USER'S GUIDE

EM57000 Series *V3.5*

Integrtated Development System

ELAN MICROELECTRONICS CORP.

Trademark Acknowledgments

IBM is a registered trademark and PS/2 is a trademark of IBM.

Microsoft, MS, MS-DOS, and Windows are registered trademarks of Microsoft Corporation.

Easy Sound is a registered trademark of ELAN Microelectronics Corp.

Printed in Taiwan, ROC 2001

ELAN MICROELECTRONICS CORPORATION

Hsinchu Headquarters
No. 12, Innovation 1sr Rd.,
Science-based Industrial Park
Hsinchu 300, Taiwan, ROC
Phone: +886 3 563-9977
Fax: +886 3 578-0617
http:// www.emc.com.tw

Hong Kong Office
Rm.1005B, 10/F., Empire Centre, 68 Mody Road.,
Tsimshatsui, Kowloon,
Hong Kong
Phone: +852 2838-8715
Fax: +852 2838-0497

CONTENT

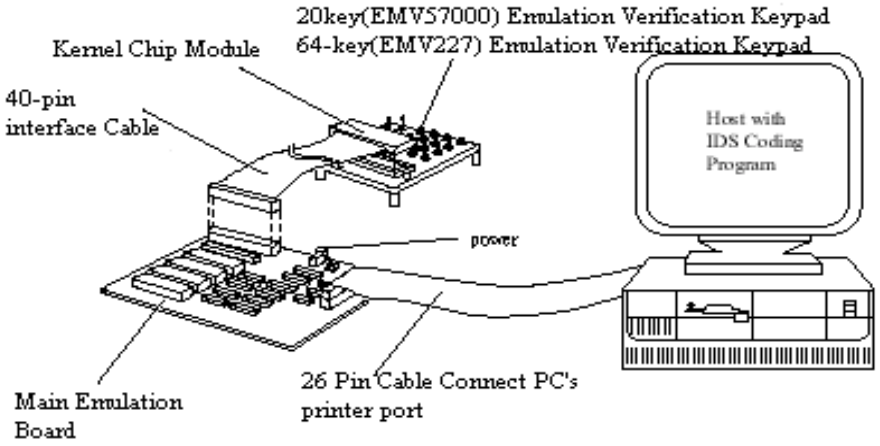
Chapter 1

Overview

The EM57000 Integrated Development System (IDS) is a user-friendly emulation tool that is used to edit application programs, coding voice/melody/sound effect files, and generate emulation/layout files for EM57000 series of EMC chips.

When connected to a host computer running under IDS Coding Program, user will be able to monitor, test and evaluate on-line, the voice/melody/sound effect coding results and modify the program to satisfy user‘s requirements. The final layout file can then be applied and used for commercial production of the EM57000 series IC’s.

1.1 Hardware Architecture



All Overview of EM57000 integrated Development System(IDS)

The complete set of IDS system comprises of the following hardware modules, cables (see figure above), and software**:**

- A Main Emulation Board 2.1
- A Verification Module, consisting of Kernel Chip Module (EMM57000) and the 20-Key Emulation Verification Keypad (EMV57000) or the 64-Key Emulation Verification Keypad (EMV227KB)
- An IDS to PC Interface Cable with 26-pin printer port
- A Main Emulation Board to Verification Module Interface Cable with 40-pin connectors at both ends
- One 3.5” diskette containing the IDS Coding Program

1.1.1 Main Emulation Board

The Main Emulation Board connects to the host computer through the printer port and the interface cable

The emulator houses the components that allow the IDS to perform coding without the traditional burning of EPROM‘s. This method saves time and effort in the development of EM57000 series chips. It supports emulation for all series of EM57000 chips.

User can download program and voice data from host to Main Emulation Board. It then tests and verifies functional quality of such data in the Verification Module.

1.1.2 EM57000 Verification Module

This module is an assembly of the Kernel Chip Module (EMM57000) and one of the following**:**

- 20-Key Emulation Verification Keypad (EMV57000)
- 64-Key Emulation Verification Keypad (EMV227KB)

The Verification Module assembly validates EM57000 series chips performance with the newly developed voice and program in Main Emulation Board.

The Kernel Chip Module is installed piggy-back on the EMV Keypad and houses the kernel chip, peripheral connectors, and DIP socket that connects the Main Emulation Board to Verification Board. Programmed EPROM is also plugged on the DIP socket during Stand-Alone operation.

The Emulation Verification Keypad houses the keypad that is used to test the audio results (with user provided speaker installed) of the compiled program on the Kernel Chip Module. Both the 20-Key and 64-Key EMV Keypads are provided to meet both users' simple and complex applications.

The Verification Module assembly may be operated under stand-alone configuration to evaluate a newly developed voice/melody/sound effect program loaded in an EPROM. The EPROM is plugged into the Kernel Chip Module DIP socket.

1.2 Introduction to IDS Coding Program

The IDS Coding Program is a text windowed coding program, designed for EM57000 series chips development. It runs under WIN95 environment. The program performs editing, conversions, and compilation of voice, melody and sound effect files under various formats, such as MLD (melody), SND (sound effect), ASM (assembly), and ESY (easy) formats.

See Chapter 4 to 6 for the detailed description of working with the IDS Coding Program under the various formats mentioned above.

1.3 Introduction to EM57000 Series IC's

EM57000 series are the latest generation of EMC's voice IC's. The following are its exceptional features that make the chips excel other voice IC's in the market today:

- Programmable
- Powerful I/O
- Full Voice, Melody and Sound Effect Capability
- Unique embedded Tiny Controller. This controller allows arbitrary configuration of I/O and chunking of voice into sections for replay. This in turn enables the voice IC to generate complex sound and LED effect.
- Built-in Melody/ Sound Effect Generator that generates musical and special sound effects

Refer to Appendix C for the pin assignments and electrical specifications of the chips.

1.3.1 EM57001

EM57001 is a single chip Voice/Dual tone Melody/Dual tone Sound Effect synthesizer IC. It contains two 4-bit I/O ports and a tiny controller. User's applications, such as section combination, trigger modes, control outputs, keyboard matrix, and other logic functions, are easily implemented by programming through its tiny controller.

1.3.1.1 What's Unique with EM57001?

- Single power supply 2.4 V ~ 5 V
- 10Kx10 bits ROM (program and data)
- Two 4-bit I/O ports and 16x4 bits RAM
- 8K (maximum) program ROM
- One 6-bit timer overflow control
- ASPCM synthesizer and dual tone melody/sound effect generator
- 4K~32K Hz playing speed for voice play-back
- Multiple tempos for dual tone melody/sound effect play-back
- Variable beats for dual tone melody/sound effect play-back
- Multiple levels of volume control
- 8 steps variable current D/A output to drive external connected transistor for voice output

1.3.2 EM57100 ~ EM57700

EM57100 ~ EM57700 is a series of single chip Voice/Dual tone Melody/Dual tone Sound Effect synthesizer IC. It contains one 4-bit input port, two 4-bit I/O ports, and a tiny controller. User's applications, such as section combination, trigger modes, control

outputs, keyboard matrix, IR communication and other logic functions, are easily implemented by programming through its tiny controller.

1.3.2.1 What's Unique with EM57100 ~ EM57700?

- Single power supply 2.4 V ~ 5 V
- Wide range of ROM (program and data)
  - EM57100 – 16Kx10 bits
  - EM57200 – 32Kx10 bits
  - EM57300 – 64Kx10 bits
  - EM57400 – 128Kx10 bits
  - EM57500 – 256Kx10 bits
  - EM57600 – 512Kx10 bits
  - EM57700 – 1024Kx10 bits
- One 4-bit input port, two 4-bit I/O ports, 32x4 bits RAM for EM57100~57500 and 128x4 bits RAM for EM57600~57700
- 8K (maximum) program ROM for EM57100~57500 and 32K (maximum) program ROM for EM57600~57700
- One 6-bit timer overflow control
- ASPCM synthesizer and dual tone melody/sound effect generator
- 4K~32K Hz playing speed for voice play-back
- Multiple tempos for dual tone melody/sound effect play-back
- Variable beats for dual tone melody/sound effect play-back
- Multiple levels of volume control
- 8 steps variable current D/A output to drive external connected transistor for voice output
- A pin (except EM57400) can be programmed as an IR communication pin which generate 38k Hz carrier.

Chapter 2

System Installation

2.1 Required User Provided Equipment

2.1.1 Host Computer

The IDS requires a host that meets the following configuration**:**

- IBM PC AT (**486** or higher level) or compatible computers
- **Win95，Win98，Win2000，WindowsNT**
- To recommend **800*600** pixels and **Small Fonts**
- **10M** or more free hard disk space
- Mouse is optional but highly recommended

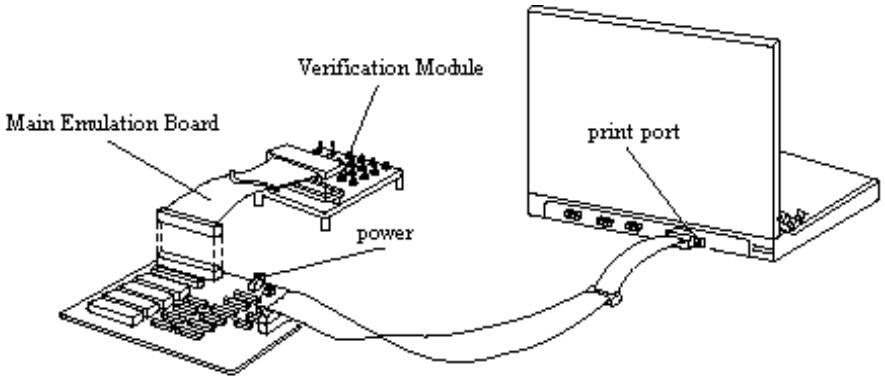
2.1.2 External Power Source

Requires power source of +5.0VDC (power adapter or battery) to provide power for Main Emulation Board and Verification Module when IDS is connected to host through printer port.

2.1.3 Audio Speaker

An 8Ω external speaker for audio testing of the user's program on EM57000 Kernel Chip Module (EMM57000).

2.2 Connecting IDS with Host Computer



Linking IDS with Host Computer

The IDS connects to the host through the IDS Emulation Board to PC Interface Cable. As illustrated above, the 26-pin connector at one end of the cable is plugged into the PRN-CON connector of the Main Emulation Board. At computer side, the other end of the cable is connected to the host printer port as shown above (using the cable 26-pin connector). This, however, will require an external +5VDC power source (battery or power adapter) to provide power to Main Emulation Board and Verification Module.

WARNING!

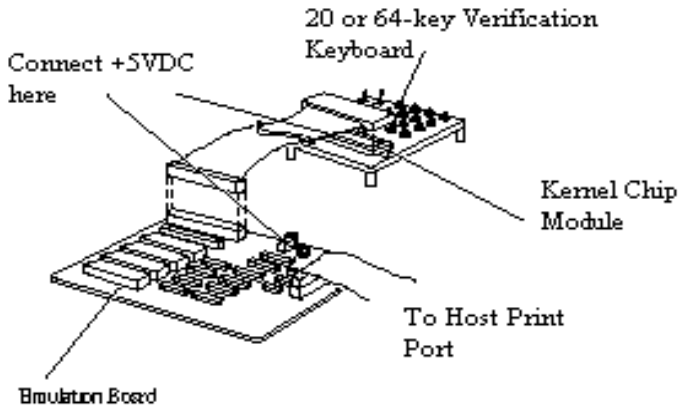
Before applying power to host or emulator board, ensure that all Interface Cable connectors are correctly plugged into their corresponding connectors on host and Emulation Board to prevent damage to the equipment.

2.3 Connecting Emulation Board with Verification Module

NOTE

Remove EPROM and external power (if installed) from the EMM57000 Kernel Chip Module.

1. Plug the 40-pin male connector of the Emulation Board to Verification Module Interface Cable to DIP socket on the Kernel Chip Module (EMM57000).
2. Then plug the female connector at the other end of the interface cable to V-ROM socket on Emulation Board as illustrated in the following figure.
3. Referring to the schematic diagram of EMM57000 in Section 2.4, set Jumper JP1 to "4M" position and Jumper JP2 to "1M" position on the Kernel Chip Module for Emulation Mode operation.



Linking Emulation Board with EM57000 Verification Module

2.3.1 Emulation Verification Installation

The emulation verification (EMV) is used the verification of EM57000 series. To use EMV, the "P10" "P11" male connectors of the kernel chip must plug the "P10" "P11" female connectors of EMV. The left four female connectors in EMV don't care them. The following display the operation of eight jumpers in EMV. The application circuit refers to section 9.1.2:

1.To use the EM57001, user must set eight jumpers to "EM57001" position.

2.To use the EM57100~EM57700, user must set eight jumpers to "EM57100" position.

2.3.2 Optional Speaker Installation

To provide audible voice, melody, or sound effect of user's application, user may connect an 8Ω external speaker to the

EMM57000 Kernel Chip Module at connector labeled “SP” as shown in the Section 2.4.

> ***NOTE***
>
> *If the played voice is distorted, do one of the following:*
>
> 1. *Connect a resistor at “Rb” location (see Section 2.4).*
> 2. *Check and change externally connected battery if found low in power.*

2.4 Installation for Stand Alone Operation

EPROM written with binary files generated by IDS Coding Program may be tested on Verification Module under stand-alone configuration (without connecting to Emulation Board and PC). For optimum power saving efficiency, use CMOS EPROM with fast access time of 150ns.

Stand-alone configuration is accomplished with the following steps and referring to the figure below**:**

1. Plug EPROM on the Kernel Chip Module (EMM57000) DIP socket.

> ***NOTE***
>
> *EPROM must be inserted before applying power*

2. Connect an 8Ω external speaker to the “SP” connector on the Kernel Chip Module.
3. Connect +5VDC power to power (±) connector on the Kernel Chip Module.
4. The left four jumpers of Kernel Chip Module don’t move to the “Green” position.
5. If the size of EPROM is lower than “4M”. The IDS coding program will generate a binary file (XX.BIN). And the kernel chip jumpers is accomplished with the following steps.
   a. Set Jumper JP1 to “4M” position.
   b. Set Jumper JP2 to “1M” or “512K” position depending on actual type of EPROM installed, as follows**:**

EPROM Type	JP2 Position
27C010 / 27C4001	1M
27C256 / 27C512	512K

6. If the size of EPROM is larger than “4M”. The IDS coding program will generate binary files (XX.BIN, XX.B40,

XX.B41, XX.B42, XX.B43 or XX.B80, XX.B81) according to the size of the binary file. And the kernel chip jumpers is accomplished with the following steps.

a. When the kernel chip connects two 4M EPROM chips, the jumpers “JP1” and “JP2” must connect the “4M” and “1M” position, and the “CE” of the second EPROM connects “CE41”. The binary files (XX.B40 and XX.B41) must be separately written to the first EPROM and the second EPROM

b. When the kernel chip connects three 4M EPROM chips, the jumpers “JP1” and “JP2” must connect the “4M” and “1M” position, the “CE” of the second EPROM connects “CE41”, and the “CE” of the third EPROM connects “CE42”. The binary files (XX.B40, XX.B41 and XX.B42) must be separately written to the first EPROM, the second EPROM and the third EPROM.

c. When the kernel chip connects four 4M EPROM chips, the jumpers “JP1” and “JP2” must connect the “4M” and “1M” position, the “CE” of the second EPROM connects “CE41”, the “CE” of the third EPROM connects “CE42”, and the “CE” of the fourth EPROM connects “CE43”. The binary files (XX.B40, XX.B41, XX.B42 and XX.B43) must be separately written to the first EPROM, the second EPROM, the third EPROM and the fourth EPROM.

d. When the kernel chip connects one 8M EPROM chip, the jumpers “JP1” and “JP2” must connect the “8M” and “1M” position. The binary file (XX.BIN) must be written to the EPROM.

e. When the kernel chip connects two 8M EPROM chips, the jumpers “JP1” and “JP2” must connect the “8M” and “1M” position, the “CE” of the second EPROM connects “CE81”. The binary files (XX.B80 and XX.B81) must be separately written to the first EPROM and the second EPROM.

NOTE

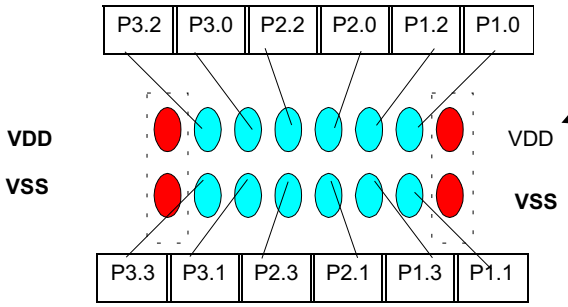
You can carry the EPROM one-on-one by bending out the CE pins and connect them to the contacts on the EMM57000 modules

NOTE

If the played voice is distorted, do one of the following:

1. *Connect a resistor at "Rb" location (see figure above).*
2. *Check and change externally connected battery if found low in power.*

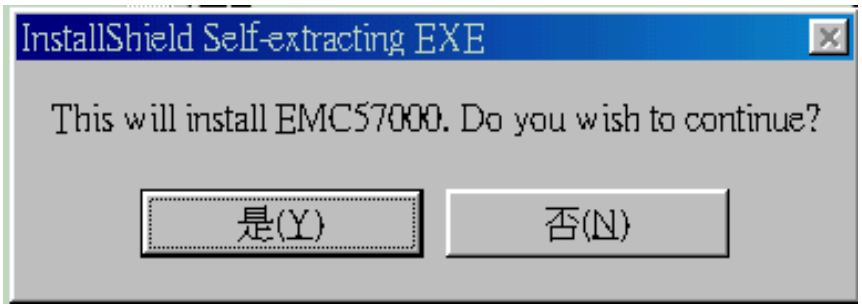
With everything correctly set as described above, start pressing the Verification Module Keypad keys according to your requirements.



EMM57000 I/O Port Assignment for Stand-Alone Operation

2.5 IDS Coding Program Installation

Insert the IDS Coding Program diskette into the floppy disk drive of the host computer and execute setup program. The install program will copy files into a directory in the hard disk.



After install the main program of EM57000, it show a dialog for user to continue install supporting program. If user's system is NT or Win2000 , it is necessary to install this supporting program.

Refer to Chapter 3 for getting started and operation of the program

Chapter 3

Getting Started with IDS Coding Program

3.1 IDS Coding Program Installation

Install program as described in Section 2.5 of Chapter 2.

3.2 User Source Files Format

The IDS Coding Program requires that user's source files conform to the following format**:**

File Type	Required File Extension	Description
Assembly Program	USERNAME**.ASM**	Assembly program to control EM57000 series chips.
Easy Format Program	USERNAME**.ESY**	Easy format program to control EM57000 series chips.
Melody File	USERNAME**.MLD**	Text type melody section file.
	USERNAME**.MDY**	Binary type melody section file that can be used without translation.
Sound Effect File	USERNAME**.SND**	Graphic type sound effect section file.
	USERNAME**.SDY**	Binary type sound effect section file that can be used without translation.
Speech File	USERNAME**.XXX***	PCM or EMC's VDS format speech section file.

* Filenames for speech files may end with any extensions except "**BIN**" or "**LAY**". Speech files, including "**RAW DATA**" format, "**WAV**" format, or EMC's **SCS (VDS)** format are supported. IDS Coding Program will be able to auto-detect speech files under these formats.

The smallest playback unit for EM57000 series is called a Section. Each Section can either be a Sound Effect File, a Melody File or a Speech File.

3.2.1 Data Update During "Save As" and "Close" Commands

IDS Coding Program offers an editing features (under Edit window) that can edit user's programs and melody files. It cannot, however, edit speech files. To edit speech files, use the Speech Coding System (SCS) software tool from EMC for *.SCS files. You may also use commercial tools, such as SoundForge®, CoolEdit, or GoldWave under Windows3.1/Windows95 to edit speech files.

When closing edit window with "Close" command, IDS will check whether edit window contents have been modified. If modified, IDS will prompt user to "Save" or "Abandon" contents before exiting.

Whenever user saves a file into the disk, IDS will auto-detect the type of the edited file and automatically perform translation (if file is ***.**MLD) or perform compilation (if file is ***.**ASM or ***.**ESY) before saving and closing the edit window. Thus, only the latest version of ***.**MLD, ***.**ASM, and ***.**ESY will exist and be maintained in the program. The generated compiled files (***.**LAY and ***.**BIN) are also saved in the disk.

The ***.**LAY file is used for commercial fabrication of EM57000 IC's, while ***.**BIN file is used for burning EPROM on Verification Module for test and demo purposes.

3.2.2 Conflicting Filenames

To avoid compiling problem, refrain from giving the same main filenames for two different files with different file extensions. IDS Coding Program have a tendency to get mixed up with such files.

For example, if a melody file is already assigned with filename **VOICE.**MLD, none of the speech files should be named **VOICE.**VDS. Otherwise IDS will have difficulty distinguishing between the two files.

3.3 Main Emulation Board Power Up

If the IDS Main Emulation Board is connected to the PC through the host printer port, to be sure external power (+5VDC) is provided. Power on to the Main Emulation Board before invoking the IDS Coding Program from the host computer.

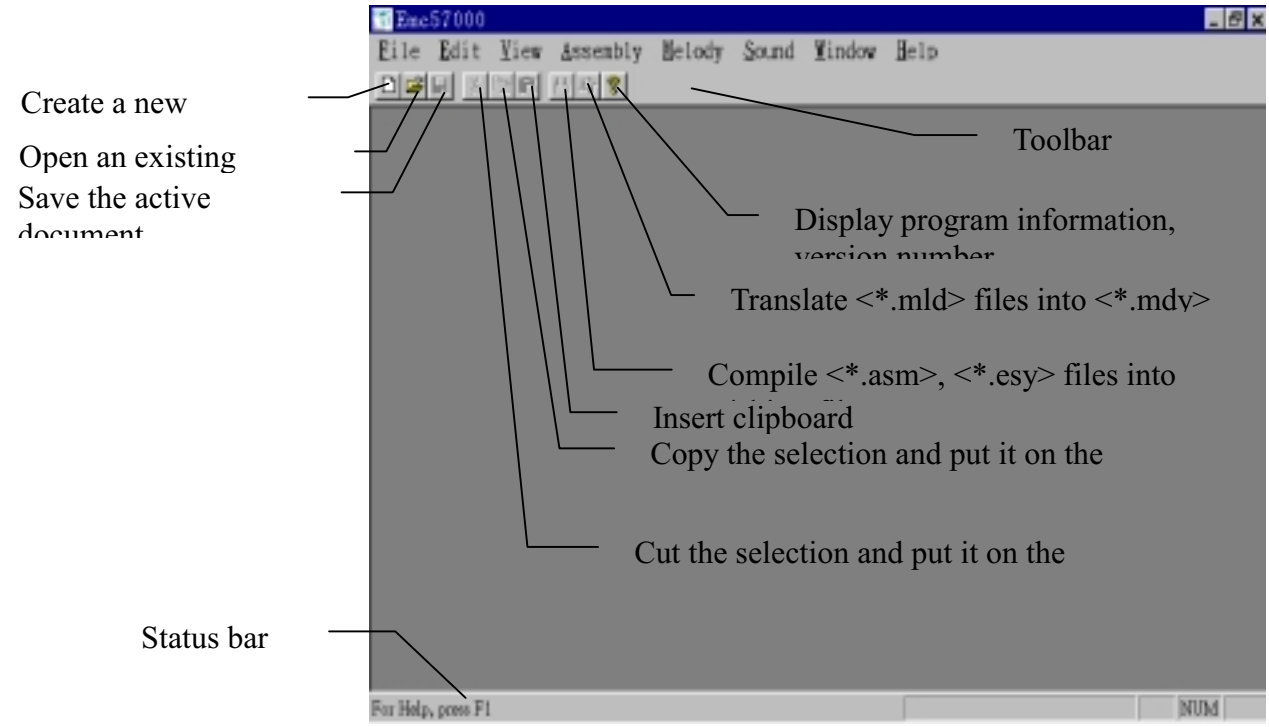
3.4 Invoking IDS Coding Program

To run the IDS Coding Program, click

EMC57000

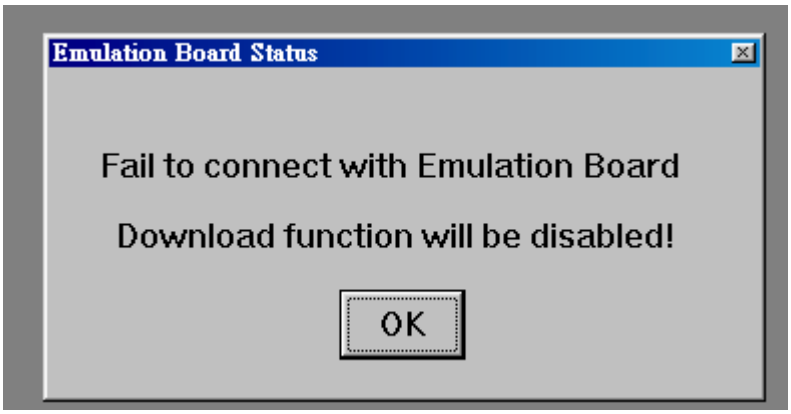
under **EMC** program group.

The IDS start-up window environment is as shown below.



IDS Start-up Window

At the beginning of the program operation, the IDS Coding Program will initialize the Emulation Board. If IDS failed to achieve normal communication with the Emulation Board, error message will be displayed as below.



Communication Failure Error Message

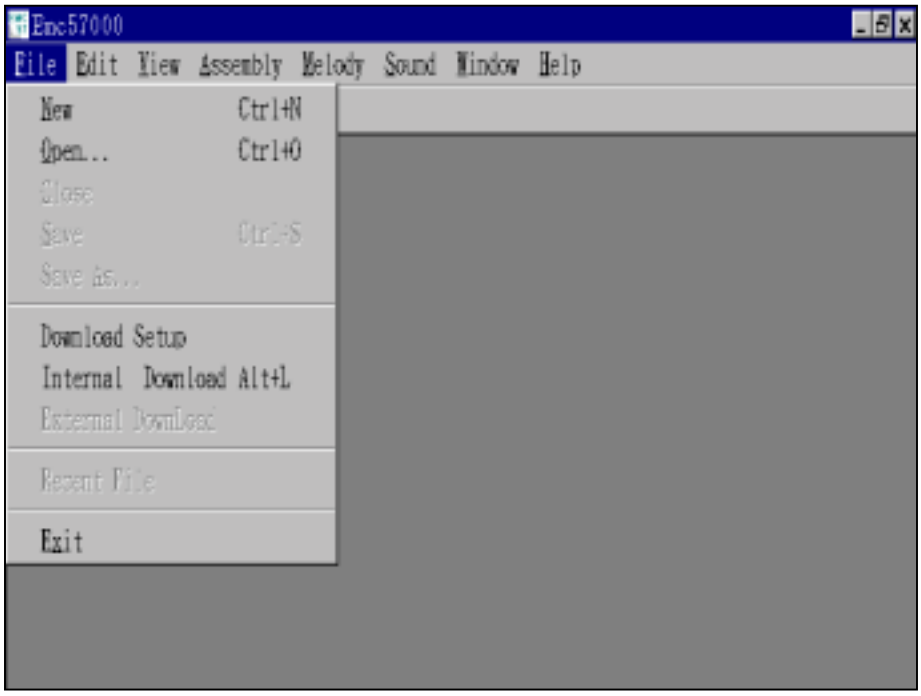
3.5 A Tour to Menus and Commands

The IDS Coding Program window has eight menu groups displayed on its Menu Bar. These are the-

- File Menu
- Edit Menu
- View Menu
- Assembly Menu
- Melody Menu
- Sound Menu
- Window Menu
- Help Menu

Invoking each of these menus displays the next level of drop-down menu, which further display lists of individual commands for user to choose and perform his desired applications. These menus and commands are further discussed in the following sections.

3.5.1 File Menu



File Menu Commands

New	Create a new "*.ASM", "*.ESY", "*.MLD", or "*.SND" document.
Open	Open an existing "*.ASM", "*.ESY", "*.MLD" or "*.SND" document.
Close	Close the active document. IDS will prompt user whether to save or abandon data in the working window.
Save	Save the active document in the current working window and compile updated document. If no error is found, the file is automatically downloaded <*.bin> file into the emulation board.
Save As	Save the active document with a new filename in the current working window and compile data. If no error is found, the file is automatically downloaded <*.bin> file into the emulation board.
Download Setup	Internal download and external download setup
Internal Download	Download <*.bin> file into emulation board.
External Download	Use external ROM emulator to download <*.bin> file.
Exit	Quit the application from IDS Coding Program.

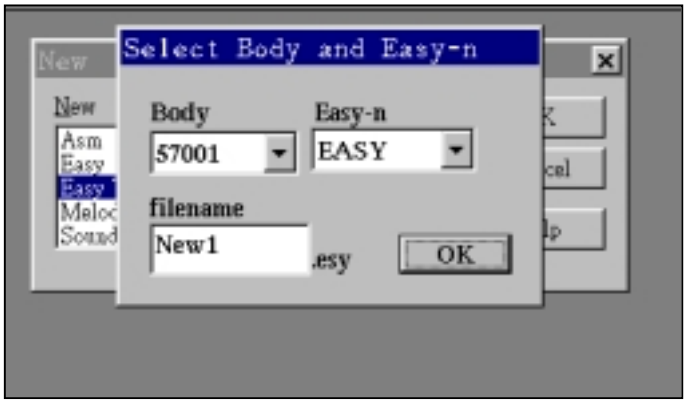
NOTE
Function keys shown on the right column of each command are the corresponding short-cut keys of the command.

3.5.1.1 New Menu

Selecting "New" menu command will display the following dialog box.

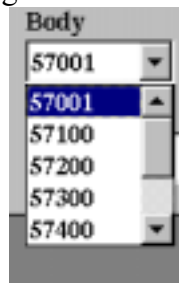
"New Process File" Dialog Box

To press the left key of a mouse selects a new "*.asm", "*.esy", "*.mld", or "*.snd" and "OK" button needs to be pressed. If you want to use Easy Format in EM57000 series, you can select the Easy Template. The window will display the following dialog box.



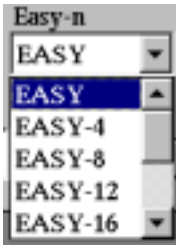
"Easy Template Process File" dialog Box

To select the "Body", press the ▼ in "Body" box and the window will display the Body dialog box as below. Click a body you want.



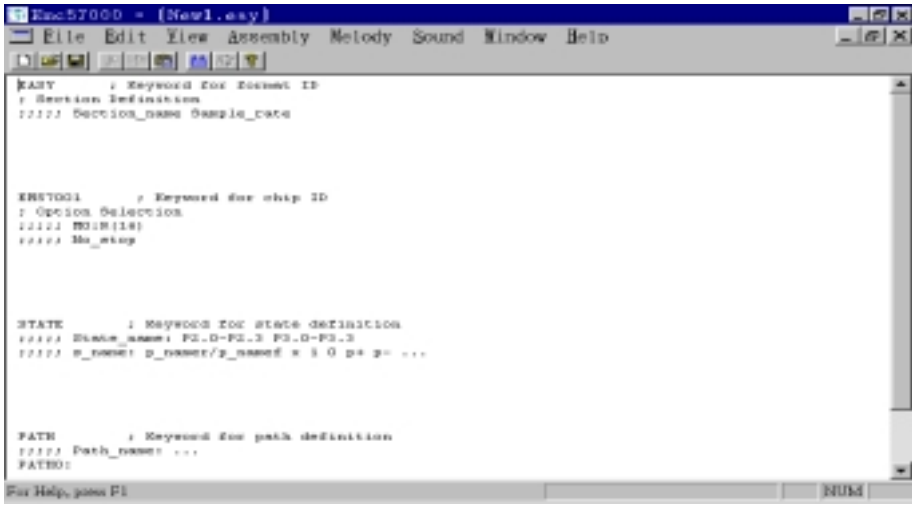
"Body" dialog box

To select the “Easy-n”, press the ▼ in “Easy-n” box and the window will display the Easy –n dialog box as below. Click an Easy Format you want.



“Easy-n” dialog box

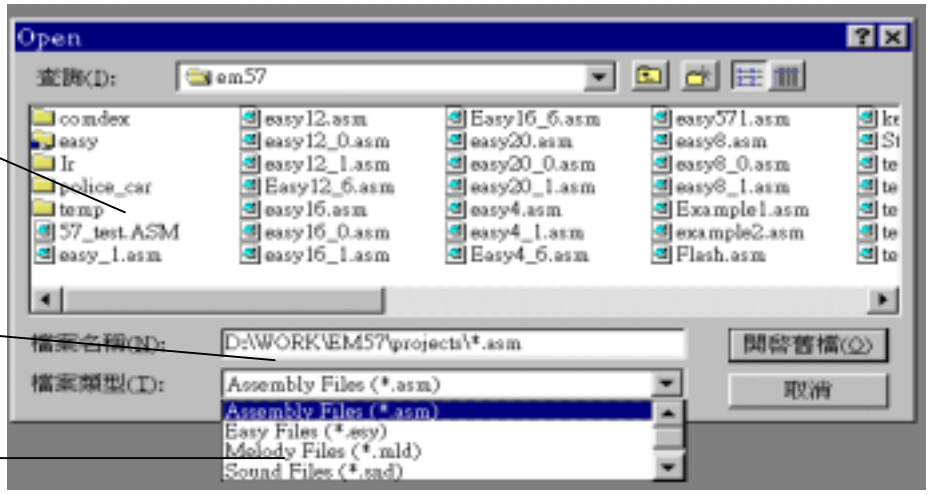
To name a file, simply key-in the filename. Otherwise the IDS will provide a “New1.esy”. If user selects the Body, Easy-n and filename, and “OK” button is pressed. The window will display the Easy-n program file format for EM57000 series chips. The following displays the “Easy–n program file format “ window. Refer to Chapter 7~ 9 for the details of Easy-n program file format for EM57000 series chips.



“Easy-n program file format ” window

3.5.1.2 Open Menu

Selecting “Open” menu command will display the following dialog box.



Double click from displayed filename

Key-in filename

Select type of filenames to be displayed on “File

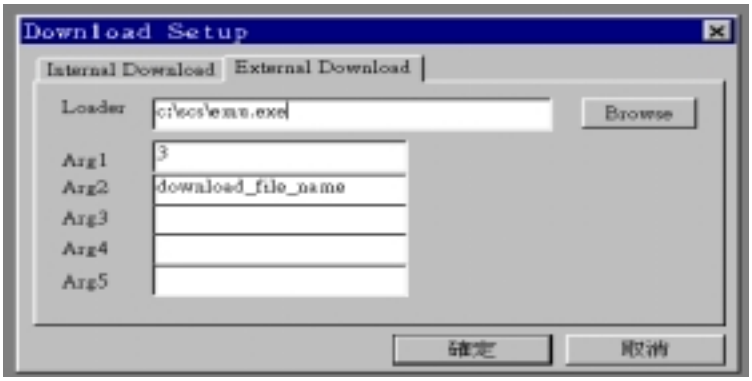
“Open file “ dialog box

To select a file, simply key-in the filename or its extension in the “File Name” field, or select one from the “File Type” box, then click the “Open” button. Or double click the filename listed in the “Files” box to open file.

If the selected file has the filename extension of “***.ASM**” or “***.ESY**”, the Assembly menu command becomes active and the Melody menu grayed. Likewise, if the extension is “***.MLD**”, the Melody menu command becomes active and the Assembly menu grayed. Otherwise the IDS will only provide editing function.

3.5.1.3 Download Setup

3.5.1.3.1 Internal Download Setup

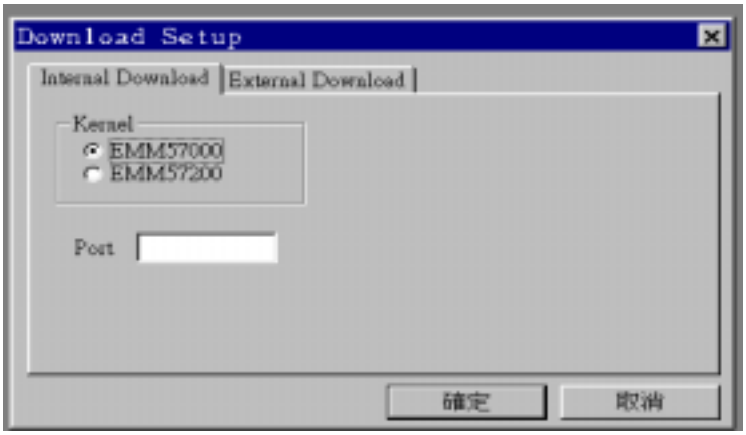


“ Internal Download setup” dialog box

To use the internal download, the IDS environment provides to select the “EMM57000” or “EMM57200” as the kernel chip.

If you use the old EMK by add-on card, you can change the desired I/O port used by add-on card by keying in the port. If you use the EMK v2.1, you don’t care the port

3.5.1.3.2 External Download Setup

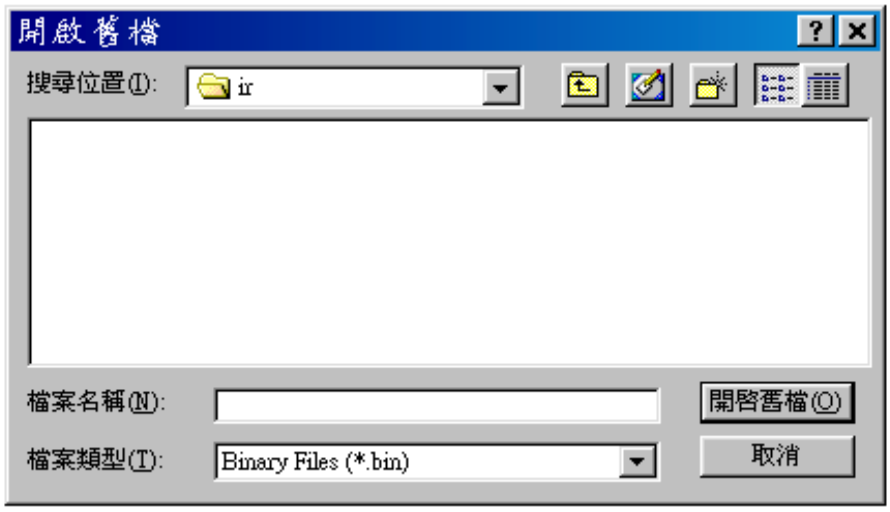


"External Download Setup" dialog box

This "External Download Setup" dialog box is used to enable IDS Coding Program to download the resulting binary file through a commercial (non-EMC) ROM emulator

The IDS Coding Program can be used with different ROM emulator from other vendor. The "External Download Setup" dialog box is used for this purpose. If necessary, user can also fill out the required parameters for the program in the arg1~arg5. After that, to click the "External Download" menu of "File", the IDS window displays the box of section 3.5.1.5.Please refer to section 3.5.1.5.

3.5.1.4 Internal download

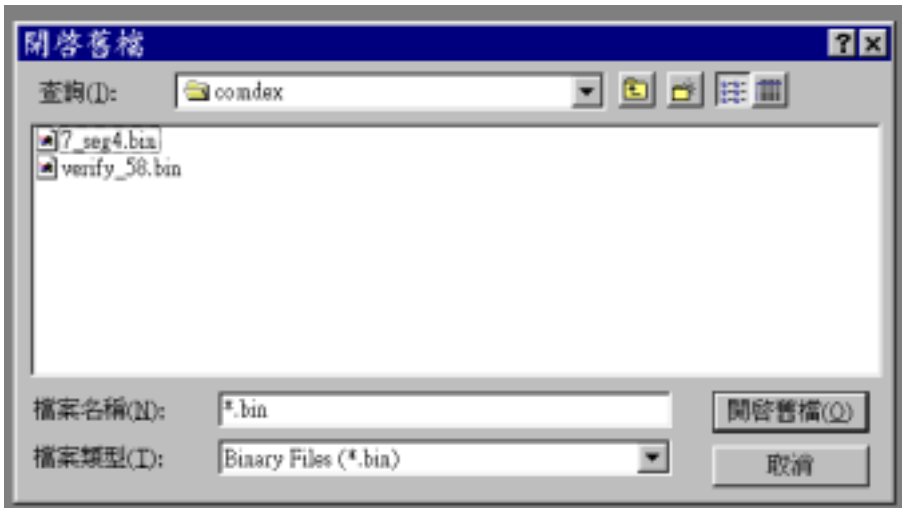


"Internal Download" dialog box

To click "Internal Download", the IDS window displays the dialog

box. The dialog box operation is the same with the section 3.5.1.2. When user selects the “*.bin “ file, the IDS will automatically download into the emulation board

3.5.1.5 External Download

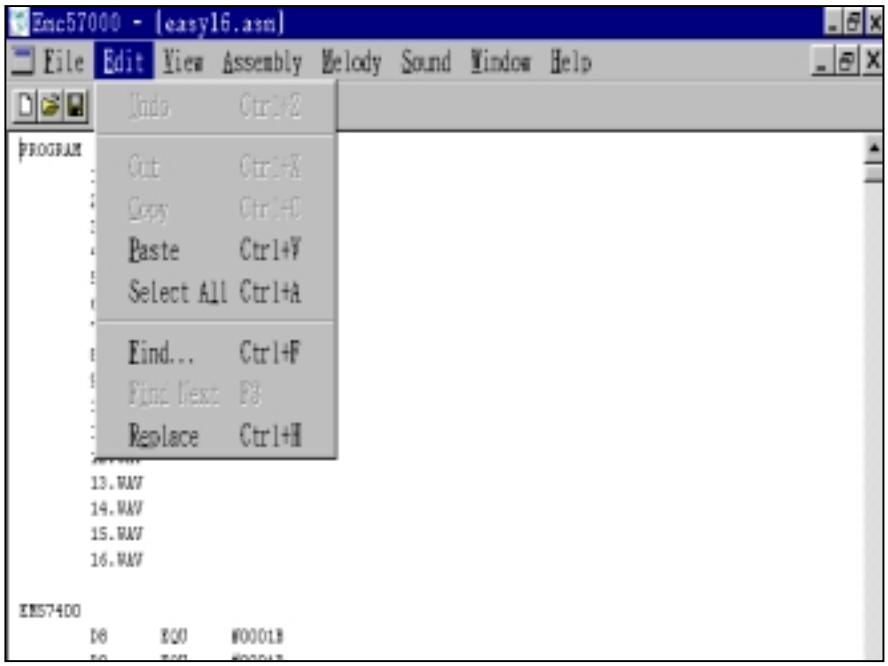


"External Download" dialog box

After filling out the required parameters for Section 3.5.1.3.2,click the "External Download" command of "File" menu. The IDS window displays the "External Download" dialog box. The dialog box is as shown above.

To select or key in the binary files, the IDS Coding Program will be able to download the resulting binary file using a commercial ROM emulator.

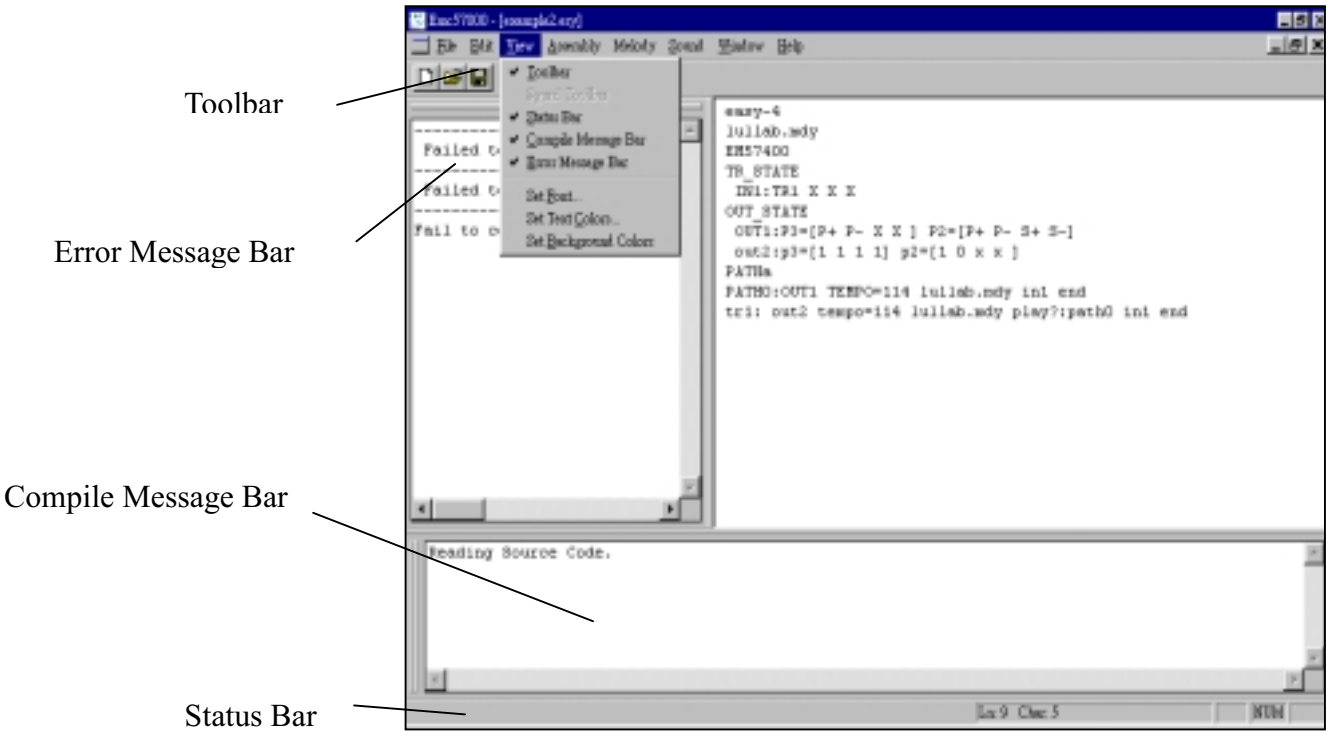
3.5.2 Edit Menu



"Edit Menu" Commands

Undo	Undo the last action command.
Cut	Cut the selection and put it on the Clipboard.
Copy	Copy the selection and put it on the Clipboard.
Paste	Insert Clipboard contents to cursor position in the edit window.
Select All	Select the enter document from edit window
Find	Find the specified text.
Find Next	Repeat the last find action.
Replace	Replace the specified text with different text.

3.5.3 View Menu



View Menu Command

Toolbar Open or hide the toolbar.

Sound Toolbar Open or hide the sound toolbar

Status Bar Open or hide the status bar and display the explanation of menu command.

Compile Message Bar Open or hide the compile message bar and display the result of compiling the file.

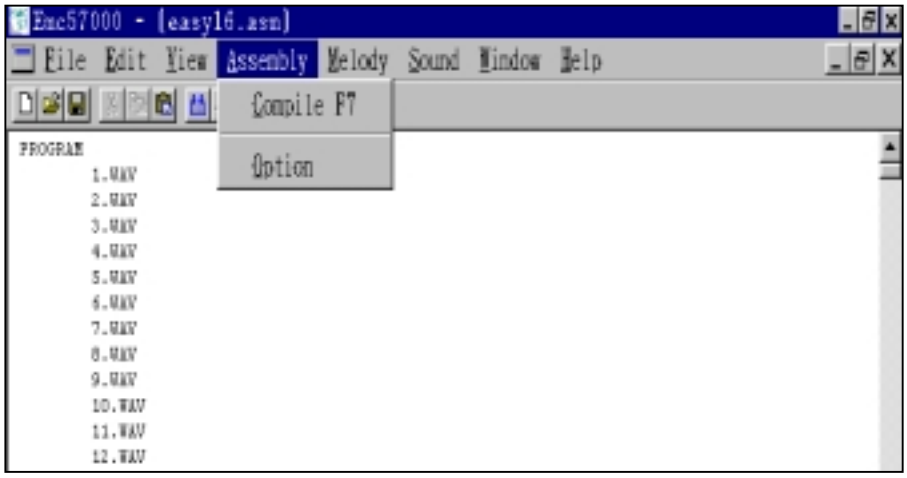
Error Message Bar Open or hide the error message bar and display the error result of compiling the file.

Set Font Set font for this window.

Set Text Color Set text color

Set Background Color Set text background color

3.5.4 Assembly Menu



Assembly Menu Command

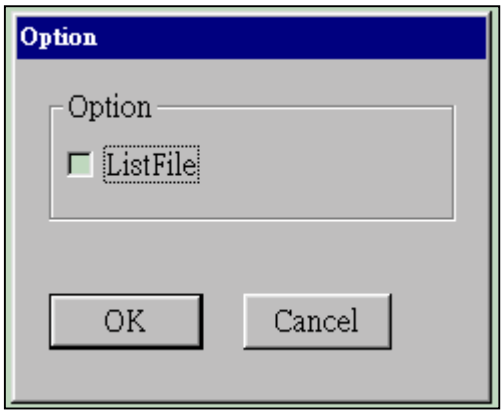
Compile Compile <*.esy>,<*.asm>files into <*.bin>files

Option Enable/disable the List file

When "*.ASM" or "*.ESY" file is opened, Assembly menu is activated and the Melody menu is grayed. When IDS is compiling, a message bar pops up to display the compilation results Refer to the section 3.5.3

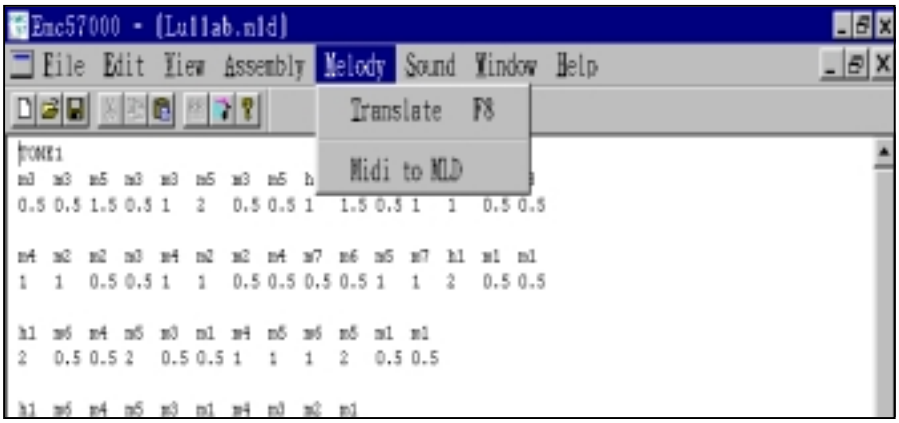
The option command displays a menu enabling/disabling generation of the Listfile during program development.

Every time user selects the "Compile" command, IDS Coding Program will automatically compile the file and doesn't generate corresponding LIST files. If user wishes to generate the LIST files during program development, he can simply click the "ListFile"of this option to generate "√" and LIST file will be generated when "COMPILE" command is invoked. The following displays the "option "dialog box



"Option" dialog box

3.5.5 Melody Menu



Melody Menu Command

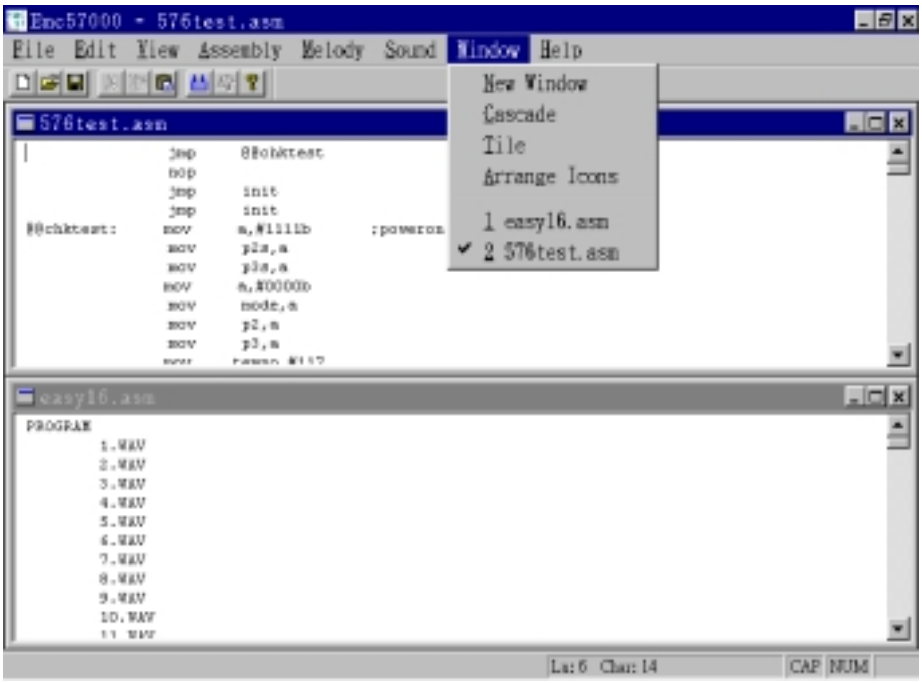
Transfer Transfer <*.mld>files into <*.mdy>files

Midi to MLD Convert MIDI file(*MID) into melody (*.MLD/*.MDY) format

When "*.MLD" file is opened, Melody menu is activated and the Assembly menu is grayed.

When both Assembly and Melody working windows are opened and displayed on the screen, only one of the menus can be active at a time. That is, if the cursor is on Assembly (working) window, the displayed Melody window becomes inactive (grayed).

3.5.6 Windows Menu



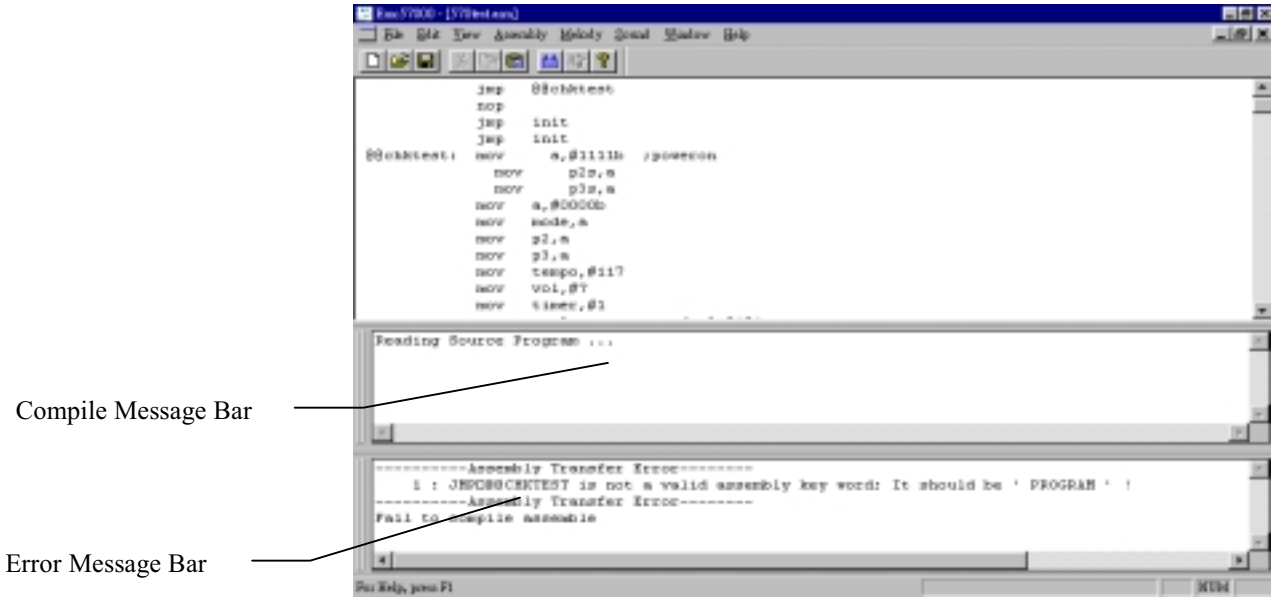
Windows Menu Commands

New Window	Open another window for the active document.
Cascade	Cascade all opened windows.
Tile	Tile all opened windows.
Arrange Icons	Arrange icons at the bottom of window

3.5.6.1 Error Message Bar and Compile Message Bar Window

During processing of user‘s source files (translating, compiling, downloading, etc.), IDS will display various messages, e.g., file status, error messages, etc. in a pop-up Message window. Cause of error is also explained in a separate viewport at the bottom of the working window as shown in the illustration below.

IDS Coding Program allows user to modify and correct error on-line while the error message is on display. The user can modify the size and position of error message bar and compile message bar.



A typical Message Bar Window

Chapter 4

Processing MLD Program

4.1 Introduction

The development of a melody file for EM57000 series chips on IDS emulator with the IDS Coding Program is contingent to user's valid "MLD" text file. Valid "MLD" format are discussed in Section 4.2 of this chapter.

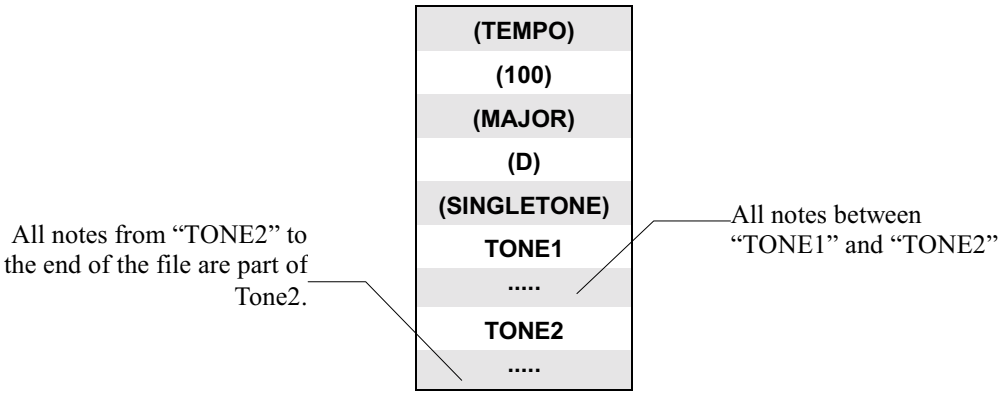
Valid "MLD" file is downloaded into the IDS Main Emulator Board and Verification Module (using Translate command under Melody menu) for performance test, modifications, and its eventual development into "MDY" file. Translation process is discussed in Section 4.4 of this chapter.

4.1.1 "MLD" Text File Editor

The user's "MLD" file is retrieved from its directory directly into IDS Coding Program editor where further editing of the file may be done if required. Refer to Section 3.5 of Chapter 3 for the many features of the IDS Coding Program editor.

If user is already quite familiar with "MLD" text file structure or his "MLD" text file needs no further editing, he may skip the following sections and go directly to procedure on loading "MLD" file into IDS Main Emulator Board and Verification Module as explained in Section 4.4.

4.2 “MLD” Text File Format



A Typical Melody Text File Data Lines Arrangement and Formation

The text lines in an “MLD” text file consists mainly of data lines:

- The data lines comprise of “keyword” lines, and “tone codes” lines as illustrated above.
- The “keyword” lines, which are always located on top of the text file, are optional and may consist of “tempo”, “major”, or “singletone” lines or all of the three.
- The “tone codes” lines are the main core of the melody text file as it contains the notes and beats data of the file.
- Comment lines or strings (prefixed with a semi-colon (**;**) comment identifier) may be inserted between any of the data line or at the end of “tone codes.” All text after a comment identifier is ignored during melody translation processes.

4.2.1 TEMPO Keyword

When used, “TEMPO” keyword is followed by a TEMPO value. This keyword is used in the “MLD” text file for user’s reference only.

4.2.2 MAJOR Keyword

“MAJOR” keyword shifts all notes in a melody into a higher or lower notes level. When used, the keyword is followed by a notes level as one of the following**:**

D- E- F- G- A- B- C D E F G A B

If none is specified, the “C” will default as the “MAJOR” notes level.

4.2.3 SINGLETONE Keyword

"SINGLETONE" is an optional keyword that is inserted before keyword "TONE1" (explained in the Section 4.2.4 below). It indicates that the "MLD" text file contains a single "TONE1" channel only. It has no "TONE2" portion. See Example 1 in Section 4.3.

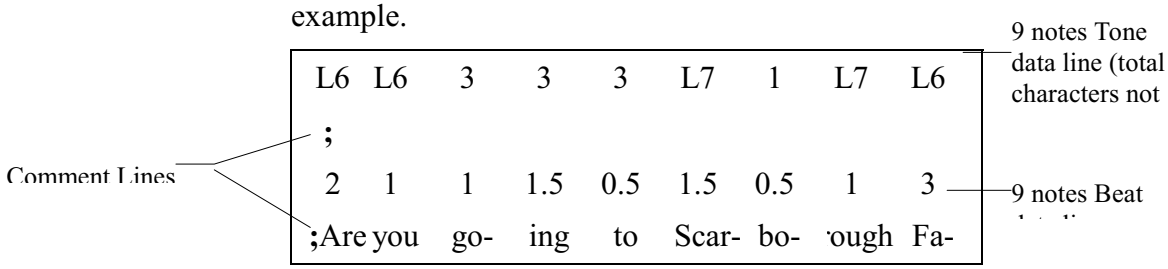
If the "MLD" file contains both "TONE1" and "TONE2" channels, the keyword is not required. This is illustrated in Example 2 of Section 4.3. Under this configuration, the total beats length (i.e., the summation of all beat values) in Tone2 must be equal to that of Tone1.

4.2.4 TONE1/TONE2 Keyword

"TONE1" keyword marks the start of melody notes data. All data between "TONE1" and "TONE2" keywords are the Tone1 data.

All data from "TONE2" keyword to the end of the file are the Tone2 melody notes data.

Each melody note is defined by a Tone and its corresponding Beat data. The Tone data line is on top line and the Beat data line is right under it unless separated by a comment line as shown in the following example.



An Example of a Pair of Melody Note Line

Tone Data Line

Each Tone data is always paired with a Beat data. IDS Coding Program supports a total of 37 tones (at 4 octave levels range including half tones). Their respective codes for each octave level are displayed in the table below.

Octave Levels	Octave Note Codes for Tone Data Line Use											
2												S7
3	L1	#L1	L2	#L2	L3	L4	#L4	L5	#L5	L6	#L6	L7
4	1	#1	2	#2	3	4	#4	5	#5	6	#6	7
5	M1	#M1	M2	#M2	M3	M4	#M4	M5	#M5	M6	#M6	M7
6	H1	#H1	H2									

"0" (zero) is used as rest note code.

The number of notes in a Tone data line should not exceed 255 characters (including "0" and "space" characters that separate each note).

Beat Data Line

The number of beats in a Beat data line must be equal to the number of notes of its corresponding Tone data line.

IDS Coding Program supports the following musical beats and users can use them directly as beat codes.

Beat	0.25	0.5	0.75	1	1.5	2	3	4

4.3 Examples of "MLD" Text File

Example 1. Shown below is a sample "MLD" text file for a two data lines single channel melody using default tempo and major. Note that the blank line separating the first and second section is optional and does not affect the file operation. It is used only to easily distinguish sectional grouping of the data lines.

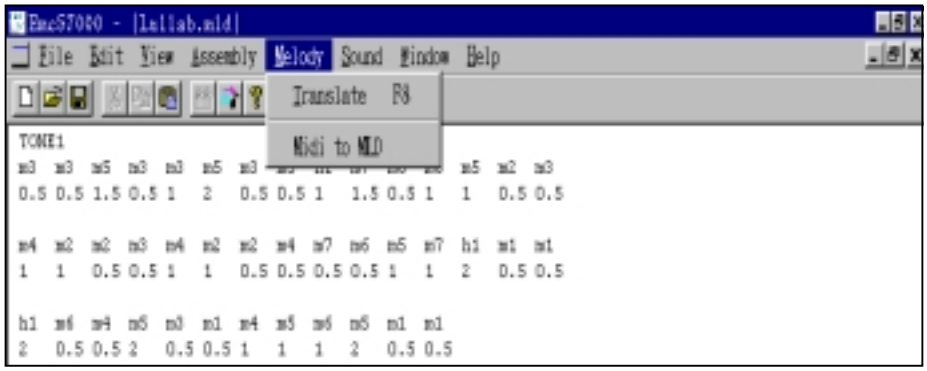
```
; First Part of Scarborough Fair (Waltz, 3/4)
; Only one tone part is used in the melody
SINGLETONE
TONE1
L6    L6    3     3     3     L7    1     L7    L6
2     1     1     1.5   0.5   1.5   0.5   1     3
;Are   you   go-   ing   to    Scar- bo-   rough Fa-

6     0     3     5     6     5     3     #4    2
3     1     1     1     2     1     1     1     1
;ir         Pars- ley   Sad   Ro-   se    Mary  and
```

Example 2. This example shows a multiple data lines "MLD" text file with two "TONE1/2" channels melody using default tempo and major. No sectional spacing is used on this example.

```
; Second Part of Scarborough Fair (Waltz, 3/4)
TONE1
 3          3          3          3     0     6
 3          3          3          1     1     1
 ;ie   .  .  .  .  .  .  .  .  .  .  Re-
 6     6    5     3    3    2    1    L7
 2     1    2     1    1    1    1    3
 ;mem- .  ber  me   .  to   one  who  lives  there  .  .
TONE2
 L3   #5   h3   #5   7    #5   L3   #5   m3   #5   7    #5
 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
 ;ie  .    .    .    .    .    .    .    .    .    .    .
 L3   #5   m3   #5   7    #5   L3   #5   m3   #5   7    #5
 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
 ;    .    .    .    .    .    .    .    .    .    Re-  .
 L6   6    m3   6    m1   6    1    5    m3   5    m1   5
 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
 ;Am                       C
 ;mem- .    .    .    ber  .    me   .    .    .    to   .
 1    5    m3   5    m1   5    L5   5    m5   5    7    5
 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
 ;C                        G
 ;one  .    who  .    lives .    there .    .    .    .    .
```

4.4 Melody Verification and Saving “MLD” File

With the “MLD” melody file on screen and ready for testing, click on Melody menu and execute the Translate command (or press “F8” key).



Melody Menu Showing the Translation Command

If no error is found, the melody file is downloaded into the emulation board. From the Verification Module keypad, press any key to test-play the resulting melody.

If the resulting melody is unsatisfactory, re-edit the “MLD” file then redo test-play processes. Repeat the cycle until the desired melody is achieved.

To save the “MLD” file (and generate the “MDY” file at the same time) follow these steps:

1. Click to open File menu.
2. Choose Save (or press “F2” key) or Save as (or press “F4” key) command.
3. Aside from saving an “MLD” file, IDS Coding Program will automatically create and save a corresponding “MDY” file too. Both files will have the same main filename and are located in the same directory.

Chapter 5

MIDI to Melody File Conversion

5.1 Introduction

The IDS package includes a MIDI Translator program that can simultaneously convert a user's MIDI file format into both text type melody section file (*.MLD) format and binary type melody section file (*.MDY) format. The IDS emulation board requires and accepts *.MLD or *.MDY files only.

5.2 Applicable Conversion Format

5.2.1 MIDI File Format Requirements

User should see it that his MIDI file meets the following conditions for accurate conversion.

1. The MIDI file must have the file extension of *.MID.
2. MIDI file format should conform to "Format 0" or "Format 1" of the General MIDI (GM) 1.0 Specifications as published by the MIDI Manufacturers Association (MMA). Both formats are accessible by MS Windows.

   Compatibility for "Format 2" (which can not be accessed by MS Windows) and that of other specification, (e.g., GS), are not supported with the current version of the IDS Coding Program.

5.2.2 Format 0 Structure

Format 0 of MIDI files has a single multi-channel track. The MIDI Translator will separate the MIDI notes events depending on MIDI channel information. The note events for the first channel (not necessarily for Channel "0") of the MIDI file are consigned to Channel 1 of the melody file. Likewise, the note events for the second MIDI channel (not necessarily for Channel "1") are consigned to Channel 2.

5.2.3 Format 1 Structure

User may set the melody file format to be generated to either Single Tone Mode (only Channel 1 is utilized) or Dual Tone Mode (utilizing both Channels 1 and 2).

Format 1 of MIDI files accepts multi-track placement in the same file. This allows the MIDI Translator to collect MIDI notes into "channels of melody file**"** depending on the tracks status, i.e.,

- If there is only a single "note-track" (track containing "note on" and "note off" MIDI events), such MIDI notes will be translated into single-tone melody and all the note events in the track are consigned to Channel 1 of the MLD file.
- If the MIDI file contains more than one "note-tracks", all note events in the second note-track are consigned to Channel 2 (consequently dual-tone is created). Any other note events in other "note-tracks" are ignored.

5.2.4 Polyphony Format

Polyphony (chord) is not supported. That is, at any given MIDI delta-time, each track can only have one single pair of "note on" and "note off" event.

5.2.5 SMPTE Format

SMPTE format is not supported. Use MTC format instead

5.2.6 PPQ Note

PPQ (Pulses Per Quarter) note should be set with one of following numbers**:**

"96", "120", "168", "192", "240", "360", "384", "480"

"240" is popularly used number and is recommended.

5.3 How the MIDI Translator Program Works

The following are the MIDI notes subset that are supported and should be used in the user's MIDI file.

Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
2	-	-	-	-	-	-	-	-	-	-	-	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86-	-	-	-	-	-	-	-	-	-

MIDI Notes Subset

The following shows how the above becomes melody subset after conversion by MIDI Translator program.

Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
2	-	-	-	-	-	-	-	-	-	-	-	S7
3	L1	#L1	L2	#L2	L3	L4	#L4	L5	#L5	L6	#L6	L7
4	1	#1	2	#2	3	4	#4	5	#5	6	#6	7
5	M1	#M1	M2	#M2	M3	M4	#M4	M5	#M5	M6	#M6	M7
6	H1	#H1	H2	-	-	-	-	-	-	-	-	-

Melody (*.MLD or *.MDY) Notes Subset

Example: **"**C3" (48) of the MIDI subset is translated into "L1" in the melody subset. "C6" (84) becomes "H1", Middle C ("C4") becomes "1", etc.

As shown above, there are only 37 notes that can be defined in the MIDI file. However, if the defined notes exceed the specified quantity, but their dynamic range (the difference between the highest and lowest note value) is less than 37, the MIDI Translator will auto-shift all notes to fit the melody specification.

Example: With the notes of the MIDI file varies from "C4" to "D6", and "D6" exceeds the specification for *.MLD format, all notes will shift 2 steps (semitone) to the left. Finally, "D6" is translated into "H1" and "C4" translated into "#L6." The MIDI Translator provides a scrollbar where user can manually shift the notes level up and down to fine-tune for best sound effect (see Section 5.4 of this chapter).

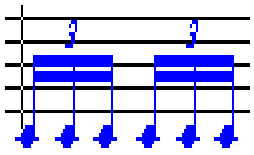
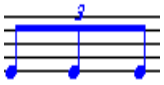
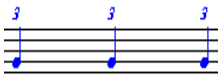
To prevent translation error in measuring beat length for MIDI note, user should quantize all MIDI notes with user's MIDI software to accomplish precision timing in translation.

5.3.1 MIDI Files with Triplets

For songs that contain triplets, the MIDI Translator provides options for MIDI file in triplet or quarter note base translations.

- Under quarter note base translation, normal notes (16th /8th /4th /half /whole notes) will retain its original beat length after translation. The triplets will be partitioned into 0.25 beat or 0.5 beat in accordance with their order of appearance.
- Under triplet base translation, the triplets are always partitioned into 0.5 beat while the normal notes are scaled into their related beat length.

The following table shows the resulting notes and beats of a number of sample MIDI notes.

Item	Pre-Translation Notes	Notes After Translation	
		Quarter Beat Base	Triplet Beat Base
1		*(Note)* 1 1 1 *(Beat)* 0.25 0.5 0.25	*(Note)* 1 1 1 *(Beat)* 0.25 0.25 0.25
2		*(Note)* 3 3 3 *(Beat)* 0.25 0.5 0.25	*(Note)* 3 3 3 *(Beat)* 0.5 0.5 0.5
3		*(Note)* 5 5 5 *(Beat)* 0.5 1.0 0.5	*(Note)* 5 5 5 *(Beat)* 1.0 1.0 1.0
4		*(Note)* 1 *(Beat)* 0.25	*(Note)* 1 *(Beat)* 0.25*
5		*(Note)* 3 *(Beat)* 0.5	*(Note)* 3 *(Beat)* 0.75
6		*(Note)* M1 *(Beat)* 0.75	*(Note)* M1 *(Beat)* 1*
7		*(Note)* 5 *(Beat)* 1	*(Note)* 5 *(Beat)* 1.5

8		*(Note)* M1 *(Beat)* 1.5	*(Note)* M1 0 *(Beat)* 2 0.25
9		*(Note)* 1 *(Beat)* 2	*(Note)* 1 *(Beat)* 3
10		*(Note)* 1 *(Beat)* 3	*(Note)* 1 0 *(Beat)* 4 0.5
11		*(Note)* 1 *(Beat)* 4	*(Note)* 1 0 *(Beat)* 4 2

IMPORTANT NOTES

When the MIDI file uses quarter note as beat base, triplets are shortened or stretched, but the resulting beat length remains equal to the total of the original beat length.

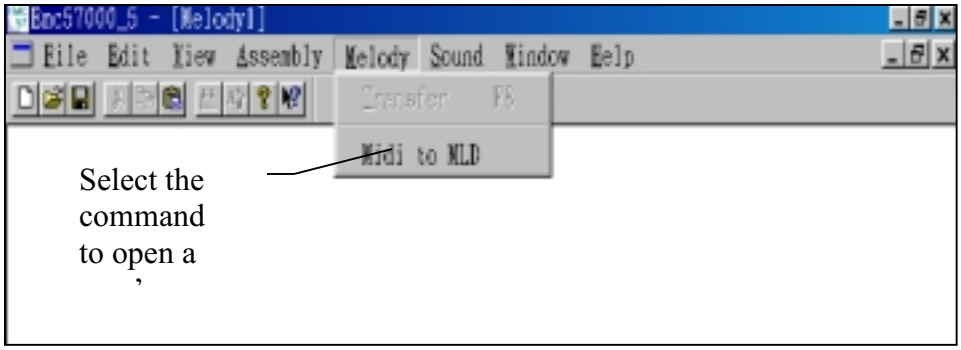
When using triplet as beat base, the resulting beat length for each note will be 1.5 (3/2) times longer than the original beat length. If the IDS Translator Program cannot find a perfect full beat for the resulting beat length, a rest note is automatically inserted to make up for the beat length difference. These are illustrated in Items 8 (dotted quarter note), Item 10 (dotted half note), and Item 11 (whole note) in the above table.

However, for the 16th notes and dotted 8th notes (see Items 4 and 6 of the table) events, inserting rest notes cannot help to make up for the resulting beat difference. The leftover beat length are accumulated until it is long enough to be implemented; e.g., when a 16th note occurs at the second time, the translator can now insert a rest note of 0.25 beat length at the end of the 16th note.

5.4 Converting a MIDI File into Melody Files

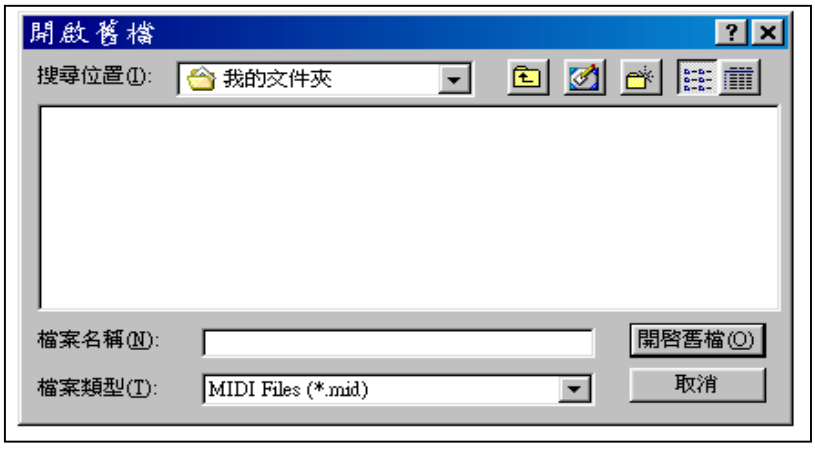
With the IDS Coding Program running, observe the following steps for converting user's MIDI file into melody (*.MLD/*.MDY) format files.

1. Select the "Midi to MLD" command from the "Melody" menu as illustrated below.



Melody Menu

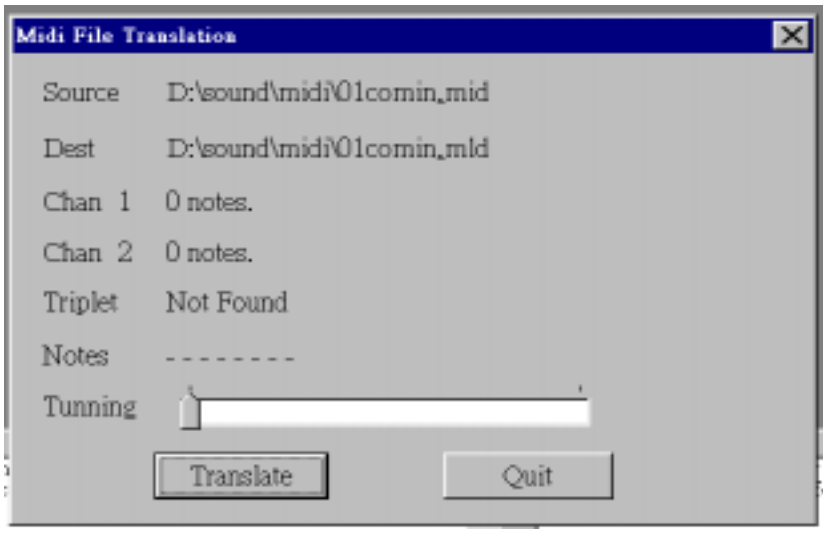
2. The “Open Midi File” dialog box will display as shown below. Select or key-in the appropriate directory where the source (*.MID) files are located.
3. Choose the file to be converted and press the “Open” button.



“Open Midi File” Dialog Box

4. The “Midi File Translation” dialog box (see sample below) will then display. If the first time conversion is performed, no conversion data will appear on the dialog box.

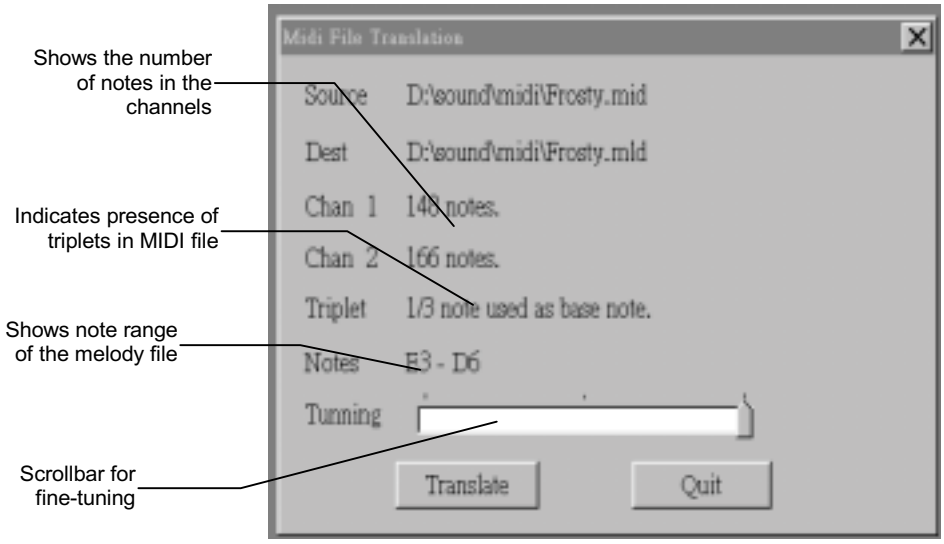
   Press “**Translate**” button to start conversion or press “**Quit**” to abort.



”Midi File Translation” Dialog Box

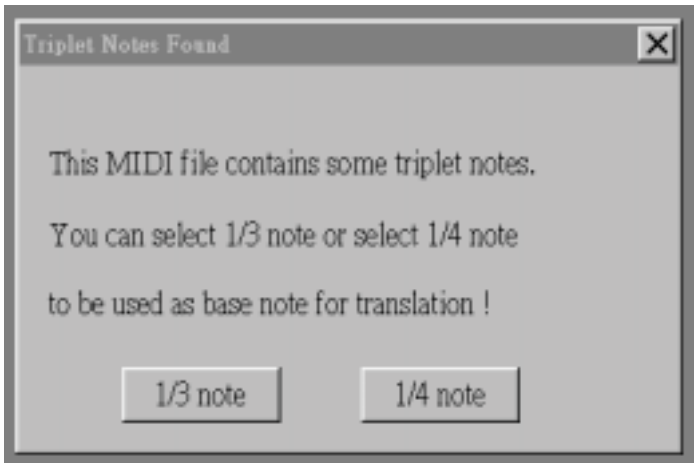
Note that IDS Coding Program will automatically save the converted melody files into the same directory where user's source files are located. It will use the "*.MLD" and *.MDY" extensions while retaining the same base filename of the source file.

5. After clicking the "Translate" button, conversion process starts. If the file contains triplets, another dialog box will appear (see figure at the bottom of the page) to select triplet or quarter note option. If no error is found, the melody files "*.MLD" and "*.MDY" files are generated into the source directory. The dialog box is then filled with the data and information of the converted melody files.



"Midi File Translation" Dialog Box After Conversion

6. If the notes used in the MIDI file do not swing from the lowest note to the highest note, user may use the scrollbar to fine-tune the note's range when necessary and repeat the conversion process.
7. If the MIDI file contains triplets as in the case of the above example, another dialog box (see below) will appear to let user select the translation base. For songs like "SLOW ROCK", selecting triplet base will be a good idea to get better result. Otherwise choose quarter note as translation base.



Translation Base Selection Dialog Box

Chapter 6

Processing SND Program

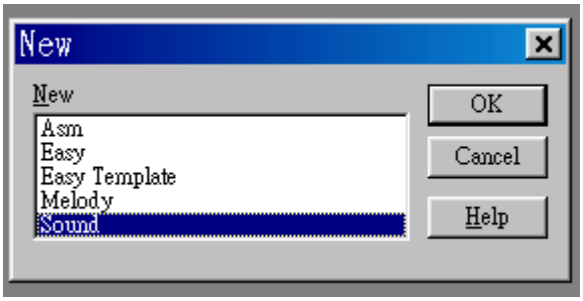
6.1 Introduction

The IDS package includes a Sound Effect Development System that can simultaneously convert a user's speech file ("**WAV**" format) into both graphic type sound effect file (*.SND) format and binary type sound effect file (*.SDY) format.

6.2 Invoking Sound Effect Development System

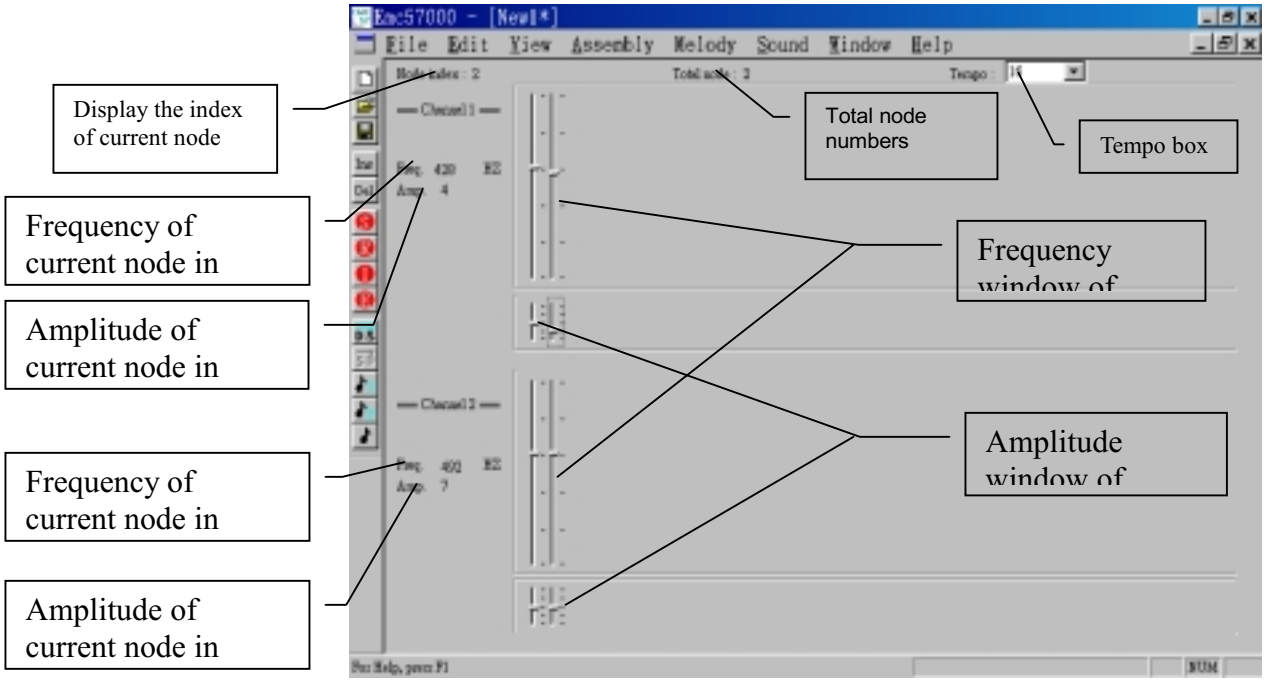
6.2.1 Open a new sound effect file

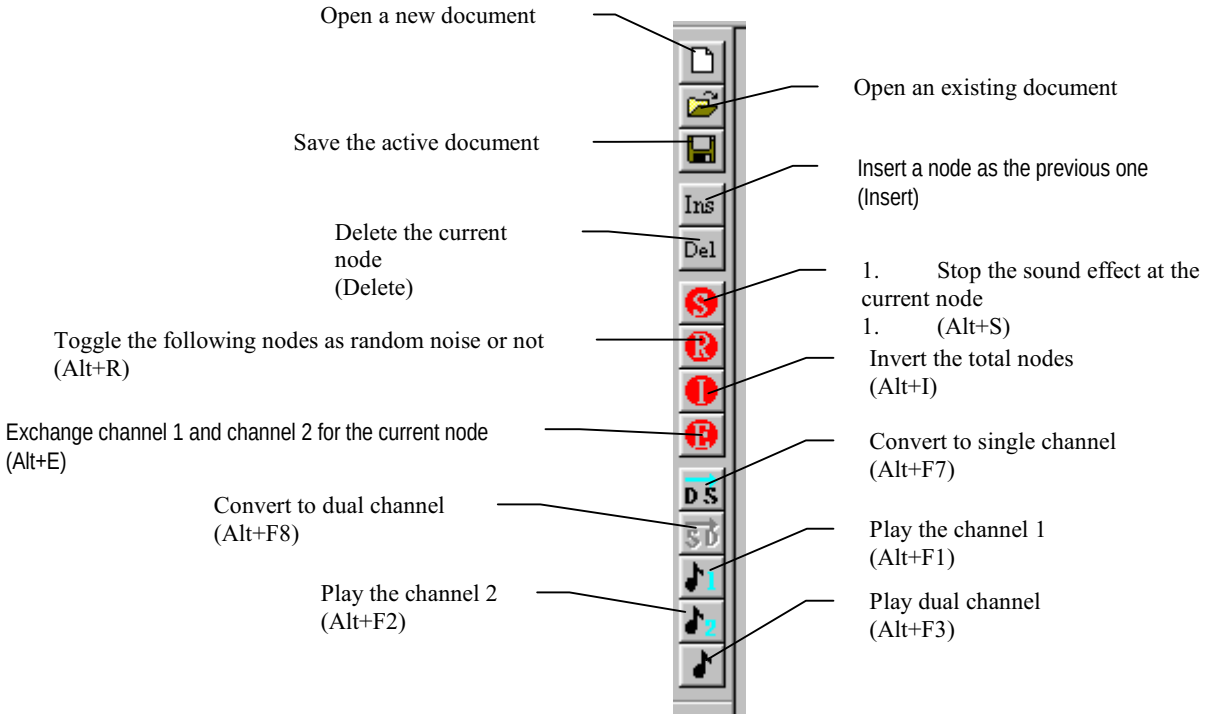
To create a new sound effect file, click the "New" menu of "File" in the IDS. The window displays the "New" dialog box.



"New" dialog box

After that, select the "sound" menu and click the "OK" button. The Sound Effect Development System environments will show as below.





Sound Effect Development System Window
(the key in "( )" is the shortcut key)

6.2.1.1 Frequency window:

There are 318 kinds of frequency are provided for channel 1 and channel 2. And the random noise is only provided for channel 1

> ***Note***
> *The random noise is only permitted in channel 1.*

6.2.1.2 Amplitude window:

Sixteen kinds of amplitude (0-15) are provided for channel 1 and channel 2.

6.2.1.3 Tempo box:

Sixteen kinds of tempo (ms/node) are provided as below.

2ms	4ms	6ms	8ms	10ms	12ms	14ms	16ms
18ms	20ms	22ms	24ms	26ms	28ms	30ms	32ms

When using EM57000 series, before playing a sound effect section file, the programmer must set the tempo value of the sound effect section file.

> ***NOTE***
> *Because of the sensitivity in frequency and amplitude window, it is recommended to select the node by pressing the mouse on the top or bottom of node.*

6.2.2 Shortcut Keys

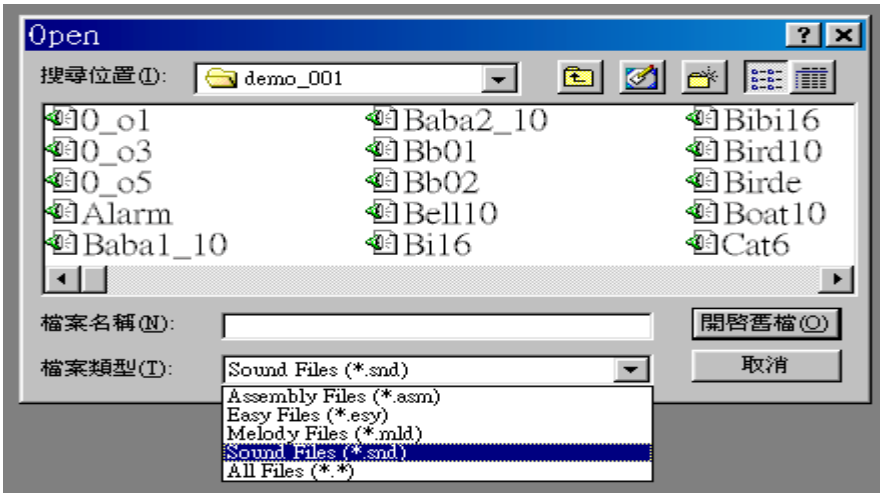
Except for using the mouse, the Sound Effect Development System environment provides several shortcut keys for more efficient application. These shortcut keys are as follows:

Shortcut Key	Function
<Home>	Increase the value of Freq. or Amp. to the top.
<End>	Decrease the value of Freq. or Amp. to the bottom.
<Page Up>	Quickly increase the value of Freq. or Amp.
<Page Down>	Quickly decrease the value of Freq. or Amp.
<Up arrow>	Slowly increase the value of Freq. or Amp.
<Down arrow>	Slowly decrease the value of Freq. or Amp.
<Alt+ Up arrow>	Switch the windows of Freq. and Amp.
<Alt+ Down arrow>	Switch the windows of Freq. and Amp.
<Alt+ Right arrow>	Increase the node index by moving the scroll bar.
<Alt+ Left arrow>	Decrease the node index by moving the scroll bar.
<Right arrow>	Increase the node index by moving the cursor bar.

<Left arrow>	Decrease the node index by moving the cursor bar.

6.2.3 Open an existing sound effect file

To open an existing sound effect file, click the "open" menu of "File" in the IDS. The window displays the "open" dialog box as below.

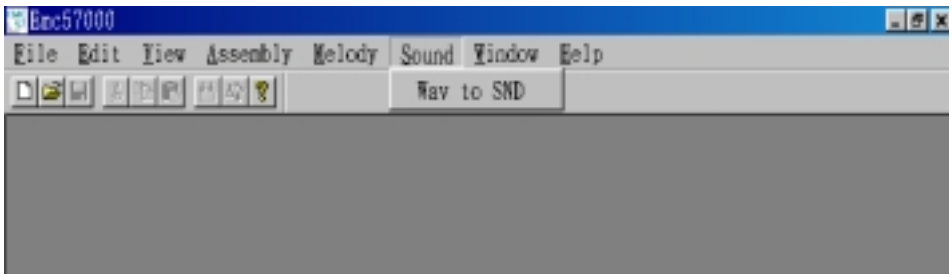


"Open file" dialog box

The operation of "open file" dialog box refers to section 3.5.1.2. Double clicking a file or keying in the filename. After that, the Sound Effect Development System environment will be shown as that in the section 6.2.1. Please refer to the section 6.2.1

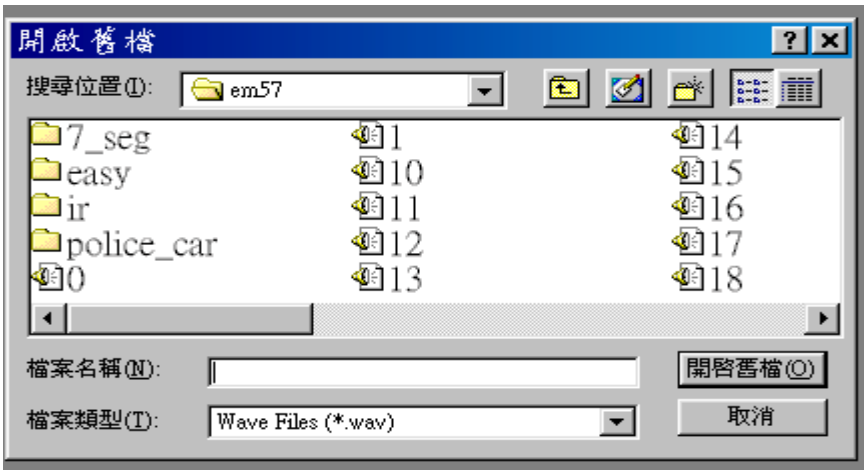
6.3 Convert the speech file to sound effect file

To convert the speech file to the "*.SND" file in the Sound Effect Development System, click the "sound" button and then click the " Wav to SND " button from menu command in IDS Coding Program. The window is shown as below.



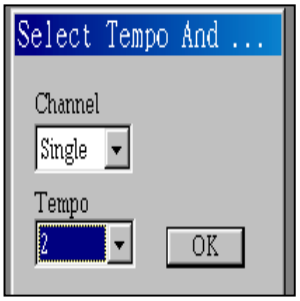
Sound Menu Command in IDS Coding Program

After that, the IDS Development System will display the “open file” dialog box. It is shown as below.



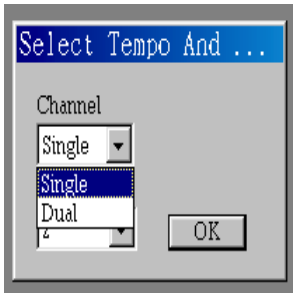
“Open file” Dialog Box of Sound Effect System

The operation of “Open file” dialog box refers to section 3.5.1.2. After selecting the file, the window will display the following dialog box. Before coding, you must correctly set the option of the tempo and channel.



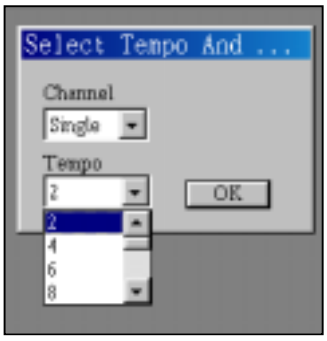
Channel and Tempo Option

To select the “Channel”, simply press the ▼ of “Channel” box and the window will display the “Channel” dialog box.



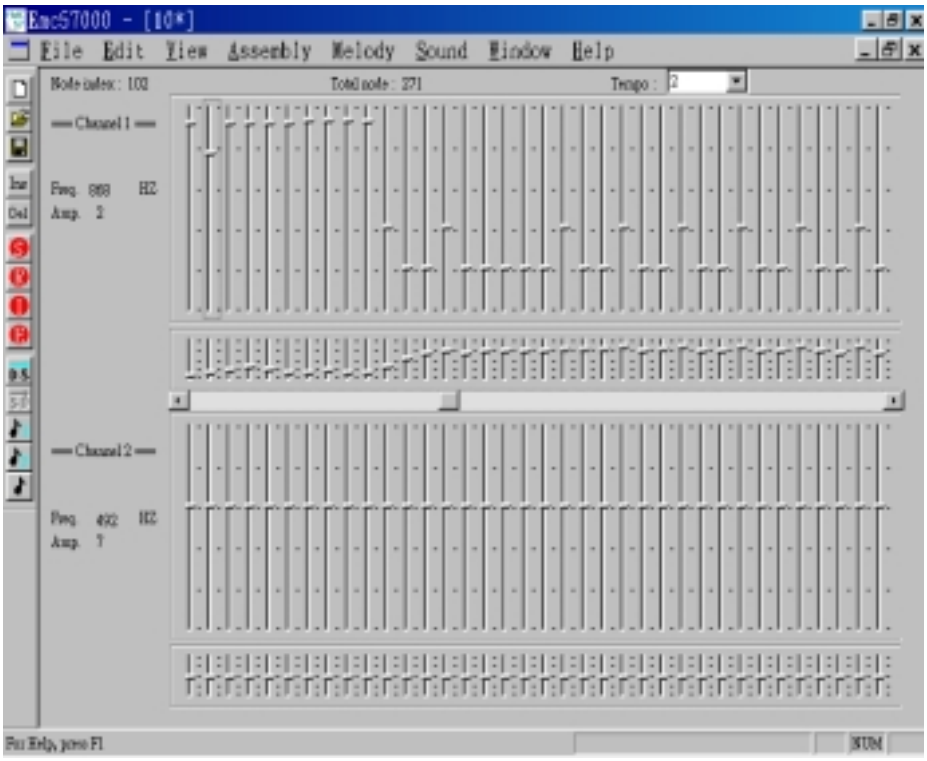
“Channel” dialog box

To select the “Tempo”, simply press the ▼ of “Tempo” box and the window will display the “Tempo” dialog box.



“Tempo” dialog box

After that, the Sound Effect Development System environment is shown as below.



Sound Effect Development System environment

The operation of Sound Effect Development System environment refers to the section 6.2.1.

6.4 Suggestions for Conversion

There are some suggestions for converting speech files to sound effect section files as following.

1. The higher sample rate for speech files, the better performance for resulting sound effect files.
2. The higher tempo value, the less ROM size occupied.
3. The smoother frequency, the better effects.

Chapter 7

Processing ASM Programs for EM57000 Series

7.1 Writing an Assembly (ASM) Program

7.1.1 File Format

The following is the Assembly file format profile applicable to EM57000 series chips.

; Comments	
PROGRAM	**;** 1) Format identification
Section_name1 **Section_name2** **----------**	**;** 2) Section definition
EM57100	**;** 3) Chip identification
String EQU #0001b String1 EQU m0	**;** 4) EQU definition
POWERON:	**;** 5) Main assembly program begins
---------- **----------**	
INIT:	

---------- ----------	; 6) Program begins at wake up

7.1.1.1 Format Identification

Identifies the program file format. “PROGRAM” identifies the file as an assembly file format.

7.1.1.2 Section Definition

Defines in this section, all the voice, melody (MLD/MDY), and sound effect (SND/SDY) files required for user’s application.

7.1.1.3 Chip Identification

Defines appropriate type of EM57000 series chip that meets user’s required application. This could be one of the following:

EM57001 (10Kx10 bits)	EM57100 (16Kx10 bits)	EM57200 (32Kx10 bits)
EM57300 (64Kx10 bits)	EM57400 (128Kx10 bits)	EM57500 (256Kx10 bits)
EM57600 (512Kx10 bits)	EM57700 (1024Kx10 bits)	

7.1.1.4 EQU definition

“EQU” can define the string of user favorite. It will let user string which is equal to a assembly string 、a memory string or number.

7.1.1.5 Main Assembly Program Begins

This section contains the main application program. The program must contain the power conservation signal labels “POWERON” and “INIT.”

“POWERON” is where program operation starts after power source (i.e., battery) is switched on.

7.1.1.6 Program Begins at Wake Up

"INIT" is one of power conservation signal labels as explained above. It is the starting point where program operation starts or "wake-up" after a power-down or "sleep" period.

> ***NOTE***
>
> *"Wake-up" is initiated whenever a rising /falling edge pulse is provided to Port 2 (P2.3, P2.2, P2.1, P2.0) for **EM57001** during "sleep" or power-down period.*
>
> *Without invoking the KEYB instruction, "wake-up" is initiated whenever a rising /falling edge pulse occurs at Port 1 (P1.3, P1.2, P1.1, P1.0) or Port 2 (P2.3, P2.2, P2.1, P2.0) for **EM57100~ EM57700.***
>
> *With the KEYB instruction is invoked, "wake-up" is initiated only when a rising /falling edge pulse occurs at Port 1 (P1.3, P1.2, P1.1, P1.0) for **EM57100~ EM57700.***

7.1.2 Assembly Program Instruction Set

No	Symbol		Description
1	PLAY SEC,#RATE	*1	Play the voice file with filename “SEC” at the speed of “RATE”
2	STOP		Stop the voice file currently on play
3	MOV A,P1	*2	Transfer data from port1 to register A
4	MOV A,P2		Transfer data from port2 to register A
5	MOV A,P3		Transfer data from port3 to register A
6	MOV P2,A		Transfer data from register A to port2
7	MOV P3,A		Transfer data from register A to port3
8	MOV P2S,A	*3	Transfer data from register A to status register of port2
9	MOV P3S,A	*4	Transfer data from register A to status register of port3
10	MOV TEMPO,#TDATA	*5	Transfer immediate data to TEMPO register
11	MOV VOL,#VDATA	*6	Transfer immediate data to VOL register
12	MOV MODE,A	*7	Transfer data from register A to MODE register
13	INCA		Increment the content of register A by 1
14	MOV M?,A	*8	Transfer data from register A to Memory
15	MOV M?,#DATA		Transfer immediate data to Memory, where data can be 0~ 15 or 0000b~ 1111b
16	MOV A,M?		Transfer data from Memory to register A
17	MOV A,#DATA		Transfer immediate data to register A, where data can be 0~ 15 or 0000b~ 1111b
18	AND A,M?		(A) and (M?)---> (A)
19	OR A,M?		(A) or (M?) ---> (A)
20	XOR A,M?		(A) xor (M?) ---> (A)
21	PAGE0	*9	Select Memory of first page
22	PAGE1	*9	Select Memory of second page
23	PAGE2	*9	Select Memory of third page
24	PAGE3	*9	Select Memory of fourth page
25	PAGE4	*9	Select Memory of fifth page
26	PAGE5	*9	Select Memory of sixth page
27	PAGE6	*9	Select Memory of seventh page
28	PAGE7	*9	Select Memory of eighth page
29	JMP LABEL		Change program location to “LABEL”
30	CAJE #DATA,LABEL		If (A)=#data, then change program location to “LABEL”, where data can be 0~ 15 or 0000b~ 1111b
31	CAJE M?,LABEL		If (A)=(M?), then change program location to “LABEL”
32	CJP LABEL		If there is voice playing, then change program location to “LABEL”
33	CJC LABEL	*10	If internal timer overflows, then change program location to “LABEL”

34	MOV TIMER,#TDATA *11	Transfer immediate data to TIMER register
35	CALL LABEL	Subroutine call (There is only one level subroutine call)
36	RET	Return from subroutine
37	RSTC *12	Reset internal timer
38	END	Enter power down mode
39	NOP	No operation
40	KEYB *13	Change P2 impedance and disable wake up function

NOTES:

*1 SEC is the voice file filename; The available speech sample rate is shown as following:

3906	4386	5000	5814	6944	8621	11364	16667
3968	4464	5102	5952	7143	8929	11905	17857
4032	4545	5208	6098	7353	9259	12500	19231
4098	4630	5319	6250	7576	9615	13158	20833
4167	4717	5435	6410	7813	10000	13889	22727
4237	4808	5556	6579	8065	10417	14706	25000
4310	4902	5682	6757	8333	10870	15625	27778
							31250

#RATE can be #4, #5,…, #22, or 4K, 5K,…, 22K, to represent 4k, 5k,…, 22k Hz playing speed respectively. Besides, #RATE can be a real integer between #4000 and #32000 to represent playing speed (Hz), the tools will find the value which is nearest to the above mentioned sample rate for this play command.

NOTE

1.When the speech file is played, the content of A register will be modified.

2.If many speech files be sequentially played, it is recommended that after playing a speech file and detecting voice to be over, next speech file is sequentially played.

*2 P1 is a 4-bit input port provided for **EM57100~ EM57700.** This instruction is not provided for **EM57001**.

*3 P2S is a 4-bit status register for Port 2. If P2S bit is “1”, the corresponding bit of Port 2 will be of tri-state; otherwise, it will be “1” or “0” according to the content of register P2.

*4 P3S is a 4-bit status register for Port 3. If P3S bit is “1”, the corresponding bit of Port 3 will be of tri-state; otherwise, it will be “1” or “0” according to the content of register P3.

> ***NOTE***
>
> *1.When P2S(P3S) is set to "1", note that the corresponding pin must NOT be floating. This will further conserve power during power-down mode.*
>
> *2.If Em57000 series chip start or reset, the bit in P2S(P3S) initially is set to "1" and the bit in P2(P3) is set to "0"..*

*5 "MOV TEMPO,#TDATA" is used to determine the tempo for melody (beat/minute) or sound effect (ms/node) playing. The value of TDATA can be one of the following**:**

Melody	57	61	65	70	76	83	92	102
(beat/minute)	114	131	153	183	229	305	458	916

Sound effect	32ms	30ms	28ms	26ms	24ms	22ms	20ms	18ms
(ms/node)	16ms	14ms	12ms	10ms	8ms	6ms	4ms	2ms

> ***NOTE***
>
> *When the instruction is executed, the content of A register will be changed.*

*6 "MOV VOL,#VDATA" is used to determine the playing volume increment of the voice/melody/sound effect. The VDATA value ranges from 0 to 7 where 0 is the lowest volume and 7 the highest.

> ***NOTE***
>
> *When the instruction is executed, the content of A register will be modified.*

*7 If the bit0 of MODE register equals "1", pin P3.3 will be treated as an LED output and will flash according to volume change whether it is tri-state or output state. If the bit0 of MODE register equals "0", pin P3.3 will be treated as a general purpose I/O pin.

If the bit3~ 1 of MODE register equals "101", pin P3.2 will generate 38K Hz square wave while it is in output state. If the bit3~ 1 of MODE register equals "000", pin P3.2 will be treated as a general purpose I/O pin. This is used in the infrared ray application.

> ***NOTE***
>
> *1. If user uses P3.3 as an LED output by setting the bit0 of MODE register to "1", it is recommended that user changes the bit0 of MODE register to "0" every time the chip switches to power-down mode. More power can be saved under this mode.*
>
> *2. The infrared ray application is provided for EM57100~ EM57700 except EM57400.*

*8, *9 With **EM57001**, there are 16x4 bits RAM available. With **EM57100 ~ EM57500**, 2 pages of memory can be selected by the instructions "PAGE0" and "PAGE1". With **EM57600** and **EM57700**, 8 pages of memory can be selected by the instructions "PAGE0"~ "PAGE7" at 16×4 bits RAM per page. Hence, "M?" can be one of [M0, M1,..., M15].

*10 EM57000 series is provided with a 6-bit timer. When internal timer overflow occurs (from 111111 to 000000), an "overflow" flag is set to "1". When the instruction "CJC" is executed and the "overflow" flag is "1", the program will change to location "LABEL" and reset the "overflow" flag. Otherwise, the next location of assembly program will be executed.

*11 "MOV TIMER,#TDATA" is used to determine the timer input clock. The value of TDATA can be one of the following**:**

TDATA	0	1	2	3
Timer input clock (KHz)	64	8	2	1
Period between overflows (ms)	1	8	32	64

NOTE
When the instruction is executed, the content of A register will be changed.

*12 When the instruction "RSTC" is executed, the internal 6-bit timer is reset to 000000. At the same time the "overflow" flag is reset to "0."

*13 When the instruction KEYB is executed for **EM57100~ EM57700**, the input pull low resistor of P2 will change from 1M to 10K (approximate). P2 wake up function is disabled at the same time. The instruction is not provided for **EM57001.**

Chapter 8

Processing ESY Programs for EM57001

8.1 Writing an Easy Format® (ESY) Program

There are 3 distinct types of Easy program file format for EM57001 series chips.

- One that contains I/O pins assignment (easy)
- One that contains triggers and outputs assignment (easy-4/8/12/16)
- One that contains only triggers assignment (easy-20)

These files should be separately written.

8.1.1 File Format with I/O Pins Assignment State (EASY)

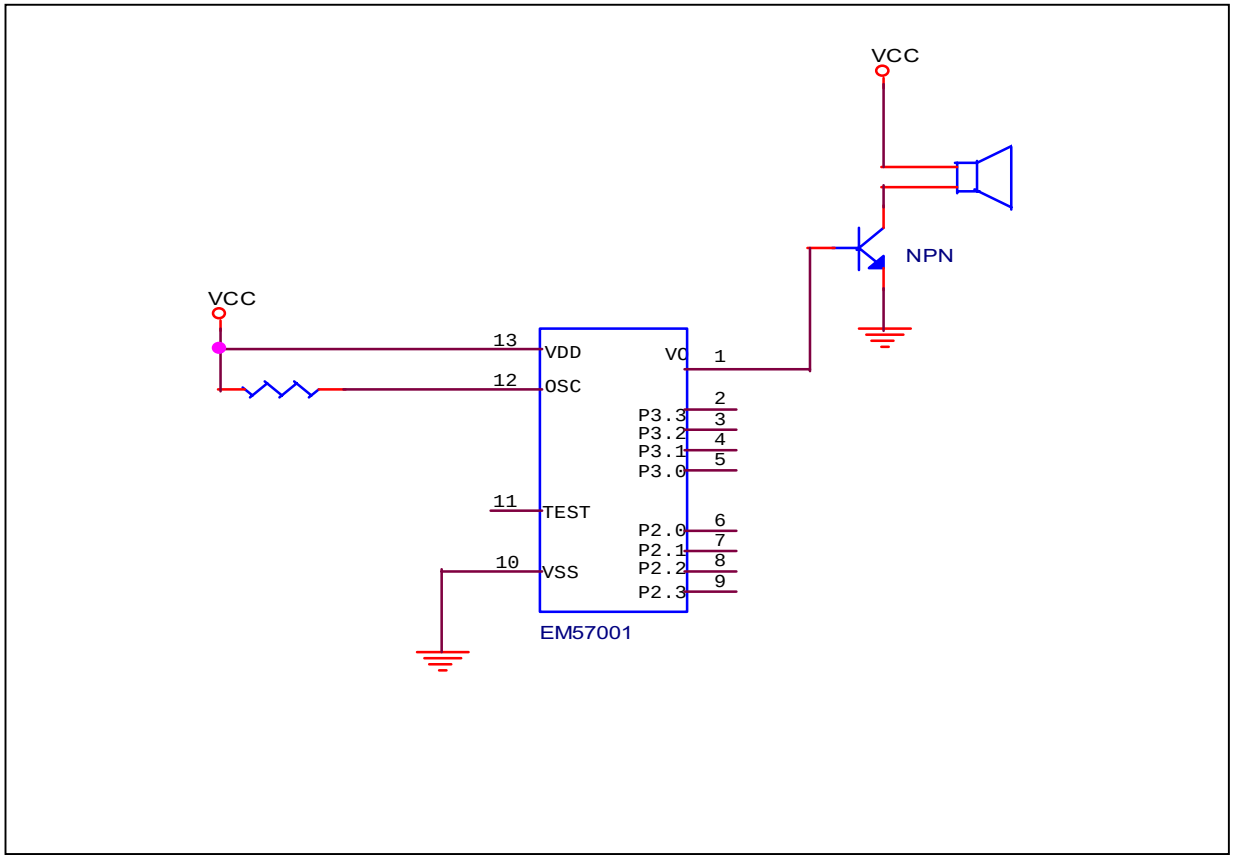
The following shows the basic requirements of the file format with I/O pins assignment. Refer to Section 8.1.4 for the details of each section definition.

```
; Comments
EASY                    ; 1) Format identification
Section_name1           ; 2) Section definition
Section_name2
    ------------------
EM57001                 ; 3) Chip identification
M0:R(4)                 ; 4) Option selection
M1:R(5)
NO_STOP
STATE                   ; 5) P2.0,P2.1,P2.2,P2.3,P3.0,P3.1,P3.2,P3.3
S_name1:                ;    (Define I/O pins assignment state)
S_name2:
    ------------------
PATH                    ; 6) Path definition
PATH0:
P_name1:
P_name2:
    ------------------
```

8.1.1.1 Format Identification

Identifies the program file format. “EASY” identifies the file as an ”easy” file format.

The application circuit is illustrated below.



Easy Application Circuit

8.1.2 File Format with Triggers and Outputs Assignment State (EASY-4/8/12/16)

The following shows the basic requirements of the "Easy-n" (n=4/8/12/ 16) file format with triggers and outputs assignment. Refer to Section 8.1.4 for the details of each section definition.

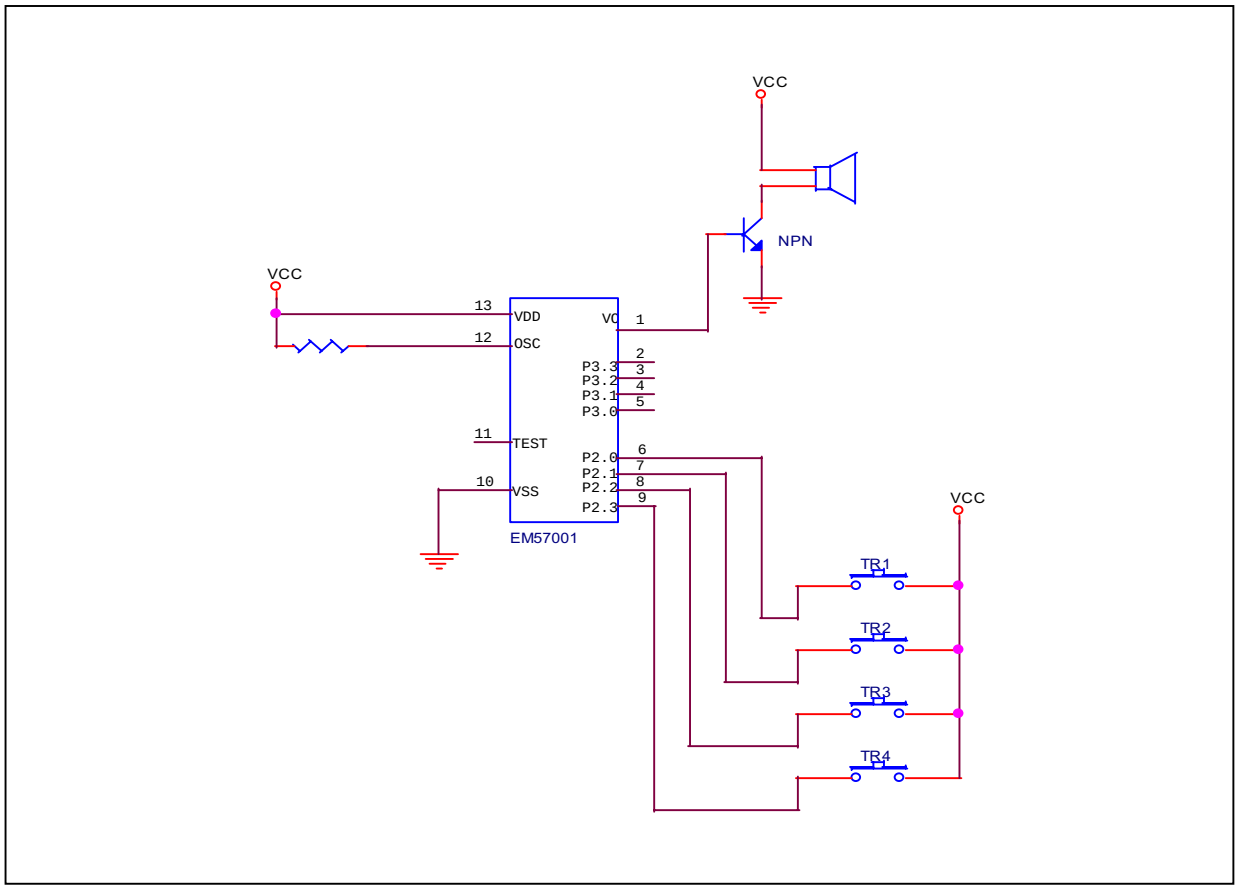
```
; Comments
EASY–n                          ; 1) Format identification
                                ;     (n=4/8/12/16)
Section_name1                   ; 2) Section definition
Section_name2
    ------------------
EM57001                         ; 3) Chip identification
M0:R(4)                         ; 4) Option selection
M1:R(5)
M2:R(16)
NO_STOP
TR_STATE                        ; 5) Trigger state definition
T_name1:
T_name2:
    ------------------
OUT_STATE                       ; 6) Output state definition for Port 3
O_name1:                        ;     (P3.3, P3.2, P3.1, P3.0)
O_name2:
    ------------------
PATH                            ; 7) Path definition
PATH0:
P_name1:
P_name2:
    ------------------
```

8.1.2.1 Format Identification

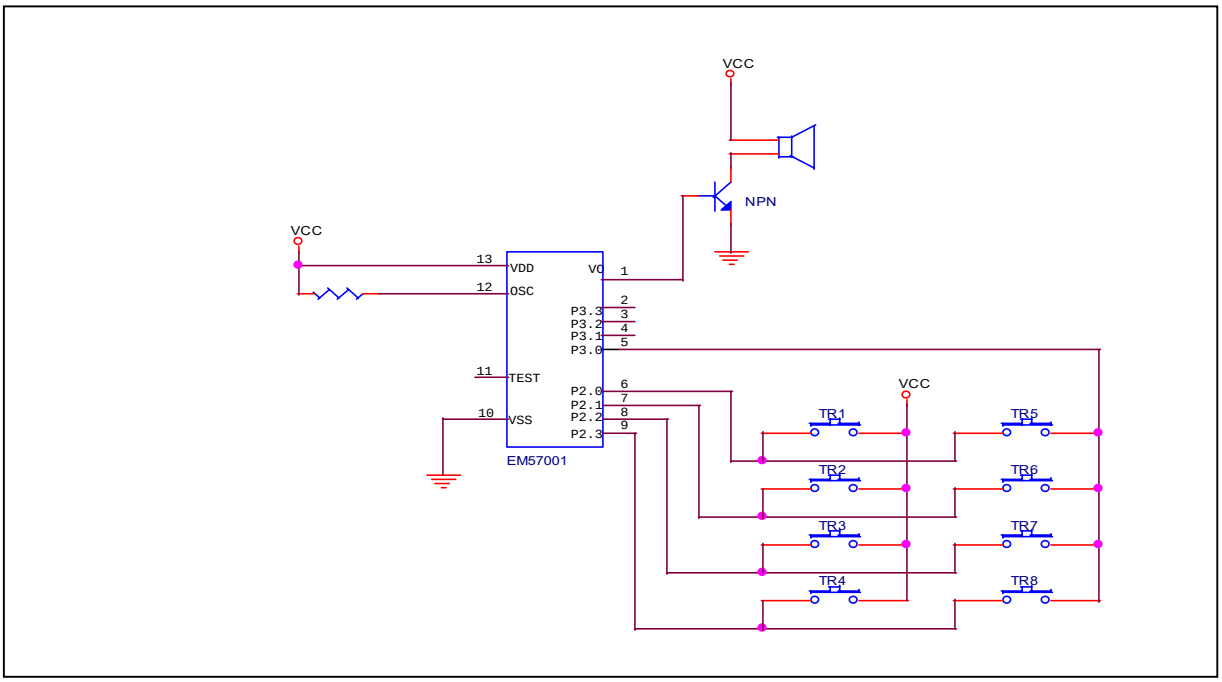
Identifies the program file format. "EASY-n" identifies the file as one of the following**:**

Format	Maximum Trigger Number	I/O Pin Available
Easy-4	4	P3.3, P3.2, P3.1, P3.0
Easy-8	8	P3.3, P3.2, P3.1
Easy-12	12	P3.3 , P3.2
Easy-16	16	P3.3

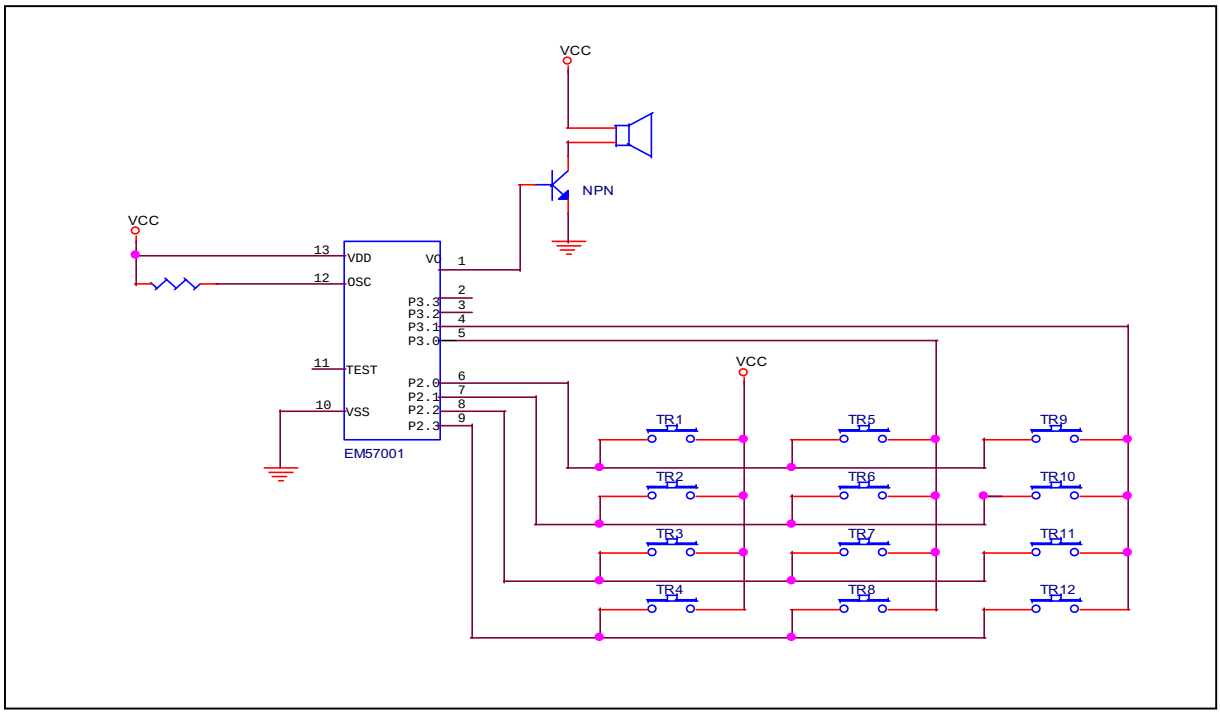
The application circuits for each format are illustrated below.



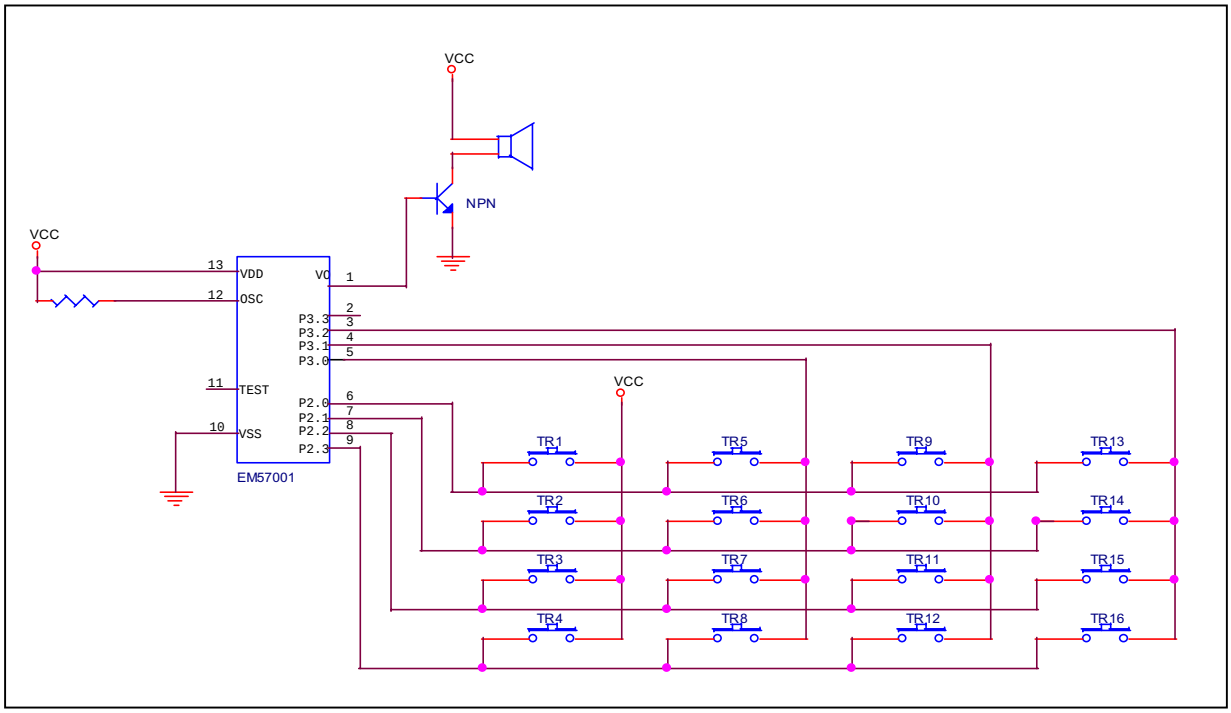
Easy-4 Application Circuit



Easy-8 Application Circuit



Easy-12 Application Circuit



Easy-16 Application Circuit

8.1.3 File Format with only Triggers Assignment State (EASY-20)

The following shows the basic requirements of the "Easy-20" file format with only triggers assignment. Refer to Section 8.1.4 for the details of each section definition.

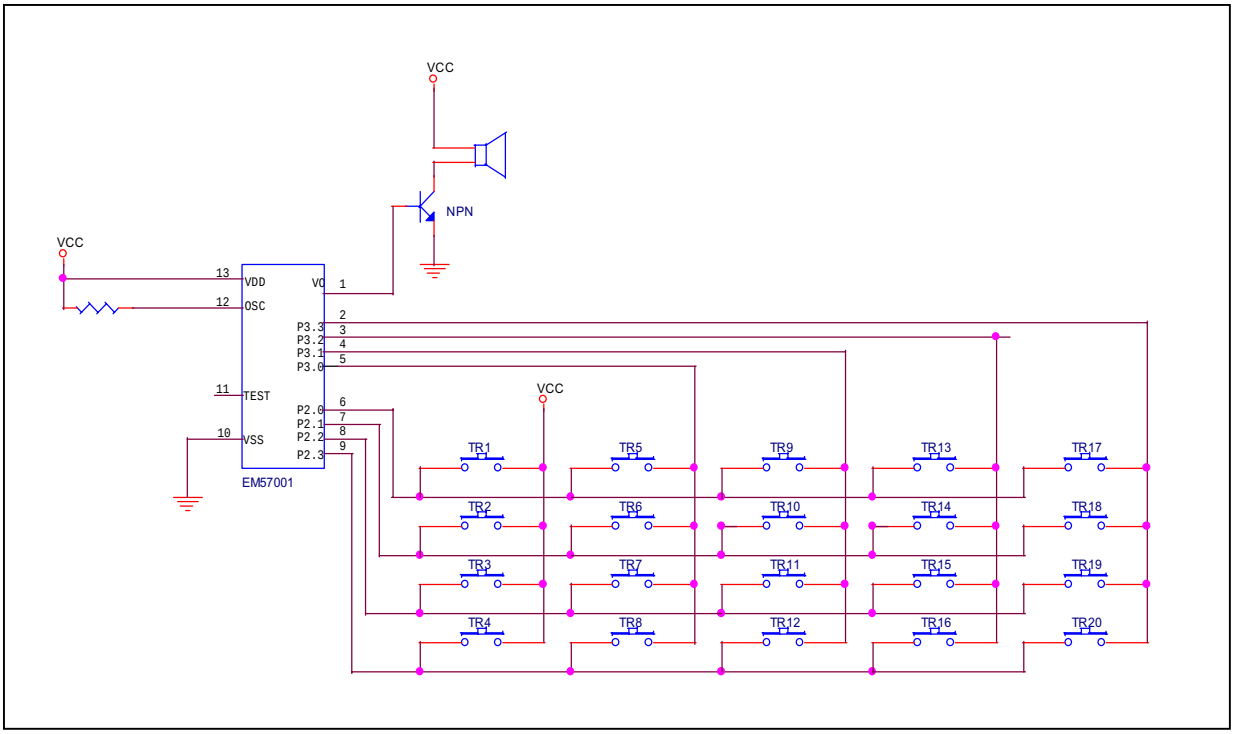
```
Comments
EASY–20                          ; 1) Format identification
Section_name1                    ; 2) Section definition
Section_name2
    ------------------
EM57001                          ; 3) Chip identification
M0:R(4)                          ; 4) Option selection
M1:R(5)
M2:R(16)
NO_STOP
TR_STATE                         ; 5) Trigger state definition
T_name1:
T_name2:
    ------------------
PATH                             ; 6) Path definition
PATH0:
P_name1:
P_name2:
    ------------------
```

8.1.3.1 Format Identification

Identifies the program file format.

Format	Maximum Trigger Number	I/O Pin Available
Easy-20	20	No

The application circuit is illustrated below.



Easy-20 Application Circuit

8.1.4 Section Description for EASY, EASY-4/8/12/16 and EASY-20

The following describes the details of each section definition in each Easy file format.

8.1.4.1 Section Definition

Defines in this section, all the voice, melody, and sound effect files required for user's application.

8.1.4.2 Chip Identification

Defines appropriate type of EM57000 series chip that meets user's required application. This should be EM57001 chip

8.1.4.3 Option Selection

M0, M1, M2

The following 4-bit registers are available for user's application depended on different Easy file format. Each register may be defined as a general-purpose register or a random number. When option is defined as "**M0: R(n)**", M0 is a random number with range from 0 to n-1, where "n" can be 2 to 16.

EASY	EASY4	EASY8	EASY12	EASY16	EASY20
M0, M1	M0, M1, M2	M0, M1	M0	M0	M0, M1, M2

When option is not defined, M0 is treated as a general-purpose register. This is also true to M1 and M2.

NO_STOP

By default, when trigger is received while the chip is playing, it is immediately stopped. If **NO_STOP** option is defined, playing will continue allowing user to do something else during playing.

8.1.4.4 I/O Pin State Definition for EASY

User can define a maximum of **16** I/O pins assignment statement, each under its own "state" name (i.e., S_name1, S_name2,... etc.). A statement is prefixed by a "state "name followed by 8 I/O options of the corresponding I/O pins. The I/O options are separated by spaces or commas (,). These I/O options can be one of the following**:**

1. **P_name:** when the rising pulse happens at the corresponding I/O pin, the path equation with path name "P_name" is executed.
2. **/P_name:** when the falling pulse happens at the corresponding I/O pin, the path equation with path name "P_name" is executed.
3. **P_name1/p_name 2:** when the rising pulse happens at the corresponding I/O pin, the path equation with path name "P_name1" is executed; when the falling pulse happens at the corresponding I/O pin, the path equation with path name "P_name2" is executed.
4. **1:** Assign the corresponding I/O pin to "H".
5. **0:** Assign the corresponding I/O pin to "L".
6. **P+:** Send a pulse train beginning from "H" to the corresponding I/O pin.
7. **P-:** Send a pulse train beginning from "L" to the corresponding I/O pin.
8. **X:** No action of the corresponding I/O pin.

NOTE

1. *When user uses the Easy file format, the Port2 is connected to the 1M pull-down resistor (refer to Appendix A). The sink current of Port2 will be smaller.*
2. *When the Easy file format powers on, if the I/O pin is immediately defined as "X" (don't care), the I/O pin will set to "H".*
3. *During executing the Easy file format, if the I/O pin is defined as "X" (don't care), the I/O pin will be unchanged.*

8.1.4.5 Trigger State Definition for EASY-4/8/12/16 and EASY-20

Trigger State is defined for each trigger state assignment. User can define up to **16** states maximum with different trigger state names, e.g., T_name1, T_name2,... etc.

Trigger state name must be followed by its corresponding trigger selection option "N", where "N" is the number 1 to 4 of the Easy-4 format, 1 to 8 of the Easy-8 format,… ,or 1 to 20 of the Easy-20 format. Selection of options are separated by spaces or commas (,).

The trigger selection options can be one of the following:

1. **P_name** when the corresponding trigger is pressed, the path equation with path name "P_name" is executed.
2. **/P_name** when the corresponding trigger is released, the path equation with path name "P_name" is executed.
3. **P_name1/P_name2** when the corresponding trigger is pressed, the path equation with path name "P_name1" is executed; when the corresponding trigger is released, the path equation with path name "P_name2" is executed.
4. **X** disables the corresponding trigger (no action).

8.1.4.6 Output State Definition for EASY-4/8/12/16

Define output state to set the output state of P3. User can define as many P3 output states as needed. Each should have different output state name, such as O_name1, O_name2,... etc.

These output state names are followed by the output settings of Port 3 as follows**:**

P3 = [O3 O2 O1 O0]

where O3, O2, O1 and O0 stand for P3.3, P3.2, P3.1, and P3.0 respectively.

Each of the O3 to O0 elements may be defined to be one of the following**:**

1. **1** Define the corresponding I/O pin to be "H".
2. **0** Define the corresponding I/O pin to be "L".
3. **P+** Send a pulse train beginning from "H" to the corresponding I/O pin.
4. **P–** Send a pulse train beginning from "L" to the corresponding I/O pin.
5. **S+** Send a positive pulse to the corresponding I/O pin.
6. **S–** Send a negative pulse to the corresponding I/O pin.
7. **F** Define the designated I/O pins as tri_state. With EM57001 series, all Port 3 I/O pins (P3.3, P3.2, P3.1, P3.0) can be set to tri_state under Easy-4 format, only P3.3, P3.2 and P3.1 can do under Easy-8 format, only P3.3, and P3.2 can do under Easy-12 format, and only P3.3 can do under Easy-16 format.
8. **X** keep the corresponding output pin to be unchanged

> ***NOTE***
> *When the Easy-n format powers on, if the output pin is immediately defined as "X" (don't care), the output pin will set to "H".*

Under EASY-n format, the pins of Port 3 are not configurable as they are used for keyboard scan matrix. Hence the following pins must be "X" (don't care):

EASY-8	EASY-12	EASY-16
P3.0 (O0)	P3.0 (O0), P3.1 (O1)	P3.0 (O0), P3.1 (O1), P3.2 (O2)

8.1.4.7 Path Definition

User can define as many path equations as needed and assign each with different path names, i.e., P_name1, P_name2,... etc. Each path equation line should contain no more than 255 characters (maximum). If a line is not enough in defining a path equation, additional lines may be added but the additional lines must begin with an ampersand (&) symbol to indicate continuation to previous line of the same path equation definition.

Each path equation definition may contain one or more of the following parameters as user may require**:**

- **State name** **for EASY**
  such as "S_name1" to set the I/O state for EASY.
- **Tr_State name** **for EASY-4/8/12/16, EASY-20**
  such as "T_name1" to define the trigger state for EASY-4~20.
- **Out_State name** **for ESAY-4/8/12/16**
  such as "O_name1" to define the output state of Port 3 for EASY-4~16.
- ***Out_State name/**
  [Out_State name] **for ESAY-4/8/12/16**
  such as "O_name1" to define the output state of Port 3 for EASY-4~16.

> ***NOTE***
> *When a section is playing, in order to change the Out_State and do not interrupt the playing by pressing the trigger. User must use this parameter and connect the "**RETURN**" parameter to return the section's playing.*

- **Path name** such as "P_name1" to execute the path equation with the path name "P_name1".
- **Section name(Sample rate)** play the voice/ melody/ sound effect file. "Sample rate" defines the playing

speed of the voice file. It could be either 4, 5,…, 22, or 4K, 5K,…, 22K, to represent 4k, 5k,…, 22k Hz playing speed respectively. Besides, it could be a real integer between 4000 and 32000 to represent playing speed (Hz).

- ***Section name(Sample rate)/ [Section name(Sample rate)]** play the voice/ melody/ sound effect file without calling the scan loop.
- **Section name(Sample rate)×n** play the voice/ melody/ sound effect file that is repeated by n times.
- ***Section name(Sample rate)×n/ [Section name(Sample rate)]×n** play the voice/ melody/ sound effect file that is repeated by n times without calling the scan loop.
- **{Section name1(Sample rate)×n, [Section name2(Sample rate)]×m ….}×g** after playing the local looping, it plays global looping which is the looping of a combination of section names.

  g: the number of global looping of section name1, section name2.

  n: the number of local looping of section name1.

  m: the number of local looping of section name2 and so on.

> ***NOTE***
> 1. *Section name can be replaced by #number according to the sequence of Section name in the section definition. The minimum number is "0".*
> 2. *The all parameters in the path definition can be placed in the "{ }".*

- **TEMPO=data** define the tempo for melody (beat/ minute) or sound effect (ms/node) playing. "data" could be one of the following**:**

Melody	57	61	65	70	76	83	92	102
(beat/minute)	114	131	153	183	229	305	458	916

Sound effect	32ms	30ms	28ms	26ms	24ms	22ms	20ms	18ms
(ms/node)	16ms	14ms	12ms	10ms	8ms	6ms	4ms	2ms

- **VOL=data** set the voice/melody/sound effect playing volume in 8 steps increment. "data" is any value from 0 to 7 where 0 is the lowest and 7 the highest.
- **Delay(n)** delay "n" seconds. The minimum delay unit is 0.008 second. For example, Delay(0.2) means 0.2 second delay and Delay(8.6) means 8.6 seconds delay.
- **Rate=n** **for EASY, EASY-4/8/12/16** define the flash rate of flash output pins that are set to "P+" or "P-".

> ***NOTE***
>
> *"n" can be number 1 to 8 of the eight flash rate variables 16/**n** Hz. The parameter for EASY-20 format can not be used.*

- **Mi=data/Mj** define the content of register "Mi" as data/ or Mj.

> ***NOTE***
>
> *Data can be 0, 1, 2, 3, 4,.....15.*

> ***NOTE***
>
> *For the following paragraphs, if DELAY(n) is not used, Mi, Mj and Mk can be defined as one of the followings according to "EASY" and "EASY-n" format:*

EASY	EASY-4	EASY-8	EASY-12	EASY-16	EASY-20
M0~ M4	M0~ M5	M0~ M4	M0~ M3	M0~ M3	M0~ M6

> ***NOTE***
>
> *Or if DELAY(n) is used, Mi, Mj and Mk can be defined as one of the following according to "EASY" and "EASY-n" format:*

EASY	EASY-4	EASY-8	EASY-12	EASY-16	EASY-20
M0, M1	M0~ M2	M0, M1	M0	M0	M0~ M3

- **Mi=P2** **for EASY**
  define the content of register "Mi" as the content that is read from "P2".
- **Mi=P3** **for EASY, EASY-4/8/12/16**
  define the content of register "Mi" as the content that is read from "P3".

> ***NOTE***
> *According to different "EASY-n" format, the following bits of register "Mi" will be "0".*

EASY-4	EASY-8	EASY-12	EASY-16
-	Mi(0)	Mi(0), Mi(1)	Mi(0), Mi(1), Mi(2)

- **Mi=Mj+data** define the content of register "Mi" as the content of register "Mj" plus data.
- **Mi=Mj-data** define the content of register "Mi" as the content of register "Mj" subtract data.
- **Mi=Mj&data/**
  Mi=Mj.and.data define the content of register "Mi" as the content of register "Mj" **AND** data.
- **Mi=Mj&Mk/**
  Mi=Mj.and.Mk define the content of register "Mi" as the content of register "Mj" **AND** that of register "Mk".
- **Mi=Mj|data/**
  Mi=Mj.or.data define the content of register "Mi" as the content of register "Mj" **OR** data.
- **Mi=Mj|Mk/**
  Mi=Mj.or.Mk define the content of register "Mi" as the content of register "Mj" **OR** that of register "Mk".
- **Mi=Mj^data/**
  Mi=Mj.xor.data define the content of register "Mi" as the content of register "Mj" **XOR** data.
- **Mi=Mj^Mk/**
  Mi=Mj.xor.Mk define the content of register "Mi" as the content of register "Mj" **XOR** that of register "Mk".

- **Mi=~Mj/**
  Mi=.inv.Mj define the content of register "Mi" as the **INVERT**ed content of register "Mj"
- **Mi(j)=1/0** define the "jth" bit of register "Mi" as 1/ or 0, where j can be 0 to 3.
- **P2=Mi/data for EASY**
  define the output content of port2 as the content of register "Mi" or data.
- **P3=Mi/data for EASY, EASY-4/8/12/16**
  define the output content of port3 as the content of register "Mi" or data.

NOTE

1. *For "Easy" format, user must take care for the status of trigger pins (in port2 or port3) to let trigger state take effect.*
2. *According to different "EASY-n" format, the some pins of port3 are used to be scanning pins, the following bits of register "Mi" and data will be ignored.*

EASY-4	EASY-8	EASY-16	EASY-16
---	Bit 0	Bit 0~1	Bit 0~2

- **Mi?data:p_name** if the content of register "Mi" equals to data, the path equation with the path name "p_name" will be executed.
- **Mi?Mj:p_name** if the content of register "Mi" equals to that of register "Mj", the path equation with the path name "p_name" will be executed.
- **Mi(j)?1/0:p_name** if the "jth" bit of register "Mi" equals to 1/ or 0, the path equation with the path name "p_name" will be executed.
- **Mi?[d x d x]:[p_0 x p_2 …]** If Mi(3), Mi(1) equals to "00", the path equation with path name "p_0" will be executed. If Mi(3), Mi(1) equals to "01", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If Mi(3), Mi(1)equals to "10", the path equation with path name "p_2" will be executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

NOTE

1. "X" means "don't care".

2. For "Mi?[d x d x]", the parameters in "[ ]" can be "d" or "x", the decision of jump or not is depended on the corresponding bit of register "Mi" that is defined as "d".

- **Mi:[p_0 x p_2 ...]** if the content of register "Mi" equals to "0", the path equation with path name "p_0" will be executed. If that of register "Mi" equals to "1", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If that of register "Mi" equals to "2", the path equation with path name "p_2" will be executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

- **Px.x?H/L:p_name for EASY**
  if the pin Px.x is "H"/ or "L", the path equation with path name "p_name" will be executed. Where "Px.x" is any pin of P2.0, P2.1,…, P3.3 under EASY format.

- **TRn?H/L:p_name for EASY-4/8/12/16, EASY-20**
  if the trigger TRn is "H"/ or "L", the path equation with path name "p_name" will be executed. Where "TRn" is any value from TR1 to TR4 under Easy-4 format, TR1 to TR8 under Easy-8 format,…, TR1 to TR20 under EASY-20 format.

- **P2?[I'3 I'2 I'1 I'0]:p_name for EASY**
  if the I/O pins of Port2 (P2.3, P2.2, P2.1, P2.0) is equal to [I'3 I'2 I'1 I'0], the path equation with path name "p_name" will be executed. Where I'3 to I'0 can be"1", "0", or "X".

- **P2?[d x d x]:[p_0 x p_2 …] for EASY**
  If P2.3, P2.1 equals to "00", the path equation with path name "p_0" will be executed. If P2.3, P2.1 equals to "01", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If P2.3, P2.1 equals to "10", the path equation with path name "p_2" will be executed, so on and so forth. The

maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

> ***NOTE***
> *For "P2?[d x d x]", the parameters in "[ ]" can be "d" or "x", the decision of jump or not is depended on the corresponding pin of port2 that is defined as "d".*

- **P3?[I3 I2 I1 I0]:p_name for EASY, EASY-4/8/12/16**

  if the I/O pins of Port3 (P3.3, P3.2, P3.1, P3.0) is equal to [I3 I2 I1 I0], the path equation with path name "p_name" will be executed. Where I3 to I0 can be"1", "0", or "X".

> ***NOTE***
> *"X" means "don't care". The following parameters must be "X" according to different "EASY-n" format:*

EASY-4	EASY-8	EASY-12	EASY-16
-	I0	I1, I0	I2, I1, I0

- **P3?[d x d x]:[p_0 x p_2…..] for EASY, EASY-4/8/12/16**

  If P3.3, P3.1 equals to "00", the path equation with path name "p_0" will be executed. If P3.3, P3.1 equals to "01", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If P3.3, P3.1 equals to "10", the path equation with path name "p_2" will be executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

> ***NOTE***
> *1. For "P3?[d x d x]", the parameters in "[ ]" can be "d" or "x", the decision of jump or not is depended on the corresponding pin of port3 that is defined as "d".*
> *2."X" means "don't care". The following parameters must be "X" according to different "EASY-n" format:*

EASY-4	EASY-8	EASY-12	EASY-16
-	P3.0	P3.1, P3.0	P3.2, P3.1, P3.0

- **PLAY?:p_name** if there is voice/ melody/ sound effect playing, the path equation with the path name "p_name" will be executed.
- **FV_ON** **for EASY, EASY-4/8/12/16**
  define pin P3.3 as LED output that will flash according to voice volume. If this option is used, P3.3 will be only treated as LED output and must set to an output purpose pin. If the option is not used, P3.3 will be treated as a general purpose I/O pin.

NOTE

*If P3.3 is set to be an input pin, the input function will be disabled after **FV_ON** under EASY format. This parameter can not be used under EASY-20 format.*

- **FV_OFF** **for EASY, EASY-4/8/12/16**
  turn off LED output flash function.

NOTE

It is recommended that programmer turn off LED output flash function every time he defines a chip power down to further minimize power consumer during power down. This parameter can not be used under EASY-20 format.

- **RETURN** return to the location where interruption occurred due to some other actions.
- **STOP** force to stop the currently playing voice, melody or sound effect.
- **END** power down the chip and stop the program.

NOTE

1. *Path equation "PATH0" must be defined for power on execution.*
2. *Only when one of Port 2 pins gets a rising/ falling edge pulse that the EM57001 series chips will wake up from power down under EASY format.*
3. *Only when one of the trigger inputs gets a rising/ falling edge pulse that the EM57001 series chips will wake up from power down under EASY-n format.*

8.1.4.8 Summary of Parameters in Path Equation

The following table lists the parameters that can be used in the path equation.

Parameters	EASY	EASY-4~ 16	EASY-20
State name	Y	---	---
Tr_State name	---	Y	Y
Out_State name	---	Y	---
[Out_State name]	---	Y	---
Path name	Y	Y	Y
Section name(Sample rate)	Y	Y	Y
[Section name(Sample rate)]	Y	Y	Y
Section name(Sample rate) × n	Y	Y	Y
[Section name(Sample rate)] × n	Y	Y	Y
{Section name1 (Sample rate) × n, [Section name2 (Sample rate)] × m.… } × g.	Y	Y	Y
TEMPO=data	Y	Y	Y
VOL=data	Y	Y	Y
Delay(n)	Y	Y	Y
Rate=n	Y	Y	---
Mi=data/ Mj	Y	Y	Y
Mi=P2	Y	---	---
Mi=P3	Y	Y	---
Mi=Mj+data/ Mj-data	Y	Y	Y
Mi=Mj.and.data/ Mj.and.Mk	Y	Y	Y
Mi=Mj.or.data/ Mj.or.Mk	Y	Y	Y
Mi=Mj.xor.data/ Mj.xor.Mk	Y	Y	Y
Mi=.inv.Mj	Y	Y	Y
Mi(j)=1/ 0	Y	Y	Y
P2=Mi/ data	Y	---	---
P3=Mi/ data	Y	Y	---
Mi?data:p_name	Y	Y	Y
Mi?Mj:p_name	Y	Y	Y
Mi(j)?1/ 0:p_name	Y	Y	Y
Mi?[d x d x]:[p_0 x p_2…]	Y	Y	Y
Mi:[p_0 x p_2...]	Y	Y	Y
Px.x?H/ L:p_name	Y	---	---
TRn?H/ L:p_name	---	Y	Y
P2?[I'3 I'2 I'1 I'0]:p_name	Y	---	---
P2?[d x d x]:[p_0 x p_2…]	Y	---	---

P3?[I3 I2 I1 I0]:p_name	Y	Y	---
P3?[d x d x]:[p_0 x p_2…]	Y	Y	---
PLAY?:p_name	Y	Y	Y
FV_ON	Y	Y	---
FV_OFF	Y	Y	---
RETURN	Y	Y	Y
STOP	Y	Y	Y
END	Y	Y	Y

8.2 Examples for Easy Format® (ESY) Programs

The following examples demonstrate the EASY Format programming capability. Comments of the program are followed by the ";" sign.

8.2.1 Example 1: ONE SHOT (port 2.0) – Irretriggerable in EASY Format

```
EASY

ONE.WAV    7K

EM57001

STATE
;Statename    P2.0      P2.1  P2.2  P2.3  P3.0  P3.1  P3.2  P3.3
STATE0:       PATH1     X     X     X     X     X     X     X
BUSY:         X         X     X     X     X     X     X     X

PATH
PATH0: STATE0 END
PATH1: BUSY ONE.WAV(8K) STATE0 END
```

8.2.2 Example 2: ONE SHOT (trigger 1) – Retriggerable by itself in EASY-4 Format

```
EASY-4

ONE.WAV    7K

EM57001

TR_STATE
;Statename    Tr1        Tr2   Tr3   Tr4
STATE0:       PATH1      X     X     X

OUT_STATE
OUT0: P3=[0 0 0 0]
OUT1: P3=[1 1 1 1]

PATH
PATH0: OUT0 STATE0 END
PATH1: OUT1 ONE.WAV OUT0 END
```

8.2.3 Example 3: ONE SHOT (trigger 1 to trigger 4) – Retriggerable by the other pins in EASY-8 Format

```
EASY-8


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57001


TR_STATE

;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8

STATE0:      P1    P2    P3    P4    X     X     X     X
STATE1:      X     P2    P3    P4    X     X     X     X
STATE2:      P1    X     P3    P4    X     X     X     X
STATE3:      P1    P2    X     P4    X     X     X     X
STATE4:      P1    P2    P3    X     X     X     X     X


OUT_STATE

OUT0: P3=[0 0 0 X]
OUT1: P3=[1 1 1 X]


PATH

PATH0: OUT0 STATE0 END
P1: OUT1 STATE1 ONE.WAV OUT0 STATE0 END
P2: OUT1 STATE2 TWO.WAV OUT0 STATE0 END
P3: OUT1 STATE3 THREE.WAV OUT0 STATE0 END
P4: OUT1 STATE4 FOUR.WAV OUT0 STATE0 END
```

8.2.4 Example 4: LEVEL HOLD (trigger 1 to trigger 4) – Repeated playing, Retriggerable by the other pins in EASY-12 Format

```
EASY-12


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57001


TR_STATE

;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8…Tr12

STATE0:      P1    P2    P3    P4    X     X     X     X X X X X
STATE1:      /P11  P2    P3    P4    X     X     X     X X X X X
STATE2:      P1    /P11  P3    P4    X     X     X     X X X X X
STATE3:      P1    P2    /P11  P4    X     X     X     X X X X X
STATE4:      P1    P2    P3    /P11  X     X     X     X X X X X


OUT_STATE

OUT0: P3=[0 0 X X]
OUT1: P3=[1 1 X X]


PATH

PATH0: OUT0 STATE0 END
P1: OUT1 STATE1 ONE.WAV P1
P2: OUT1 STATE2 TWO.WAV P2
P3: OUT1 STATE3 THREE.WAV P3
P4: OUT1 STATE4 FOUR.WAV P4
P11: OUT0 STATE0 END
```

8.2.5 Example 5: LEVEL HOLD (trigger 1 to trigger 4) – Unrepeated playing, Retriggerable by the other pins in EASY-16 Format

```
EASY-16


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57001


TR_STATE

;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8…Tr16

STATE0:      P1    P2    P3    P4    X     X     X     X X X X…
STATE1:      /P11  P2    P3    P4    X     X     X     X X X X…
STATE2:      P1    /P11  P3    P4    X     X     X     X X X X…
STATE3:      P1    P2    /P11  P4    X     X     X     X X X X…
STATE4:      P1    P2    P3    /P11  X     X     X     X X X X…


OUT_STATE

OUT0: P3=[0 X X X]
OUT1: P3=[1 X X X]


PATH

PATH0: OUT0 STATE0 END
P1: OUT1 STATE1 ONE.WAV OUT0 STATE0 END
P2: OUT1 STATE2 TWO.WAV OUT0 STATE0 END
P3: OUT1 STATE3 THREE.WAV OUT0 STATE0 END
P4: OUT1 STATE4 FOUR.WAV OUT0 STATE0 END
P11: OUT0 STATE0 END
```

8.2.6 Example 6: LEVEL HOLD (trigger 1 to trigger 4) – Complete cycle, Retriggerable in EASY-20 Format

```
EASY-20


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57001


TR_STATE
;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8…Tr20
STATE0:      P1    P2    P3    P4    X     X     X     X X X X…


PATH
PATH0: STATE0 END
P1: ONE.WAV TR1?H:P1 STATE0 END
P2: TWO.WAV TR2?H:P2 STATE0 END
P3: THREE.WAV TR3?H:P3 STATE0 END
P4: FOUR.WAV TR4?H:P4 STATE0 END
```

8.2.7 Example 7: LEVEL HOLD (trigger 1 to trigger 4) – Complete cycle, Irretriggerable in EASY-20 Format

```
EASY-20


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57001


TR_STATE
;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8…Tr20
STATE0:      P1    P2    P3    P4    X     X     X     X X X X…
BUSY:        X     X     X     X     X     X     X     X X X X…


PATH
PATH0: STATE0 END
P1: BUSY ONE.WAV TR1?H:P1 STATE0 END
P2: BUSY TWO.WAV TR2?H:P2 STATE0 END
P3: BUSY THREE.WAV TR3?H:P3 STATE0 END
P4: BUSY FOUR.WAV TR4?H:P4 STATE0 END
```

Chapter 9

Processing ESY Programs for EM57100~ EM57700

9.1 Writing an Easy Format® (ESY) Program

There are 4 distinct types of Easy program file format for EM57100~ EM57700 series chips.

- One that contains I/O pins assignment (easy)
- One that contains triggers and outputs assignment (easy-4/8/12/16/20)
- One that contains outputs and fixed trigger paths assignment (easy-32)
- One that contains only fixed trigger paths assignment (easy-64/128)

These files should be separately written.

9.1.1 File Format with I/O Pins Assignment State (EASY)

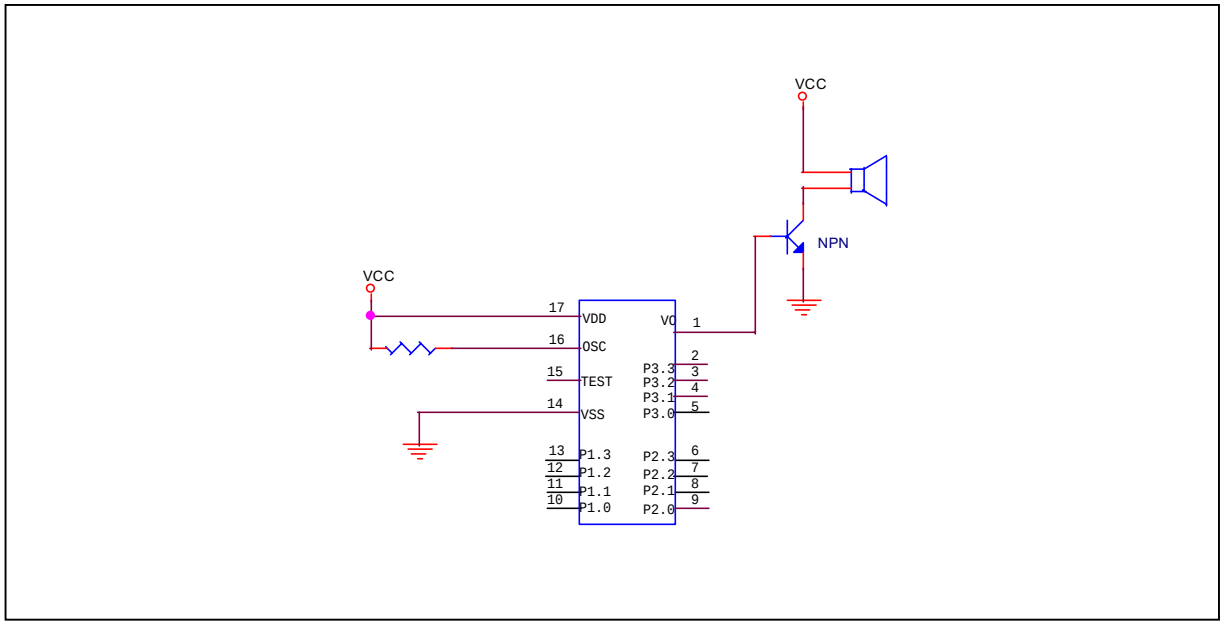
The following shows the basic requirements of the file format with I/O pins assignment. Refer to Section 9.1.5 for the details of each section definition.

```
; Comments
EASY                    ; 1) Format identification
Section_name1           ; 2) Section definition
Section_name2
  ----------------
EM57100                 ; 3) Chip identification
M0:R(4)                 ; 4) Option selection
M1:R(5)
M2:R(16)
NO_STOP
STATE                   ; 5) P1.0, P1.1, P1.2, P1.3, P2.0, P2.1, P2.2,
S_name1:                ;    P2.3, P3.0, P3.1, P3.2, P3.3
S_name2:                ;    (Define I/O pins assignment state)
  ----------------
PATH
PATH0:                  ; 6) Path definition
P_name1:
P_name2:
  ----------------
```

9.1.1.1 Format Identification

Identifies the program file format. “EASY” identifies the file as an ”easy” file format.

The application circuit is illustrated below.



Easy Application Circuit

9.1.2 File Format with Triggers and Outputs Assignment State (EASY-4/8/12/16/20)

The following shows the basic requirements of the "Easy-n" (n=4/8/12/16/20) file format with triggers and outputs assignment. Refer to Section 9.1.5 for the details of each section definition.

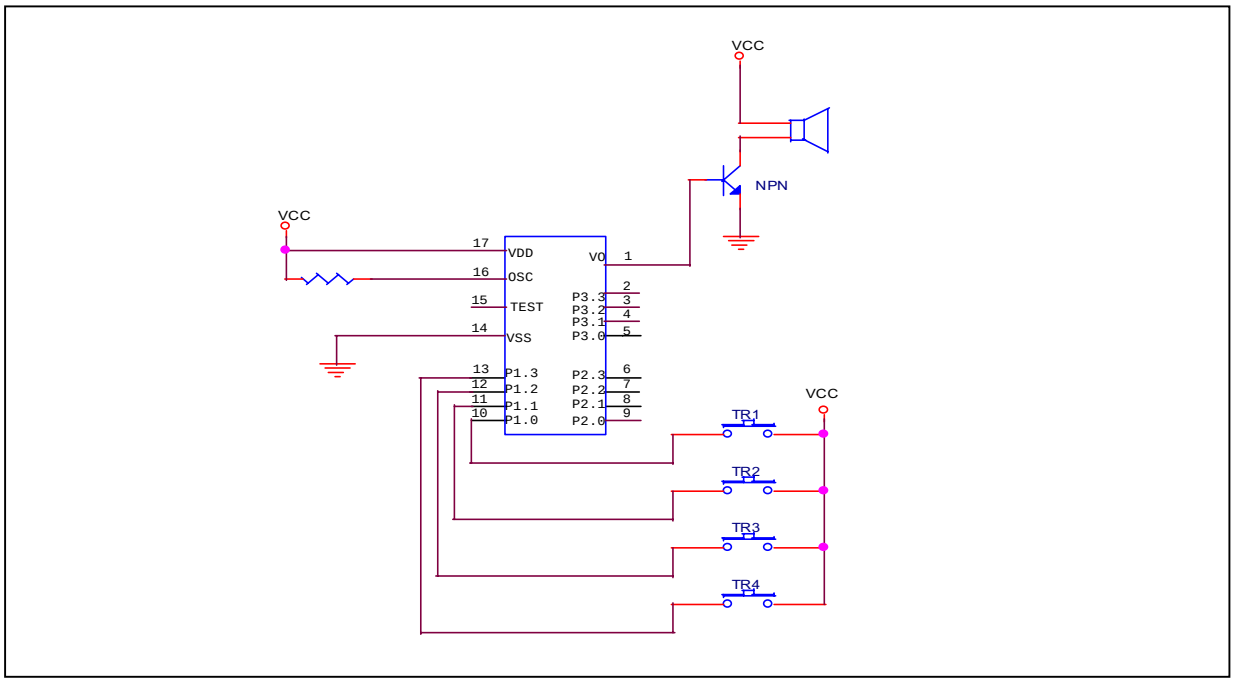
```
; Comments
EASY–n                          ; 1) Format identification
                                ;    (n=4/8/12/16/20)
Section_name1                   ; 2) Section definition
Section_name2
    ------------------
EM57100                         ; 3) Chip identification
M0:R(4)                         ; 4) Option selection
M1:R(5)
M2:R(16)
NO_STOP
TR_STATE                        ; 5) Trigger state definition
T_name1:
T_name2:
    ------------------
OUT_STATE                       ; 6) Output state definition for Port 3
O_name1:                        ;    and Port 2 (P3.3, P3.2, P3.1,
O_name2:                        P3.0,
    ------------------          ;    P2.3, P2.2, P2.1, P2.0)
PATH                            ; 7) Path definition
PATH0:
P_name1:
P_name2:
    ------------------
```

9.1.2.1 Format Identification

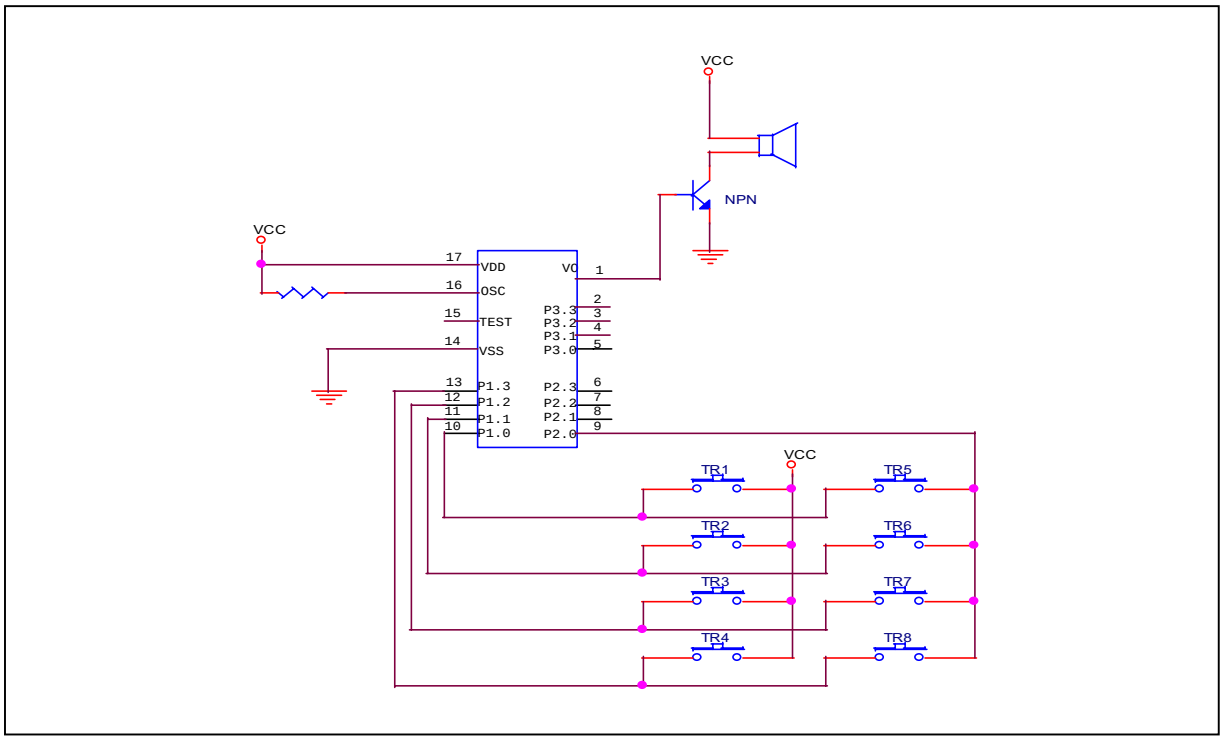
Identifies the program file format. "EASY-n" identifies the file as one of the following**:**

Format	Maximum Trigger Number	I/O Pin Available
Easy-4	4	P3.3~ P3.0; P2.3~ P2.0
Easy-8	8	P3.3~ P3.0; P2.3~ P2.1
Easy-12	12	P3.3~ P3.0; P2.3, P2.2
Easy-16	16	P3.3~ P3.0; P2.3
Easy-20	20	P3.3~ P3.0

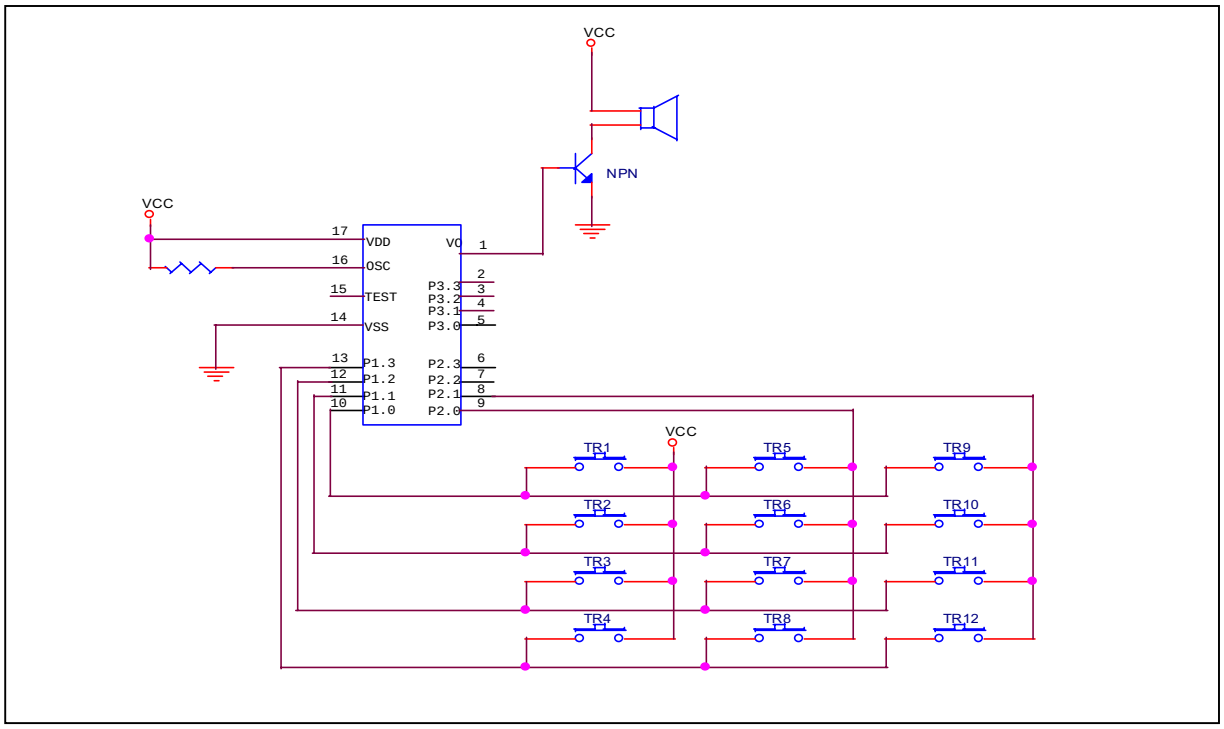
The application circuits for each format are illustrated below.



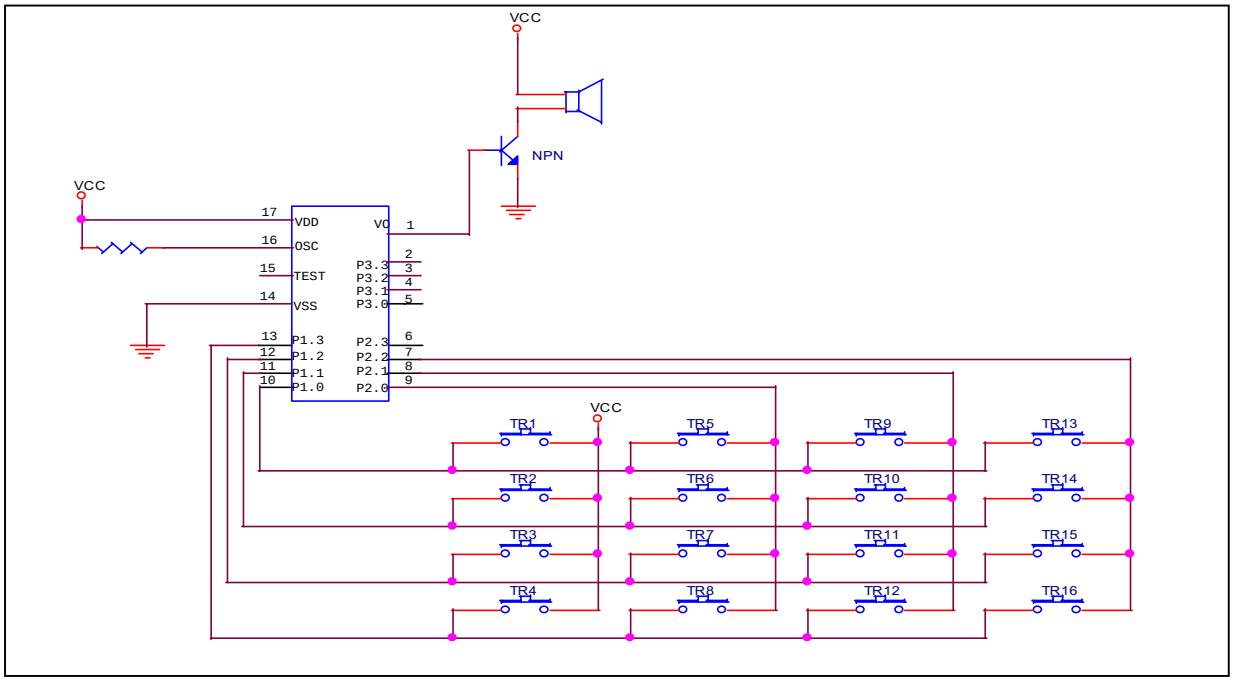
Easy-4 Application Circuit



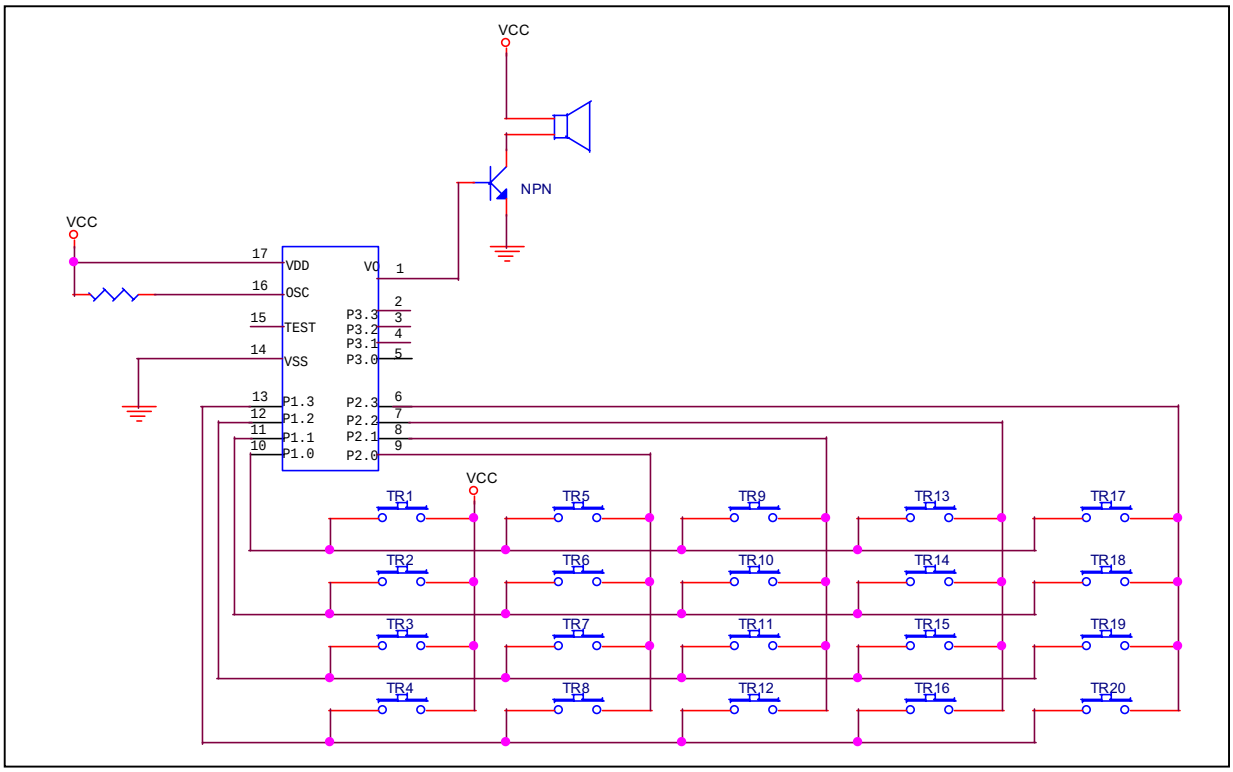
Easy-8 Application Circuit



Easy-12 Application Circuit



Easy-16 Application Circuit



Easy-20 Application Circuit

9.1.3 File Format with Outputs and Fixed Trigger Paths Assignment State (EASY-32)

The following shows the basic requirements of the "Easy-32" file format with outputs and fixed trigger paths assignment. Refer to Section 9.1.5 for the details of each section definition.

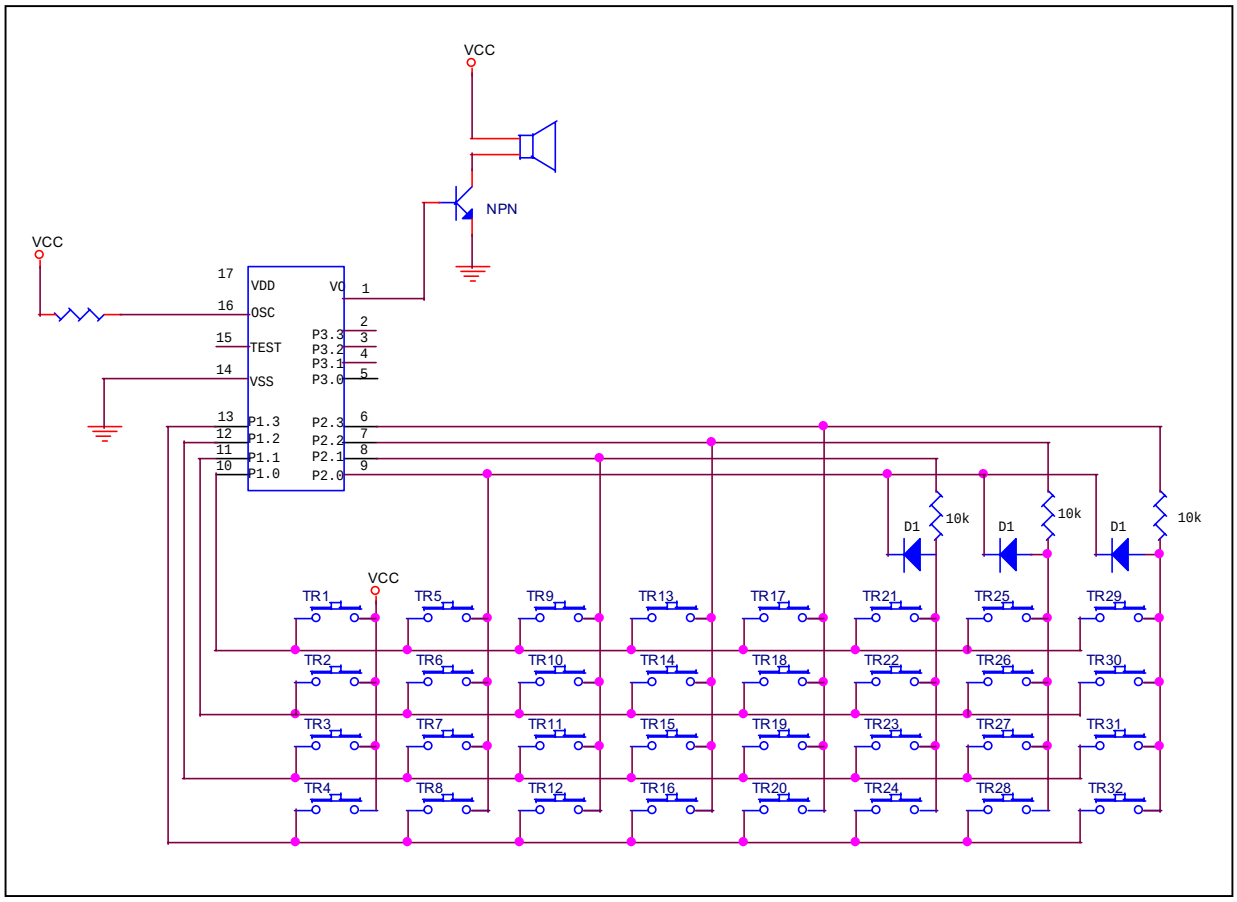
```
; Comments
EASY–32                          ; 1) Format identification
Section_name1                    ; 2) Section definition
Section_name2
    ------------------
EM57100                          ; 3) Chip identification
M0:R(4)                          ; 4) Option selection
M1:R(5)
M2:R(16)
NO_STOP
OUT_STATE                        ; 5) Output state definition for Port 3
O_name1:                         ;    (P3.3, P3.2, P3.1, P3.0)
O_name2:
    ------------------
PATH                             ; 6) Path definition
PATH0:
TR1:                             ; 7) TR1 to TR32 are system defined
TR2:                             ;    path names. When trigger inputs
    ------------------           ;    (Trigger 1 to Trigger 32) are
TR32:                            ;    pressed and a rising edge trigger
TRF:                             ;    is
                                 ;    enabled, the path equations with
                                 ;    the path names TR1 to TR32 are
                                 ;    executed. TRF is a system
                                 ;    defined
                                 ;    path name. When trigger is
                                 ;    released and a falling edge
                                 trigger
P_name1:                         ;    is enabled, the path equation
    ------------------           with
                                 ;    the path name TRF is executed.
                                 ; 8) User defined path names.
```

9.1.3.1 Format Identification

Identifies the program file format.

Format	Maximum Trigger Number	I/O Pin Available
Easy-32	32	P3.3~ P3.0

The application circuit is illustrated below.



Easy-32 Application Circuit

9.1.4 File Format with only Fixed Trigger Paths Assignment State (EASY-64/128)

The following shows the basic requirements of the "Easy-n" (n=64/128) file format with only fixed trigger paths assignment. Refer to Section 9.1.5 for the details of each section definition.

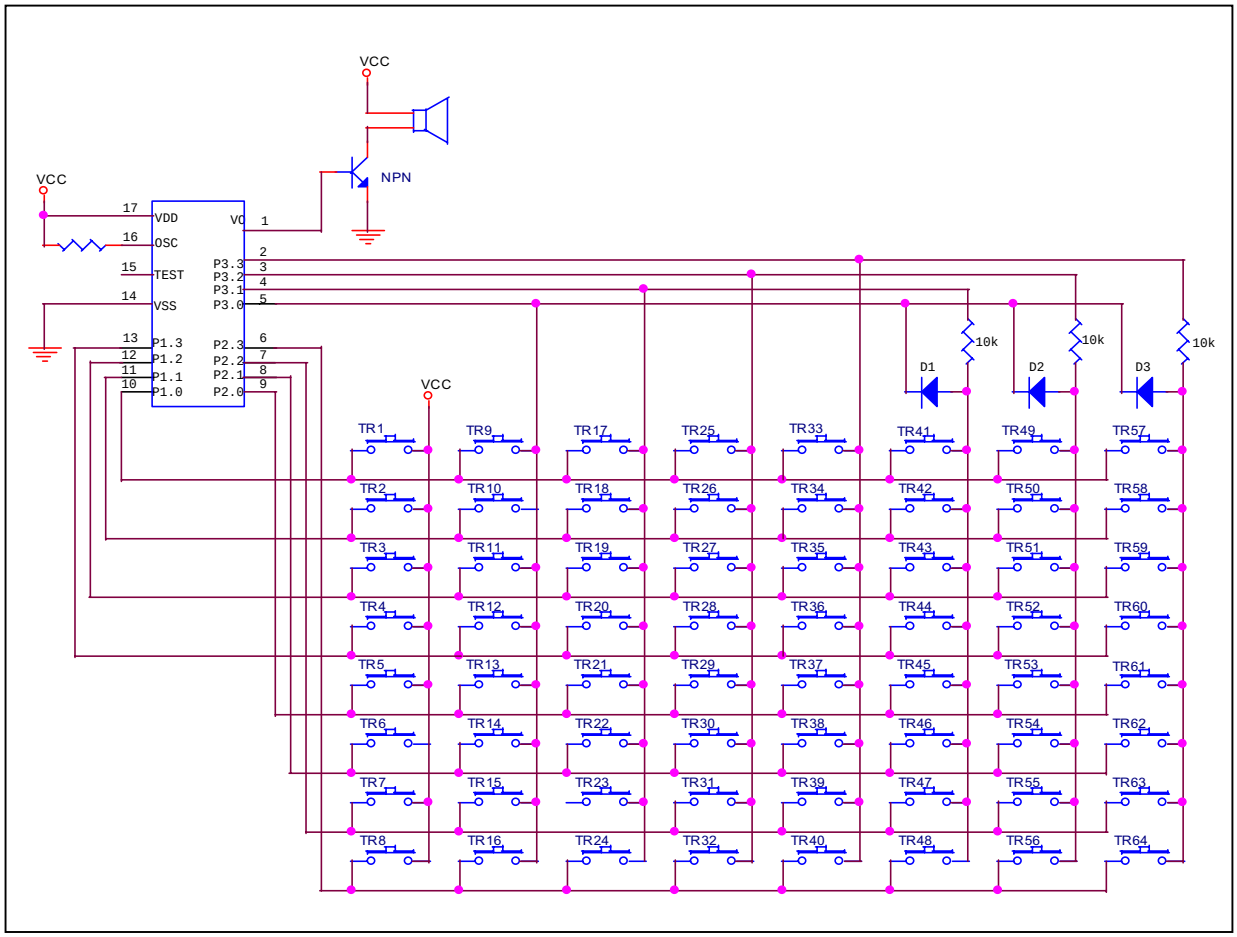
```
; Comments
EASY–n                          ; 1) Format identification
                                ;    (n=64/128)
Section_name1
Section_name2                   ; 2) Section definition
    ------------------
EM57100                         ; 3) Chip identification
M0:R(4)
M1:R(5)                         ; 4) Option selection
M2:R(16)
NO_STOP
PATH                            ; 5) Path definition
PATH0:
TR1:                            ; 6) TR1 to TRn are system defined
TR2:                            ;    path names. When trigger inputs
    ------------------          ;    (Trigger 1 to Trigger n) are
TRn:                            ;    pressed and a rising edge trigger
TRF:                            ;    is
                                ;    enabled, the path equations with
                                ;    the path names TR1 to TRn are
                                ;    executed ("n" can be 64 for
                                ;    EASY-64 format, 128 for
                                ;    EASY-128 format). TRF is a
                                ;    system defined path name. When
                                ;    trigger is released and a falling
                                ;    edge trigger is enabled, the path
                                ;    equation with the path name TRF
                                ;    is executed.
P_name1:
    ------------------          ; 7) User defined path names.
```

9.1.4.1 Format Identification

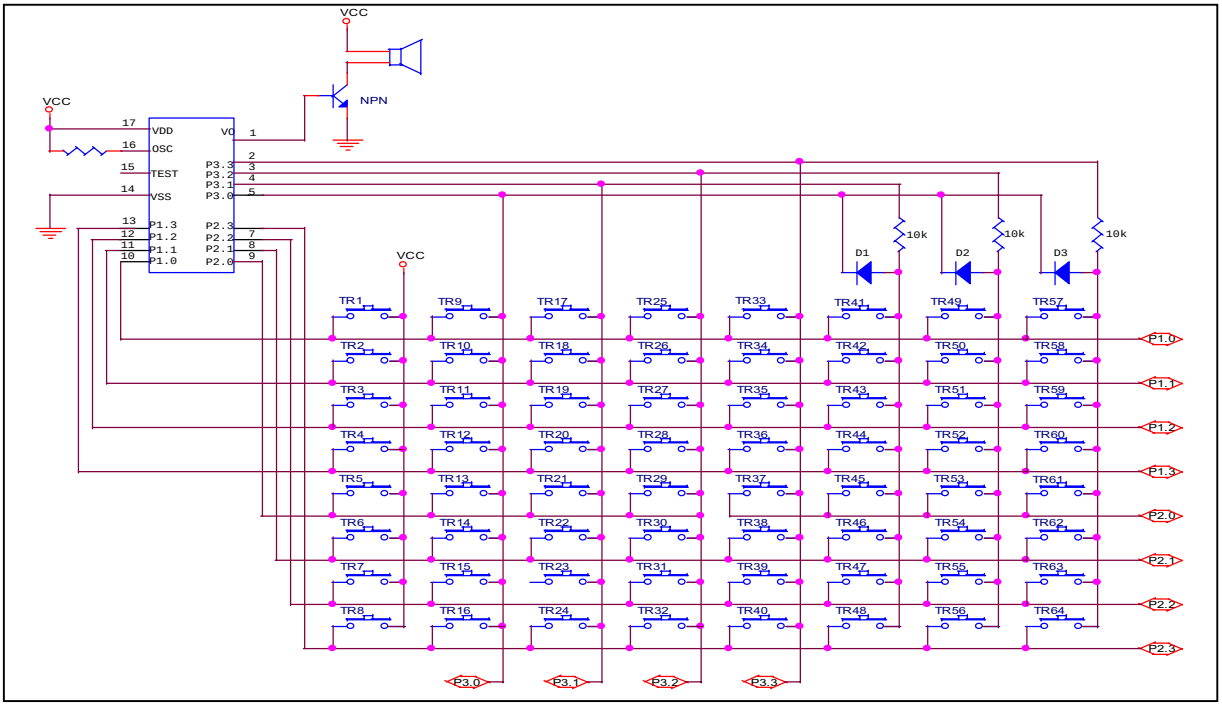
Identifies the program file format. "EASY-n" identifies the file as one of the following**:**

Format	Maximum Trigger Number	I/O Pin Available
Easy-64	64	None
Easy-128	128	None

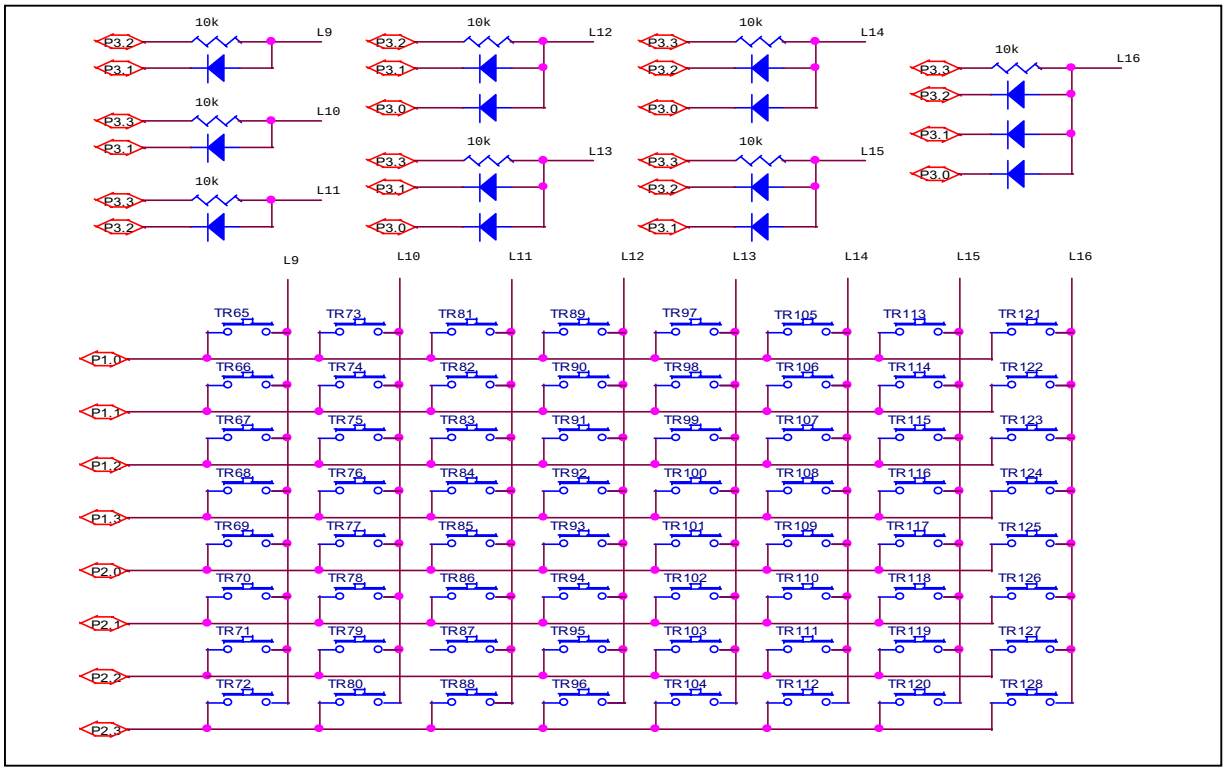
The application circuits for each format are illustrated below.



Easy-64 Application Circuit



Easy-128 Application Circuit (A)



Easy-128 Application Circuit (B)

9.1.5 Section Description for EASY, EASY-4/8/12/16/20, EASY-32 and EASY-64/128

The following describes the details of each section definition in each Easy file format.

9.1.5.1 Section Definition

Defines in this section, all the voice, melody, and sound effect files required for user's application.

9.1.5.2 Chip Identification

Defines appropriate type of EM57000 series chip that meets user's required application. This should be one of the following:

EM57100 (16Kx10 bits)	EM57200 (32Kx10 bits)	EM57300 (64Kx10 bits)
EM57400 (128Kx10 bits)	EM57500 (256Kx10 bits)	EM57600 (512Kx10 bits)
EM57700 (1024Kx10 bits)		

9.1.5.3 Option Selection

M0, M1, M2

Three 4-bit registers (M0, M1, and M2) are available for user's application. Each register may be defined as a general-purpose register or a random number. When option is defined as "**M0: R(n)**", M0 is a random number with range from 0 to n-1, where "n" can be 2 to 16.

When option is not defined, M0 is treated as a general-purpose register. This is also true to M1 and M2.

NO_STOP

By default, when trigger is received while the chip is playing, it is immediately stopped. If **NO_STOP** option is defined, playing will continue allowing user to do something else during playing.

9.1.5.4 I/O Pin State Definition for EASY

User can define a maximum of **256** I/O pins assignment statement, each under its own "state" name (i.e., S_name1, S_name2,... etc.). A statement is prefixed by a "state" name followed by 4 input and 8 I/O options of the corresponding I/O pins. The I/O options are separated by spaces or commas (,). These I/O options can be one of the following**:**

1. **P_name:** when the rising pulse happens at the corresponding I/O pin, the path equation with path name "P_name" is executed.

2. **/P_name:** when the falling pulse happens at the corresponding I/O pin, the path equation with path name "P_name" is executed.
3. **P_name1/p_name 2:** when the rising pulse happens at the corresponding I/O pin, the path equation with path name "P_name1" is executed; when the falling pulse happens at the corresponding I/O pin, the path equation with path name "P_name2" is executed.
4. **1:** Assign the corresponding I/O pin to "H".
5. **0:** Assign the corresponding I/O pin to "L".
6. **P+:** Send a pulse train beginning from "H" to the corresponding I/O pin.
7. **P-:** Send a pulse train beginning from "L" to the corresponding I/O pin.
8. **S+** Send a positive pulse to the corresponding I/O pin.
9. **S–** Send a negative pulse to the corresponding I/O pin.
10. **X:** No action of the corresponding I/O pin.

NOTE
1. ***Only "P_name", "/P_name", "P_name1/P_name2", and "X" can be provided for P1.0, P1.1, P1.2, and P1.3 because Port 1 is an input port.***
2. ***When user uses the Easy file format, the Port2 is connected to the 1M pull-down resistor (refer to Appendix A). The sink current of Port2 will be smaller.***
3. ***When the Easy file format powers on, if the I/O pin is immediately defined as "X" (don't care), the I/O pin will set to "H".***
4. ***During executing the Easy file format, if the I/O pin is defined as "X" (don't care), the I/O pin will be unchanged.***

9.1.5.5 Trigger State Definition for EASY-4/8/12/16/20

Trigger State is defined for each trigger state assignment. User can define up to **256** states maximum with different trigger state names, e.g., T_name1, T_name2,... etc.

Trigger state name must be followed by its corresponding trigger selection option "N", where "N" is the number 1 to 4 of the Easy-4 format, 1 to 8 of the Easy-8 format, …,or 1 to 20 of the Easy-20 format. Selection of options are separated by spaces or commas (,).

The trigger selection options can be one of the following:

1. **P_name** when the corresponding trigger is pressed, the path equation with path name "P_name" is executed.

2. **/P_name** when the corresponding trigger is released, the path equation with path name "P_name" is executed.

3. **P_name1/P_name2** when the corresponding trigger is pressed, the path equation with path name "P_name1" is executed; when the corresponding trigger is released, the path equation with path name "P_name2" is executed.

4. **X** disables the corresponding trigger (no action).

9.1.5.6 Output State Definition for EASY-4/8/12/16/20 and EASY-32

Define output state to set the output state of P3 and P2. User can define as many P3 and P2 output states as needed. Each should have different output state name, such as O_name1, O_name2,... etc.

These output state names are followed by the output settings of Port 3 and Port2 as follows**:**

P3 = [O3 O2 O1 O0], P2 = [O'3 O'2 O'1 O'0] for EASY-4/8/12/16

P3 = [O3 O2 O1 O0] for EASY-20/32

Where O3, O2, O1 and O0 stand for P3.3, P3.2, P3.1, and P3.0 respectively.

O'3, O'2, O'1 and O'0 stand for P2.3, P2.2, P2.1, and P2.0 respectively.

Each of the O3 to O0 and O'3 to O'0 elements may be defined to be one of the following**:**

9. **1** Define the corresponding I/O pin to be "H".

10. **0** Define the corresponding I/O pin to be "L".

11. **P+** Send a pulse train beginning from "H" to the corresponding I/O pin.

12. **P–** Send a pulse train beginning from "L" to the corresponding I/O pin.

13. **S+** Send a positive pulse to the corresponding I/O pin.

14. **S–** Send a negative pulse to the corresponding I/O pin.

15. **F** Define the designated I/O pins as tri_state. With EM57100~ EM57700 series, all Port 3/2 I/O pins can be set to tri_state under Easy-4 format, only Port 3, P2.3, P2.2, and P2.1 can do under Easy-8 format, only Port 3, P2.3, and P2.2 can do under Easy-12 format, and only Port 3 and P2.3 can do under Easy-16 format, only Port 3 can do under Easy-20/32 format.

16.**X** keep the corresponding output pin to be unchanged.

> *NOTE*
> ***When the Easy-n format powers on, if the output pin is immediately defined as "X" (don't care), the output pin will set to "H".***

Under EASY-n format, the pins of Port 2 are not configurable as they are used for keyboard scan matrix. Hence the following pins must be "x" (don't care):

EASY-8	EASY-12	EASY-16
P2.0 (O'0)	P2.0 (O'0), P2.1 (O'1)	P2.0 (O'0), P2.1 (O'1), P2.2 (O'2)

9.1.5.7 Path Definition

User can define as many path equations as needed and assign each with different path names, i.e., P_name1, P_name2,... etc. Each path equation line should contain no more than 255 characters (maximum). If a line is not enough in defining a path equation, additional lines may be added but the additional lines must begin with an ampersand (&) symbol to indicate continuation to previous line of the same path equation definition.

For EASY-32 and EASY-64/128, TR1, TR2, ... TRn, TRF are system defined path names and "n" is 32 for EASY-32, 64 for EASY-64, and 128 for EASY-128 format.

> *NOTE*
> ***The path equation is executed when trigger inputs Trigger 1 to Trigger n are pressed and the rising edge trigger is enabled. The path equation with path name "TRF" is executed when trigger is released and the falling edge trigger is enabled. The other path names, P_name1, P_name2,… etc., are user defined path name for special applications.***

Each path equation definition may contain one or more of the following parameters as user may require**:**

- **State name** **for EASY**
  such as "S_name1" to set the I/O state for EASY.
- **Tr_State name** **for EASY-4/8/12/16/20**
  such as "T_name1" to define the trigger state for EASY-4~20.
- **Out_State name** **for ESAY-4/8/12/16/20, EASY-32**
  such as "O_name1" to define the output state of Port3/ 2 for EASY-4~16 and Port3 for EASY-20/ 32.
- ***Out_State name/**
  [Out_State name] **for ESAY-4/8/12/16/20, EASY-32**

such as “O_name1” to define the output state of Port3/ 2 for EASY-4~16 and Port3 for EASY-20/ 32.

> *NOTE*
> ***When a section is playing, in order to change the Out_State and do not interrupt the playing by pressing the trigger. User must use this parameter and connect the “RETURN” parameter to return the section’s playing.***

- **Path name** such as “P_name1” to execute the path equation with the path name “P_name1”.
- **Section name(Sample rate)** play the voice/ melody/ sound effect file.“Sample rate” defines the playing speed of the voice file. It could be, either 4, 5,…, 22, or 4K, 5K,…, 22K, to represent 4k, 5k,…, 22k Hz playing speed respectively. Besides, it could be a real integer between 4000 and 32000 to represent playing speed (Hz).
- ***Section name(Sample rate)/ [Section name(Sample rate)]** play the voice/ melody/ sound effect file without calling the scan loop.
- **Section name(Sample rate)×n** play the voice/ melody/ sound effect file that is repeated by n times.

- ***Section name(Sample rate)×n/**
  [Section name(Sample rate)]×n play the voice/ melody/ sound effect file that is repeated by n times without calling the scan loop.
- **{Section name1(Sample rate)×n, [Section name2(Sample rate)]×m ….}×g** after playing the local looping, it plays global looping which is the looping of a combination of section names.
  g: the number of global looping of section name1, section name2.
  n: the number of local looping of section name1.
  m: the number of local looping of section name2 and so on.

NOTE

1. ***Section name can be replaced by #number according to the sequence of Section name in the section definition. The minimum number is "0".***
2. ***The all parameters in the path definition can be placed in the "{}".***

- **TEMPO=data** define the tempo for melody (beat/minute) or sound effect (ms/node) playing.
  "data" could be one of the following**:**

Melody	57	61	65	70	76	83	92	102
(beat/minute)	114	131	153	183	229	305	458	916

Sound effect	32ms	30ms	28ms	26ms	24ms	22ms	20ms	18ms
(ms/node)	16ms	14ms	12ms	10ms	8ms	6ms	4ms	2ms

- **VOL=data** set the voice/ melody/ sound effect playing volume in 8 steps increment. "data" is any value from 0 to 7 where 0 is the lowest and 7 the highest.
- **Delay(n)** delay "n" seconds. The minimum delay unit is 0.008 second. For example, Delay(0.2) means 0.2 second delay and Delay(8.6) means 8.6 seconds delay.
- **Rate=n** **for EASY, EASY-4/8/12/16/20, EASY-32** define the flash rate of flash output pins that are set to "P+" or "P-".

NOTE
"n" can be number 1 to 8 of the eight flash rate variables 16/n Hz. The parameter for EASY-64/128 format can not be used.

- **Mi=data/Mj** define the content of register "Mi" as data/ or Mj.

NOTE
Data can be 0, 1, 2, 3, 4,.....15,or #0000b,#0001b,.....#1111b

NOTE
For the following paragraphs, if DELAY(n) is not used, Mi, Mj and Mk can be defined as one of the following according to "EASY" and "EASY-n" format:

	EASY	EASY-4/8	EASY-12/16	EASY-20	EASY-32	EASY-64/128
EM57100~ EM57500	M0~ M15	M0~ M15	M0~ M13	M0~ M15	M0~ M22	M0~ M26
EM57600, EM57700	M0~ M111	M0~ M111	M0~ M109	M0~ M111	M0~ M118	M0~ M122

NOTE
Or if DELAY(n) is used, Mi, Mj and Mk can be defined as one of the following according to "EASY" and "EASY-n" format:

	EASY	EASY-4/8	EASY-12/16	EASY-20	EASY-32	EASY-64/128
EM57100~ EM57500	M0~ M12	M0~ M12	M0~ M10	M0~ M12	M0~ M19	M0~ M23
EM57600, EM57700	M0~ M108	M0~ M108	M0~ M106	M0~ M108	M0~ M115	M0~ M119

- **Mi=P1** **for EASY**
  define the content of register "Mi" as the content that is read from "P1".
- **Mi=P2** **for EASY, EASY-4/8/12/16**
  define the content of register "Mi" as the content that is read from "P2".

NOTE
According to different "EASY-n" format, the following bits of register "Mi" will be "0".

EASY-4	EASY-8	EASY-12	EASY-16
-	Mi(0)	Mi(0), Mi(1)	Mi(0), Mi(1), Mi(2)

- **Mi=P3** **for EASY, EASY-4/8/12/16/20, EASY-32** define the content of register "Mi" as the content that is read from "P3".
- **Mi=Mj+data** define the content of register "Mi" as the content of register "Mj" plus data.
- **Mi=Mj-data** define the content of register "Mi" as the content of register "Mj" subtract data.
- **Mi=Mj&data/ Mi=Mj.and.data** define the content of register "Mi" as the content of register "Mj" **AND** data.
- **Mi=Mj&Mk/ Mi=Mj.and.Mk** define the content of register "Mi" as the content of register "Mj" **AND** that of register "Mk".
- **Mi=Mj|data/ Mi=Mj.or.data** define the content of register "Mi" as the content of register "Mj" **OR** data.
- **Mi=Mj|Mk/ Mi=Mj.or.Mk** define the content of register "Mi" as the content of register "Mj" **OR** that of register "Mk".
- **Mi=Mj^data/ Mi=Mj.xor.data** define the content of register "Mi" as the content of register "Mj" **XOR** data.
- **Mi=Mj^Mk/ Mi=Mj.xor.Mk** define the content of register "Mi" as the content of register "Mj" **XOR** that of register "Mk".
- **Mi=~Mj/ Mi=.inv.Mj** define the content of register "Mi" as the **INVERT**ed content of register "Mj".
- **Mi(j)=1/0** define the "jth" bit of register "Mi" as 1/ or 0, where j can be 0 to 3.

- **P2=Mi/data** **for EASY, EASY-4/8/12/16**
  define the output content of port2 as the content of register "Mi" or data.

NOTE
3. For "Easy" format, user must take care for the status of trigger pins (in port2 or port3) to let trigger state take effect.   ***4. According to different "EASY-n" format, the some pins of port2 are used to be scanning pins, the following bits of register "Mi" and data will be ignored.***

EASY-4	EASY-8	EASY-12	EASY-16
---	Bit 0	Bit 0~1	Bit 0~2

- **P3=Mi/data** **for EASY, EASY-4/8/12/16/20, EASY-32**
  define the output content of port3 as the content of register "Mi" or data.

- **Mi?data:p_name** if the content of register "Mi" equals to data, the path equation with the path name "p_name" will be executed.
- **Mi?Mj:p_name** if the content of register "Mi" equals to that of register "Mj", the path equation with the path name "p_name" will be executed.
- **Mi(j)?1/0:p_name** if the "jth" bit of register "Mi" equals to 1/ or 0, the path equation with the path name "p_name" will be executed.

- **Mi?[d x d x]:[p_0 x p_2 …]** If Mi(3), Mi(1) equals to "00", the path equation with path name "p_0" will be executed. If Mi(3), Mi(1) equals to "01", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If Mi(3), Mi(1) equals to "10", the path equation with path name "p_2" will be executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

NOTE
1. "X" means "don't care".   ***2. For "Mi?[d x d x]", the parameters in "[ ]" can be "d" or "x", the decision of jump or not is depended on the corresponding bit of register "Mi" that is defined as "d".***

- **Mi:[p_0 x p_2 ...]** if the content of register "Mi" equals to "0", the path equation with path name "p_0" will be executed. If that of register "Mi" equals to "1", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If that of register "Mi" equals to "2", the path equation with path name "p_2" will be executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

- **Px.x?H/L:p_name for EASY**
  if the pin Px.x is "H"/ or "L", the path equation with path name "p_name" will be executed. Where "Px.x" is any pin of P1.0, P1.1,…, P3.3 under EASY format.

- **TRn?H/L:p_name for EASY-4/8/12/16/20**
  if the trigger TRn is "H"/ or "L", the path equation with path name "p_name" will be executed. Where "TRn" is any value from TR1 to TR4 under Easy-4 format, TR1 to TR8 under Easy-8 format,…, TR1 to TR20 under EASY-20 format.

- **TR?H:p_name for EASY-32, EASY-64/128**
  if any trigger is "H", the path equation with path name "p_name" will be executed.

- **TR?L:p_name for EASY-32, EASY-64/128**
  if all triggers are "L", the path equation with path name "p_name" will be executed.

- **P1?[I3 I2 I1 I0]:p_name for EASY**
  if the I/O pins of Port1 (P1.3, P1.2, P1.1, P1.0) is equal to [I3 I2 I1 I0], the path equation with path name "p_name" will be executed. Where I3 to I0 can be"1", "0", or "X".

- **P1?[d x d x]:[p_0 x p_2…..] for EASY**
  If P1.3, P1.1 equals to "00", the path equation with path name "p_0" will be executed. If P1.3, P1.1 equals to "01", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If P1.3, P1.1 equals to "10", the path equation with path name "p_2" will be

executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

> *NOTE*
> ***For "P1?[d x d x]", the parameters in "[ ]" can be "d" or "x", the decision of jump or not is depended on the corresponding pin of port1 that is defined as "d".***

- **P2?[I'3 I'2 I'1 I'0]:p_name for EASY, EASY-4/8/12/16**
  if the I/O pins of Port2 (P2.3, P2.2, P2.1, P2.0) is equal to [I'3 I'2 I'1 I'0], the path equation with path name "p_name" will be executed. Where I'3 to I'0 can be"1", "0", or "X".

> *NOTE*
> ***"X" means "don't care". The following parameters must be "X" according to different "EASY-n" format:***

EASY-4	EASY-8	EASY-12	EASY-16
-	I'0	I'1, I'0	I'2, I'1, I'0

- **P2?[d x d x]:[p_0 x p_2 …] for EASY, EASY-4/8/12/16**
  If P2.3, P2.1 equals to "00", the path equation with path name "p_0" will be executed. If P2.3, P2.1 equals to "01", the corresponding path meets the "X" (don't care), the equation will execute next parameter. If P2.3, P2.1 equals to "10", the path equation with path name "p_2" will be executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket "[ ]" is 16.

> *NOTE*
> ***1. For "P2?[d x d x]", the parameters in "[ ]" can be "d" or "x", the decision of jump or not is depended on the corresponding pin of port2 that is defined as "d".***
> ***2."X" means "don't care". The following parameters must be "X" according to different "EASY-n" format:***

EASY-4	EASY-8	EASY-12	EASY-16
-	P2.0	P2.1, P2.0	P2.2, P2.1, P2.0

- **P3?[I3 I2 I1 I0]:p_name for EASY, EASY-4/8/12/16/20/32**
  if the I/O pins of Port3 (P3.3, P3.2, P3.1, P3.0) is equal to [I3 I2 I1 I0], the path equation with path name "p_name" will be

executed. Where I3 to I0 can be“1”, ”0”, or ”X”.

- **P3?[d x d x]:[p_0 x p_2…..] for EASY, EASY-4/8/12/16/20/32**
  If P3.3, P3.1 equals to “00”, the path equation with path name “p_0” will be executed. If P3.3, P3.1 equals to “01”, the corresponding path meets the “X” (don’t care), the equation will execute next parameter. If P3.3, P3.1 equals to “10”, the path equation with path name “p_2” will be executed, so on and so forth. The maximum number of path names that can be accommodated in the bracket “[ ]” is 16.

> *NOTE*
> ***For ”P3?[d x d x]”, the parameters in “[ ]” can be “d” or “x”, the decision of jump or not is depended on the corresponding pin of port3 that is defined as “d”.***

- **PLAY?:p_name** if there is voice, melody or sound effect playing, the path equation with the path name “p_name” will be executed.
- **FV_ON** **for EASY, EASY-4/8/12/16/20/32**
  define pin P3.3 as LED output that will flash according to voice volume. If this option is used, P3.3 will be only treated as LED output and must set to an output purpose pin. If the option is not used, P3.3 will be treated as a general purpose I/O pin.

> *NOTE*
> ***If P3.3 is set to be an input pin, the input function will be disabled after FV_ON under EASY format. This parameter can not be used under EASY-64/128 format.***

- **FV_OFF** **for EASY, EASY-4/8/12/16/20/32**
  turn off LED output flash function.

> *NOTE*
> ***It is recommended that programmer turn off LED output flash function every time he defines a chip power down to further minimize power consumer during power down. This parameter can not be used under EASY-64/128 format.***

- **RETURN** return to the location where interruption occurred due to some other actions.

- **STOP** force to stop the currently playing voice, melody or sound effect.
- **END** power down the chip and stop the program.

> *NOTE*
> ***1. Path equation "PATH0" must be defined for power on execution.***
> ***2. Only when one of Port 1 or Port 2 pins gets a rising/ falling edge pulse that the EM57100~ EM57700 series chips will wake up from power down under EASY format.***
> ***3. Only when one of the trigger inputs gets a rising/ falling edge pulse that the EM57100~EM57700 series chips will wake up from power down under EASY-n format.***

- **MODE=[RF]** **for EASY-32, EASY-64/128**
  set the rising/falling trigger interrupt to enable or disable as follows**:**
  [RF]=[00] Both rising and falling triggers are disabled.
  [RF]=[01] Rising trigger is disabled. Falling trigger is enabled.
  [RF]=[10] Rising trigger is enabled. Falling trigger is disabled.
  [RF]=[11] Both rising and falling triggers are enabled.
- **DISABLE** **for EASY-32, EASY-64/128**
  is an internal path equation used to return to the location where an undesired interrupt occurred.

  When a trigger interrupt occurs and the interrupt is unwanted, just put the path name "DISABLE" or a conditional check "Mi?data:DISABLE" to back to the position where just be interrupted.

9.1.5.8 Summary of Parameters in Path Equation

The following table lists the parameters that can be used in the path equation.

Parameters	EASY	EASY-4~ 16	EASY-20	EASY-32	EASY-64/128
State name	Y	---	---	---	---
Tr_State name	---	Y	Y	---	---
Out_State name	---	Y	Y	Y	---
[Out_State name]	---	Y	Y	Y	---
Path name	Y	Y	Y	Y	Y
Section name(Sample rate)	Y	Y	Y	Y	Y
[Section name(Sample rate)]	Y	Y	Y	Y	Y
Section name(Sample rate) ×n	Y	Y	Y	Y	Y
[Section name(Sample rate)] × n	Y	Y	Y	Y	Y
{Section name1(Sample rate) ×n, [Section name2(Sample rate)] × m… } × g	Y	Y	Y	Y	Y
TEMPO=data	Y	Y	Y	Y	Y
VOL=data	Y	Y	Y	Y	Y
Delay(n)	Y	Y	Y	Y	Y
Rate=n	Y	Y	Y	Y	---
Mi=data/ Mj	Y	Y	Y	Y	Y
Mi=P1	Y	---	---	---	---
Mi=P2	Y	Y	---	---	---
Mi=P3	Y	Y	Y	Y	---
Mi=Mj+data/ Mj-data	Y	Y	Y	Y	Y
Mi=Mj.and.data/ Mj.and.Mk	Y	Y	Y	Y	Y
Mi=Mj.or.data/ Mj.or.Mk	Y	Y	Y	Y	Y
Mi=Mj.xor.data/ Mj.xor.Mk	Y	Y	Y	Y	Y
Mi=.inv.Mj	Y	Y	Y	Y	Y
Mi(j)=1/ 0	Y	Y	Y	Y	Y
P2=Mi/ data	Y	Y	---	---	---
P3=Mi/ data	Y	Y	Y	Y	---
Mi?data:p_name	Y	Y	Y	Y	Y
Mi?Mj:p_name	Y	Y	Y	Y	Y
Mi(j)?1/ 0:p_name	Y	Y	Y	Y	Y
Mi?[d x d x]:[p_0 x p_2…]	Y	Y	Y	Y	Y
Mi:[p_0 x p_2...]	Y	Y	Y	Y	Y

Px.x?H/ L:p_name	Y	---	---	---	---
TRn?H/ L:p_name	---	Y	Y	---	---
TR?H/ L:p_name	---	---	---	Y	Y
P1?[I3 I2 I1 I0]:p_name	Y	---	---	---	---
P1?[d x d x]:[p_0 x p_2…]	Y	---	---	---	---
P2?[I'3 I'2 I'1 I'0]:p_name	Y	Y	---	---	---
P2?[d x d x]:[p_0 x p_2…]	Y	Y	---	---	---
P3?[I3 I2 I1 I0]:p_name	Y	Y	Y	Y	---
P3?[d x d x]:[p_0 x p_2…]	Y	Y	Y	Y	---
PLAY?:p_name	Y	Y	Y	Y	Y
FV_ON	Y	Y	Y	Y	---
FV_OFF	Y	Y	Y	Y	---
RETURN	Y	Y	Y	Y	Y
STOP	Y	Y	Y	Y	Y
END	Y	Y	Y	Y	Y
MODE=[RF]	---	---	---	Y	Y
DISABLE	---	---	---	Y	Y

9.2 Examples for Easy Format® (ESY) Programs

The following examples demonstrate the Easy format programming capability. Comments of the program are followed by the ";" sign.

9.2.1 Example 1: ONE SHOT (port 1.0) – Irretriggerable in EASY Format

```
EASY

ONE.WAV    7K

EM57100

STATE
;Statename   P1.0      P1.1  P1.2  P1.3  P2.0  P2.1  P2.2…P3.3
STATE0:      PATH1     X     X     X     X     X     X…X
BUSY:        X         X     X     X     X     X     X…X

PATH
PATH0: STATE0 END
PATH1: BUSY ONE.WAV(8K) STATE0 END
```

9.2.2 Example 2: ONE SHOT (trigger 1) – Retriggerable by itself in EASY-4 Format

```
EASY-4

ONE.WAV    7K

EM57100

TR_STATE
;Statename    Tr1        Tr2   Tr3   Tr4
STATE0:       PATH1      X     X     X

OUT_STATE
OUT0: P3=[0 0 0 0] P2=[0 0 0 0]
OUT1: P3=[1 1 1 1] P2=[1 1 1 1]

PATH
PATH0: OUT0 STATE0 END
PATH1: OUT1 ONE.WAV OUT0 END
```

9.2.3 Example 3: ONE SHOT (trigger 1 to trigger 4) – Retriggerable by the other pins in EASY-8 Format

```
EASY-8


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57100


TR_STATE

;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8

STATE0:      P1    P2    P3    P4    X     X     X     X
STATE1:      X     P2    P3    P4    X     X     X     X
STATE2:      P1    X     P3    P4    X     X     X     X
STATE3:      P1    P2    X     P4    X     X     X     X
STATE4:      P1    P2    P3    X     X     X     X     X


OUT_STATE

OUT0: P3=[0 0 0 0] P2=[0 0 0 X]
OUT1: P3=[1 1 1 1] P2=[1 1 1 X]


PATH

PATH0: OUT0 STATE0 END
P1: OUT1 STATE1 ONE.WAV OUT0 STATE0 END
P2: OUT1 STATE2 TWO.WAV OUT0 STATE0 END
P3: OUT1 STATE3 THREE.WAV OUT0 STATE0 END
P4: OUT1 STATE4 FOUR.WAV OUT0 STATE0 END
```

9.2.4 Example 4: LEVEL HOLD (trigger 1 to trigger 4) – Repeated playing, Retriggerable by the other pins in EASY-12 Format

```
EASY-12


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57100


TR_STATE

;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8…Tr12

STATE0:      P1    P2    P3    P4    X     X     X     X X X X X
STATE1:      /P11  P2    P3    P4    X     X     X     X X X X X
STATE2:      P1    /P11  P3    P4    X     X     X     X X X X X
STATE3:      P1    P2    /P11  P4    X     X     X     X X X X X
STATE4:      P1    P2    P3    /P11  X     X     X     X X X X X


OUT_STATE

OUT0: P3=[0 0 0 0] P2=[0 0 X X]
OUT1: P3=[1 1 1 1] P2=[1 1 X X]


PATH

PATH0: OUT0 STATE0 END
P1: OUT1 STATE1 ONE.WAV P1
P2: OUT1 STATE2 TWO.WAV P2
P3: OUT1 STATE3 THREE.WAV P3
P4: OUT1 STATE4 FOUR.WAV P4
P11: OUT0 STATE0 END
```

9.2.5 Example 5: LEVEL HOLD (trigger 1 to trigger 4) – Unrepeated playing, Retriggerable by the other pins in EASY-20 Format

```
EASY-20


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57100


TR_STATE

;Statename   Tr1   Tr2   Tr3   Tr4   Tr5   Tr6   Tr7   Tr8…Tr20

STATE0:      P1    P2    P3    P4    X     X     X     X X X X…
STATE1:      /P11  P2    P3    P4    X     X     X     X X X X…
STATE2:      P1    /P11  P3    P4    X     X     X     X X X X…
STATE3:      P1    P2    /P11  P4    X     X     X     X X X X…
STATE4:      P1    P2    P3    /P11  X     X     X     X X X X…


OUT_STATE

OUT0: P3=[0 0 0 0]
OUT1: P3=[1 1 1 1]


PATH

PATH0: OUT0 STATE0 END
P1: OUT1 STATE1 ONE.WAV OUT0 STATE0 END
P2: OUT1 STATE2 TWO.WAV OUT0 STATE0 END
P3: OUT1 STATE3 THREE.WAV OUT0 STATE0 END
P4: OUT1 STATE4 FOUR.WAV OUT0 STATE0 END
P11: OUT0 STATE0 END
```

9.2.6 Example 6: LEVEL HOLD (trigger 1 to trigger 4) – Complete cycle, Retriggerable in EASY-32 Format

```
EASY-32


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57100


OUT_STATE

OUT0: P3=[0 0 0 0]
OUT1: P3=[1 1 1 1]


PATH

PATH0: OUT0 MODE=[10] END
TR1: OUT1 ONE.WAV TR?H:TR1 OUT0 END
TR2: OUT1 TWO.WAV TR?H:TR2 OUT0 END
TR3: OUT1 THREE.WAV TR?H:TR3 OUT0 END
TR4: OUT1 FOUR.WAV TR?H:TR4 OUT0 END
TR5: ……..
…….
TR32: …….
TRF: DISABLE
```

9.2.7 Example 7: LEVEL HOLD (trigger 1 to trigger 4) – Complete cycle, Irretriggerable in EASY-64 Format

```
EASY-64


ONE.WAV          7K
TWO.WAV
THREE.WAV
FOUR.WAV


EM57100


PATH
;M0: Key pressed status
PATH0: MODE=[10] END
TR1: M0=1 MODE=[01] ONE.WAV M0?0:PATH0 TR?H:TR1
&       PATH0
TR2: M0=1 MODE=[01] TWO.WAV M0?0:PATH0 TR?H:TR2
&       PATH0
TR3: M0=1 MODE=[01] THREE.WAV M0?0:PATH0 TR?H:TR3
&       PATH0
TR4: M0=1 MODE=[01] FOUR.WAV M0?0:PATH0 TR?H:TR4
&       PATH0
TR5: ……
…….
TR64: …….
TRF: M0=0 DISABLE
```

9.3 Writing an Infrared Ray Application Using Easy Format

9.3.1 IR Description for EM57000 Series

The infrared ray application is only provided for EM57100~ EM57700. If EM57100~ EM57700 is set up for the IR mode, the pin P3.2 (except EM57400) will generate 38k Hz square wave and the pin P1.3 will be treated as an IR receiver pin for Easy format.

The following shows the IR parameters in the path equation of easy program file format.

- **IRtx=Mi/data for EASY, EASY-4/8/12/16/20 (not for EM57400)**
  define the content of transmitting for IR as the content of register "Mi" or data.
- **Mi=IRrx for EASY, EASY-4/8/12/16/20**
  define the content of register "Mi" as the content of receiving for IR.

NOTE

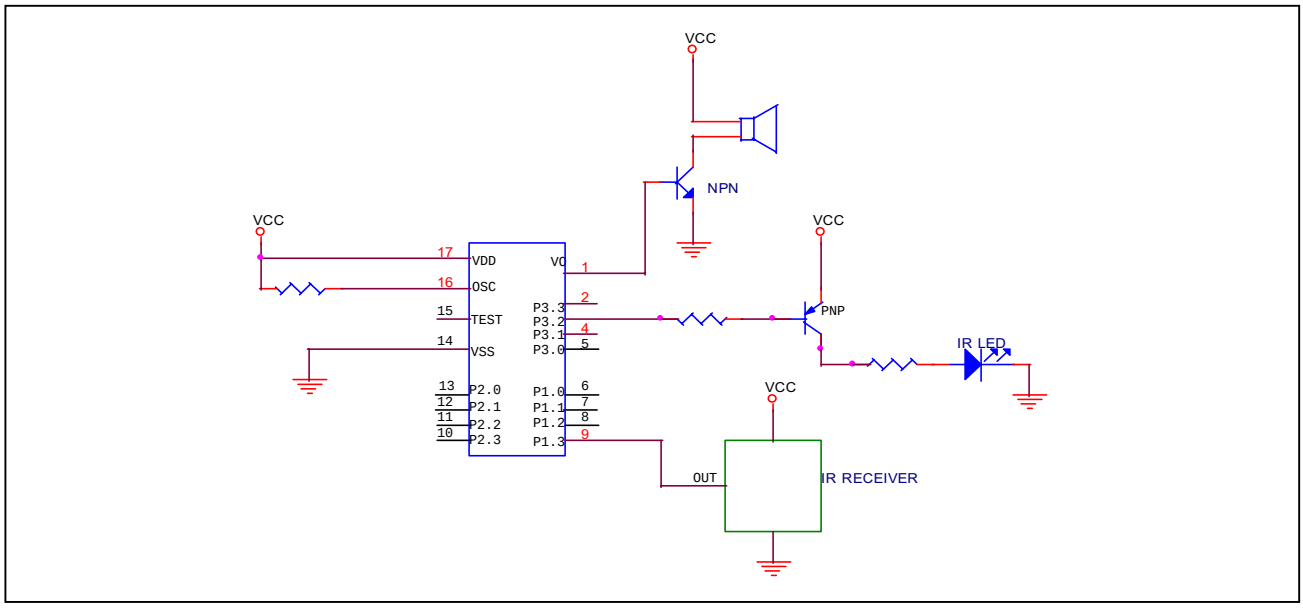
1. If EM57100~ EM57700 uses the IR receiver in EASY-n format, the pin P1.3 is treated as the receiver pin. Some trigger paths connecting the pin P1.3 must have the same path names in trigger state definition. The following shows these trigger paths in EASY-n format.

EASY-4	*EASY-8*	*EASY-12*	*EASY-16*	*EASY-20*
tr4	*tr4, tr8*	*tr4, tr8, tr12*	*tr4, tr8, tr12, tr16*	*tr4, tr8, tr12, tr16, tr20*

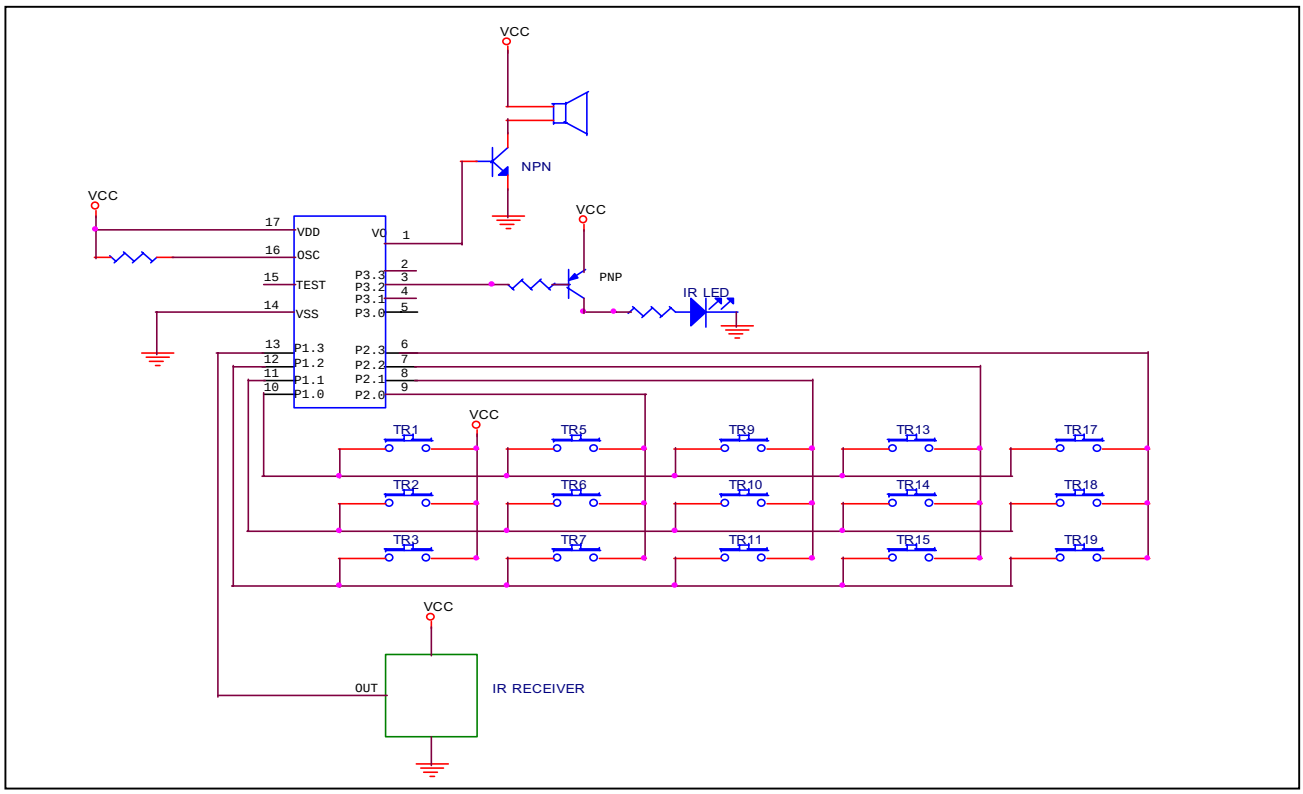
2. If EM57100~ EM57700 uses the IR receiver, the IR receiver path equation will be executed while the IR receiver pin receives the falling pulse in the sleep mode. Refer to section 9.3.2.

3. It is recommended that "Mi=IRrx" is the first parameter of the IR receiver path equation. Refer to section 9.3.2.

The transmitting and receiving application circuit for IR is illustrated as below. The IR application circuits for EASY-4/8/12/16 are similar to that of EASY-20. It is recommended that an I/O pin can be used to control the Vcc pin in the IR receiver module to prevent the leakage current.



IR application circuit for Easy



IR application circuit for Easy-20

9.3.2 Examples for IR Application in Easy Format (ESY) Programs

The following examples demonstrate the **IR** application in Easy format programming. Comments of the program are followed by the ";" sign.

9.3.2.1 Example 1: IR Program in EASY Format

```
EASY

0.WAV          8K

EM57500

STATE
;Statename: P1.0 P1.1 P1.2 P1.3 P2.0 P2.1 P2.2 P2.3 P3.0 P3.1 P3.2 P3.3
STATE0:      X    X    X    /Rx  X    X    X     X    X    X    X    Tx

PATH
PATH0: STATE0 END
Tx: M1=0 IRtx=M1 PATH0
Rx: M13=IRrx M13?0:P0 PATH0
P0: 0.WAV PATH0
```

9.3.2.2 Example 2: IR Program in EASY-20 Format

```
EASY-20

0.WAV    8K
1.WAV    8K
2.WAV    8K
lullab.mdy

EM57500

TR_STATE
;Statename: tr1 tr2 tr3 tr4 tr5 tr6 tr7 tr8 tr9 tr10 tr11 tr12 tr13 tr14 tr15 tr16 tr17 tr18 tr19
                tr20
STATE0:      tr1 tr2 tr3 /rx tr5 tr6 tr7 /rx tr9 tr10 tr11 /rx tr13 tr14 tr15 /rx tr17 tr18 tr19 /rx

OUT_STATE

PATH
PATH0: STATE0 END
tr1: m4=1 IRtx=m4 PATH0
tr2: IRtx=2 PATH0
tr3: lullab.mdy m1=3 IRtx=m1 PATH0
rx: m13=IRrx m13?1:p0 m13?2:p1 m13?3:p2 PATH0
p0: 0.wav PATH0
p1: 1.wav PATH0
p2: 2.wav PATH0
……
```

Appendix A

Sleep/Wake-up and Power Consumption of I/O Ports

A.1 Introduction

EM57000 provides flexible I/O functions. At power down (sleep) mode the current consumption is less than 1.0 μA at 3V. If I/O pins are not used properly, the standby current consumption will be a lot higher.

The following pages will describe I/O pins design, their characteristics, and the areas where attention is required to optimize their working condition.

For information on pin assignment and their electrical characteristics, refer to the next appendix (Appendix B).

A.2 I/O Status during Wake Up and Sleep Mode

When power is applied to the IC, the initialization sequence that takes 64ms is started. During the initialization period, the IC ignores the changes outside the I/O port.

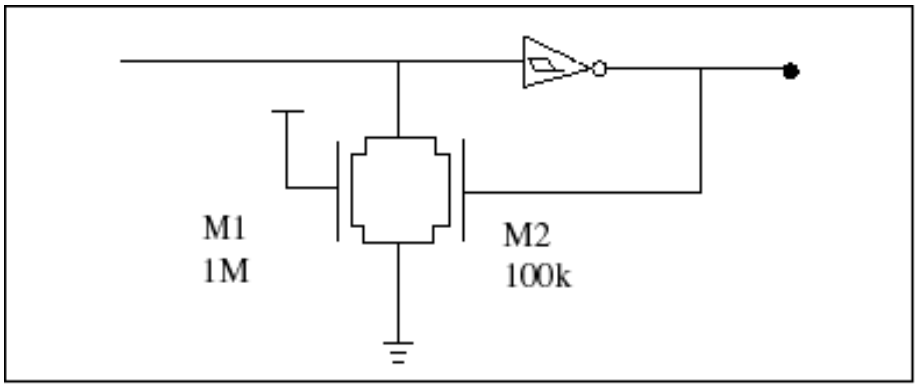
During power down (sleep) mode, if Port 1 or Port 2 (Port 2 only for EM57001) is triggered by a qualified rising or falling edge, the IC will wake up and starts to execute the program. This wake up process takes about 1 to 2 ms to complete. If the rising/falling edge trigger with the IC in the wake up mode, it has no effect to the program execution.

A.2.1 “KEYB” Instruction with EM57100~ EM57700

"KEYB" instruction, which is available only to EM57100~ EM57700; is used to suppress or ignore wake up function on Port 2. That is, when EM57100~ EM57700 series IC is in power down (sleep) mode, rising/falling edge trigger on Port 2 is ignored and the IC remains under sleep mode.

A.2.2 Port 1 Structure

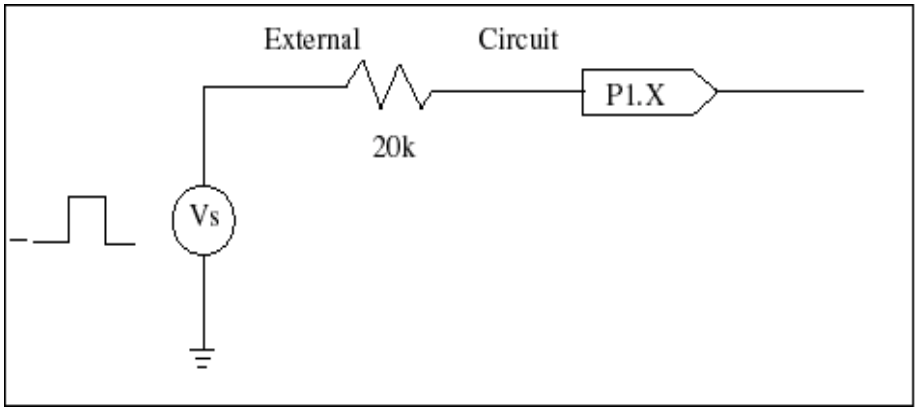
There are two pull-down resistors inside Port 1. Their values are 1M and 100k respectively. The structure is as illustrated below**:**



Port 1 Structure

When an "H" occurs, M2 is turned OFF.

Due to the Port 1 pull-down resistors, the external circuit output impedance should be smaller than 20k in order to fetch a higher signal.



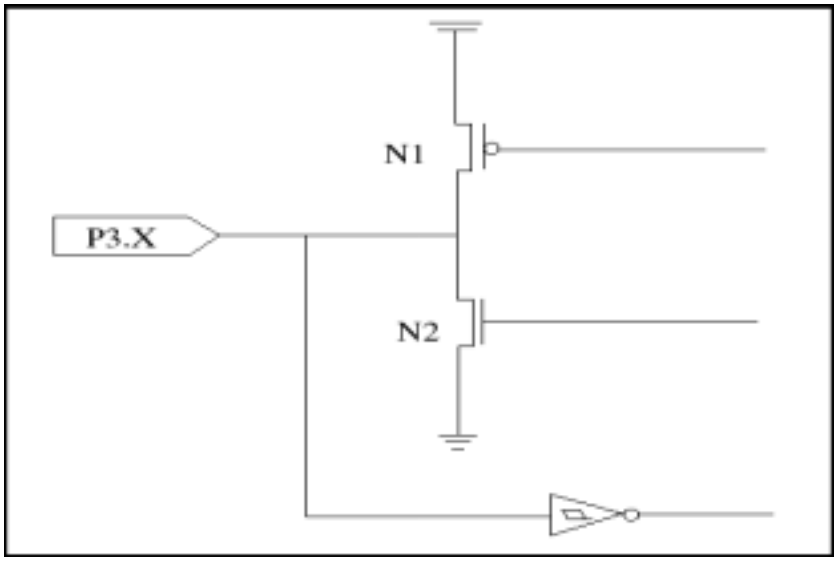
Port 1 Output Impedance External Circuit

A.2.3 Port 3 Structure

The Port 3 structure as shown below, contains an output circuit that is controlled by two instructions-

"MOV P3S, A"

"MOV P3, A"



Port 3 Structure

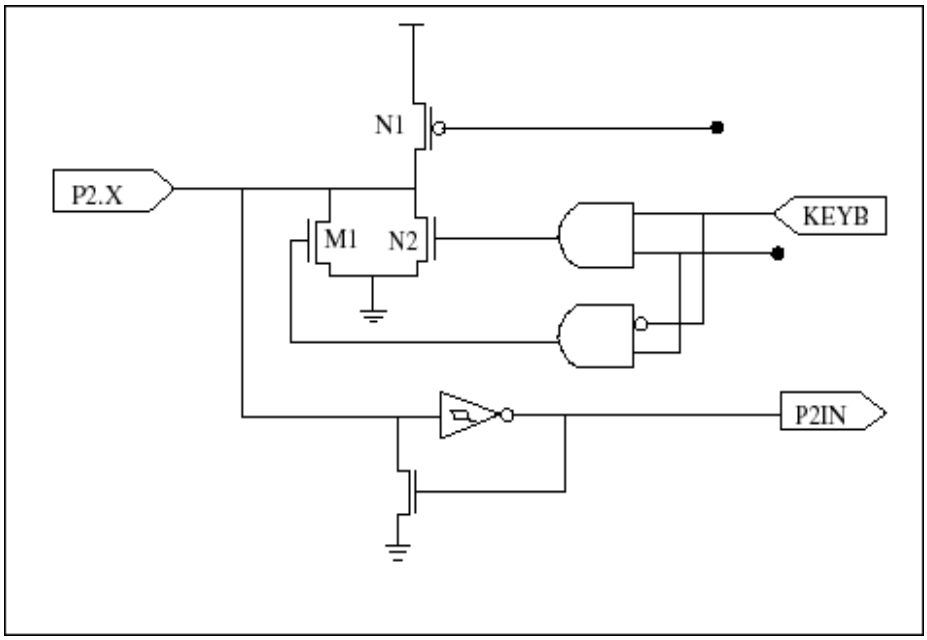
The result is as follows.

P3S	P3	P3.X
1	0	F
1	1	F
0	0	0
0	1	1

A.2.4 Port 2 Structure

The Port 2 structure is the combination of Port 1 and Port 3 structures. During power on, N2 is turned off. The output circuit is formed by N1 and M1. Under this condition, Port 2 behaves like Port 1. When "KEYB" is executed, M1 is turned off, while N1 and N2 form the output circuit. At this stage, Port 2 behaves close to the characteristics of Port 3.

When high impedance is required by the application circuit, M1 should be selected, as the impedance of N2 is lower. When higher sink current is required, N2 should be selected. Also, P2S can be set as 1 to turn off all of N1, N2, and M1 of P2.X.



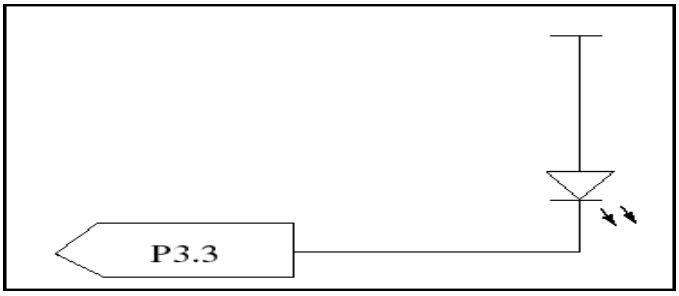
Port 2 Structure

A.2.5 Port 3.3 Flash-with-Volume Function

P3.3 is a special pin. It is built with a "Flash-with-Volume" function. The following instruction set turns On or Off this special function:

"**MOV A, #1**"
"**MOV MODE, A**" *(turn On)*

"**MOV A, #0**"
"**MOV MODE, A**" *(turn Off)*

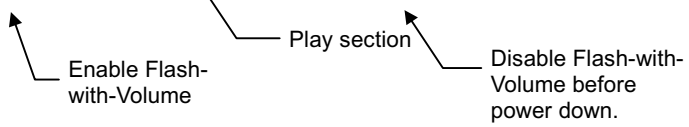


A.2.6 FV_ON, FV_OFF, and Output Setting in Easy Format

When “FV_ON” instruction is used, the “Flash-with-Volume” function is enabled. Remember to turn off this function (using "FV-OFF" command) before power down.

Example 1**:**

```
PATH1    FV_ON    #1       #2       FV_OFF    END
:
```



Example 2**:** Output setting (take note of the “F” instruction)

```
OUT1 : P3 = [    P+        X         F         0    ]
```

Where P3.1 is set as "F"(floating) and at the same time, the external circuit must provide a "0" or "1", and should not let it float for the IC to enter power down (sleep) mode.

A.2.7 Mode and Output in Assembly

After "Flash-with-Volume" is turned on, the following instructions must be executed before the "END" instruction:

```
"MOV     A,       #0"
"MOV     MODE,        A"    (turn off this Flash function)
"END"                       (then power down)
```

If P2S or P3S is set as "1", the external application circuit should not be floating.

Example**:**

```
"MOV     A,       #0100B"
"MOV     P3S,     A"        (P3.2 floating)
```

The above instruction will set pin P3.2 into the floating condition. Under this condition, the external circuitry should provide logic "0" or "1" before power down (preferably at "0" level). If the external application circuit does not provide "1" or "0" level and enters the power down state, higher standby current will occur.

A.2.8 External Circuit for Port 1

Due to the built in pull down resistor at Port1, the external application circuit should be floating or at "0" state before going into power down mode