt on 15,625KHz/122 => all
7,8ms

pISR_TM50 = gongISR ;

_EI() ;                   // Enabled interrupt

MK1L.6 = 0 ;              // Enable Timer 50 interrupt
PR1L.6 = 0 ;

TMC50 = TMC50 | 0x80 ;// Enable Timer 50


}
//************************* End Gong
****************************************

//
*********************************************************************
// Function:     GongISR
// Parameter:    None
// Description: Transfers sound generator amplitude values from a
// look-up table to the respective SG SFR.
// In a second time this function disable interrupts.
//*********************************************************************
```

```
void gongISR (void)
{
static unsigned char gongcounter = 0;

if (gongstatus)                                  // Gong enabled?
  {
  SGAM = 0x63 ;                       //Nec sound_3[gongcounter++];
                                        // Set new SG Amplitude
  SGBR = NEC_Sound_freq[gongcounter++]  ;

  if (gongcounter>127)                    // Gong done?
    {
    gongcounter = 0;                       // Reset Gong Counter
    gongstatus  = 0;                       // Disable Gong
    SGCR = SGCR & 0x0f ;                  // Disable Sound Generator
    MK1L = MK1L | 0x40 ;                  // Disable Gong Interrupt
    TMC50 = TMC50 & 0x7F ;          // Disable Gong Interrupt Source
    pISR_TM50 = NoFunction ;   // Assign NoFunction as ISR of TM50
    }
  }

}//********************** End GongISR ****************************
```

2.1.6 Table to Produce a Quick Frequency Variation

```
//
*********************************************************************
//***************** NEC Electronics France **********************
//
*********************************************************************
//
// Date : 29/05/98 (d/m/y).
//
//
// Description : All the 7,8 ms a new value will be store in
    SGBR.
//
//
*********************************************************************
const unsigned char NEC_Sound_freq[128] =
{
            0x00,   //    1.  Output frequency after     0.0  ms
            0x00,   //    2.  Output frequency after     7.8  ms
            0x00,   //    3.  Output frequency after    15.6  ms
            0x00,   //    4.  Output frequency after    23.4  ms
            0x01,   //    5.  Output frequency after    31.2  ms
            0x01,   //    6.  Output frequency after    39.0  ms
            0x01,   //    7.  Output frequency after    46.8  ms
            0x01,   //    8.  Output frequency after    54.6  ms
            0x02,   //    9.  Output frequency after    62.4  ms
            0x02,   //   10.  Output frequency after    70.2  ms
            0x02,   //   11.  Output frequency after    78.0  ms
            0x02,   //   12.  Output frequency after    85.8  ms
            0x03,   //   13.  Output frequency after    93.6  ms
            0x03,   //   14.  Output frequency after   101.4  ms
            0x03,   //   15.  Output frequency after   109.2  ms
            0x03,   //   16.  Output frequency after   117.0  ms
            0x04,   //   17.  Output frequency after   124.8  ms
            0x04,   //   18.  Output frequency after   132.6  ms
            0x04,   //   19.  Output frequency after   140.4  ms
            0x04,   //   20.  Output frequency after   148.2  ms
            0x05,   //   21.  Output frequency after   156.0  ms
            0x05,   //   22.  Output frequency after   163.8  ms
            0x05,   //   23.  Output frequency after   171.6  ms
            0x05,   //   24.  Output frequency after   179.4  ms
            0x06,   //   25.  Output frequency after   187.2  ms
            0x06,   //   26.  Output frequency after   195.0  ms
            0x06,   //   27.  Output frequency after   202.8  ms
            0x06,   //   28.  Output frequency after   210.6  ms
            0x07,   //   29.  Output frequency after   218.4  ms
            0x07,   //   30.  Output frequency after   226.2  ms
            0x07,   //   31.  Output frequency after   234.0  ms
            0x07,   //   32.  Output frequency after   241.8  ms
            0x08,   //   33.  Output frequency after   249.6  ms
            0x08,   //   34.  Output frequency after   257.4  ms
            0x08,   //   35.  Output frequency after   265.2  ms
            0x08,   //   36.  Output frequency after   273.0  ms
            0x09,   //   37.  Output frequency after   280.8  ms
            0x09,   //   38.  Output frequency after   288.6  ms
            0x09,   //   39.  Output frequency after   296.4  ms
            0x09,   //   40.  Output frequency after   304.2  ms
```

```
0x0A,   //  41.  Output frequency after  312.0  ms
0x0A,   //  42.  Output frequency after  319.8  ms
0x0A,   //  43.  Output frequency after  327.6  ms
0x0A,   //  44.  Output frequency after  335.4  ms
0x0B,   //  45.  Output frequency after  343.2  ms
0x0B,   //  46.  Output frequency after  351.0  ms
0x0B,   //  47.  Output frequency after  358.8  ms
0x0B,   //  48.  Output frequency after  366.6  ms
0x0C,   //  49.  Output frequency after  374.4  ms
0x0C,   //  50.  Output frequency after  382.2  ms
0x0C,   //  51.  Output frequency after  390.0  ms
0x0C,   //  52.  Output frequency after  397.8  ms
0x0D,   //  53.  Output frequency after  405.6  ms
0x0D,   //  54.  Output frequency after  413.4  ms
0x0D,   //  55.  Output frequency after  421.2  ms
0x0D,   //  56.  Output frequency after  429.0  ms
0x0E,   //  57.  Output frequency after  436.8  ms
0x0E,   //  58.  Output frequency after  444.6  ms
0x0E,   //  59.  Output frequency after  452.4  ms
0x0E,   //  60.  Output frequency after  460.2  ms
0x0F,   //  61.  Output frequency after  468.0  ms
0x0F,   //  62.  Output frequency after  475.8  ms
0x0F,   //  63.  Output frequency after  483.6  ms
0x0F,   //  64.  Output frequency after  491.4  ms
0x0F,   //  65.  Output frequency after  499.2  ms
0x0F,   //  66.  Output frequency after  507.0  ms
0x0F,   //  67.  Output frequency after  514.8  ms
0x0F,   //  68.  Output frequency after  522.6  ms
0x0E,   //  69.  Output frequency after  530.4  ms
0x0E,   //  70.  Output frequency after  538.2  ms
0x0E,   //  71.  Output frequency after  546.0  ms
0x0E,   //  72.  Output frequency after  553.8  ms
0x0D,   //  73.  Output frequency after  561.6  ms
0x0D,   //  74.  Output frequency after  569.4  ms
0x0D,   //  75.  Output frequency after  577.2  ms
0x0D,   //  76.  Output frequency after  585.0  ms
0x0C,   //  77.  Output frequency after  592.8  ms
0x0C,   //  78.  Output frequency after  600.6  ms
0x0C,   //  79.  Output frequency after  608.4  ms
0x0C,   //  80.  Output frequency after  616.2  ms
0x0B,   //  81.  Output frequency after  624.0  ms
0x0B,   //  82.  Output frequency after  631.8  ms
0x0B,   //  83.  Output frequency after  639.6  ms
0x0B,   //  84.  Output frequency after  647.4  ms
0x0A,   //  85.  Output frequency after  655.2  ms
0x0A,   //  86.  Output frequency after  663.0  ms
0x0A,   //  87.  Output frequency after  670.8  ms
0x0A,   //  88.  Output frequency after  678.6  ms
0x09,   //  89.  Output frequency after  686.4  ms
0x09,   //  90.  Output frequency after  694.2  ms
0x09,   //  91.  Output frequency after  702.0  ms
0x09,   //  92.  Output frequency after  709.8  ms
0x08,   //  93.  Output frequency after  717.6  ms
0x08,   //  94.  Output frequency after  725.4  ms
0x08,   //  95.  Output frequency after  733.2  ms
0x08,   //  96.  Output frequency after  741.0  ms
```

```
    0x07,   //   97.  Output frequency after   748.8  ms
    0x07,   //   98.  Output frequency after   756.6  ms
    0x07,   //   99.  Output frequency after   764.4  ms
    0x07,   //  100.  Output frequency after   772.2  ms
    0x06,   //  101.  Output frequency after   780.0  ms
    0x06,   //  102.  Output frequency after   787.8  ms
    0x06,   //  103.  Output frequency after   795.6  ms
    0x06,   //  104.  Output frequency after   803.4  ms
    0x05,   //  105.  Output frequency after   811.2  ms
    0x05,   //  106.  Output frequency after   819.0  ms
    0x05,   //  107.  Output frequency after   826.8  ms
    0x05,   //  108.  Output frequency after   834.6  ms
    0x04,   //  109.  Output frequency after   842.4  ms
    0x04,   //  110.  Output frequency after   850.2  ms
    0x04,   //  111.  Output frequency after   858.0  ms
    0x04,   //  112.  Output frequency after   865.8  ms
    0x03,   //  113.  Output frequency after   873.6  ms
    0x03,   //  114.  Output frequency after   881.4  ms
    0x03,   //  115.  Output frequency after   889.2  ms
    0x03,   //  116.  Output frequency after   897.0  ms
    0x02,   //  117.  Output frequency after   912.6  ms
    0x02,   //  119.  Output frequency after   920.4  ms
    0x02,   //  120.  Output frequency after   928.2  ms
    0x01,   //  121.  Output frequency after   936.0  ms
    0x01,   //  122.  Output frequency after   943.8  ms
    0x01,   //  123.  Output frequency after   951.6  ms
    0x01,   //  124.  Output frequency after   959.4  ms
    0x0,    //  125.  Output frequency after   967.2  ms
    0x0,    //  126.  Output frequency after   975.0  ms
    0x0     //  127.  Output frequency after   990.6  ms

};
```

2.1.7 Interrupt Vector Pointers

```
//*************************************************************************
//*************************** NEC Electronic  ******************************
//*************************************************************************
//
//   Application      :    General Purpose
//
//   Module            :    in78094x.c
//   Device            :    uPD78094X
//
//   Description      :    Within this module all interrupt vectors are
//                           assign to dedicated ISRs.
//                           These ISRs contain function calls using
//                           pointers to functions.
//                           This enables the use of different ISRs
//                           within one interrupt vector, without
//                           reprogrammming the device.
//
//                           Furthermore a function called “NoFunction”
//                           is defined.
//
//   Used on Chip         :    All Interrupt Vectors of uPD78(F)094x
//
//
*************************************************************************

//————————————— Include Files ——————————————-
# pragma language  = extended

# include <c:\iar\inc\io78094X.H>

# include <c:\iar\inc\in78000.H>
//————————————— Function Prototyps ———————————————
void NoFunction (void);

//————————————— Global Definitions ———————————————-
//— Pointer to ISRs —-
void (*pISR_WDT) ()       = NoFunction;
void (*pISR_AD) ()        = NoFunction;
void (*pISR_OVF) ()       = NoFunction;
void (*pISR_TM20) ()      = NoFunction;
void (*pISR_TM21) ()      = NoFunction;
void (*pISR_TM22) ()      = NoFunction;
void (*pISR_P0) ()        = NoFunction;
void (*pISR_P1) ()        = NoFunction;
void (*pISR_P2) ()        = NoFunction;
void (*pISR_P3) ()        = NoFunction;
void (*pISR_P4) ()        = NoFunction;
void (*pISR_CE) ()        = NoFunction;
void (*pISR_CR) ()        = NoFunction;
void (*pISR_CT0) ()       = NoFunction;
void (*pISR_CT1) ()       = NoFunction;
void (*pISR_CSI0) ()      = NoFunction;
void (*pISR_CSI1) ()      = NoFunction;
```

```
void (*pISR_SER) ()       = NoFunction;
void (*pISR_SR) ()        = NoFunction;
void (*pISR_ST) ()        = NoFunction;
void (*pISR_TM00) ()      = NoFunction;
void (*pISR_TM01) ()      = NoFunction;
void (*pISR_TM50) ()      = NoFunction;
void (*pISR_TM51) ()      = NoFunction;
void (*pISR_WE) ()        = NoFunction;
void (*pISR_WTI) ()       = NoFunction;
void (*pISR_WT) ()        = NoFunction;
void (*pISR_BRK) ()       = NoFunction;

//————————————  Interrupts Definitions ————————————————
interrupt [INTWDT_vect] void ISRWDT(void)        {(*pISR_WDT) ();}
interrupt [INTAD_vect] void ISRAD(void)          {(*pISR_AD) ();}
interrupt [INTOVF_vect] void ISROVF(void)        {(*pISR_OVF) ();}
interrupt [INTTM20_vect] void ISRTM20(void)      {(*pISR_TM20) ();}
interrupt [INTTM21_vect] void ISRTM21(void)      {(*pISR_TM21) ();}
interrupt [INTTM22_vect] void ISRTM22(void)      {(*pISR_TM22) ();}
interrupt [INTP0_vect] void ISRP0(void)          {(*pISR_P0) ();}
interrupt [INTP1_vect] void ISRP1(void)          {(*pISR_P1) ();}
interrupt [INTP2_vect] void ISRP2(void)          {(*pISR_P2) ();}
interrupt [INTP3_vect] void ISRP3(void)          {(*pISR_P3) ();}
interrupt [INTP4_vect] void ISRP4(void)          {(*pISR_P4) ();}
interrupt [INTCE_vect] void ISRCE(void)          {(*pISR_CE) ();}
interrupt [INTCR_vect] void ISRCR(void)          {(*pISR_CR) ();}
interrupt [INTCT0_vect] void ISRCT0(void)        {(*pISR_CT0) ();}
interrupt [INTCT1_vect] void ISRCT1(void)        {(*pISR_CT1) ();}
interrupt [INTCSI0_vect] void ISRCSI0(void)      {(*pISR_CSI0) ();}
interrupt [INTCSI1_vect] void ISRCSI1(void)      {(*pISR_CSI1) ();}
interrupt [INTSER_vect] void ISRSER(void)        {(*pISR_SER) ();}
interrupt [INTSR_vect] void ISRSR(void)          {(*pISR_SR) ();}
interrupt [INTST_vect] void ISRST(void)          {(*pISR_ST) ();}
interrupt [INTTM00_vect] void ISRTM00(void)      {(*pISR_TM00) ();}
interrupt [INTTM01_vect] void ISRTM01(void)      {(*pISR_TM01) ();}
interrupt [INTTM50_vect] void ISRTM50(void)      {(*pISR_TM50) ();}
interrupt [INTTM51_vect] void ISRTM51(void)      {(*pISR_TM51) ();}
interrupt [INTWE_vect] void ISRWE(void)          {(*pISR_WE) ();}
interrupt [INTWTI_vect] void ISRWTI(void)        {(*pISR_WTI) ();}
interrupt [INTWT_vect] void ISRWT(void)          {(*pISR_WT) ();}
interrupt [BRK_I_vect] void ISRBRK(void)         {(*pISR_BRK) ();}

//
*************************************************************************
// Function:     NoFunction
// Parameter:    None
// Description: Dummy Function -> No Operation
//
*************************************************************************
void NoFunction (void)
{
_NOP();
}//************************** End NoFunction **************************
```

2.2 Program to Execute "Au Clair de la Lune"

```
//
*********************************************************************
//****************************** NEC Electronics*******************
//
*********************************************************************
//
// Name : SERENADE
//
// Date : 20/05/98 (d/m/y).
// Update : 29/05/98.
//
// Description : Demonstration of the sound generator of the
// µPD78K0948 - How to set frequency of the output signal.
//
// Inputs : None.
// Outputs : None.
// Functions needed : None
//
//*********************************************************************

// ***** Includes :

# pragma language  = extended

# include <c:\iar\inc\io78094X.H>

# include <c:\iar\inc\in78000.H>

// ***** Code :

void main (void)
{

int i,k ;
int j = 0;

// —— Init. CPU ——-

// Main clock = fx and main clock enable :
PCC = 0x00 ;
// fx = 8 Mhz :
BRPRS = 0x00 ;
// Set CPU to internal memory :
MEM = 0 ;
// Set memory size for D78F0948 :
IMS = 0xCF ;

// —— Init. Amplitude register ——-
// Buzzer output amplitude :
// This demonstration will not play with the sound amplitude.
// Setting of SGAM to have an amplitude of 100 about a maximum of
// 128.
SGAM = 0x63 ;

// —— Init. port ——-
// Set SGO pin to output mode :
PM3.4 = 0 ;
```

```
// —— Start of the sound demonstration ——-

// Loop to repeat the first sound three times :
while ( j < 12 )
{
  i = 0 ;

   if ( (j < 3) || (j==7) || (j==11) )
   {
   // * 1st Sound *
   // —— Init. frequency ——-
   // Maxi and mini value of frequency buzzer output:
   // Settings of SGCL0, SGCL1 and SGCL2
   // Max freq = 0,460 Hz
   // Min freq = 0,244 Hz, with fx = 8 Mhz.
     SGCR.0 = 0 ;
     SGCR.1 = 1 ;
     SGCR.2 = 1 ;


   // Buzzer output frequency Fsg = 0.244 Khz :
   // Fsg = [2e(SGCL0-SGCL1-2*SGCL2-7)]*[Fx/(SGBR+17)]
     SGBR = 0x0F ;

   // TCE = 1 to start the sound generator operation :
     SGCR.7 = 1 ;

     while (i<30000) i++ ;
   // TCE = 0 to stop the sound generator :
     SGCR.7 = 0 ;

   // Wait loop
     i = 0 ;

     while (i<20000) i++ ;
   }

   if ( (j==4) || (j==6) || (j==9) || (j==10) )
   {
     //  * 2nd Sound *
     i=0 ;

     // Max freq = 3,677 Khz
     // Min freq = 1,953 Khz
     SGCR.0 = 0 ;
     SGCR.1 = 1 ;
     SGCR.2 = 1 ;

     // Output frequency = 2,604 KHz :
     SGBR = 0x0B ;

     // Start
     SGCR.7 = 1 ;

     i = 0 ;
     while (i<30000) i++ ;
     // Stop
     SGCR.7 = 0 ;
```

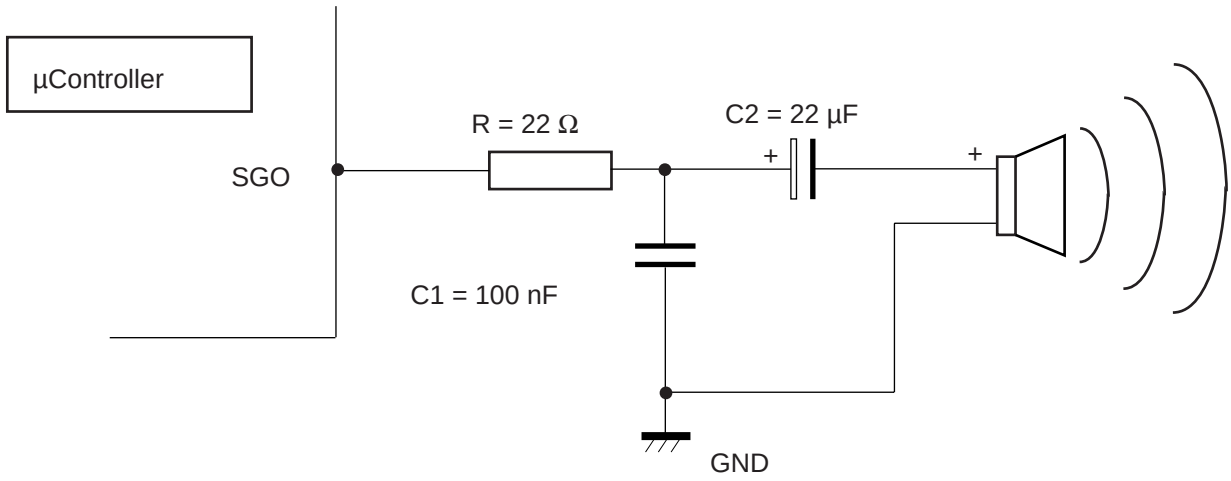
```
      // Wait
      i = 0 ;
      while (i<20000) i++ ;

    }

    if ( (j==5) || (j==8) )
    {
    // * 3rd Sound *
     // Max freq = 7,354 Khz
     // Min freq = 3,906 Khz
       SGCR.0 = 0 ;
       SGCR.1 = 1 ;
       SGCR.2 = 1 ;

     // Output frequency = 5 Khz
        SGBR = 0x08 ;

        SGCR.7 = 1 ;

        i = 0 ;
        while (i<30000) i++;
        SGCR.7 = 0 ;

     // Wait loop
       i = 0 ;
       while (i<20000)  i++;
    }

   j++ ;
   }
}
```

[Memo]

Chapter 3 Hardware

We can remark that the output sound is a PWM. So we need to eliminate the very high and the very low frequencies to reproduce a wave and to protect the speaker.
The following picture shows how to built a security circuit by filtering.

Figure 3-1: Security Circuit by Filtering



[Memo]

NEC

Facsimile Message

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

From:

Name

Company

Tel. FAX

Address

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-551-0451

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑

CS 95.12