MiniGUI User Manual

Version 3.0 (revised edition 5)
For MiniGUI Version 3.0.

Beijing FMSoft Technologies Co. Ltd.
Feb, 2018

Contents

Copyright Claim 1
1 Introduction to MiniGUI 1
1.1 A Brief Introduction 1
1.2 Documents for MiniGUI 2
1.3 MiniGUI Source Code and Samples 2
1.4 Optional Components of MiniGUI 3
1.5 miniStudio development tools 4
1.6 About this Manual...5
2 Configuring, Compiling, and Installing MiniGUI 6
2.1 Customization of Compiling Configuration Options...............................6
2.1.1 Configuration in GNU Development Environment by Configure Script......................7
2.1.2 Configuration under Non-GNU environment 11
2.2 Detailed Description on Compiling, Configuration Options of MiniGUI 11
2.2.1 Operating System Options and Macros...11
2.2.2 Target Board Related Options and Macros 12
2.2.3 Runtime Mode Related Options and Macros 13
2.2.4 Graphics Engine Related Options and Macros...14
2.2.5 Input Engine Related Options and Macros 16
2.2.6 Keyboard Layout Related Options and Macros..17
2.2.7 System Global Configuration Options and Macros 18
2.2.8 Character Set and Font Related Options and Macros 19
2.2.9 Image File Format Related Options and Macros..23
2.2.10 Appearance Style Related Options and Macros 24
2.2.11 Control Related Options and Macros 24
2.2.12 Other Options and Macros...25
2.3 Minimum Configuration Options...26
2.3.1 Using GNU Configure Script...26
2.3.2 Corresponding mgconfig.h 27
2.4 Compiling and Installing MiniGUI 37
2.4.1 compile and install the dependent library 37
2.4.2 compile and install the virtual framebuffer program 38
2.4.3 Compiling and Installing MiniGUI in the GNU Development Environment................39
2.4.4 Install MiniGUI Resource Package 40
2.4.5 compile and run MiniGUI sample..40
2.5 Compiling and Installing MiniGUI in Non-GNU Development Environment 40
2.6 Use Ubuntu on Windows to configure and compile MiniGUI........................43
3 MiniGUI runtime configuration options 44
3.1 Configuration File 45
3.1.1 Section system...45
3.1.2 Section fbcon 46
3.1.3 Section qvfb..46
3.1.4 Section pc_xvfb..46
3.1.5 Section rawbitmapfonts, varbitmapfonts, qpf, truetypefonts, and type1fonts...........47
3.1.5 Section systemfont 48
3.1.6 Section mouse 49
3.1.7 Section event...49
3.1.8 Section classic..50
3.1.9 Default Configuration File...52
3.2 Incore Configuration Options...60
3.2.1 Structure ETCSETCTION 60
3.2.2 ETC_S Structure...62
3.2.3 Listing of mgetc.c 62
3.3 Sample of Configuration 76
3.3.1 Runtime Configuration when only Support for ISO8859-1 Charset 76

3.3.2 Specifying Different Graphic Engine and Input Engine..76
4 Developing MiniGUI Application in Windows..78
Appendix A Frequent Ask Questions (FAQs)..80
A.1 Questions Relevant to GPL Versions 80
A.2 Questions Relevant to MiniGUI Application Fields......................................80
A.3 Questions Relevant to Portability 81
A.4 Questions Relevant to Compilation...81
A.5 Questions Relevant to Input Engines 82
A.6 Runtime Questions 83
A.7 Common Error Messages 83

1 Introduction to MiniGUI

1.1 A Brief Introduction

MiniGUI, developed by Beijing FMSoft Technologies Co. Ltd.[1], originates from a world famous free software project, which is initiated by Wei Yongming. MiniGUI aims to provide a fast, stable and lightweight graphics user interface (GUI) support system for real-time embedded systems. MiniGUI is "a cross-operating-system graphics user interface support system for embedded devices", and "an embedded graphics middleware". After over nine years of development since the end of 1998, MiniGUI has become a stable and reliable one for widespread application in a variety of products and programs; it can run on Linux/uClinux, eCos, VxWorks, pSOS, ThreadX, Nucleus, OSE, and even uC/OS-II, also on the Win32 platform.

MiniGUI defines a set of lightweight windowing and graphics interfaces for applications. Using these interfaces, an application can create multiple main windows and controls in them, such as buttons and edit boxes. MiniGUI provides powerful graphics functions for developers, helping to display all kinds of bitmaps and draw complicated graphics in windows.

However, the versions that you download freely from our site would be only used to develop GPL applications. If you are using MiniGUI for developing commercial applications or other software that are not covered by the terms listed in GPL, you should have a commercial license for MiniGUI from Feynman Software.

Currently, MiniGUI V3.0.x provides support for multi-process-based operating systems, like Linux; and provides support for traditional real-time embedded operating systems, which are multi-thread- or multi-task- based. The former provides support for the runtime modes MiniGUI-Processes and MiniGUI-Threads, and the later provides support for the runtime mode MiniGUI-Threads.

The official release of MiniGUI and its components source code package, sample package, etc., can be downloaded from the MiniGUI official website[2].

In addition, the complete source code of MiniGUI and its components is now hosted on GitHub, which contains the source code repository for the under development (yet to be released) MiniGUI and its components[3].

MiniGUI 3.0 is divided into a series of products according to the operating systems, please see Table 1.1. Table 1.1 also illustrates the runtime mode(s) provided by the products.

[1] FMSoft: http://www.fmsoft.cn

[2] http://www.minigui.com/en/download/

[3] https://github.com/VincentWei

Table 1.1 MiniGUI os and runtime modes supported

Products and versions	Runtime mode(s) supported
MiniGUI 3.0.x for Linux	MiniGUI-Processes MiniGUI-Threads MiniGUI-Standalone
MiniGUI 3.0.x for uClinux	MiniGUI-Threads MiniGUI-Standalone
MiniGUI 3.0.x for VxWorks	MiniGUI-Threads MiniGUI-Standalone
MiniGUI 3.0.x for ThreadX	MiniGUI-Threads MiniGUI-Standalone
MiniGUI 3.0.x for uC/OS-II	MiniGUI-Threads MiniGUI-Standalone

Except for the difference of runtime modes supported, these two versions have the almost same features.

For the detailed description about runtime modes and MiniGUI features, please refer to *MiniGUI Technology White paper for V3.0* and *Datasheet for MiniGUI V3.0*.

1.2 Documents for MiniGUI

Except for this manual, Feynman Software have provided the following documents available through the official website of MiniGUI download or visit[4]:

- *MiniGUI Programming Guide* Version 3.0-5. This guide describes in detail the foundation knowledge of MiniGUI on developing embedded application software, technical documents and development skills, the content of which involves various aspects of MiniGUI programming, include message looping, window procedure, dialog box, controls, graphics interfaces, and so on.
- Datasheet for MiniGUI V3.0.x. MiniGUI feature table.
- *MiniGUI API Reference Manual* for MiniGUI Version 3.0. This manual describes the APIs of MiniGUI V3.0.x (MiniGUI-Processes runtime mode) in detail[5].
- *MiniGUI Technology White paper for V3.0.*

MiniGUI developers also maintain a wiki[6] site to maintain the latest version of the above documents, please visit.

1.3 MiniGUI Source Code and Samples

In the download area of MiniGUI official website, the following MiniGUI source code package and sample package are listed:

[4] http://www.minigui.com/zhcn/documentation/

[5] Only English edition in HTML format and Windows CHM format

[6] http://wiki.minigui.com/twiki/bin/view

- MiniGUI Core Lib: libminigui-3.0.x-<os>.tar.gz, MiniGUI V3.0.x library source code for <os> (such as linux) operating system.
- MiniGUI Resources: minigui-res-3.0.x.tar.gz, the resources used by MiniGUI, including basic fonts, icons, bitmaps and mouse cursors.
- MiniGUI Samples: mg-samples-3.0.x.tar.gz, a sample program of "MiniGUI Programming Guide".

Also provide the following source code package:

- MiniGUI component.
- Tools and dependencies, virtual buffer program GVFB and freetype, libjpeg, libpng, zlib and other libraries.

1.4 Optional Components of MiniGUI

Except for the MiniGUI product, Feynman Software also provides some MiniGUI component products and other MiniGUI applications such as mSpider. Figure 1.1 shows the product line of Feynman Software.

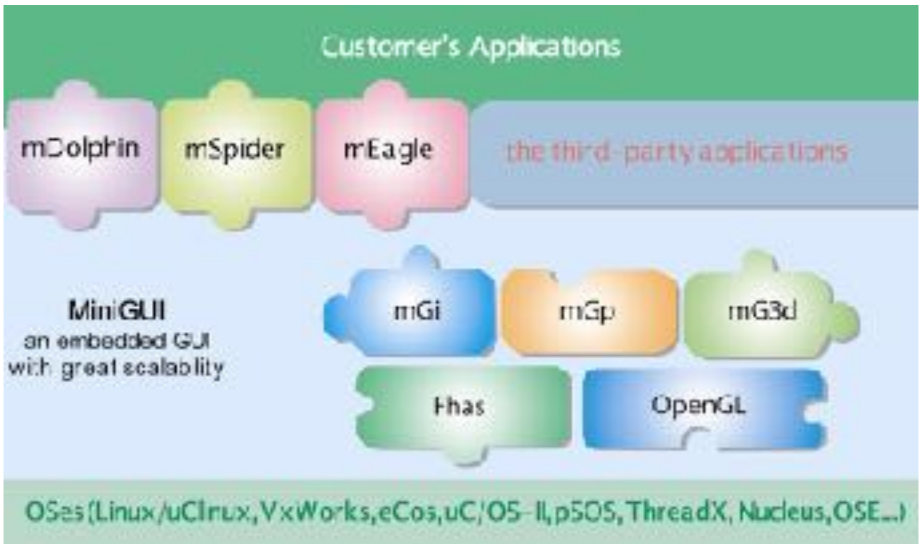


Figure 1.1 Product line of Feynman Software

mGUtils provides users with several functional templates that allow users to write code for commonly used functions.

mGPlus component is an extension and enhancement of the MiniGUI graphics rendering interface. It mainly provides support for 2D vector graphics and advanced graphics algorithms such as paths, gradient fills and color combinations.

mGEff provides an animation framework for MiniGUI applications. mGEff provides a large number of stable and efficient effects for developers to quickly flip, enlarge, scroll, pages and other commonly used animation has facilitated. In addition, mGEff can be combined with MiniGUI to develop an animation interface for the main window animation based on double buffering.

mGNCS - In the development of miniStudio, in order to the WYSWYG design of visual graphical interface, Feynman Software has developed a new set of controls based on the existing interface of MiniGUI. The new control set introduced by miniStudio is based on the original MiniGUI control set. It is distinguished from MiniGUI Intrinsic Control Set and is called "New Control Set". As a new MiniGUI Component mGNCS is released. mGNCS is mainly used with miniStudio. It can also be directly used as a component of MiniGUI 3.0 and can be mixed with the control of the centralized control. We strongly recommend that the new MiniGUI application be developed using mGNCS instead of the MiniGUI built-in control.

mGi provides input method framework for applications based on MiniGUI. mGi now provides the framework for soft-keyboard and hand writing input methods. mGi also provides an IME container for user to add self-defined IME to it. On the other hand, you can use self-defined keyboard bitmap for the soft-keyboard and add your self-defined translation method to it.

mGp provides a printing engine for applications based on MiniGUI so that applications using mGp will have the printing function. At present, mGp provides printing support for Epson, HP and some other printers. Note that mGp only provides the support for Linux operating system.

mG3d is a 3D rendering library for applications based on MiniGUI. By using this library, you can render 3D objects in your applications.

Except for these three component products above, Feynman Software also provides MiniGUI SDK for Win32. By using MiniGUI SDK for Win32, you can run MiniGUI and its applications on Win32 platform. You can even write and debug MiniGUI applications by using Visual Studio IDE tool. However, there are some limitations:

- MiniGUI SDK for Win32 only provides the support for the runtime MiniGUI-Threads.
- When you use MiniGUI SDK for Win32 to develop MiniGUI application, please do not invoke any function specific to Win32, because the function may not exist on your target operating system.

For the complete Feynman products, please visit the following web page:

```
http://www.minigui.com/en/download/
```

1.5 miniStudio development tools

MiniStudio is an integrated development environment for MiniGUI, providing users with WYSWYG interface design, automatic generation and maintenance of MiniGUI program framework, code editing, compiling, running and debugging based on Eclipse, speeding up the development of MiniGUI applications and reducing Use the threshold of MiniGUI. When using MiniGUI, users can focus more on the specific applications related to the business and greatly reduce the R & D costs of MiniGUI related applications and provide better products.

miniStudio is a non-open source commercial software product developed by Feynman Software. It provides two versions of Windows and Ubuntu Linux. You can visit

MiniGUI's official website to download the product for evaluation or trial license.

1.6 About this Manual

This manual mainly describes the compile-time configuration options and the runtime configuration options of MiniGUI.

2 Configuring, Compiling, and Installing MiniGUI

In general, Embedded Systems are special systems, and they have different requirement for graphics system. Some system required a basic graphics function but some one required a complete graphics, window and controls supporting. So an embedded graphics system must be constituted. MiniGUI provides a lot of configuration options. You can specify the functions of MiniGUI library. Generally, we can configure MiniGUI as follows:

- Specify the operating system and the target board on which MiniGUI runs.
- Specify MiniGUI running mode: MiniGUI-Threads base on thread, MiniGUI-Processes based on processes or the simple MiniGUI-Standalone.
- Specify the graphics engine and the input engine, as well as the options of these engines.
- Specify font class supported and the type of incore fonts.
- Specify the supporting character set.
- Specify the supporting image file format.
- Specify the supporting control class.
- Specify the style of the controls, i.e. CLASS style, FLAT style or FASHION style.

In this chapter we will discuss the compiling configuration options, in order that user can create a most suitable MiniGUI for their embedded system. We will discuss the compiling and installing of MiniGUI too.

2.1 Customization of Compiling Configuration Options

A file named **`mgconfig.h`** is located in the root directory of MiniGUI source code. A lot of ANSI C macros are defined in this file. We can configure MiniGUI by enabling or disabling these macros. Generally, we can modify this file in order to configure MiniGUI. You must recompile MiniGUI if this file is modified. After that you should install the header files and the libraries on your system. If your applications are static linking to MiniGUI, you should rebuild your applications, too. Please note that you should placed the **`mgconfig.h`** in a MiniGUI header file directory which your compiler can find it and overwrite the old one.

In general, the contents of **`mgconfig.h`** as the follows:

```
...

/* Define if compile for VxWorks operating system */
#define __VXWORKS__ 1

/* Define if include advanced 2D graphics APIs */
#define _MGHAVE_ADV_2DAPI 1

/* Define if support Arabic charset */
/* #undef _MGCHARSET_ARABIC */

/* Define if include the 2440 IAL engine */
/* #undef _MGIAL_2440 */

/* Define if include the automatic IAL engine */
/* #undef _MGIAL_AUTO */

/* Define if support BIG5 charset */
```

```
#define _MGCHARSET_BIG5 1

/* Define if include clipboard support */
#define _MGHAVE_CLIPBOARD 1

...
```

Above produces is a piece of **`mgconfig.h`**. Macro __VXWORKS__ is defined in this file and this macro will open the VxWorks support code in the MiniGUI source code. Macro _MGHAVE_CLIPBOARD is defined in this file, too. It will open the clipboard support code. Macro _MGIAL_AUTO is not defined in this file and MiniGUI will not support for Auto input engine.

The attention, in **`mgconfig.h`** also contains other some macro definitions, for instance MiniGUI version number and so on. Please maintain these macro definitions to be invariable; do not have voluntarily to revise these macro definitions.

The handwork revises **`mgconfig.h`** the procedure extremely tediously, moreover is easy to make a mistake. If you use the GNU development environment, then may use the configure script to configure MiniGUI. The following section introduces how to use the configure script automatically to produce the **`mgconfig.h`** file in the GNU development environment.

2.1.1 Configuration in GNU Development Environment by Configure Script

It's known that we can conveniently maintain the program package using makefile. Through makefile, we may compile, clean or install the function library, executable file and header files in the software package, etc. Although it is possible to organize a big project with makefile, it is not an easy job to create such a makefile manually. When we need to maintain a large-scale source code directory tree, the makefile maintenance work can greatly increase. Therefore, the Free Software Foundation's GNU project has developed the Autoconf/Automake tool for many software projects, which is based on the C language. Using this tool, we may automatically produce the makefile, and can check the system configuration information, which helps enhancement application software probability.

MiniGUI (MiniGUI library and sample programs package) is through the GNU Automake/Autoconf script organization. Therefore, if you use the GNU compatible development environment, for instance the Linux platform or Cygwin environment in Windows platform and so on, you may use MiniGUI's Automake/Autoconf configuration script to configure MiniGUI. Uses MiniGUI's Automake/Autoconf configuration script, certainly does not need to install Automake/Autoconf tool itself, but you just run the configure script in the MiniGUI source code package then to complete the configuration.
If you run the configure script, it can produce not only makefile, but also **`mgconfig.h`** file base on each of option in the configure script. Afterwards, we just need run make and make install commands to compile MiniGUI, and then MiniGUI library and header files will be installed to the directory, which you assigned.

[NOTE] The MiniGUI configure script only can be used in the GNU compatible development environment. The GNU compatible development environment usually has: the Linux system, the cygwin environment running on Windows and so on, It may apply to MiniGUI product version like Linux, uClinux, eCos.

There are lot of options in the MiniGUI configure script, and each configuration option corresponds a certain macro in **`mgconfig.h`**. If you enable an option when run configure, then the correspondence macro will be defined; otherwise can't define this macro. Run the following command.

```
user$ ./configure --help
```

You can obtain the whole options detailed list. For instance, supposing you use Ubuntu Linux 16.04(i386) as your development environment, the command runs in the MiniGUI source code directory and the running result as follows (this command output may have differently on other Linux release version):

```
`configure' configures this package to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.
```

```
$ ./configure --help
`configure' configures libminigui 3.0.13 to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...
...

System types:
  --build=BUILD     configure for building on BUILD [guessed]
  --host=HOST       cross-compile to build programs to run on HOST [BUILD]
  --target=TARGET   configure for building compilers for TARGET [HOST]

Optional Features:
  --disable-option-checking  ignore unrecognized --enable/--with options
  --disable-FEATURE       do not include FEATURE (same as --enable-FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --enable-silent-rules   less verbose build output (undo: "make V=1")
  --disable-silent-rules  verbose build output (undo: "make V=0")
  --enable-shared=PKGS  build shared libraries default=yes
  --enable-static=PKGS  build static libraries default=yes
  --enable-fast-install=PKGS  optimize for fast installation default=yes
  --enable-dependency-tracking
                          do not reject slow dependency extractors
  --disable-dependency-tracking
                          speeds up one-time build
  --disable-libtool-lock  avoid locking (might break parallel builds)
  --enable-debug           build with debugging messages <default=no>
  --enable-tracemsg        trace messages of MiniGUI <default=no>
  --enable-msgstr          include symbol name of message <default=no>
  --enable-procs           build MiniGUI-Processes version <default=no>
  --enable-standalone      build MiniGUI-Standalone version <default=no>
  --enable-incoreres       use incore resource instead file IO to initialize MiniGUI
<default=no>
  --enable-miniguientry    use minigui_entry function in MiniGUI <default=no>
  --enable-fixedmath       include fixed math routines <default=yes>
  --enable-dblclk          mouse button can do double click <default=yes>
  --enable-cursor          include cursor support <default=yes>
  --enable-clipboard       include clipboard support <default=yes>
  --enable-ownstdio        use own implementation of stdio functions <default=no>
  --enable-ownmalloc       use own implementation of malloc functions <default=no>
  --enable-ownpthread      use own implementation of pthread functions <default=no>
  --enable-adv2dapi        include advanced 2D graphics APIs <default=yes>
  --enable-minimalgdi      build a minimal GDI library only <default=no>
  --enable-productid       insert a productid into the library file
            <default=no>
  --enable-splash          enable splash <default=yes>
  --enable-screensaver     enable screensaver <default=yes>
  --enable-flatlf          include flat Look and Feel renderer <default=yes>
  --enable-skinlf          include skin Look and Feel renderer <default=yes>
...

Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-gnu-ld           assume the C compiler uses GNU ld default=no
  --with-pic              try to use only PIC/non-PIC objects default=use both
  --with-ttfsupport=ft1/ft2/none
...

Some influential environment variables:
  CC          C compiler command
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  LIBS        libraries to pass to the linker, e.g. -l<library>
  CPPFLAGS    (Objective) C/C++ preprocessor flags, e.g. -I<include dir> if
              you have headers in a nonstandard directory <include dir>
  CPP         C preprocessor

Use these variables to override the choices made by `configure' or to help
it to find libraries and programs with nonstandard names/locations.

Report bugs to the package provider.  --with-style=classic/flat/fashion
--with-ttfcachesize=64/128/256/512/1024
--with-mttfcachenum=10/20/40

Some influential environment variables:
  CC          C compiler command
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  CPPFLAGS    C/C++ preprocessor flags, e.g. -I<include dir> if you have
              headers in a nonstandard directory <include dir>
  CPP         C preprocessor

Use these variables to override the choices made by `configure' or to help
it to find libraries and programs with nonstandard names/locations.
```

Above these parameters were already configured parameter which established in the configure script, and these parameters are allowed to control which function codes were supported when compile MiniGUI. For example, run:

```
user$ ./configure --enable-procs --with-ttfsupport=ft2
```

You may configure MiniGUI that is the Freetype2 Truetype font support and the MiniGUI-Process runtime mode. If you run:

```
user$ ./configure --disable-cursor --disable-screensaver
```

Then configure MiniGUI that is disable the cursor and default screen saver function.

`./configure` command will produce a Makefile with default configuration options. Each compiling configuration option has provided a default setting in its explanation: <**`default=yes`**> (Expressed this compiling configuration option is enabled default) or

<default=no> (Expressed this compiling configuration option is disabled default).

Besides the MiniGUI has defined configuration options, the configure script also has some important general compiling configuration options.

1) Prefix Option

This compiling configuration option assigns the MiniGUI library where to install. The default installation path is **/usr/local**. If you run:

```
user$ ./configure --prefix=/home/test
```

After executing 'make install' command, the function library, header files and reference document are installed in **/home/test/lib**, **/home/test/include** and **/home/test/man** directory.

2) Cross Compiling Option

The compiling configuration options **--build**, **--host** and **--target** are very important to cross compile applications. For example, if you use the arm-linux cross compiling toolchain, then you may assign option like **--build**, thus let the configure script produces the makefile file used to complete the arm-linux's cross compiling:

```
user$ CC=arm-linux-gcc ./configure --prefix=/usr/local/arm/2.95.3/arm-linux/ \
    --build=i386-linux \
    --host=arm-unknown-linux \
    --target=arm-unknown-linux
```

In above command, the **--prefix** option is used to set the installing MiniGUI configuration file, the function library and the header files directory's prefix, when you executed **make install** command, MiniGUI configuration file, the library file and header files will be installed in the following position:

- **/usr/local/arm/2.95/arm-linux/etc/**
- **/usr/local/arm/2.95.3/arm-linux/lib/**
- **/usr/local/arm/2.95.3/arm-linux/include/**

3) --enable-static and --enable-shared

The two configuration options assign whether generating static function library or dynamic function library. If you do not need to produce the static library, then you may use the **--disable-static** configuration option, it will take less time to compile the library than default.

There are several predefined targets in the makefile, which produced by the configure script supply for user, here only several summaries as follow:

The function storehouse, a document and so on are installed in the way, which assigns
- make all: Produce the target set. Only run make command also to be allowed, this time can start to compile the source code, then link it and produces the executable file or function library.
- make clean: Clean the previous object files(*.o).
- make install: Install the function library, header files and so on to the directory which you assigned.

2.1.2 Configuration under Non-GNU environment

A majority of traditional embedded operating system supported by MiniGUI, user usually can use the integrated development environment running on Windows platform, such as Tornado, ADS, etc. Because these environment provide the development tools chain that is not GNU compatible, therefore, we are unable to use the configure script that is described in section 2.1.1 to produce makefile and the **mgconfig.h** file automatically. In this kind of situation, we need voluntarily to revise the **mgconfig.h** file to complete the MiniGUI compiling configuration. Fortunately, Feynman Software already prepares the **mgconfig.h** file for the majority operating system, which can directly be used (store in MiniGUI source code **build/** directory); moreover Feynman Software also prepared the corresponding development environment project file. You may directly manually revise the **mgconfig.h** file based on these project environments, and compile the MiniGUI library. For more detail information, please refer to the section 2.4.2.

2.2 Detailed Description on Compiling, Configuration Options of MiniGUI

In this chapter, we will give detailed description on all compiling, configuration options of MiniGUI. MiniGUI has many compiling, configuration options, for your actual demand; you can combine these options to generate MiniGUI function library.

In GNU development environment, we implement the most of configuration options of MiniGUI that based on **--disable-FEATURE** and **--enable-FEATURE**, while MiniGUI configuration script also provides **--with-configuration** option, you can use this configuration option to choose one option from multiple specified configuration. For example, you can use **--with-style** configuration option to specify the style of window and control of MiniGUI. Finally, these configuration options were defined macros, whatever use **--disable-FEATURE** or **--enable-FEATURE** or **--with-configuration** option to specify configuration option.

In the next chapter, we will give configuration option of MiniGUI by classify. We will description on configuration names of configure script and macro names in the **mgconfig.h** file.

2.2.1 Operating System Options and Macros

MiniGUI provides support for multiple operating systems, you can specify operating system when execute configure script, default operating system is Linux. If you want to run MiniGUI on uClinux, you can execute command as the follow:

```
user$ ./configure  --with-osname=uclinux
```

If you specify an operating system, the corresponding macro was defined in **mgconfig.h**. For some operating systems, we will open other some macros. Table 2.1 lists relevant options and macros of operating systems.

Table 2.1 operating systems relevant options and macros

Configuration options	Macro	Other relevant macro	Memo

--with-osname=linux	__LINUX__		Default value, for Linux operating system
--with-osname=uclinux	__uClinux__		For uClinux operating system
--with-osname=ecos	__ECOS__	__NOUNIX__	For eCos operating system
--with-osname=ucos2	__UCOSII__	__NOUNIX__ _INCORE_RES _USE_OWN_MALLOC _USE_OWN_STDIO _USE_OWN_PTHREAD	For uC/OS-II operating system
--with-osname=swlinux	__WINBOND_SWLINUX__		For SWLinux operating system, mutation of uClinux operating system
--with-osname=vxworks	__VXWORKS__	__NOUNIX__ _USE_OWN_STDIO _USE_OWN_PTHREAD	For VxWorks operating system
--with-osname=cygwin	__CYGWIN__	__NOUNIX__	For cygwin environment
--with-osname=win32	WIN32	__NOUNIX__	For Win32 platform
--with-osname=darwin	__DARWIN__	__NOUNIX__	For MacOS X operating system
--with-osname=threadx	__THREADX__	__NOUNIX__ _INCORE_RES _USE_OWN_MALLOC _USE_OWN_STDIO _USE_OWN_PTHREAD	For ThreadX operating system
--with-osname=nucleus	__NUCLEUS__	__NOUNIX__ _INCORE_RES _USE_OWN_MALLOC _USE_OWN_STDIO _USE_OWN_PTHREAD	For Nucleus operating system
--with-osname=ose	__OSE__	__NOUNIX__ _INCORE_RES _USE_OWN_PTHREAD	For OSE operating system
--with-osname=psos	__PSOS__	__NOUNIX__ _INCORE_RES _USE_OWN_PTHREAD	For pSOS operating system

According to operating system, we divide MiniGUI value-added release, so the MiniGUI value-added release product for certain operating system cannot run on anther operating system. In order to run MiniGUI value-added release product on corresponding operating system, you make sure that the above macros were defined when you modify configuration.

2.2.2 Target Board Related Options and Macros

In MiniGUI certain codes are related with a special target board; if you want run MiniGUI must on these target boards correctly, you need to assign the name of these development boards. When you run configure script, through the **`--with-targetname`** option, may assign the special target board name and the default name is unknown. The target board related options usually use for assign the sub-driver of graphics engine when MiniGUI uses the Shadow graphics engine or the CommLCD graphics engine, in other words, when uses these two engines, through the target board name you can determine which sub-driver contains. The table 2.2 lists the target board related options and macros.

Table 2.2 target board related options and macros

Configuration options	Macro	Memo
--with-targetname=stb810	__TARGET_STB810__	Philips STB810 development board base on Linux
--with-targetname=vfanvil	__TARGET_VFANVIL__	VisualFone development board base on ThreadX
--with-targetname=vxi386	__TARGET_VXI386__	i386 target base on VxWorks
--with-targetname=qvfb	__TARGET_QVFB__	Include qvfb sub-driver of Shadow engine base on Linux
--with-targetname=wvfb	__TARGET_WVFB__	Include wvfb sub-driver of Shadow engine base on Windows
--with-targetname=fbcon	__TARGET_FBCON__	Include fbcon sub-driver of Shadow engine base on Linux
--with-targetname=mx21	__TARGET_MX21__	MX21 development board base on OSE
--with-targetname=c33l05	__TARGET_C33L05__	Epson C33L05 development board base on axLinux
--with-targetname=bfin	__TARGET_BLACKFIN__	BlackFin537 development board base on uClinux
--with-targetname=vxppc	__TARGET_PPC__	PowerPC target base on VxWorks
--with-targetname=monaco	__TARGET_MONACO__	monaco development board base on Nucleus
--with-targetname=unkown	__TARGET_UNKNOWN__	Unknown development board: default value

2.2.3 Runtime Mode Related Options and Macros

We can configure MiniGUI as one of three kind of runtime mode: MiniGUI-Processes runtime mode base on multi-processes, MiniGUI-Threads runtime mode base on multi-thread, as well as MiniGUI-Standalone runtime mode base on non-multi-processes also non-multi-thread. MiniGUI-Threads runtime mode is the default mode when MiniGUI use the default configuration option. The table 2.3 lists runtime mode related options and macros.

Table 2.3 runtime mode related options and macros

Configuration options	Macro	Memo	Default

not assigned	_MGRM_THREADS	MiniGUI-Threads runtime mode	Enabled
procs	_MGRM_PROCESSES	MiniGUI-Processes runtime mode, support Linux operating system only	Disabled
standalone	_MGRM_STANDALONE	MiniGUI-Standalone runtime mode, support all operating system.	Disabled

2.2.4 Graphics Engine Related Options and Macros

MiniGUI supports many kinds of graphics engine. The commonly used graphics engine mainly includes the Dummy graphics engine, Qt Virtual FrameBuffer engine, Linux FrameBuffer console graphics engine, the COMMLCD graphics engine, the Shadow graphics engine, Windows Virtual FrameBuffer graphics engine and so on. Through the configuration option or macro, we may contain a certain graphics engine to MiniGUI. But if you assign MiniGUI to use a certain graphics engine, then you need to assign a special runtime configuration option. For instance, if you assign MiniGUI to use the dummy graphics engine, you may assign the runtime configuration option **`gal_engine=dummy`** in [**`system`**] section, the graphics engine name is on the right of the equal sign. The attention, the engine name is case sensitivity. About how to revises the runtime configuration option, please refer the 3rd chapter of *MiniGUI Runtime Configuration Options* this handbook. The table 2.5 lists the graphics engine related options, macros and the name.

Table 2.5 graphics engine related options and macros

Configuration options	Macro	Engine name	Memo	Default
videodummy	_MGGAL_DUMMY	dummy	All operating system	Enabled
videofbcon	_MGGAL_FBCON	fbcon	Linux/uClinux	Enabled
videoqvfb	_MGGAL_QVFB	qvfb	Linux	Enabled
videowvfb	_MGGAL_WVFB	wvfb	Win32; virtual buffer graphics engine, use Win32 。	Disabled
videowvfb	_MGGAL_PCXVFB	pc_xvfb	Linux/Win32 Suitable for the PC's virtual buffer graphics engine, does not depend on the specific implementation platform.	Disabled
videocommlcd	_MGGAL_COMMLCD	commlcd	All operating system	Disabled
videoshadow	_MGGAL_SHADOW	shadow	All operating system, MiniGUI-Threads, MiniGUI-Standalone runtime mode	Disabled
videodfb	_MGGAL_DFB	dfb	Run MiniGUI on DirectFB, Linux	Disabled

The Dummy is a graphics engine ("mute" graphics engine), which it does not make any actual output. Therefore, if the graphics engine for your development board still cannot work, you can run MiniGUI using this graphics engine.

The Qvfb graphics engine uses in the Linux operating system. Using qvfb, we can run the MiniGUI program in X Window; it may greatly facilitate the application debugging. Similar with the qvfb graphics engine, when uses MiniGUI SDK for Win32 run MiniGUI program on Win32 platform, it run on Windows Virtual in the FrameBuffer actually, and use the wvfb graphics engine.

It should be noted that the original QVFB (Qt Virtual Frame Buffer) and WVFB (Windows Virtual Frame Buffer) have been replaced with the newly designed XVFB general purpose virtual buffer graphics engine in MiniGUI 3.0.

In MiniGUI also has a special Shadow graphics engine, uses the Shadow graphics engine, MiniGUI may support the graphic display devices which it is lower than 8 bit colors, also support the screen rotation. The Shadow graphics engine has used the sub-driver concept; it determined which sub-driver contains through the target board name. Only one sub-driver can be contained at one time, it determined by the target board configuration option (sees section 2.2.2). The attention, the Shadow graphics engine is disabled as the default; moreover it is only suitable for the MiniGUI-Threads and MiniGUI-Standalone runtime mode at present.

The sub-drivers of the Shadow graphics in MiniGUI are (in MiniGUI source code directory **`src/newgal/shadow`**):

- **`unknown`**: the default sub-driver, similar with the dummy graphics engine, user may modify this sub-driver in order to operate and visit the low graphics devices.
- **`qvfb`**: sub-driver for Linux QVFB all display mode, support low than 8-bit color display mode and screen rotation.
- **`fbcon`**: sub-driver for Linux console FrameBuffer, support low than 8-bit color display mode and screen rotation.
- **`wvfb`**: sub-driver for Windows Virtual FrameBuffer(wvfb), support low than 8-bit color display mode and screen rotation.

We can rotate the screen by Shadow engine. Table 2.6 lists the screen rotation related options and macros.

Table 2.6 screen rotation related options and macros

Configuration options	Macro		Macro value	Comment	Default
coortrans_cw	_COOR_TRANS	_ROT_DIR_CW	1	Rotate screen clockwise	Disabled
coortrans_ccw		_ROT_DIR_CW	0	Rotate screen anticlockwise	Disabled

The CommLCD graphics engine is the most used graphics engine when MiniGUI run on the tradition embedded operating system. CommLCD also uses the sub-driver structure like Shadow graphics engine. At present, sub-drivers for CommLCD graphics engine are:

- **`vxi386`**: Sub-driver for VxWorks i386 target board.
- **`unknown`**: If is eCos operating system, then use standard interface of eCos to

implement a sub-driver. Otherwise, the sub-driver needs to be defined by the user. The rtos/ directory of the MiniGUI source tree contains the CommLCD graphics engine implementation for each operating system. You can modify this file to support your own LCD controller.

2.2.5 Input Engine Related Options and Macros

MiniGUI provides some input engine, which can be used directly for many kinds of development board. Generally the input engines include the Dummy input engine, Qt Virtual FrameBuffer engine, Linux FrameBuffer console input engine, the COMM input engine, the Random input engine, Windows Virtual FrameBuffer input engine and so on. Through the configuration options or macros, we can contain an input engine to MiniGUI. But if assign MiniGUI to use a certain input engine, then you need to assign a special runtime configuration option. For instance, If you assign MiniGUI to use the dummy input engine, you may assign the runtime configuration option **`ial_engine=dummy`** in [**`system`**] section, the input engine name is on the right of the equal sign. The attention, the engine name is case sensitivity. About how to revises the runtime configuration option, please refer the 3rd chapter of *MiniGUI Runtime Configuration Options* this handbook. The table 2.7 lists the input engine related options and macros.

Table 2.7 input engines related options and macros

Configuration options	Macro	Engine name	Comment	Default
dummyial	_MGIAL_DUMMY	dummy	Dummy input engine, for all operating system	Enabled
autoial	_MGIAL_AUTO	auto	Automatic input engine, for all operating system	Disabled
qvfbial	_MGIAL_QVFB	qvfb	QVFB input engine, Linux, use QVFB graphics engine	Enabled
consoleial	_MGIAL_CONSOLE	console	Linux console input engine, Linux	Enabled
randomial	_MGIAL_RANDOM	random	Random input engine, for all operating system	Disabled
wvfbial	_MGIAL_WVFB	wvfb	WVFB input engine, Win32, use WVFB graphics engine	Disabled
commial	_MGIAL_COMM	comm	COMM input engine, for all operating system	Disabled
dfbial	_MGIAL_DFB	dfb	Base on DirectFBinput engine, Linux, use DFB graphics engine	Disabled
tslibial	_MGIAL_TSLIB	tslib	Base on tab engine, Linux, use DFB graphics engine	Disabled
qemuial	_MGIAL_QEMU	qemu	QEMU input IAL, Linux.	Disabled

custodial	_MGIAL_CUSTOM	custom	Use on graphics engine that custom by MiniGUI application; any operating system.	Disabled

The Dummy input engine ("mute" input engine) is not connected to any actual input device; therefore it can't get any input. Therefore, if the input engine for your development board still cannot to work, you can run MiniGUI using this input engine. Attention, MiniGUI use Dummy input engine when it cannot find the matched input engine in configuration options.

Like the Dummy input engine, MiniGUI provide other two input engine, which it is not associated to any device, for instance Auto input engine and Random input engine. The Auto engine may circulation produce the events automatic according the previous setting; But the Random input engine produce the random input event. These two engines may use for MiniGUI and its application software test automation.

The Console input engine aims at the PC console of Linux operating system. This input engine supports the standard PC keyboard as well as many kinds of mouse protocol. You need configure mtype and mdev field in [`system`] section assign the mouse protocol and the mouse device when use the console input engine.
Mouse protocol related options and macros, which console input engine supported, are listed in table 2.8. Attention, although MiniGUI support intelligence mouse, but MiniGUI does not support in the middle key and the hoop input event.

Table 2.8 Mouse protocol related options and macros

configuration options	Macro	Comment	Default
consoleps2	_MGCONSOLE_PS2	Support PS2 mouse protocol	Enabled
consoleimps2	_MGCONSOLE_IMPS2	Support intelligence mouse(IMPS/2) protocol	Enabled
consolems	_MGCONSOLE_MS	Support old MS serial-port mouse	Enabled
consolems3	_MGCONSOLE_MS3	Support MS3 mouse protocol	Enabled
consolegpm	_MGCONSOLE_GPM	Support GPM Daemon processes	Enabled

Except the options above, MiniGUI has also provided mouse and touch screen adjustment interfaces for applications. If you want to use this interfaces, you need to open the option about touch screen adjusts. The table 2.9 lists touch screen adjustment related options and macros.

Table 2.9 mouse and touch screen adjustment related options and macros

configuration options	Macro	Comment	Default
mousecalibrate	_MGHAVE_MOUSECALIBRATE	Support touch screen adjustment	Enabled

2.2.6 Keyboard Layout Related Options and Macros

The MiniGUI keyboard layout uses for control the behavior of function TranslateMessage. Different keyboard layout will translate a same key as a different character (distinguish by the scan code). This translation process is implemented through query the scan code mapping table. At present, in MiniGUI contains the Western Europe country commonly used keyboard layout support, standard American 1.01/102 keyboard as default. If you want to use different keyboard layout in your program, you should call the function SetKeyboardLayout by the keyboard layout name. For more information, please refer *MiniGUI Programming Guide V3.0-5*. Table 2.10 listed the keyboard layout related options, macros and the name.

Table 2.10 keyboard layout related options and macros

configuration options	Macro	Keyboard layout name	Comment	Default
Kbdfrpc	_MGKBDLAYOUT_FRPC	frpc	Keyboard layout for French PC keyboard (non-US 102 keys)	Disabled
Kbdfr	_MGKBDLAYOUT_FR	fr	Keyboard layout for French	Disabled
Kbdde	_MGKBDLAYOUT_DE	de	Keyboard layout for German	Disabled
kbddelatin1	_MGKBDLAYOUT_DELATIN1	delatin1	Keyboard layout for German Latin1	Disabled
Kbdit	_MGKBDLAYOUT_IT	it	Keyboard layout for Italian	Disabled
Kbdes	_MGKBDLAYOUT_ES	es	Keyboard layout for Spanish	Disabled
kbdescp850	_MGKBDLAYOUT_ESCP850	escp850	Keyboard layout for Spanish CP850	Disabled
kbdhebrewpc	_MGKBDLAYOUT_HEBREWPC	hebrewpc	Keyboard layout for Hebrew PC keyboard	Disabled
kbdarabicpc	_MGKBDLAYOUT_ARABICPC	arabicpc	Keyboard layout for Arabic PC keyboard	Disabled

2.2.7 System Global Configuration Options and Macros

The table 2.11 lists system global configuration options and macros.

Table 2.11 system global configuration options and macros

configuration options	Macro	Comment	Default
incoreres	_MGINCORE_RES	Use MiniGUI in-core resource	Disabled
miniguientry	_USE_MINIGUIENTRY	Use MiniGUI minigui_entry function	Disabled
debug	_DEBUG	Include debug information	Disabled
tracemsg	_MGHAVE_TRACE_MSG	Trace MiniGUI message	Disabled
msgstr	_MGHAVE_MSG_STRING	Include the string name of the message	Disabled
dblclk	_MGMISC_DOUBLE_CLICK	Support mouse double click	Enabled

cursor	_MGHAVE_CURSOR	Support mouse cursor	Enabled
clipboard	_MGHAVE_CLIPBOARD	Support clipboard	Enabled
savebitmap	_MGMISC_SAVESCREEN	Support SaveBitmap related functions	Enabled
aboutdlg	_MGHAVE_FIXED_MATH	Include About dialog box	Enabled
savescreen	_MGHAVE_SAVESCREN	Support screen capture	Enabled
splash	_MG_ENABLE_SPLASH	MiniGUI Splash screen	Enabled
fixedmath	_MGHAVE_FIXED_MATH	Use fixed math functions	Enabled
adv2dapi	_MGHAVE_ADV_2DAPI	Support advanced 2D graphics API	Enabled
screensaver	_MG_ENABLE_SCREENSAVER	Screen saver	Enabled

Some important configurations are introduced as the follow:

The **`incoreres`** option is used to control whether MiniGUI needs fonts, bitmaps, cursors, icons and so on construct in the function library. This option is very useful for tradition embedded operating system. Because in the majority situation, the tradition embedded operating system has not file system support, supporting by the in-core resource, it was allowed to construct the above resources in the function library, and MiniGUI can run without file system. Attention in, when uses in-core resources, MiniGUI runtime configuration options can be compiled into MiniGUI library directly.

The **`miniguientry`** option uses for control how to implement the function MiniGUIMain. In the default situation (disabled this option), The function MiniGUIMain can be expanded to the function main, so application should not define the main function. The function MiniGUIMain can be expanded to the function **`minigui_entry`** when option **`miniguientry`** is enabled. It is easy for debug and system integration for some tradition embedded operating system.

The **`fixedmath`** option uses for control whether fixed math is included in MiniGUI library, such as fixcos and so on. The clipboard option uses for control whether MiniGUI is support clipboard or not; if this option is disabled, and the editor cannot support cut and copy. The **`adv2api`** option is control whether the MiniGUI include the advanced 2D graphics API.

The debug, **`tracemsg`** and **`msgstr`** use for MiniGUI debugging, it is not suggested user use it.

MiniGUI supports mouse cursor default. When target system has not any fix point device like mouse or touch screen, we do not need display the mouse cursor, so we can disabled the mouse cursor supporting from the configuration options.

Splash and screensaver options are used to define the splash screen and MiniGUI built-in screen saver program. In the actual project, you can usually close these two options.

2.2.8 Character Set and Font Related Options and Macros

MiniGUI has rich support for font. It supports RBF font, VBF font (these two kinds of font are defined by MiniGUI), UPF/QPF font, TrueType font, Adobe Type1 font and so on. Because MiniGUI supports many kinds of font, so there are many flexible configuration options for font.

Like the type of font, MiniGUI provides a well support for character set. A special character set support also can be flexible configured. Table 2.13 lists character set and font related options and macros.

Table 2.13 character set and font related options and macros

configuration options	Macro	Comment	Default
latin2support	_MGCHARSET_LATIN2	Include East European (Latin 2, ISO-8859-2) charset support	Disabled
latin3support	_MGCHARSET_LATIN3	Include South European (Latin 3, ISO-8859-3) charset support	Disabled
latin4support	_MGCHARSET_LATIN4	Include North European (Latin 4, ISO-8859-4) charset support	Disabled
cyrillicsupport	_MGCHARSET_CYRILLIC	Include Cyrillic (ISO-8859-5) charset support	Disabled
arabicsupport	_MGCHARSET_ARABIC	Include Arabic (ISO-8859-6) charset support	Disabled
greeksupport	_MGCHARSET_GREEK	Include Greek (ISO-8859-7) charset support	Disabled
hebrewsupport	_MGCHARSET_HEBREW	Include Hebrew (ISO-8859-8) charset support	Disabled
latin5support	_MGCHARSET_LATIN5	Include Turkish (Latin 5, ISO-8859-9) charset support	Disabled
latin6support	_MGCHARSET_LATIN6	Include Nordic, Latin 6, ISO-8859-10) charset support	Disabled
thaisupport	_MGCHARSET_THAI	Include Thai (ISO-8859-11) charset support	Disabled
latin7support	_MGCHARSET_LATIN7	Include Latin 7 (ISO-8859-13) charset support	Disabled
latin8support	_MGCHARSET_LATIN8	Include Latin 8 (ISO-8859-14) charset support	Disabled
latin9support	_MGCHARSET_LATIN9	Include Latin 9 (ISO-8859-15, West Extended) charset support	Disabled
latin10support	_MGCHARSET_LATIN10	Include Latin 10 (ISO-8859-16, Romanian) charset support	Disabled
gbsupport	_MGCHARSET_GB	Include EUC encoding of GB2312 charset support	Enabled
gbksupport	_MGCHARSET_GBK	Include GBK charset support	Enabled

gb18030support	_MGCHARSET_GB18030	Include GB18030-0 charset support	Disabled
big5support	_MGCHARSET_BIG5	Include BIG5 charset support	Enabled
euckrsupport	_MGCHARSET_EUCKR	Include support for EUC encoding of KSC5636 and KSC5601 charsets	Disabled
eucjpsupport	_MGCHARSET_EUCJP	Include support for EUC encoding of JISX0201 and JISX0208 charsets	Disabled
shiftjissupport	_MGCHARSET_SHIFTJIS	Include support for Shift-JIS encoding of JISX0201 and JISX0208 charsets	Disabled
unicodesupport	_MGCHARSET_UNICODE	Include UNICODE (ISO-10646-1 and UTF-8 encoding) support	Enabled
rbfsupport	_MGFONT_RBF	Include RBFfont support	Enabled
rbfvgaoem	_MGINCORERBF_LATIN1_VGAOEM	Include incore RBF font of ISO8859-1 8x16 fixed font	Disabled
rbfterminal	_MGINCORERBF_LATIN1_TERMINAL	Include incore RBF font of ISO8859-1 12x24 fixed font	Disabled
rbffixedsys	_MGINCORERBF_LATIN1_FIXEDSYS	Include incore RBF font of GB2312 12x12 fixed/song font	Enabled
vbfsupport	_MGFONT_VBF	Include var bitmap font support	Enabled
fontsserif	_MGINCOREFONT_SANSSERIF	Include incore VBF font sansserif	Enabled
fontcourier	_MGINCOREFONT_COURIER	Include incore VBF font courier	Enabled
fontsystem	_MGINCOREFONT_SYSTEM	Include incore VBF font symbol	Disabled
upfsupport	_MGFONT_UPF	Support FMSoft Unicode Prerendered Font(UPF).	Enabled
fonttimes	_MGINCOREFONT_TIMES	Include income Times UPF font	Enabled
qpfsupport	_MGFONT_QPF	Include Qt Prerendered Font (QPF) support	Enabled
ttfsupport=ft1	_MGFONT_TTF	Include TrueType Library support	Disabled
ttfsupport=ft2	_MGFONT_FT2	Include FreeType2 font support	Disabled
ttfcachesize=256	_MGTTF_CACHE_SIZE	Include TrueType cache support	256
mttfcachenum=10	_MGMAX_TTF_CACHE	Include TrueType cache num	10

The options latin2support, latin3support, cyrillicsupport, arabicsupport, greeksupport, hebrewsupport, latin5support, latin6support, thaisupport, latin7support, latin8support, latin9support, latin10support control ISO8859-2 to ISO8859-16 character set support, they are single byte character set. There are supporting for ASCII character and ISO8859-1 (Latin1) build in MiniGUI. No configuration options for these two character sets.

The options gbsupport, gbksupport, gb18030support, big5support, euckrsupport, eucjpsupport, shiftjissupport, unicodesupport control GB2312, GBK, GB18030, BIG5, EUCKR, EUCJP, SHIFTJIS, UNICODE character set/code system support.

The option rbfsupport control whether include the support for Raw Bitmap Font (RBF) font, it is enabled as the default. Because RBF is the default font format, so it is not suggested that user disable the support for this font type.

rbfvgaoem, rbfterminal, rbffixedsys and other configuration options to control whether the corresponding RBF dot matrix font built in MiniGUI library. These compiler configuration options are enabled by default, so that MiniGUI can still run normally when no font is loaded.

The option vbfsupport control whether include support for Variable Bitmap Font (VBF) font, it is enabled default. If this option is disabled, you not only disable the support for VBF font but also disable the VBF font build in MiniGUI. When MiniGUI is running, the runtime option [**`varbitmapfonts`**] section is ignored.

The fontsserif configuration options as well as fontcourier, fontsystem compilation configuration options to control whether the MiniGUI library built-in SanSerif, Courier and System VBF fonts. These built-in font options are on by default and are not affected by the incoreres option.

The option upfsupport controls whether support for FMSoft Unicode Prerendered Font (UPF) fonts is included in the MiniGUI library. Because UPF fonts use UNICODE encoding, allowing UPF fonts support will automatically enable MiniGUI's UNICODE character set support.

The option qpfsupport control whether support for Qt/Embedded Prerendered Font (QPF). Because QPF font uses UNICODE coding, so if support QPF font in MiniGUI, the UNICODE support is enabled automatically. If incoreres option is enabled, some QPF fonts will be built in MiniGUI.

The option ft2support control whether support for FreeType2 library in MiniGUI library. MiniGUI can render the TrueType font by FreeType2 library version 2.3.4. If FreeType2 library is not installed in your system, the configuration will disable this option automatically.

The option ttfsupport control whether support for TrueType in MiniGUI library. MiniGUI also can render the TrueType font by FreeType library version 1.3.0. If FreeType library version 1.3.0 is not installed in your system, the configuration will disable this option automatically. The attention, the interfaces of FreeType 2 are not compatible with FreeType 1.

The option ttfcache control whether support TrueType cache for FreeType1, it is enabled default. If ttfcache need enable, the option ttfsupport should be enabled first.

The option **`--with-mttfcachenum`** uses for appoint the number of the cache block when TrueType cache is enabled. The default value is 10.

The option **`--with-ttfcachesize`** uses for appoint the size of cache block when TrueType cache is enabled, the default value is 64k.

Table 2.14 and table 2.15 list the TrueType cache related parameters, options and macros.

Table 2.14 TrueType cache related options and macros

Configure option	Macro	Macro value	Memo
--with-mttfcachenum=10	_MGMAX_TTF_CACHE	10	Default value
--with-mttfcachenum=20		20	
--with-mttfcachenum=40		40	

Table 2.15 TrueType cache related options and macros

Configure option	Macro	Macro value	Memo
--with-ttfcachesize=64	_MGTTF_CACHE_SIZE	64	Default value
--with-ttfcachesize=128		128	
--with-ttfcachesize=256		256	
--with-ttfcachesize=512		512	
--with-ttfcachesize=1024		1024	

2.2.9 Image File Format Related Options and Macros

MiniGUI support for multiple image file formats, idiographic, MiniGUI include Windows BMP, GIF, JPEG, PNG, PCX, LBM/PBM, TGA and so on. Thereinto, MiniGUI only support Windows BMP in incore resource, so there is not corresponding configuration option; The configuration option of GIF, JPEG, PNG file is enabled; The configuration option of PCX, LBM/PBM, TGA is disabled. It should be noted that if you want to MiniGUI support JECG and PNG picture format, you need to install corresponding libjpeg and libpng libraries into your system, there is the source code of these two function libraries in the MiniGUI official website.

The table 2.16 listed image file format related configuration options and macros.

Table 2.16 image file format related configuration options and macros

configuration option	Macro	Comment	Default value
gifsupport	_MGIMAGE_GIF	Support for GIF file	Enable
jpgsupport	_MGIMAGE_JPG	Support for JPG file	Enable
pngsupport	_MGIMAGE_PNG	Support for PNG file	Enable
pcxsupport	_MGIMAGE_PCX	Support for PCX file	Disable
lbmsupport	_MGIMAGE_LBM	Support for LBM/PBM file	Disable
tgasupport	_MGIMAGE_TGA	Support for TGA file	Disable

2.2.10 Appearance Style Related Options and Macros

In MiniGUI 3.0, we introduced Look and Feel (LF) concept. The original flat, classic, fashion window style abstraction as a new LF renderer (LFRDR), retained flat, classic renderer, while introducing a skin renderer, while the original Fashion style through mGPlus. Where the classic renderer is built-in, flat and skin renderers are controlled by configuration options. Table 2.17 shows the appearance renderer configuration options and the corresponding macros.

Table 2.17 appearance style related configuration options and macros

configuration option	Macro	Comment	Memo
--enable-flatlf	_MGLF_RDR_FLAT	Simple flat style	Enabled
--enable-skinlf	_MGLF_RDR_SKIN	Skin style, window and control fill by bitmap.	Enabled

2.2.11 Control Related Options and Macros

MiniGUI provides configuration options for all base controls. MiniGUI base controls refer to the controls contained in the MiniGUI core library. From MiniGUI 3.0, we provide a new set of controls through the mGNCS component. The new control set is well-designed and elegantly interfaced, which completely replaces the MiniGUI's base control set. Therefore, we strongly suggest that the new MiniGUI application be developed using mGNCS instead of the MiniGUI built-in base control. Because there is a better mGNCS, MiniGUI most of the base control configuration options are turned off by default. If your application uses these controls, please open the relevant configuration items.

Table 2.18 give all controls related configuration options and macros.

Table 2.18 control related configuration options and macros

configuration option	Macro	Comment	Default value
ctrlstatic	_MGCTRL_STATIC	Include STATIC control	Enable
ctrlbutton	_MGCTRL_BUTTON	Include BUTTON control	Enable
ctrlsledit	_MGCTRL_SLEDIT	Include Simple EDITcontrol	Enable
ctrlbidiedit	_MGCTRL_BIDIEDIT	Include BIDI EDIT control	Disable
newtextedit	_MGCTRL_TEXTEDIT _MGCTRL_TEXTEDIT_USE_NEW_IMPL	Include new textedit control	Enable
ctrllistbox	_MGCTRL_LISTBOX	Include LISTBOX control	Enable
ctrlpgbar	_MGCTRL_PROGRESSBAR	Include PROGRESSBAR control	Enable
ctrlcombobox	_MGCTRL_COMBOBOX	Include COMBOBOX control	Enable
ctrlpropsheet	_MGCTRL_PROPSHEET	Include MENUBUTTON control	Enable
ctrltrackbar	_MGCTRL_TRACKBAR	Include TRACKBARcontrol	Disable
ctrlscrollbar	_MGCTRL_SCROLLBAR	Include SCROLLBAR control	Disable

ctrlnewtoolbar	_MGCTRL_NEWTOOLBAR	Include NEWTOOLBAR control	Disable
ctrlmenubtn	_MGCTRL_MENUBUTTON	Include MENUBUTTON control	Disable
ctrlscrollview	_MGCTRL_SCROLLVIEW	Include SCROLLVIEW control	Disable
ctrltextedit	_MGCTRL_TEXTEDIT	Include base ScrollView support textedit control	Disable
ctrlmonthcal	_MGCTRL_MONTHCAL	Include MONTHCALENDAR control	Disable
ctrltreeview	_MGCTRL_TREEVIEW	Include TREEVIEW control	Disable
ctrlspinbox	_MGCTRL_SPINBOX	Include SPINBOX control	Disable
ctrlcoolbar	_MGCTRL_COOLBAR	Include COOLBAR control	Disable
ctrllistview	_MGCTRL_LISTVIEW	Include LISTVIEW control	Disable
ctrliconview	_MGCTRL_ICONVIEW	Include ICONVIEW control	Disable
ctrlgridview	_MGCTRL_GRIDVIEW	Include gridview control	Disable
ctrlanimation	_MGCTRL_ANIMATION	Include the ANIMATION control and provides support for GIF89a files	Enable

2.2.12 Other Options and Macros

MiniGUI implemented some function families of the standard C function libraries to be fit in with all kinds of embedded operating system environment, it include malloc function family (malloc, calloc, free function and so on), stdio format input and output function family (printf, sprintf and so on) and POSIX thread function library interface (pthread_create, sem_post and so on). Default, these function families compile configuration options is disabled, and that they are useful in the some traditional embedded operating system based on thread and task. If you want to enable these options in the some operating systems, you can refer to 2.2.1 chapter. Table 2.19 listed MiniGUI implemented C library interface configuration options and corresponding macros.

Table 2.19 MiniGUI implemented C library interface related configurations and macros

configuration option	Macro	Comment	Default value
ownmalloc	_MGUSE_OWN_MALLOC	Use MiniGUI implemented malloc function family	Disable
ownstdio	_MGUSE_OWN_STDIO	Use MiniGUI implemented stdio format input and output function family	Disable
ownpthread	_MGUSE_OWN_PTHREAD	Use MiniGUI implemented thread function family	Disable

Otherwise, you must define this macro: __MINIGUI_LIB__ , when you use yourself makefile to compile MiniGUI function library in the Non-GNU development environment.

Table 2.20 other compile macros

Macro	Comment	Memo
__MINIGUI_LIB__	Compile MiniGUI library macro	You must define this macros, when you use the Non-GNU makefile

Start with MiniGUI 3.0, you can specify the name suffix of the MiniGUI library through the configure option. By default, the name of the MiniGUI library varies depending on the operating mode, for example, libminigui-ths.so, libminigui-procs.so, libminigui-sa.so , Respectively, corresponding to MiniGUI-Threads, MiniGUI-Processes and MiniGUI-Standalone mode of operation.

You can specify a special library name suffix with the --with-libsuffix option.

2.3 Minimum Configuration Options

In this chapter, we will give an example of minimum configuration options in MiniGUI.

2.3.1 Using GNU Configure Script

There is a buildlib-min script in the MiniGUI source codes build directory. The buildlib-min script will be as the following:

```
#!/bin/sh

./configure \
    --disable-dblclk \
    --disable-cursor \
    --disable-mousecalibrate \
    --disable-clipboard \
    --disable-adv2dapi \
    --disable-splash \
    --disable-screensaver \
    --disable-flatlf \
    --disable-skinlf \
    --disable-rbfvgaoem \
    --disable-rbfterminal \
    --disable-vbfsupport \
    --disable-qpfsupport \
    --disable-upfsupport \
    --disable-bmpfsupport \
    --disable-latin9support   \
    --disable-gbsupport       \
    --disable-gbksupport      \
    --disable-unicodesupport  \
    --disable-savebitmap \
    --disable-jpgsupport \
    --disable-pngsupport \
    --disable-gifsupport \
    --disable-aboutdlg \
    --disable-savescreen \
    --disable-mousecalibrate \
    --disable-ctrlanimation \
    --disable-ctrlnewtextedit \
    --disable-consoleps2 \
    --disable-consoleimps2 \
    --disable-consolems \
    --disable-consolems3 \
    --disable-consolegpm
```

By this script, you can configure MiniGUI to the minimum function library that only supports ISO8859-1 charset.

- Compiling MiniGUI to be MiniGUI-Threads.
- No support for double click mouse button.
- No support for cursor.
- No support for code doing mouse calibration.
- No support for clipboard.
- No including VGAOEM/Terminal incoreres font.
- No support for VBF font.
- No support for Qt Prerendered Font(QPF).
- No support for UPF Prerendered Font(UPF).
- No support for TrueType font.
- No support bitmap font.
- No support for Latin 9(ISO-8859-15, West Extended) charset.
- No support for EUC GB2312 charset.
- No support for GBK charset.
- No support for BIG5 charset.
- No support for UNICODE (ISO-10646-1and UTF-8).
- No support for BITMAP saving function.
- No support for JPG image format.
- No support for PNG image format.
- No support for GIF image format.
- No including "About MiniGUI" dialog box.
- No support for screen save function.
- No support for advanced 2D graphics APIs
- No include new TEXTEDIT support.
- No building the console engine subdriver for PS2 mouse.
- No building the console engine subdriver for IntelligentMouse (IMPS/2).
- No building the console engine subdriver for old MS serial mouse.
- No building the console engine subdriver for MS3 mouse.
- No building the console engine subdriver for GPM daemon.
- No Skin and Flat support.

Based on the configuration above, you can also delete some functions if you want. For example, if you do not use menu button control in your application, you can add –**`disable-ctrlanimation`** option in the configuration script above, so there is not GIF animation control in your compiled functions library, the MiniGUI functions library is made smaller.

2.3.2 Corresponding mgconfig.h

The **`mgconfig.h`** file to be generated in the configuration script above, listed as follows:

```
...

/* MiniGUI configure file name */
#define ETCFILENAME "MiniGUI.cfg"

...

/* Binary age of MiniGUI */
#define MINIGUI_BINARY_AGE 0

/* Interface age of MiniGUI */
```

```
#define MINIGUI_INTERFACE_AGE 0

/* Major version of MiniGUI */
#define MINIGUI_MAJOR_VERSION 3

/* Micro version of MiniGUI */
#define MINIGUI_MICRO_VERSION 13

/* Minor version of MiniGUI */
#define MINIGUI_MINOR_VERSION 0

...

/* Define if support Arabic charset */
/* #undef _MGCHARSET_ARABIC */

/* Define if support BIG5 charset */
/* #undef _MGCHARSET_BIG5 */

/* Define if support Cyrillic charset */
/* #undef _MGCHARSET_CYRILLIC */

/* Define if support EUCJP charset */
/* #undef _MGCHARSET_EUCJP */

/* Define if support EUCKR charset */
/* #undef _MGCHARSET_EUCKR */

/* Define if support GB2312 charset */
/* #undef _MGCHARSET_GB */

/* Define if support GB18030 charset */
/* #undef _MGCHARSET_GB18030 */

/* Define if support GBK charset */
/* #undef _MGCHARSET_GBK */

/* Define if support Greek charset */
/* #undef _MGCHARSET_GREEK */

/* Define if support Hebrew charset */
/* #undef _MGCHARSET_HEBREW */

/* Define if support Latin 10 charset */
/* #undef _MGCHARSET_LATIN10 */

/* Define if support Latin 2 charset */
/* #undef _MGCHARSET_LATIN2 */

/* Define if support Latin 3 charset */
/* #undef _MGCHARSET_LATIN3 */

/* Define if support Latin 4 charset */
/* #undef _MGCHARSET_LATIN4 */

/* Define if support Latin 5 charset */
/* #undef _MGCHARSET_LATIN5 */

/* Define if support Latin 6 charset */
/* #undef _MGCHARSET_LATIN6 */

/* Define if support Latin 7 charset */
/* #undef _MGCHARSET_LATIN7 */

/* Define if support Latin 8 charset */
/* #undef _MGCHARSET_LATIN8 */

/* Define if support Latin 9 charset */
/* #undef _MGCHARSET_LATIN9 */

/* Define if support SHIFTJIS charset */
/* #undef _MGCHARSET_SHIFTJIS */

/* Define if support Thai charset */
/* #undef _MGCHARSET_THAI */

/* Define if support UNICODE */
/* #undef _MGCHARSET_UNICODE */

/* Define if include GPM mouse subdriver */
/* #undef _MGCONSOLE_GPM */

/* Define if include IMPS2 mouse subdriver */
/* #undef _MGCONSOLE_IMPS2 */

/* Define if include MS mouse subdriver */
/* #undef _MGCONSOLE_MS */

/* Define if include MS3 mouse subdriver */
/* #undef _MGCONSOLE_MS3 */

/* Define if include PS2 mouse subdriver */
/* #undef _MGCONSOLE_PS2 */

/* Define if your Linux have text mode */
#define _MGCONSOLE_TEXTMODE 1

/* Define if include ANIMATION control */
/* #undef _MGCTRL_ANIMATION */

/* Define if include BIDISLEDIT control */
/* #undef _MGCTRL_BIDISLEDIT */

/* Define if include BUTTON control */
#define _MGCTRL_BUTTON 1

/* Define if include COMBOBOX control */
#define _MGCTRL_COMBOBOX 1

/* Define if include COOLBAR control */
/* #undef _MGCTRL_COOLBAR */

/* Define if include GRIDVIEW control */
/* #undef _MGCTRL_GRIDVIEW */

/* Define if include ICONVIEW control */
/* #undef _MGCTRL_ICONVIEW */

/* Define if include LISTBOX control */
#define _MGCTRL_LISTBOX 1

/* Define if include LISTVIEW control */
/* #undef _MGCTRL_LISTVIEW */
/* Define if include MENUBUTTON control */
/* #undef _MGCTRL_MENUBUTTON */

/* Define if include MONTHCALENDAR control */
/* #undef _MGCTRL_MONTHCAL */

/* Define if include NEWTOOLBAR control */
/* #undef _MGCTRL_NEWTOOLBAR */

/* Define if include PROGRESSBAR control */
#define _MGCTRL_PROGRESSBAR 1

/* Define if include PROPSHEET control */
#define _MGCTRL_PROPSHEET 1

/* Define if include SCROLLBAR control */
/* #undef _MGCTRL_SCROLLBAR */

/* Define if include SCROLLVIEW control */
/* #undef _MGCTRL_SCROLLVIEW */

/* Define if include SLEDIT control */
#define _MGCTRL_SLEDIT 1

/* Define if include SPINBOX control */
/* #undef _MGCTRL_SPINBOX */

/* Define if include STATIC control */
#define _MGCTRL_STATIC 1

/* Define if include TEXTEDIT control */
/* #undef _MGCTRL_TEXTEDIT */

/* Define if use new implementation of TEXTEDIT control */
/* #undef _MGCTRL_TEXTEDIT_USE_NEW_IMPL */

/* Define if include TRACKBAR control */
/* #undef _MGCTRL_TRACKBAR */

/* Define if include TREEVIEW control */
/* #undef _MGCTRL_TREEVIEW */

/* Define if include TREEVIEWRDR control */
/* #undef _MGCTRL_TREEVIEW_RDR */

/* Define if support Bitmap fonts */
/* #undef _MGFONT_BMPF */

/* Define if support TrueType font based on FreeType2 */
/* #undef _MGFONT_FT2 */

/* Define if support QPF font */
/* #undef _MGFONT_QPF */

/* Define if support raw bitmap fonts */
#define _MGFONT_RBF 1

/* Define if support SEF scripteary font */
/* #undef _MGFONT_SEF */

/* Define if support TrueType font */
/* #undef _MGFONT_TTF */
/* Define if include ttf cache */
/* #undef _MGFONT_TTF_CACHE */

/* Define if support UPF font */
/* #undef _MGFONT_UPF */

/* Define if support var bitmap fonts */
/* #undef _MGFONT_VBF */

/* Define if include NEWGAL engine for BF533 OSD via SPI */
/* #undef _MGGAL_BF533 */

/* Define if include NEWGAL engine for Common LCD */
/* #undef _MGGAL_COMMLCD */

/* Define if include custom NEWGAL engine */
/* #undef _MGGAL_CUSTOMGAL */

/* Define if include NEWGAL engine for DirectFB */
/* #undef _MGGAL_DFB */

/* Define if include ST7167 subdriver for NEWGAL engine of DirectFB */
/* #undef _MGGAL_DFB_ST7167 */

/* Define if include dummy NEWGAL engine */
#define _MGGAL_DUMMY 1

/* Define if include NEWGAL engine for EM85xx OSD */
/* #undef _MGGAL_EM85XXOSD */

/* Define if include NEWGAL engine for EM85xx YUV */
/* #undef _MGGAL_EM85XXYUV */

/* Define if include NEWGAL engine for EM86xx GFX */
/* #undef _MGGAL_EM86GFX */

/* Define if include FrameBuffer console NEWGAL engine */
#define _MGGAL_FBCON 1

/* Define if include GDL Video NEWGAL engine */
/* #undef _MGGAL_GDL */

/* Define if include Hi35XX Video NEWGAL engine */
/* #undef _MGGAL_HI3510 */

/* Define if include Hi35XX Video NEWGAL engine */
/* #undef _MGGAL_HI3560 */

/* Define if include Hi3560A Video NEWGAL engine */
/* #undef _MGGAL_HI3560A */

/* Define if include NEWGAL engine for mb93493 YUV FrameBuffer driver */
/* #undef _MGGAL_MB93493 */

/* Define if include MLShadow NEWGAL engine */
/* #undef _MGGAL_MLSHADOW */

/* Define if include mstar NEWGAL engine */
/* #undef _MGGAL_MSTAR */

/* Define if include nexus NEWGAL engine */
/* #undef _MGGAL_NEXUS */

/* Define if include PC Virtual FrameBuffer NEWGAL engine */
#define _MGGAL_PCXVFB 1

/* Define if include Qt Virtual FrameBuffer NEWGAL engine */
/* #undef _MGGAL_QVFB */

/* Define if include RTOS Virtual FrameBuffer NEWGAL engine */
/* #undef _MGGAL_RTOSXVFB */

/* Define if include s3c6410 NEWGAL engine */
/* #undef _MGGAL_S3C6410 */

/* Define if include Shadow NEWGAL engine */
/* #undef _MGGAL_SHADOW */

/* Define if include sigma8654 NEWGAL engine */
/* #undef _MGGAL_SIGMA8654 */

/* Define if include NEWGAL engine for STGFB */
/* #undef _MGGAL_STGFB */

/* Define if include NEWGAL engine for SVPXX OSD */
/* #undef _MGGAL_SVPXXOSD */

/* Define if include NEWGAL engine for UTPMC */
/* #undef _MGGAL_UTPMC */

/* Define if include windows Virtual FrameBuffer NEWGAL engine */
/* #undef _MGGAL_WVFB */

/* Define if include advanced 2D graphics APIs */
/* #undef _MGHAVE_ADV_2DAPI */

/* Define if include clipboard support */
/* #undef _MGHAVE_CLIPBOARD */

/* Define if include cursor support */
/* #undef _MGHAVE_CURSOR */

/* Define if include fixed math routines */
#define _MGHAVE_FIXED_MATH 1

/* Define if support menu */
#define _MGHAVE_MENU 1

/* Define if include code for mouse calibration */
/* #undef _MGHAVE_MOUSECALIBRATE */

/* Define if include message string names */
/* #undef _MGHAVE_MSG_STRING */

/* Define if PCIAccess lib is available */
/* #undef _MGHAVE_PCIACCESS */

/* Define if trace message dispatching of MiniGUI */
/* #undef _MGHAVE_TRACE_MSG */

/* Define if include the 2440 IAL engine */
/* #undef _MGIAL_2440 */
/* Define if include the automatic IAL engine */
/* #undef _MGIAL_AUTO */

/* Define if include IAL engine for Cisco touchpad */
/* #undef _MGIAL_CISCO_TOUCHPAD */

/* Define if include the common IAL engine */
/* #undef _MGIAL_COMM */

/* Define if include console (Linux console) IAL engine */
#define _MGIAL_CONSOLE 1

/* Define if include IAL engine for customer's board */
/* #undef _MGIAL_CUSTOM */

/* Define if include the DAVINCI6446 IAL engine */
/* #undef _MGIAL_DAVINCI6446 */

/* Define if include the DFB IAL engine */
/* #undef _MGIAL_DFB */

/* Define if include dlcustom IAL engine */
/* #undef _MGIAL_DLCUSTOM */

/* Define if include the dummy IAL engine */
#define _MGIAL_DUMMY 1

/* Define if include IAL engine for iPAQ H3600 */
/* #undef _MGIAL_IPAQ_H3600 */

/* Define if include IAL engine for iPAQ H5400 */
/* #undef _MGIAL_IPAQ_H5400 */

/* Define if include the JZ4740 IAL engine */
/* #undef _MGIAL_JZ4740 */

/* Define if include the lide IAL engine */
/* #undef _MGIAL_LIDE */

/* Define if include IAL engine for MStar */
/* #undef _MGIAL_MSTAR */

/* Define if include IAL engine for net's board */
/* #undef _MGIAL_NET */

/* Define if include IAL engine for Nexus */
/* #undef _MGIAL_NEXUS */

/* Define if include the QEMU IAL engine */
/* #undef _MGIAL_QEMU */

/* Define if include the QVFB IAL engine */
/* #undef _MGIAL_QVFB */

/* Define if include the random IAL engine */
/* #undef _MGIAL_RANDOM */

/* Define if include IAL engine for TSLIB */
/* #undef _MGIAL_TSLIB */

/* Define if include the WVFB IAL engine */
/* #undef _MGIAL_WVFB */
/* Define if support GIF bmp file format */
/* #undef _MGIMAGE_GIF */

/* Define if support JPEG bmp file format */
/* #undef _MGIMAGE_JPG */

/* Define if support LBM bmp file format */
/* #undef _MGIMAGE_LBM */

/* Define if support PCX bmp file format */
/* #undef _MGIMAGE_PCX */

/* Define if support PNG bmp file format */
/* #undef _MGIMAGE_PNG */

/* Define if support TGA bmp file format */
/* #undef _MGIMAGE_TGA */

/* Define if include in-core font: Courier */
/* #undef _MGINCOREFONT_COURIER */

/* Define if include in-core font: SansSerif */
/* #undef _MGINCOREFONT_SANSSERIF */

/* Define if include in-core font: System */
/* #undef _MGINCOREFONT_SYSTEM */

/* Define if include in-core UPF Times fonts */
/* #undef _MGINCOREFONT_TIMES */

/* Define if include in-core FixedSys RBF for ISO8859-1 */
#define _MGINCORERBF_LATIN1_FIXEDSYS 1

/* Define if include in-core Terminal RBF for ISO8859-1 */
/* #undef _MGINCORERBF_LATIN1_TERMINAL */

/* Define if include in-core VGAOEM RBF for ISO8859-1 */
/* #undef _MGINCORERBF_LATIN1_VGAOEM */

/* Define if build MiniGUI for no file I/O system (use in-core resources) */
/* #undef _MGINCORE_RES */

/* Define if use the Arabic PC keyboard layout */
/* #undef _MGKBDLAYOUT_ARABICPC */

/* Define if use the German keyboard layout */
/* #undef _MGKBDLAYOUT_DE */

/* Define if use the German-Latin1 keyboard layout */
/* #undef _MGKBDLAYOUT_DELATIN1 */

/* Define if use the Spanish keyboard layout */
/* #undef _MGKBDLAYOUT_ES */

/* Define if use the Spanish CP850 keyboard layout */
/* #undef _MGKBDLAYOUT_ESCP850 */

/* Define if use the French keyboard layout */
/* #undef _MGKBDLAYOUT_FR */

/* Define if use the French PC keyboard layout */
/* #undef _MGKBDLAYOUT_FRPC */

/* Define if use the Hebrew PC keyboard layout */
/* #undef _MGKBDLAYOUT_HEBREWPC */

/* Define if use the Italian keyboard layout */
/* #undef _MGKBDLAYOUT_IT */

/* Define if include flat Look and Feel */
/* #undef _MGLF_RDR_FLAT */

/* Define if include skin Look and Feel */
/* #undef _MGLF_RDR_SKIN */

/* MiniGUI library suffix */
#define _MGLIB_SUFFIX "ths"

/* Define if compile max ttf cahce number for 10 (default value) */
/* #undef _MGMAX_TTF_CACHE */

/* Define if include About MiniGUI Dialog Box */
/* #undef _MGMISC_ABOUTDLG */

/* Define if mouse button can do double click */
/* #undef _MGMISC_DOUBLE_CLICK */

/* Define if include SaveBitmap function */
/* #undef _MGMISC_SAVEBITMAP */

/* Define if include code for screenshots */
/* #undef _MGMISC_SAVESCREEN */

/* Define if build MiniGUI-Processes */
/* #undef _MGRM_PROCESSES */

/* Define if build MiniGUI-Standalone */
/* #undef _MGRM_STANDALONE */

/* Define if build MiniGUI-Threads */
#define _MGRM_THREADS 1

/* Define if the unit of timer is 10ms */
#define _MGTIMER_UNIT_10MS 1

/* Define if compile max ttf cahce size for 256k */
/* #undef _MGTTF_CACHE_SIZE */

/* Define if use own implementation of malloc functions */
/* #undef _MGUSE_OWN_MALLOC */

/* Define if use own implementation of pthread functions */
/* #undef _MGUSE_OWN_PTHREAD */

/* Define if use own implementation of stdio functions */
/* #undef _MGUSE_OWN_STDIO */

/* Define if build the mgeff support version */
/* #undef _MG_MINIMALGDI */

/* Define if insert a productid into the library file */
/* #undef _MG_PRODUCTID */
/* Define if build MiniGUI-Standalone (back-compatibility definition) */
/* #undef _STAND_ALONE */

/* Define if use minigui_entry function in MiniGUI */
/* #undef _USE_MINIGUIENTRY */

/* Define if compile for Cygwin platform */
/* #undef __CYGWIN__ */

/* Define if compile for OpenDarwin */
/* #undef __DARWIN__ */

/* Define if compile for eCos */
/* #undef __ECOS__ */

/* Define if compile for Linux */
#define __LINUX__ 1

/* Define if compile for non-UNIX like OS */
/* #undef __NOUNIX__ */

/* Define if compile for Nucleus */
/* #undef __NUCLEUS__ */

/* Define if compile for OSE */
/* #undef __OSE__ */

/* Define if compile for pSOS */
/* #undef __PSOS__ */

/* Define for Blackfin run uClinux */
/* #undef __TARGET_BLACKFIN__ */

/* Define for EPSON C33L05 (axLinux) */
/* #undef __TARGET_C33L05__ */

/* Define for FMSoft internal use */
/* #undef __TARGET_FMSOFT__ */

/* Define for Monaco ANVIL target */
/* #undef __TARGET_MONACO__ */

/* Define for FMSoft miniStudio */
/* #undef __TARGET_MSTUDIO__ */

/* Define for OSE on mx21 */
/* #undef __TARGET_MX21__ */

/* Define for VxWorks on PowerPC */
/* #undef __TARGET_PPC__ */

/* Define for Philips STB810 target */
/* #undef __TARGET_STB810__ */

/* Define for unknown target */
#define __TARGET_UNKNOWN__ 1

/* Define for VirualFone ANVIL target */
/* #undef __TARGET_VFANVIL__ */

/* Define for VxWorks on i386 */
/* #undef __TARGET_VXI386__ */
/* Define if compile for ThreadX */
/* #undef __THREADX__ */

/* Define if compile for uC/OS-II */
/* #undef __UCOSII__ */

/* Define if compile for VxWorks */
/* #undef __VXWORKS__ */

/* Define if compile for Winbond SWLinux */
/* #undef __WINBOND_SWLINUX__ */

/* Define if compile for uClinux */
/* #undef __uClinux__ */

...
```

2.4 Compiling and Installing MiniGUI

2.4.1 compile and install the dependent library

Before running MiniGUI, you need to install the dependent libraries required by MiniGUI. MiniGUI mainly uses LibFreeType, LibPNG, LibJPEG, LibZ and other third-party dependent libraries.

These dependent library source code packages basically use the GNU Automake / Autoconf script to organize projects and compile and install these libraries by specifying specific environment variables and certain options when running ./configure commands. We can also check the acceptable switch parameters for each configure script by running the ./configure --help command in these dependent source files.

Currently, these dependencies are basically standard configurations of mainstream Linux distributions (such as Ubuntu, RedHat, etc.). However, if you want to find these libraries while compiling MiniGUI, you need to install these SDKs. For example, on Ubuntu Linux, FreeType 2, LibPNG, LibJPEG development kits can be installed by executing the following command:

This section is given below in the source code package based on the compiler, install these dependent libraries steps, for reference only.

```
$ sudo apt-get install libfreetype6-dev libpng12-dev libjpeg-dev
```

LibFreeType

The FreeType Library is an open source, high quality, and portable font engine that provides a unified interface for accessing a variety of font format files including TrueType, OpenType, Type1, CID, CFF, Windows FON / FNT, X11 PCF, etc. . MiniGUI uses the FreeType library to render TrueType fonts. Historically, FreeType has two major versions, one is FreeType 1, such as FreeType v1.3.1; the other is FreeType 2, such as FreeType v2.5.2. As mentioned above, MiniGUI can choose to use TrueType font with FreeType 1 or FreeType 2. Currently, FreeType 1 development has been stagnant, while FreeType 2 is the mainstream. Therefore, FreeType 2 should be given priority to support TrueType fonts if there is no special case.

Download the source code package of FreeType 2 from the official website of MiniGUI or the FreeType official website and unzip it into the source directory, then run the following command:

```
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

The FreeType 2 library and header files will be installed in /usr/local directory.

LibJPEG, LibPNG, LibZ and other dependent libraries

The library on which MiniGUI runs depends on libjpeg for JPEG images, libpng for PNG images, and more. Like the FreeType library, these libraries are included in common Linux distributions.

First install the LibZ library. The LibZ library provides the compression and decompression function of the Z algorithm, while the PNG image format uses the same compression algorithm, so before installing and installing LibPNG, first install the LibZ library. Download and unzip LibZ library source code package, and then enter the source root directory, execute the following command:

```
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

The LibZ library and header files will be installed in /usr/local directory.

Download LibPng library source code, untied into the root directory of the source code, execute the following command:

```
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

Download LibJPEG library source code, untied into the root directory of the source code, execute the following command:

```
$ ./configure --prefix=/usr/local --enable-shared
$ make
$ sudo make install
```

The installation process may be prompted to create certain files, then you need to see the directory you want to install there is no corresponding directory, if you do not have to create your own. This JPEG library header files, dynamic libraries and static libraries will be installed to the /usr/local directory.

2.4.2 compile and install the virtual framebuffer program

The default virtual framebuffer graphics engine in MiniGUI 3.0 is pc_xvfb. The graphics engine defines a virtual frame buffer program (XVFB) specification that does not depend on a specific implementation. Under this specification, we can use the gvfb program on Linux Use Gtk+ development), or use the qvfb2 program (developed using Qt) to display the output of MiniGUI and its application in the window of gvfb or qvfb.

gvfb

gvfb is a virtual framebuffer program that is compatible with MiniGUI 3.0 XVFB specification and was developed using Gtk+ 2.0. To compile and install gvfb, to ensure that the system has been installed Gtk+ 2.0 development kits. Under Ubuntu Linux, use the following command to install the appropriate development kit:

```
$ sudo apt-get install libgtk2.0-dev
```

Then enter the gvfb source code directory, run the following command:

```
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

qvfb2

qvfb2 is an upgraded version of qvfb that is compatible with the XVFB specification proposed by MiniGUI 3.0.

To compile qvfb2, you need to install Qt development package, and Qt version needs to be greater than or equal to 3.0.3. Specific installation process can refer to the source code in the README file. Here's an example of the specific process of installing qvfb2 in ubuntu environment.

```
$ sudo apt-get install build-essential xorg-dev
```

Qt3 library and its header files and other related content installation:

```
$ sudo apt-get install libqt3-headers libqt3-mt libqt3-mt-dev
```

Then enter the qvfb2 source code directory, run the following command:

```
$ ./configure --prefix=/usr/local \
                       --with-qt-includes=/usr/include/qt3/ \
                  --with-qt-libraries=/usr/lib/qt3/
$ make
$ sudo make install
```

The --prefix option specifies the installation path for qvfb2; *--with-qt-includes* option specifies the Qt3 header file path; - with-qt-libraries option specifies the Qt3 library file path.

If the above command is successful, then qvfb2 program will be installed to /usr/local/bin directory.

2.4.3 Compiling and Installing MiniGUI in the GNU Development Environment

If you configure MiniGUI with configure script in GNU development environment, you can compile and install MiniGUI with make tool.

For example, assuming that you used MiniGUI for Linux product, in the PC computer for running Linux, you can execute several commands as the following in your MiniGUI source code directory to configure, compile and install MiniGUI to your system.

```
user$ ./configure
user$ make
user$ su -c ‘make install’
```

You can also use configure script to specify a cross-compiling directory and installing directory and so on.

2.4.4 Install MiniGUI Resource Package

MiniGUI resource package (minigui-res) is also organized by GNU autoconf/automake script, so just run the following command to install:

```
user$ ./configure
user$ make
user$ sudo make install
```

Similarly, we can also specify the installation path using the --prefix option.

2.4.5 compile and run MiniGUI sample

After compiling and installing MiniGUI according to the above steps, you can compile and run the sample in mg-samples. By default, MiniGUI will use the pc_xvfb graphics and input engine, and the actual virtual framebuffer is gvfb.

Run the following command to configure and compile the mg-samples code package:

```

user$ sudo ldconfig
user$ ./configure
user$ make
```

The first command to refresh the Linux system dynamic library cache system. Because by default MiniGUI dynamic libraries are installed in the /usr/local/lib directory, the system uses a cache to maintain a list of all the dynamic libraries installed in the system. If the cache is not refreshed, It may not be found installed dynamic library problem.

To run the demo in MiniGUI-Processes runtime mode, you need to start the mginit program first and then run the other sample programs. The following is the process of running the same game in MiniGUI-Processes mode:

```

user$ cd mginit
user$ ./mginit &
user$ cd ../same
user$ ./same
```

On MiniGUI-Threads runtime mode to run the demo program, more simple, direct run sample demo. Here's how to run the same game in thread mode:

```

user$ cd same
user$ ./same
```

2.5 Compiling and Installing MiniGUI in Non-GNU Development Environment

In the Non-GNU development environment (generally, it is Windows platform), we first organize MiniGUI source code solution for project of special Integration Development Environment (for example, Tornado and ADS). Secondly, we compile MiniGUI. At last, we compile MiniGUI application.

But using cygwin development environment for Windows platform, it is very convenient. We can compile and install MiniGUI. In theory, this method is applicable to any development environment, which runs on Windows platform, so we will give detailed description on this method in this chapter.

Cygwin is an open source software project and Linux-like environment for Windows. After installing cygwin on Windows, we can execute many applications of Linux platform, for example, BASH script, VIM editor, PERL script interpreter, make tool of Linux, gcc compiler and so on. In the cygwin environment, we can also call other Windows applications. Thus, if we write makefile for MiniGUI according to GNU rules and use make tool of cygwin to call corresponding compiler and linker, we can compile and generate MiniGUI functions library.

Many OSes (Operating System) development environments include cygwin such as OSE. If there is not cygwin in your development environment, you can download and install it from http://www.cygwin.com. Please make sure you have installed make tool, compiler and BASH shell script software package and so on.

In MiniGUI source code, in order to compile MiniGUI conveniently in the Non-GNU development environment, the following things have been done.

- In order to distinguish makefile of cygwin from GNU makefile, the GNU makefile is generated by configure tool, the makefile of cygwin has **`.ng`** suffix (the **`.ng`** expresses non-GNU).
- Provide template header file for special platform and operating system, the rules of nomenclature is like config-<os>-<platform>.h.
- Provide a self-compiled rule file (the name is **`rules.make`**). The **`rules.make`** is in the MiniGUI source code top directory. In rules.make, we need provide different TARGET_RULES value for different OS development environment.
- Provide some spare **`rules.make`** files for different OS (Operating System) development environment. We save these files to the MiniGUI source code **`build/`** directory. The rules of nomenclature in these files is like **`rules-<platform>.<os>`**.

Firstly, we copy build/ config-<os>-<platform>.h to MiniGUI source code top directory, and rename it as mgconfig.h. Secondly we modify **`rules.make`** file according to actual development environment. Lastly, we compile MiniGUI using cygwin make command. For example, we want to compile MiniGUI for VxWorks X86 platform (rules file corresponding with **`build/rules-pc.vxworks`**[7]), we need follow the following step:

Copy build/config-vxworks-i386.h to MiniGUI source code top directory, and rename it as mgconfig.h (we resume that current directory is MiniGUI source code top directory):

```
cygwin$ cp build/config-vxworks-i386.h  mgconfig.h
```

Modify TARGET_RULES value in rules.make file:

```
TARGET_RULES=build/rules-pc.vxworks
```

Then we compile MiniGUI using make tool of cygwin:

```
cygwin$ /usr/bin/make -f makefile.ng
```

[7] Note that we only provide this file in the VxWorks OS MiniGUI product.

Note that **makefile.ng** supports commands of clean and make. If you execute the command as follow:

```
cygwin$ /usr/bin/make –f makefile.ng install
```

You can install MiniGUI header files and library to the directory, which is specified by **rules-<platform>.<os>**. If you execute the command as the following:

```
cygwin$ /usr/bin/make –f makefile.ng clean
```

You can clean all object files to compile afresh.

Note: if you modify **mgconfig.h** and other files in the cygwin environment, first of all you execute the command above to clean all object files, then compile MiniGUI afresh.

By using cygwin environment and **makefile.ng** to compile MiniGUI, our main work is in editing right **rules.make** file, actually. You must define variables accurately in the table 2.21, when you compile **rules.make** under yourself development environment.

Table 2.21 the variables needed by makefile.ng

Variants name	Purpose	Memo
CC	Specify C compiler	
CPP	Specify C++ compiler	
AR	Specify archiving tool, the tool is used to generate static library	
RANLIB	Specify static library index tool	
MAKE	Specify make tool	Generally, the make tool is /usr/bin/make in the cygwin environment
ARFLAGS	The option that controls the archiving tool generate static library	
COFLAG	The option that it control the compiler to compile, but not link	
OBJ	The suffix name of the object file	
LIBA	The suffix of the static library file	
PREFIX	The prefix of the installation directory	
INCS	Specify the search directory option of head file	
CFLAGS	The C compiler option	

build/rules-pc.vxworks file was listed as follows:

```
# rules for pc-vxworks

AS=
CC=ccpentium
CXX=c++pentium
CPP=ccpentium
AR=arpentium
RANLIB=ranlibpentium
MAKE=/usr/bin/make
```

```
ARFLAGS=crus
COFLAG=-c

OBJ=o
LIBA=a

PREFIX=c:/cross

#vxworks
TARGET_DIR=C:/Tornado2.2x86/target

INCS+=-I${TARGET_DIR}/h

CFLAGS+=-g -mcpu=pentium  -march=pentium -Wall -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -
DCPU=PENTIUM
```

Note that the make tool will install MiniGUI header files to the **`$PREFIX/include/minigui`** directory under the **`makefile.ng`** project file of cygwin, the function libraries were installed to the **`$PREFIX/lib/`** directory. The **`rules.make`** file above will install MiniGUI header files to the **`c:/cross/include/minigui`** directory and MiniGUI libraries to the **`c:/cross/lib`** directory.

Referring to table 2.21 and the **`rules.make`** file above, you can write correct rules.make file based on actually development environment.

Because the format of the **`makefile.ng`** `is` compatible with GNU makefile, so we can use **`makefile.ng`** to compile MiniGUI in the Linux environment, actually. This kind of circumstance usually occurs during using cross-compile tool chain for uClinux. If you work in the Linux environment, you can execute make command.

```
user$ make –f makefile.ng
```

About other contents related with portion and configuration of MiniGUI, please refer to Chapter 18 "*GAL and IAL Engines*" and Appendix A "*A Universal Startup API for RTOSes*" in MiniGUI Programming Guide V3.0-5.

2.6 Use Ubuntu on Windows to configure and compile MiniGUI

Specifically, since Windows 10, Microsoft has reinstated the POSIX-compliant subsystem on the Windows platform with WSL(Windows Subsystem for Linux) as well as an Ubuntu distribution through the Microsoft Store. In this way, we can use the Ubuntu environment running on Windows 10 to configure and compile MiniGUI. This will bring us a lot of convenience because Ubuntu running on Windows is a complete GNU development environment so we can use the GNU *autoconf/automake* script to configure MiniGUI for operating systems like VxWorks and its development environment.

3 MiniGUI runtime configuration options

In this chapter, we describe the MiniGUI runtime configuration options, which effect some actions about MiniGUI running, for example, running GAL and IAL used, device font, bitmap, and cursor etc. It is known that MiniGUI runtime configuration options is loaded from **`MiniGUI.cfg`**, but if compiling MiniGUI with in-core options, the options is included MiniGUI libraries.

MiniGUI.cfg

In GNU development environment, after installing MiniGUI by default configuration, the file **`etc/MiniGUI-classic.cfg`** in MiniGUI source tree will be installed in **`/usr/local/etc/`** directory, and rename to **`MiniGUI.cfg`**. When MiniGUI application starts, It will find MiniGUI.cfg as follow:

- the application first search **`MiniGUI.cfg`** in current directory, then search **`.MiniGUI.cfg`** in home directory.
- then search **`MiniGUI.cfg`** in **`/usr/local/etc`**, at last in **`/etc/`**.
- If user don't create the file **`MiniGUI.cfg`** in current directory and home directory, the application will use the file MiniGUI.cfg in **`/usr/local/etc/`** as default configuration file.

When we compile MiniGUI with **`--enable-incoreres`** option, MiniGUI application doesn't need the file **`MiniGUI.cfg`**. The required options are given in the file **`src/sysres/mgetc.c`**.

Look And Feel Renderer

MiniGUI 3.0 appearance of the window and the control drawing implementation, using a completely different from the previous old version of the implementation mechanism. Previous versions had to be compiled and configured before compilation, the style was chosen, and only one of three styles fashion, classic and flat was chosen. MiniGUI 3.0 uses Look And Feel renderer technology to draw the appearance of windows and controls. MiniGUI support four kinds of renderers, in practical applications choose one. The advantage of the renderer technology is that the appearance can be modified through the MiniGUI.cfg file, and the appearance can also be controlled by API. The user can even customize its own renderer, which provides great convenience for the application to flexibly customize its own appearance based on the actual application environment. For details about MiniGUI renderer interface, please refer to MiniGUI Programming Manual.

MiniGUI Look and Feel divide the window and control attribute to some parts, and then use the drawing interface definition how to draw, forming a complete set of appearance renderer mechanism. MiniGUI 3.0 provides four kinds of renderer: classic, flat, fashion, skin. classic is the default renderer, that is when MiniGUI is initialized, the classic renderer is used to draw windows and controls by default. The fashion renderer needs support by mGPlus component. MiniGUI itself does not provide support for the fashion renderer.

The application can choose to use a particular renderer for a window and define the appearance of the window's elements. Applications can also define their own renderer to draw.

This chapter first describes the runtime configuration options when using configuration files and then describes how to specify runtime configuration options in built-in resources.

Below, we first describe running configuration options with configuration file, and with incore resources.

3.1 Configuration File

The section describes configuration options in detail by **`MiniGUI.cfg`**.

The format of configuration file is compact, and you can modify it easily. The following shows the format.

```
[section-name1]
key-name1=key-value1
key-name2=key-value2

[section-name2]
key-name3=key-value3
key-name4=key-value4
```

The parameters in the configuration file are grouped in sections, such as notation (#), section, key, and key value. The line that the first character is '#' is notation line. The values of the section are specified in the form of **`section-name`**. The values of the key and key value are specified in the form of **`key=value`**. Some important sections are listed as follows.

3.1.1 Section system

The section **`system`** not only defines the graphics engine (**`gal_engine`**) and the input engine (**`ial_engine`**) in runtime MiniGUI, which must be one of engines configured on MiniGUI compiling, but also defines the mouse device (**`mdev`**) and the mouse protocol type (**`mtype`**).

The definition of the keys in section **`system`** is as follows:

- **`gal_engine`**: The graphics engine used.
- **`defaultmode`**: The graphics engine display mode used, its format is widthxheight-bpp.
- **`ial_engine`**: The input engine used.
- **`mdev`**: The mouse device file.
- **`mtype`**: The mouse protocol type.

The contents of the section **`system`** in **`MiniGUI.cfg`** are as follow:

```
[system]
# GAL engine and default options
gal_engine=qvfb
defaultmode=800x600-16bpp

# IAL engine
ial_engine=qvfb

mdev=/dev/input/mice
mtype=IMPS2
```

Since MiniGUI Version 1.6.8, you can modify the graphics and input engine via environment variable. For example, if you define **`fbcon`** and **`qvfb`** graphics engine and

console and **qvfb** input engine, and you choose the qvfb engine in **MiniGUI.cfg** or in-core resources. Then when configure MiniGUI, you can change the engine to fbcon and console in runtime by the following method, and needn't modify **MiniGUI.cfg** or in-core resources configuration file.

```
$ export gal_engine=fbcon
$ export ial_engine=console
$ export mdev=/dev/input/mice
$ export mtype=ps2
$ export defaultmode=1024x768-16bpp
```

3.1.2 Section fbcon

The section **fbcon** is only available when you define the **gal_engine** in section **system** for fbcon. It define default display mode of the **fbcon** engine. When the section is undefined or key value is empty, the fbcon engine using the key value of system section.

The definition of the key in section **fbcon** is as follows:

- **defaultmode**: The display mode of graphics engine used, the format is **widthxheight-bpp**.

The content of the section in **MiniGUI.cfg** is as follows:

```
[fbcon]
defaultmode=1024x768-16bpp
```

3.1.3 Section qvfb

The section **qvfb** is only available when you define the **gal_engine** in section **system** for qvfb. It shows display and display mode of X window used when running qvfb.

The definition of the keys in section **qvfb** is as follows:

- **defaultmode**: The display mode of graphics engine used, its format is **widthxheight-bpp**.
- **display**: Display mode of X window used when running qvfb, default value is 0.

The content of the section in **MiniGUI.cfg** is as follows:

```
[qvfb]
defaultmode=640x480-16bpp
display=0
```

3.1.4 Section pc_xvfb

The section pc_x**vfb** is only available when you define the **gal_engine** in section **system** for pc_xvfb. It has been supported by Linux(Ubuntu) and Window.

The definition of the keys in section pc_x**vfb** is as follows:

- **defaultmode**: The display mode of graphics engine used, its format is **widthxheight-bpp**.
- **window_caption**: Window caption title of XVFB window.
- **exec_file**: gvfb exe file path.

The content of the section in **`MiniGUI.cfg`** is as follows:

```
#{{ifdef _MGGAL_PCXVFB
[pc_xvfb]
defaultmode=800x600-16bpp
window_caption=XVFB-for-MiniGUI-3.0-(Gtk-Version)
exec_file=/usr/local/bin/gvfb
#}}
```

3.1.5 Section rawbitmapfonts, varbitmapfonts, qpf, truetypefonts, and type1fonts

These sections define information of loading `device fonts`, `number` of fonts, and name and file of fonts.

The format of device fonts used by MiniGUI is as follows:

```
<type>-<facename>-<style>-<width>-<height>-<charset1[,charset2,...]>
```

The definitions for each part of device **`font`** are as follow:

- **`<type>`**: The type of device font, for example, RBF, VBF, QPF, TrueType, and Adobe Type1 device font are rbf, vbf, qpf, ttf, and tlf.
- **`<facename>`**: The name of device font. Such as courier, Times etc.
- **`<style>`**: The style of device font, it is grouped into six alphabets. Such as bold, italic, underline or strikethrough etc. Generally the string is "rrncnn".
- **`<width>`**: The width of device font, for var-width fonts set to be maximum width; for vector fonts set to be 0.
- **`<height>`**: The height of device font, for vector fonts set to be 0.
- **`<charset1, charset2>`**: The charset of device font supported.

  Each of these sections defines font_number, name**`<NR>`**, and fontfile**`<NR>`** keys.

- **`font_number`**: The number of device font loaded.
- **`name<NR>`**: The name of device font that number is **`<NR>`**.
- **`fontfile<NR>`**: The font file of device font that number is **`<nr>`**.

If you don't need to use a specific type of device font, you can skip the configuration option by set **`font_number = 0`**.

The content of these sections in **`MiniGUI.cfg`** are as follow:

```
[rawbitmapfonts]
font_number=4
name0=rbf-fixed-rrncnn-8-16-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/8x16-iso8859-1.bin
name1=rbf-fixed-rrncnn-16-16-GB2312-0
fontfile1=/usr/local/lib/minigui/res/font/song-16-gb2312.bin
name2=rbf-fixed-rrncnn-6-12-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/6x12-iso8859-1.bin
name3=rbf-fixed-rrncnn-12-12-GB2312-0
fontfile3=/usr/local/lib/minigui/res/font/song-12-gb2312.bin

[varbitmapfonts]
font_number=6
name0=vbf-Courier-rrncnn-8-13-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/Courier-rr-8-13.vbf
name1=vbf-Helvetica-rrncnn-11-12-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/Helvetica-rr-11-12.vbf
name2=vbf-Times-rrncnn-10-12-ISO8859-1
```

```
fontfile2=/usr/local/lib/minigui/res/font/Times-rr-10-12.vbf
name3=vbf-Courier-rrncnn-10-15-ISO8859-1
fontfile3=/usr/local/lib/minigui/res/font/Courier-rr-10-15.vbf
name4=vbf-Helvetica-rrncnn-15-16-ISO8859-1
fontfile4=/usr/local/lib/minigui/res/font/Helvetica-rr-15-16.vbf
name5=vbf-Times-rrncnn-13-15-ISO8859-1
fontfile5=/usr/local/lib/minigui/res/font/Times-rr-13-15.vbf

[upf]
font_number=0

[qpf]
font_number=3
name0=qpf-unifont-rrncnn-16-16-ISO8859-1,ISO8859-15,GB2312-0,GBK,BIG5
fontfile0=/usr/local/lib/minigui/res/font/unifont_160_50.qpf
name1=qpf-times-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile1=/usr/local/lib/minigui/res/font/smoothtimes_100_50.qpf
name2=qpf-helvetica-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile2=/usr/local/lib/minigui/res/font/helvetica_100_50.qpf
name3=qpf-micro-rrncnn-4-4-ISO8859-1,ISO8859-15
fontfile3=/usr/local/lib/minigui/res/font/micro_40_50.qpf

[truetypefonts]
font_number=3
name0=ttf-arial-rrncnn-0-0-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/arial.ttf
name1=ttf-times-rrncnn-0-0-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/times.ttf
name2=ttf-pinball-rrncnn-0-0-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/pinball.ttf
```

3.1.5 Section systemfont

The section **`systemfont`** defines MiniGUI system font and font number, and defines system default font, which would be used to render text on captions, menus, and controls, as well as the default font of a window.

System font is the logic font[10] that is created by the function **`CreateLogFontFromName`** based on device fonts, which is defined by MiniGUI sections such as **`rawbitmapfonts`**, **`varbitmapfonts`**, **`qpf`**, **`truetypefonts`**, and **`t1fonts`**.

The content of the section in **`MiniGUI.cfg`** is as follows:

```
<type>-<facename>-<style>-<width>-<height>-<charset1>
```

The definition of each part of a logic font name is as follows:

- **`<type>`** is the desired device font type, if you do not want to specify it, use **`*`**.
- **`<facename>`** is to define the font face name, such as courier and times etc.
- **`<style>`** is the string of six alphabets to define style of a logic font, such as italic, bold, underline or strikethrough etc.
- **`<width>`** is to define the width of the logic font. Usually do not need to specify, use **`*`** instead.
- **`<height>`** is to define the height of the logic font.
- **`<charset>`** is to define charset of the logic font being created.

Furthermore, MiniGUI V2.0.x provides auto-scaling the font glyph. If you want to use this function, you only need use ‘S’ in forth character when you define logical font styles. Note that you don’t need to use this style when you use vector font, such as TrueType, because vector font can produce corresponding font glyph according to desired logical font size.

The definition of the keys in section **`systemfont`** is as follows:

- **`font_number`**: The number of system fonts created
- **`font<NR>`**: The number <NR> logical font name
- **`default`**: System default font(single character set). Its value is the number of logical font.
- **`wchar_def`**: Default font used by multiple character set. Its value is the number of above logical font.
- **`fixed`**: The font used by fixed width character set. Its value is the number of above logical font.
- **`caption`**: The caption font. Its value is the number of above logical font.
- **`menu`**: The menu font. Its value is the number of above logical font.

You can change the number of system font created. But you must create a single character set (for example: ISO8859-1) at least. MiniGUI defines the system default charsets according to **`default`**, **`wchar_def`** system fonts, and this would affect the return value of **`GetSysCharset`**, **`GetSysCharWidth`**, **`GetSysCCharWidth`** and **`GetSysHeight`** functions. Commonly, **`default`** and **`wchar_def`** `must fixed width dot-matrix font, i.e RBF. And the width of multiply character set must be twice with the width of single character set.`

The content of the section in **`MiniGUI.cfg`** is as follows:

```
# The first system font must be a logical font using RBF device font.
[systemfont]
font_number=6
font0=rbf-fixed-rrncnn-8-16-ISO8859-1
font1=*-fixed-rrncnn-*-16-GB2312
font2=*-Courier-rrncnn-*-16-GB2312
font3=*-SansSerif-rrncnn-*-16-GB2312
font4=*-Times-rrncnn-*-16-GB2312
font5=*-Helvetica-rrncnn-*-16-GB2312

default=0
wchar_def=1
fixed=1
caption=2
menu=3
control=3
```

3.1.6 Section mouse

The section **`mouse`** defines the time of mouse **`double clicked`**. It is used to handle with system inner events. Generally, it is unnecessary changed.
The definition of the keys in the section is as follows:

- **`dblclicktime`**: The mouse double clicked time in ms

The content of the section in **`MiniGUI.cfg`** is as follows:

```
[mouse]
dblclicktime=300
```

3.1.7 Section event

The section **`event`** defines event timeout and auto-repeat time used by system internal event process. Generally, it is unnecessary changed.
The definition of the keys in the section is as follows:

- **`timeoutusec`**: Event timeout time in ms
- **`repeatusec`**: Event repeat time in ms

The content of the section in **`MiniGUI.cfg`** is as follows:

```
timeoutusec=300000
repeatusec=50000
```

3.1.8 Section classic

The section **`classic`** defines default window element color used. Generally, it is unnecessary changed.

Table 3.1 window element division and name in the configuration file and code

Configure Option	Code name	Comment
caption	WE_METRICS_CAPTION	Caption size
	WE_FONT_CAPTION	Caption fonts
fgc_active_caption	WE_FGC_ACTIVE_CAPTION	Focus status caption foreground color
bgca_active_caption	WE_BGCA_ACTIVE_CAPTION	Focus status caption background color gradient starting color
bgcb_active_caption	WE_BGCB_ACTIVE_CAPTION	Focus Status caption background color gradient ending Color
fgc_inactive_caption	WE_FGC_INACTIVE_CAPTION	Non-focus status caption foreground color
bgca_inactive_caption	WE_BGCA_INACTIVE_CAPTION	Non-focus status caption background color gradient starting color
bgcb_inactive_caption	WE_BGCB_INACTIVE_CAPTION	Non-focus status caption background color gradient ending color
menu	WE_METRICS_MENU	Menu item, height of the menu bar
	WE_FONT_MENU	Menu font
fgc_menu	WE_FGC_MENU	Menu foreground color
bgc_menu	WE_BGC_MENU	Menu background color
border	WE_METRICS_WND_BORDER	Window border width
fgc_active_border	WE_FGC_ACTIVE_WND_BORDER	Focus status window border color
fgc_inactive_border	WE_FGC_INACTIVE_WND_BORDER	Non-focus status window border color
scrollbar	WE_METRICS_SCROLLBAR	Scroll bar size
fgc_msgbox	WE_FGC_MESSAGEBOX	Message box foreground color
fgc_msgbox	WE_FONT_MESSAGEBOX	Message box font
fgc_tip	WE_FGC_TOOLTIP	Prompt box foreground color
bgc_tip	WE_BGC_TOOLTIP	Prompt box background color
	WE_FONT_TOOLTIP	Prompt box font
fgc_window	WE_FGC_WINDOW	Window foreground
bgc_window	WE_BGC_WINDOW	Window background color
fgc_3dbox	WE_FGC_THREED_BODY	The color of the symbol on the surface of the 3D box, such as the color of check mark, arrow, etc.
mainc_3dbox	WE_MAINC_THREED_BODY	Three-dimensional box border and surface color

fgc_selected_item	WE_FGC_SELECTED_ITEM	The foreground color of the selected menu item (list item)
bgc_selected_item	WE_BGC_SELECTED_ITEM	The background color of the selected menu item (list item)
bgc_selected_lostfocus	WE_BGC_SELECTED_LOSTFOCUS	The background color after the selected menu item (list item) loses focus
fgc_disabled_item	WE_FGC_DISABLED_ITEM	Foreground color of invalid menu item (list item)
bgc_disabled_item	WE_BGC_DISABLED_ITEM	Invalid menu item (list item) background color
fgc_hilight_item	WE_FGC_HIGHLIGHT_ITEM	Highlight the foreground color of the menu item (list item)
bgc_hilight_item	WE_BGC_HIGHLIGHT_ITEM	Highlight the background color of the menu item (list item)
fgc_significant_item	WE_FGC_SIGNIFICANT_ITEM	Foreground color of important menu item (list item)
bgc_significant_item	WE_BGC_SIGNIFICANT_ITEM	Background color of important menu items (list items)
bgc_desktop	WE_BGC_DESKTOP	Desktop background color

The content of the section in **`MiniGUI.cfg`** is as follows:

```
[classic]
# Note that max number defined in source code is 5.
iconnumber=5
icon0=form.ico
icon1=failed.ico
icon2=help.ico
icon3=warning.ico
icon4=excalmatory.ico

# default icons for new OpenFileDialogBox
dir=folder.ico
file=textfile.ico

# default icons for TreeView control
treefold=fold.ico
treeunfold=unfold.ico

# bitmap used by BUTTON control
radiobutton=classic_radio_button.bmp
checkbutton=classic_check_button.bmp

# background picture, use your favirate photo
bgpicture=none
bgpicpos=center

#window element metrics
caption=20
menu=25
border=2
scrollbar=16

#window element colors
fgc_active_caption=0xFFFFFFFF
bgca_active_caption=0xFF6A240A
bgcb_active_caption=0xFF6A240A
```

```
fgc_menu=0xFF000000
bgc_menu=0xFFCED3D6


fgc_msgbox=0xFF000000

fgc_tip=0xFF000000
bgc_tip=0xFFE7FFFF

fgc_active_border=0xFFCED3D6
fgc_inactive_border=0xFFCED3D6

fgc_inactive_caption=0xFFC8D0D4
bgca_inactive_caption=0xFF808080
bgcb_inactive_caption=0xFF808080

fgc_window=0xFF000000
bgc_window=0xFFFFFFFF

fgc_3dbox=0xFF000000
mainc_3dbox=0xFFCED3D6

fgc_selected_item=0xFFFFFFFF
bgc_selected_item=0xFF6B2408
bgc_selected_lostfocus=0xFFBDA69C

fgc_disabled_item=0xFF848284
bgc_disabled_item=0xFFCED3D6

fgc_hilight_item=0xFFFFFFFF
bgc_hilight_item=0xFF6B2408

fgc_significant_item=0xFFFFFFFF
bgc_significant_item=0xFF6B2408

bgc_desktop=0xFFC08000
```

3.1.9 Default Configuration File

Below is the default runtime configuration file for MiniGUI library:

```

# MiniGUI Ver 3.0.x
# This configuration file is for classic window style.
#
# Copyright (C) 2002~2017 Feynman Software
# Copyright (C) 1998~2002 Wei Yongming.
#
# Web:   http://www.minigui.com
# Web:   http://www.minigui.org
#
# This configuration file must be installed in /etc,
# /usr/local/etc or your home directory. When you install it in your
# home directory, it should be named ".MiniGUI.cfg".
#
# The priority of above configruation files is ~/.MiniGUI.cfg,
# /usr/local/etc/MiniGUI.cfg, and then /etc/MiniGUI.cfg.
#
# If you change the install path of MiniGUI resource, you should
# modify this file to meet your configuration.
#
# NOTE:
```

```
# The format of this configuration file has changed since the last release.
# Please DONT forget to provide the latest MiniGUI.cfg file for your MiniGUI.
#

[system]
# GAL engine and default options
gal_engine=pc_xvfb
defaultmode=800x600-16bpp

# IAL engine
ial_engine=pc_xvfb
mdev=/dev/input/mice
mtype=IMPS2

[fbcon]
defaultmode=1024x768-16bpp

[qvfb]
defaultmode=640x480-16bpp
display=0

#{{ifdef _MGGAL_PCXVFB
[pc_xvfb]
defaultmode=800x600-16bpp
window_caption=XVFB-for-MiniGUI-3.0-(Gtk-Version)
exec_file=/usr/local/bin/gvfb
#}}

# The first system font must be a logical font using RBF device font.
[systemfont]
font_number=6
font0=rbf-fixed-rrncnn-8-16-ISO8859-1
font1=*-fixed-rrncnn-*-16-GB2312
font2=*-Courier-rrncnn-*-16-GB2312
font3=*-SansSerif-rrncnn-*-16-GB2312
font4=*-Times-rrncnn-*-16-GB2312
font5=*-Helvetica-rrncnn-*-16-GB2312

default=0
wchar_def=1
fixed=1
caption=2
menu=3
control=3

[rawbitmapfonts]
font_number=4
name0=rbf-fixed-rrncnn-8-16-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/8x16-iso8859-1.bin
name1=rbf-fixed-rrncnn-16-16-GB2312-0
fontfile1=/usr/local/lib/minigui/res/font/song-16-gb2312.bin
name2=rbf-fixed-rrncnn-6-12-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/6x12-iso8859-1.bin
name3=rbf-fixed-rrncnn-12-12-GB2312-0
fontfile3=/usr/local/lib/minigui/res/font/song-12-gb2312.bin

[varbitmapfonts]
font_number=6
name0=vbf-Courier-rrncnn-8-13-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/Courier-rr-8-13.vbf
name1=vbf-Helvetica-rrncnn-11-12-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/Helvetica-rr-11-12.vbf
```

```
name2=vbf-Times-rrncnn-10-12-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/Times-rr-10-12.vbf
name3=vbf-Courier-rrncnn-10-15-ISO8859-1
fontfile3=/usr/local/lib/minigui/res/font/Courier-rr-10-15.vbf
name4=vbf-Helvetica-rrncnn-15-16-ISO8859-1
fontfile4=/usr/local/lib/minigui/res/font/Helvetica-rr-15-16.vbf
name5=vbf-Times-rrncnn-13-15-ISO8859-1
fontfile5=/usr/local/lib/minigui/res/font/Times-rr-13-15.vbf

[qpf]
font_number=3
name0=qpf-unifont-rrncnn-16-16-ISO8859-1,ISO8859-15,GB2312-0,GBK,BIG5
fontfile0=/usr/local/lib/minigui/res/font/unifont_160_50.qpf
name1=qpf-times-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile1=/usr/local/lib/minigui/res/font/smoothtimes_100_50.qpf
name2=qpf-helvetica-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile2=/usr/local/lib/minigui/res/font/helvetica_100_50.qpf
name3=qpf-micro-rrncnn-4-4-ISO8859-1,ISO8859-15
fontfile3=/usr/local/lib/minigui/res/font/micro_40_50.qpf

[upf]
font_number=0

[truetypefonts]
font_number=3
name0=ttf-arial-rrncnn-0-0-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/arial.ttf
name1=ttf-times-rrncnn-0-0-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/times.ttf
name2=ttf-pinball-rrncnn-0-0-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/pinball.ttf

[mouse]
dblclicktime=300

[event]
timeoutusec=300000
repeatusec=50000

[cursorinfo]
# Edit following line to specify cursor files path
cursorpath=/usr/local/lib/minigui/res/cursor/
cursornumber=23
cursor0=d_arrow.cur
cursor1=d_beam.cur
cursor2=d_pencil.cur
cursor3=d_cross.cur
cursor4=d_move.cur
cursor5=d_sizenwse.cur
cursor6=d_sizenesw.cur
cursor7=d_sizewe.cur
cursor8=d_sizens.cur
cursor9=d_uparrow.cur
cursor10=d_none.cur
cursor11=d_help.cur
cursor12=d_busy.cur
cursor13=d_wait.cur
cursor14=g_rarrow.cur
cursor15=g_col.cur
cursor16=g_row.cur
cursor17=g_drag.cur
cursor18=g_nodrop.cur
```

```
cursor19=h_point.cur
cursor20=h_select.cur
cursor21=ho_split.cur
cursor22=ve_split.cur
[resinfo]
respath=/usr/local/share/minigui/res/

[classic]
# Note that max number defined in source code is 5.
iconnumber=5
icon0=form.ico
icon1=failed.ico
icon2=help.ico
icon3=warning.ico
icon4=excalmatory.ico

# default icons for new OpenFileDialogBox
dir=folder.ico
file=textfile.ico

# default icons for TreeView control
treefold=fold.ico
treeunfold=unfold.ico

# bitmap used by BUTTON control
radiobutton=classic_radio_button.bmp
checkbutton=classic_check_button.bmp

# background picture, use your favirate photo
bgpicture=none
bgpicpos=center
# bgpicpos=upleft
# bgpicpos=downleft
# bgpicpos=upright
# bgpicpos=downright
# bgpicpos=upcenter
# bgpicpos=downcenter
# bgpicpos=vcenterleft
# bgpicpos=vcenterright
# bgpicpos=none

#window element metrics
caption=20
menu=25
border=2
scrollbar=16

#window element colors
fgc_active_caption=0xFFFFFFFF
bgca_active_caption=0xFF6A240A
bgcb_active_caption=0xFF6A240A

fgc_menu=0xFF000000
bgc_menu=0xFFCED3D6


fgc_msgbox=0xFF000000

fgc_tip=0xFF000000
bgc_tip=0xFFE7FFFF

fgc_active_border=0xFFCED3D6
```

```
fgc_inactive_border=0xFFCED3D6

fgc_inactive_caption=0xFFC8D0D4
bgca_inactive_caption=0xFF808080
bgcb_inactive_caption=0xFF808080

fgc_window=0xFF000000
bgc_window=0xFFFFFFFF

fgc_3dbox=0xFF000000
mainc_3dbox=0xFFCED3D6

fgc_selected_item=0xFFFFFFFF
bgc_selected_item=0xFF6B2408
bgc_selected_lostfocus=0xFFBDA69C

fgc_disabled_item=0xFF848284
bgc_disabled_item=0xFFCED3D6

fgc_hilight_item=0xFFFFFFFF
bgc_hilight_item=0xFF6B2408

fgc_significant_item=0xFFFFFFFF
bgc_significant_item=0xFF6B2408

bgc_desktop=0xFFC08000

#{{ifdef _MGLF_RDR_FLAT
[flat]
# Note that max number defined in source code is 5.
iconnumber=5
icon0=form-flat.ico
icon1=failed-flat.ico
icon2=help-flat.ico
icon3=warning-flat.ico
icon4=excalmatory-flat.ico

# default icons for new OpenFileDialogBox
dir=folder-flat.ico
file=textfile-flat.ico

# default icons for TreeView control
treefold=fold-flat.ico
treeunfold=unfold-flat.ico

# bitmap used by BUTTON control
radiobutton=flat_radio_button.bmp
checkbutton=flat_check_button.bmp

# background picture, use your favirate photo
bgpicture=none
bgpicpos=center

#window element metrics
caption=20
menu=25
border=1
scrollbar=16

#window element colors
fgc_active_caption=0xFFFFFFFF
bgca_active_caption=0xFF000000
```

```
bgcb_active_caption=0xFF000000

fgc_inactive_caption=0xFF000000
bgca_inactive_caption=0xFFFFFFFF
bgcb_inactive_caption=0xFFFFFFFF

fgc_menu=0xFF000000
bgc_menu=0xFFD8D8D8

fgc_msgbox=0xFF000000

fgc_tip=0xFF000000
bgc_tip=0xFFE7FFFF

fgc_active_border=0xFF000000
fgc_inactive_border=0xFF848284

fgc_window=0xFF000000
bgc_window=0xFFFFFFFF

fgc_3dbox=0xFF000000
mainc_3dbox=0xFFFFFFFF

fgc_selected_item=0xFFFFFFFF
bgc_selected_item=0xFF000000
bgc_selected_lostfocus=0xFFBDA69C

fgc_disabled_item=0xFF848284
bgc_disabled_item=0xFF000000

fgc_hilight_item=0xFFFFFFFF
bgc_hilight_item=0xFF664E4A

fgc_significant_item=0xFFFFFFFF
bgc_significant_item=0xFF000000

bgc_desktop=0xFFC08000

flat_tab_normal_color=0xFFC6D2CF
#}}

#{{ifdef _MGLF_RDR_SKIN
[skin]
# Note that max number defined in source code is 5.
iconnumber=5
icon0=form.ico
icon1=failed.ico
icon2=help.ico
icon3=warning.ico
icon4=excalmatory.ico

# default icons for new OpenFileDialogBox
dir=folder.ico
file=textfile.ico

# default icons for TreeView control
treefold=fold.ico
treeunfold=unfold.ico

# background picture, use your favirate photo
bgpicture=none
bgpicpos=center
```

```
#window element metrics
caption=25
menu=25
border=1
scrollbar=17

fgc_active_caption=0xFFFFFFFF
bgca_active_caption=0xFFE35400
bgcb_active_caption=0xFF686868

fgc_menu=0xFF000000
bgc_menu=0xFFD4D6FF

fgc_msgbox=0xFF000000

fgc_tip=0xFF000000
bgc_tip=0xFFFFFFFF

fgc_active_border=0xFFC8D0D4
fgc_inactive_border=0xFFC8D0D4

fgc_inactive_caption=0xFFF8E4D8
bgca_inactive_caption=0xFFDF967A
bgcb_inactive_caption=0xFF686868

fgc_window=0xFF000000
bgc_window=0xFFFFFFFF

fgc_3dbox=0xFF000000
mainc_3dbox=0xFFD8E9EC

fgc_selected_item=0xFFFFFFFF
bgc_selected_item=0xFFC56A31
bgc_selected_lostfocus=0xFFD8E9EC

fgc_disabled_item=0xFF99A8AC
bgc_disabled_item=0xFFFFFFFF

fgc_hilight_item=0xFFFFFFFF
bgc_hilight_item=0xFFC56A31

fgc_significant_item=0xFFFFFFFF
bgc_significant_item=0xFFC56A31

bgc_desktop=0xFF984E00

skin_bkgnd=skin_bkgnd.bmp
skin_caption=skin_caption.gif
skin_caption_btn=skin_cpn_btn.gif

#for scrollbar
skin_scrollbar_hshaft=skin_sb_hshaft.bmp
skin_scrollbar_vshaft=skin_sb_vshaft.bmp
skin_scrollbar_hthumb=skin_sb_hthumb.bmp
skin_scrollbar_vthumb=skin_sb_vthumb.bmp
skin_scrollbar_arrows=skin_sb_arrows.bmp

#for border
skin_tborder=skin_tborder.bmp
skin_bborder=skin_bborder.bmp
skin_lborder=skin_lborder.bmp
```

```
skin_rborder=skin_rborder.bmp

skin_arrows=skin_arrows.gif
skin_arrows_shell=skin_arrows_shell.bmp

skin_pushbtn=skin_pushbtn.gif
skin_radiobtn=skin_radiobtn.gif
skin_checkbtn=skin_checkbtn.bmp

#for treeview
skin_tree=skin_tree.bmp

skin_header=skin_header.bmp
skin_tab=skin_tab.gif

#for trackbar
skin_tbslider_h=skin_tbslider_h.gif
skin_tbslider_v=skin_tbslider_v.gif
skin_trackbar_horz=skin_tb_horz.gif
skin_trackbar_vert=skin_tb_vert.gif

#for progressbar
skin_progressbar_htrack=skin_pb_htrack.gif
skin_progressbar_vtrack=skin_pb_vtrack.gif
skin_progressbar_hchunk=skin_pb_htruck.bmp
skin_progressbar_vchunk=skin_pb_vtruck.bmp
#}}


[fashion]
# Note that max number defined in source code is 5.
iconnumber=5
icon0=form.ico
icon1=failed.ico
icon2=help.ico
icon3=warning.ico
icon4=excalmatory.ico

# default icons for new OpenFileDialogBox
dir=folder.ico
file=textfile.ico

# default icons for TreeView control
treefold=fold.ico
treeunfold=unfold.ico

# bitmap used by BUTTON control
radiobutton=fashion_radio_btn.bmp
checkbutton=fashion_check_btn.bmp

# background picture, use your favirate photo
bgpicture=none
bgpicpos=center

#window element metrics
caption=25
menu=25
border=1
scrollbar=17

fgc_active_caption=0xFFFFFFFF
bgca_active_caption=0xFFE35400
```

```
bgcb_active_caption=0xFFFF953D

fgc_menu=0xFF000000
bgc_menu=0xFFFFE4BF

fgc_msgbox=0xFF000000

fgc_tip=0xFF000000
bgc_tip=0xFFFFFFFF

fgc_active_border=0xFFC8D0D4
fgc_inactive_border=0xFFC8D0D4

fgc_inactive_caption=0xFFF8E4D8
bgca_inactive_caption=0xFFDF967A
bgcb_inactive_caption=0xFFEBB99D

fgc_window=0xFF000000
bgc_window=0xFFEBB99D

fgc_3dbox=0xFF000000
mainc_3dbox=0xFFD8E9EC

fgc_selected_item=0xFFFFFFFF
bgc_selected_item=0xFFC56A31
bgc_selected_lostfocus=0xFFD8E9EC

fgc_disabled_item=0xFF99A8AC
bgc_disabled_item=0xFFFFFFFF

fgc_hilight_item=0xFFFFFFFF
bgc_hilight_item=0xFFC56A31

fgc_significant_item=0xFFFFFFFF
bgc_significant_item=0xFFC56A31

bgc_desktop=0xFF984E00
```

3.2 Incore Configuration Options

When use incore resources, MiniGUI don't need the file **`MiniGUI.cfg`**. The appropriate configuration options are defined in the file **`src/sysres/mgetc.c`**.

Similar with the structure in **`MiniGUI.cfg`**, MiniGUI defines an structure ETCSECTION, array **`_etc_sections`** and variable MGETC in **`mgetc.c`**. The array mgetc_sections is appropriate with section in configuration file. MGETC that is ETC_S type is appropriate with configuration file.

3.2.1 Structure ETCSETCTION

The structure **`ETCSECTION`** is defined in the file named '**`minigui.h`**'. The following is in detail.

```

/** Etc The current config section information */
typedef struct _ETCSECTION
{
    /** Allocated number of keys */
    int key_nr_alloc;
    /** Key number in the section */
    int key_nr;
    /** Name of the section */
    char *name;
    /** Array of keys */
    char** keys;
    /** Array of values */
    char** values;
} ETCSECTION;
```

The **key_nr_alloc** is the interface of other configuration options. Its value must be 0 in incore. The **key_nr** defines the number of the key in section. The name defines the name of section. The keys and values is the array of key and value. The number of key array and value array is corresponded with the number of the **key_nr**.

Below is the definition of _etc_sections in the **mgetc.c** file.

```
static ETCSECTION _etc_sections [] = {
    {0, 5, "system", _system_keys,_system_values },
    {0, 1, "fbcon", _fbcon_keys,_fbcon_values },
    {0, 2, "qvfb", _qvfb_keys,_qvfb_values },
#ifdef _MGGAL_PCXVFB
    {0, 3, "pc_xvfb", _pc_xvfb_keys,_pc_xvfb_values },
#endif
    {0, 1, "rtos_xvfb", _rtos_xvfb_keys,_rtos_xvfb_values },
#ifdef _MGGAL_SHADOW
    {0, 3, "shadow", _shadow_keys,_shadow_values },
#endif
#ifdef _MGGAL_MLSHADOW
    {0, 4, "mlshadow", _mlshadow_keys,_mlshadow_values },
#endif
    {0, 12, "systemfont", _systemfont_keys,_systemfont_values },
    {0, 1, "rawbitmapfonts", _rawbitmapfonts_keys,_rawbitmapfonts_values },
    {0, 1, "varbitmapfonts", _varbitmapfonts_keys,_varbitmapfonts_values },
    {0, 1, "upf", _upf_keys,_upf_values },
    {0, 1, "qpf", _qpf_keys,_qpf_values },
    {0, 1, "truetypefonts", _truetypefonts_keys,_truetypefonts_values },
    {0, 1, "mouse", _mouse_keys,_mouse_values },
    {0, 2, "event", _event_keys,_event_values },
    {0, 25, "cursorinfo", _cursorinfo_keys,_cursorinfo_values },
    {0, 1, "resinfo", _resinfo_keys,_resinfo_values },
    {0, 45, "classic", _classic_keys,_classic_values },
#ifdef _MGLF_RDR_FLAT
    {0, 46, "flat", _flat_keys,_flat_values },
#endif
#ifdef _MGLF_RDR_SKIN
    {0, 71, "skin", _skin_keys,_skin_values },
#endif
    {0, 45, "fashion", _fashion_keys,_fashion_values }
};
```

The section in **_etc_sections** must be defined (fbcon or qvfb is optional.). Other notation sections are optional. The meaning of sections is same as the sections in MiniGUI.cfg. Commonly, you can only change the GAL engine, the IAL engine, display mode and the sections of system and fbcon: SYSTEM_VALUES and FBCON_VALUES defined in the **mgetc-xxx.c** file, such as **mgetc-pc.c**.

The **systemfont** section defines incore font used by system. Currently, MiniGUI 3.0.x supports ISO8859-1, GB2312, RBF, BIG5, SHIFT_JIS, and QPF. MiniGUI doesn’t support the TTF and Type1 font in incore resources.

3.2.2 ETC_S Structure

ETC_S structure was defined in the file minigui.h, the content of ETC_S listed as the follow:

```
/** ETC_S The current config file information*/
typedef struct _ETC_S
{
    /** Allocated number of sections */
    int sect_nr_alloc;
    /** Number of sections */
    int section_nr;
    /** Pointer to section arrays */
    PETCSECTION sections;
} ETC_S;
```

Therefore, **sect_nr_alloc** is the interface of the other configuration options, it's value must be 0 in incore, **sect_nr** specify the number of section, sections is ETCSECTION type structure array, the number of item is not less than the value, the first item specified this value.

The **mgetc_sections** array was defined as the follow in the **mgetc.c** file.

```
static ETC_S _ETC = {
    0,
    sizeof(_etc_sections)/sizeof(ETCSECTION),
    _etc_sections
};
```

The number of section is sizeof(_etc_sections)/sizeof(ETCSECTION) in the MGETC structure; the section array is mgetc_sections array above.

3.2.3 Listing of mgetc.c

```
/************************************************
* This is inside mode of system res configuation *
* It's generated by the mgcfg-trans, version 1.0 *
* author : dongjunjie in feynman                 *
* please donnot modify this file, if you want,   *
* please change your input file and regenerate   *
* this file                                      *
************************************************/
#include <stdio.h>
#include "common.h"
#include "minigui.h"

#ifdef _MGINCORE_RES

// This configuration file is for MiniGUI V3.0.x
//
// Copyright (C) 2002~2008 Feynman Software
// Copyright (C) 1998~2002 Wei Yongming.
//
// Web: http://www.minigui.com
//
// This configuration file must be installed in /etc,
// /usr/local/etc or your home directory. When you install it in your
// home directory, it should be named ".MiniGUI.cfg".
//
// The priority of above configruation files is ~/.MiniGUI.cfg,
// /usr/local/etc/MiniGUI.cfg, and then /etc/MiniGUI.cfg.
//
// If you change the install path of MiniGUI resource, you should
// modify this file to meet your configuration.
//
// NOTE:
// The format of this configuration file has changed since the last release.
// Please DONT forget to provide the latest MiniGUI.cfg file for your MiniGUI.
//
// Section: system
static char* _system_keys[]={
// GAL engine and default options
    "gal_engine",
    "defaultmode",
// IAL engine
    "ial_engine",
    "mdev",
    "mtype"
};
static char* _system_values[]={
// GAL engine and default options
    "pc_xvfb",
    "800x600-16bpp",
// IAL engine
    "pc_xvfb",
    "/dev/input/mice",
    "IMPS2"
};
// Section: fbcon
static char* _fbcon_keys[]={
    "defaultmode"
};
static char* _fbcon_values[]={
    "1024x768-16bpp"
};
// Section: qvfb
static char* _qvfb_keys[]={
    "defaultmode",
    "display"
};
static char* _qvfb_values[]={
    "640x480-16bpp",
    "0"
};
#ifdef _MGGAL_PCXVFB
// Section: pc_xvfb
static char* _pc_xvfb_keys[]={
    "defaultmode",
    "window_caption",
    "exec_file"
};
static char* _pc_xvfb_values[]={
    "800x600-16bpp",
    "XVFB-for-MiniGUI-3.0-(Gtk-Version)",
    "/usr/local/bin/gvfb"
};
#endif
// Section: rtos_xvfb
static char* _rtos_xvfb_keys[]={
    "defaultmode"
};
static char* _rtos_xvfb_values[]={
    "800x600-16bpp"
};
#ifdef _MGGAL_SHADOW
// Section: shadow
static char* _shadow_keys[]={
    "real_engine",
    "defaultmode",
    "rotate_screen"
};
static char* _shadow_values[]={
    "pc_xvfb",
    "800x600-16bpp",
    "normal"
};
#endif
#ifdef _MGGAL_MLSHADOW
// Section: mlshadow
static char* _mlshadow_keys[]={
    "real_engine",
    "defaultmode",
    "def_bgcolor",
    "double_buffer"
};
static char* _mlshadow_values[]={
    "qvfb",
    "800x600-16bpp",
    "0x00FF00",
    "enable"
};
#endif
// The first system font must be a logical font using RBF device font.
// Section: systemfont
static char* _systemfont_keys[]={
    "font_number",
    "font0",
    "font1",
    "font2",
    "font3",
    "font4",
    "default",
    "wchar_def",
    "fixed",
    "caption",
    "menu",
    "control"
};
static char* _systemfont_values[]={
    "5",
    "rbf-FixedSys-rrncnn-8-16-ISO8859-1",
    "*-FixedSys-rrncnn-*-16-ISO8859-1",
    "*-Courier-rrncnn-*-16-ISO8859-1",
    "*-SansSerif-rrncnn-*-16-ISO8859-1",
    "*-System-rrncnn-*-16-ISO8859-1",
    "0",
    "4",
    "1",
    "4",
    "2",
    "3"
```

```
};
// Section: rawbitmapfonts
static char* _rawbitmapfonts_keys[]={
    "font_number"
};
static char* _rawbitmapfonts_values[]={
    "0"
};
// Section: varbitmapfonts
static char* _varbitmapfonts_keys[]={
    "font_number"
};
static char* _varbitmapfonts_values[]={
    "0"
};
// Section: upf
static char* _upf_keys[]={
    "font_number"
};
static char* _upf_values[]={
    "0"
};
// Section: qpf
static char* _qpf_keys[]={
    "font_number"
};
static char* _qpf_values[]={
    "0"
};
// Section: truetypefonts
static char* _truetypefonts_keys[]={
    "font_number"
};
static char* _truetypefonts_values[]={
    "0"
};
// Section: mouse
static char* _mouse_keys[]={
    "dblclicktime"
};
static char* _mouse_values[]={
    "300"
};
// Section: event
static char* _event_keys[]={
    "timeoutusec",
    "repeatusec"
};
static char* _event_values[]={
    "300000",
    "50000"
};
// Section: cursorinfo
static char* _cursorinfo_keys[]={
// Edit following line to specify cursor files path
    "cursorpath",
    "cursornumber",
    "cursor0",
    "cursor1",
    "cursor2",
    "cursor3",
    "cursor4",
    "cursor5",
    "cursor6",
    "cursor7",
    "cursor8",
    "cursor9",
    "cursor10",
    "cursor11",
    "cursor12",
    "cursor13",
    "cursor14",
    "cursor15",
    "cursor16",
    "cursor17",
    "cursor18",
    "cursor19",
    "cursor20",
    "cursor21",
    "cursor22"
};
static char* _cursorinfo_values[]={
// Edit following line to specify cursor files path
    "/usr/local/share/minigui/res/cursor/",
    "23",
    "d_arrow.cur",
    "d_beam.cur",
    "d_pencil.cur",
    "d_cross.cur",
    "d_move.cur",
    "d_sizenwse.cur",
    "d_sizenesw.cur",
    "d_sizewe.cur",
    "d_sizens.cur",
    "d_uparrow.cur",
    "d_none.cur",
    "d_help.cur",
    "d_busy.cur",
    "d_wait.cur",
    "g_rarrow.cur",
    "g_col.cur",
    "g_row.cur",
    "g_drag.cur",
    "g_nodrop.cur",
    "h_point.cur",
    "h_select.cur",
    "ho_split.cur",
    "ve_split.cur"
};
// Section: resinfo
static char* _resinfo_keys[]={
    "respath"
};
static char* _resinfo_values[]={
    "/usr/local/share/minigui/res/"
};
// Section: classic
static char* _classic_keys[]={
// Note that max number defined in source code is 5.
    "iconnumber",
    "icon0",
    "icon1",
    "icon2",
    "icon3",
    "icon4",
// default icons for new OpenFileDialogBox
    "dir",
    "file",
// default icons for TreeView control
    "treefold",
    "treeunfold",
// bitmap used by BUTTON control
    "radiobutton",
    "checkbutton",
// background picture, use your favirate photo
    "bgpicture",
    "bgpicpos",
// bgpicpos=upleft
// bgpicpos=downleft
// bgpicpos=upright
// bgpicpos=downright
// bgpicpos=upcenter
// bgpicpos=downcenter
// bgpicpos=vcenterleft
// bgpicpos=vcenterright
// bgpicpos=none
//window element metrics
    "caption",
    "menu",
    "border",
    "scrollbar",
//window element colors
    "fgc_active_caption",
    "bgca_active_caption",
    "bgcb_active_caption",
    "fgc_menu",
    "bgc_menu",
    "fgc_msgbox",
    "fgc_tip",
    "bgc_tip",
    "fgc_active_border",
    "fgc_inactive_border",
    "fgc_inactive_caption",
    "bgca_inactive_caption",
    "bgcb_inactive_caption",
    "fgc_window",
    "bgc_window",
    "fgc_3dbox",
    "mainc_3dbox",
    "fgc_selected_item",
    "bgc_selected_item",
    "bgc_selected_lostfocus",
    "fgc_disabled_item",
    "bgc_disabled_item",
    "fgc_hilight_item",
    "bgc_hilight_item",
    "fgc_significant_item",
    "bgc_significant_item",
    "bgc_desktop"
};
static char* _classic_values[]={
// Note that max number defined in source code is 5.
    "5",
    "form.ico",
    "failed.ico",
    "help.ico",
    "warning.ico",
    "excalmatory.ico",
// default icons for new OpenFileDialogBox
    "folder.ico",
    "textfile.ico",
// default icons for TreeView control
    "fold.ico",
    "unfold.ico",
// bitmap used by BUTTON control
    "classic_radio_button.bmp",
    "classic_check_button.bmp",
// background picture, use your favirate photo
    "none",
    "center",
// bgpicpos=upleft
// bgpicpos=downleft
// bgpicpos=upright
// bgpicpos=downright
// bgpicpos=upcenter
// bgpicpos=downcenter
// bgpicpos=vcenterleft
// bgpicpos=vcenterright
// bgpicpos=none
//window element metrics
    "20",
    "25",
    "2",
    "16",
//window element colors
    "0xFFFFFFFF",
    "0xFF6A240A",
    "0xFF6A240A",
    "0xFF000000",
    "0xFFCED3D6",
    "0xFF000000",
    "0xFF000000",
    "0xFFE7FFFF",
    "0xFFCED3D6",
    "0xFFCED3D6",
    "0xFFC8D0D4",
    "0xFF808080",
    "0xFF808080",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFF000000",
    "0xFFCED3D6",
    "0xFFFFFFFF",
    "0xFF6B2408",
    "0xFFBDA69C",
    "0xFF848284",
    "0xFFCED3D6",
    "0xFFFFFFFF",
    "0xFF6B2408",
    "0xFFFFFFFF",
    "0xFF6B2408",
    "0xFFC08000"
};
#ifdef _MGLF_RDR_FLAT
// Section: flat
static char* _flat_keys[]={
// Note that max number defined in source code is 5.
    "iconnumber",
    "icon0",
    "icon1",
    "icon2",
    "icon3",
    "icon4",
// default icons for new OpenFileDialogBox
    "dir",
    "file",
// default icons for TreeView control
    "treefold",
    "treeunfold",
// bitmap used by BUTTON control
    "radiobutton",
    "checkbutton",
// background picture, use your favirate photo
    "bgpicture",
    "bgpicpos",
//window element metrics
    "caption",
    "menu",
    "border",
    "scrollbar",
//window element colors
    "fgc_active_caption",
    "bgca_active_caption",
    "bgcb_active_caption",
    "fgc_inactive_caption",
    "bgca_inactive_caption",
    "bgcb_inactive_caption",
    "fgc_menu",
    "bgc_menu",
    "fgc_msgbox",
    "fgc_tip",
    "bgc_tip",
    "fgc_active_border",
    "fgc_inactive_border",
    "fgc_window",
    "bgc_window",
    "fgc_3dbox",
    "mainc_3dbox",
    "fgc_selected_item",
    "bgc_selected_item",
    "bgc_selected_lostfocus",
    "fgc_disabled_item",
    "bgc_disabled_item",
    "fgc_hilight_item",
    "bgc_hilight_item",
    "fgc_significant_item",
    "bgc_significant_item",
    "bgc_desktop",
    "flat_tab_normal_color"
};
static char* _flat_values[]={
// Note that max number defined in source code is 5.
    "5",
    "form-flat.ico",
    "failed-flat.ico",
    "help-flat.ico",
    "warning-flat.ico",
    "excalmatory-flat.ico",
// default icons for new OpenFileDialogBox
    "folder-flat.ico",
    "textfile-flat.ico",
// default icons for TreeView control
    "fold-flat.ico",
    "unfold-flat.ico",
// bitmap used by BUTTON control
    "flat_radio_button.bmp",
    "flat_check_button.bmp",
// background picture, use your favirate photo
    "none",
    "center",
//window element metrics
    "20",
    "25",
    "1",
    "16",
//window element colors
    "0xFFFFFFFFF",
    "0xFF000000",
    "0xFF000000",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFFFFFFFF",
    "0xFF000000",
    "0xFFD8D8D8",
    "0xFF000000",
    "0xFF000000",
    "0xFFE7FFFF",
    "0xFF000000",
    "0xFF848284",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFFFFFFFF",
    "0xFF000000",
    "0xFFBDA69C",
    "0xFF848284",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFF664E4A",
    "0xFFFFFFFF",
    "0xFF000000",
    "0xFFC08000",
    "0xFFC6D2CF"
};
#endif
#ifdef _MGLF_RDR_SKIN
// Section: skin
static char* _skin_keys[]={
// Note that max number defined in source code is 5.
    "iconnumber",
    "icon0",
    "icon1",
    "icon2",
    "icon3",
    "icon4",
// default icons for new OpenFileDialogBox
    "dir",
    "file",
// default icons for TreeView control
    "treefold",
    "treeunfold",
// background picture, use your favirate photo
    "bgpicture",
    "bgpicpos",
//window element metrics
    "caption",
    "menu",
    "border",
    "scrollbar",
    "fgc_active_caption",
    "bgca_active_caption",
    "bgcb_active_caption",
    "fgc_menu",
    "bgc_menu",
    "fgc_msgbox",
    "fgc_tip",
    "bgc_tip",
    "fgc_active_border",
    "fgc_inactive_border",
    "fgc_inactive_caption",
    "bgca_inactive_caption",
    "bgcb_inactive_caption",
    "fgc_window",
    "bgc_window",
    "fgc_3dbox",
    "mainc_3dbox",
    "fgc_selected_item",
    "bgc_selected_item",
    "bgc_selected_lostfocus",
    "fgc_disabled_item",
    "bgc_disabled_item",
    "fgc_hilight_item",
    "bgc_hilight_item",
    "fgc_significant_item",
    "bgc_significant_item",
    "bgc_desktop",
    "skin_bkgnd",
    "skin_caption",
    "skin_caption_btn",
//for scrollbar
    "skin_scrollbar_hshaft",
    "skin_scrollbar_vshaft",
    "skin_scrollbar_hthumb",
    "skin_scrollbar_vthumb",
    "skin_scrollbar_arrows",
//for border
    "skin_tborder",
    "skin_bborder",
    "skin_lborder",
    "skin_rborder",
    "skin_arrows",
    "skin_arrows_shell",
    "skin_pushbtn",
    "skin_radiobtn",
    "skin_checkbtn",
//for treeview
    "skin_tree",
    "skin_header",
    "skin_tab",
//for trackbar
    "skin_tbslider_h",
    "skin_tbslider_v",
    "skin_trackbar_horz",
    "skin_trackbar_vert",
//for progressbar
    "skin_progressbar_htrack",
    "skin_progressbar_vtrack",
    "skin_progressbar_hchunk",
    "skin_progressbar_vchunk"
};
static char* _skin_values[]={
// Note that max number defined in source code is 5.
    "5",
    "form.ico",
    "failed.ico",
    "help.ico",
    "warning.ico",
    "excalmatory.ico",
// default icons for new OpenFileDialogBox
    "folder.ico",
    "textfile.ico",
// default icons for TreeView control
    "fold.ico",
    "unfold.ico",
// background picture, use your favirate photo
    "none",
    "center",
//window element metrics
    "25",
    "25",
    "1",
    "17",
    "0xFFFFFFFF",
    "0xFFE35400",
    "0xFF686868",
    "0xFF000000",
    "0xFFD4D6FF",
    "0xFF000000",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFFC8D0D4",
    "0xFFC8D0D4",
    "0xFFF8E4D8",
    "0xFFDF967A",
    "0xFF686868",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFF000000",
    "0xFFD8E9EC",
    "0xFFFFFFFF",
    "0xFFC56A31",
    "0xFFD8E9EC",
    "0xFF99A8AC",
    "0xFFFFFFFF",
    "0xFFFFFFFF",
    "0xFFC56A31",
    "0xFFFFFFFF",
    "0xFFC56A31",
    "0xFF984E00",
    "skin_bkgnd.bmp",
    "skin_caption.gif",
    "skin_cpn_btn.gif",
//for scrollbar
    "skin_sb_hshaft.bmp",
    "skin_sb_vshaft.bmp",
    "skin_sb_hthumb.bmp",
    "skin_sb_vthumb.bmp",
    "skin_sb_arrows.bmp",
//for border
    "skin_tborder.bmp",
    "skin_bborder.bmp",
    "skin_lborder.bmp",
    "skin_rborder.bmp",
    "skin_arrows.gif",
    "skin_arrows_shell.bmp",
    "skin_pushbtn.gif",
    "skin_radiobtn.gif",
    "skin_checkbtn.bmp",
//for treeview
    "skin_tree.bmp",
    "skin_header.bmp",
    "skin_tab.gif",
//for trackbar
    "skin_tbslider_h.gif",
    "skin_tbslider_v.gif",
    "skin_tb_horz.gif",
    "skin_tb_vert.gif",
//for progressbar
    "skin_pb_htrack.gif",
    "skin_pb_vtrack.gif",
    "skin_pb_htruck.bmp",
    "skin_pb_vtruck.bmp"
};
#endif
// Section: fashion
static char* _fashion_keys[]={
// Note that max number defined in source code is 5.
    "iconnumber",
    "icon0",
    "icon1",
    "icon2",
    "icon3",
    "icon4",
// default icons for new OpenFileDialogBox
    "dir",
    "file",
// default icons for TreeView control
    "treefold",
    "treeunfold",
// bitmap used by BUTTON control
    "radiobutton",
    "checkbutton",
// background picture, use your favirate photo
    "bgpicture",
    "bgpicpos",
//window element metrics
    "caption",
    "menu",
    "border",
    "scrollbar",
    "fgc_active_caption",
    "bgca_active_caption",
    "bgcb_active_caption",
    "fgc_menu",
    "bgc_menu",
    "fgc_msgbox",
    "fgc_tip",
    "bgc_tip",
    "fgc_active_border",
    "fgc_inactive_border",
    "fgc_inactive_caption",
    "bgca_inactive_caption",
    "bgcb_inactive_caption",
    "fgc_window",
    "bgc_window",
    "fgc_3dbox",
    "mainc_3dbox",
    "fgc_selected_item",
    "bgc_selected_item",
    "bgc_selected_lostfocus",
    "fgc_disabled_item",
    "bgc_disabled_item",
    "fgc_hilight_item",
    "bgc_hilight_item",
    "fgc_significant_item",
    "bgc_significant_item",
    "bgc_desktop"
};
static char* _fashion_values[]={
// Note that max number defined in source code is 5.
    "5",
    "form.ico",
    "failed.ico",
    "mg_help.ico",
    "warning.ico",
    "excalmatory.ico",
// default icons for new OpenFileDialogBox
    "folder.ico",
    "textfile.ico",
// default icons for TreeView control
    "fold.ico",
    "unfold.ico",
// bitmap used by BUTTON control
    "fashion_radio_btn.bmp",
    "fashion_check_btn.bmp",
// background picture, use your favirate photo
    "none",
    "center",
//window element metrics
    "25",
    "25",
    "1",
    "17",
    "0xFFFFFFFF",
    "0xFFE35400",
    "0xFFFF953D",
    "0xFF000000",
    "0xFFFFE4BF",
    "0xFF000000",
    "0xFF000000",
    "0xFFFFFFFF",
    "0xFFC8D0D4",
    "0xFFC8D0D4",
    "0xFFF8E4D8",
    "0xFFDF967A",
    "0xFFEBB99D",
    "0xFF000000",
    "0xFFEBB99D",
    "0xFF000000",
    "0xFFD8E9EC",
    "0xFFFFFFFF",
    "0xFFC56A31",
    "0xFFD8E9EC",
    "0xFF99A8AC",
    "0xFFFFFFFF",
    "0xFFFFFFFF",
    "0xFFC56A31",
    "0xFFFFFFFF",
    "0xFFC56A31",
    "0xFF984E00"
};
static ETCSECTION _etc_sections [] = {
    {0, 5, "system", _system_keys,_system_values },
    {0, 1, "fbcon", _fbcon_keys,_fbcon_values },
    {0, 2, "qvfb", _qvfb_keys,_qvfb_values },
#ifdef _MGGAL_PCXVFB
    {0, 3, "pc_xvfb", _pc_xvfb_keys,_pc_xvfb_values },
#endif
    {0, 1, "rtos_xvfb", _rtos_xvfb_keys,_rtos_xvfb_values },
#ifdef _MGGAL_SHADOW
    {0, 3, "shadow", _shadow_keys,_shadow_values },
#endif
#ifdef _MGGAL_MLSHADOW
    {0, 4, "mlshadow", _mlshadow_keys,_mlshadow_values },
#endif
    {0, 12, "systemfont", _systemfont_keys,_systemfont_values },
    {0, 1, "rawbitmapfonts", _rawbitmapfonts_keys,_rawbitmapfonts_values },
    {0, 1, "varbitmapfonts", _varbitmapfonts_keys,_varbitmapfonts_values },
    {0, 1, "upf", _upf_keys,_upf_values },
    {0, 1, "qpf", _qpf_keys,_qpf_values },
    {0, 1, "truetypefonts", _truetypefonts_keys,_truetypefonts_values },
    {0, 1, "mouse", _mouse_keys,_mouse_values },
    {0, 2, "event", _event_keys,_event_values },
    {0, 25, "cursorinfo", _cursorinfo_keys,_cursorinfo_values },
    {0, 1, "resinfo", _resinfo_keys,_resinfo_values },
    {0, 45, "classic", _classic_keys,_classic_values },
#ifdef _MGLF_RDR_FLAT
    {0, 46, "flat", _flat_keys,_flat_values },
#endif
#ifdef _MGLF_RDR_SKIN
    {0, 71, "skin", _skin_keys,_skin_values },
#endif
    {0, 45, "fashion", _fashion_keys,_fashion_values }
};

////////////////////////////////////////////////////

static ETC_S _ETC = {
    0,
    sizeof(_etc_sections)/sizeof(ETCSECTION),
    _etc_sections
};

GHANDLE __mg_get_mgetc (void)
{
    return (GHANDLE) &_ETC;
}

#endif /* _MGINCORE_RES */
```

3.3 Sample of Configuration

Under most circumstances, we modify runtime configuration file, we will be limited to several sections. The system section and font related several sections are primary sections. In this chapter, we will give two configuration examples.

3.3.1 Runtime Configuration when only Support for ISO8859-1 Charset

1) Configuration File

```
# The first system font must be a logical font using RBF device font.
[systemfont]
font_number=1
font0=rbf-fixed-rrncnn-8-16-ISO8859-1

default=0
wchar_def=0
fixed=0
caption=0
menu=0
control=0

[rawbitmapfonts]
font_number=1
name0=rbf-fixed-rrncnn-8-16-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/8x16-iso8859-1.bin

[varbitmapfonts]
font_number=0

[qpf]
font_number=0

[truetypefonts]
font_number=0

[type1fonts]
font_number=0
```

2) Incore Configuration Options

```
static char *SYSTEMFONT_KEYS[] =
{"font_number", "font0", "default", "wchar_def", "fixed", "caption", "menu", "control"};

static char *SYSTEMFONT_VALUES[] =
{
    "1","rbf-fixed-rrncnn-8-16-ISO8859-1", "0", "0", "0", "0", "0", "0"
};
```

3.3.2 Specifying Different Graphic Engine and Input Engine

1) Configuration File

```
[system]
# GAL engine and default options
gal_engine=commlcd

# IAL engine
ial_engine=auto

mdev=/dev/ts
mtype=IMPS2
```

2) Incore Configuration Option

```
static char *SYSTEM_KEYS[] = {"gal_engine", "ial_engine", "mdev", "mtype"};

static char *SYSTEM_VALUES[] = {"commlcd", "auto", "/dev/ts", "IMPS2"};
```

4 Developing MiniGUI Application in Windows

Feynman provides two methods for developer, which is accustomed to develop application in Window platform.

- Using the package of MiniGUI for Win32. It is pre-compiled standard development package in Win32. It contains wvfb, MiniGUI function library (libminigui and libmgext) and header files.
- Using MiniGUI SDK for Win32. This is an optional component in MiniGUI. It contains the whole source codes and provides users the convenience for customizing the package of MiniGUI for Win32.

By using the package of MiniGUI for Win32 or the component product of MiniGUI SDK for Win32, developer can compile and debug MiniGUI application in Windows.

This chapter describes how to use the package of MiniGUI for Win32. User can contact Feynman to purchase the component product of MiniGUI SDK for Win32.

To develop MiniGUI application in Windows, you must install MS Visual Studio 98. First, you decompress arbitrary directory in windows. Secondly you open the helloworld project file in VC according to README. Figure 4.1 shows it.

After compiling successfully, you should run wvfb first and run helloworld. Note that you need copy **`helloworld.ext`** to directory dll. Fig 4.2 shows running result.

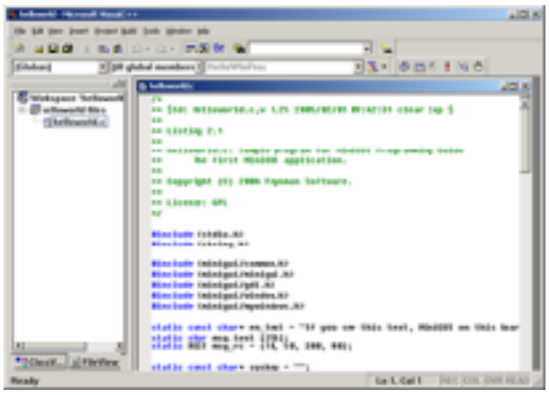
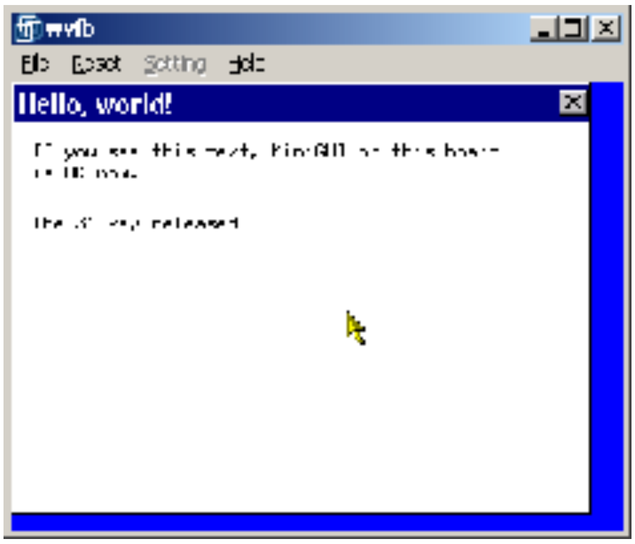


Fig 4.1 open MiniGUI helloworld project



Hello, world!

If you see this text, MiniGUI on this board is OK now.

The [illegible] key released

Fig 4.2 Compiling and Running MiniGUI Application in Windows

Refer to above helloworld, you can create, develop and compile new MiniGUI application in VC. But give your attention to the following:

- Because the package of MiniGUI for Win32 is pre-compiled library, the function, compiling configuration options, and running configuration options are fixed, and only support MiniGUI-Threads runtime mode.
- Using the package of MiniGUI for Win32 to develop applications, please don’t call Windows special API, which isn’t supported possibility by target OS.

Appendix A Frequent Ask Questions (FAQs)

A.1 Questions Relevant to GPL Versions

Q1. Do I need to pay Feynman Software for the license fee if I use GPL versions of MiniGUI?

A1. The GPL versions of MiniGUI are available at Feynman Software website; you can use them freely if you use MiniGUI under GPL license. However, the release of your applications that are based on MiniGUI GPL versions should also complies with GPL. If you use MiniGUI to develop commercial purpose applications, i.e., you do not want to release them under GPL terms, you then should pay Feynman Software for the licensing fee.

Q2. When you use MiniGUI GPL versions, what kind of behaviors would violate Feynman Software's legal rights?

A2. Feynman Software owns the copyright of several free software projects. We release that software under GPL with the purpose of helping users to understand software inner mechanism well and customize them freely and easily. However, most users are not familiar with GPL terms; they would sometimes act against GPL terms unconsciously. The behaviors below would violate Feynman Software's legal rights:

- Pirate part or whole source code to use in other occasions; the worse thing is to pirate MiniGUI and sell it as private software. Such behavior has already seriously offended against the copyright laws.
- Modify source code of free software, and use them in commercial purpose, but they are not released according to GPL terms.

Under GPL terms, applications based-on MiniGUI should be released under GPL. If you do not release MiniGUI applications under GPL, neither buying MiniGUI commercial licenses, this behavior belongs to software pirate.

A.2 Questions Relevant to MiniGUI Application Fields

Q3. What kinds of products that use MiniGUI are successfully launched in market?

A3. MiniGUI is widely used in the products like mobile phones, IPTVs, digital TVs, industry control systems, information terminals, industrial meters, and so on. For the detailed introduction for some typical products, you can visit:

```
http://www.minigui.com/en/introduction/application/
```

Q4. How is about the stability of MiniGUI?

A5. It is hard to answer this question as the factor that influences system stability is sometimes due to applications instead of the libraries. However, we can offer you some information as reference:

- For a complicated MiniGUI application, a test shows that there is no problem for the shift in between multi-windows by pressing key 100,000 times in two days.
- Many industrial control systems that are developed based on MiniGUI can now stably run under real industrial situations.

A.3 Questions Relevant to Portability

Q5. What operating systems does MiniGUI support?

A5. By now, MiniGUI provides the support for many popular embedded operating systems including Linux/uClinux, VxWorks, ThreadX, Nucleus, pSOS, OSE, eCos, and even uC/OS-II. MiniGUI can also run on Win32 platform.

Q6. Which CPUs have MiniGUI run on successfully so far? Moreover, what is the lowest frequency of CPU MiniGUI needed?

A6. There are successful cases for MiniGUI running in ARM-based CPUs (such as StongARM, xScale, S3C2410, S3C2440, EM8511, EM8620), PowerPC, MIPS, M68k, FRV. In those CPUs, the one with lowest main frequency is about 20 MHz (20 MIPS).

Q7. Would MiniGUI provide support for monochrome LCD?

A7. Yes. Actually, MiniGUI can provide support for almost all LCD controllers in various modes, such as monochrome, gray, 256-color, 4096-color, and 65536-color.

Q8. Which resolution of screen can MiniGUI run properly?

A8. In theory, the running of MiniGUI is not influenced by the resolution of screen.

A.4 Questions Relevant to Compilation

Q9. Why are there so many compilation errors when I enable the option to support TrueType font?

A9. The main reason is that the libttf version supporting TrueType font in your system is too high. MiniGUI uses libttf 1.3.1. In several Linux distributions such as RedHat Linux 7, the library libttf 2.0 is installed by the default. In this case, you can install libttf 1.3.1 or use **`--with-ttfsupport=no`** option to disable the support for TrueType font of MiniGUI.

Q10. During compiling the library, why does the mistake below occur sometimes?

```
can not make hard link filename.o to filename.lo.
```

A10. Symbol links and hard links are the specialized file types in UNIX file system. If you compile library being maintained by Automake/Autoconf script, you cannot create these links on a non-UNIX file system. Please check your file system to make sure if it is not FAT32 file system.

Q11. When I use the Open File Dialog Box, why does the mistake below occur?

```
undefined reference to ShowOpenDialog
```

A11. The function ShowOpenDialog is included in the mGUtils component. If you want to use this function, you should include two header files: <**mgutils/mgutils.h**>. When make the executable, please make sure to link mGUtils(**-lmgutils**). In addition, if you run MiniGUI on some embedded operating systems, which are lack of the support for file system, you can't use the Open File Dialog Box.

Q12. My system does not support 64-bit integer. Is the data type of Uint64 in MiniGUI essential?

A12. The data type of Uint64 in MiniGUI is used to generate the complex graphics. If your system does not support 64-bit integer, you can use the following configuration option to disable the usage of 64-bit integer:

```
--disable-fixedmath
```

A.5 Questions Relevant to Input Engines

Q13. On Linux PC boxes, what kinds of mouse types does MiniGUI support?

A13. Currently, the mouse protocols supported by MiniGUI are MS, MS3, PS2, and Intelligent PS2 (IMPS2).

Q14. On Linux PC boxes, I would like to use the old serials mouse. What should I do?

A14. MiniGUI can provide support for almost all mouse types via GPM. Please configure it as follows:

1) Run **gpm -k** to kill gpm that is running.
2) Run **mouse-test** to confirm your mouse device and protocol.
3) Run **gpm** to set mouse device and protocol as follows.

```
gpm -R -t <yourmousetype> -m <yourmousedevice>
```

4) Edit **MiniGUI.cfg** file, set **mtype** as **gpm**; and set mdev as **/dev/gpmdata**:

```
[system]
...
mtype=gpm
mdev=/dev/gpmdata
```

Then, start up MiniGUI. Please note you can use the option **-R** when you set the mouse protocol by **gpm**. **-R** option is used to transfer original mouse protocol to GPM defined mouse protocol, and make it shown in **/dev/gpmdata** file.

A.6 Runtime Questions

Q15. On Linux, How would I capture the screen of MiniGUI?

A15. When running MiniGUI program, you can capture the screen as a BMP file in the current directory by pressing **<PrtSc>** key. The file name is **0-<NO>.bmp**, therein **<NO>** is the number of times of pressing **<PrtSc>** key. You can save the BMP file of the current active main window as **<HWND>-<NO>.bmp**, therein **<HWND>** is the handle of the active main window while **<NO>** is the number of times of pressing **<Ctrl+PrtSc>** key.

Q16. Why does the program exit after displaying two dialog boxes when I run **mginit** in Mg-samples?

A16. The main reason is that MiniGUI being installed does not provide support for PNG image files. In some Linux distributions (such as early TurboLinux), as the version of their PNG graphics support library (libpng) is too old, it would automatically disable the support for PNG image when you configure MiniGUI. In this case, **LoadBitmapFromFile** function of MiniGUI cannot correctly load PNG image files, while Mg-samples **mginit** needs to load two PNG files for running. That is why **mginit** exits.

To solve this problem, there are two ways. First, you can download and install the latest libpng library from INTERNET. Secondly, modify **nr** value in section **[mginit]** in **mginit.rc**, and make the value less than 8.

Another reason that may cause such error is that you do not start up **mginit** in its directory. Please change to the directory, then run **mginit**.

Q17. Under MiniGUI-Processes runtime mode, how would I switch from MiniGUI to other console?

A17. Under MiniGUI-Processes runtime mode, if you are using the **console** input engine, you can switch from MiniGUI to other virtual console by pressing **<Right_Ctrl+Fx>** key, also, you can quit MiniGUI by pressing **<Ctrl+Alt+Backspace>**. Currently, MiniGUI-Threads does not provide such functions.

A.7 Common Error Messages

Q18. Why is the following message shown when I run programs in mg-samples on Linux?

```
AttachSharedResource: No such file or directory
Error in step 6: Can not attach shared resource!
Initialize minigui failure when using /etc/MiniGUI.cfg as cfg file.
```

A18. If you configure MiniGUI as MiniGUI-Processes or MiniGUI-Standalone, you should run **mginit** program first. As MiniGUI-Processes or MiniGUI-Standalone adopts a C/S architecture, you have to start up the sever program, **mginit**, before running client programs. In Mg-samples package, you should run **mginit** in **mginit/** directory first, then run demo programs in other directories.

Q19. Why do I see the information below when I run MiniGUI?

```
    GAL ENGINE: Error when opening /dev/fb0: Permission denied. Please check your kernel config.
    GAL: Init GAL engine failure.
    Error in step 3: Can not initialize graphics engine!
    Initialize minigui failure when using /usr/local/etc/MiniGUI.cfg as cfg file
```

A19. The main reason is that you have not activated the FrameBuffer driver yet, or the permission of **/dev/fb0** is incorrect.

Q20. Under MiniGUI-Processes runtime mode, why does it give error information below when I run mginit in mg-samples?

```
    Error in step 2 : There is already an instance of minigui.
    Initialize minigui failure when using /usr/local/etc/MiniGUI.cfg as config file.
```

A20. Usually, there are two possible reasons. One is that you have already run an **mginit** program; other is that you did not exit MiniGUI properly when you run **mginit** last time. If it is the second reason, you can delete **minigui** file and **mginit** file in **/var/tmp/** directory. If it still does not work, please restart your computer.

Q21. Why do the following statement show when I run MiniGUI?

```
NEWGAL: Does not find matched engine: fbcon.
Error in step 3: Can not get graphics engine information!
```

A21. The possible problem is that **FBCON** engine in NEWGAL interface fails when initializing FrameBuffer device. The main reasons are that your kernel does not support FrameBuffer driver, or does not activate FrameBuffer driver, or you have no proper access permission to open **/dev/fb0** device.

Q22. On Linux, what is the meaning of the error information below?

```
vesafb does not support changing the video mode
```

A22. It is a warning that can be ignored. It aims at VESA FrameBuffer driver. VESA FrameBuffer driver does not support the display mode switch during running. It can only set video mode by the boot option for kernel. Moreover, once set, it cannot be changed unless you modify the boot option and restart your system.

Q23. On Linux, what is the meaning of the error information below?

```
NEWGAL: No video mode large enough for the resolution specified.
NewGAL: Set video mode failure.
```

A23. The main reason is that the display resolution being set in **MiniGUI.cfg** is higher than that supported by your FrameBuffer driver. Therefore, you can try to set a smaller resolution by modify **MiniGUI.cfg** file.