NEC

User's Manual

78K0R - Say it!

Demonstration Kit for the 78K0R 16-bit microcontroller family

Document No. U19237EE1V0UM00
Date Published June 2008

- **The information in this document is current as of June, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC 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 NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".
  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
  "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.
  "Special": 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, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11-1

CAUTION

This is a Test- and Measurement equipment with possibility to be significantly altered by user through hardware enhancements/modifications and/or test or application software. Thus, with respect to Council Directive 89/336/EEC (Directive on compliance with the EMC protection requirements), this equipment has no autonomous function. Consequently this equipment is not marked by the CE-symbol.

EEDT-ST-005-10

CAUTION

This equipment should be handled like a CMOS semiconductor device. The user must take all precautions to avoid build-up of static electricity while working with this equipment. All test and measurement tool including the workbench must be grounded. The user/operator must be grounded using the wrist strap. The connectors and/or device pins should not be touched with bare hands.

EEDT-ST-004-10

For customers in the European Union only

Redemption of Waste Electrical and Electronic Equipment (WEEE) in accordance with legal regulations applicable in the European Union only: This equipment (including all accessories) is not intended for household use. After use the equipment cannot be disposed of as household waste. NEC Electronics (Europe) GmbH offers to take back the equipment. All you need to do is register at http://www.eu.necel.com/weee

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: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 0
Fax: 0211-65 03 1327

Sucursal en España
Madrid, Spain
Tel: 091- 504 27 87
Fax: 091- 504 28 60

Succursale Française
Vélizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

Filiale Italiana
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

Branch The Netherlands
Eindhoven, The Netherlands
Tel: 040-244 58 45
Fax: 040-244 45 80

Branch Sweden
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

United Kingdom Branch
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

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.
Singapore
Tel: 65-6253-8311
Fax: 65-6250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Guarulhos, Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

Revision History

Date	Revision	Chapter	Description
17-06-2008	V1.00	---	First release

Table of Contents

List of Figures

List of Tables

1. Introduction

78K0R - Say it! is a demonstration kit for the NEC 78K0R 16-bit microcontroller family. It allows the development of an sound system based on the 78K0R/KG3 device. It supports onboard debugging and real time execution of application programs. The board is prepared to be connected to user hardware parts such as digital I/O or analog signals.

1.1 Main features of *78K0R – Say it!*

- Easy to use device demonstration capabilities
  78K0R - Say it! contains elements to easily demonstrate simple I/O-functions, i.e. navigator switch, a photo-IC illuminance sensor, I/O lines, analog inputs and outputs, UART serial interface etc.

- On-Board debug function (TK-78K0R debugging)
  The *78K0R – Say it!* supports an On-Board debug function by using the IAR C-SPY debugger without a need of additional debug hardware. It allows FLASH downloading and standard debug functions like code execution, single stepping, breakpoints, memory manipulation etc.

- Audio parts
  The *78K0R - Say it!* provides an audio filter, amplifier, onboard speaker and additional a loudspeaker connector to build up sound systems based on the 78K0R/KG3 device.

- Power supply by USB interface

- Analog to digital signal conversion

- Digital to analog signal conversion

- Various input / output signals available, such as

  - I/O ports prepared to be connected to user hardware
  - Timer input / output signals
  - Two or three wire serial I/O
  - Virtual UART interface, via the µPD78F0731 78K0 8-bit microcontroller with on-board USB interface
  - 16 analog input lines
  - 2 analog output lines
  - Navigation switch prepared for key interrupt generation

- The IAR Embedded Workbench for 78K and the IAR C-SPY debugger / simulator are included. These packages are restricted in such that maximum program code size is limited to 16 kByte.

- Full documentation is included for the NEC 78K0R/KG3 microcontroller, IAR Systems Embedded Workbench and IAR Systems C-SPY debugger / simulator.

***78K0R – Say it!* is not intended for code development. NEC does not allow and does not support in any way any attempt to use *78K0R - Say it!* in a commercial or technical product.**

1.2 System requirements

HOST PC	A PC supporting Windows 2000 or Windows XP is required for the IAR Systems Embedded Workbench demo-version. A Pentium processor with at least 1 GHz CPU performance, with at least 256 Mbytes of RAM, allowing you to fully utilize and take advantage of the product features. 500 Mbytes of free disk space, and an additional 10 Mbytes of free disk space on the Windows system drive. A web browser and Adobe Acrobat Reader to be able to access all the product documentation.
Host interface	USB interface that enables communication based on USB (Ver1.1 or later)

1.3 Package contents

Please verify that you have received all parts listed in the package contents list attached to the *78K0R - Say it!* package. If any part is missing or seems to be damaged, please contact the dealer from whom you received your *78K0R - Say it!*.

Note: Updates of the IAR Embedded Workbench for 78K, documentation and/or utilities for *78K0R - Say it!*, if available, may be downloaded from the NEC WEB page(s) at **http://www.eu.necel.com/updates**

1.4 Trademarks

IAR Embedded Workbench, visualSTATE, IAR MakeApp and C-SPY are registered trademarks of IAR Systems AB. Microsoft and Windows are registered trademarks of Microsoft Corporation. Adobe and Acrobat Reader are registered trademarks of Adobe Systems Incorporated.

All other product names are trademarks or registered trademarks of their respective owners.

2. *78K0R - Say it!* system configuration

The *78K0R - Say it!* system configuration is given in the diagram below:

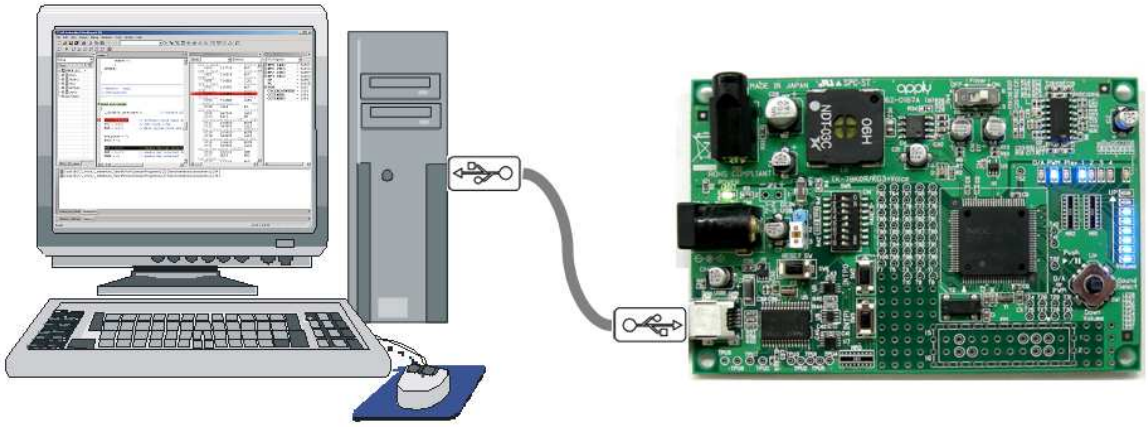


Figure 1: *78K0R - Say it!* system configuration

2.1 78K0R - Say it!

78K0R – Say it! is a demonstration kit for the 78K0R/KG3 16-bit microcontroller of the 78K0R family. The demonstration board is connected to the host system via USB interface cable. The host system may be used for On-Chip debugging by using the IAR C-SPY debugger and to allow execution of application programs on *78K0R – Say it!* starterkit.

78K0R - Say it! runs the microcontroller at 20 MHz operating speed. The sub-clock is provided with 32.768 kHz.

2.2 Host computer

The USB host interface enables communication to the *78K0R - Say it!* board. The µPD78F0731 78K0 8-Bit microcontroller with on-chip USB interface and the NEC virtual UART driver allows application software to access the USB device in the same way as it would access a standard RS232 interface. The NEC virtual UART driver appears to the windows system as an extra Com Port, in addition to any existing hardware Com Ports.

2.3 Power supply via USB interface

The *78K0R - Sayl it!* board is powered by the USB interface. Optional the power supply can be applied via the connectors JP2 or CN1.

3. *78K0R - Say it!* components

The *78K0R - Say it!* board is equipped with a navigation switch, a loudspeaker, a photo-IC illuminance sensor, LED's and with several connectors in order to be connected to host computers, FLASH programmer or any external target hardware.

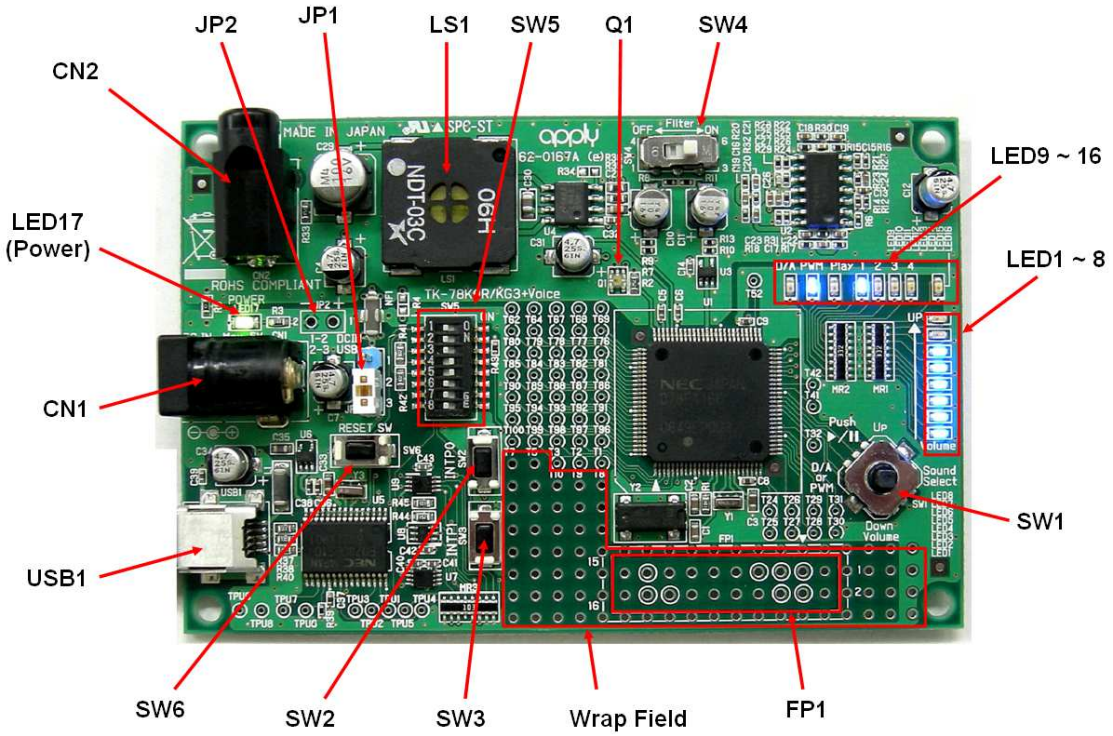
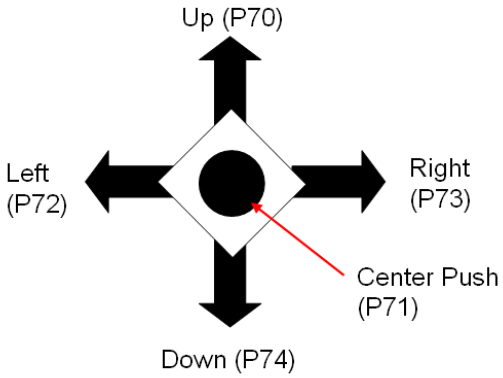


Figure 2: *78K0R - Say it!* board connectors and switches

Some of the *78K0R – Say it!* components are free for user application hardware and software. Please read the user's manual of the 78K0R/KG3 device carefully to get information about the electrical specification of the available I/O ports before you connect any external signals to the *78K0R – Say it!* board.

3.1 SW1, Navigation switch

Button SW1 is a navigation switch connected to the key interrupt pins of the 78K0R/KG3 device. It operates in four directions and has a center push function. When the navigation switch is moved to one of the four directions or it is pushed a low-level signal (Vss) is applied to the corresponding pin of the 78K0R/KG3 device. The connection of SW1 to the microcontroller is shown in the table below:



SW1	Connection to the 78K0R/KG3 device
Left	P72/EX18/KR2
Down	P74/EX20/KR4/INTP8
Center Push	P71/EX17/KR1
Right	P73/EX19/KR3
Up	P70/EX16/KR0

Figure 3: Navigation switch SW1

For information about the pull-up resistor setting of the corresponding port, please refer to the user's manual of the 78K0R/KG3 device.

3.2 SW2, Switch (INTP0)

SW2 is a push button connecting VSS to external interrupt input INTP0 of the microcontroller. This is equal to port "P120/INTP0/EXLVI" of the 78K0R/KG3 device. The port may be programmed to generate the external interrupt INTP0. The necessary initialisation for this purpose is described in the user's manual of the 78K0R/KG3 device. Please note, when using SW2 turn ON the built-in pull-up resistor of the 78K0R/KG3 device, register PU12.

3.3 SW3, Switch (INTP1)

SW3 is a push button connecting VSS to external interrupt input INTP1 of the microcontroller. This is equal to port " P46/INTP1/TI05/TO05" of the 78K0R/KG3 device. The port may be programmed to generate the external interrupt INTP1. The necessary initialisation for this purpose is described in the user's manual of the 78K0R/KG3 device. Please note, when using SW3 turn ON the built-in pull-up resistor of the 78K0R/KG3 device, register PU4.

3.4 SW4, Switch (Filter)

SW4 is the slide switch to select if the onboard Filter circuit (LMV324M) should be used or not. If SW4 is set to “OFF”, the Filter is not used and the sound output signal of the microcontroller is directly connected to the amplifier.

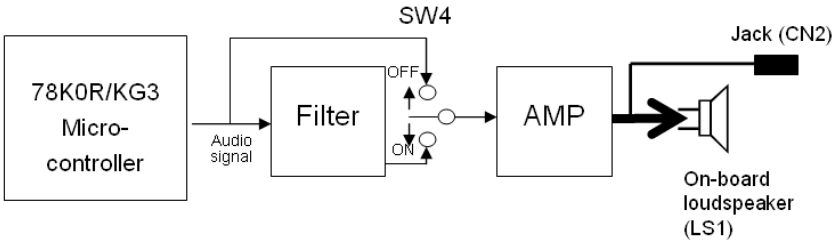


Figure 4: Switch SW4, Filter / Amplifier circuit

3.5 SW5, Configuration switch

The different operation modes of the *78K0R - Say it!* board can be set by switch SW5. The bits 1-5 of switch SW5 are for the mode setting of the board and bits 6-8 are DIP switches connected to P75-P77 pins of the 78K0R microcontroller and can be used for user application purpose.

3.5.1 SW5 bits1-5, On-Board debug mode (TK-78K0R debugging)

To use the TK-78K0R On-Board debugging function of the *78K0R- Say it!* board please set switch SW5 bits 1-5 to the following configuration.

SW5/bit	Configuration
1	ON / OFF (*)
2	ON
3	ON
4	OFF
5	OFF

Table 1: On-Board debug mode setting, switch SW5

(*) = When bit1 is set to “ON” the microcontroller stays in reset state till TK-78K0R debugging is started.

When bit 1 is set to “OFF” the microcontroller immediately starts code execution from the internal FLASH memory after power is applied to the board.

Note: After changing the configuration of SW5 bits1-5 it is necessary to power-up the *78K0R - Say it!* board to make changing active. This can be done by simply dis- and re-connecting the USB interface cable.

3.5.2 SW5 bits1-5, Stand alone mode

To run a program stored in built-in flash memory of the 78K0R/KG3 device without using the On-Board debugging mode please set switch SW5 bits 1-5 to the following configuration. Additionally, when using PG-FP4 for FLASH programming or QB-MINI2 for debugging purpose please use the same configuration.

SW5/bit	Configuration
1	OFF
2	OFF
3	OFF
4	OFF / ON (*)
5	OFF / ON (*)

Table 2: Stand alone mode setting, switch SW5

(*) = When setting bits 4-5 to “ON” the UART3 signals RxD3 and TxD3 or the 78K0R/KG3 device are connected to the µPD78F0731 USB microcontroller. Within this mode standard serial communication to a terminal program running on the HOST PC can be established.

Note: After changing the configuration of SW5 bits1-5 it is necessary to power-up the *78K0R - Say it!* board to make changing active. This can be done by simply dis- and re-connecting the USB interface cable.

3.5.3 SW5 bits6-8, General purpose switches

The bits 6-8 of switch SW5 are connected to the pins P75-P77 of the 78K0R/KG3 microcontroller. They are free for any application purpose. Setting a corresponding bit to “ON” applies low-level (Vss) to the dedicated microcontroller pin. Please note, when using SW3 turn ON the built-in pull-up resistor of the 78K0R/KG3 device, register PU7. For doing so, please refer to the user’s manual of the 78K0R/KG3 device.

SW5/bit	Connection to the 78K0R/KG3 device
6	P75
7	P76
8	P78

Table 3: General purpose switches, switch SW5

3.6 SW6, RESET button

SW6 is the reset button. It activates the power on reset. Switch SW6 controls the reset input signal of the 78K0R/KG3 microcontroller.

3.7 JP1, Power Supply selector

Jumper JP1 is the power supply selector of the *78K0R – Say it!* board.

JP1	Configuration	Mode
1-2	Closed	power supply from AC adapter via connector CN1
2-3	Closed (default)	power supply from USB via connector USB1
1-2-3	Open	power supply from connectors JP2 or FP1

Table 4: Power supply selector, JP1

3.8 JP2, External power connector

By using connector JP2 (not assembled) external power supply can be applied to the *78K0R – Say it!* board without a need of an active USB connection.

JP2	Input
1	VDD (+5V)
2	GND

Table 5: External power connector, JP2

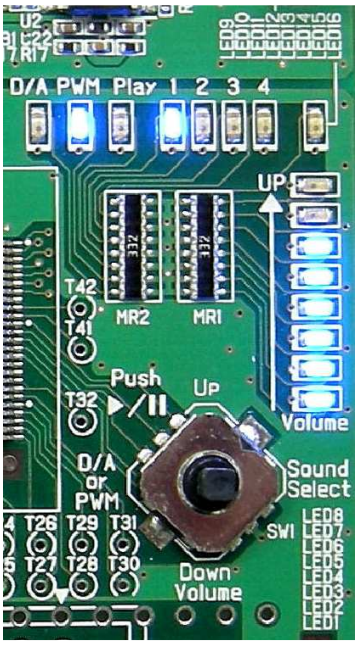
Note: Be sure to unplug the USB connection before applying external power supply to input JP2 and "Open" jumper JP1.

3.9 Photo-IC illuminance sensor, Q1

Q1 is the photo-IC illuminance sensor. It is connected to the A/D converter input "P157/ANI15" of the 78K0R/KG3 microcontroller. The output voltage of the sensor rises in case of the light radiation increases and is falls in case of the light radiation decreases. For details on how to configure the A/D converter of the 78K0R/KG3 microcontroller accordingly, please refer to the user's manual of the device.

3.10 LED1~16, general purpose LEDs

LED1~16 are general purpose LEDs. A low level signal at the corresponding I/O port pin of the 78K0R/KG3 microcontroller switches the LED on.



Label	Location	78K0R/KG3 I/O pin
Volume	LED1	P80/EX0
	LED2	P81/EX1
	LED3	P82/EX2
	LED4	P83/EX3
	LED5	P84/EX4
	LED6	P85/EX5
	LED7	P86/EX6
	LED8	P87/EX7
D/A	LED9	P50/EX8
PWM	LED10	P51/EX9
Play	LED11	P52/EX10
1	LED12	P53/EX11
2	LED13	P54/EX12
3	LED14	P55/EX13
4	LED15	P56/EX14
	LED16	P57/EX15

Table 6: General purpose LEDs, LED1~16

3.11 LED17, power LED

LED17 is the power LED of the *78K0R – Say it!* board. It indicates if power is applied to the *78K0R – Say it!* board.

3.12 CN1, AC power supply connector

CN1 is the AC power supply connector of the *78K0R – Say it!* board. Caution, please connect only a power supply of maximum +5V to the board. There is no voltage regulator assembled on the *78K0R – Say it!* board. Higher supply voltage can damage the board.

CN1	Input
Center	VDD (+5V)
Ring	GND

Note: Be sure to unplug the USB connection before connecting a +5V AC power supply to CN1 and set jumper JP1 to "1-2 closed".

3.13 CN2, external speaker jack

CN2 is the jack for external speakers. You can connect an external speaker to improve the sound quality when playing sounds.

CN2 external speaker jack	
Supported jack	3,5mm (monaural)

Table 7: External speaker jack CN2

3.14 FP1, MINICUBE2 / PG-FP4 connector

Connector FP1 (not assembled) allows connecting the PG-FP4 FLASH programmer to *78K0R - Say it!* board in order to program application software into the 78K0R/KG3 internal flash memory. Please note, the PG-FP4 FLASH programmer is a separate product from NEC and it is not included in this package. Additional FP1 allows connecting the QB-MINI2 On-Chip debug emulator to the *78K0R - Say it!* board in order to use On-Chip debug function of the 78K0R/KG3 device. Please note, QB-MINI2 is a separate product from NEC and it is not included in this starterkit package.

FP1	Signal
1	GND
2	RESET
3	SI
4	VDD
5	SO
6	N.C.
7	N.C.
8	N.C.
9	N.C.
10	N.C.
11	N.C.
12	N.C.
13	N.C.
14	FLMD0
15	RESET_IN
16	CLK_IN

Table 8: PG-FP4 / QB-MINI2 connector FP1

When using PG-FP4 for FLASH programming or QB-MINI2 for debugging purpose, please configure switch SW5 bits1-5 of the *78K0R - Say it!* board as following:

SW5/bit	Configuration
1	OFF
2	OFF
3	OFF
4	OFF / ON (*)
5	OFF / ON (*)

Table 9: Configuration of SW5 bits1-5 when using PG-FP4 or QB-MINI2

(*) = individual selectable by user.

3.15 USB1, serial interface connector

This interface allows connecting the IAR C-SPY debugger to the *78K0R - Say it!* board in order to use the On-Board debug function (TK-78K0R debugging). The TK-78K0R interface supports On-board FLASH erasing / programming and standard debug features like code execution, single stepping, breakpoints, memory manipulation etc.

For standard communication to a host computer - i.e. by using a terminal program - the input/output signals of UART3 of the 78K0R/KG3 device can be redirected to the USB1 connector via the µPD78F0731 USB microcontroller.

The power supply of the *78K0R - Say it!* board is also provided by the USB1 connector.

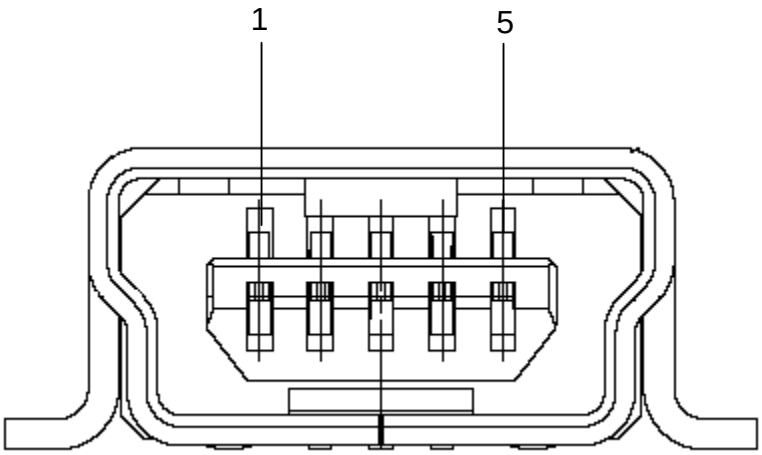


Figure 5: USB1, USB Mini-B Type Host Connector Pin Configuration

Connector USB1	Signal Name
1	VBUS
2	D-
3	D+
4	ID_NC
5	GND

Table 10: Pin Configuration of Connector USB1

For connection with the host machine, use a USB cable (Mini-B type). For confirmation, NEC Electronics used only the USB cable delivered with the *78K0R - Say it!* board.

3.16 Wrap field

For the integration of additional application hardware and user circuits the *78K0R – Say it!* board offers a wrap field. Please read the user's manual of the 78K0R/KG3 device carefully to get information about the electrical specification of the available I/O ports before you connect any external signals to the *78K0R – Say it!* board.

3.17 T1~T100, test pads

Several pins of the 78K0R/KG3 microcontroller are connected to the test pads T1~T100. The corresponding assignment can be found in table below.

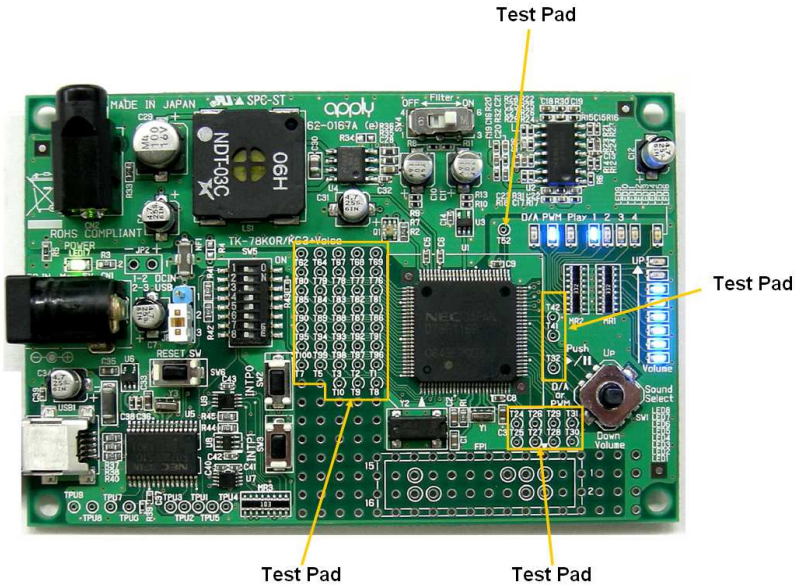


Figure 6: Test pads, T1~T100

Test pad	78K0R/KG3 I/O pin
T1	P142 / SCK20 / SCL20
T2	P141 / PCLBUZ1 / INTP7
T3	P140 / PCLBUZ0 / INTP6
T5	P47 / INTP2
T7	P45 / SO01
T8	P44 / SI01
T9	P43 / SCK01
T10	P42 / TI04 / TO04
T24	P60 / SCL0
T25	P61 / SDA0
T26	P62
T27	P63
T28	P31 / TI03 / TO03 / INTP4
T29	P64 / RD
T30	P65 / WR0
T31	P66 / WR1
T32	P67 / ASTB
T41	P06 / WAIT
T42	P05 / CLKOUT
T52	P30 / INTP3 / RTC1HZ
T62	P17 / EX31 / TI02 / TO02
T64	P15 / EX29 / RTCDIV /RTCCL

Test pad	78K0R/KG3 I/O pin
T67	P12 / EX26 / SO00 / TxD0
T68	P11 / EX25 / SI00 / RxD0
T69	P10 / EX24 / SCK00
T76	P156 / ANI14
T77	P155 / ANI13
T78	P154 / ANI12
T79	P153 / ANI11
T80	P152 / ANI10
T81	P151 / ANI9
T82	P150 / ANI8
T83	P27 / ANI7
T84	P26 / ANI6
T85	P25 / ANI5
T86	P24 / ANI4
T87	P23 / ANI3
T88	P22 / ANI2
T89	P21 / ANI1
T90	P20 / ANI0
T91	P130
T92	P131 / TI06 / TO06
T93	P04 / SCK10 / SCL10
T94	P03 / SI10 / RxD1 / SDA10
T95	P02 / SO10 / TxD1
T96	P01 / TO00
T97	P00 / TI00
T98	P145 / TI07 / TO07
T99	P144 / SO20 / TxD2
T100	P143 / SI20 / RxD2 / SDA20

Table 11: Test pads, T1~T100

4. On-Chip debugging

The *78K0R - Say it!* board offers two possibilities to use On-Chip debugging (OCD). The TK-78K0R On-Board debug function of *78K0R – Say it!* allows On-Chip debugging without a need of external debug hardware. Within this mode the default USB connection to the Host computer based on the virtual UART driver is used as debug interface. All standard debug functions are available in the On-Board debugging mode like FLASH programming / downloading, code execution, single stepping, breakpoints, memory manipulation etc.

Additionally *78K0R – Say it!* supports the QB-MINI2 On-Chip debug emulator in order to use On-Chip debug function of the 78K0R/KG3 device. The system configuration for On-Chip debugging is shown in figure below.

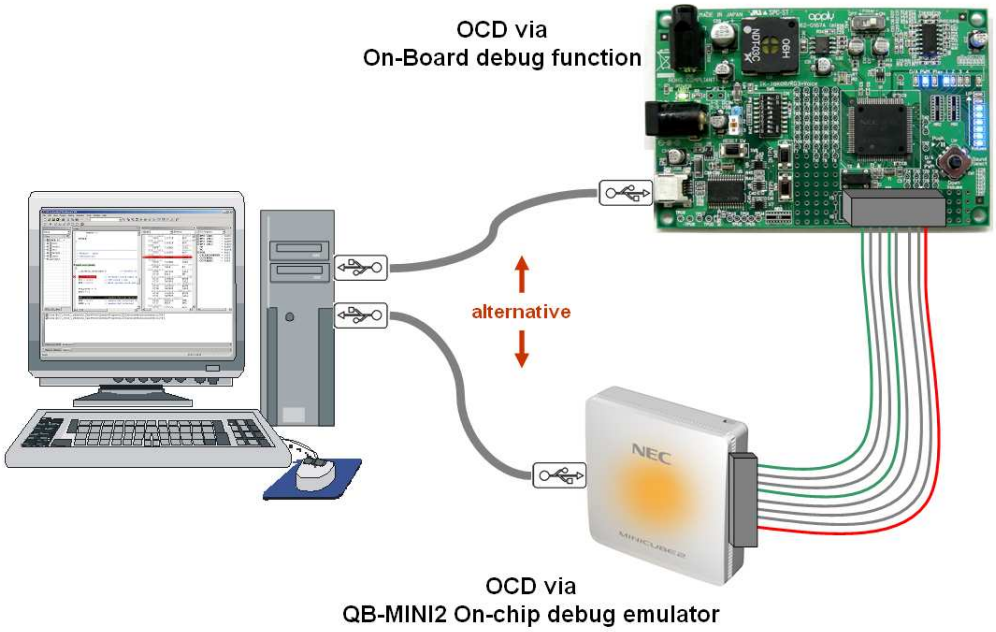


Figure 7: On-Chip debugging

4.1 OCD via TK-78K0R On-Board debug function

To operate the *78K0R - Say it!* board within the On-Board debug mode, configure switch SW5 bits1-5 as following:

SW5/bit	Configuration
1	ON / OFF (*)
2	ON
3	ON
4	OFF
5	OFF

Table 12: OCD via TK-78K0R On-Board debug function

(*) = individual selectable by user.

4.2 OCD via QB-MINI2 emulator

To operate the *78K0R - Say it!* board together with the QB-MINI2 On-Chip debug emulator, configure switch SW5 bits1-5 as following:

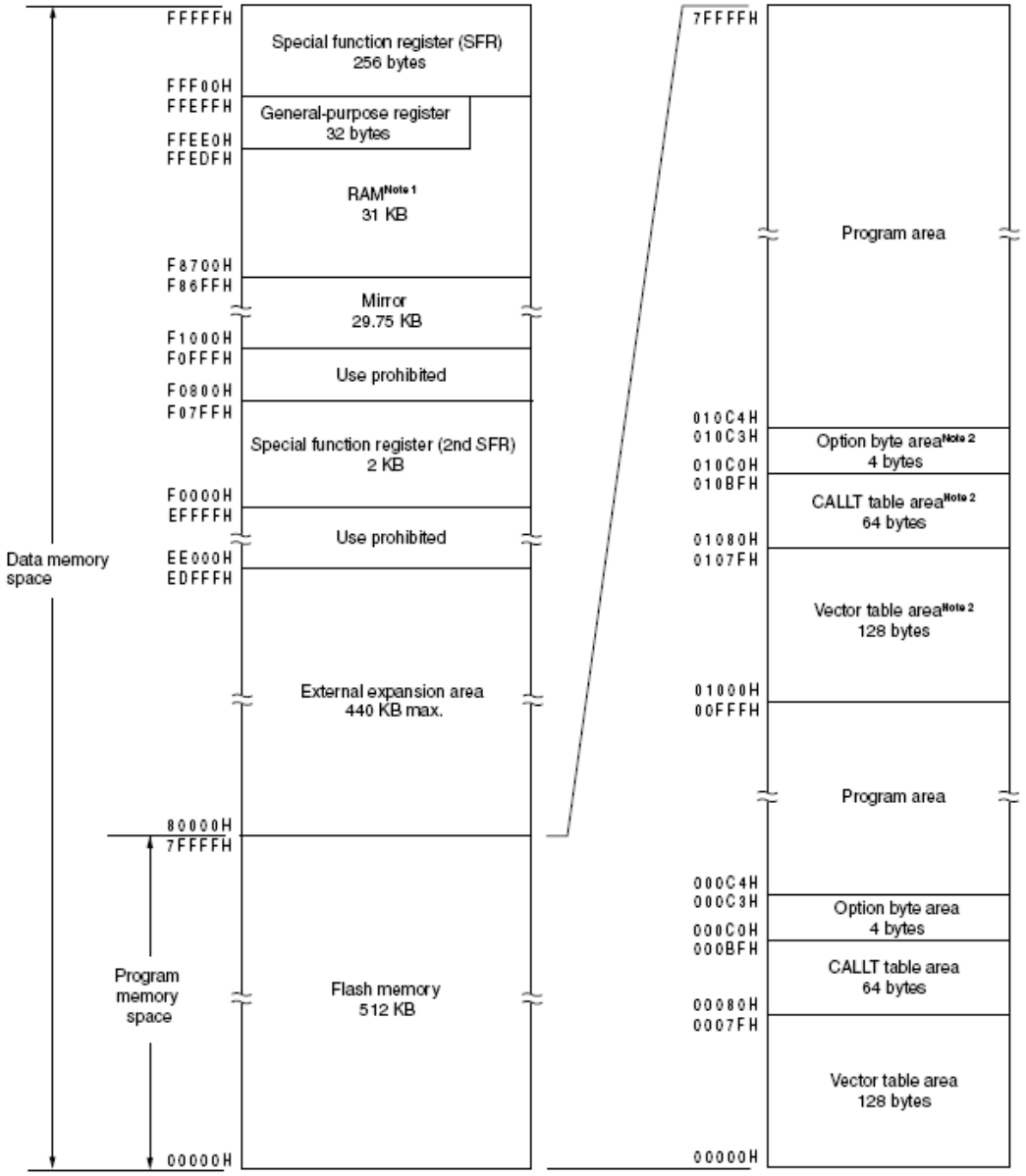
SW5/bit	Configuration
1	OFF
2	OFF
3	OFF
4	OFF / ON (*)
5	OFF / ON (*)

Table 13: OCD via QB-MINI2 emulator

(*) = individual selectable by user.

5. 78K0R/KG3 memory map

The memory layout of 78K0R/KG3 device is shown in the figure below.



Notes 1. Use of the area F8700H to F8EFFH is prohibited when using the self-programming function.

2. When using boot swap, write the contents of 00000H through 00FFFH to 01000H through 01FFFH.

Figure 8: 78K0R/KG3 memory map

The *78K0R – Say it!* does not reserve any resources of the 78K0R/KG3 microcontroller, consequently all available memory of the device is free for application software.

6. *78K0R – Say it!* installation and operation

6.1 Getting started

The IAR Embedded Workbench including the C-SPY debugger allows to build and download application programs to the *78K0R - Say it!* starterkit. As communication interface between the PC host system and the *78K0R - Say it!* board a standard USB interface line is needed. Before you can download and run a program, software and hardware must be installed properly.

6.1.1 CD-ROM contents

The CD-ROM shows following directory structure:

78K0R – Say it! (F:)	CD-ROM ROOT
Acrobat	- Acrobat Reader for 32Bit Windows OS
Doc	- Documentation
IAR	- IAR Embedded Workbench for 78K
SamplePrograms	- Sample programs for 78K0R – *Say it! including:* o *78K0R - Say it! Voice Samples* o *78K0R - Say it! Download Sample* o *CvADPCM conversion / download Tool* o *Sound / Wave samples*

Table 14: *78K0R - Say it!* CD-ROM directory structure

7. Hardware installation

After unpacking *78K0R - Say it!,* connect the board to your host computer using the provided USB interface cable. When *78K0R - Say it!* is connected, the USB driver needs to be installed on the host machine. Please refer to the following **CHAPTER 8 SOFTWARE INSTALLATION**.

8. Software installation

The *78K0R - Say it!* package comes with the following software demo packages:

- IAR Systems Embedded Workbench for 78K, including C compiler, assembler, linker, librarian and IAR C-SPY debugger / simulator
- Sample programs

The IAR Systems Embedded Workbench must be installed on your PC. For detailed installation hints, refer to the following chapters and to the corresponding documentation of the IAR Embedded Workbench.

8.1 IAR Systems Embedded Workbench for 78K installation

To install the IAR Systems Embedded Workbench for 78K0/K0S/K0R including C-SPY debugger / simulator, select the AUTORUN program in the directory \IAR\ of the CDROM. The setup dialogues will guide you through the installation process.

8.2 Sample program installation

To install the sample/demonstration programs for the *78K0R – Say it!* board select the SETUP program in the directory \SamplePrograms\ of the CDROM. The setup dialogues will guide you through the installation process.

8.3 USB Driver Installation

In order to use the *78K0R - Say it!* board for On-Chip debugging the USB driver needs to be installed on the host machine. Install the driver according to the following procedure:

Note: The USB driver is part of the IAR Embedded Workbench software package. Therefore please install the IAR Embedded Workbench first.

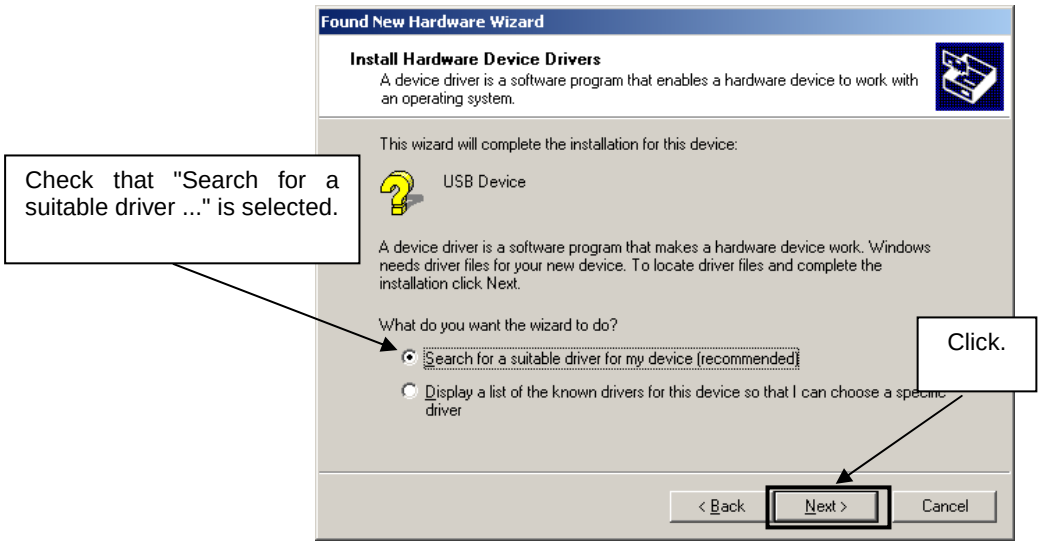
8.3.1 Installation on Windows 2000

1. When the *78K0R - Say it!* board is connected with the host machine, the board is recognized by <Plug and Play>, and the wizard for finding new hardware is started. Click Next>.

Figure 9: Found New Hardware Wizard (Windows 2000)

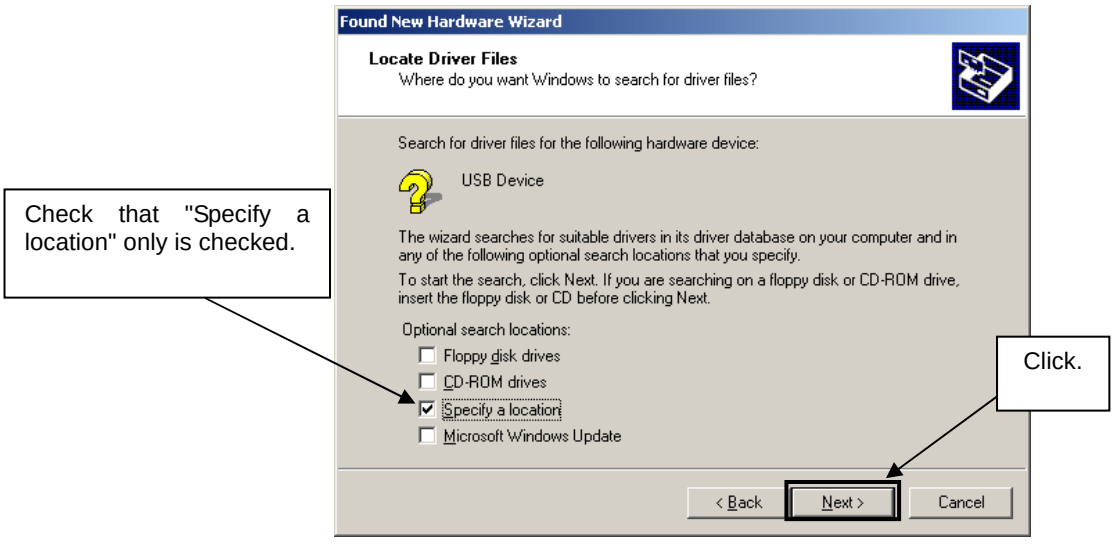
2. Following the window below is displayed. So, check that "Search for a suitable driver ..." is selected, then click Next>.

Figure 10: Search Method (Windows 2000)



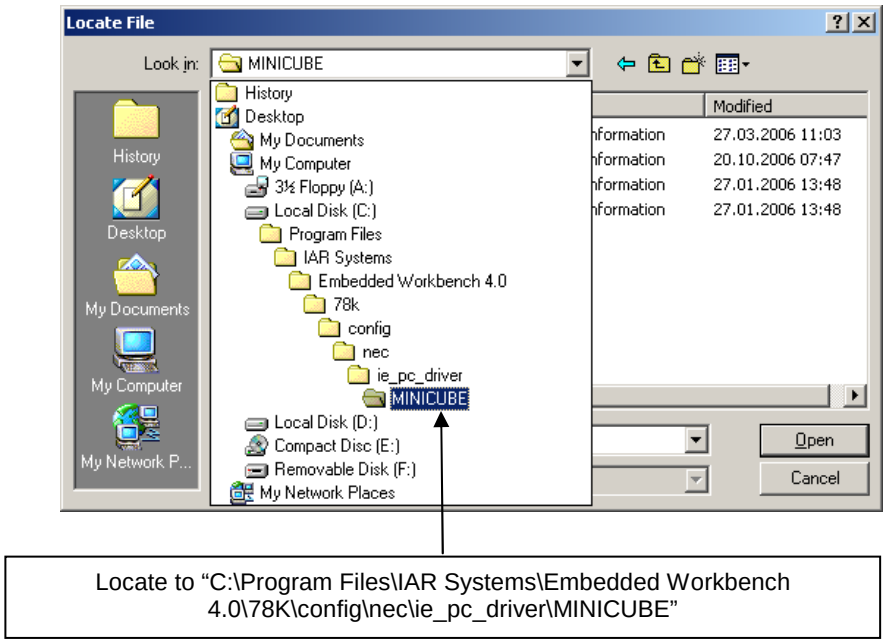
3. Check the "Specify a location" check box only, then click Next>.

Figure 11: Driver File Location (Windows 2000)



4. Locate to the folder "C:\Program Files\IAR Systems\Embedded Workbench 4.0\78K\config\nec\ie_pc_driver\MINICUBE".

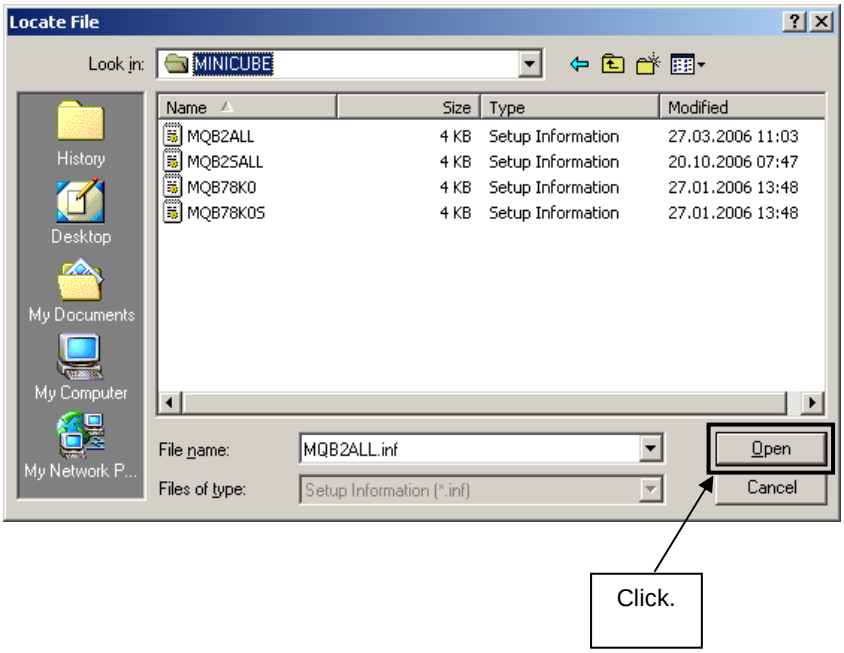
Figure 12: Address Specification 1 (Windows 2000)



Remark If the installation destination folder is changed at the time of IAR Embedded Workbench installation, enter "new-folder\78K\config\nec\ie_pc_driver\MINICUBE".

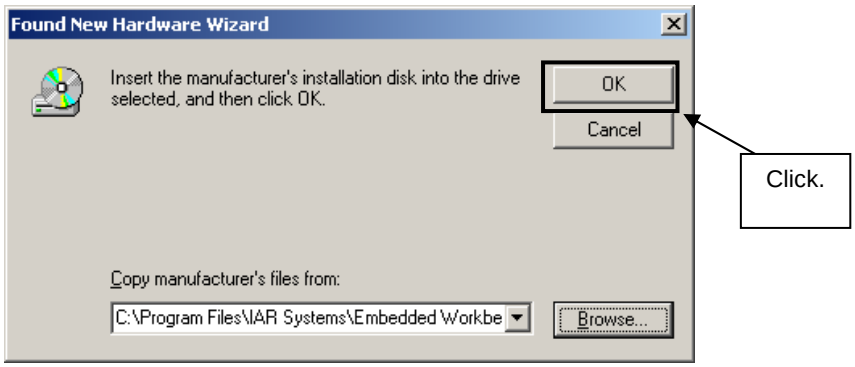
5. The setup information file "MQB2ALL.inf" is automatic selected, then click Open to proceed within driver installation.

Figure 13: Address Specification 2 (Windows 2000)



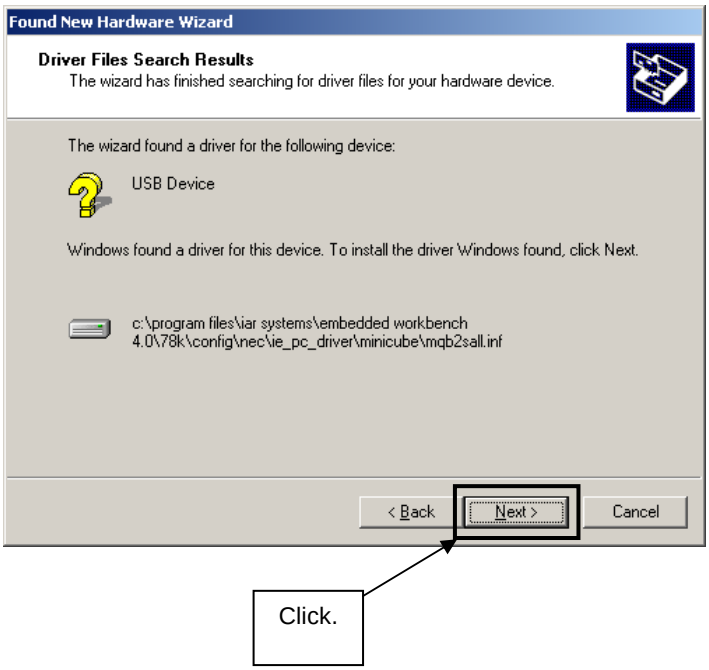
6. After the location of the USB driver has been specified click OK to proceed.

Figure 14: Address Specification 3 (Windows 2000)



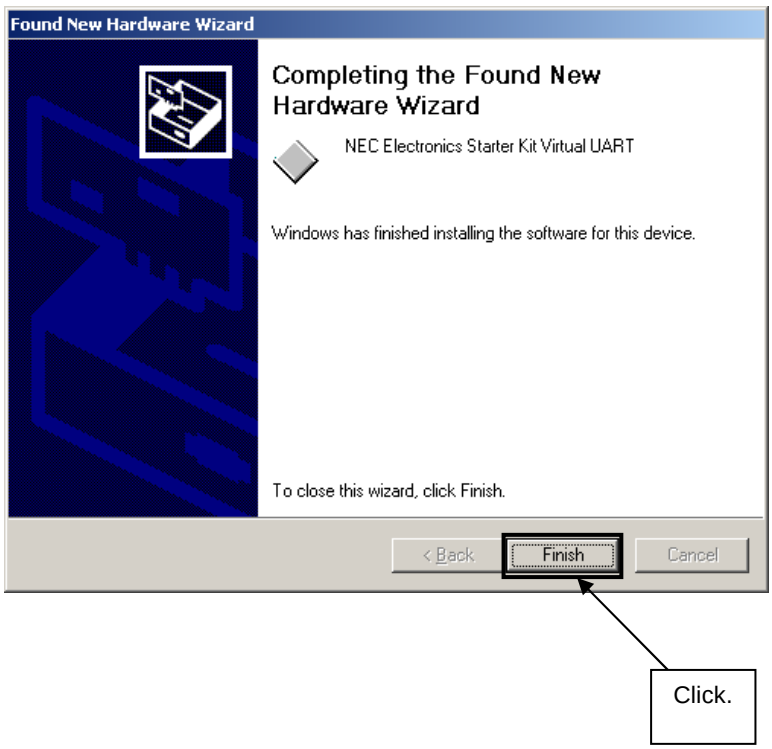
5. Click Next>.

Figure 15: Driver File Search (Windows 2000)



6. Click Finish to complete the installation of the USB driver.

Figure 16: USB Driver Installation Completion (Windows 2000)



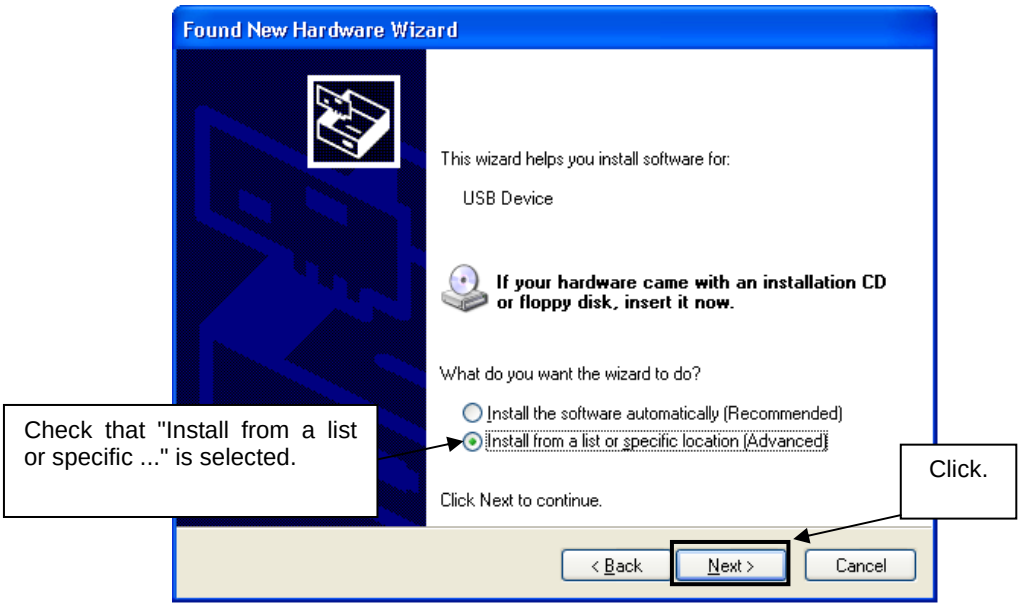
8.3.2 Installation on Windows XP

1. When the *78K0R - Say it!* board is connected with the host machine, the board is recognized by Plug and Play, and the wizard for finding new hardware is started. At first the hardware wizard will ask if windows should search on the windows update web, check "No, not this time" and then click Next>.

Figure 17: Found New Hardware Wizard 1 (Windows XP)

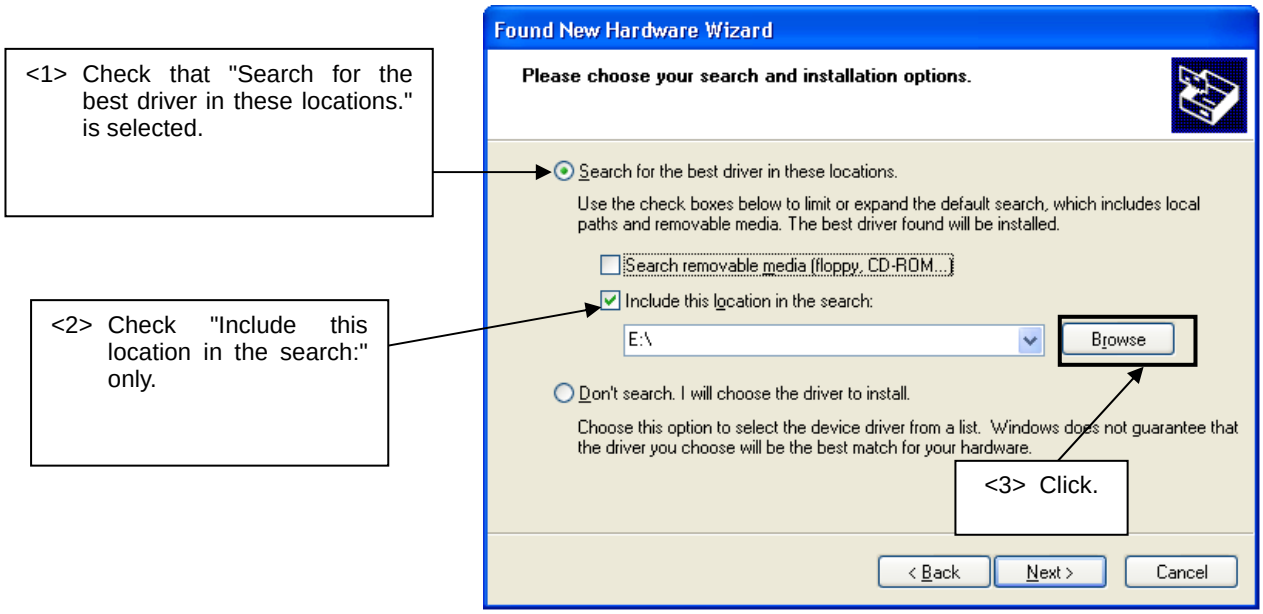
2. Check that "Install from a list or specific location (Advanced)" is selected, then click Next>.

Figure 18: Found New Hardware Wizard 2 (Windows XP)



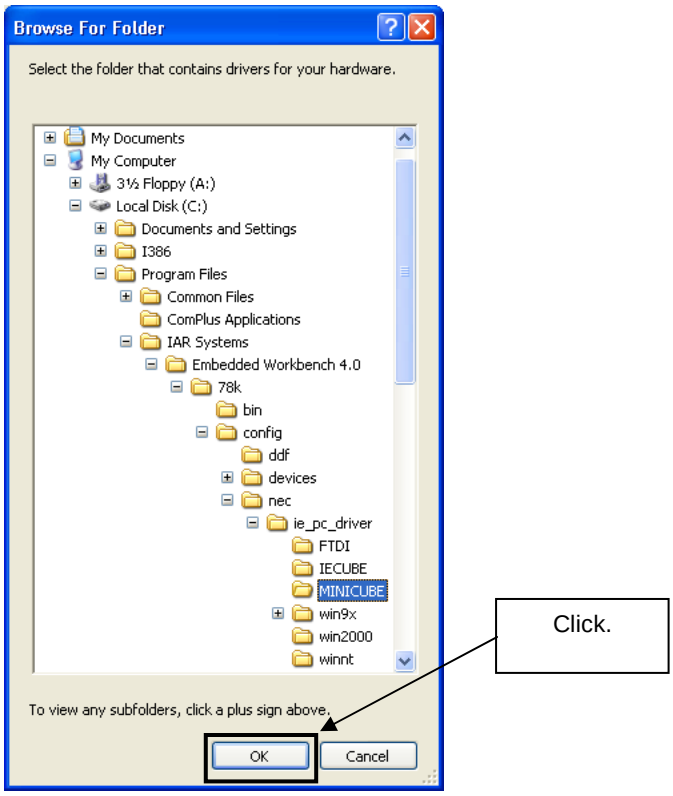
3. Check that "Search for the best driver in these locations." is selected. Select the "Include this location in the search:" check box and then click Browse.

Figure 19: Search Location Specification 1 (Windows XP)



4. Locate the folder "C:\Program Files\IAR Systems\Embedded Workbench 4.0\78K\config\nec\ie_pc_driver\MINICUBE" and click OK.

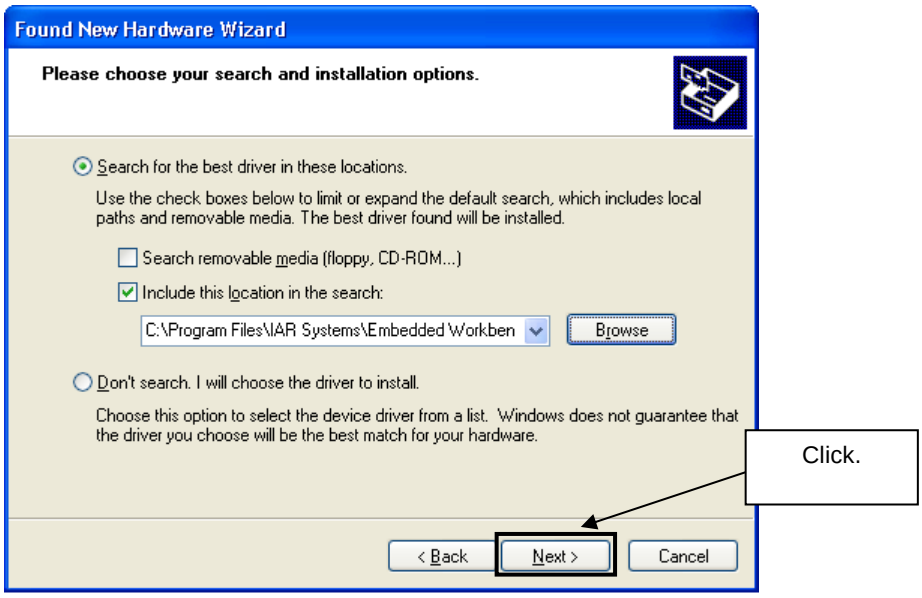
Figure 20: Search Location Specification 2 (Windows XP)



Remark If the installation destination folder is changed at the time of IAR Embedded Workbench installation, enter "new-folder\78K\config\nec\ie_pc_driver\MINICUBE".

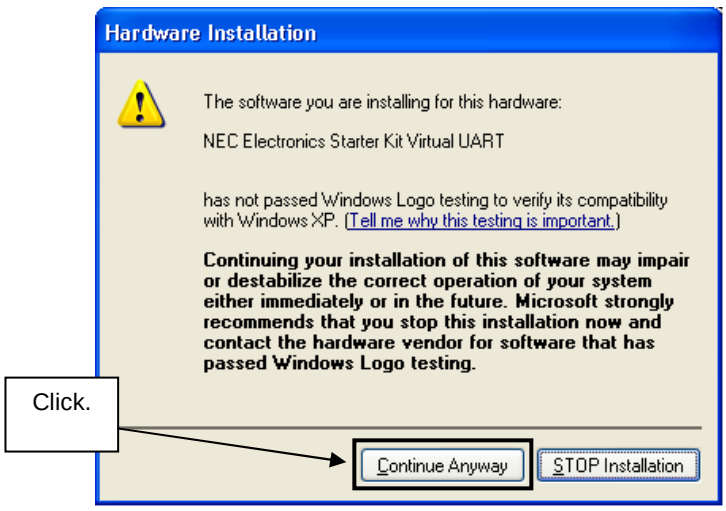
5. After the location of the USB driver has been specified click Next> to continue driver installation.

Figure 21: Search Location Specification 3 (Windows XP)



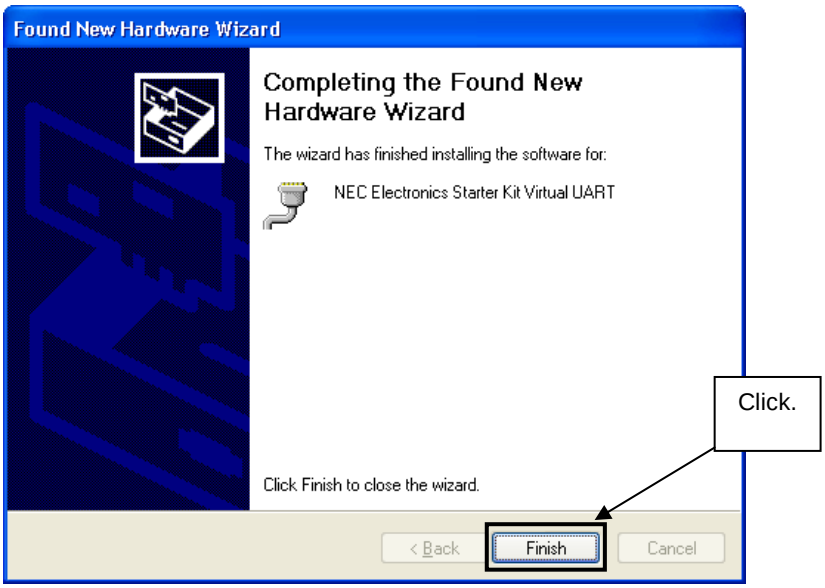
6. As shown below, "NEC Electronics Starter Kit Virtual UART has not passed Windows Logo testing to verify its compatibility with Windows XP." is displayed. Click Continue Anyway.

Figure 22: Windows XP Logo Testing (Windows XP)



7. After the installation of the USB driver is completed the window below is displayed. Click Finish to close the hardware wizard.

Figure 23: USB Driver Installation Completion (Windows XP)

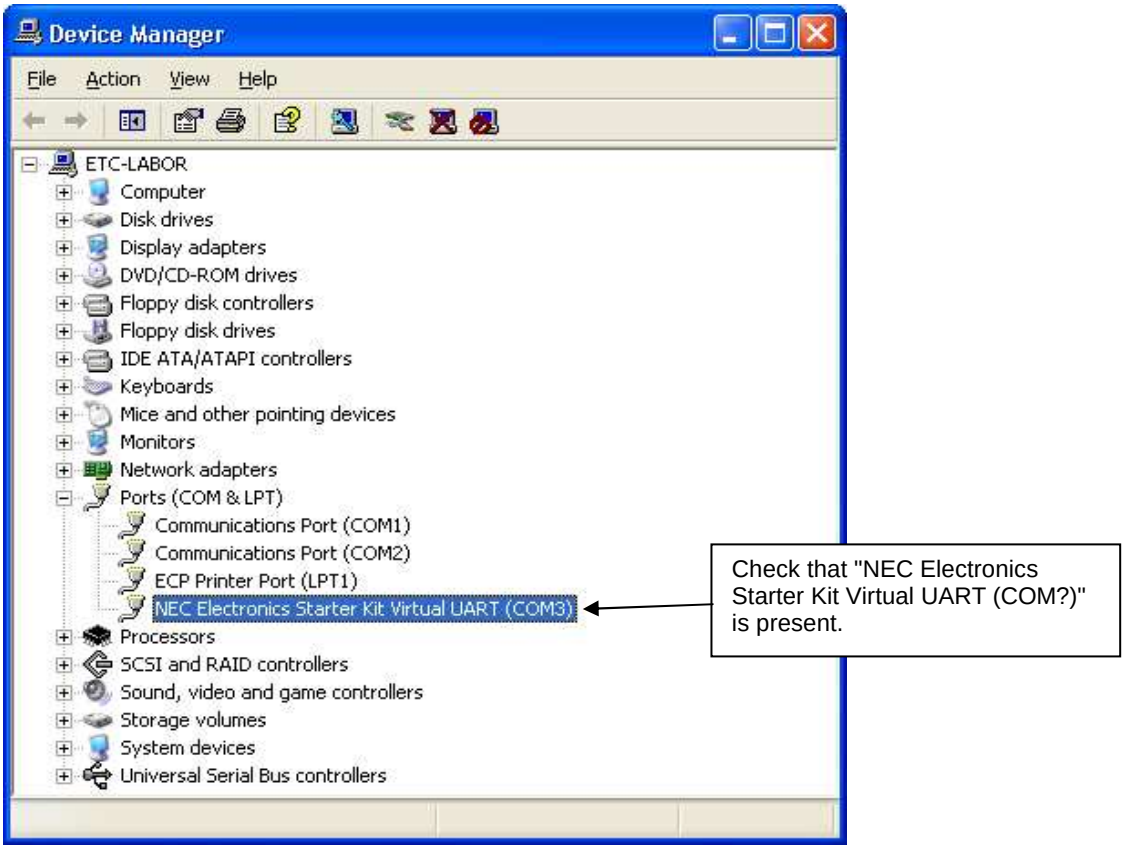


8.4 Confirmation of USB Driver Installation

After installing the USB driver, check that the driver has been installed normally, according to the procedure below. When using the *78K0R - Say it!* board in combination with IAR C-SPY debugger the “NEC Electronics Starter Kit Virtual UART” should be present like in the figure below.

By choosing the "Device Manager" within the Windows Properties (“Hardware” tab), check that the driver is installed normally.

Figure 24: Device Manager



9. IAR sample session

When everything is set up correctly the IAR Embedded Workbench can be started. To do so, start the Embedded Workbench from Windows “Start” menu > “Programs” > folder “IAR Systems” > “IAR Embedded Workbench Kickstart for 78K”. The following screen appears:

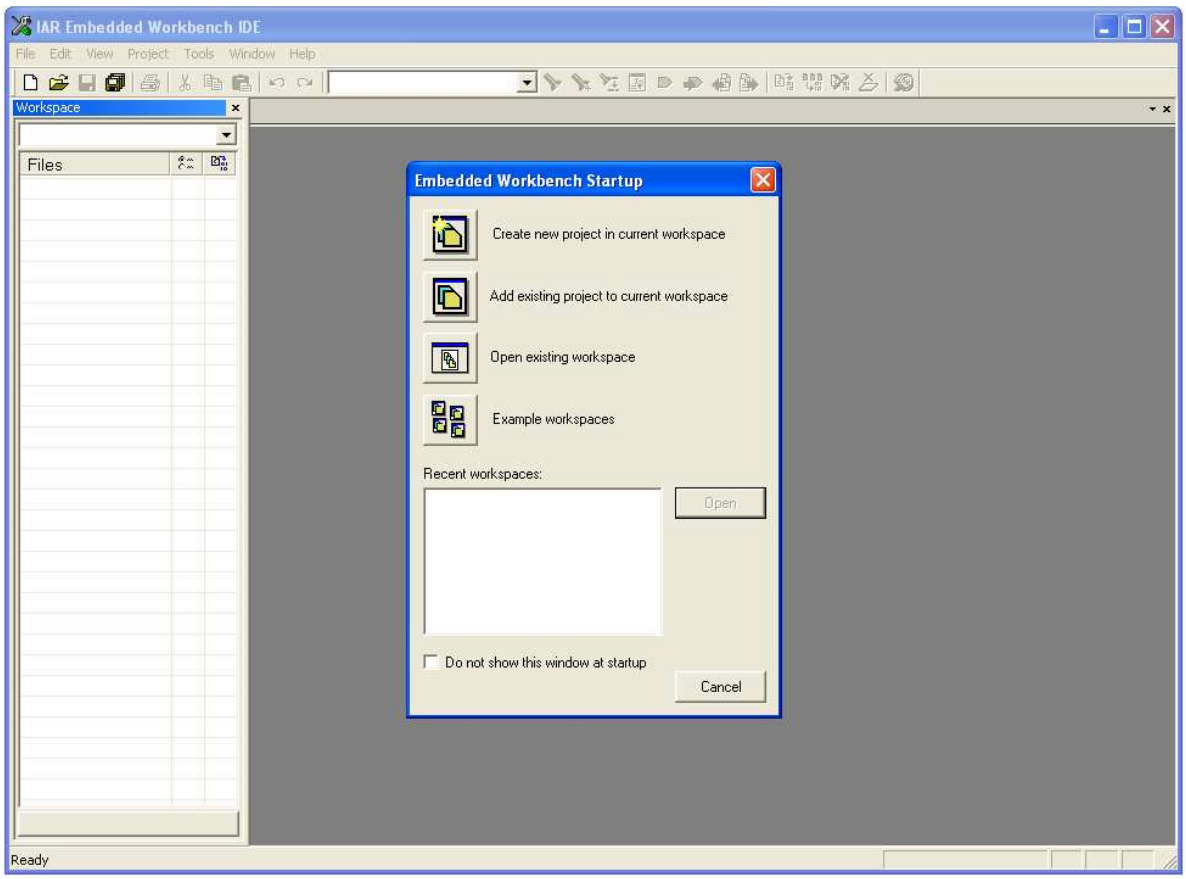


Figure 25: IAR Embedded Workbench

Now select the option “Open exiting workspace” from the “File” menu and locate the sample project “78K0R_Sayit_VoiceDemo_Obj”. Open the file “78K0R_Sayit_VoiceDemo.eww”. This is the workspace file that contains general information about the demo projects and corresponding settings.

After the demo workspace has been opened the files contained in the workspace are displayed. Now click on the little “+” sign next to the project filename “78K0R_Sayit_VoiceDemo - Debug” to show all files that were part of the selected demonstration project. The screen should now look similar to this:

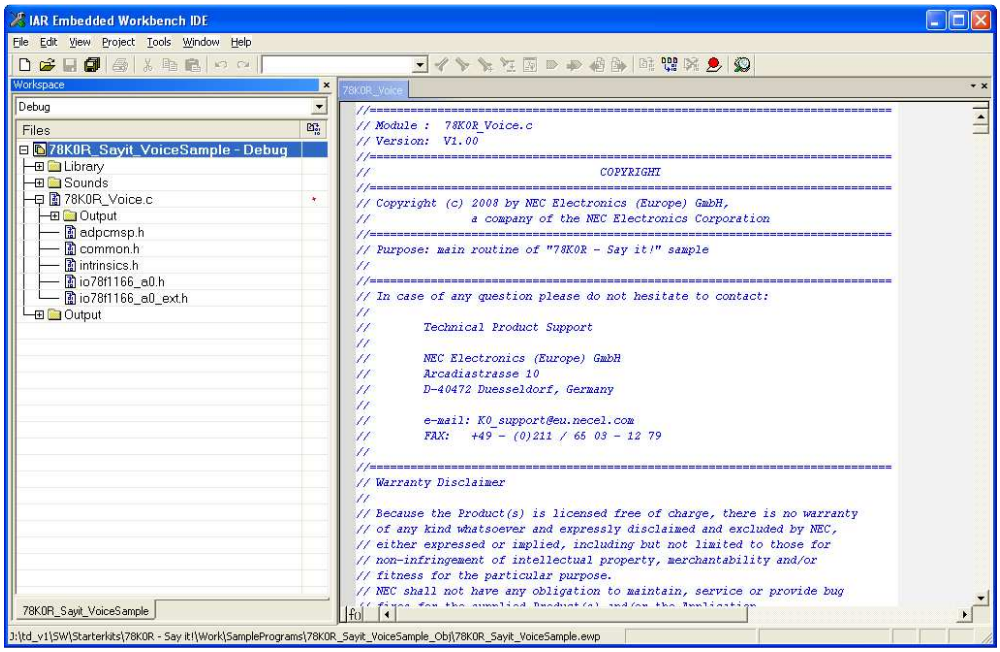


Figure 26: IAR project workspace

As a next step check some settings of the IAR Embedded Workbench that have to be made for correct operation and usage of the On-Board debug function of the *78K0R – Say it!* board. First highlight the upper project folder called “78K0R_Sayit_VoiceDemo – Debug” in the workspace window. Then select “Project” > “Options” from the pull-down menus. Next select the category “Debugger”. Make sure that the driver is set to “TK-78” in order to use the On-Board debug function of the *78K0R – Say it!* board. The device description file must be set to “io78f1166_a0.ddf”. The corresponding COM port where the *78K0R – Say it!* board is connected to the host PC will be detected automatically by the IAR C-SPY debugger.

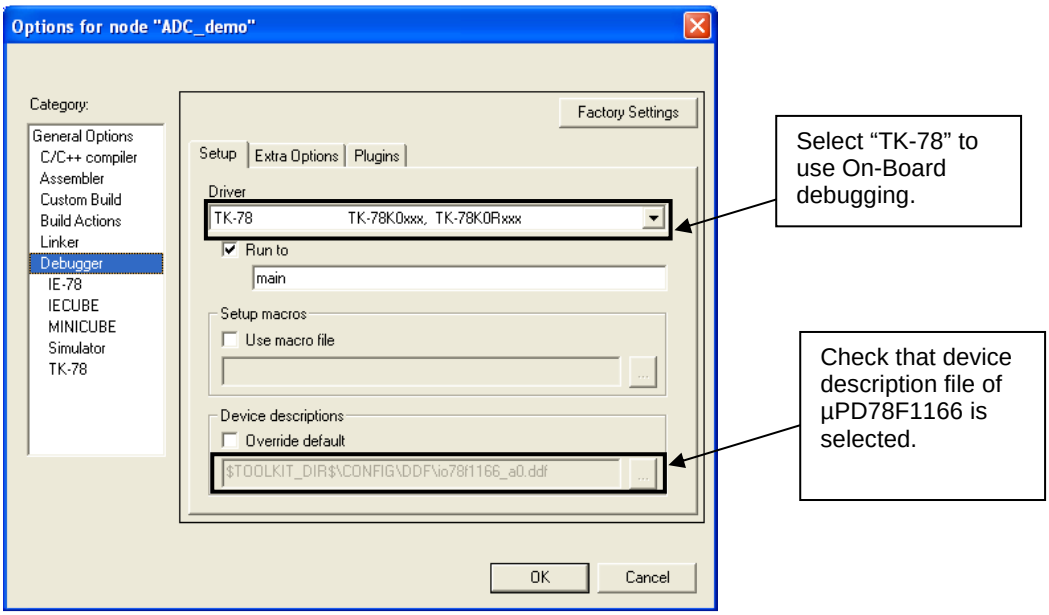


Figure 27: IAR debugger options

Next the correct linker settings of the demo project will be checked. This can be done in the “Linker” category as shown below. Select the “Config” tab and check that the linker command file “lnk78f1166_adpcmsp.xcl” is selected. This file is used by the linker and contains information on where to place the different sections of code, data and constants that may be used within the demo project:

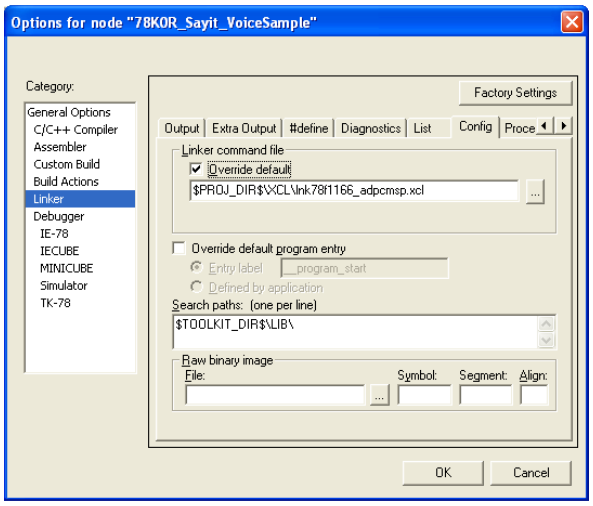


Figure 28: IAR Linker options

Now after everything has been setup correctly it’s time to compile and link the demonstration project. Close the Options menu and select “Rebuild All” from the “Project” menu. If the project is compiled and linked without errors or warnings it can now be downloaded to the *78K0R – Say it!* board and debugged. To start the IAR C-SPY debugger select the option “Debug” from the “Project” menu or press the ( ) “Debugger” button. In the next step the TK-78 Emulator has to be configured before downloading a new application. Press the OK button to enter the emulator hardware setup. Set the configuration as show in the figure below and start the download by pressing the OK button.

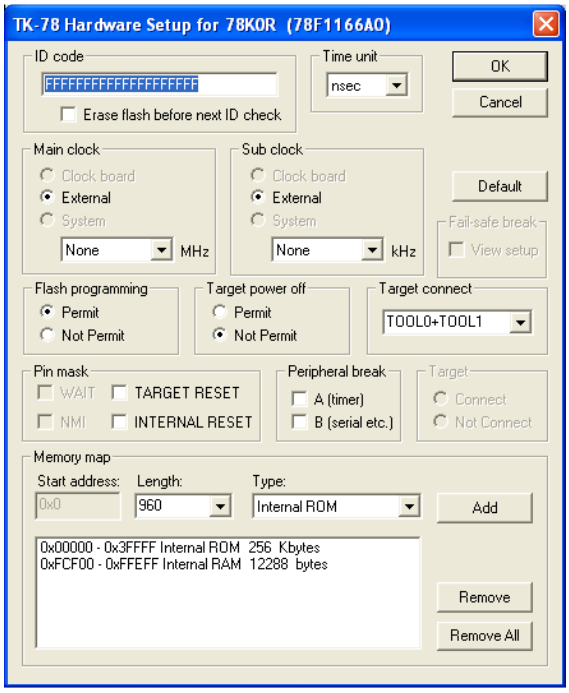


Figure 29: TK-78 hardware setup menu

Now the debugger is started and the demo project is downloaded to the *78K0R – Say it!* board. The progress of downloading is indicated by blue dots in the TK-78 Emulator window. Please note, downloading of larger executables can take some time.

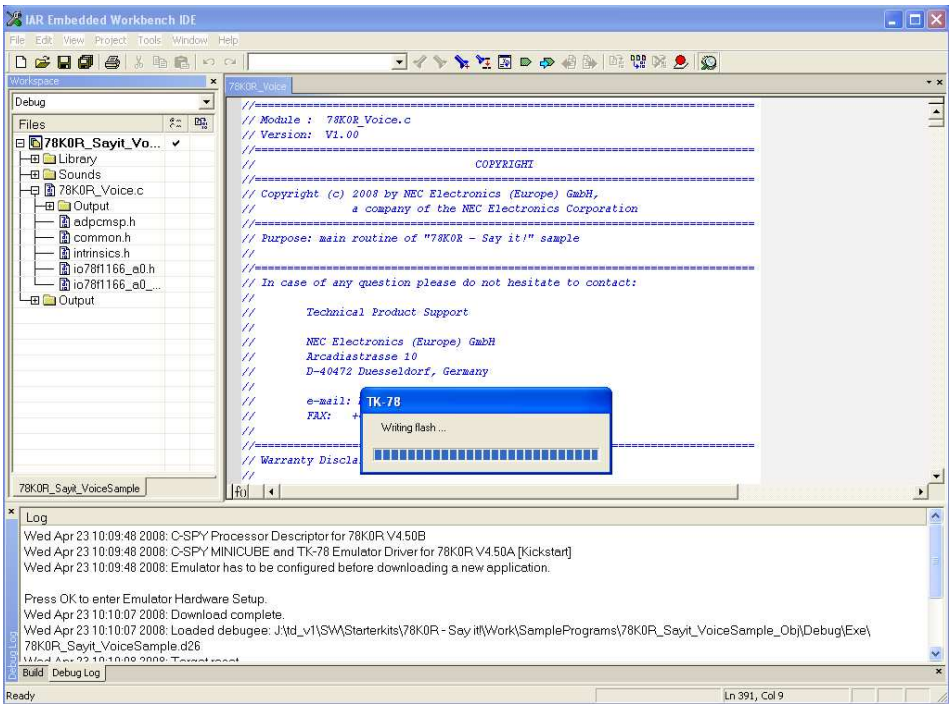


Figure 30: IAR project download

After the download was completed all debug features of IAR C-SPY debugger are available, i.e. Single Stepping, Step Over/-In/-Out, Go-Execution, Breakpoints, Register / Memory view etc.

To get more details on the debugger configuration and capabilities please refer to the “78K IAR Embedded Workbench IDE User Guide” of the IAR installation.

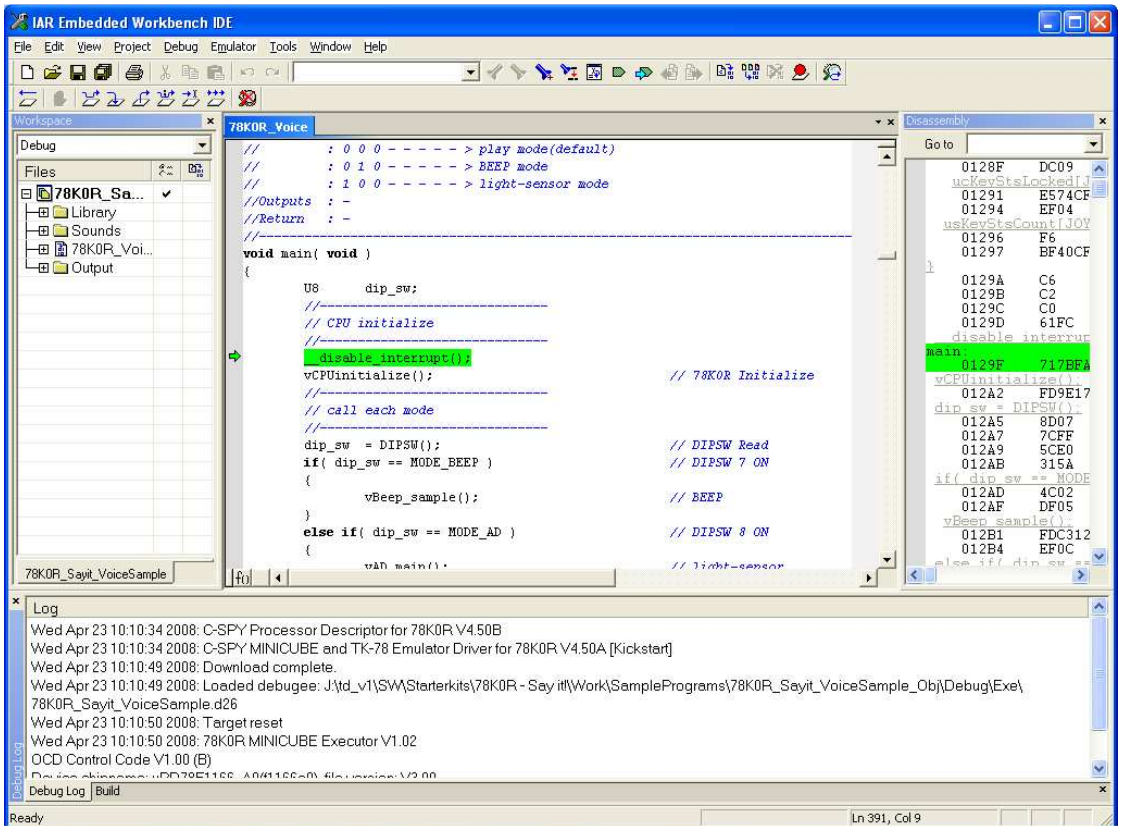


Figure 31: IAR C-SPY debugger

10. Troubleshooting

In some cases it might happen that the connection to the 78K0R – Say it! can not be established. This can be caused by the following two situations:

- ***Wrong security ID***: The security ID is required to prevent the FLASH memory of the 78K0R/KG3 microcontroller from being read by an unauthorized person. The security ID is located in the internal flash memory at addresses 0xC4-0xCD of the 78K0R/KG3 microcontroller. The IAR C-SPY debugger starts only when the security ID that is set during debugger start-up and the security ID set at addresses 0xC4 to 0xCD do match.

- ***Disabled On-Chip debug***: The On-Chip debug function of the 78K0R/KG3 microcontroller can be controlled by a dedicated Option Byte located at address 0xC3 in the internal flash memory. By disabling the On-Chip debug operation no connection to device can be established neither using the TK-78 interface nor using the QB-MINI2 On-Chip debug emulator.

In the above mentioned cases it is necessary to erase the internal flash memory of the 78K0R/KG3 microcontroller to restore the security ID and to enabled the On-Chip debug function.

In case of a security ID mismatch the following message box is displayed by the IAR C-SPY debugger. Click the YES button to enter the Hardware Setup menu.

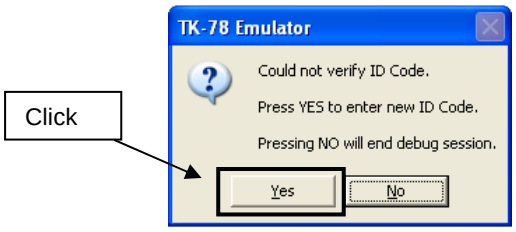


Figure 32: TK-78 enter Hardware Setup

Specify the default security ID <1> - the default security ID of an erased flash is equal to 10bytes 0xFF each - and enable the “erase flash before next ID check” option <2>. Then press the OK button <3> to start flash erasing and to establish the debugging session.

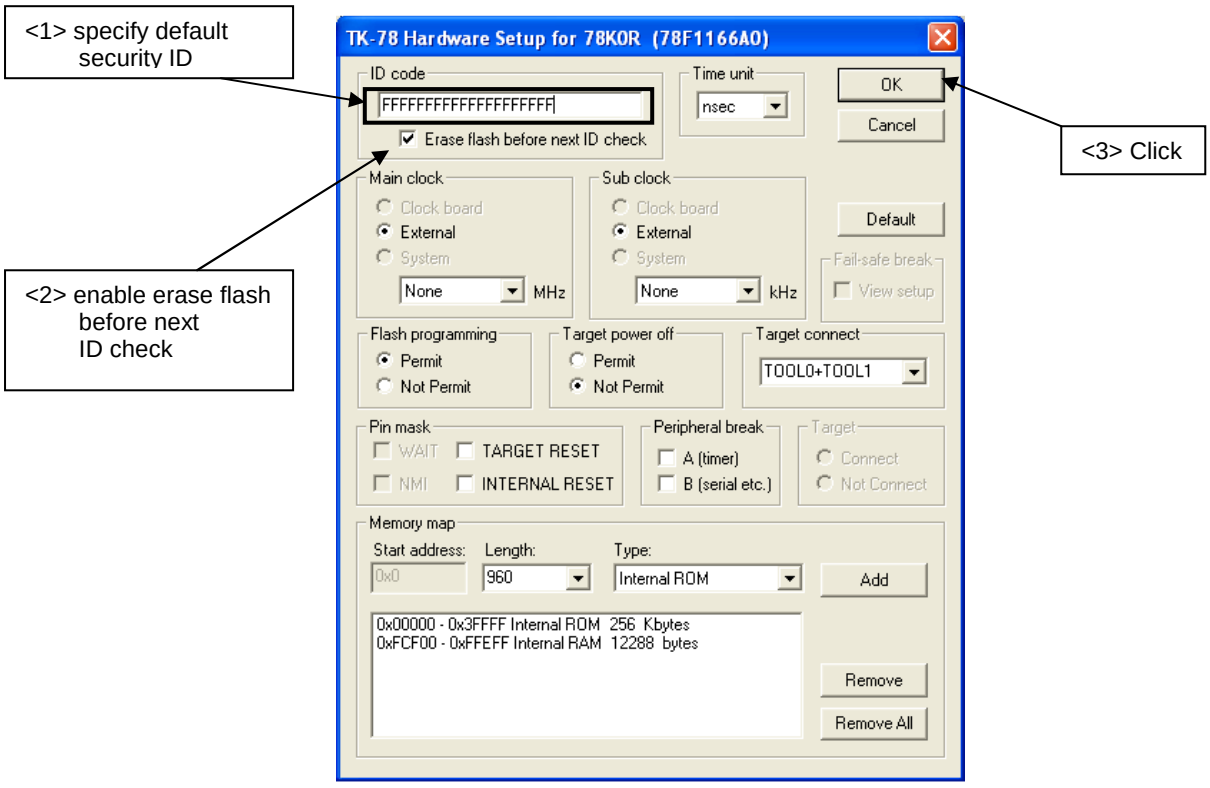


Figure 33: TK-78 Hardware Setup menu

The progress of flash erasing is indicated by blue dots in the TK-78 Emulator window. Following the debugger starts downloading the executable to the *78K0R – Say it!* board like shown in figure 28.

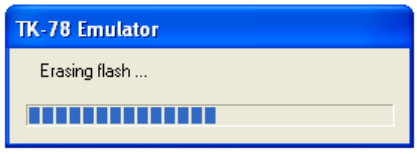


Figure 34: TK-78 flash erasing

11. Sample programs

11.1 General Introduction

Each of the sample programs is located in a single directory, which will be called main-directory of the sample. This main directory of each sample contains the complete project inclusive all output files of the development tool and the workspace file for instance "**78K0R_Sayit_VoiceDemo.eww**". All sample programs use the same directory structure:

78K0R_Sayit_DownloadDemo	Download sample project
78K0R_Sayit_VoiceDemo_Obj	Voice sample project
Debug	debug output files for IAR C-SPY debugger
inc	C header files
lib	ADPCM library
settings	configuration files, IAR Embedded Workbench
sound	Sound files
source	C source files
xcl	Linker control file
78K0R_Sayit_VoiceSample.eww	workspace file, IAR Embedded Workbench
78K0R_Sayit_VoiceSample.dep	dependency information file, IAR Embedded Workbench
78K0R_Sayit_VoiceSample.ewd	project setting file, IAR C-SPY debugger
78K0R_Sayit_VoiceSample.ewp	project file, IAR Embedded Workbench
78K0R_Sayit_VoiceDemo_Src	Voice sample project
CvADPCM	CvADPCM voice conversion and download tool
Sounds	Sample waves

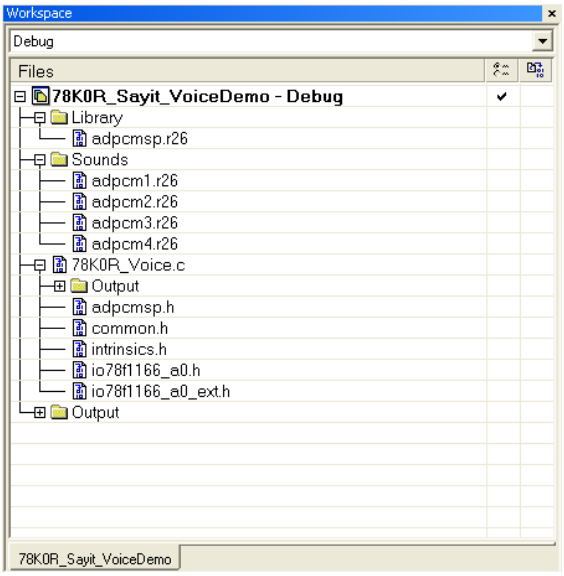
Table 15: Example directory structure

The main directory contains only the project and workspace files for the IAR Systems Embedded Workbench for 78K. All source files are located in the directories `/source` and `/sound`. The `/inc` directory contains the header files. The `/xcl` directory contains the linker control file of the 78K0R/KG3 device. All output files including the object files, list files, debug information and finally the executable file are stored in the directory `/Debug`.

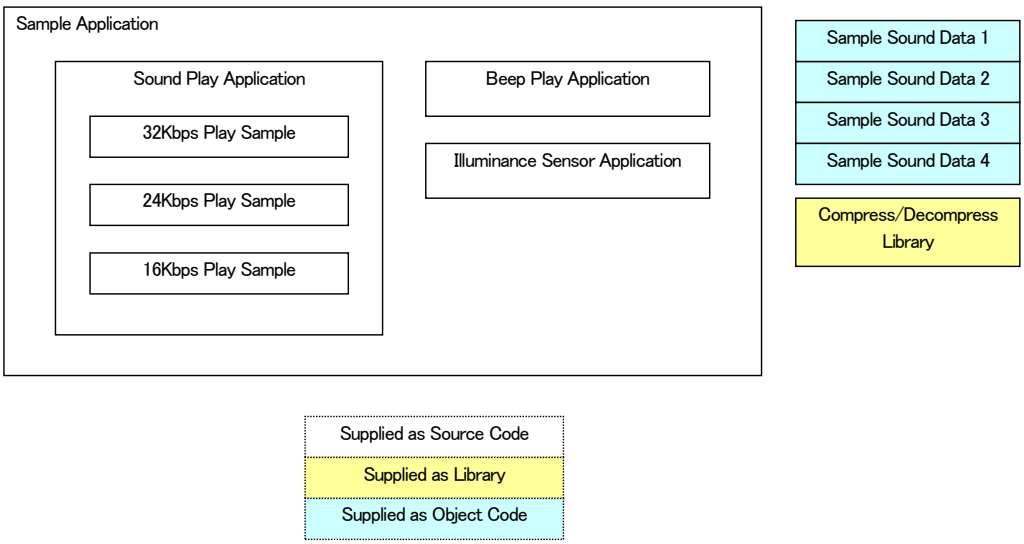
For details of using the IAR Embedded Workbench and the IAR C-SPY debugger please refer to the "78K IAR Embedded Workbench IDE User Guide".

11.2 “78K0R_Sayit_VoiceDemo_Obj” sample program

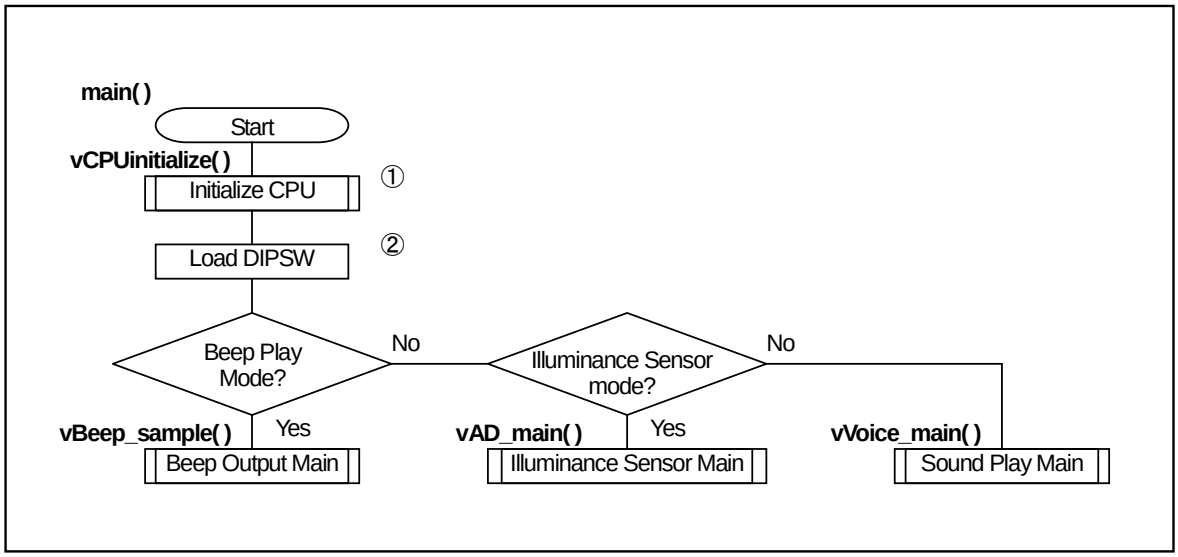
This sample program is a simple sound play application that uses the ADPCM compress/decompress library. The sample is divided into three major parts, the sound play, beep play and illuminance sensor application. Please use the source code of the different application parts as reference on how a sound play system based on the 78K0R device can be realized. To get more details on the source program itself, please refer to the following chapters. Because the *78K0R – Say it!* starterkit comes with the IAR Embedded Workbench Kickstart version – which is limited to generate maximum 16KByte of code size – the sound data within this sample is provided as object code, files “adpcm*.r26”. If you want to change the sound data please refer to the sample “78K0R_Sayit_VoiceDemo_Src”, chapter 11.3 of this document.



The structure of the sample program is shown below.



The flowchart of the “78K0R_Sayit_VoiceDemo” is given below.



① Initialize the microcontroller when power is on.

② Depending on the condition of switch SW5, it branches to different functions.

1. Mode to beep (high/low pitched sound) repeatedly.
2. Mode to output LED repeatedly with reading illuminance sensor data (A/D input).
3. Mode to play sound stored in built-in flash memory using PWM or D/A output.

11.2.1 How to run the sample program

The sample program supports the following three modes.

- Sound Play Mode

  Within this mode the *78K0R – Say it!* board plays one of four sounds. The selection of the corresponding sound data to be played can be done by the navigation switch SW1.

- Illuminance Sensor Mode

  By using this mode the light radiation is determined via the A/D converter by measuring the output voltage of the onboard illuminance sensor. The light radiation is displayed by the VOLUME LEDs. Additionally, pending on the light radiation the speech data "It got dark" or "It got light" is played by driving the onboard loudspeaker.

- Beep Play Mode

  This simple demo, outputs alternately a high pitched, silent and low pitched beep sound by driving the loudspeaker.

11.2.2 Sound Play Function

This sample application plays a selected sound from a set of four different sound data. To run this function please set switch SW5 / bits6-8 to “OFF”. Depending on, if you want to run the sample within stand-alone mode or if you want to use On-Board debugging please set the bits1-5 of configuration switch SW5 accordingly.

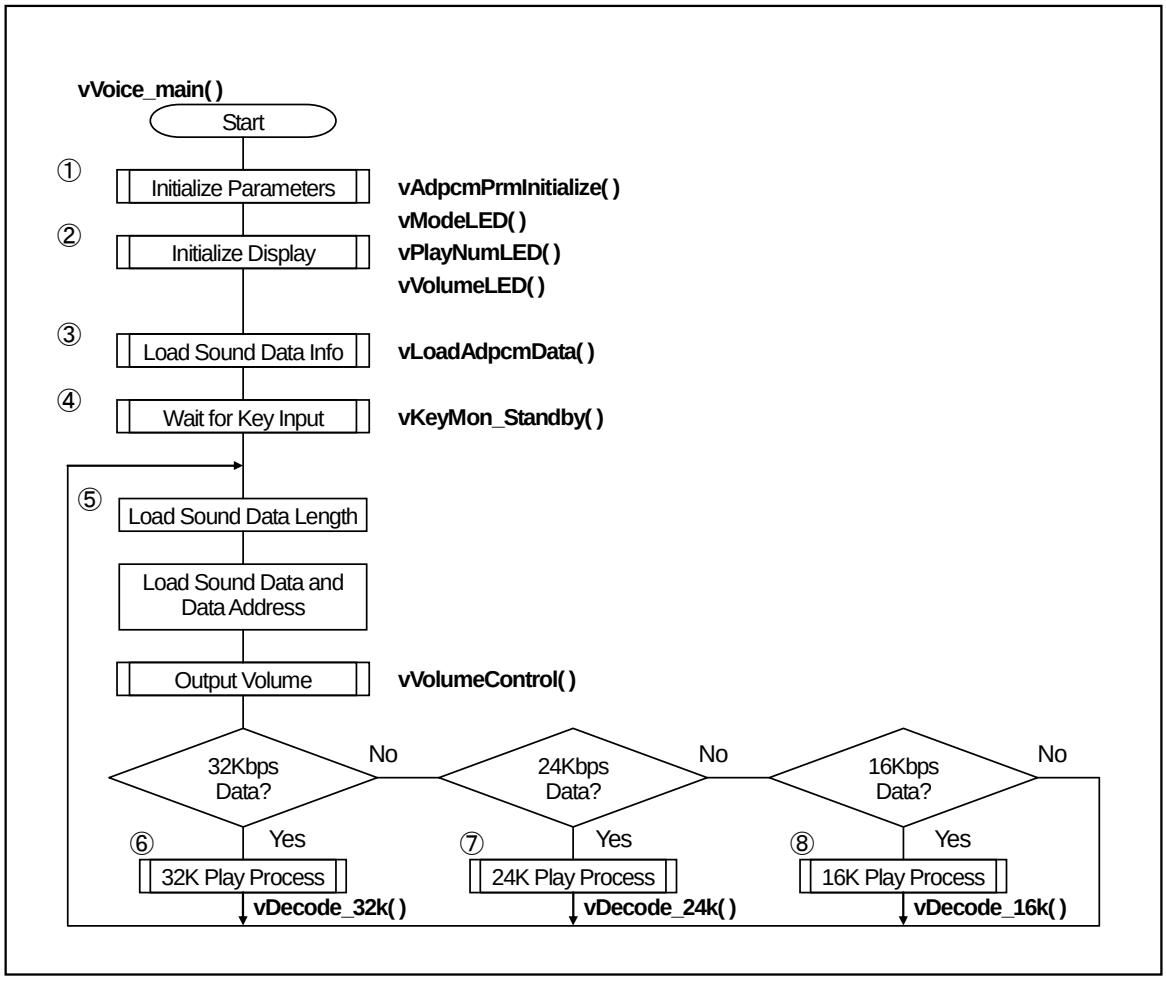
SW5 Configuration							
1	2	3	4	5	6	7	8
OFF	①	①	②	②	OFF	OFF	OFF

→ When using TK-78K0R On-Board debugging: ①=ON, ②=OFF
→ When using stand-alone (normal) mode: ①=OFF, ②=ON

The sound play sample can be controlled by the navigation switch SW1. The functionality of the navigation switch is a following:

SW1	Function
Left	This selects the wave output via PWM or D/A. The PWM LED and D/A LED shows which output mode is selected. The mode cannot be changed while the sound data is played.
UP / Down	This changes the sound output volume. There are 9 volume levels: OFF, 1-8. The VOLUME LEDs are showing the selected level. The sound volume can be changed during sound it played.
Center Push	The sound playing can be started, paused or stopped by pressing the navigation switch. Pressing the switch when a sound is playing pauses the sound output. Within the pause mode the PLAY LED starts blinking. By pressing the SW1 for more than 1.5 seconds the sound output is stopped. Playing of the sound can be started by pressing SW1 one’s again.
Right	This selects the sound data to play from totally four sound data stored in built-in FLASH memory of the 78K0R/KG3 device. The SOUND LEDs 1-4 shows which sound is actually selected. The sound data cannot be changed while the sound data is played.

Below the flow chart of the main sound play process is shown.



① Initialize parameters that used by application.

② Initialize displays.

③ Load sound data information from memory:
1. compression bit info of sound data
2. data size
3. data start address

④ Wait until the navigation switch input (up/down/left/right/push).

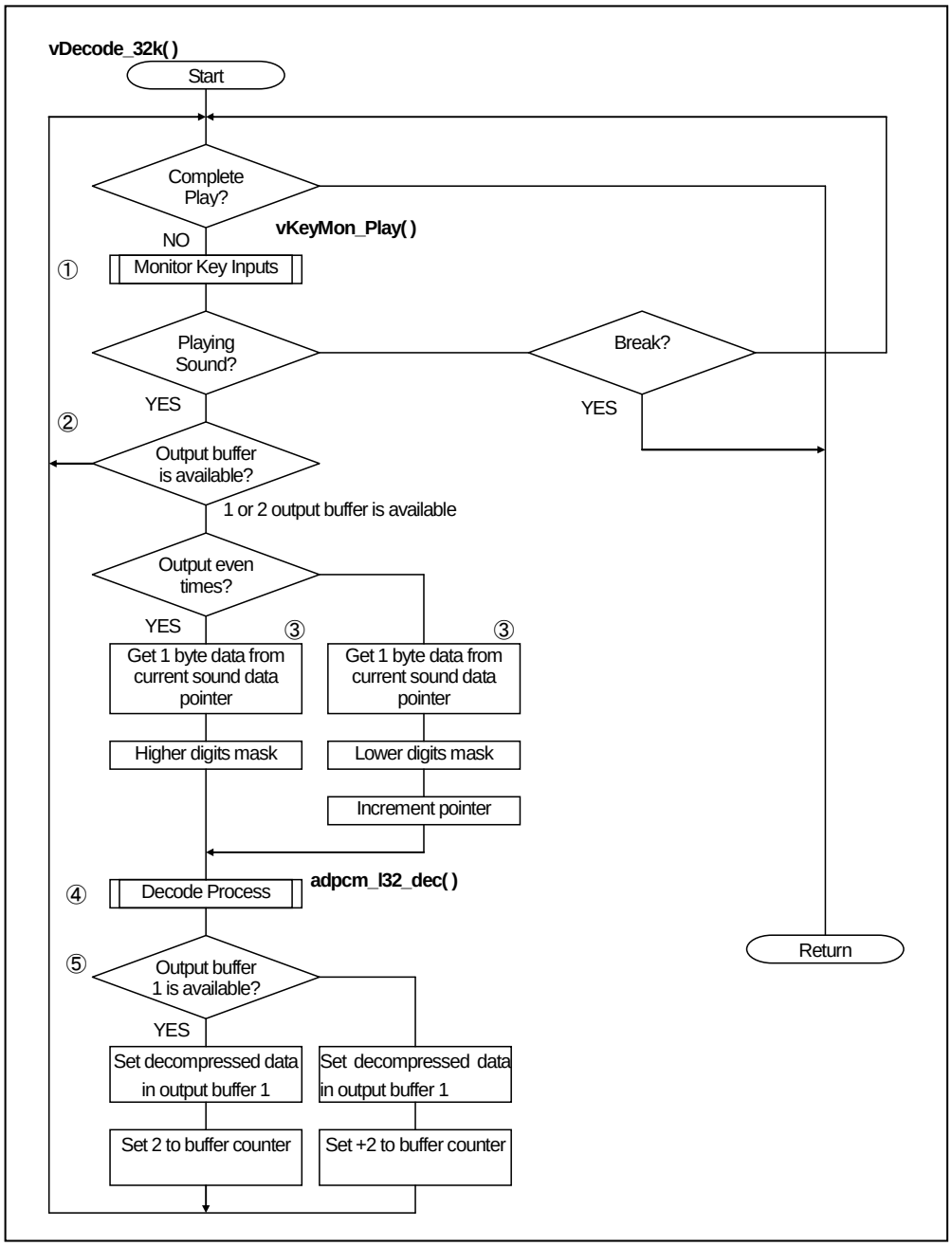
⑤ Load sound data information of selected sound.

⑥ Decompression process when compression bit is 4 bit.

⑦ Decompression process when compression bit is 3 bit.

⑧ Decompression process when compression bit is 2 bit.

As reference, the flow of the 32kbps decompression is shown in the chart below.



① Monitor key inputs while playing sound.

② It does not start the decompression process if no output buffer area is available.

③ Extracts the higher or lower 4 bit of 1 sound data (8 bit) depending on number of outputs (even or odd).

④ Decode the 4 bit data.

⑤ Store decompressed data in one of output buffer area depending on the availability.

11.2.3 Illuminance Sensor Function

By using this mode the light radiation is determined via the A/D converter by measuring the output voltage of the onboard illuminance sensor. The light radiation is displayed by the VOLUME LEDs. Additionally, pending on the light radiation the speech data "It got dark" or "It got light" is played by driving the onboard loudspeaker. The corresponding speech data is stored in the “adpcm1.r26” and “adpcm2.r26” object files. If you change the sound data by re-flashing of the corresponding memory area different sound data can be played.

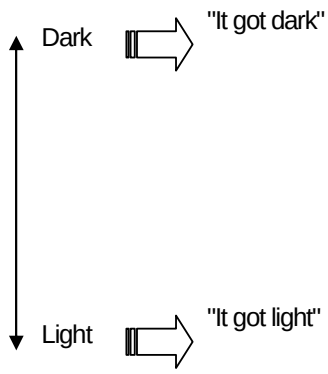
To run this function please set switch SW5 to the following configuration.

SW5 Configuration							
1	2	3	4	5	6	7	8
OFF	①	①	②	②	OFF	OFF	ON

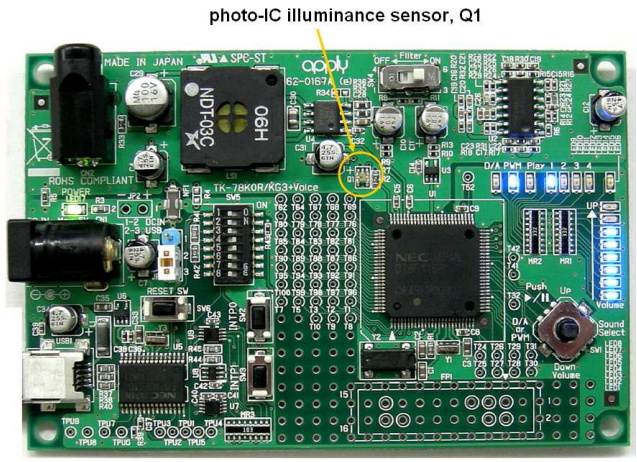
→ When using TK-78K0R On-Board debugging: ①=ON, ②=OFF
→ When using stand-alone (normal) mode: ①=OFF, ②=ON

A configuration via the navigation switch is not needed and supported within this sample. Depending on the measured light radiation, the number of driven VOLUME LEDs is changed. The relationship between the A/D conversion value and the driven VOLUME LEDs is shown in the table below.

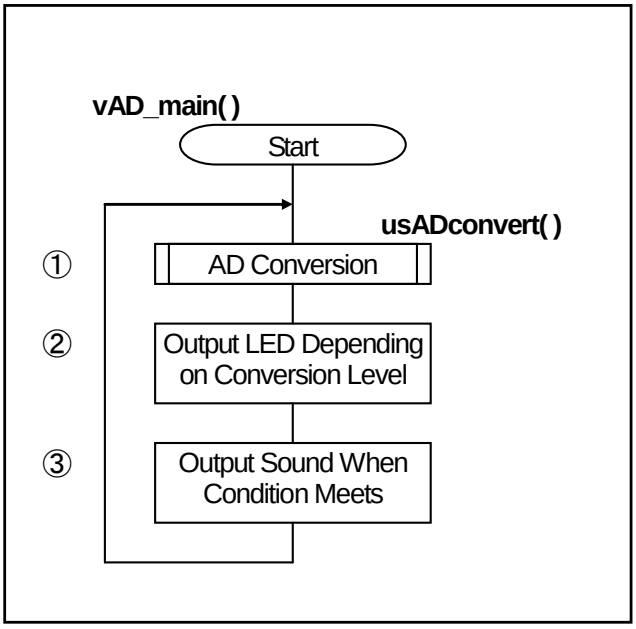
A/D Conversion Value	LED Output
Conversion Value < 0021 H	No Light
Conversion Value < 0042 H	1 Light
Conversion Value < 0063 H	2 Lights
Conversion Value < 0084 H	3 Lights
Conversion Value < 00A5 H	4 Lights
Conversion Value < 00C6 H	5 Lights
Conversion Value < 00E7 H	6 Lights
Conversion Value < 0108 H	7 Lights
Conversion Value < 0200 H	8 Lights



To simple demonstrate the functionality of the sample spread you hand over the illuminance sensor to change the light radiation. After the A/D threshold value stays for 1.5 seconds the corresponding speech data is played. The speech data is played again not until the illuminance value gets back to the center value area.



The flow of the Illuminance sensor application example is given below.



① Read illuminance sensor data using the A/D converter of the 78K0R/KG3 microcontroller.

② Output LED volume (9 levels) depending on A/D in value (bit information).

③ Play sounds depending on the condition.

Relationships between the A/D conversion value and LED output are shown below.

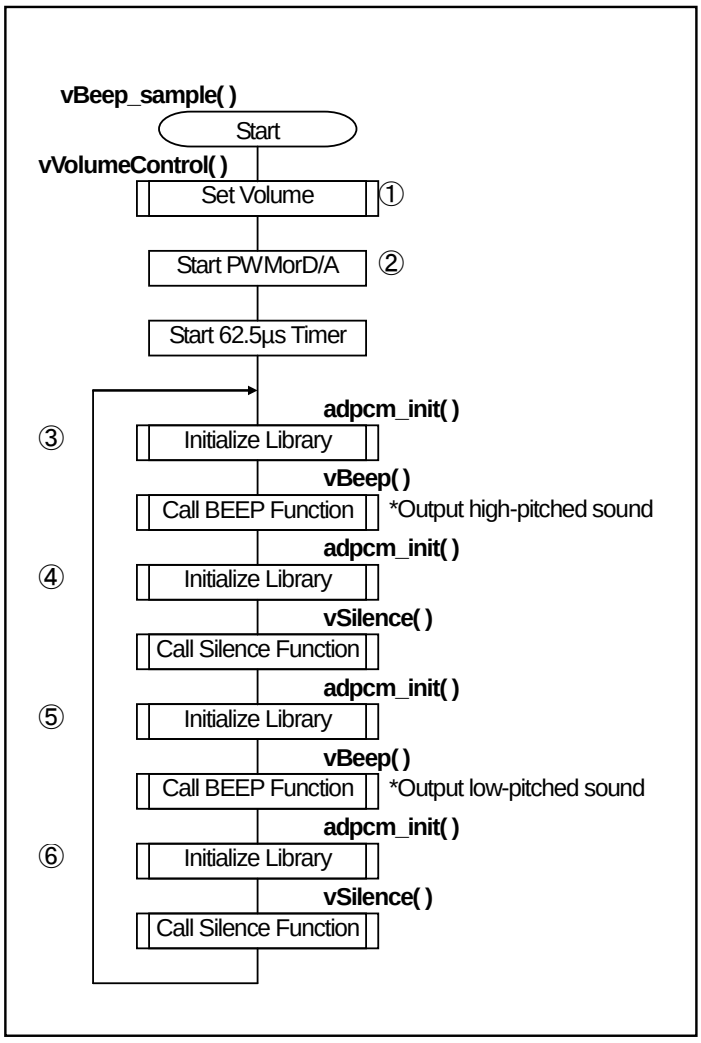
11.2.4 Beep Play Function

This simple demo, outputs alternately a high pitched, silent and low pitched beep sound by driving the loudspeaker. To run this function please set switch SW5 to the following configuration.

SW5 Configuration							
1	2	3	4	5	6	7	8
OFF	①	①	②	②	OFF	ON	OFF

→ When using TK-78K0R On-Board debugging: ①=ON, ②=OFF
→ When using stand-alone (normal) mode: ①=OFF, ②=ON

The flowchart of the beep play sample is shown below.



① Fix the volume level.

② Use PWM as output

③ Initialize compression/decompression library, then call BEEP output function.
(set "high-pitched sound" and "2sec" as the input parameters)

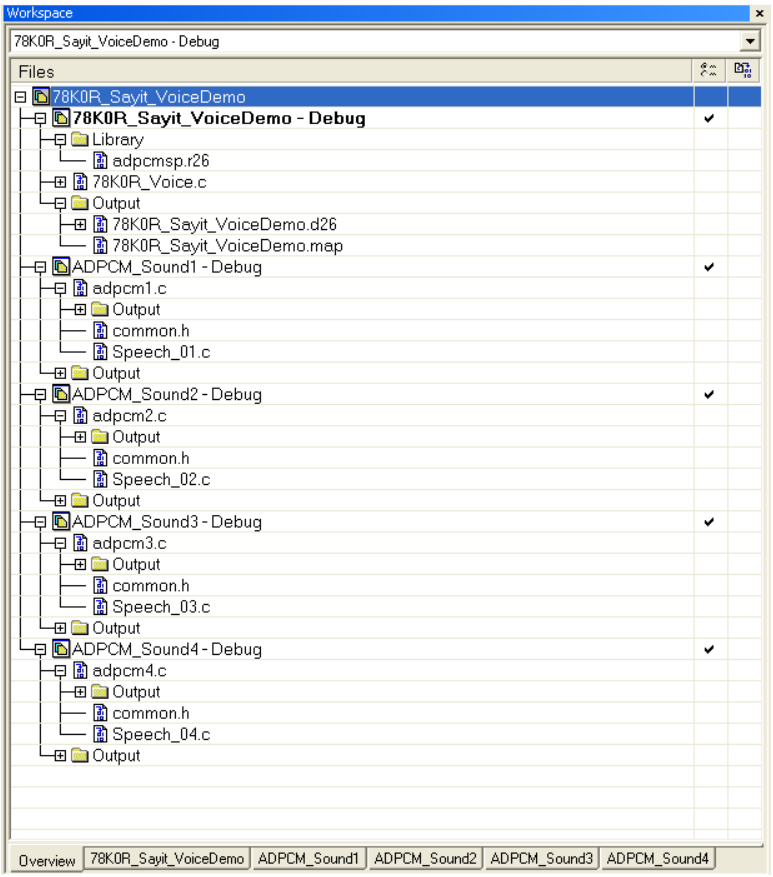
④ Initialize compression/decompression library, then call Silence output function.
(set "500msec" as the input parameter)

⑤ Initialize compression/decompression library, then call BEEP output function.
(set "low-pitched sound" and "2sec" as the input parameters)

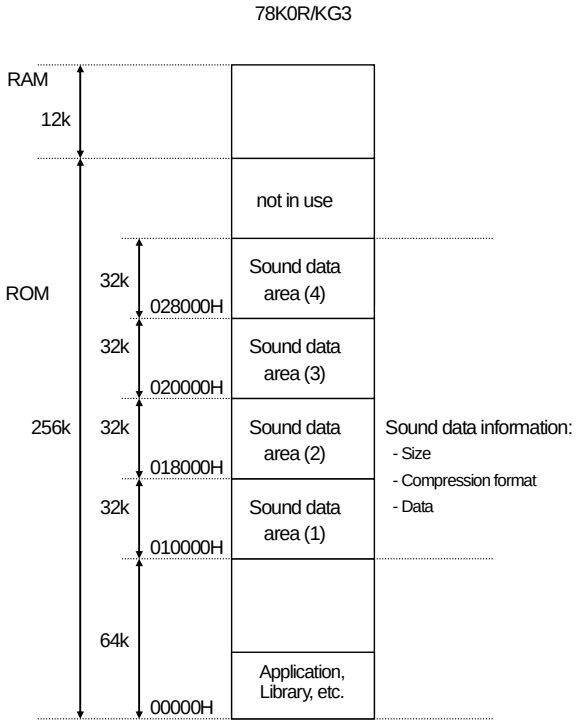
⑥ Initialize compression/decompression library, then call Silence output function.
(set "500msec" as the input parameter)

11.3 “78K0R_Sayit_VoiceDemo_Src” sample program

This program has the same functionality like the “78K0R_Sayit_VoiceDemo_Obj” sample described in chapter 11.2. By using this sample you are able to change the sound data and replace it by your own preferred sound data. For this purpose the project is split into the main application - project “78K0R_Sayit_VoiceDemo” - and four sound projects “ADPCM_Sound1” to “ADPCM_Sound4”, containing the sound data. The corresponding workspace and the partitioning of the projects can be seen in the figure below.



The main project doesn’t contain any sound data. The reference to the sound data itself is done by pointers to the corresponding sound data areas. For each of the four sound data a fixed memory area is defined. The corresponding mapping can be seen in the figure below.

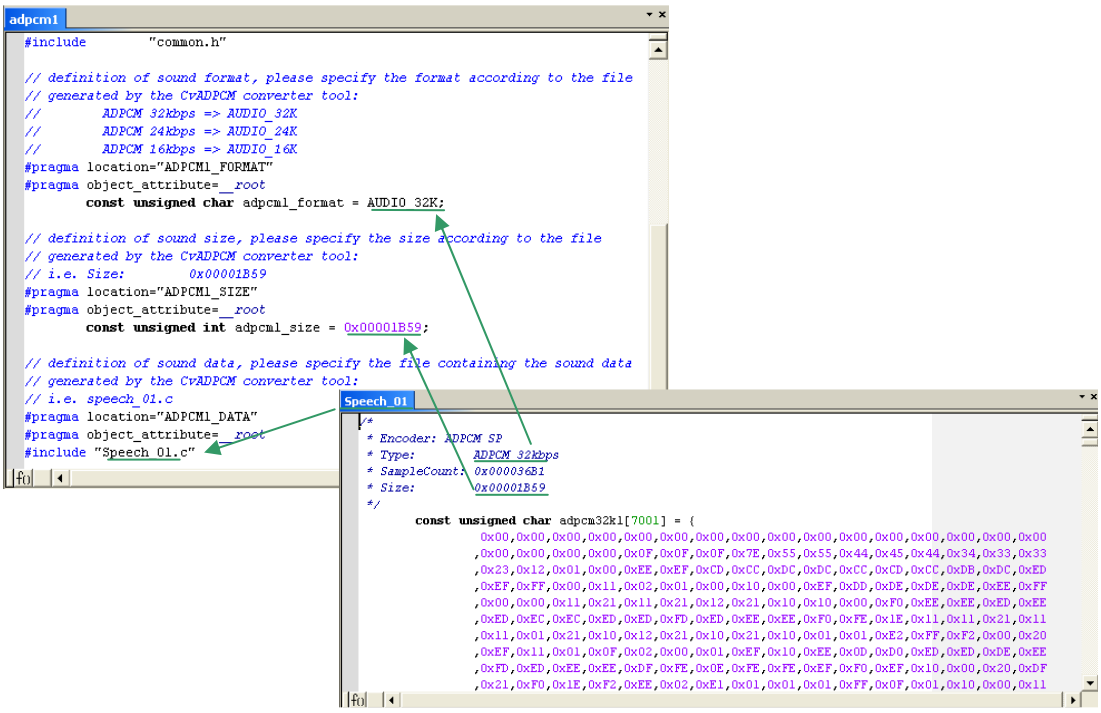


The assignment of the memory for the different sound data is done within the linker directive file “lnk78f1166_adpcmsp.xcl”, which is mutual for the main and the four sound data projects. Each sound data area contains about a ADPCM_FORMAT, ADPCM_SIZE and ADPCM_DATA section. The definition and mapping of the corresponding sections within the linker directive file can be seen in following figure.

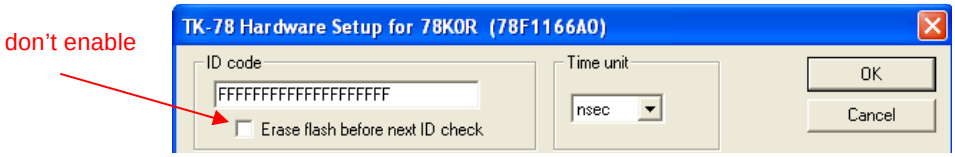
lnk78f1166_adpcmsp.xcl

```
//------------------------------------------------------------------------
//      Const segments of ADPCM sound data
//------------------------------------------------------------------------
-Z(CONST)ADPCM1_FORMAT=10000
-Z(CONST)ADPCM1_SIZE=10002
-Z(CONST)ADPCM1_DATA=10004

-Z(CONST)ADPCM2_FORMAT=18000
-Z(CONST)ADPCM2_SIZE=18002
-Z(CONST)ADPCM2_DATA=18004

-Z(CONST)ADPCM3_FORMAT=20000
-Z(CONST)ADPCM3_SIZE=20002
-Z(CONST)ADPCM3_DATA=20004

-Z(CONST)ADPCM4_FORMAT=28000
-Z(CONST)ADPCM4_SIZE=28002
-Z(CONST)ADPCM4_DATA=28004
```

The projects “ADPCM_Sound1” to “ADPCM_Sound4” do include only the corresponding sound data for the four sound data areas. The definition of the sound data is done within the “adpcm.c” file. This file contains the definition of the sound format, the sound size and the raw sound data. The sound data itself is generated by the CvADPCM converter tool. The figure below shows for instance the definition for the sound data area1.



The definition of the sound format and sound size is done by the constants “adpcm1_format” and “adpcm1_size”. They have to be set according to the sound data file, in this example the file “Speech_01.c”, generated by the CvADPCM converter tool. The sound data file itself is included and placed to the “ADPCM1_DATA” section. Analog to this, the format and size information is placed to the “ADPCM1_FORMAT” and “ADPCM1_SIZE” sections. The three sections are referenced and accessed within the main sample project via pointers. The creation of the sound data is done by simply re-building the corresponding project. By doing so an Intel-Hex file is generated that is mapped to the corresponding sound data area. To download new sound data to the *78K0R – Say it!* board and furthermore to re-flash the reserved sound data area within the 78K0R/KG3 device please press the ( ) “Debugger” button within the IAR Embedded Workbench. Note, be sure that the function “Erase flash before next ID check” is not active within the “TK-78 Hardware Setup” menu. Otherwise the complete FLASH memory of the 78K0R/KG3 device will be erased before programming the sound data.



After the IAR C-SPY debugger has finished re-programming of the chosen sound data area, please close the debugger. The new sound data is now successfully updated. Please proceed in the same way for the remaining sound areas. Because the *78K0R – Say it!* starterkit comes with the IAR Embedded Workbench Kickstart version, the maximum size for each sound data is limited to 16KByte. After you have programmed or changed the sound data, please activate the “78K0R_Sayit_VoiceDemo” project and debug it as usual. Please note, also by debugging the sample, be sure that the function “Erase flash before next ID check” is not active within the “TK-78 Hardware Setup” menu. Otherwise the complete FLASH memory of the 78K0R/KG3 device will be erased before programming the sample project and the sound data gets lost.

11.4 "78K0R_Sayit_DownloadDemo" sample program

This sample application allows you to download and modify the four different sound data areas by using the CvADPCM tool. Additionally it offers the same sound play function like the "78K0R_Sayit_VoiceDemo". The CvADPCM tool uses the USB interface and the UART3 of the 78K0R/KG3 device for communication. Consequently the TK-78K0R On-Board debugging function can not be used by running the "78K0R_Sayit_DownloadDemo" program. The *78K0R – Say it!* starterkit comes with this sample program as default configuration. The "78K0R_Sayit_DownloadDemo" is provided as HEX file only.
Please note, by using the download sample the sound data size for each sound data area is limited to 32kByte.

To run this sample program please set switch SW5 to the following configuration.

SW5 Configuration							
1	2	3	4	5	6	7	8
OFF	OFF	OFF	ON	ON	ON	ON	ON

The download sample can be controlled by the navigation switch SW1. The functionality of the navigation switch is a following:

SW1	Function
Left	No function.
UP / Down	No function.
Center Push	This starts the download process. The *78K0R – Say it!* waits for data to be transferred from the host PC controller by the CvADPCM tool.
Right	This selects the sound data area that should be modified. The SOUND LED shows which of the four sound data areas is selected.

The configuration and operation of the download sample is displayed by the LED's.

LED	Function
D/A	Starts blinking when downloading.
PWM	Starts blinking when downloading.
PLAY	It starts blinking when the board is ready for downloading.
SOUND	It shows which sound data storage area (1-4) is selected by driving the corresponding LED.
VOLUME	Before download : no light Starting download : all are lighted, then reduce the number of lights (FLASH deletion time) While downloading : no light, then increase the number of lights (FLASH writing progress) Complete download : no light Error occurred : all are lighted

To program the *78K0R – Say it!* starterkit with the download sample please open the "78K0R_Sayit_DownloadDemo" project within the IAR Embedded Workbench. Please press the () "Debugger" button to start FLASH programming. After the IAR C-SPY debugger has finished re-programming, please close the debugger. The download sample is now successfully programmed. To use the IAR Embedded workbench for reprogramming please set configuration switch SW5 accordingly.

11.4.1 Procedure to change sound data by downloading via CvADPCM tool

In this section, how to change the sound data by downloading is explained.

1. Connection of devices

Connect the *78K0R – Say it!* starterkit and the host PC with USB cable.
PLAY LED start blinking.

2. Preparation of the application

Select the data area by shifting the navigation switch to the RIGHT direction, then press the navigation switch. The selected sound data area is indicated by the SOUND LED.
After the navigation switch has been pressed, all VOLUME LED start lighting, then the number of driven LED reduces depending of the FLASH erase progress.

3. Download sound data

Start downloading by selecting the sound data with the CvADPCM tool.

Caution: Following compression format can be used.

"ADPCM SP -4bit/Sample"
"ADPCM SP -3bit/Sample"
"ADPCM SP -2bit/Sample"

For operation of CvADPCM tool, refer to "Audio Data Conversion Tool (CvADPCM) User's Manual".

You can find sample WAVE data in SampleProgram folder, directory “\SamplePrograms\Sounds”.

4. Start download

During downloading, the PWM LED and the D/A LED start blinking alternately and the lights of VOLUME LED increases depending on the download progress.
When the download is completed, all VOLUME LED are lighted and PLAY LED starts blinking.
* If you wish to continue downloading, repeat from step 2.

5. Play downloaded sound data

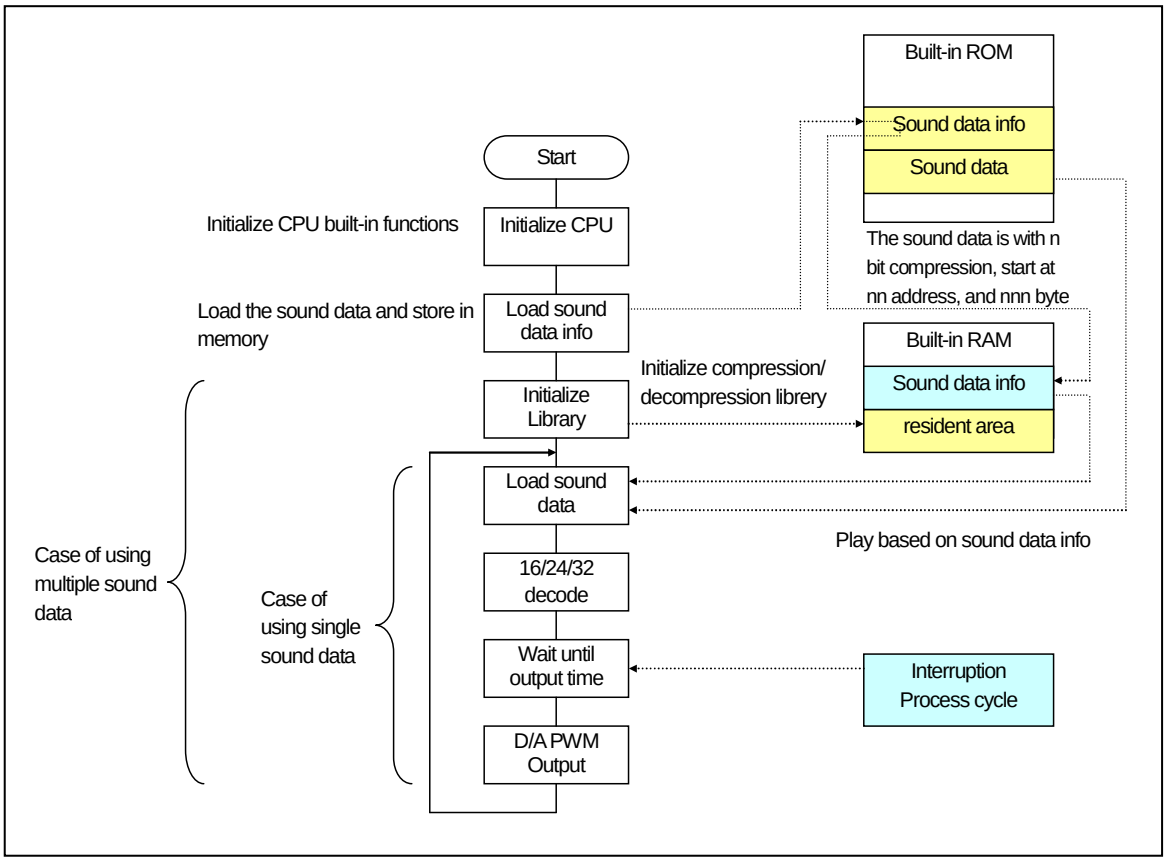
Set configuration switch SW5 / bits6, 7, 8 to **OFF** and reset the board by pressing the reset button SW6.

SW5 Configuration							
1	2	3	4	5	6	7	8
OFF	OFF	OFF	ON	ON	OFF	OFF	OFF

The sound play application starts. Select the sound data and play it.

11.5 “78K0R_Sayit_VoiceDemo” source code description

You can develop an application by using the sound data decompression library for 2, 3, 4 bit compression sound data. Following chart shows the process of the application.



With this example, it stores the sound data information (such as compression info, sound data address, data size, etc.) from the built-in ROM to RAM in advance. When it is time to play sounds, it calls the library initialize API to start the process. It loads and calls decompression API the sound data based on the preloaded sound data information. The output code from the decompression API is outputted several times depending of the data size by PWM or D/A.

11.5.1 Example of Creating Sound Play Application

The “78K0R_Sayit_VoiceDemo” sample application plays one sound from four sounds stored in built-in flash memory by using PWM or D/A output, based on inputs from user interface. The processes consist of 4 coding blocks.

1. Initialize
2. Wait for user input
3. Pre-process for sound data
4. Process to play sound

These blocks will be explained based on “78K0R_Sayit_VoiceDemo” application.

11.5.1.1 Initialize

Initialize internal parameters and set displays on *78K0R – Say it!* as default.

```
/*-------------------------------------------------------------------
* Function : play main
* Descr    : void vVoice_main( void )
* Inputs   : -
* Outputs  : -
* Return   : -
*-------------------------------------------------------------------*/
void vVoice_main( void )
{
   U8 play_mode = PWM_MODE;
   U8 play_num = PLAY_NUM_MIN;
   U8 play_volume = PLAY_VOLUME_4;

   U8 data_type;
   U8 ucKeySts;
   U32 sound_size;
   U8 __far *_adpcmDataAdr;
   /*******************************/
①  /* Initialize                  */
   /*******************************/
   vAdpcmPrmInitialize();
   /*******************************/
   /* LED initialize              */
   /*******************************/
②  vModeLED( play_mode );
   vPlayNumLED( play_num );
   vVolumeLED( play_volume );
   /*******************************/
   /* read data                   */
③  /*******************************/
   vLoadAdpcmData();
```

① Initialize parameters/flags that are used by application.
② Initialize displays (display initial screen).
③ Load sound data information (compression bit info, address, data length, etc.).

11.5.1.2 Wait for user input

Wait for user input by activating the navigation switch SW1.

```
①  /*******************************/
   /* Start                       */
   /*******************************/
   __enable_interrupt();
   while(1)
   {
      ucPlaySts = STATUS_STANDBY;
      /*******************************/
      /* Wait for Key                */
      /*******************************/
      ucKeySts = 0xFF;
②     while( ucKeySts )
      {
         ucKeySts = JOYSTK();
      }
      vPlayPrmInitialize();
      /*******************************/
      /* Wait for Joystick push      */
③     /*******************************/
      INT_1MSEC_ENABLE();
      vKeyMon_Standby( &play_mode , &play_num , &play_volume );
      PlayMode = play_mode;
```

① Start play application with global interrupts enabled

② Initialize internal parameters in loop process.

③ Wait until the navigation switch to be pushed to start playing.

11.5.1.3 Pre-process for sound data

As the first initialization the sound data information is loaded. The "voice_adpcm" array contains compression bit info, data address, data length, etc. In this block, the sound data information of user selection is loaded.

```
        /*******************************/
        /* prepare                     */
        /*******************************/
①       data_type   = voice_adpcm[play_num-1].datatype;
        if( data_type==AUDIO_32K )
        {
②          sound_size  = voice_adpcm[play_num-1].datacount * 2;
        }
        else if( data_type==AUDIO_24K )
        {
③         sound_size  = (voice_adpcm[play_num-1].datacount / 3 ) * 8;
        }
        else if( data_type==AUDIO_16K )
        {
④          sound_size  = voice_adpcm[play_num-1].datacount * 4;
        }
        else
        {
           sound_size  = 0;
        }
⑤       _adpcmDataAdr  = voice_adpcm[play_num-1].address;
```

① Get compression bit info.
② Calculate data size if compression bit is 4 bit.
③ Calculate data size if compression bit is 3 bit.
④ Calculate data size if compression bit is 2 bit.
⑤ Get the start address of compressed sound data.

11.5.1.4 Process to play sound

After all sound data information are loaded and initialized, the corresponding play function is called dependent on the compression.

```
            /*******************************/
            /* Play                        */
            /*******************************/
①           vVolumeControl( play_volume );
②           adpcm_init( Adpcm_Work );
            PLAY_LED( LED_ON );
            ucPlaySts = STATUS_PLAY;

            if( data_type==AUDIO_32K )
③           {
               vDecode_32k( play_mode , &play_volume , sound_size , _adpcmDataAdr );
            }
            if( data_type==AUDIO_24K )
④           {
               vDecode_24k( play_mode , &play_volume , sound_size , _adpcmDataAdr );
            }
            if( data_type==AUDIO_16K )
⑤           {
               vDecode_16k( play_mode , &play_volume , sound_size , _adpcmDataAdr );
            }
            /*******************************/
            /* end of play                 */
            /*******************************/
            INT_1MSEC_DISABLE();
            PLAY_LED( LED_OFF );
         }
      }
```

① Set volume level for amplifier device on the *78K0R – Say it!* board.
② Initialize compression/decompression library
③ Call decompression function if compression bit is 4 bit.
④ Call decompression function if compression bit is 3 bit.
⑤ Call decompression function if compression bit is 2 bit.
* Those decompression function is called with parameters of the output mode (PWM, D/A), volume level, sound size, and start address of sound data.

11.5.2 Decode Process API

The “78K0R_Sayit_VoiceDemo” project data contains the ADPCM-SP compression/decompression library (/lib/adpcmsp.r26). This library provide following API for compression/decompression process. However, the sample program uses only the decompression process.

11.5.2.1 Initialization of Decode Process API

Initialize the memory area (32 byte RAM) used for decompression process. This work area should be located in resident area. This API should be called every time before you start decompression process.

11.5.2.2 32Kbps Decompression API

Decompress 32Kbps ADPCM-SP code.

11.5.2.3 24Kbps Decompression API

Decompress 24Kbps ADPCM-SP code.

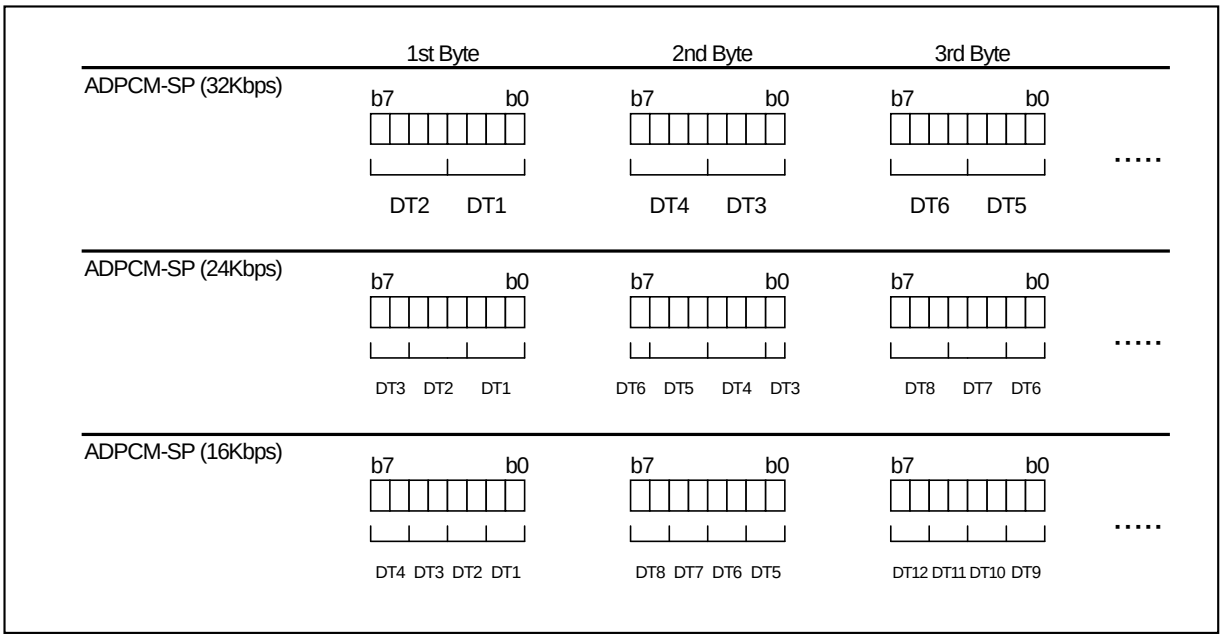
11.5.2.4 16Kbps Decompression API

Decompress 16Kbps ADPCM-SP code.

Above 4 APIs are available for decompression.

11.5.3 Format of Decompressed Data

The format of decompressed sound data is shown below.



In case of 32Kbps compression, 1 sound data is represented by 4 bits. This means, that 3 byte of memory can included 6 sound data packages.

In case of 24Kbps compression, 1 sound data is represented by 3 bits. If you just use 1 byte, it causes a fraction. This fraction should be treated as connected 3 bit with the last bits of following byte. This means that you need 3 byte of memory to include 8 sound data packages.

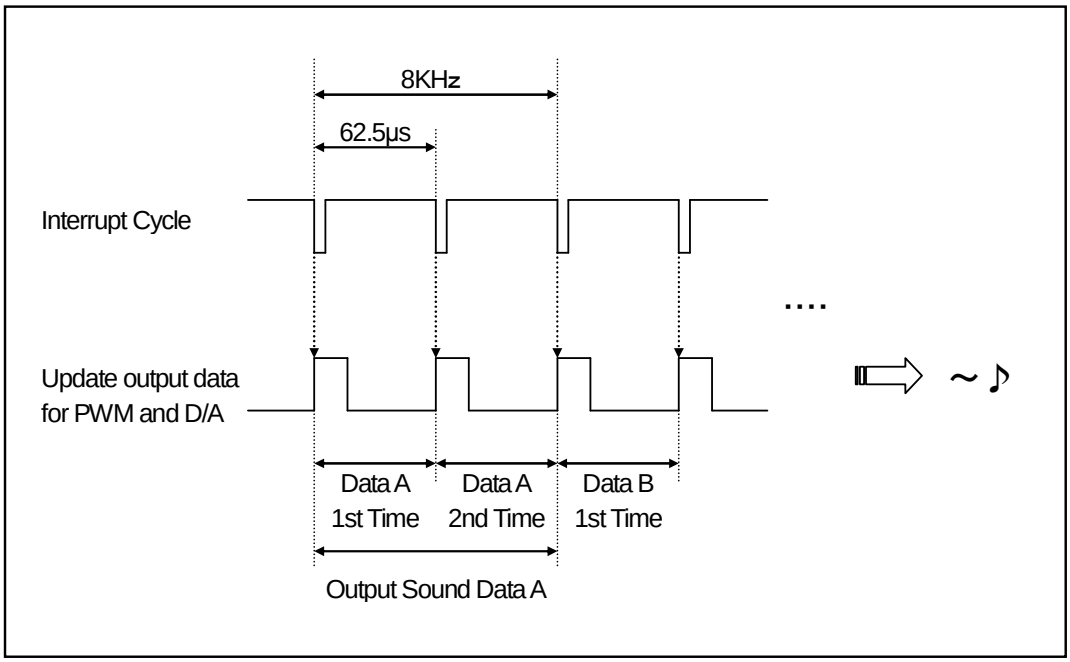
In case of 16Kbps compression, 1 data is data is represented by 2 bits. This means, that 3 byte to memory can carry 12 sound data packages.

As it mentioned previously, the compression/decompression library processes the decompression to DTn. Therefore, the application needs to gain the compression bit information of 8 bit sound data.

11.5.4 Output Cycle and Interruption

This application targets to the sampling cycle of 8KHz. This means that it plays sound data once in 125µs. It depends on speaker play band and hearing sense of people, but it means the sound in 8KHz. Based on this, the application outputs the same 16KHz (62.5µs) data for 2 times.

Following chart describes the cycle of outputs and interrupts.



11.5.5 Decode 32Kbps Compressed Data

This is the sample function to decompress 32Kbps compressed sound data. Depending on the compressed data type, it loads the compressed data, decompress, and store the playable data to the playing buffer area.

```
void vDecode_32k( U8 p_mode , U8 *p_volume , U32 size , U8 __far * adr )
{
    U32   loop    = 0;
    U8    adpcmPos = 0;
    U8    adpcmBuf;
    U8    adpcmCode;
    U16   usPCMdata;

    output_count = 0;
    if( p_mode == PWM_MODE )
    {
        START_PWM();
    }
①   else
    {
        START_DA_A();
    }
    START_625();
    vSilence( 500 , p_mode );

②   while( loop < size )
    {
        /*******************************/
        /* key check                   */
        /*******************************/
③       vKeyMon_Play(p_volume);
        /*******************************/
        /* Playing                     */
        /*******************************/
        if( ucPlaySts == STATUS_PLAY )
        {
            if( output_count <= 2 )
            {
                if( adpcmPos == 0 )
                {
                    adpcmBuf   = *adr;
                    adpcmCode  = adpcmBuf & 0x0F;
                    adpcmPos   = 1;
④               }
                else
                {
                    adpcmCode  = adpcmBuf >> 4;
                    adpcmPos   = 0;
                    adr++;
                }
⑤               usPCMdata = adpcm_l32_dec( adpcmCode , Adpcm_Work );
```

① Enable one of PWM or D/A output, and allow only interrupt of 62.5µs cycle time.
② Loop depending on the size of playing sound data.
③ Monitor key inputs.
④ Extract compressed data.
As this function decompresses 4 bit compressed data, it extracts the lower 4 bit and then the higher 4 bit compressed data.
⑤ Send the 4 bit data to decompression library, and store the data in "usPCMdata".

```
①                __disable_interrupt();
                 if( output_count == 0 )
                 {
                    if( p_mode == PWM_MODE )
                    {
                       output_data[0] = (U16)((U16)((usPCMdata)+0x8000)>>6);
                    }
                    if( p_mode == DA_MODE )
                    {
                       output_data[0] = (U8)((U16)((usPCMdata)+0x8000)>>8);
                    }
                    output_count   = 2;
                 }
                 else
                 {
                    if( p_mode == PWM_MODE )
                    {
                       output_data[1] = (U16)((U16)((usPCMdata)+0x8000)>>6);
                    }
                    if( p_mode == DA_MODE )
                    {
                       output_data[1] = (U8)((U16)((usPCMdata)+0x8000)>>8);
                    }
                    output_count   = output_count + 2;
                 }
                 __enable_interrupt();
                 loop++;
              }
           }
           /*******************************/
           /* break!                      */
           /*******************************/
②          else if( ucPlaySts==STATUS_BREAK )
           {
              break;
           }
        }
③      while( output_count!=0 );

       vVolumeControl( 0 );
       if( p_mode == PWM_MODE )
       {
          STOP_PWM();
       }
④      else
       {
          STOP_DA_A();
       }
       STOP_625();
    }
```

① Store decompressed data in playing buffer area.
② Respond to key inputs while playing.
③ Loop until it outputs all data.
④ Turn off the volume, disable PWM and D/A output, and disable interrupt after finishing playing sounds.

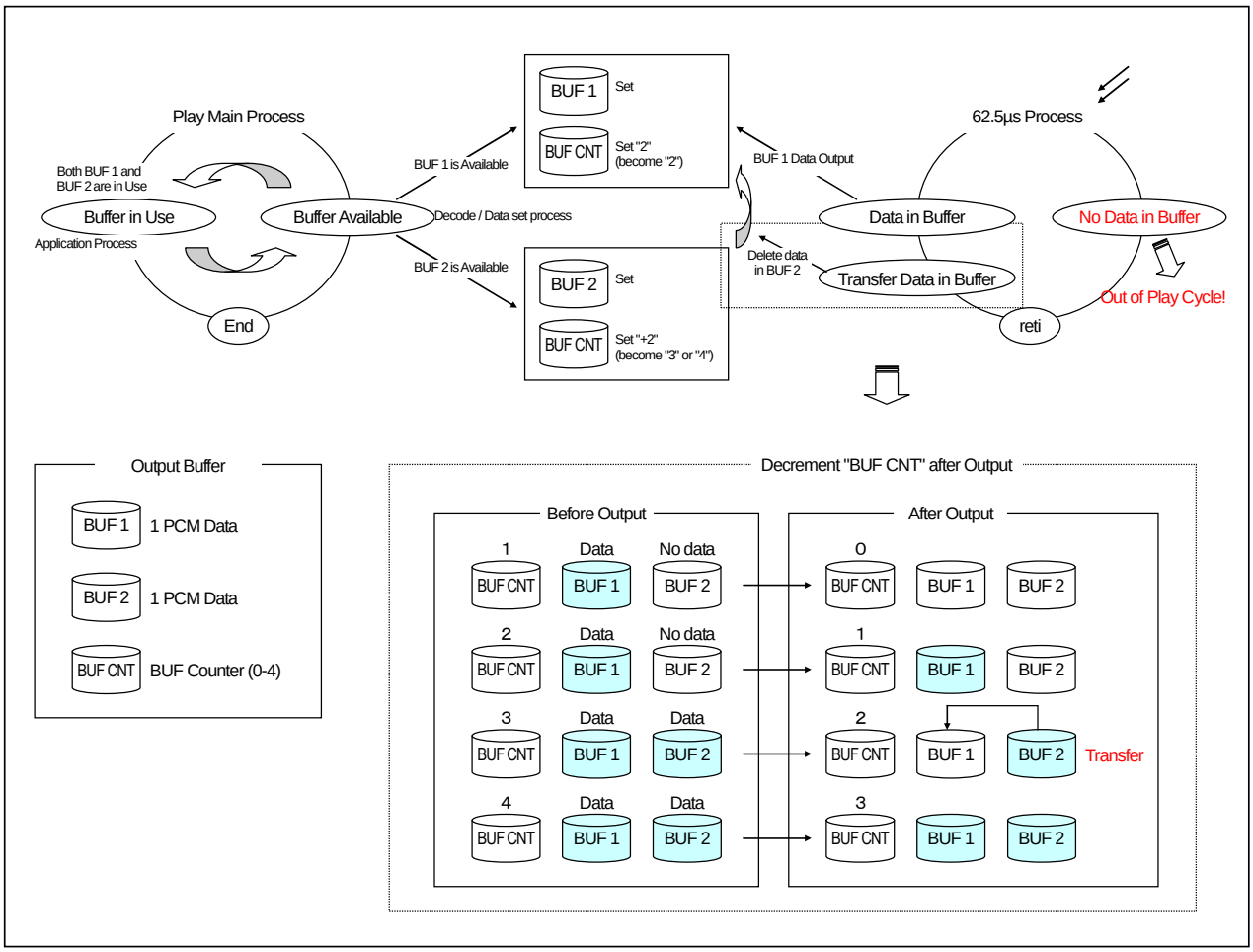
11.5.6 Sound Output Process (62.5µs Interrupt)

Those output data from decompression process is outputted by D/A or PWM with the 62.5µs Interrupt cycle.

```
    void vINTTM04_hdr( void )
    {
①      if( output_count!=0 )
       {
           if( PlayMode == PWM_MODE )
           {
               TDR01   = (U16)output_data[ 0 ];
           }
②          if( PlayMode == DA_MODE )
           {
               DACS0   =(U8)output_data[ 0 ];
           }

           output_count--;
           if( output_count==2 )
③          {
               output_data[ 0 ]  = output_data[ 1 ];
           }
       }
    }
```

① Check if there is decompressed data ready to output.
② Output by PWM or D/A depending on the output mode.
③ Transfer play data to buffer.

Play data buffer in decode process interrupt process is shown in following chart.



This application allow other process to be taken even it is waiting for the next cycle of 62.5µs output process while playing process is running (in loop).

There are 2 decompressed output data areas (output_data[0], output_data[1]) and 1 buffer counter area (output_count).

It outputs the data from lower buffer area (output_data[0]) with the cycle of 62.5µs output process.

It does not allow interrupts when it handles output buffer in decode process.

Example of Application Using Illuminance Sensor:

In this section, the application example to output Sound A when it gets lighter and Sound B when it gets darker using illuminance sensor on *78K0R – Say it!* board, will be explained. Following processes are needed to develop this application.

1. Read Current Illuminance Level by Illuminance Sensor
2. Compare Current Illuminance Level with Light/Dark reference Value
3. Output Sound A respectively Sound B when it matches with the corresponding condition

```
/*--------------------------------------------------------------------
* Function : light-sensor mode
* Descr    : void vAD_main( void )
* Inputs   : -
* Outputs  : -
* Return   : -
*--------------------------------------------------------------------*/
void vAD_main( void )
{
   U8   play_mode   = PWM_MODE;
   U8   play_num    = 2;
   U8   play_volume = 4;
   U32  readAD;
   U8   i;
   U16  cmp;
   U8   playEndflg_Low  = 0;
   U8   playEndflg_High = 0;
   U8   playEnvflg      = 0;
   U8   data_type;
   U32   sound_size;
   U8   __far *_adpcmDataAdr;

①  vLoadAdpcmData();

②  uiIntCounter1   = 0;
   __enable_interrupt();
   INT_1MSEC_ENABLE();

   while(1)
   {
      readAD   = uiADconvert();
      for( i=0 , cmp=50 ; i<8 ; i++ , cmp = cmp + 50 )
      {
         if( readAD < cmp )
         {
            vVolumeLED(i);
③           break;
         }
      }
      if( i==8 )
      {
         vVolumeLED(i);
      }
```

The function "vAD_main( )" reads current Illuminance data and reflects the data by driving the volume LEDs.

① Load sound data information from built-in memory.
*It is assumed that the sound data this application use are Sound Data areas (1) and (2)

② Enable interrupts that the application does use.

③ Use "uiADconvert （） " to convert Illuminance level to voltage value by A/D conversion.

Determine output of volume LED by the value of return value "readAD". The relationships between the A/D conversion value and LED output is shown in the table below.

A/D Conversion Value	LED Output
Conversion Value < 50	No Light
Conversion Value < 100	1 Light
Conversion Value < 150	2 Lights
Conversion Value < 200	3 Lights
Conversion Value < 250	4 Lights
Conversion Value < 300	5 Lights
Conversion Value < 350	6 Lights
Conversion Value < 400	7 Lights
Conversion Value > 400	8 Lights

Next, you are going to add conditional statement to compare with light and dark and play the different sound.

```
①       if( (i==0) && (playEndflg_Low==0) )
        {
           if( uiIntCounter1 > KEY_LONG_ON_TIME )
           {
              play_num      = 1;
              playEnvflg     = 1;
              playEndflg_Low  = 1;
              playEndflg_High = 0;
           }
        }
②       else if( (i>7) && (playEndflg_High==0) )
        {
           if( uiIntCounter1 > KEY_LONG_ON_TIME )
           {
              play_num      = 2;
              playEnvflg     = 1;
              playEndflg_Low  = 0;
              playEndflg_High = 1;
           }
        }
③       else
        {
           uiIntCounter1  = 0;
        }

④       if( (i>=3) && (i<=5) )
        {
           uiIntCounter1   = 0;
           playEndflg_Low  = 0;
           playEndflg_High = 0;
        }
```

① "i==0" indicates "dark".
To prevent for outputting the sound repeatedly by keeping the “dark” status, it contains the statement "playEndFlg_Low==0" (it is "still" dark).
With the statement "uiIntCounter1 > KEY_LONG_ON_TIME", it can check if it passed off certain period of time. "uiIntCounter1" is general purpose 1msec counter using cycle interrupt. Since "KEY_LONG_ON_TIME" is set to "1500", the cycle is more than 1.5 seconds.
When it meets the condition, it sends the play sound number with "play_num = 1", and play permission with "playEnvflg = 1" to output process.

② "i>7 （==8） " indicates "light".
In the same way, it contains the statement "playEndFlg_High==0" (it is still “light”) to prevent outputting the sound successively.
When it meets the condition, it sends the play sound number with "play_num = 2", and play permission with "playEnvflg = 1" to output process.

③ If other than above, clear the cycle interrupt counter.

④ When the Illuminance level is close to intermediate level, initialize those flags to be able to check light/dark.

```
①       if( playEnvflg==1 )
        {
            /*******************************/
            /* Prepare                     */
            /*******************************/
            data_type  = audio_adpcm[play_num-1].datatype;
            if( data_type==AUDIO_32K )
            {
               sound_size  = audio_adpcm[play_num-1].datacount * 2;
            }
            else if( data_type==AUDIO_24K )
            {
               sound_size  = (audio_adpcm[play_num-1].datacount / 3 ) * 8;
②           }
            else if( data_type==AUDIO_16K )
            {
               sound_size  = audio_adpcm[play_num-1].datacount * 4;
            }
            else
            {
               sound_size  = 0;
            }
            _adpcmDataAdr   = audio_adpcm[play_num-1].address;
            /*******************************/
            /* Play                        */
            /*******************************/
            vVolumeControl( play_volume );
            adpcm_init( Adpcm_Work );
            ucPlaySts = STATUS_PLAY;
            PlayMode  = play_mode;

            if( data_type==AUDIO_32K )
            {
③              vDecode_32k( play_mode , &play_volume , sound_size , _adpcmDataAdr );
            }
            if( data_type==AUDIO_24K )
            {
               vDecode_24k( play_mode , &play_volume , sound_size , _adpcmDataAdr );
            }
            if( data_type==AUDIO_16K )
            {
               vDecode_16k( play_mode , &play_volume , sound_size , _adpcmDataAdr );
            }
④           playEnvflg   = 0;
            uiIntCounter1 = 0;
        }
    }
}
```

① Process if it has play permission.
② Get sound data information such as sound data number, data size, data address, etc.
③ Call decompression process depending of compression data information.
④ Reset play permission flag and set "playEndflg = 0" to prevent from play repeatedly.

11.5.7 Example of A/D Conversion

This is the sample program for A/D conversion process, which is used in the illuminance sensor application. To deal with fluctuation of Illuminance sensor data, it samples the data several times.

```
/*--------------------------------------------------------------------
* Function : light-sensor control
* Descr    : U32 uiADconvert( void )
* Inputs   : -
* Outputs  : -
* Return   : 10 bit A/D Conversion Data
*--------------------------------------------------------------------*/
U32 uiADconvert( void )
{
   U16   i,j;
   U32   addata;
   U32   caladdata = 0;

①  ADCE  = 1;
   for( i=0 ; i<10000 ; i++ ) NOP5();   // wait
   ADCS  = 1;

②  ADIF  = 0;
   while( !ADIF );

   for( i=0 ; i<64 ; i++ )
   {
      caladdata  = 0;
      for( j=0 ; j<64 ; j++ )
      {
③        ADIF   = 0;
         while( !ADIF );
         caladdata  = caladdata + ADCR;
      }
      addata = addata + (caladdata >> 6);
   }
   addata  = addata >> (6+6);
   ADCS   = 0;
④  ADCE   = 0;
   return( addata );
}
```

① Start the A/D conversion. It needs to wait 1µs after setting the "ADCE".
② Skip the first conversion data.
③ Sample illuminance sensor data.
It samples 64 times, and calculates the average value.
Again, it samples 64 times, and calculates the average value.
This loop process uses about 55ms. During this time, it rounds the value from illuminance sensor (about 5 times).
④ Stop the A/D conversion.

The statement "addata = addata >> (6+6)" is the shifting process of the average data of 64 times sampling + 10 bit A/D.

11.6 Function Specifications

The specifications of functions in bundled application program are explained in this section.

11.6.1 void main(void)

Function	main()
Syntax	void main(void)
Description	Application Main Process Get the status of DIPSW and branch to appropriate process, after initializing CPU
Input Value	-
Return Value	-
Remarks	-

11.6.2 void vVoice_main(void)

Function	vVoice_main()
Syntax	void vVoice_main(void)
Description	Sound Play Application Main Process
Input Value	-
Return Value	-
Remarks	-

11.6.3 void vCPUinitialize(void)

Function	vCPUinitialize()
Syntax	void vCPUinitialize(void)
Description	Initialize CPU
Input Value	-
Return Value	-
Remarks	-

11.6.4 void vPlayPrmInitialize(void)

Function	vPlayPrmInitialize()
Syntax	void vPlayPrmInitialize(void)
Description	Initialize information table used by application
Input Value	-
Return Value	-
Remarks	-

11.6.5 void vAdpcmPrmInitialize(void)

Function	vAdpcmPrmInitialize()
Syntax	void vAdpcmPrmInitialize(void)
Description	Initialize sound data information table
Input Value	-
Return Value	-
Remarks	-

11.6.6 void vKeyMon_Standby(U8 *p_mode , U8 *p_num , U8 *p_volume)

Function	vKeyMon_Standby()
Syntax	void vKeyMon_Standby(U8 *p_mode , U8 *p_num , U8 *p_volume)
Description	Monitor key input before the start of sound play. Process depending on key input.
Input Value	U8 *p_mode (Output mode information) PWM mode : 0 D/A mode : 1 U8 *p_num (Play number information) Play number : 1-4 U8 *p_volume (Volume information) Volume : 0-8
Return Value	-
Remarks	The function does not end until key is pushed.

11.6.7 void vKeyMon_Play(U8* p_volume)

Function	vKeyMon_Play()
Syntax	void vKeyMon_Play(U8* p_volume)
Description	Monitor key input while sound is playing. Process depending on key input.
Input Value	U8* p_volume (Volume information) Volume : 0-8
Return Value	-
Remarks	-

11.6.8 void vVolumeLED(U8 lvl)

Function	vVolumeLED()
Syntax	void vVolumeLED(U8 lvl)
Description	Output volume LED
Input Value	U8 lvl (Volume infromation) Volume : 0-8
Return Value	-
Remarks	0: no light on - - - - - 8: all lights on

11.6.9 void vModeLED(U8 mode)

Function	vModeLED()
Syntax	void vModeLED(U8 mode)
Description	Output mode LED
Input Value	U8 mode (Output mode information) PWM mode : 0 D/A mode : 1
Return Value	-
Remarks	-

11.6.10 void vPlayNumLED(U8 num)

Function	vPlayNumLED()
Syntax	void vPlayNumLED(U8 num)
Description	Output play number LED
Input Value	U8 num (Play number information) Play sound number : 1-4
Return Value	-
Remarks	-

11.6.11 void vVolumeControl(U8 lvl)

Function	vVolumeControl()
Syntax	void vVolumeControl(U8 lvl)
Description	Control amplifier volume
Input Value	U8 lvl (Volume information) Volume : 0-8
Return Value	-
Remarks	Settings for amplifier (8 bit D/A) 0: 0x00 1: 0x60 2: 0x80 3: 0x90 4: 0xA0 5: 0xB0 6: 0xC0 7: 0xE0 8: 0xFF

11.6.12 void vDecode_32k(U8 p_mode , U8 *p_volume , U32 size , U8 __far * adr)

Function	vDecode_32k()
Syntax	void vDecode_32k(U8 p_mode , U8 *p_volume , U32 size , U8 __far * adr)
Description	Decompress 4 bit compressed data
Input Value	U8 p_mode (Output mode information) PWM mode : 0 D/A mode : 1 U8 *p_volume (Volume information) Volume : 0-8 U32 size (Play data size in byte) U8 __far * adr (Play data start address)
Return Value	-
Remarks	-

11.6.13 void vDecode_24k(U8 p_mode , U8 *p_volume , U32 size , U8 __far * adr)

Function	vDecode_24k()
Syntax	void vDecode_24k(U8 p_mode , U8 *p_volume , U32 size , U8 __far * adr)
Description	Decompress 3 bit compressed data
Input Value	U8 p_mode (Output mode information) PWM mode : 0 D/A mode : 1 U8 *p_volume (Volume information) Volume : 0-8 U32 size (Play data size in byte) U8 __far * adr (Play data start address)
Return Value	-
Remarks	-

11.6.14 void vDecode_16k(U8 p_mode , U8 *p_volume , U32 size , U8 __far * adr)

Function	vDecode_16k()
Syntax	void vDecode_16k(U8 p_mode , U8 *p_volume , U32 size , U8 __far * adr)
Description	Decompress 2 bit compressed data
Input Value	U8 p_mode (Output mode information) PWM mode : 0 D/A mode : 1 U8 *p_volume (Volume information) Volume : 0-8 U32 size (Play data size in byte) U8 __far * adr (Play data start address)
Return Value	-
Remarks	-

11.6.15 void vLoadAdpcmData(void)

Function	vLoadAdpcmData()
Syntax	void vLoadAdpcmData(void)
Description	Extract sound data to sound data information table
Input Value	-
Return Value	-
Remarks	-

11.6.16 void vINTTM04_hdr(void)

Function	vINTTM04_hdr()
Syntax	void vINTTM04_hdr(void)
Description	62.5µs cycle interrupt handler Trigger to update output sound data
Input Value	-
Return Value	-
Remarks	-

11.6.17 void vINTTM05_hdr(void)

Function	vINTTM05_hdr()
Syntax	void vINTTM05_hdr(void)
Description	1ms cycle interrupt handler Used as general purpose timer Monitor joystick key input
Input Value	-
Return Value	-
Remarks	-

11.6.18 void vBeep_sample(void)

Function	vBeep_sample()
Syntax	void vBeep_sample(void)
Description	Sample function to output beep
Input Value	-
Return Value	-
Remarks	-

11.6.19 void vAD_main(void)

Function	vAD_main()
Syntax	void vAD_main(void)
Description	Get illuminance sensor data
Input Value	-
Return Value	-
Remarks	-

11.6.20 void vSilence(U32 msec , U8 p_mode)

Function	vSilence()
Syntax	void vSilence(U32 msec , U8 p_mode)
Description	Output silence
Input Value	U32 msec (output time in ms) Valid range : 1-10000 (10 seconds) U8 p_mode (Output mode) PWM mode : 0 D/A mode : 1
Return Value	-
Remarks	-

11.6.21 void vBeep(U8 p_mode , U8 tone , U32 msec)

Function	vBeep()
Syntax	void vBeep(U8 p_mode , U8 tone , U32 msec)
Description	Output beep
Input Value	U8 p_mode (Output mode) PWM mode : 0 D/A mode : 1 U8 tone (Output sound tone) high-pitched : 1 low-pitched : 0 U32 msec (output time in ms) Valid range : 1-10000 (10 seconds)
Return Value	-
Remarks	-

11.6.22 U32 uiADconvert(void)

Function	uiADconvert()
Syntax	U32 usADconvert(void)
Description	Process 10 bit A/D conversion
Input Value	-
Return Value	Conversion result
Remarks	-

11.7 Macro Specifications

The specifications of macros in bundled application program are explained in this section.

11.7.1 START_PWM()

Macro	START_PWM()
Description	Start PWM
Source File	78K0R_Voice.c
Input Value	-
Return Value	-
Remarks	-

11.7.2 STOP_PWM()

Macro	STOP_PWM()
Description	Stop PWM
Source File	78K0R_Voice.c
Input Value	-
Return Value	-
Remarks	-

11.7.3 START_DA_A()

Macro	START_DA_A()
Description	Start D/A output (volume)
Source File	78K0R_Voice.c
Input Value	-
Return Value	-
Remarks	-

11.7.4 STOP_DA_A()

Macro	STOP_DA_A()
Description	Stop D/A output (volume)
Source File	78K0R_Voice.c
Input Value	-
Return Value	-
Remarks	-

11.7.5 START_625()

Macro	START_625()
Description	Enable 62.5µs cycle interrupt
Source File	78K0R_Voice.c
Input Value	-
Return Value	-
Remarks	-

11.7.6 STOP_625()

Macro	STOP_625()
Description	Disable 62.5µs cycle interrupt
Source File	78K0R_Voice.c
Input Value	-
Return Value	-
Remarks	-

11.7.7 DA_LED(x)

Macro	DA_LED(x)
Description	Turn ON/OFF D/A output mode LED
Source File	common.h
Input Value	LED_ON : Light ON LED_OFF : Light OFF
Return Value	-
Remarks	-

11.7.8 PWM_LED(x)

Macro	PWM_LED(x)
Description	Turn ON/OFF PWM output mode LED
Source File	common.h
Input Value	LED_ON : Light ON LED_OFF : Light OFF
Return Value	-
Remarks	-

11.7.9 PLAY_LED(x)

Macro	PLAY_LED(x)
Description	Turn ON/OFF PLAY LED
Source File	common.h
Input Value	LED_ON : Light ON LED_OFF : Light OFF
Return Value	-
Remarks	-

11.7.10 PLAY1_LED(x)

Macro	PLAY1_LED(x)
Description	Turn ON/OFF play number 1 LED
Source File	common.h
Input Value	LED_ON : Light ON LED_OFF : Light OFF
Return Value	-
Remarks	-

11.7.11 PLAY2_LED(x)

Macro	PLAY2_LED(x)
Description	Turn ON/OFF play number 2 LED
Source File	common.h
Input Value	LED_ON : Light ON LED_OFF : Light OFF
Return Value	-
Remarks	-

11.7.12 PLAY3_LED(x)

Macro	PLAY3_LED(x)
Description	Turn ON/OFF play number 3 LED
Source File	common.h
Input Value	LED_ON : Light ON LED_OFF : Light OFF
Return Value	-
Remarks	-

11.7.13 PLAY4_LED(x)

Macro	PLAY4_LED(x)
Description	Turn ON/OFF play number 4 LED
Source File	Common.h
Input Value	LED_ON : Light ON LED_OFF : Light OFF
Return Value	-
Remarks	-

11.7.14 VLM_LED(x)

Macro	VLM_LED(x)
Description	Turn ON/OFF volume LED
Source File	common.h
Input Value	ucAmpLebelTable[9] table value
Return Value	-
Remarks	-

11.7.15 DIPSW()

Macro	DIPSW()
Description	Get DIPSW status
Source File	common.h
Input Value	-
Return Value	0-7 (ON: 1) to show DIPSW8,7,6 (3 bit)
Remarks	-

11.7.16 JOYSTK()

Macro	JOYSTK()
Description	Get joystick status
Source File	common.h
Input Value	-
Return Value	Status of joystick with 5 bit (ON: 1) BIT 0: UP Key BIT 1: PUSH Key BIT 2: LEFT Key BIT 3: RIGHT Key BIT 4: DOWN Key
Remarks	-

11.8 Variable/Constant Specifications

The specifications of variables/constants in bundled application program are explained in this section.

11.8.1 Adpcm_Work[16]

Variable/Constant	Adpcm_Work[16]
Declaration	static U16 Adpcm_Work[16]
Source File	78K0R_Voice.c
Use	Work area for compression/decompression library
Remarks	This should be located in resident area

11.8.2 output_data[2]

Variable/Constant	output_data[2]
Declaration	static U16 output_data[2]
Source File	78K0R_Voice.c
Use	Output data buffer Store output data
Remarks	It outputs "output_data[0]" with the cycle interrupts

11.8.3 output_count

Variable/Constant	output_count
Declaration	static U8 output_count
Source File	78K0R_Voice.c
Use	62.5µs output counter This indicates the status of output data buffer
Remarks	-

11.8.4 ucPlaySts

Variable/Constant	ucPlaySts
Declaration	static U8 ucPlaySts
Source File	78K0R_Voice.c
Use	Application play status Before Play : STATUS_STANDBY Playing : STATUS_PLAY Stop Playing : STATUS_STOP Break : STATUS_BREAK
Remarks	-

11.8.5 stop_Led

Variable/Constant	stop_Led
Declaration	static U8 stop_Led
Source File	78K0R_Voice.c
Use	LED blinking action when play is stopped
Remarks	-

11.8.6 PlayMode

Variable/Constant	PlayMode
Declaration	static U8 PlayMode
Source File	78K0R_Voice.c
Use	To determine output mode (PWM, D/A) with interrupt process
Remarks	-

11.8.7 uiIntCounter1

Variable/Constant	uiIntCounter1
Declaration	U32 uiIntCounter1
Source File	78K0R_Voice.c
Use	General purpose 1msec timer
Remarks	-

11.8.8 uiIntCounter2

Variable/Constant	uiIntCounter2
Declaration	U32 uiIntCounter2
Source File	78K0R_Voice.c
Use	General purpose 1msec timer
Remarks	-

11.8.9 usKeyStsCount[5]

Variable/Constant	usKeyStsCount[5]
Declaration	U16 usKeyStsCount[5]
Source File	78K0R_Voice.c
Use	Joystick status counter Increment when ON with 1msec cycle handler usKeyStsCount[0] : UP Key continuous ON time in msec usKeyStsCount[1] : PUSH Key continuous ON time in msec usKeyStsCount[2] : LEFT Key continuous ON time in msec usKeyStsCount[3] : RIGHT Key continuous ON time in msec usKeyStsCount[4] : DOWN Key continuous ON time in msec
Remarks	-

11.8.10 ucKeyStsLocked[5]

Variable/Constant	ucKeyStsLocked [5]
Declaration	U8 ucKeyStsLocked [5]
Source File	78K0R_Voice.c
Use	Joystick lock flag Set when continuous ON time exceeds certain value with 1msec cycle handler ucKeyStsLocked [0]: KEY_SHORT_ON (UP Key ON locked) : KEY_SHORT_OFF (UP Key OFF) ucKeyStsLocked [1]: KEY_SHORT_ON (PUSH Key ON locked) : KEY_LONG_ON (PUSH Key Long-ON locked) : KEY_SHORT_OFF (PUSH Key OFF) ucKeyStsLocked [2]: KEY_SHORT_ON (LEFT Key ON locked) : KEY_SHORT_OFF (LEFT Key OFF) ucKeyStsLocked [3]: KEY_SHORT_ON (RIGHT Key ON locked) : KEY_SHORT_OFF (RIGHT Key OFF) ucKeyStsLocked [4]: KEY_SHORT_ON (DOWN Key ON locked) : KEY_SHORT_OFF (DOWN Key OFF)
Remarks	-

11.8.11 voice_adpcm[4]

Variable/Constant	voice_adpcm[4]
Declaration	static struct tstsSOUND voice_adpcm[4]
Source File	78K0R_Voice.c
Use	Sound data information array U8 datatype (compression bit information) : 2 (ADPCM SP 16K 8x2bit) : 3 (ADPCM SP 24K 8x3bit) : 4 (ADPCM SP 32K 8x4bit) U32 datacount : Data count information U8 __far *address: Data address information
Remarks	-

11.8.12 ucAmpLevelTable[9]

Variable/Constant	ucAmpLevelTable[9]
Declaration	static const U8 ucAmpLevelTable[9]
Source File	78K0R_Voice.c
Use	Amplifier volume level table (D/A output value) ucAmpLevelTable[0] : VLVL_AMP0 ucAmpLevelTable[1] : VLVL_AMP1 ucAmpLevelTable[2] : VLVL_AMP2 ucAmpLevelTable[3] : VLVL_AMP3 ucAmpLevelTable[4] : VLVL_AMP4 ucAmpLevelTable[5] : VLVL_AMP5 ucAmpLevelTable[6] : VLVL_AMP6 ucAmpLevelTable[7] : VLVL_AMP7 ucAmpLevelTable[8] : VLVL_AMP8
Remarks	-

11.8.13 ucVolumeLevelLEDTable[9]

Variable/Constant	ucVolumeLevelLEDTable[9]
Declaration	static const U8 ucVolumeLevelLEDTable[9]
Source File	78K0R_Voice.c
Use	Volume level LED output table (I/O output value) ucVolumeLevelLEDTable[0] : VLVL_LED0 ucVolumeLevelLEDTable[1] : VLVL_LED1 ucVolumeLevelLEDTable[2] : VLVL_LED2 ucVolumeLevelLEDTable[3] : VLVL_LED3 ucVolumeLevelLEDTable[4] : VLVL_LED4 ucVolumeLevelLEDTable[5] : VLVL_LED5 ucVolumeLevelLEDTable[6] : VLVL_LED6 ucVolumeLevelLEDTable[7] : VLVL_LED7 ucVolumeLevelLEDTable[8] : VLVL_LED8
Remarks	-

12. Cables

12.1 USB interface cable (Mini-B type)

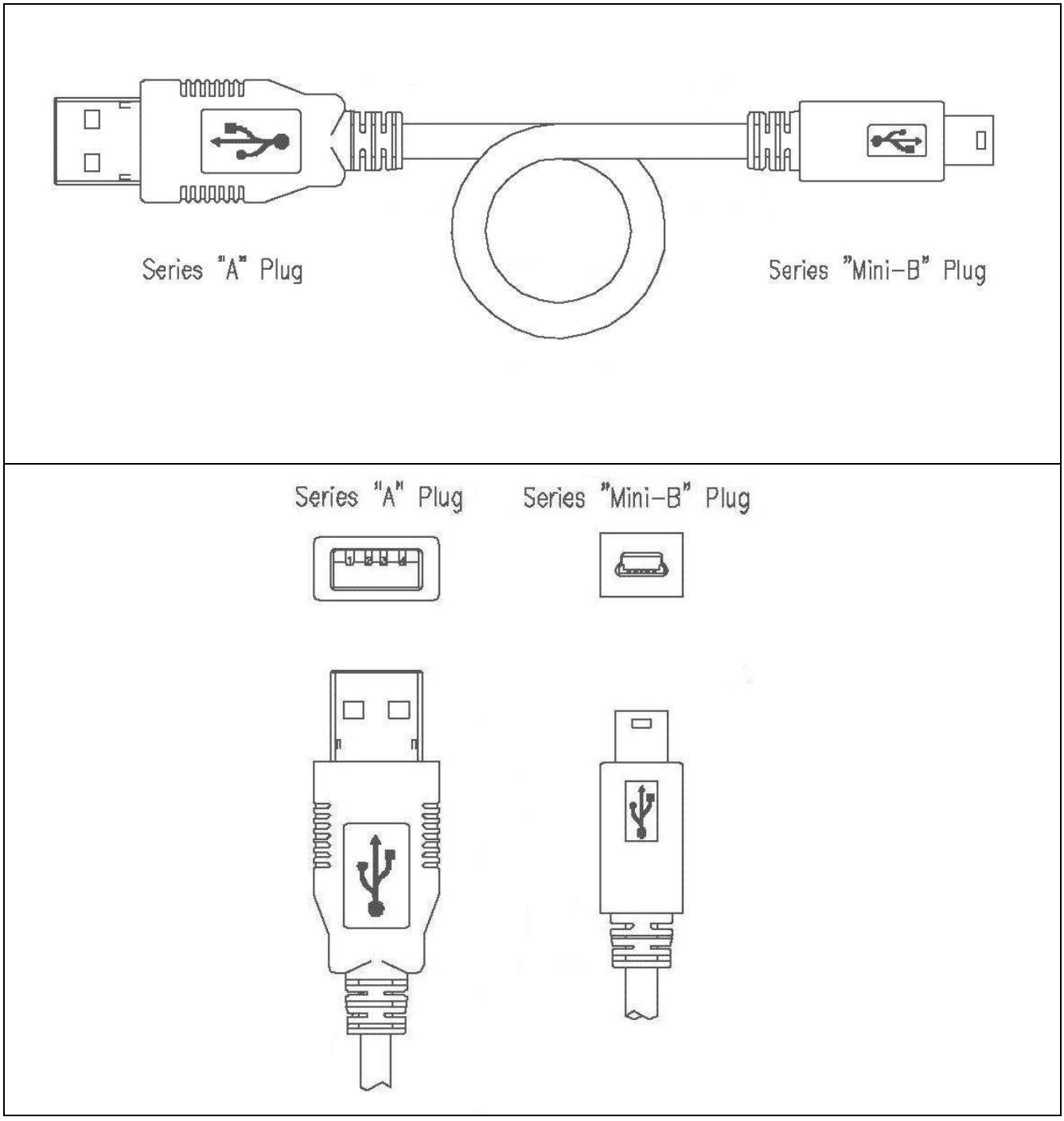


Figure 35: USB interface cable (Mini-B type)

13. Schematics

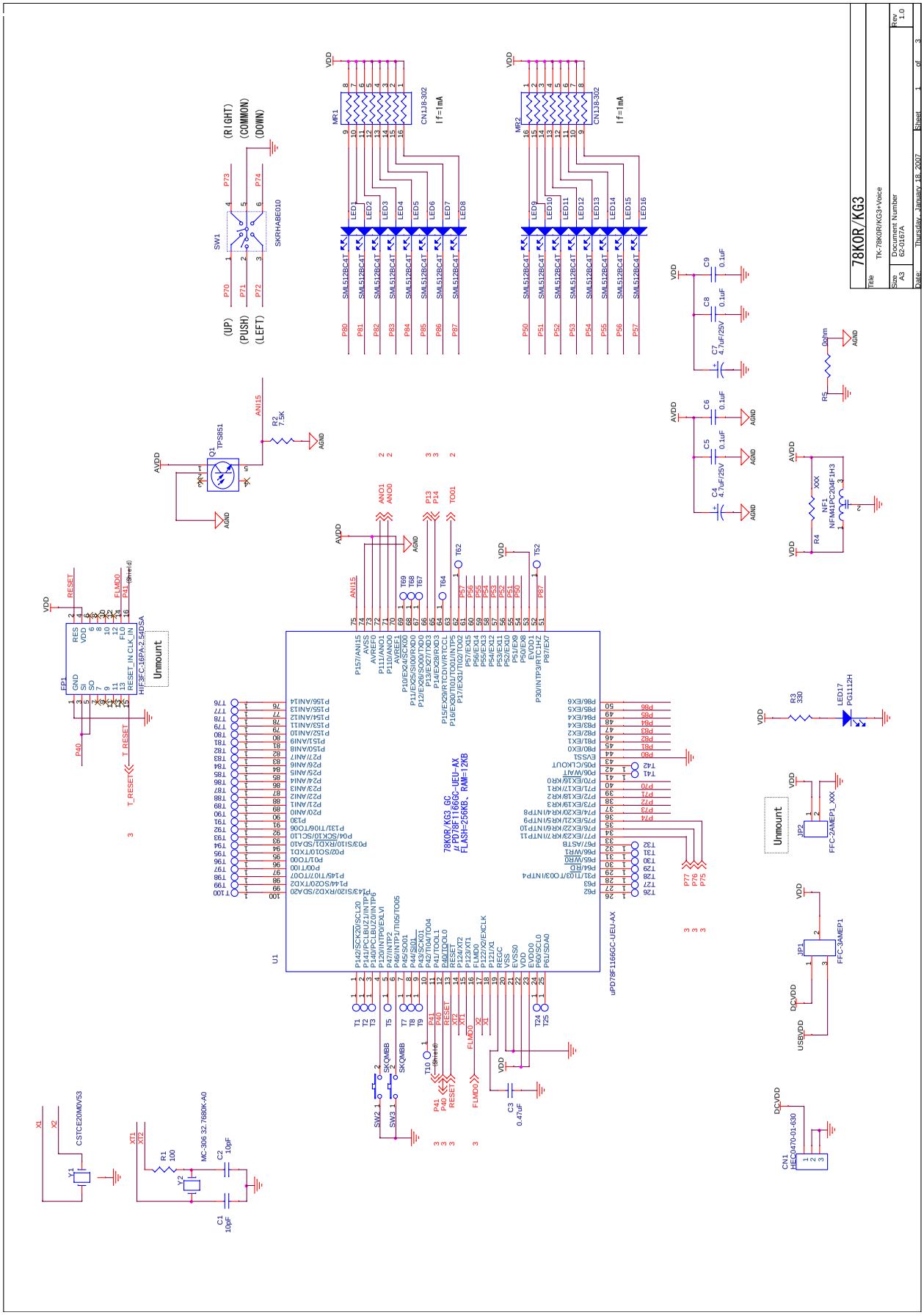


Figure 36: *78K0R - Say it!* schematics 1/3

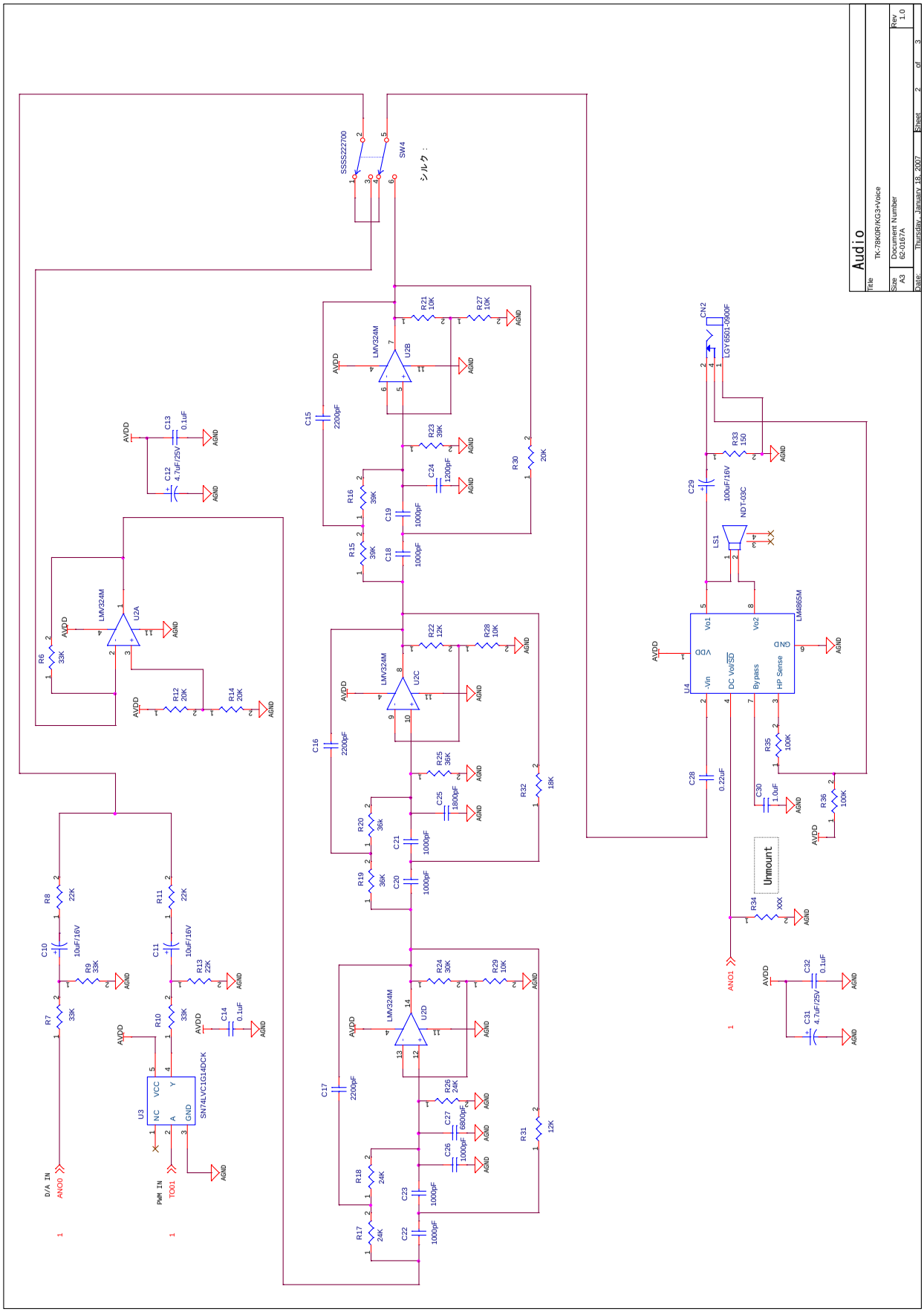


Figure 37: *78K0R - Say it!* schematics 2/3

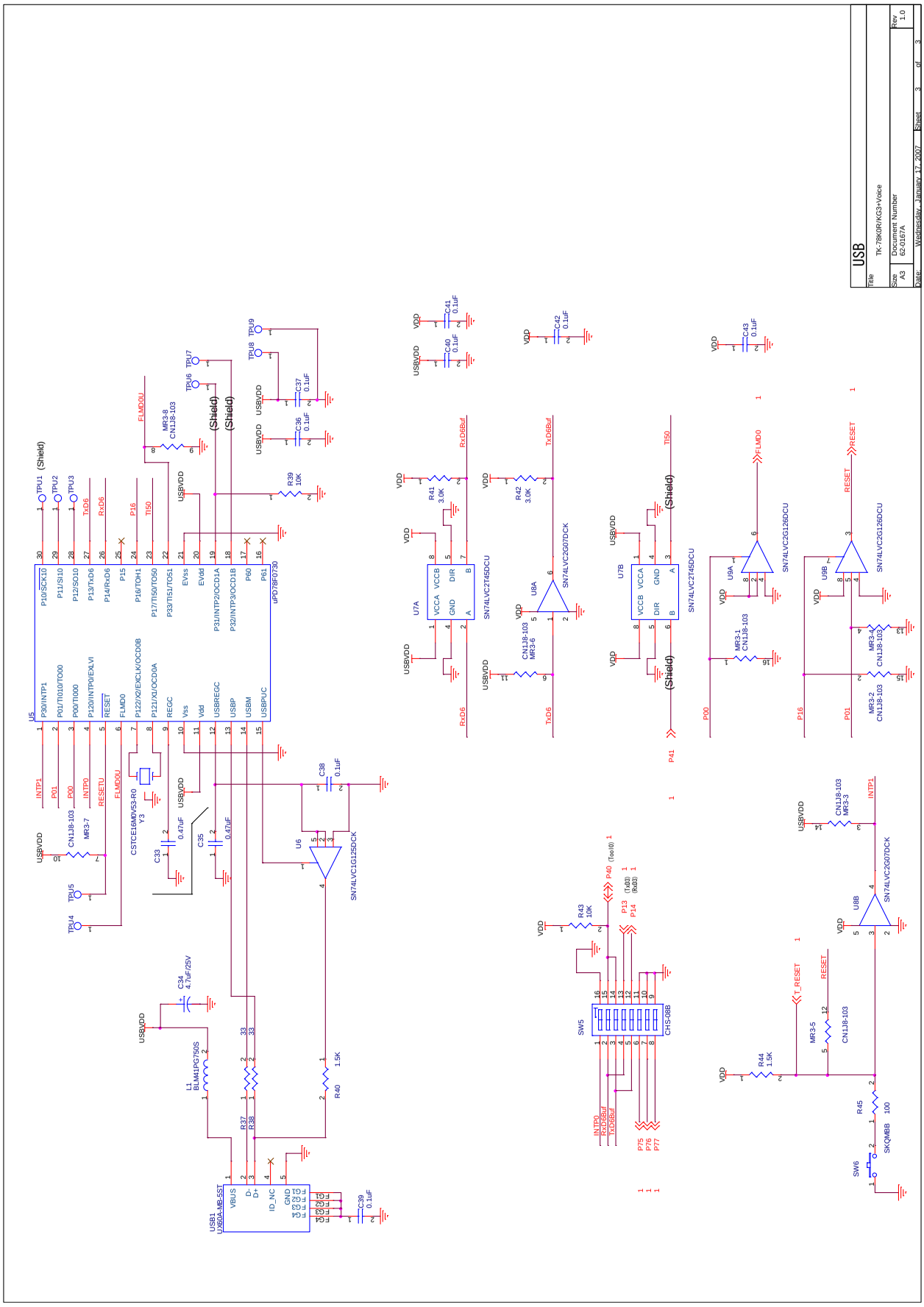


Figure 38: *78K0R - Say it!* schematics 3/3

[MEMO]