LearningGUI

用户参考手册

Graphic Library with Graphical User Interface

软件版本: Ver0.2
修订日期: 2017-10-12

免责声明

本软件系统所提供的软件源码、相关工具软件和相关资料等均为本软件系统开发者饶佑坤提供及发布。用户可以自由选择是否使用本软件系统。本软件系统开发者不提供任何形式的担保（不论是明确的或隐含的）、不承担因用户使用该软件系统和文档所带来的任何法律责任。

版权声明

LearningGUI 是一款嵌入式 GUI 中间件，其版权属于自然人饶佑坤所有。LearningGUI 使用 C 语言开发，采用源码发布，以两种协议方式发布：一是使用 GPLv3 开源协议发布，免费使用，在该协议下，LearningGUI 应用程序也需要遵循 GPLv3 协议开源发布；二是使用商业协议发布，用户必须获取版权所有者的授权，并且需要向版权所有者付费。

GPLv3 开源协议声明如下：

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast,the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying,distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works

for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work,and which are not combined with it such as to form a larger program,in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work

from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

{one line to give the program's name and a brief idea of what it does.}
Copyright (C) {year} {name of author}
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>. Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

{project} Copyright (C) {year} {fullname}
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary.For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

联系方式

电子邮件：　960747373@qq.com
QQ:　960747373
QQ 群：　275782855
电话：　15976988952(微信)

版本发布历史记录

LearningGUI 版本发布历史记录如表 0-1 所示。

版本	发布日期	描述
0.2.0	2017-10-12	初始发布版本

表 0-1　版本发布历史记录

关于本文档的假设和约定

对 LearningGUI 应用开发者的假定

本文档假定 LearningGUI 应用开发者具备以下基本技能:

一、C 语言软件开发环境的使用。

二、C 语言编程、调试基本能力。

三、GUI 基本知识。

四、嵌入式系统基本知识。

本文档缩写约定

本文档缩写约定如表 0-2 所示。

缩写	描述
GUI	Graphical User Interface
MTJT	Mouse-Touch-Joystick-Tablet（鼠标类）
KB	Kilo-byte（千字节）
2D	二维平面
系统	如特殊说明，特指 LearningGUI
应用	如特殊说明，特指 LearningGUI 应用程序
IDE	软件集成开发环境

表 0-2　本文档缩写约定表

目　录

第一部分　系统基础知识

第一章　系统概述

LearningGUI(儿科 GUI)是一种高度可裁剪、占用资源小、简易的通用嵌入式 GUI。

系统 100%使用 C 语言开发，编码遵行 ISO/C89 和 ISO C99 标准；系统显示输出采用像素点输出方式；系统提供开放式驱动接口；开源系统。因此，系统的通用性、跨平台性、移植性非常好。LearningGUI 应用既能在裸机中运行，也能在操作系统中运行(Linux、RTOS)。

系统支持多任务（多线程）处理。

系统汉字支持能力强。能支持 GB2312-80 标准多字节字库，同时，也能支持 Unicode 编码字符集。

系统分为 Basic 版本和 Windows 版本。　Basic 版本是基础版本，包括 2D 绘图功能（画点、画线、画矩形、填充矩形等等）、文本显示功能（字符串显示）、图像显示功能、消息驱动机制等等。Basic 版本使用静态内存分配策略。Windows 版本则是建立在 Basic 版本之上的 window 窗口管理系统。Widget 小部件是 Windows 版本的基本管理单元。常见小部件包括 Frame 框架部件、GroupBox 分组部件、Cell 单元部件、Label 标签部件、PushButton 普通按钮部件、RadioButton 多选一按钮部件、CheckBox 二选一部件、LineEdit 单行编辑部件、ListBox 列表部件、ComBox 组合框部件、Image 图像部件、ProgressBar 进度条部件、SliderBar 滑竿部件等等。Windows 版本采用动态内存分配策略。Windows 版本属于可裁剪项。

系统没有使用浮点数运算。

LearningGUI 设计占用资源规格（不包括字库）如表 1-1 所示。对于 Windows 版本，需要的 ROM 与所支持的部件类型多少有关，需要的 RAM 与应用所创建的具体部件数量相关。

系统版本	ROM(KB)(动态库)	RAM(KB)
Basic	约 128	约 16
Windows	约 256	约 64

表 1-1　LearningGUI 设计规格表

第二章　系统数据类型

系统定义了与 CPU 字长相关、符号相关、硬件平台相关等等数据类型。在某些情况下，用户需要根据自己的硬件平台修改相关的定义。系统基本数据类型定义如表 2-1 所示。

系统数据类型	C 语言数据类型
INT8	char
INT16	short int
INT32	int
INT64	long long
UINT8	unsigned char
UINT16	unsigned short int
UINT32	unsigned int
UINT64	unsigned long long
UINT	unsigned int
BYTE	unsigned char
WORD	unsigned short int
DWORD	unsigned int
UCHAR	unsigned char
BOOL	Unsigned char
BINT	int
BUINT	unsigned int

表 2-1　LearningGUI 数据类型定义表

其中 BINT 和 BUNIT 涉及到优化，一般定义成 SRAM(SDRAM)数据 BUS 宽度。

第三章　坐标系及坐标变换

3.1　系统所使用的坐标系

系统使用两种坐标系统：绝对坐标系（物理坐标系）和相对坐标系（逻辑坐标系）。物理坐标系的原点位于显示屏幕的左上角，X 轴方向由左向右水平增加，Y 轴方向由上向下垂直增加。逻辑坐标系原点可能是显示屏幕的坐上角，也有可能是父窗口的左上角，X 轴方向由左向右水平增加，Y 轴方向由上向下垂直增加。坐标系单位均为像素。

对于系统接口 API 中的参数，使用的是逻辑坐标系。在 Windows 版本中，窗口和部件坐标是相对于父窗口的逻辑坐标。在系统内部，逻辑坐标最终转化成物理坐标。

但是在用户显示驱动接口中，使用的是物理坐标系。

3.2　坐标变换

系统坐标变换是指逻辑坐标和物理坐标的相互变换和平移变换。系统坐标变换都是在内部自动完成，用户无需干预坐标的变换。同时，系统并不给用户提供坐标系和坐标变换的接口。

第四章　内存管理

Basic 版本设计目标是针对小型嵌入系统（存储资源紧张）的情形。因此，Basic 版本在内存管理上是采用静态内存分配策略，比如系统内置 DC 静态分配、邮寄消息队列的静态分配等等。

Windows 版本设计目标是针对中型嵌入系统（存储资源相对丰富）的情形。因此，Windows 版本在内存管理上主要是采用动态内存分配策略，比如动态建立主窗口、动态建立各种部件等等。动态内存管理由 IDE 完成。

第五章　颜色管理

5.1　颜色管理概述

系统使用 32 位 RGB 颜色空间进行颜色管理和颜色输出。

5.2　物理颜色

物理颜色是显示屏幕能够显示的颜色，它的位数和 RGB 顺序与具体显示硬件相关。系统定义 SCREEN_COLOR 数据类型为物理颜色类型。SCREEN_COLOR 定义如下：

```
#define    SCREEN_COLOR          UINT32
```

5.3　逻辑颜色

逻辑颜色是系统所管理的颜色，独立于物理颜色。应用面向逻辑颜色编程。系统定义 GUI_COLOR 数据类型为逻辑颜色类型。GUI_COLOR 定义如下：

```
#define    GUI_COLOR              UINT32
```

RGB 顺序定义：0x00RRGGBB。BB 表示蓝色分量，GG 表示绿色分量，RR 表示红色分量，即最低位字节表示蓝色分量，第二低字节表示绿色分量，次高字节表示红色分量。

系统预定义了常见的逻辑颜色，如表 5-1 所示。

系统颜色	颜色数值
GUI_RED	0x00FF0000
GUI_GREEN	0x0000FF00
GUI_BLUE	0x000000FF
GUI_BLACK	0x00000000
GUI_LIGHT_BLACK	0x00101010
GUI_HEAVY_DARK	0x00202020
GUI_DARK	0x00404040
GUI_LIGHT_DARK	0x00606060
GUI_GRAY	0x00808080
GUI_LIGHT_GRAY	0x00C0C0C0
GUI_YELLOW	0x00FFFF00
GUI_BROWN	0x00A52A2A
GUI_MAGENTA	0x008B008B
GUI_CYAN	0x0000FFFF
GUI_LIGHT_GREEN	0x0080FF80
GUI_LIGHT_RED	0x00FF8080
GUI_LIGHT_CYAN	0x00FFFF80
GUI_LIGHT_MAGENTA	0x00FF80FF
GUI_LIGHT_BLUE	0x008080FF
GUI_LIGHT_WHITE	0x00E0E0E0
GUI_WHITE	0x00FFFFFF

表 5-1　LearningGUI 预定义颜色表

同时，用户可以使用 GUI_ARGB 宏自定义逻辑颜色。GUI_ARGB 宏参数如下：

```
GUI_ARGB(alpha,red,green,blue)
```

其中，alpha 表示 alpha 通道数值（预留），red 表示红色分量，green 表示绿色分量，blue 表示蓝色分量。red、green、blue 取值范围为 0~255（0~0xFF）。

5.4 颜色组

系统对颜色进行分组管理。系统颜色组定义如表 5-1 所示。

颜色组名称	颜色组标示	颜色组说明
无效组（禁止组）	DISABLED_GROUP	不能获得焦点。不处理键盘、鼠标事件。
非活动组	INACTIVE_GROUP	非焦点组。
活动组	ACTIVE_GROUP	焦点组。键盘当前输入焦点。

表 5-1 系统颜色组定义表

颜色组对 Windows 版本有意义，而对 Basic 版本来说，三组合一，没有区分意义。

5.5 颜色角色

系统对颜色进行分角色管理。系统颜色角色定义如表 5-2 所示。

颜色角色名称	颜色角色标示	颜色角色说明
背景颜色	BACK_ROLE	点、线、面的背景颜色
前景颜色	FORE_ROLE	点、线、面前景颜色
文本背景颜色	TEXT_BACK_ROLE	文本背景颜色
文本前景颜色	TEXT_FORE_ROLE	文本显示颜色

表 5-2 系统颜色角色定义表

5.6 调色板

此功能不常见。在某些情况下，颜色并不是直接表示，而是通过表示在预先定义的颜色索引表中索引号来表示颜色的，这张预先定义的颜色索引表就是通常所说的调色板。系统定义的调色板结构 GUI_PALETTE 如下：

```
struct  _GUI_PALETTE
{
    UINT            num;
    GUI_COLOR       *entries;
};
typedef  struct  _GUI_PALETTE    GUI_PALETTE;
```

其中 num 成员变量表示调色板大小，*entries 成员变量表示调色板颜色入口。

系统提供了 1 位、2 位、4 位、8 位全局调色板，全局变量分别定义如下：

```
GUI_PALETTE    l1pal;
GUI_PALETTE    l2pal;
GUI_PALETTE    l4pal;
GUI_PALETTE    l8pal;
```

1 位全局调色板颜色定义如下：

```
GUI_COLOR    l1pal_color[] =
{
   0x00000000,0x00FFFFFF
};
```

2 位全局调色板颜色定义如下：

```
GUI_COLOR    l2pal_color[] =
{
   0x000000FF,0x00FF00FF,0x0000FFFF,0x00FFFFFF
};
```

4 位全局调色板颜色定义如下：

```
GUI_COLOR    l4pal_color[] =
{
   0x00000000,0x00800000,0x00008000,0x00808000,
   0x00000080,0x00800080,0x00008080,0x00808080,
   0x00C0C0C0,0x00FF0000,0x0000FF00,0x00FFFF00,
   0x000000FF,0x00FF00FF,0x0000FFFF,0x00FFFFFF
};
```

8 位全局调色板颜色定义如下：

```
GUI_COLOR    l8pal_color[] =
{
   0x00000000,0x00800000,0x00008000,0x00808000,
   0x00000080,0x00800080,0x00008080,0x00C0C0C0,
   0x00C0DCC0,0x00A6CAF0,0x00402000,0x00602000,
   0x00802000,0x00A02000,0x00C02000,0x00E02000,
   0x00004000,0x00204000,0x00404000,0x00604000,
   0x00804000,0x00A04000,0x00C04000,0x00E04000,
   0x00006000,0x00206000,0x00406000,0x00606000,
   0x00806000,0x00A06000,0x00C06000,0x00E06000,
   0x00008000,0x00208000,0x00408000,0x00608000,
   0x00808000,0x00A08000,0x00C08000,0x00E08000,
   0x0000A000,0x0020A000,0x0040A000,0x0060A000,
   0x0080A000,0x00A0A000,0x00C0A000,0x00E0A000,
   0x0000C000,0x0020C000,0x0040C000,0x0060C000,
   0x0080C000,0x00A0C000,0x00C0C000,0x00E0C000,
   0x0000E000,0x0020E000,0x0040E000,0x0060E000,
   0x0080E000,0x00A0E000,0x00C0E000,0x00E0E000,
   0x00000040,0x00200040,0x00400040,0x00600040,
   0x00800040,0x00A00040,0x00C00040,0x00E00040,
   0x00002040,0x00202040,0x00402040,0x00602040,
   0x00802040,0x00A02040,0x00C02040,0x00E02040,
   0x00004040,0x00204040,0x00404040,0x00604040,
   0x00804040,0x00A04040,0x00C04040,0x00E04040,
   0x00006040,0x00206040,0x00406040,0x00606040,
   0x00806040,0x00A06040,0x00C06040,0x00E06040,
   0x00008040,0x00208040,0x00408040,0x00608040,
   0x00808040,0x00A08040,0x00C08040,0x00E08040,
   0x0000A040,0x0020A040,0x0040A040,0x0060A040,
   0x0080A040,0x00A0A040,0x00C0A040,0x00E0A040,
   0x0000C040,0x0020C040,0x0040C040,0x0060C040,
   0x0080C040,0x00A0C040,0x00C0C040,0x00E0C040,
   0x0000E040,0x0020E040,0x0040E040,0x0060E040,
   0x0080E040,0x00A0E040,0x00C0E040,0x00E0E040,
   0x00000080,0x00200080,0x00400080,0x00600080,
   0x00800080,0x00A00080,0x00C00080,0x00E00080,
   0x00002080,0x00202080,0x00402080,0x00602080,
   0x00802080,0x00A02080,0x00C02080,0x00E02080,
   0x00004080,0x00204080,0x00404080,0x00604080,
   0x00804080,0x00A04080,0x00C04080,0x00E04080,
   0x00006080,0x00206080,0x00406080,0x00606080,
   0x00806080,0x00A06080,0x00C06080,0x00E06080,
   0x00008080,0x00208080,0x00408080,0x00608080,
   0x00808080,0x00A08080,0x00C08080,0x00E08080,
   0x0000A080,0x0020A080,0x0040A080,0x0060A080,
   0x0080A080,0x00A0A080,0x00C0A080,0x00E0A080,
   0x0000C080,0x0020C080,0x0040C080,0x0060C080,
   0x0080C080,0x00A0C080,0x00C0C080,0x00E0C080,
   0x0000E080,0x0020E080,0x0040E080,0x0060E080,
   0x0080E080,0x00A0E080,0x00C0E080,0x00E0E080,
   0x000000C0,0x002000C0,0x004000C0,0x006000C0,
   0x008000C0,0x00A000C0,0x00C000C0,0x00E000C0,
   0x000020C0,0x002020C0,0x004020C0,0x006020C0,
   0x008020C0,0x00A020C0,0x00C020C0,0x00E020C0,
   0x000040C0,0x002040C0,0x004040C0,0x006040C0,
   0x008040C0,0x00A040C0,0x00C040C0,0x00E040C0,
   0x000060C0,0x002060C0,0x004060C0,0x006060C0,
```

```
    0x008060C0,0x00A060C0,0x00C060C0,0x00E060C0,
    0x000080C0,0x002080C0,0x004080C0,0x006080C0,
    0x008080C0,0x00A080C0,0x00C080C0,0x00E080C0,
    0x0000A0C0,0x0020A0C0,0x0040A0C0,0x0060A0C0,
    0x0080A0C0,0x00A0A0C0,0x00C0A0C0,0x00E0A0C0,
    0x0000C0C0,0x0020C0C0,0x0040C0C0,0x0060C0C0,
    0x0080C0C0,0x00A0C0C0,0x00FFFBF0,0x00A0A0A4,
    0x00808080,0x00FF0000,0x0000FF00,0x00FFFF00,
    0x000000FF,0x00FF00FF,0x0000FFFF,0x00FFFFFF
};
```

系统默认调色板是 8 位调色板。
同时系统提供了管理调色板接口。

5.6.1 设置系统默认调色板：palette_set

函数原型：int palette_set(GUI_PALETTE *palette)
返回值：调用成功返回 1，否则返回-1。

5.6.2 获取系统默认调色板：palette_get

函数原型：int palette_get(GUI_PALETTE *palette)
返回值：调用成功返回 1，否则返回-1。

调色板功能属于可裁剪项。

5.7 颜色转换

逻辑颜色和物理颜色可以相互转换，系统提供较常见的颜色转换函数。但是，在更多情况下，用户需要根据特定显示硬件来编写颜色转换函数。

GUI_COLOR 转换成 R5G6B5 格式函数定义如下：

```
SCREEN_COLOR  GUI_TO_R5G6B5(GUI_COLOR gui_color)
{
    SCREEN_COLOR  r, g, b;

    b = gui_color&0xF8;
    g = gui_color&0xFC00;
    r = gui_color&0xF80000;
    return  (b>>3)|(g>>5)|(r>>8);
}
```

R5G6B5 格式转换成 GUI_COLOR 函数定义如下：

```
GUI_COLOR  R5G6B5_TO_GUI(SCREEN_COLOR screen_color)
{
    GUI_COLOR  r, g, b;

    b = screen_color&0x1F;
    g = screen_color&0x07E0;
    r = screen_color&0xF800;
    return  (b<<3)|(g<<5)|(r<<8);
}
```

第六章　系统应用开发初步

6.1　系统源码包获取

系统以源码压缩包形式发布在互联网上。通过 QQ、WeChat、电子邮件、网站下载等方式获取源码。对于商业版本系统，则需要授权获取。

6.2　系统源码包结构

解压系统源码压缩包，形成系统发布根目录，系统根目录下有使用协议文本 License.txt 文件，同时主要包括以下目录：

1) CODE/SOURCE 目录
   系统源程序目录。
2) CODE/INCLUDE 目录
   应用开发所需的头文件目录。其中 lgui.h 包含了应用开发所必需的头文件，是总头文件。
3) PCTOOLS 目录
   存放 PC Windows 图片转换工具软件 LguiTool.exe 目录。
4) EXAMPLES 目录
   存放演示应用目录。
5) MANUAL 目录
   存放用户开发使用参考手册目录。

6.3　建立系统开发调试模拟环境

系统在 PC Linux(Fedora release 13)主机中开发完成，演示应用都在开发主机环境中编译运行通过，同时也在 ARM-Linux 环境中编译运行通过（需要 root 权限；默认的显示设备 FrameBuffer 是/dev/fb0，默认的鼠标设备名称是/dev/input/event3，默认的键盘设备名称是/dev/input/event4；用户需要根据实际情况修改相应的设备名称）。同理，系统也能在 IAR、MDK 等环境中开发运行。

在 CODE/INCLUDE 目录中，PLATFORM 子目录下的文件表示与用户平台、LearningGUI 系统配置相关。用户需要根据自己平台特征，或者出于功能裁减、性能优化等目的，可能需要修改该子目录下的文件。Basic_config.h 文件是 Basic 版本配置宏文件；window_config.h 文件是 Windows 版本配置宏文件；type.h 文件是与平台相关的数据类型定义文件。

在实际 GUI 应用开发中，建议用户将系统源代码编译成库模块形式，之后，完成下列步骤即可开发 LearningGUI 应用：

1) 在应用项目工程中指定 LearningGUI 库模块名称和路径；
2) 在应用项目工程中指定 LearningGUI 头文件路径；
3) 在用户应用源程序中包含 lgui.h 头，形式如下：

```
#include  "lgui.h"
```

6.4　系统函数调用说明

系统支持多任务（多线程）处理。为适应多任务（多线程）机制处理机制，系统函数划分为内部函数和外部函数。内部函数是非可重入函数，函数名称使用 in_开头标示；外部函数是可重入函数。LearningGUI 应用编程是面对外部函数编程，而驱动开发中则是面对内部函数编程。对于颜色转换函数、字符编码转换函数等等非 GUI 函数（纯算法函数），没有必要区分内外函数，使用大写函数名称或者使用 in_开头标示函数名称。

第七章　系统驱动接口开发

7.1　系统驱动接口概述

系统驱动程序是指显示驱动程序、键盘驱动程序、MTJT(鼠标)驱动程序、多任务（多线程）同步信号驱动程序。系统提供开放式注册驱动接口，系统驱动需要用户编写。在系统驱动接口开发中，不建议调用 LearningGUI 函数。如果需要的话，只能调用内部函数。

7.2　注册系统驱动接口函数

系统预定义五个驱动类型宏，如表 7-1 所示。

系统驱动类型	描述
DRIVER_SCREEN	显示驱动
DRIVER_KEYBOARD	键盘驱动
DRIVER_MTJT	鼠标驱动
DRIVER_THREAD_GUI_LOCKER	GUI 同步信号驱动
DRIVER_THREAD_CALLBACK_LOCKER	回调函数同步信号驱动

表 7-1　LearningGUI 驱动类型定义表

注册驱动接口函数声明：

```
int  in_driver_register(unsigned int driver_type, void *driver);
```

其中 driver_type 取值范围如表 7-1 所示，*driver 是用户驱动结构体指针。

注册系统驱动接口就是在准备好*driver 驱动结构体指针数据好，调用 in_driver_register 函数过程。显示驱动是必须的，多数情况下键盘和鼠标驱动是必要的。在单任务系统中，没有必要注册同步信号驱动。

注册驱动函数是 LearningGUI 系统最先调用的函数，必须在 gui_open 函数之前调用，而且各个驱动接口只能调用一次。

7.3　系统显示驱动接口

显示驱动结构体 GUI_SCREEN 定义如下：

```
struct  _GUI_SCREEN
{
    BUINT          is_hline_accelerate;
    BUINT          is_vline_accelerate;
    BUINT          is_rect_fill_accelerate;

    unsigned int   width;
    unsigned int   height;

    int            (*open)(void);
    int            (*close)(void);

    SCREEN_COLOR   (*gui_to_screen_color)(GUI_COLOR  gui_color);
    GUI_COLOR      (*screen_to_gui_color)(SCREEN_COLOR  screen_color);

    int            (*output_sequence_start)(void);
    int            (*output_pixel)(void *context, int x, int y, SCREEN_COLOR color);
    int            (*output_hline)(void *context, int left, int right, int top, SCREEN_COLOR color);
    int            (*output_vline)(void *context, int left, int top, int bottom, SCREEN_COLOR color);
    int            (*output_rect_fill)(void *context, int left, int top, int right, int bottom, SCREEN_COLOR color);
    int            (*output_sequence_end)(void);

    int            (*input_sequence_start)(void);
    int            (*input_pixel)(void *context, int x, int y, SCREEN_COLOR *color);
    int            (*input_sequence_end)(void);

    int            (*control)(void *p1, void *p2);
    int            (*on)(void);
```

```
    int             (*off)(void);
    int             (*reinit)(void);

    #ifndef  _LG_WINDOW_
    int             (*clear)(void *context);
    #endif
};
typedef   struct     _GUI_SCREEN  GUI_SCREEN;
```

GUI_SCREEN 结构体各个成员描述如表 7-2 所示。

结构体成员	描述说明
is_hline_accelerate	是否使用画水平线硬件加速标志 如果数值是 0，使用打点函数输出；否则使用硬件加速函数
is_vline_accelerate	是否使用画垂直线硬件加速标志 如果数值是 0，使用打点函数输出；否则使用硬件加速函数
is_rect_fill_accelerate	使用使用填充矩形硬件加速标志 如果数值是 0，使用打点函数输出；否则使用硬件加速函数
width	屏宽度（像素）
height	屏高度（像素）
(*open)(void)	打开屏函数指针
(*close)(void)	关闭屏函数指针
(*gui_to_screen_color)(GUI_COLOR gui_color)	逻辑颜色转换物理颜色函数指针
(*screen_to_gui_color)(SCREEN_COLOR screen_color)	物理颜色转换逻辑颜色函数指针
(*output_sequence_start)(void)	写屏前，系统所执行的函数指针
(*output_pixel)(void *context, int x, int y, SCREEN_COLOR color)	写像素点函数指针（打点函数）
(*output_hline)(void *context, int left, int right, int top, SCREEN_COLOR color)	画水平线硬件加速函数指针
(*output_vline)(void *context, int left, int top, int bottom, SCREEN_COLOR color)	画垂直线硬件加速函数指针
(*output_rect_fill)(void *context, int left, int top, int right, int bottom, SCREEN_COLOR color)	填充矩形硬件加速函数指针
(*output_sequence_end)(void)	写屏结束后，系统所执行的函数指针
(*input_sequence_start)(void)	读屏前，系统所执行的函数指针
(*input_pixel)(void *context, int x, int y, SCREEN_COLOR *color)	读像素点函数指针（读点函数）。鼠标指针显示、抓屏等功能需要调用此函数。
(*input_sequence_end)(void)	读屏结束后，系统所执行的函数指针
(*control)(void *p1, void *p2)	控制屏函数指针
(*on)(void)	开屏函数指针
(*off)(void)	关屏函数指针
(*reinit)(void)	重新初始化屏函数指针
(*clear)(void *context)	清屏函数指针（Basic 版专有函数）

表 7-2 GUI_SCREEN 结构体成员描述表

注册显示驱动代码示例代码如下：

```
int  register_screen(void)
{
    GUI_SCREEN  screen ;
```

```
    memset(&screen, 0, sizeof(screen));

    screen.is_hline_accelerate      = 1;                        /* 画水平线硬件加速标志 */
    screen.is_vline_accelerate      = 1;                        /* 画垂直线硬件加速标志 */
    screen.is_rect_fill_accelerate  = 1;                        /* 填充矩形硬件加速标志 */

    screen.width                    = 640;                      /* 屏宽度 */
    screen.height                   = 480;                      /* 屏高度 */

    screen.open                     = fb_open;                  /* 用户自定义函数 */
    screen.close                    = fb_close;                 /* 用户自定义函数 */

    screen.gui_to_screen_color      = in_gui_to_r8g8b8;         /* 用户自定义函数 */
    screen.screen_to_gui_color      = in_r8g8b8_to_gui;         /* 用户自定义函数 */

    screen.output_sequence_start    = fb_output_sequence_start; /* 用户自定义函数 */
    screen.output_pixel             = fb_output_pixel;          /* 用户自定义函数 */
    screen. output_hline            = fb_output_hline;          /* 用户自定义函数 */
    screen. output_vline            = fb_output_vline;          /* 用户自定义函数 */
    screen. output_rect_fill        = fb_output_rect_fill;      /* 用户自定义函数 */
    screen.output_sequence_end      = fb_output_sequence_end;   /* 用户自定义函数 */

    screen.input_sequence_start     = fb_input_sequence_start;  /* 用户自定义函数 */
    screen.input_pixel              = fb_input_pixel;           /* 用户自定义函数 */
    screen.input_sequence_end       = fb_input_sequence_end;    /* 用户自定义函数 */

    screen.control                  = fb_control;               /* 用户自定义函数 */
    screen.on                       = fb_on;                    /* 用户自定义函数 */
    screen.off                      = fb_off;                   /* 用户自定义函数 */
    screen.reinit                   = fb_reinit;                /* 用户自定义函数 */

    #ifndef  _LG_WINDOW_
    screen.clear                    = fb_clear;                 /* 用户自定义函数 */
    #endif

    in_driver_register(DRIVER_SCREEN, &screen);

    return  1;
}
```

其中显示鼠标指针和屏幕快照等功能需要回读接口；如果显示屏幕不支持回读接口，则不能使用显示鼠标指针和屏幕快照功能。在屏幕不支持回读接口而用户需要使用二者功能情况下，需要用户在驱动接口中开辟显存镜像，从显存镜像中进行回读。用户负责维护显存镜像。

7.4 系统键盘驱动接口

键盘驱动结构体 GUI_KEYBOARD 定义如下：

```
struct  _GUI_KEYBOARD
{
    int  (*open)(void);
    int  (*close)(void);

    int  (*read)(void *msg);
    int  (*write)(void *buffer, unsigned int len);

    int  (*control)(void *p1, void *p2);
    int   (*reinit)(void);
};
typedef   struct     _GUI_KEYBOARD  GUI_KEYBOARD;
```

GUI_KEYBOARD 结构体各个成员描述如表 7-3 所示。

结构体成员	描述说明
(*open)(void)	打开键盘函数指针

(*close)(void)	关闭键盘函数指针
(*read)(void *msg)	读键盘函数指针
(*write)(void *buffer, unsigned int len)	写键盘函数指针
(*control)(void *p1, void *p2)	控制键盘函数指针
(*reinit)(void)	重新初始化键盘函数指针

表 7-3 GUI_KEYBOARD 结构体成员描述表

注册键盘驱动代码示例代码如下：

```
int  register_keyboard(void)
{
    GUI_KEYBOARD   keyboard;

    memset(&keyboard, 0, sizeof(keyboard));

    keyboard.open        = input_open_key;     /* 用户自定义函数 */
    keyboard.close       = input_close_key;    /* 用户自定义函数 */

    keyboard.read        = input_read_key;     /* 用户自定义函数 */
    keyboard.write       = input_write_key;    /* 用户自定义函数 */

    keyboard.control     = input_control_key; /* 用户自定义函数 */
    keyboard.reinit      = input_reinit_key;  /* 用户自定义函数 */

    in_driver_register(DRIVER_KEYBOARD, &keyboard);

    return  1;
}
```

7.5 系统 MTJT（鼠标）驱动接口

MTJT 驱动结构体 GUI_MTJT 定义如下：

```
struct  _GUI_MTJT
{
    int   cur_x;                          /* 保留系统内部使用 */
    int   cur_y;                          /* 保留系统内部使用 */

    int  (*open)(void);
    int  (*close)(void);

    int  (*read)(void *msg);
    int  (*write)(void *buffer, unsigned int len);

    int  (*control)(void *p1, void *p2);
    int   (*reinit)(void);
};
typedef   struct     _GUI_MTJT  GUI_MTJT;
```

GUI_MTJT 结构体各个成员描述如表 7-4 所示。

结构体成员	描述说明
(*open)(void)	打开 MTJT 函数指针
(*close)(void)	关闭 MTJT 函数指针
(*read)(void *msg)	读 MTJT 函数指针
(*write)(void *buffer, unsigned int len)	写 MTJT 函数指针
(*control)(void *p1, void *p2)	控制 MTJT 函数指针
(*reinit)(void)	重新初始化 MTJT 函数指针

表 7-4 GUI_MTJT 结构体成员描述表

注册 MTJT 驱动代码示例代码如下：

```
int  register_mtjt(void)
```

```
{
    GUI_MTJT   mtjt;

    memset(&mtjt, 0, sizeof(mtjt));

    mtjt.open        = input_open_mtjt;         /* 用户自定义函数 */
    mtjt.close       = input_close_mtjt;        /* 用户自定义函数 */

    mtjt.read        = input_read_mtjt;         /* 用户自定义函数 */
    mtjt.write       = input_write_mtjt;        /* 用户自定义函数 */

    mtjt.control     = input_control_mtjt;      /* 用户自定义函数 */
    mtjt.reinit      = input_reinit_mtjt;       /* 用户自定义函数 */

    in_driver_register(DRIVER_MTJT, &mtjt);

    return  1;
}
```

7.6 同步信号驱动接口

同步信号驱动结构体 GUI_LOCKER 定义如下：

```
struct  _GUI_LOCKER
{
    int  (*init)(void);
    int  (*destroy)(void);

    int  (*lock)(void);
    int  (*unlock)(void);
};
typedef  struct  _GUI_LOCKER   GUI_LOCKER;
```

GUI_LOCKER 结构体各个成员描述如表 7-5 所示。

结构体成员	描述说明
(*init)(void)	初始化同步信号函数指针
(*destroy)(void)	摧毁同步信号函数指针
(*lock)(void)	加锁函数指针
(*unlock)(void)	解锁函数指针

表 7-5 GUI_LOCKER 结构体成员描述表

注册同步信号驱动代码示例代码如下：

```
int  register_locker(void)
{
    GUI_LOCKER   locker;

    memset(&locker, 0, sizeof(locker));
    locker.init        = user_gui_init_lock;            /* 用户自定义函数 */
    locker.lock        = user_gui_lock;                 /* 用户自定义函数 */
    locker.unlock      = user_gui_unlock;               /* 用户自定义函数 */
    locker.destroy     = user_gui_destroy_lock;         /* 用户自定义函数 */
    in_driver_register(DRIVER_THREAD_GUI_LOCKER, &locker);

    memset(&locker, 0, sizeof(locker));
    locker.init        = user_callback_init_lock;       /* 用户自定义函数 */
    locker.lock        = user_callback_lock;            /* 用户自定义函数 */
    locker.unlock      = user_callback_unlock;          /* 用户自定义函数 */
    locker.destroy     = user_callback_destroy_lock;    /* 用户自定义函数 */
    in_driver_register(DRIVER_THREAD_CALLBACK_LOCKER, &locker);

    return  1;
}
```

第八章　消息（事件）及回调函数机制

8.1　消息及回调函数机制概述

LearningGUI 系统是基于消息驱动的系统。应用主循环是一个消息处理循环。回调函数是消息处理路由。用户可以改变消息处理路由。系统消息划分为系统预定义消息和用户自定义消息，有的系统消息只能由系统来操作处理。同时根据消息处理方式，将消息划分即时消息和邮寄消息两种，系统内置邮寄消息队列，即时消息就是不经过系统邮寄消息队列，直接发送给消息路由来执行处理的消息；邮寄消息是发送给系统内置邮寄消息队列，排队延时处理的消息。即时消息和邮寄消息分别由不同的函数接口来处理。

回调函数本质是一个函数，但是回调函数不是由用户主动调用的，而是由系统自动调用的。用户编写的回调函数是用户回调函数，系统内置的回调函数是系统回调函数。

8.2　GUI_MESSAGE 结构体及相关数据结构定义

消息结构体 GUI_MESSAGE 相关定义如下：

```
union  _GUI_MESSAGE_UNION
{
    int           value;
    struct
    {
        UINT16    high;
        UINT16    low;
    };
    struct
    {
        UINT8     a;
        UINT8     b;
        UINT8     c;
        UINT8     d;
    };
    int           *pint;
    INT16         *pint16;
    char          *pchar;
};
typedef  union  _GUI_MESSAGE_UNION    GUI_MESSAGE_UNION;

struct  _GUI_MESSAGE
{
    UINT               id;
    #ifdef  _LG_WINDOW_
    void              *to_hwnd;
    int                callback_flag;
    #endif  /* _LG_WINDOW_ */
    GUI_MESSAGE_UNION  data0;
    GUI_MESSAGE_UNION  data1;
    GUI_MESSAGE_UNION  data2;
    #ifdef  _LG_LONG_MESSAGE_
    GUI_MESSAGE_UNION  data3;
    GUI_MESSAGE_UNION  data4;
    #endif  /* _LG_LONG_MESSAGE_ */
};
typedef  struct  _GUI_MESSAGE    GUI_MESSAGE;
```

GUI_MESSAGE_UNION 是消息数据联合体，适应消息的不同数据类型。GUI_MESSAGE 结构成员描述如表 8-1 所示。

成员	描述
id	消息 ID
*to_hwnd	处理消息的窗口（Windows 专用）
callback_flag	消息回调标志（Windows 专用）
data0	0 号数据成员

data1	1 号数据成员
data2	2 号数据成员
data3	3 号数据成员（需要_LG_LONG_MESSAGE_宏支持）
data4	4 号数据成员（需要_LG_LONG_MESSAGE_宏支持）

表 8-1 GUI_MESSAGE 结构体成员表

当用户生成一条新消息时，用户可以对 GUI_MESSAGE 中成员直接赋值。但是当解析或者修改消息时，禁止用户直接对 GUI_MESSAGE 中成员进行修改，只能通过相关的消息接口函数来操作 GUI_MESSAGE 中的成员。

8.3 系统预定义消息 ID

系统预定义消息 ID 如表 8-2 所示。

消息 ID	消息描述	消息分类
MSG_QUIT	退出系统消息	退出
MSG_KEY_DOWN	键盘按键按下消息	键盘
MSG_KEY_UP	键盘按键释放消息	
MSG_KEY_CHAR	键盘字符输入消息	
MSG_MTJT_MOVE	鼠标移动消息	鼠标
MSG_MTJT_LBUTTON_DOWN	鼠标左键按下消息	
MSG_MTJT_LBUTTON_UP	鼠标左键释放消息	
MSG_MTJT_LBUTTON_DBLCLK	鼠标左键双击消息	
MSG_MTJT_RBUTTON_DOWN	鼠标右键按下消息	
MSG_MTJT_RBUTTON_UP	鼠标右键释放消息	
MSG_MTJT_RBUTTON_DBLCLK	鼠标右键双击消息	
MSG_MTJT_MBUTTON_DOWN	鼠标中键按下消息	
MSG_MTJT_MBUTTON_UP	鼠标中键释放消息	
MSG_MTJT_MBUTTON_DBLCLK	鼠标中键双击消息	
MSG_MTJT_WHEEL	鼠标滚动消息	
MSG_COUNTER	计数器消息	计数器
MSG_TIMER	定时器消息	定时器
MSG_CREATE	窗口创建消息	Windows 消息
MSG_CREATE_NEXT	窗口创建后继消息	
MSG_CLOSE	窗口关闭	
MSG_CLOSE_NEXT	窗口关闭后续消息	
MSG_PAINT	窗口绘制消息	
MSG_PAINT_NEXT	窗口绘制后续消息	
MSG_HIDE	窗口隐藏消息	
MSG_MAXIZE	窗口极大化消息	
MSG_NORMAL	窗口恢复正常显示消息	
MSG_MOVE	窗口移动消息	
MSG_RESIZE	窗口调整大小消息	
MSG_GET_FOCUS	窗口得到焦点消息	
MSG_LOST_FOCUS	窗口失去焦点消息	
MSG_CARET	显示定位符消息	
MSG_NOTIFY_SEL_CHANGED	在列表项中选择项发生变化消息	

MSG_NOTIFY_VALUE_CHANGED	显示数值发生变化消息	
MSG_USER	用户自定义消息基数消息	用户消息

表 8-2 系统预定义消息 ID 表

8.4 用户自定义消息 ID

用户可以自定义多个消息，消息 ID 依次从(MSG_USER+1)递增。如用户自定义 MSG_DISPLAY_USER_DATA、MSG_DISPLAY_USER_TIME 消息如下：

```
#define    MSG_DISPLAY_USER_DATA        (MSG_USER+1)
#define    MSG_DISPLAY_USER_TIME        (MSG_USER+2)
```

在使用上，用户自定义的消息和其它的消息没有区别。

8.5 Windows 接收消息句柄

在 Windows 中，在绝大数情况下，消息的发送和处理都是依据窗口句柄来进行的。GUI_MESSAGE 中*to_hwnd 成员是表示消息所发送或者接受的窗口句柄。

8.6 Windows 消息处理标志

在 Windows 中，消息处理标志定义为 HWND_APP_CALLBACK 和 HWND_IN_CALLBACK，二者可以或运算，HWND_APP_CALLBACK 表示消息需要在窗口用户回调函数中处理，HWND_IN_CALLBACK 表示消息需要在窗口系统回调函数中处理。

8.7 消息数据项

当用户处理一条消息时，有时候需要解析消息中的数据项；当用户生成一条新消息时，往往需要填充消息中的数据项。GUI_MESSSAGE 包含多个数据项：data0、data1、data2 等，对于不同的消息，每个数据项并不是必须的，甚至可以忽略数据项。数据项成员的使用是依赖不同的消息，有的系统消息由系统自动管理数据项，有的系统消息则是由用户来解析数据项，而用户消息则完全由用户来解析或者填充相关的数据项。

8.7.1 键盘消息数据项

键盘消息数据项是表示系统键码。系统键码定义如表 8-3 所示。

键码	键码描述	键码分类
GUI_KEY_ESCAPE	ESC 退出键	退出
GUI_KEY_BACKSPACE	BACKSPACE 回格键	回格
GUI_KEY_TAB	TAB 键	TAB
GUI_KEY_BACK_TAB	BACK TAB 键	
GUI_KEY_PAUSE	暂停键	暂停
GUI_KEY_PRINT	屏幕打印键	打印请求
GUI_KEY_SYS_REQ	系统请求键	
GUI_KEY_CLEAR	清除键	编辑功能键
GUI_KEY_INSERT	插入键	
GUI_KEY_ENTER	回车键	
GUI_KEY_DELETE	删除键	
GUI_KEY_FIRST_MOVE	移动开始键码	移动
GUI_KEY_LEFT	左移键	
GUI_KEY_RIGHT	右移键	
GUI_KEY_UP	上移键	
GUI_KEY_DOWN	下移键	
GUI_KEY_HOME	移到头键	

GUI_KEY_END	移到尾键	
GUI_KEY_PAGE_UP	上翻页键	
GUI_KEY_PAGE_DOWN	下翻页键盘	
GUI_KEY_END_MOVE	移动结束键码	
GUI_KEY_FIRST_CHAR	ASCII 开始码	
GUI_KEY_SPACE		
GUI_KEY_EXCLAM	!	
GUI_KEY_QUOTE_DBL	\	
GUI_KEY_NUMBER_SIGN	#	
GUI_KEY_DOLLAR	$	
GUI_KEY_PERCENT	%	
GUI_KEY_AMPERSAND	&	
GUI_KEY_APOSTROPHE	'	
GUI_KEY_PAREN_LEFT	(	
GUI_KEY_PAREN_RIGHT	)	
GUI_KEY_ASTERISK	*	
GUI_KEY_PLUS	+	
GUI_KEY_COMMA	,	
GUI_KEY_MINUS	-	
GUI_KEY_DOT	.	
GUI_KEY_SLASH	/	
GUI_KEY_0	0	
GUI_KEY_1	1	ASCII 键
GUI_KEY_2	2	
GUI_KEY_3	3	
GUI_KEY_4	4	
GUI_KEY_5	5	
GUI_KEY_6	6	
GUI_KEY_7	7	
GUI_KEY_8	8	
GUI_KEY_9	9	
GUI_KEY_COLON	:	
GUI_KEY_SEMICOLON	;	
GUI_KEY_LESS	<	
GUI_KEY_EQUAL	=	
GUI_KEY_GREATER	>	
GUI_KEY_QUESTION	?	
GUI_KEY_AT	@	
GUI_KEY_A	A	
GUI_KEY_B	B	
GUI_KEY_C	C	
GUI_KEY_D	D	
GUI_KEY_E	E	

GUI_KEY_F	F
GUI_KEY_G	G
GUI_KEY_H	H
GUI_KEY_I	I
GUI_KEY_J	J
GUI_KEY_K	K
GUI_KEY_L	L
GUI_KEY_M	M
GUI_KEY_N	N
GUI_KEY_O	O
GUI_KEY_P	P
GUI_KEY_Q	Q
GUI_KEY_R	R
GUI_KEY_S	S
GUI_KEY_T	T
GUI_KEY_U	U
GUI_KEY_V	V
GUI_KEY_W	W
GUI_KEY_X	X
GUI_KEY_Y	Y
GUI_KEY_Z	Z
GUI_KEY_BRACKET_LEFT	[
GUI_KEY_BACK_SLASH	\
GUI_KEY_BRACKET_RIGHT	]
GUI_KEY_ASCII_CIRCUM	^
GUI_KEY_UNDERSCORE	_
GUI_KEY_GRAVE	`
GUI_KEY_a	a
GUI_KEY_b	b
GUI_KEY_c	c
GUI_KEY_d	d
GUI_KEY_e	e
GUI_KEY_f	f
GUI_KEY_g	g
GUI_KEY_h	h
GUI_KEY_i	i
GUI_KEY_j	j
GUI_KEY_k	k
GUI_KEY_l	l
GUI_KEY_m	m
GUI_KEY_n	n
GUI_KEY_o	o
GUI_KEY_p	p

GUI_KEY_q	q	
GUI_KEY_r	r	
GUI_KEY_s	s	
GUI_KEY_t	t	
GUI_KEY_u	u	
GUI_KEY_v	v	
GUI_KEY_w	w	
GUI_KEY_x	x	
GUI_KEY_y	y	
GUI_KEY_z	z	
GUI_KEY_BRACE_LEFT	{	
GUI_KEY_BAR	|	
GUI_KEY_BRACE_RIGHT	}	
GUI_KEY_ASCII_TILDE	~	
GUI_KEY_END_CHAR	ASCII 结束码	
GUI_KEY_F1	F1	功能
GUI_KEY_F2	F2	
GUI_KEY_F3	F3	
GUI_KEY_F4	F4	
GUI_KEY_F5	F5	
GUI_KEY_F6	F6	
GUI_KEY_F7	F7	
GUI_KEY_F8	F8	
GUI_KEY_F9	F9	
GUI_KEY_F10	F10	
GUI_KEY_F11	F11	
GUI_KEY_F12	F12	
GUI_KEY_SHIFT	Shift 键	组合
GUI_KEY_CONTROL	Ctrl 键	
GUI_KEY_ALT	Alt 键	
GUI_KEY_META	Meta 键	
GUI_KEY_CAPS_LOCK	CapsLock 键	控制
GUI_KEY_NUM_LOCK	NumLock 键	
GUI_KEY_SCROLL_LOCK	ScrollLock 键	
GUI_KEY_UNKNOWN		未知

表 8-3 系统键码定义表

在键盘驱动接口中，用户需要将键盘扫描码转换成系统键码。

8.8 回调函数及其消息路由

回调函数指针 GUI_CALLBACK 声明如下：

```
typedef  int  (*GUI_CALLBACK)(GUI_MESSAGE *msg)
```

用户必须依据 GUI_CALLBACK 来定义需要的回调函数。

系统依据 MESSAGE_QUEUE_LEN 宏定义了消息队列，用于缓存邮寄消息，并且依次逐条处理邮寄消息。当系统获取一条邮寄消息后，首先调用用户当前的消息处理回调函数，如果该回调函数返回值大于 0，则继续调用 Windows 默认的消息回调函数（如果是 Windows 版的话），之后调用系统默认的消息处理回调函数。

8.9 消息操作接口函数

GUI_MESSAGE 处理操作都是通过函数接口进行的。

8.9.1 获取消息 ID 宏：MESSAGE_GET_ID( )

宏原型：MESSAGE_GET_ID(GUI_MESSAGE *msg)
返回 MSG 中 ID。

8.9.2 设置消息 ID 宏：MESSAGE_SET_ID( )

宏原型：MESSAGE_SET_ID(GUI_MESSAGE *msg, UINT id)
将 MSG 中的 ID 替换成 id。

8.9.3 获取键码值：MESSAGE_GET_KEY_VALUE( )

宏原型：MESSAGE_GET_KEY_VALUE(GUI_MESSAGE *msg)
返回 MSG_KEY_DOWN 或者 MSG_KEY_UP 消息中键码。

8.9.4 设置键码值：MESSAGE_SET_KEY_VALUE( )

宏原型：MESSAGE_SET_KEY_VALUE(GUI_MESSAGE *msg, int value)
将 MSG_KEY_DOWN 或者 MSG_KEY_UP 消息中键码替换成 value。

8.9.5 是否是鼠标类消息：MESSAGE_IS_MTJT( )

宏原型：MESSAGE_IS_MTJT(GUI_MESSAGE *msg)
如果是鼠标类消息，返回 1；否则返回 0。

8.9.6 获取鼠标 X 坐标值：MESSAGE_GET_MTJT_X( )

宏原型：MESSAGE_GET_MTJT_X(GUI_MESSAGE *msg)
返回鼠标 X 坐标值。

8.9.7 设置鼠标 X 坐标值：MESSAGE_SET_MTJT_X( )

宏原型：MESSAGE_SET_MTJT_X(GUI_MESSAGE *msg, int x)
设置鼠标 X 坐标值。

8.9.8 获取鼠标 Y 坐标值：MESSAGE_GET_MTJT_Y( )

宏原型：MESSAGE_GET_MTJT_Y(GUI_MESSAGE *msg)
返回鼠标 Y 坐标值。

8.9.9 设置鼠标 Y 坐标值：MESSAGE_SET_MTJT_Y( )

宏原型：MESSAGE_SET_MTJT_Y(GUI_MESSAGE *msg, int y)
设置鼠标 Y 坐标值。

8.9.10 获取计数器 ID 宏：MESSAGE_GET_COUNTER_ID( )

宏原型：MESSAGE_GET_COUNTER_ID(GUI_MESSAGE *msg)
返回计数器 ID。

8.9.11 设置计数器 ID 宏：MESSAGE_SET_COUNTER_ID( )

宏原型：MESSAGE_SET_COUNTER_ID(GUI_MESSAGE *msg, UINT id)
将计数器 ID 替换成 id。

8.9.12 获取定时器 ID 宏：MESSAGE_GET_TIMER_ID( )

宏原型：MESSAGE_GET_TIMER_ID(GUI_MESSAGE *msg)
返回定时器 ID。

8.9.13 设置定时器 ID 宏：MESSAGE_SET_TIMER_ID( )

宏原型：MESSAGE_SET_TIMER_ID(GUI_MESSAGE *msg, UINT id)
将定时器 ID 替换成 id。

8.9.14 是否是 MSG_QUIT 退出消息：MESSAGE_IS_QUIT( )

宏原型：MESSAGE_IS_QUIT(GUI_MESSAGE *msg)
如果是 MSG_QUIT 退出消息，返回 1；否则返回 0。

8.9.15 设置消息路由：message_set_routine( )

函数原型：int message_set_routine(GUI_CALLBACK routine)
返回 1。
说明：用户可以更改不同的用户消息路由，该函数用于设置当前的用户消息路由。

8.9.16 邮寄消息：message_post( )

函数原型：int message_post(GUI_MESSAGE *msg)
如果消息放入系统消息队列，返回 1；否则返回 0。

8.9.17 邮寄 MSG_QUIT 退出消息：message_post_quit( )

函数原型：int message_post_quit(void)
如果 MSG_QUIT 消息放入系统消息队列，返回 1；否则返回 0。

8.9.18 清空消息队列：message_clear_queue( )

函数原型：int message_clear_queue(void)
返回 1。
说明：该函数清空邮寄消息。

8.9.19 发送即时消息：message_send( )

函数原型：int message_send(GUI_MESSAGE *msg)
返回 1。
说明：立即执行消息路由。

8.9.20 获取消息：message_get( )

函数原型：int message_get(GUI_MESSAGE *msg)
如果获取到消息，返回 1；否则返回 0。
说明：该函数从消息队列中取出一条消息，或者生成一条消息，拷贝到 msg 中。

8.9.21 调度消息：message_dispatch( )

函数原型：int message_dispatch(GUI_MESSAGE *msg)
返回 1。
说明：该函数等同于 message_send 函数。

8.9.22 调度队列中所有的消息：message_dispatch_all( )

函数原型：int message_dispatch_all(void)
返回 1。
说明：该函数一次性将消息队列中所有消息调度完成。

消息函数操作接口汇总如表 8-4 所示。

函数原型	函数简述	分类
MESSAGE_GET_ID(GUI_MESSAGE *msg)	获取消息 ID	宏
MESSAGE_SET_ID(GUI_MESSAGE *msg, UINT id)	设置消息 ID	
MESSAGE_GET_KEY_VALUE(GUI_MESSAGE *msg)	获取键码值	
MESSAGE_SET_KEY_VALUE(GUI_MESSAGE *msg, int value)	设置键码值	
MESSAGE_IS_MTJT(GUI_MESSAGE *msg)	是否是鼠标类消息	

MESSAGE_GET_MTJT_X(GUI_MESSAGE *msg)	获取鼠标 X 坐标值	
MESSAGE_SET_MTJT_X(GUI_MESSAGE *msg, int x)	设置鼠标 X 坐标值	
MESSAGE_GET_MTJT_Y(GUI_MESSAGE *msg)	获取鼠标 Y 坐标值	
MESSAGE_SET_MTJT_Y(GUI_MESSAGE *msg, int y)	设置鼠标 Y 坐标值	
MESSAGE_GET_COUNTER_ID(GUI_MESSAGE *msg)	获取计数器 ID	
MESSAGE_SET_COUNTER_ID(GUI_MESSAGE *msg, UINT id)	设置计数器 ID	
MESSAGE_GET_TIMER_ID(GUI_MESSAGE *msg)	获取定时器 ID	
MESSAGE_SET_TIMER_ID(GUI_MESSAGE *msg, UINT id)	设置定时器 ID	
MESSAGE_IS_QUIT(GUI_MESSAGE *msg)	是否是 MSG_QUIT 退出消息	
int message_set_routine(GUI_CALLBACK routine)	设置消息路由	
int message_post(GUI_MESSAGE *msg)	邮寄消息	
int message_post_quit(void)	邮寄 MSG_QUIT 退出消息	
int message_clear_queue(void)	清空消息队列	函数
int message_send(GUI_MESSAGE *msg)	发送即时消息	
int message_get(GUI_MESSAGE *msg)	获取消息	
int message_dispatch(GUI_MESSAGE *msg)	调度消息	
int message_dispatch_all(void)	调度队列中所有的消息	

表 8-4 消息操作接口函数汇总表

8.10 系统消息循环

系统消息循环就是不断轮询消息队列，如果获得新消息，那么立即处理该消息；如果没有得到消息，则暂停当前线程执行一段时间，或者休眠一段时间。

系统消息循环示例代码如下：

```
/* 系统消息循环 */
GUI_MESSAGE  msg;
int  ret;

while ( 1 )
{
    ret = message_get(&msg);
    if (ret > 0)
    {
        if ( MESSAGE_IS_QUIT(&msg) )
            break;

        message_dispatch(&msg);
    } else {
        /* POSIX 函数*/
        usleep(10);
    }
}
```

8.11 计数器 COUNTER 消息

计数器 COUNTER 表示系统消息循环的执行次数，同时在无时间系统中，可以作为粗略的定时器。计数器由用户创建和删除。在计数器创建后，计数器消息 MSG_COUNTER 由系统产生，其消息捕捉由用户来进行，当用户不再需要该计数器时，用户负责删除该计数器。

8.11.1 创建计数器：gui_counter_create( )

函数原型：int gui_counter_create(unsigned int id, unsigned int counter, void *para)

参数：

id 表示计数器 ID，全局唯一。

counter 表示计数间隔次数。

para 表示附加输入参数。在 Windows 中表示捕捉 MSG_COUNTER 消息的窗口句柄。

返回值：如果创建成功，返回 1；否则返回-1。

8.11.2 删除计数器：gui_counter_delete( )

函数原型：int gui_counter_delete(unsigned int id)

参数： id 表示计数器 ID。

返回值：如果删除成功，返回 1；否则返回-1。

8.12 定时器 TIMER 消息

定时器 TIMER 表示用户定时一段时间间隔，时间间隔流逝后，产生 MSG_TIMER 消息。定时器由用户创建和删除。在定时器创建后，定时器消息 MSG_TIMER 由系统产生，其消息捕捉由用户来进行，当用户不再需要该定时器时，用户负责删除该定时器。

定时器需要 POSIX 标准中 time 函数支持。

8.12.1 创建定时器：gui_timer_create( )

函数原型：int gui_timer_create(unsigned int id, unsigned int millisecond, void *para)

参数：

id 表示定时器 ID，全局唯一。

millisecond 表示定时毫秒数。

para 表示附加输入参数。在 Windows 中表示捕捉 MSG_TIMER 消息的窗口句柄。

返回值：如果创建成功，返回 1；否则返回-1。

8.12.2 删除定时器：gui_timer_delete( )

函数原型：int gui_timer_delete(unsigned int id)

参数： id 表示定时器 ID。

返回值：如果删除成功，返回 1；否则返回-1。

第九章　字体及字符集

9.1　字体及字符集概述

系统根据字形规则性将字体划分为两种类型：单一字体(MONO)和混合字体(MIXED)（非单一字体），单一字体是所有字体具备相同的高度和相同的宽度，混合字体是所有字体并不具备相同的高度或相同的宽度。同时，系统根据字符编码规则将字体划分为两种类型：字符集字体（CHARSET）和离散字体(DISCRETE)。因此，LearningGUI 系统支持四种类型子字库：单一字符集子字库（MONO_CHARSET_FONT_TYPE）、单一离散子字库（MONO_DISCRETE_FONT_TYPE）、混合字符集子字库（MIXED_CHARSET_FONT_TYPE）、混合离散子字库（MIXED_DISCRETE_FONT_TYPE）。

系统提供开放式字库访问接口。在用户定义字库时，不仅定义字库的数据，而且需要定义字库的访问方法，因此从理论来说，系统支持任何字符集访问。系统支持单字节、多字节字符编码体系，也能够支持 Unicode 字符编码体系，但二者互斥。

系统不仅提供 ASCII 字库，而且提供 GB2312-80 标准汉字字库，并且可以裁减 GB2312-80 字库。同时，系统提供中日韩通用 CJK Unicode 字库。

多字节体系需要_LG_MULTI_BYTE_CODE_VERSION_宏支持，Unicode 体系需要_LG_UNICODE_VERSION_宏支持。

9.2　TCHAR 通用字符数据类型

为适应 Unicode 字符编码体系和非 Unicode 字符编码体系，系统定义了通用字符数据类型：TCHAR。其定义如下：

```
#ifdef  _LG_UNICODE_VERSION_
#define  TCHAR                    UINT16
#else
#define  TCHAR                    char
#endif
```

在 Unicode 编码系统中，TCHAR 定义为 UINT16；在非 Unicode 体系中，TCHAR 定义为 char。

9.3　MONO_CHARSET_FONT 单一字符集子字库

系统定义 MONO_CHARSET_FONT 结构体来标示单一字符集子字库。MONO_CHARSET_FONT 结构体定义如下：

```
struct _MONO_CHARSET_FONT
{
    UCHAR                       name[FONT_NAME_LEN];
    UCHAR                       id;
    int                         (*is_in_this_charset_block)(const TCHAR *code);
    unsigned int                (*get_data_start_index)(const TCHAR *code);
    UCHAR                       width;
    UCHAR                       height;
    void                       *data;
    struct _MONO_CHARSET_FONT  *next;
};
typedef   struct _MONO_CHARSET_FONT  MONO_CHARSET_FONT;
```

MONO_CHARSET_FONT 结构成员描述如表 9-1 所示。

成员	描述
name[FONT_NAME_LEN]	子字库名称
id	子字库 ID
int (*is_in_this_charset_block)(const TCHAR *code)	判断字符 code 是否属于本子字库函数指针
unsigned int (*get_data_start_index)(const TCHAR *code)	获得字符 code 开始索引号函数指针
width	字符宽度
height	字符高度

*data	子字库数据指针
*next	后续 MONO_CHARSET_FONT 指针

表 9-1 MONO_CHARSET_FONT 结构体成员表

例如，8*16 ASCII MONO_CHARSET_FONT 子字库 gasc08 定义如下：

```
static  const  UCHAR  ascii_d0816_data[]  =
{
    /* " " */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /* "!" */
    0x00, 0x00, 0x18, 0x3C, 0x3C, 0x3C, 0x18, 0x18,
    0x18, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
    /* "\" */
    0x00, 0x66, 0x66, 0x66, 0x44, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /* "#" */
    0x00, 0x00, 0x00, 0x6C, 0x6C, 0xFE, 0x6C, 0x6C,
    0x6C, 0xFE, 0x6C, 0x6C, 0x00, 0x00, 0x00, 0x00,
    /* "$" */
    0x18, 0x18, 0x7C, 0xC6, 0xC2, 0xC0, 0x7C, 0x06,
    0x06, 0x86, 0xC6, 0x7C, 0x18, 0x18, 0x00, 0x00,
    /* "%" */
    0x00, 0x00, 0x00, 0x00, 0xC2, 0xC6, 0x0C, 0x18,
    0x30, 0x60, 0xC6, 0x86, 0x00, 0x00, 0x00, 0x00,
    /* "&" */
    0x00, 0x00, 0x38, 0x6C, 0x6C, 0x38, 0x76, 0xDC,
    0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00,
    /* "'" */
    0x00, 0x30, 0x30, 0x30, 0x60, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /* "(" */
    0x00, 0x00, 0x0C, 0x18, 0x30, 0x30, 0x30, 0x30,
    0x30, 0x30, 0x18, 0x0C, 0x00, 0x00, 0x00, 0x00,
    /* ")" */
    0x00, 0x00, 0x30, 0x18, 0x0C, 0x0C, 0x0C, 0x0C,
    0x0C, 0x0C, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00,
    /* "*" */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x3C, 0xFF,
    0x3C, 0x66, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /* "+" */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x7E,
    0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /* "," */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x18, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00,
    /* "-" */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    /* "." */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
    /* "/" */
    0x00, 0x00, 0x00, 0x00, 0x02, 0x06, 0x0C, 0x18,
    0x30, 0x60, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00,
    /* "0" */
    0x00, 0x00, 0x38, 0x6C, 0xC6, 0xC6, 0xC6, 0xC6,
    0xC6, 0xC6, 0x6C, 0x38, 0x00, 0x00, 0x00, 0x00,
    /* "1" */
    0x00, 0x00, 0x18, 0x38, 0x78, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x7E, 0x00, 0x00, 0x00, 0x00,
    /* "2" */
    0x00, 0x00, 0x7C, 0xC6, 0x06, 0x0C, 0x18, 0x30,
    0x60, 0xC0, 0xC6, 0xFE, 0x00, 0x00, 0x00, 0x00,
    /* "3" */
    0x00, 0x00, 0x7C, 0xC6, 0x06, 0x06, 0x3C, 0x06,
    0x06, 0x06, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
    /* "4" */
```

```
0x00, 0x00, 0x0C, 0x1C, 0x3C, 0x6C, 0xCC, 0xFE,
0x0C, 0x0C, 0x0C, 0x1E, 0x00, 0x00, 0x00, 0x00,
/* "5" */
0x00, 0x00, 0xFE, 0xC0, 0xC0, 0xC0, 0xFE, 0x06,
0x06, 0x06, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "6" */
0x00, 0x00, 0x38, 0x60, 0xC0, 0xC0, 0xFE, 0xC6,
0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "7" */
0x00, 0x00, 0xFE, 0xC6, 0x06, 0x06, 0x0C, 0x18,
0x30, 0x30, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00,
/* "8" */
0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0x7C, 0xC6,
0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "9" */
0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0x7E, 0x06,
0x06, 0x06, 0x0C, 0x78, 0x00, 0x00, 0x00, 0x00,
/* ":" */
0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00,
0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00,
/* ";" */
0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00,
0x00, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00,
/* "<" */
0x00, 0x00, 0x00, 0x06, 0x0C, 0x18, 0x30, 0x60,
0x30, 0x18, 0x0C, 0x06, 0x00, 0x00, 0x00, 0x00,
/* "=" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x00, 0x00,
0x7E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
/* ">" */
0x00, 0x00, 0x00, 0xC0, 0x30, 0x18, 0x0c, 0x06,
0x0C, 0x18, 0x30, 0x60, 0x00, 0x00, 0x00, 0x00,
/* "?" */
0x00, 0x00, 0x7C, 0xC6, 0xC6, 0x0C, 0x18, 0x18,
0x18, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
/* "@" */
0x00, 0x00, 0x00, 0x78, 0xC6, 0xC6, 0xDE, 0xDE,
0xDE, 0xDC, 0xC0, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "A" */
0x00, 0x00, 0x10, 0x38, 0x6C, 0xC6, 0xC6, 0xFE,
0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00,
/* "B" */
0x00, 0x00, 0xFC, 0x66, 0x66, 0x66, 0x7C, 0x66,
0x66, 0x66, 0x66, 0xFC, 0x00, 0x00, 0x00, 0x00,
/* "C" */
0x00, 0x00, 0x7C, 0x66, 0xC2, 0xC0, 0xC0, 0xC0,
0xC0, 0xC2, 0x66, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "D" */
0x00, 0x00, 0xF8, 0x6C, 0x66, 0x66, 0x66, 0x66,
0x66, 0x66, 0x6C, 0xF8, 0x00, 0x00, 0x00, 0x00,
/* "E" */
0x00, 0x00, 0xFE, 0x66, 0x62, 0x68, 0x78, 0x68,
0x60, 0x62, 0x66, 0xFE, 0x00, 0x00, 0x00, 0x00,
/* "F" */
0x00, 0x00, 0xFE, 0x66, 0x62, 0x68, 0x78, 0x68,
0x60, 0x60, 0x66, 0xF0, 0x00, 0x00, 0x00, 0x00,
/* "G" */
0x00, 0x00, 0x3C, 0x66, 0xC2, 0xC0, 0xC0, 0xDE,
0xC6, 0xC6, 0x66, 0x3A, 0x00, 0x00, 0x00, 0x00,
/* "H" */
0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xFE, 0xC6,
0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00,
/* "I" */
0x00, 0x00, 0x3C, 0x18, 0x18, 0x18, 0x18, 0x18,
0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "J" */
0x00, 0x00, 0x1E, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C,
0xCC, 0xCC, 0xCC, 0x78, 0x00, 0x00, 0x00, 0x00,
/* "K" */
```

```
0x00, 0x00, 0xE6, 0x66, 0x66, 0x66, 0x78, 0x78,
0x6C, 0x66, 0x66, 0xE6, 0x00, 0x00, 0x00, 0x00,
/* "L" */
0x00, 0x00, 0xF0, 0x60, 0x60, 0x60, 0x60, 0x60,
0x60, 0x62, 0x66, 0xFE, 0x00, 0x00, 0x00, 0x00,
/* "M" */
0x00, 0x00, 0xC6, 0xEE, 0xFE, 0xEF, 0xD6, 0xC6,
0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00,
/* "N" */
0x00, 0x00, 0xC6, 0xE6, 0xF6, 0xFE, 0xDE, 0xCE,
0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00,
/* "O" */
0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "P" */
0x00, 0x00, 0xFC, 0x66, 0x66, 0x66, 0x7C, 0x60,
0x60, 0x60, 0x60, 0xF0, 0x00, 0x00, 0x00, 0x00,
/* "Q" */
0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
0xC6, 0xD6, 0xDE, 0x7C, 0x0C, 0x0E, 0x00, 0x00,
/* "R" */
0x00, 0x00, 0xFC, 0x66, 0x66, 0x66, 0x7C, 0x6C,
0x66, 0x66, 0x66, 0xEC, 0x00, 0x00, 0x00, 0x00,
/* "S" */
0x00, 0x00, 0x7C, 0xC6, 0xC6, 0x60, 0x38, 0x0C,
0x06, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "T" */
0x00, 0x00, 0x7E, 0x7E, 0x5A, 0x18, 0x18, 0x18,
0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "U" */
0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "V" */
0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
0xC6, 0x6C, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00,
/* "W" */
0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xD6, 0xD6,
0xD6, 0xFE, 0xEE, 0x6C, 0x00, 0x00, 0x00, 0x00,
/* "X" */
0x00, 0x00, 0xC6, 0xC6, 0x6C, 0x7C, 0x38, 0x38,
0x7C, 0x6C, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00,
/* "Y" */
0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3C, 0x18,
0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "Z" */
0x00, 0x00, 0xFE, 0xC6, 0x86, 0x0C, 0x18, 0x30,
0x60, 0xC2, 0xC6, 0xFE, 0x00, 0x00, 0x00, 0x00,
/* "[" */
0x00, 0x00, 0x3C, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "\\" */
0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0x70, 0x38,
0x1C, 0x0E, 0x06, 0x02, 0x00, 0x00, 0x00, 0x00,
/* "]" */
0x00, 0x00, 0x38, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C,
0x0C, 0x0C, 0x0C, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "^" */
0x10, 0x38, 0x6C, 0xC6, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
/* "_" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00,
/* "`" */
0x00, 0x30, 0x18, 0x0C, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
/* "a" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x0C, 0x7C,
0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00,
/* "b" */
```

```
0x00, 0x00, 0xE0, 0x60, 0x60, 0x78, 0x6C, 0x66,
0x66, 0x66, 0x66, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "c" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0xC0,
0xC0, 0xC0, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "d" */
0x00, 0x00, 0x1C, 0x0C, 0x0C, 0x3C, 0x6C, 0xCC,
0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00,
/* "e" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0xFE,
0xC0, 0xC0, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "f" */
0x00, 0x00, 0x1C, 0x36, 0x32, 0x30, 0x78, 0x30,
0x30, 0x30, 0x30, 0x78, 0x00, 0x00, 0x00, 0x00,
/* "g" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xCC, 0xCC,
0xCC, 0xCC, 0xCC, 0x7C, 0x0C, 0xCC, 0x78, 0x00,
/* "h" */
0x00, 0x00, 0xE0, 0x60, 0x60, 0x6C, 0x76, 0x66,
0x66, 0x66, 0x66, 0xE6, 0x00, 0x00, 0x00, 0x00,
/* "i" */
0x00, 0x00, 0x18, 0x18, 0x00, 0x38, 0x18, 0x18,
0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "j" */
0x00, 0x00, 0x06, 0x06, 0x00, 0x0E, 0x06, 0x06,
0x06, 0x06, 0x06, 0x06, 0x66, 0x66, 0x3C, 0x00,
/* "k" */
0x00, 0x00, 0xE0, 0x60, 0x60, 0x66, 0x6C, 0x78,
0x78, 0x6C, 0x66, 0xE6, 0x00, 0x00, 0x00, 0x00,
/* "l" */
0x00, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18,
0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00,
/* "m" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xEC, 0xFE, 0xD6,
0xD6, 0xD6, 0xD6, 0xC6, 0x00, 0x00, 0x00, 0x00,
/* "n" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xDC, 0x66, 0x66,
0x66, 0x66, 0x66, 0x66, 0x00, 0x00, 0x00, 0x00,
/* "o" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0xC6,
0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "p" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xDC, 0x66, 0x66,
0x66, 0x66, 0x66, 0x7C, 0x60, 0x60, 0xF0, 0x00,
/* "q" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xCC, 0xCC,
0xCC, 0xCC, 0xCC, 0x7C, 0x0C, 0x0C, 0x1E, 0x00,
/* "r" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xDC, 0x76, 0x66,
0x60, 0x60, 0x60, 0xF0, 0x00, 0x00, 0x00, 0x00,
/* "s" */
0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0x60,
0x70, 0x0C, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00,
/* "t" */
0x00, 0x00, 0x10, 0x30, 0x30, 0xFC, 0x30, 0x30,
0x30, 0x30, 0x36, 0x1C, 0x00, 0x00, 0x00, 0x00,
/* "u" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xCC, 0xCC, 0xCC,
0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00,
/* "v" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0xC6, 0xC6,
0xC6, 0xC6, 0x6C, 0x38, 0x00, 0x00, 0x00, 0x00,
/* "w" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0xC6, 0xD6,
0xD6, 0xD6, 0xFE, 0x6C, 0x00, 0x00, 0x00, 0x00,
/* "x" */
0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0x6C, 0x38,
0x38, 0x38, 0x6C, 0xC6, 0x00, 0x00, 0x00, 0x00,
/* "y" */
```

```
    0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0xC6, 0xC6,
    0xC6, 0xC6, 0xC6, 0x7E, 0x06, 0x0C, 0xF8, 0x00,
    /* "z" */
    0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0xCC, 0x18,
    0x30, 0x60, 0xC6, 0xFE, 0x00, 0x00, 0x00, 0x00,
    /* "{" */
    0x00, 0x00, 0x0E, 0x18, 0x18, 0x18, 0x70, 0x18,
    0x18, 0x18, 0x18, 0x0E, 0x00, 0x00, 0x00, 0x00,
    /* "|" */
    0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00,
    /* "}" */
    0x00, 0x00, 0x70, 0x18, 0x18, 0x18, 0x0E, 0x18,
    0x18, 0x18, 0x18, 0x70, 0x00, 0x00, 0x00, 0x00,
    /* "~" */
    0x00, 0x76, 0xDC, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  int  is_in_ascii_d0816_block(const TCHAR *code)
{
    if ( *code < 0x20 )
        return  -1;

    if ( *code > 0x7E )
        return  -1;

    return  1;
}

static  unsigned int  get_ascii_d0816_start_index(const TCHAR *code)
{
    unsigned int  index = (*code)*16 - 512;

    return  index;
}

const  MONO_CHARSET_FONT  gasc08 =
{
    /* .name                     = */ "ASC0816",
    /* .id                       = */ 10,
    /* .is_in_this_charset_block = */ is_in_ascii_d0816_block,
    /* .get_data_start_index     = */ get_ascii_d0816_start_index,
    /* .width                    = */ 8,
    /* .height                   = */ 16,
    /* .data                     = */ (UCHAR *)ascii_d0816_data,
    /* .next                     = */ 0
};
```

依据 gasc08 示例，用户可以定义所需的 MONO_CHARSET_FONT 子字库。

9.4 MONO_DISCRETE_FONT 单一离散子字库

系统定义 MONO_DISCRETE_FONT 结构体来标示单一离散子字库。MONO_DISCRETE_FONT 结构体定义如下：

```
struct _MONO_DISCRETE_FONT
{
    UCHAR                       name[FONT_NAME_LEN];
    UCHAR                       id;
    UCHAR                       width;
    UCHAR                       height;
    UINT                        counter;
    UCHAR                      *mono_char;
    struct _MONO_DISCRETE_FONT  *next;
};
typedef  struct _MONO_DISCRETE_FONT  MONO_DISCRETE_FONT;
```

MONO_DISCRETE_FONT 结构成员描述如表 9-2 所示。

成员	描述
name[FONT_NAME_LEN]	子字库名称
id	子字库 ID
width	字符宽度
height	字符高度
counter	子字库所包含的字符数目
*mono_char	子字库数据指针(结构体指针)
*next	后续 MONO_DISCRETE_FONT 指针

表 9-2 MONO_DISCRETE_FONT 结构体成员表

其中，*mono_char 是一个复杂的结构体指针。

对于单字节、多字节体系来说，用户需要声明如下宏：

DECLARE_MONO_DISCRETE_UCHAR2(n)

声明该宏后，用户可以使用 MONO_DISCRETE_CHAR_n 数据结构体类型。其中 n 表示每个 MONO_DISCRETE 字符所占据的字节数目。MONO_DISCRETE_CHAR_n 结构体定义如下：

```
struct _MONO_DISCRETE_CHAR_n
{
    UCHAR  code[2];
    UCHAR  data[n];
};
typedef   struct _MONO_DISCRETE_CHAR_n  MONO_DISCRETE_CHAR_n
```

例如：对于 16*16 的汉字，n 为 32，用户声明的 MONO_DISCRETE_CHAR_32 结构体类型如下：

```
struct _MONO_DISCRETE_CHAR_32
{
    UCHAR  code[2];
    UCHAR  data[32];
};
typedef   struct _MONO_DISCRETE_CHAR_32  MONO_DISCRETE_CHAR_32
```

对于 8*16 的 ASCII 码字符，n 为 16，用户声明的 MONO_DISCRETE_CHAR_16 结构体类型如下：

```
struct _MONO_DISCRETE_CHAR_16
{
    UCHAR  code[2];
    UCHAR  data[16];
};
typedef   struct _MONO_DISCRETE_CHAR_16  MONO_DISCRETE_CHAR_16
```

例如，16*16 汉字 MONO_DISCRETE_FONT 子字库 ghzu16 定义如下：

```
DECLARE_MONO_DISCRETE_UCHAR2(32);

static  const  MONO_DISCRETE_CHAR_32  cn1616[] =
{
    {
        "中",
        {
            0x01,0x00,0x01,0x00,0x21,0x08,0x3F,0xFC,0x21,0x08,0x21,0x08,0x21,0x08,0x21,0x08,
            0x21,0x08,0x3F,0xF8,0x21,0x08,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00
        }
    },
    {
        "国",
        {
            0x00,0x00,0x7F,0xFC,0x40,0x04,0x5F,0xF4,0x41,0x04,0x41,0x04,0x41,0x04,0x4F,0xE4,
            0x41,0x44,0x41,0x24,0x41,0x24,0x5F,0xF4,0x40,0x04,0x40,0x04,0x7F,0xFC,0x40,0x04
        }
    },
    {
        "珠",
```

```
        {
            0x00,0x20,0xF9,0x20,0x21,0x28,0x21,0xFC,0x21,0x20,0x22,0x20,0xF8,0x20,0x27,0xFE,
            0x20,0x70,0x20,0xA8,0x38,0xA8,0xC1,0x24,0x02,0x26,0x04,0x24,0x08,0x20,0x00,0x20
        }
    },
    {
        "海",
        {
            0x21,0x00,0x11,0x00,0x11,0xFE,0x02,0x00,0x97,0xF8,0x52,0x88,0x52,0x48,0x12,0x08,
            0x2F,0xFE,0x22,0x88,0xE2,0x48,0x22,0x08,0x23,0xFE,0x20,0x08,0x20,0x28,0x20,0x10
        }
    },
};

const  MONO_DISCRETE_FONT  ghzu16 = {
    /* .name      = */ "CN1616",
    /* .id        = */ 1,
    /* .width     = */ 16,
    /* .height    = */ 16,
    /* .counter   = */ sizeof(cn1616)/sizeof(MONO_DISCRETE_CHAR_32),
    /* .mono_char = */(void *)cn1616,
    /* .next      = */ 0
};
```

依据 ghzu16 示例，用户可以定义所需的 MONO_DISCRETE_FONT 子字库。

9.5 MIXED_CHARSET_FONT 混合字符集子字库

系统定义 MIXED_CHARSET_FONT 结构体来标示混合字符集子字库。MIXED_CHARSET_FONT 结构体定义如下：

```
struct _MIXED_CHARSET_FONT
{
    UCHAR                       name[FONT_NAME_LEN];
    UCHAR                       id;
    int                         (*is_in_this_charset_block)(const TCHAR *code);
    unsigned int                (*get_code_index)(const TCHAR *code);
    MIXED_CHARSET_CHAR          *mixed_char;
    struct _MIXED_CHARSET_FONT  *next;
};
typedef   struct _MIXED_CHARSET_FONT  MIXED_CHARSET_FONT;
```

其中，MIXED_CHARSET_CHAR 结构体定义如下：

```
struct _MIXED_CHARSET_CHAR
{
    UCHAR  width;
    UCHAR  height;
    UCHAR  *data;
};
typedef   struct _MIXED_CHARSET_CHAR  MIXED_CHARSET_CHAR;
```

MIXED_CHARSET_CHAR 结构成员描述如表 9-3 所示。

成员	描述
width	字符宽度
height	字符高度
*data	字符数据指针

表 9-3 MIXED_CHARSET_CHAR 结构体成员表

MIXED_CHARSET_FONT 结构成员描述如表 9-4 所示。

成员	描述
name[FONT_NAME_LEN]	子字库名称
id	子字库 ID

int (*is_in_this_charset_block)(const TCHAR *code)	判断字符 code 是否属于本子字库函数指针
unsigned int (*get_code_index)(const TCHAR *code)	获得字符 code 索引号函数指针
*mixed_char	MIXED_CHARSET_CHAR 数据指针
*next	后续 MIXED_CHARSET_FONT 指针

表 9-4 MIXED_CHARSET_FONT 结构体成员表

例如，ASCII 码 MIXED_CHARSET_FONT 子字库 gascmi 定义如下：

```
static  const  UCHAR  ASCII_0x20[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

/* 6*12 */
static  const  UCHAR  ASCII_0x21[] =
{
    0x00, 0x41, 0x04, 0x10, 0x40, 0x00, 0x10, 0x00, 0x00,
};

static  const  UCHAR  ASCII_0x22[] =
{
    0x00, 0x66, 0x66, 0x66, 0x44, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x23[] =
{
    0x00, 0x00, 0x00, 0x6C, 0x6C, 0xFE, 0x6C, 0x6C,
    0x6C, 0xFE, 0x6C, 0x6C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x24[] =
{
    0x18, 0x18, 0x7C, 0xC6, 0xC2, 0xC0, 0x7C, 0x06,
    0x06, 0x86, 0xC6, 0x7C, 0x18, 0x18, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x25[] =
{
    0x00, 0x00, 0x00, 0x00, 0xC2, 0xC6, 0x0C, 0x18,
    0x30, 0x60, 0xC6, 0x86, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x26[] =
{
    0x00, 0x00, 0x38, 0x6C, 0x6C, 0x38, 0x76, 0xDC,
    0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x27[] =
{
    0x00, 0x30, 0x30, 0x30, 0x60, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x28[] =
{
    0x00, 0x00, 0x0C, 0x18, 0x30, 0x30, 0x30, 0x30,
    0x30, 0x30, 0x18, 0x0C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x29[] =
{
    0x00, 0x00, 0x30, 0x18, 0x0C, 0x0C, 0x0C, 0x0C,
    0x0C, 0x0C, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00
```

```
};

static  const  UCHAR  ASCII_0x2A[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x3C, 0xFF,
    0x3C, 0x66, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x2B[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x7E,
    0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x2C[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x18, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x2D[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x2E[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x2F[] =
{
    0x00, 0x00, 0x00, 0x00, 0x02, 0x06, 0x0C, 0x18,
    0x30, 0x60, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x30[] =
{
    0x00, 0x00, 0x38, 0x6C, 0xC6, 0xC6, 0xC6, 0xC6,
    0xC6, 0xC6, 0x6C, 0x38, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x31[] =
{
    0x00, 0x00, 0x18, 0x38, 0x78, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x7E, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x32[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0x06, 0x0C, 0x18, 0x30,
    0x60, 0xC0, 0xC6, 0xFE, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x33[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0x06, 0x06, 0x3C, 0x06,
    0x06, 0x06, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x34[] =
{
    0x00, 0x00, 0x0C, 0x1C, 0x3C, 0x6C, 0xCC, 0xFE,
    0x0C, 0x0C, 0x0C, 0x1E, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x35[] =
```

```
{
    0x00, 0x00, 0xFE, 0xC0, 0xC0, 0xC0, 0xFE, 0x06,
    0x06, 0x06, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x36[] =
{
    0x00, 0x00, 0x38, 0x60, 0xC0, 0xC0, 0xFE, 0xC6,
    0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x37[] =
{
    0x00, 0x00, 0xFE, 0xC6, 0x06, 0x06, 0x0C, 0x18,
    0x30, 0x30, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x38[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0x7C, 0xC6,
    0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x39[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0x7E, 0x06,
    0x06, 0x06, 0x0C, 0x78, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x3A[] =
{
    0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00,
    0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x3B[] =
{
    0x00, 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00,
    0x00, 0x18, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x3C[] =
{
    0x00, 0x00, 0x00, 0x06, 0x0C, 0x18, 0x30, 0x60,
    0x30, 0x18, 0x0C, 0x06, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x3D[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x7E, 0x00, 0x00,
    0x7E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x3E[] =
{
    0x00, 0x00, 0x00, 0xC0, 0x30, 0x18, 0x0c, 0x06,
    0x0C, 0x18, 0x30, 0x60, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x3F[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0xC6, 0x0C, 0x18, 0x18,
    0x18, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x40[] =
{
    0x00, 0x00, 0x00, 0x78, 0xC6, 0xC6, 0xDE, 0xDE,
    0xDE, 0xDC, 0xC0, 0x7C, 0x00, 0x00, 0x00, 0x00
```

```
};

static  const  UCHAR  ASCII_0x41[] =
{
    0x00, 0x00, 0x10, 0x38, 0x6C, 0xC6, 0xC6, 0xFE,
    0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x42[] =
{
    0x00, 0x00, 0xFC, 0x66, 0x66, 0x66, 0x7C, 0x66,
    0x66, 0x66, 0x66, 0xFC, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x43[] =
{
    0x00, 0x00, 0x7C, 0x66, 0xC2, 0xC0, 0xC0, 0xC0,
    0xC0, 0xC2, 0x66, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x44[] =
{
    0x00, 0x00, 0xF8, 0x6C, 0x66, 0x66, 0x66, 0x66,
    0x66, 0x66, 0x6C, 0xF8, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x45[] =
{
    0x00, 0x00, 0xFE, 0x66, 0x62, 0x68, 0x78, 0x68,
    0x60, 0x62, 0x66, 0xFE, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x46[] =
{
    0x00, 0x00, 0xFE, 0x66, 0x62, 0x68, 0x78, 0x68,
    0x60, 0x60, 0x66, 0xF0, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x47[] =
{
    0x00, 0x00, 0x3C, 0x66, 0xC2, 0xC0, 0xC0, 0xDE,
    0xC6, 0xC6, 0x66, 0x3A, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x48[] =
{
    0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xFE, 0xC6,
    0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x49[] =
{
    0x00, 0x00, 0x3C, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x4A[] =
{
    0x00, 0x00, 0x1E, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C,
    0xCC, 0xCC, 0xCC, 0x78, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x4B[] =
{
    0x00, 0x00, 0xE6, 0x66, 0x66, 0x66, 0x78, 0x78,
    0x6C, 0x66, 0x66, 0xE6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x4C[] =
```

```
{
    0x00, 0x00, 0xF0, 0x60, 0x60, 0x60, 0x60, 0x60,
    0x60, 0x62, 0x66, 0xFE, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x4D[] =
{
    0x00, 0x00, 0xC6, 0xEE, 0xFE, 0xEF, 0xD6, 0xC6,
    0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x4E[] =
{
    0x00, 0x00, 0xC6, 0xE6, 0xF6, 0xFE, 0xDE, 0xCE,
    0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x4F[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
    0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x50[] =
{
    0x00, 0x00, 0xFC, 0x66, 0x66, 0x66, 0x7C, 0x60,
    0x60, 0x60, 0x60, 0xF0, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x51[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
    0xC6, 0xD6, 0xDE, 0x7C, 0x0C, 0x0E, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x52[] =
{
    0x00, 0x00, 0xFC, 0x66, 0x66, 0x66, 0x7C, 0x6C,
    0x66, 0x66, 0x66, 0xEC, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x53[] =
{
    0x00, 0x00, 0x7C, 0xC6, 0xC6, 0x60, 0x38, 0x0C,
    0x06, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x54[] =
{
    0x00, 0x00, 0x7E, 0x7E, 0x5A, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x55[] =
{
    0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
    0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00

};

static  const  UCHAR  ASCII_0x56[] =
{
    0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6, 0xC6,
    0xC6, 0x6C, 0x38, 0x10, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x57[] =
{
    0x00, 0x00, 0xC6, 0xC6, 0xC6, 0xC6, 0xD6, 0xD6,
```

```
    0xD6, 0xFE, 0xEE, 0x6C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x58[] =
{
    0x00, 0x00, 0xC6, 0xC6, 0x6C, 0x7C, 0x38, 0x38,
    0x7C, 0x6C, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x59[] =
{
    0x00, 0x00, 0x66, 0x66, 0x66, 0x66, 0x3C, 0x18,
    0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x5A[] =
{
    0x00, 0x00, 0xFE, 0xC6, 0x86, 0x0C, 0x18, 0x30,
    0x60, 0xC2, 0xC6, 0xFE, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x5B[] =
{
    0x00, 0x00, 0x3C, 0x30, 0x30, 0x30, 0x30, 0x30,
    0x30, 0x30, 0x30, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x5C[] =
{
    0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0x70, 0x38,
    0x1C, 0x0E, 0x06, 0x02, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x5D[] =
{
    0x00, 0x00, 0x38, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C,
    0x0C, 0x0C, 0x0C, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x5E[] =
{
    0x10, 0x38, 0x6C, 0xC6, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x5F[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x60[] =
{
    0x00, 0x30, 0x18, 0x0C, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x61[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x0C, 0x7C,
    0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x62[] =
{
    0x00, 0x00, 0xE0, 0x60, 0x60, 0x78, 0x6C, 0x66,
    0x66, 0x66, 0x66, 0x7C, 0x00, 0x00, 0x00, 0x00
};
```

```
static  const  UCHAR  ASCII_0x63[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0xC0,
    0xC0, 0xC0, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x64[] =
{
    0x00, 0x00, 0x1C, 0x0C, 0x0C, 0x3C, 0x6C, 0xCC,
    0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x65[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0xFE,
    0xC0, 0xC0, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x66[] =
{
    0x00, 0x00, 0x1C, 0x36, 0x32, 0x30, 0x78, 0x30,
    0x30, 0x30, 0x30, 0x78, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x67[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xCC, 0xCC,
    0xCC, 0xCC, 0xCC, 0x7C, 0x0C, 0xCC, 0x78, 0x00
};

static  const  UCHAR  ASCII_0x68[] =
{
    0x00, 0x00, 0xE0, 0x60, 0x60, 0x6C, 0x76, 0x66,
    0x66, 0x66, 0x66, 0xE6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x69[] =
{
    0x00, 0x00, 0x18, 0x18, 0x00, 0x38, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x6A[] =
{
    0x00, 0x00, 0x06, 0x06, 0x00, 0x0E, 0x06, 0x06,
    0x06, 0x06, 0x06, 0x06, 0x66, 0x66, 0x00, 0x3C
};

static  const  UCHAR  ASCII_0x6B[] =
{
    0x00, 0x00, 0xE0, 0x60, 0x60, 0x66, 0x6C, 0x78,
    0x78, 0x6C, 0x66, 0xE6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x6C[] =
{
    0x00, 0x00, 0x38, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x6D[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xEC, 0xFE, 0xD6,
    0xD6, 0xD6, 0xD6, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x6E[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xDC, 0x66, 0x66,
```

```
    0x66, 0x66, 0x66, 0x66, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x6F[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0xC6,
    0xC6, 0xC6, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x70[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xDC, 0x66, 0x66,
    0x66, 0x66, 0x66, 0x7C, 0x60, 0x60, 0xF0, 0x00
};

static  const  UCHAR  ASCII_0x71[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x76, 0xCC, 0xCC,
    0xCC, 0xCC, 0xCC, 0x7C, 0x0C, 0x0C, 0x1E, 0x00
};

static  const  UCHAR  ASCII_0x72[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xDC, 0x76, 0x66,
    0x60, 0x60, 0x60, 0xF0, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x73[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x7C, 0xC6, 0x60,
    0x70, 0x0C, 0xC6, 0x7C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x74[] =
{
    0x00, 0x00, 0x10, 0x30, 0x30, 0xFC, 0x30, 0x30,
    0x30, 0x30, 0x36, 0x1C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x75[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xCC, 0xCC, 0xCC,
    0xCC, 0xCC, 0xCC, 0x76, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x76[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0xC6, 0xC6,
    0xC6, 0xC6, 0x6C, 0x38, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x77[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0xC6, 0xD6,
    0xD6, 0xD6, 0xFE, 0x6C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x78[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0x6C, 0x38,
    0x38, 0x38, 0x6C, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x79[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xC6, 0xC6, 0xC6,
    0xC6, 0xC6, 0xC6, 0x7E, 0x06, 0x0C, 0xF8, 0x00
};
```

```
static  const  UCHAR  ASCII_0x7A[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0xCC, 0x18,
    0x30, 0x60, 0xC6, 0xFE, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x7B[] =
{
    0x00, 0x00, 0x0E, 0x18, 0x18, 0x18, 0x70, 0x18,
    0x18, 0x18, 0x18, 0x0E, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x7C[] =
{
    0x00, 0x00, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,
    0x18, 0x18, 0x18, 0x18, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x7D[] =
{
    0x00, 0x00, 0x70, 0x18, 0x18, 0x18, 0x0E, 0x18,
    0x18, 0x18, 0x18, 0x70, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_0x7E[] =
{
    0x00, 0x76, 0xDC, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};


static  const  MIXED_CHARSET_CHAR  mixed_ascii_charset[ ] =
{
    { 8, 16, (UCHAR *)ASCII_0x20 },
    { 6, 12, (UCHAR *)ASCII_0x21 },  /* 6*12 */
    { 8, 16, (UCHAR *)ASCII_0x22 },
    { 8, 16, (UCHAR *)ASCII_0x23 },
    { 8, 16, (UCHAR *)ASCII_0x24 },
    { 8, 16, (UCHAR *)ASCII_0x25 },
    { 8, 16, (UCHAR *)ASCII_0x26 },
    { 8, 16, (UCHAR *)ASCII_0x27 },
    { 8, 16, (UCHAR *)ASCII_0x28 },
    { 8, 16, (UCHAR *)ASCII_0x29 },
    { 8, 16, (UCHAR *)ASCII_0x2A },
    { 8, 16, (UCHAR *)ASCII_0x2B },
    { 8, 16, (UCHAR *)ASCII_0x2C },
    { 8, 16, (UCHAR *)ASCII_0x2D },
    { 8, 16, (UCHAR *)ASCII_0x2E },
    { 8, 16, (UCHAR *)ASCII_0x2F },
    { 8, 16, (UCHAR *)ASCII_0x30 },
    { 8, 16, (UCHAR *)ASCII_0x31 },
    { 8, 16, (UCHAR *)ASCII_0x32 },
    { 8, 16, (UCHAR *)ASCII_0x33 },
    { 8, 16, (UCHAR *)ASCII_0x34 },
    { 8, 16, (UCHAR *)ASCII_0x35 },
    { 8, 16, (UCHAR *)ASCII_0x36 },
    { 8, 16, (UCHAR *)ASCII_0x37 },
    { 8, 16, (UCHAR *)ASCII_0x38 },
    { 8, 16, (UCHAR *)ASCII_0x39 },
    { 8, 16, (UCHAR *)ASCII_0x3A },
    { 8, 16, (UCHAR *)ASCII_0x3B },
    { 8, 16, (UCHAR *)ASCII_0x3C },
    { 8, 16, (UCHAR *)ASCII_0x3D },
    { 8, 16, (UCHAR *)ASCII_0x3E },
    { 8, 16, (UCHAR *)ASCII_0x3F },
    { 8, 16, (UCHAR *)ASCII_0x40 },
    { 8, 16, (UCHAR *)ASCII_0x41 },
    { 8, 16, (UCHAR *)ASCII_0x42 },
    { 8, 16, (UCHAR *)ASCII_0x43 },
```

```
    { 8, 16, (UCHAR *)ASCII_0x44 },
    { 8, 16, (UCHAR *)ASCII_0x45 },
    { 8, 16, (UCHAR *)ASCII_0x46 },
    { 8, 16, (UCHAR *)ASCII_0x47 },
    { 8, 16, (UCHAR *)ASCII_0x48 },
    { 8, 16, (UCHAR *)ASCII_0x49 },
    { 8, 16, (UCHAR *)ASCII_0x4A },
    { 8, 16, (UCHAR *)ASCII_0x4B },
    { 8, 16, (UCHAR *)ASCII_0x4C },
    { 8, 16, (UCHAR *)ASCII_0x4D },
    { 8, 16, (UCHAR *)ASCII_0x4E },
    { 8, 16, (UCHAR *)ASCII_0x4F },
    { 8, 16, (UCHAR *)ASCII_0x50 },
    { 8, 16, (UCHAR *)ASCII_0x51 },
    { 8, 16, (UCHAR *)ASCII_0x52 },
    { 8, 16, (UCHAR *)ASCII_0x53 },
    { 8, 16, (UCHAR *)ASCII_0x54 },
    { 8, 16, (UCHAR *)ASCII_0x55 },
    { 8, 16, (UCHAR *)ASCII_0x56 },
    { 8, 16, (UCHAR *)ASCII_0x57 },
    { 8, 16, (UCHAR *)ASCII_0x58 },
    { 8, 16, (UCHAR *)ASCII_0x59 },
    { 8, 16, (UCHAR *)ASCII_0x5A },
    { 8, 16, (UCHAR *)ASCII_0x5B },
    { 8, 16, (UCHAR *)ASCII_0x5C },
    { 8, 16, (UCHAR *)ASCII_0x5D },
    { 8, 16, (UCHAR *)ASCII_0x5E },
    { 8, 16, (UCHAR *)ASCII_0x5F },
    { 8, 16, (UCHAR *)ASCII_0x60 },
    { 8, 16, (UCHAR *)ASCII_0x61 },
    { 8, 16, (UCHAR *)ASCII_0x62 },
    { 8, 16, (UCHAR *)ASCII_0x63 },
    { 8, 16, (UCHAR *)ASCII_0x64 },
    { 8, 16, (UCHAR *)ASCII_0x65 },
    { 8, 16, (UCHAR *)ASCII_0x66 },
    { 8, 16, (UCHAR *)ASCII_0x67 },
    { 8, 16, (UCHAR *)ASCII_0x68 },
    { 8, 16, (UCHAR *)ASCII_0x69 },
    { 8, 16, (UCHAR *)ASCII_0x6A },
    { 8, 16, (UCHAR *)ASCII_0x6B },
    { 8, 16, (UCHAR *)ASCII_0x6C },
    { 8, 16, (UCHAR *)ASCII_0x6D },
    { 8, 16, (UCHAR *)ASCII_0x6E },
    { 8, 16, (UCHAR *)ASCII_0x6F },
    { 8, 16, (UCHAR *)ASCII_0x70 },
    { 8, 16, (UCHAR *)ASCII_0x71 },
    { 8, 16, (UCHAR *)ASCII_0x72 },
    { 8, 16, (UCHAR *)ASCII_0x73 },
    { 8, 16, (UCHAR *)ASCII_0x74 },
    { 8, 16, (UCHAR *)ASCII_0x75 },
    { 8, 16, (UCHAR *)ASCII_0x76 },
    { 8, 16, (UCHAR *)ASCII_0x77 },
    { 8, 16, (UCHAR *)ASCII_0x78 },
    { 8, 16, (UCHAR *)ASCII_0x79 },
    { 8, 16, (UCHAR *)ASCII_0x7A },
    { 8, 16, (UCHAR *)ASCII_0x7B },
    { 8, 16, (UCHAR *)ASCII_0x7C },
    { 8, 16, (UCHAR *)ASCII_0x7D },
    { 8, 16, (UCHAR *)ASCII_0x7E }
};

static  int  is_in_this_block(const TCHAR *code)
{
    if ( *code < 0x20 )
        return  -1;

    if ( *code > 0x7E )
        return  -1;
```

```
    return  1;
}

static  unsigned int  get_mixed_ascii_offset(const TCHAR *code)
{
    unsigned int  offset = 0;

    if ( *code < 0x20 )
        return  0;

    if ( *code > 0x7E )
        return  0;

    offset = *code - 0x20;

    return  offset;
}


GUI_DATA_CONST  MIXED_CHARSET_FONT  gascmi =
{
    /* .name                     = */ "MYASC",
    /* .id                       = */ 15,
    /* .is_in_this_charset_block = */ is_in_this_block,
    /* .get_code_index           = */ get_mixed_ascii_offset,
    /* .mixed_char               = */ (MIXED_CHARSET_CHAR *)mixed_ascii_charset,
    /* .next                     = */ 0
};
```

依据 gascmi 示例，用户可以定义所需的 MIXED_CHARSET_FONT 子字库。

9.6 MIXED_DISCRETE_FONT 混合离散子字库

系统定义 MIXED_DISCRETE_FONT 结构体来标示混合离散子字库。MIXED_DISCRETE_FONT 结构体定义如下：

```
struct _MIXED_DISCRETE_FONT
{
    UCHAR                        name[FONT_NAME_LEN];
    UCHAR                        id;
    UCHAR                        type;
    UINT                         counter;
    void                        *list;
    struct _MIXED_DISCRETE_FONT  *next;
};
typedef  struct _MIXED_DISCRETE_FONT  MIXED_DISCRETE_FONT;
```

其中，系统针对单字节、多字节体系定义了 MIXED_DISCRETE_UCHAR2 结构体；针对 Unicode 体系定义了 MIXED_DISCRETE_UINT16 结构体，二者定义如下：

```
struct _MIXED_DISCRETE_UCHAR2
{
    UCHAR   code[2];
    UCHAR   width;
    UCHAR   height;
    UCHAR   *data;
};
typedef  struct _MIXED_DISCRETE_UCHAR2  MIXED_DISCRETE_UCHAR2;

struct _MIXED_DISCRETE_UINT16
{
    UINT16  code;
    UCHAR   width;
    UCHAR   height;
    UCHAR   *data;
};
typedef  struct _MIXED_DISCRETE_UINT16  MIXED_DISCRETE_UINT16;
```

MIXED_DISCRETE_UCHAR2 结构成员描述如表 9-5 所示。

成员	描述
code[2]	字符编码
width	字符宽度
height	字符高度
*data	字符数据指针

表 9-5 MIXED_DISCRETE_UCHAR2 结构体成员表

MIXED_DISCRETE_UINT16 结构成员描述如表 9-6 所示。

成员	描述
code	字符编码
Width	字符宽度
Height	字符高度
*data	字符数据指针

表 9-6 MIXED_DISCRETE_UINT16 结构体成员表

MIXED_DISCRETE_FONT 结构成员描述如表 9-7 所示。

成员	描述
name[FONT_NAME_LEN]	子字库名称
id	子字库 ID
type	字符类型，取值如下： MIXED_DISCRETE_UCHAR2_TYPE：表示单字节、多字节字符体系 MIXED_DISCRETE_UINT16_TYPE：表示 Unicode 字符体系
counter	子字库所包含的字符数目
*list	MIXED_DISCRETE_UCHAR2 或者 MIXED_DISCRETE_UINT16 结构指针
*next	后续 MIXED_DISCRETE_FONT 指针

表 9-7 MIXED_DISCRETE_FONT 结构体成员表

例如，MIXED_DISCRETE_FONT 子字库 gmixdi 定义如下：

```
static  const  UCHAR  ASCII_BLANK[]  =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_A[]  =
{
    0x00, 0x00, 0x10, 0x38, 0x6C, 0xC6, 0xC6, 0xFE,
    0xC6, 0xC6, 0xC6, 0xC6, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_B[]  =
{
    0x00, 0x00, 0xFC, 0x66, 0x66, 0x66, 0x7C, 0x66,
    0x66, 0x66, 0x66, 0xFC, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_C[]  =
{
    0x00, 0x00, 0x7C, 0x66, 0xC2, 0xC0, 0xC0, 0xC0,
    0xC0, 0xC2, 0x66, 0x3C, 0x00, 0x00, 0x00, 0x00
};

static  const  UCHAR  ASCII_D[]  =
{
    0x00, 0x00, 0xF8, 0x6C, 0x66, 0x66, 0x66, 0x66,
```

```
    0x66, 0x66, 0x6C, 0xF8, 0x00, 0x00, 0x00, 0x00
};

static const UCHAR HANZI_BI[] =
{
    0x20,0x40,0x20,0x40,0x3E,0xFE,0x51,0x20,0x8A,0x10,0x01,0xF8,0x7E,0x00,0x02,0x00,
    0x03,0xF0,0x7E,0x00,0x02,0x00,0x03,0xFE,0xFE,0x00,0x02,0x04,0x02,0x04,0x01,0xFC
};

static const UCHAR HANZI_JI[] =
{
    0x40,0x00,0x21,0xFC,0x30,0x04,0x20,0x04,0x00,0x04,0x00,0x04,0xF1,0xFC,0x11,0x00,
    0x11,0x00,0x11,0x00,0x11,0x00,0x11,0x04,0x15,0x04,0x19,0x06,0x10,0xFC,0x00,0x00
};

static const UCHAR HANZI_BEN[] =
{
    0x01,0x00,0x01,0x00,0x01,0x00,0xFF,0xFE,0x03,0x80,0x03,0x40,0x05,0x40,0x05,0x20,
    0x09,0x10,0x11,0x18,0x2F,0xEE,0xC1,0x04,0x01,0x00,0x01,0x00,0x01,0x00,0x00,0x00
};

static const UCHAR HANZI_DIAN[] =
{
    0x01,0x00,0x01,0x00,0x01,0x00,0x3F,0xF8,0x21,0x08,0x21,0x08,0x3F,0xF8,0x21,0x08,
    0x21,0x08,0x21,0x08,0x3F,0xF8,0x21,0x08,0x01,0x02,0x01,0x02,0x00,0xFE,0x00,0x00
};

static const UCHAR HANZI_NAO[] =
{
    0x00,0x80,0x78,0x40,0x48,0x20,0x4B,0xFE,0x78,0x00,0x48,0x10,0x4A,0x94,0x4A,0x64,
    0x7A,0x64,0x4A,0x64,0x4A,0x94,0x4A,0x94,0x4B,0x04,0xAB,0xFC,0x92,0x04,0x00,0x00
};

static const UCHAR HANZI_HU[] =
{
    0x00,0x00,0x7F,0xFC,0x04,0x00,0x04,0x00,0x04,0x20,0x07,0xF0,0x08,0x20,0x08,0x20,
    0x08,0x20,0x08,0x20,0x1F,0xE0,0x08,0x20,0x00,0x20,0x7F,0xFE,0x00,0x00,0x00,0x00
};

static const UCHAR HANZI_LIAN[] =
{
    0x01,0x08,0xFE,0x8C,0x44,0x48,0x44,0x50,0x7F,0xFE,0x44,0x20,0x44,0x20,0x7C,0x20,
    0x47,0xFE,0x44,0x20,0x4E,0x20,0xF4,0x20,0x44,0x50,0x04,0x48,0x04,0x86,0x05,0x04
};

static const UCHAR HANZI_WANG[] =
{
    0x00,0x00,0x7F,0xFC,0x40,0x04,0x41,0x04,0x51,0x14,0x4A,0x9C,0x44,0x54,0x44,0x24,
    0x4A,0x24,0x4A,0x54,0x51,0x94,0x61,0x04,0x40,0x04,0x40,0x1C,0x40,0x08,0x00,0x00
};

static const UCHAR HANZI_FEI[] =
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3F,0xFC,0x00,0x00,0x08,0x20,0x00,0x08,0x70,
    0x00,0x08,0xC0,0x00,0x08,0x80,0x00,0x09,0x00,0x00,0x0A,0x00,0x00,0x0E,0x00,0x00,0x09,0x80,
    0x00,0x08,0xE0,0x00,0x08,0x60,0x00,0x08,0x20,0x00,0x0C,0x00,0x00,0x04,0x00,0x00,0x06,0x08,
    0x00,0x06,0x08,0x00,0x03,0x08,0x00,0x01,0xC8,0x00,0x00,0x7C,0x00,0x00,0x1C,0x00,0x00,0x00
};

static const UCHAR HANZI_LIU[] =
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0E,0x00,0x08,0x02,0x08,0x0C,0x02,0x1C,0x06,0x7D,0xE0,
    0x04,0x0C,0x00,0x01,0x08,0x40,0x41,0x10,0x20,0x31,0x20,0x30,0x1A,0x7F,0xD0,0x12,0x00,0x10,
    0x02,0x66,0x60,0x04,0x64,0x40,0x04,0x64,0x40,0x0C,0x64,0x40,0x08,0x64,0x40,0x38,0x44,0x44,
    0x18,0x44,0x44,0x18,0xC4,0x44,0x18,0x84,0x44,0x19,0x04,0x7E,0x0E,0x04,0x00,0x00,0x00,0x00
};

static const UCHAR HANZI_ZHI[] =
{
```

```
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x00,0x00,0x10,0x08,0x00,0x10,0x18,0x1F,0xFF,0xE0,
    0x00,0x10,0x00,0x02,0x30,0x60,0x03,0xCF,0xE0,0x02,0x00,0x40,0x02,0x00,0x40,0x03,0xFF,0xC0,
    0x02,0x00,0x40,0x02,0x00,0x40,0x02,0x00,0x40,0x03,0xFF,0xC0,0x02,0x00,0x40,0x02,0x00,0x40,
    0x03,0xFF,0xC0,0x02,0x00,0x40,0x02,0x00,0x40,0x02,0x00,0x4C,0x7D,0xFF,0xB0,0x00,0x00,0x00
};

static  const  UCHAR  HANZI_XIA[]  =
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00,0x1C,0x3F,0xFF,0xE0,0x00,0x10,0x00,
    0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x18,0x00,0x00,0x17,0x80,0x00,0x11,0xE0,
    0x00,0x10,0x60,0x00,0x10,0x20,0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x10,0x00,
    0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x10,0x00,0x00,0x00,0x00
};

static  const  UCHAR  HANZI_SAN[]  =
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x1F,0xFF,0xE0,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,
    0x0F,0xFF,0xE0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x08,0x7F,0xFF,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};

static  const  UCHAR  HANZI_QIAN[]  =
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x07,0xE0,0x01,0xF8,0x00,0x06,0x18,0x00,
    0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x0C,0x3F,0xFF,0xF0,
    0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,
    0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x18,0x00,0x00,0x00,0x00
};

static  const  UCHAR  HANZI_CHI[]  =
{
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x30,0x03,0xFF,0xE0,0x02,0x00,0x20,
    0x02,0x00,0x20,0x02,0x00,0x20,0x02,0x00,0x20,0x03,0xFF,0xE0,0x02,0x08,0x20,0x02,0x08,0x00,
    0x02,0x08,0x00,0x02,0x04,0x00,0x02,0x04,0x00,0x02,0x06,0x00,0x06,0x02,0x00,0x04,0x01,0x00,
    0x04,0x01,0x80,0x08,0x00,0xE0,0x10,0x00,0x78,0x10,0x00,0x1C,0x60,0x00,0x08,0x00,0x00,0x00
};

static  const  MIXED_DISCRETE_UCHAR2  mixed_uchar2_char[] =
{
    {
        " ",  4,  16,  (UCHAR *)(&ASCII_BLANK)
    },
    {
        "A",  8,  16,  (UCHAR *)(&ASCII_A)
    },
    {
        "B",  8,  16,  (UCHAR *)(&ASCII_B)
    },
    {
        "C",  8,  16,  (UCHAR *)(&ASCII_C)
    },
    {
        "D",  8,  16,  (UCHAR *)(&ASCII_D)
    },
    {
        "笔",16,16,(UCHAR *)(&HANZI_BI)
    },
    {
        "记",16,16,(UCHAR *)(&HANZI_JI)
    },
    {
        "本",16,16,(UCHAR *)(&HANZI_BEN)
    },
    {
        "电",16,16,(UCHAR *)(&HANZI_DIAN)
    },
    {
        "脑",16,16,(UCHAR *)(&HANZI_NAO)
```

```
    },
    {
        "互",16,16,(UCHAR *)(&HANZI_HU)
    },
    {
        "联",16,16,(UCHAR *)(&HANZI_LIAN)
    },
    {
        "网",16,16,(UCHAR *)(&HANZI_WANG)
    },
    {
        "飞",24,24,(UCHAR *)(&HANZI_FEI)
    },
    {
        "流",24,24,(UCHAR *)(&HANZI_LIU)
    },
    {
        "直",24,24,(UCHAR *)(&HANZI_ZHI)
    },
    {
        "下",24,24,(UCHAR *)(&HANZI_XIA)
    },
    {
        "三",24,24,(UCHAR *)(&HANZI_SAN)
    },
    {
        "千",24,24,(UCHAR *)(&HANZI_QIAN)
    },
    {
        "尺",24,24,(UCHAR *)(&HANZI_CHI)
    }
};

const  MIXED_DISCRETE_FONT  gmixdi =
{
    /* .name    = */ "MIXED",
    /* .id      = */ 9,
    /* .type    = */ MIXED_DISCRETE_UCHAR2_TYPE,
    /* .counter = */ sizeof(mixed_uchar2_char)/sizeof(MIXED_DISCRETE_UCHAR2),
    /* .list    = */ (void *)mixed_uchar2_char,
    /* .next    = */ 0
};
```

依据 gmixdi 示例，用户可以定义所需的 MIXED_DISCRETE_FONT 子字库。

9.7 系统内置子字库

系统内置了常见的子字库，如表 9-9 所示。

编码标准	子字库名称	支持宏	字符大小	字库类别	字体
ASCII	lasc06	_LG_ASCII_LATIN_D0612C_FONT_	6*12	MONO_CHARSET_FONT	
ASCII	lasc08	_LG_ASCII_LATIN_D0816C_FONT_	8*16	MONO_CHARSET_FONT	
GB2312-80	lb8012	_LG_GB2312_D1212CS_C1_FONT_ 符号宏： INCLUDE_GB2312_D1212CS_SYSBOL 二级字库宏： INCLUDE_GB2312_D1212CS_C2	12*12	MONO_CHARSET_FONT	宋体
GB2312-80	lb8016	_LG_GB2312_D1616CS_C1_FONT_ 符号宏： INCLUDE_GB2312_D1616CS_SYSBOL 二级字库宏： INCLUDE_GB2312_D1616CS_C2	16*16	MONO_CHARSET_FONT	宋体
Unicode	lcjk12	_LG_CJK_UNIFIED_D1212CS_FONT_	12*12	MONO_CHARSET_FONT	宋体
Unicode	lcjk16	_LG_CJK_UNIFIED_D1616CS_FONT_	16*16	MONO_CHARSET_FONT	宋体

表 9-9 系统内置子字库表

其中，对于 GB2312-80 标准汉字库，可以分别裁剪符号、二级字库。在资源较为紧张的系统中，用户可以应用一级字库和用户自定义小汉字库相结合处理。

9.8 GUI_FONT 系统字库定义

系统字库 GUI_FONT 结构体定义如下：

```
struct    _GUI_FONT
{
    FONT_TYPE          type;
    void              *font;
    struct _GUI_FONT  *next;
};
typedef   struct     _GUI_FONT  GUI_FONT;
```

GUI_FONT 结构成员描述如表 9-8 所示。

成员	描述
type	子字库类型，取值如下： MONO_CHARSET_FONT_TYPE MONO_DISCRETE_FONT_TYPE MIXED_CHARSET_FONT_TYPE MIXED_DISCRETE_FONT_TYPE
*font	子字库数据指针
*next	后续 GUI_FONT 指针

表 9-8 GUI_FONT 结构体成员表

例如，GUI_FONT 字库 appfnt 定义如下：

```
GUI_FONT  gb8016 =
{
    /* .type = */ MONO_CHARSET_FONT_TYPE,
    /* .font = */ (void *)(&lb8016),
    /* .next = */ 0
};

GUI_FONT  appfnt =
{
    /* .type = */ MONO_CHARSET_FONT_TYPE,
    /* .font = */ (void *)(&lasc08),
    /* .next = */ (GUI_FONT *)(& gb8016)
};
```

依据 appfnt 示例，用户可以定义所需的 GUI_FONT 系统字库。

9.9 GUI_FONT 系统字库使用

系统字库遵行先定义后使用的原则。在多数情况下需要将系统字库定义为全局变量。如何使用系统字库，参见第十章。

第十章　HDC 句柄及其操作接口

10.1　HDC 句柄概述

绘制操作需要限制在特定的矩形区域内进行，需要区分前景色和背景色、是否透明等属性。系统将这些属性封装在一个称之为设备上下文（device context）中，并且将其定义为 HDC 句柄。因此，绘制操作都依赖于 HDC 来完成。同时，系统提供操作 HDC 句柄的接口函数，严格禁止用户直接对 HDC 句柄赋值。

在 Basic 版本中，系统内置多个缺省的 basic HDC 句柄（MAX_DC_NUM 宏决定 HDC 数目），绘制操作由 basic HDC 句柄来完成。在 Windows 版本中，窗口绘制操作划分为 Window 区域绘制和客户 client 区域绘制：先是绘制 Window 区，由 window HDC 句柄来完成；之后绘制 client 区域，由 client HDC 句柄来完成。

无论是 basic HDC 句柄，还是 window HDC 句柄、client 句柄，都是 HDC 句柄，除了来源、获取方法和释放方法不同外，HDC 句柄其它的操作都是相同的。

在使用 HDC 句柄时，必须先获取 HDC 句柄，使用完 HDC 句柄后，必须释放 HDC 句柄。获取 HDC 句柄和释放 HDC 句柄必须成对出项。

10.2　HDC 属性及其相关数据结构

10.2.1　点坐标 GUI_POINT 结构体定义

GUI_POINT 结构体定义如下

```
struct  _GUI_POINT
{
    int  x;
    int  y;
};
typedef  struct _GUI_POINT   GUI_POINT;
```

GUI_POINT 结构体各个成员描述如表 10-1 所示。

结构体成员	成员描述
x	X 坐标
y	Y 坐标

表 10-1　GUI_POINT 结构体成员描述表

10.2.2　矩形绘制区域 GUI_RECT 结构体定义

GUI_RECT 结构体定义如下

```
struct  _GUI_RECT
{
    int  left;
    int  top;
    int  right;
    int  bottom;
};
typedef  struct _GUI_RECT   GUI_RECT;
```

GUI_RECT 结构体各个成员描述如表 10-2 所示。

结构体成员	成员描述
left	矩形左上角 X 坐标
top	矩形左上角 Y 坐标
right	矩形右下角 X 坐标
bottom	矩形右下角 Y 坐标

表 10-2　GUI_RECT 结构体成员描述表

GUI_RECT 的边界属于该区域。

操作 GUI_RECT 相关的宏定义如表 10-3 所示。

宏名称	宏描述
GUI_RECTW(GUI_RECT *rect)	获取*rect 指针区域宽度
GUI_RECTH(GUI_RECT *rect)	获取*rect 指针区域高度

表 10-3 操作 GUI_RECT 结构体成员宏表

10.2.3 文本输出模式

在进行文本输出时，如果同时绘制文本背景，那么是使用覆盖模式；如果不绘制文本背景，那么是采用透明模式。绘制文本模式定义如表 10-4 所示。

文本绘制模式名称	绘制模式标示
覆盖模式	MODE_COPY
透明模式	MODE_TRANSPARENCY

表 10-4 文本绘制模式定义表

10.2.4 文本输出修饰

在进行文本输出时，可以同时修饰输出格式。修饰文本输出格式 GUI_TEXT_METRIC 结构体定义如下：

```
struct  _GUI_TEXT_METRIC
{
    int    space_left;
    int    space_top;
    int    space_right;
    int    space_bottom;

    UCHAR  is_strike_out;
    UCHAR  frame_style;
    char   offset_italic;
    char   offset_escapement;
};
typedef  struct _GUI_TEXT_METRIC    GUI_TEXT_METRIC;
```

GUI_TEXT_METRIC 结构体各个成员描述如表 10-5 所示。

结构体成员	成员描述
space_left	字符左间隔像素
space_top	字符上间隔像素
space_right	字符右间隔像素
space_bottom	字符下间隔像素
is_strike_out	是否具备下划线
frame_style	字符边框式样： TEXT_FRAME_LEFT： 左边框 TEXT_FRAME_TOP： 上边框 TEXT_FRAME_RIGHT： 右边框 TEXT_FRAME_BOTTOM：下边框 字符边界式样可以位或操作
offset_italic	字符偏移像素宽度
offset_escapement	字符偏移像素高度

表 10-5 GUI_TEXT_METRIC 结构体成员描述表

10.3 HDC 属性操作接口函数

10.3.1 获取 Basic HDC：hdc_get_basic( )

函数原型：HDC hdc_get_basic(void)

返回 Basic HDC 句柄。

系统内置多个 HDC 数组，hdc_get_basic 函数获取一个未占用的 HDC。

10.3.2 释放 Basic HDC：hdc_release_basic( )

函数原型：int hdc_release_basic(HDC hdc)
函数参数：hdc 句柄：之前使用 hdc_get_basic 获取的 hdc 句柄。
返回 1。
hdc_get_basic()和 hdc_release_basic()必须成对使用。

10.3.3 获取 Window HDC：hdc_get_window( )

函数原型：HDC hdc_get_window(HWND hwnd)
返回 Window HDC 句柄。
HWND 是窗口句柄（参见第十一章节），该函数是 Windows 版本专用函数。

10.3.4 释放 Window HDC：hdc_release_window( )

函数原型：int hdc_release_window(HWND hwnd ,HDC hdc)
函数参数：
    hwnd 句柄：窗口句柄。
    hdc 句柄：之前使用 hdc_get_window 获取的 hdc 句柄。
返回 1。
hdc_get_window()和 hdc_release_window()必须成对使用。

10.3.5 获取 Client HDC：hdc_get_client( )

函数原型：HDC hdc_get_client(HWND hwnd)
返回 Client HDC 句柄。
该函数是 Windows 版本专用函数。

10.3.6 释放 Client HDC：hdc_release_client( )

函数原型：int hdc_release_client(HWND hwnd ,HDC hdc)
函数参数：
    hwnd 句柄：窗口句柄。
    hdc 句柄：之前使用 hdc_get_client 获取的 hdc 句柄。
返回 1。
hdc_get_client()和 hdc_release_client()必须成对使用。

10.3.7 获取当前颜色组：hdc_get_current_group( )

函数原型：unsigned int hdc_get_current_group(HDC hdc)
函数参数：hdc 句柄。
返回当前颜色组。

10.3.8 设置当前颜色组：hdc_set_current_group( )

函数原型：int hdc_set_current_group(HDC hdc, unsigned int group)
函数参数：hdc 句柄和 group 颜色组 ID
返回 1。

10.3.9 获取组颜色：hdc_get_group_color( )

函数原型：GUI_COLOR hdc_get_group_color(HDC hdc, unsigned int group, unsigned int role)
函数参数：hdc 句柄、group 颜色组 ID、role 颜色角色
返回逻辑颜色。

10.3.10 设置组颜色：hdc_set_group_color( )

函数原型：int hdc_set_group_color(HDC hdc, unsigned int group, unsigned int role, GUI_COLOR color)
函数参数：hdc 句柄、group 颜色组 ID、role 颜色角色，color 逻辑颜色
返回 1。

10.3.11 获取当前组颜色：hdc_get_current_group_color( )

函数原型：GUI_COLOR hdc_get_current_group_color(HDC hdc, unsigned int role)

函数参数：hdc 句柄、role 颜色角色
返回逻辑颜色。

10.3.12 设置当前组颜色：hdc_set_current_group_color( )

函数原型：int hdc_set_current_group_color(HDC hdc, unsigned int role, GUI_COLOR color)
函数参数：hdc 句柄、role 颜色角色，color 逻辑颜色
返回 1。

10.3.13 获取文本输出背景模式：hdc_get_back_mode( )

函数原型：int hdc_get_back_mode(HDC hdc)
函数参数：hdc 句柄
返回背景模式。

10.3.14 设置文本输出背景模式：hdc_set_back_mode( )

函数原型：int hdc_set_back_mode(HDC hdc, int mode)
函数参数：hdc 句柄、mode 背景模式
返回 1。

10.3.15 获取 HDC 区域：hdc_get_rect( )

函数原型：int hdc_get_rect(HDC hdc, GUI_RECT *rect)
函数参数：hdc 句柄、*rect 区域指针
返回 1。

10.3.16 设置 HDC 区域：hdc_set_rect( )

函数原型：int hdc_set_rect(HDC hdc, GUI_RECT *rect)
函数参数：hdc 句柄、*rect 区域指针
返回 1。

10.3.17 获取 HDC 字库：hdc_get_font( )

函数原型：int hdc_get_font(HDC hdc, GUI_FONT *font)
函数参数：hdc 句柄、*font 字库指针
返回 1 或者-1。

10.3.18 设置 HDC 字库：hdc_set_font( )

函数原型：int hdc_set_font(HDC hdc, GUI_FONT *font)
函数参数：hdc 句柄、*font 字库指针
返回 1。

10.3.19 获取 HDC 字库宽度：hdc_get_font_width( )

函数原型：int hdc_get_font_width(HDC hdc)
函数参数：hdc 句柄
返回 1 或者-1。

10.3.20 获取 HDC 字库高度：hdc_get_font_height( )

函数原型：int hdc_get_font_height(HDC hdc)
函数参数：hdc 句柄
返回 1 或者-1。

10.3.21 获取 HDC 文本控制格式：hdc_get_text_metric( )

函数原型：int hdc_get_text_metric(HDC hdc, GUI_TEXT_METRIC *tm)
函数参数：hdc 句柄、*tm 文本控制指针
返回 1 或者-1。

10.3.22 设置 HDC 文本控制格式：hdc_set_text_metric( )

函数原型：int hdc_set_text_metric(HDC hdc, GUI_TEXT_METRIC *tm)
函数参数：hdc 句柄、*tm 文本控制指针
返回 1 或者-1。

10.3.23　清屏：hdc_clear( )

函数原型：int hdc_clear(HDC hdc)
函数参数：hdc 句柄
返回 1 或者-1。

10.3.24　获取背景颜色：hdc_get_back_color( )

函数原型：GUI_COLOR hdc_get_back_color(HDC hdc)
函数参数：hdc 句柄
返回背景颜色。

10.3.25　设置背景颜色：hdc_set_back_color( )

函数原型：int hdc_set_back_color(HDC hdc, GUI_COLOR color)
函数参数：hdc 句柄、color 背景颜色
返回 1。

10.3.26　获取前景颜色：hdc_get_fore_color( )

函数原型：GUI_COLOR hdc_get_fore_color(HDC hdc)
函数参数：hdc 句柄
返回前景颜色。

10.3.27　设置前景颜色：hdc_set_fore_color( )

函数原型：int hdc_set_fore_color(HDC hdc, GUI_COLOR color)
函数参数：hdc 句柄、color 前景颜色
返回 1。

10.3.28　获取文本背景颜色：hdc_get_text_back_color( )

函数原型：GUI_COLOR hdc_get_text_back_color(HDC hdc)
函数参数：hdc 句柄
返回背景颜色。

10.3.29　设置文本背景颜色：hdc_set_text_back_color( )

函数原型：int hdc_set_text_back_color(HDC hdc, GUI_COLOR color)
函数参数：hdc 句柄、color 背景颜色
返回 1。

10.3.30　获取文本前景颜色：hdc_get_text_fore_color( )

函数原型：GUI_COLOR hdc_get_text_fore_color(HDC hdc)
函数参数：hdc 句柄
返回前景颜色。

10.3.31　设置文本前景颜色：hdc_set_text_fore_color( )

函数原型：int hdc_set_text_fore_color(HDC hdc, GUI_COLOR color)
函数参数：hdc 句柄、color 前景颜色
返回 1。

HDC 属性操作接口汇总如表 10-6 所示。

函数原型	函数简述
HDC hdc_get_basic(void)	获取 Basic HDC 句柄
int hdc_release_basic(HDC hdc)	释放 Basic HDC 句柄

HDC hdc_get_window(HWND hwnd)	获取 Window HDC 句柄
int hdc_release_window(HWND hwnd, HDC hdc)	释放 Window HDC 句柄
HDC hdc_get_client(HWND hwnd)	获取 Client HDC 句柄
int hdc_release_client(HWND hwnd, HDC hdc)	释放 Client HDC 句柄
unsigned int hdc_get_current_group(HDC hdc)	获取当前颜色组
int hdc_set_current_group(HDC hdc, unsigned int group)	设置当前颜色组
GUI_COLOR hdc_get_group_color(HDC hdc, unsigned int group, unsigned int role)	获取组颜色
int hdc_set_group_color(HDC hdc, unsigned int group, unsigned int role, GUI_COLOR color)	设置组颜色
GUI_COLOR hdc_get_current_group_color(HDC hdc, unsigned int role)	获取当前组颜色
int hdc_set_current_group_color(HDC hdc, unsigned int role, GUI_COLOR color)	设置当前组颜色
int hdc_get_back_mode(HDC hdc)	获取背景模式
int hdc_set_back_mode(HDC hdc, int mode)	设置背景模式
int hdc_get_rect(HDC hdc, GUI_RECT *rect)	获取 HDC 区域
int hdc_set_rect(HDC hdc, GUI_RECT *rect)	设置 HDC 区域
int hdc_set_font(HDC hdc, const GUI_FONT *font)	设置 HDC 字库
int hdc_get_font(HDC hdc, GUI_FONT *font)	获取 HDC 字库
int hdc_get_font_width(HDC hdc)	获取 HDC 第一个字库中第一个字符宽度
int hdc_get_font_height(HDC hdc)	获取 HDC 第一个字库中第一个字符高度
int hdc_get_text_metric(HDC hdc, GUI_TEXT_METRIC *tm)	获取文本控制格式
int hdc_set_text_metric(HDC hdc, GUI_TEXT_METRIC *tm)	设置文本控制格式
int hdc_clear(HDC hdc)	清除屏幕
GUI_COLOR hdc_get_back_color(HDC hdc)	获取背景颜色
int hdc_set_back_color(HDC hdc, GUI_COLOR color)	设置背景颜色
GUI_COLOR hdc_get_fore_color(HDC hdc)	获取前景颜色
int hdc_set_fore_color(HDC hdc, GUI_COLOR color)	设置前景颜色
GUI_COLOR hdc_get_text_back_color(HDC hdc)	获取文本背景颜色
int hdc_set_text_back_color(HDC hdc, GUI_COLOR color)	设置文本背景颜色
GUI_COLOR hdc_get_text_fore_color(HDC hdc)	获取文本背景颜色
int hdc_set_text_fore_color(HDC hdc, GUI_COLOR color)	设置文本背景颜色

表 10-6 HDC 属性操作接口函数汇总表

第二部分　系统级别功能

第十一章　系统初始化及其缺省设置

11.1　系统初始化、缺省设置概述

系统初始化工作主要是调用 gui_open()函数进行系统内部数据初始化、消息队列初始化、打开显示屏幕、打开外设等工作，之后用户可以系统缺省字体、系统缺省消息路由等工作。如结束系统运行，则进行系统清除工作，最后调用 gui_close()函数，退出 main()函数。

11.2　系统初始化函数

函数原型：int　gui_open(void)
如果系统初始化成功，返回 1，否则返回-1。
该函数在用户注册驱动程序后，在 LearningGUI 其它的函数前调用。

11.3　系统关闭函数

函数原型：int　gui_close(void)
成功返回 1，否则返回-1。
该函数在 LearningGUI 其它的函数后调用。

11.4　Basic 版缺省设置

11.4.1　设置缺省字库：basic_set_default_font( )

函数原型：int　basic_set_default_font(const GUI_FONT *font)
函数参数：*font 用户字库指针。
成功返回 1，否则返回-1。

11.5　Window 版缺省设置

11.5.1　设置 Window 缺省字库：win_set_window_default_font( )

函数原型：int　win_set_window_default_font(const GUI_FONT *font)
函数参数：*font 用户字库指针。
成功返回 1，否则返回-1。

11.5.2　设置 Client 缺省字库：win_set_client_default_font( )

函数原型：int　win_set_client_default_font(const void *font)
函数参数：*font 用户字库指针。
成功返回 1，否则返回-1。

第十二章　LearningGUI 应用主程序模板

12.1　注册驱动程序

使用 in_register_gui_driver 函数分别注册显示驱动、键盘驱动（可选）、鼠标驱动（可选）。

12.2　系统初始化

使用 gui_open 函数进行系统初始化工作。

12.3　系统设置

主要进行缺省字库等设置工作。该步骤可选。

12.4　设置消息路由

使用 message_set_routine 函数进行设置消息缺省路由。用户可以需求不断变换消息路由。

12.5　用户初始化、主 GUI 创建和显示

创建和显示主 GUI。

12.6　消息主循环

该步骤是系统核心步骤。

12.7　系统清理

当退出消息循环后，进行必要的系统清理工作。

12.8　结束系统运行

使用 gui_close 函数进行系统关闭工作。

系统主程序结构演示代码如下：

```
int  main(void)
{
    GUI_MESSAGE  msg;
    int          ret;

    /*
     *  Step 1: register driver(s)
     */
    register_screen();
    #ifdef    _LG_KEYBOARD_
    register_keyboard();
    #endif
    #ifdef    _LG_MTJT_
    register_mtjt();
    #endif
    #ifndef  _LG_ALONE_VERSION_
    register_locker();
    #endif

    /*
     *  Step 2: call gui_open
     */
    ret = gui_open( );
```

```
    if ( ret < 0 )
        return  -1;

    /*
     *  Step 3: init system
     */


    /*
     *  Step 4: call message_set_routine
     */
    ret = message_set_routine(message_main_routine);
    if ( ret < 0 )
        return  -1;

    /*
     *  Step 5: create and show user GUI
     */
    paint_gui_main( );

    /*
     *  Step 6: message loop
     */
    while( 1 )
    {
        ret = message_get(&msg);
        if (ret > 0)
        {
            if ( MESSAGE_IS_QUIT(&msg) )
                break;

            message_dispatch(&msg);
        } else {
            /* POSIX function */
            usleep(10);
        }
    }

    /*
     *  Step 7: user clean
     */


    /*
     *  Step 8: call gui_open
     */
    gui_close( );

    return  1;
}
```

至于应用非 GUI 的业务代码，用户根据实际情况，放在适合的地方。

第十三章　光标管理

13.1　光标概述

光标是指鼠标指针在显示屏幕所显示的形状，其形状使用 GUI_ICON 数据结构来表示。光标属于可选项，需要宏_LG_CURSOR_支持。用户可以设置光标形状、允许或者禁止光标、显示或者隐藏光标等。

系统内置光标数组指针，支持的最大光标数目为 MAX_CURSORS。

光标的显示需要显示驱动接口中回读功能支持。

13.2　光标操作接口函数

13.2.1　允许光标显示：cursor_enable( )

函数原型：int cursor_enable(void)

返回 1。

13.2.2　禁止光标显示：cursor_disable( )

函数原型：int cursor_disable(void)

返回 1。

13.2.3　显示光标：cursor_show( )

函数原型：int cursor_show(void)

返回 1。

13.2.4　隐藏光标：cursor_hide( )

函数原型：int cursor_hide(void)

返回 1。

13.2.5　获取光标坐标：cursor_get_position( )

函数原型：int cursor_get_position(GUI_POINT *point)

返回 1。

13.2.6　设置光标坐标：cursor_set_position( )

函数原型：int cursor_set_position(int x, int y)

返回 1。

13.2.7　设置光标形状：cusor_set_shape( )

函数原型：int cusor_set_shape(unsigned int cursor_id, const GUI_ICON *shape)

返回 1。

13.2.8　获取当前光标 ID：cursor_get_id( )

函数原型：int cursor_get_id(void)

返回 1。

13.2.9　设置当前光标 ID：cursor_set_id( )

函数原型：int cursor_set_id(unsigned int cursor_id)

返回 1。

13.2.10　保存自绘区域：cursor_maybe_restore_back_abs( )

函数原型：int cursor_maybe_restore_back_abs(int left, int top, int right, int bottom)

返回 1。

该函数是 Windows 专用函数。在自绘开始前，需要保存自绘区域。该函数参数是绝对坐标。

13.2.11　刷新光标：cursor_maybe_refresh( )

函数原型：int cursor_maybe_refresh(void)
返回 1。
该函数是 Windows 专用函数。自绘结束后，需要调用该函数刷新光标。

第十四章　屏幕快照

14.1　屏幕快照（抓屏）概述

屏幕快照保存为 Bitmap 文件。可以使用按键（GUI_KEY_PRINT 或者 GUI_KEY_SYS_REQ）快照、或者直接调用接口函数捕捉快照。

屏幕快照功能需要显示驱动接口中回读功能和 POSIX 标准 write（）函数支持。

14.2　屏幕快照操作接口函数

14.2.1　设置屏幕快照区域：snapshot_set_rect( )

函数原型：int snapshot_set_rect(GUI_RECT *rect)

返回 1。

14.2.2　设置快照窗口：win_snapshot_set_hwnd( )

函数原型：int win_snapshot_set_hwnd(HWND *hwnd)

函数参数：

hwnd:　　HWND 句柄

返回 1。

该函数是 Windows 专用函数。设置用户捕捉窗口的快照。

14.2.3　保存屏幕快照：snapshot_get ( )

函数原型：int snapshot_get(void)

返回 1。

第十五章　屏幕控制

15.1　屏幕控制概述

应用可以控制屏幕，如开、关背光。但是不是必须的。除非特殊需要，用户一般地不涉及到此功能。

15.2　屏幕控制操作接口函数

15.2.1　获取屏幕宽度高度：screen_get_width_height( )

函数原型：int screen_get_width_height(unsigned int *width, unsigned int *height)

函数参数：

*width:　　屏幕宽度指针

* height:　　屏幕高度指针

返回 1。

15.2.2　控制屏幕属性：screen_control( )

函数原型：int screen_control(void *p1, void *p2)

函数参数：

*p1:　　用户参数 1

*p2:　　用户参数 2

返回 1。

15.2.3　打开屏幕：screen_on( )

函数原型：int screen_on(void)

函数参数：

返回 1。

15.2.4　关闭屏幕：screen_off( )

函数原型：int screen_off(void)

函数参数：

返回 1。

15.2.5　重新初始化屏幕：screen_reinit( )

函数原型：int screen_reinit(void)

函数参数：

返回 1。

第十六章　键盘控制

16.1　键盘控制概述

同理，应用可以控制键盘。但是不是必须的。除非特殊需要，用户一般地不涉及到此功能。

16.2　键盘控制操作接口函数

16.2.1　控制屏幕属性：keyboard_control( )

函数原型：int keyboard_control(void *p1, void *p2)

函数参数：

*p1:　　用户参数 1

*p2:　　用户参数 2

返回 1。

16.2.2　重新初始化键盘：keyboard_reinit( )

函数原型：int keyboard_reinit(void)

函数参数：

返回 1。

第十七章　MTJT 控制

17.1　MTJT 控制概述

同理，应用可以控制 MTJT。但是不是必须的。除非特殊需要，用户一般地不涉及到此功能。

17.2　MTJT 控制操作接口函数

17.2.1　控制 MTJT 属性：mtjt_control( )

函数原型：int mtjt_control(void *p1, void *p2)

函数参数：

*p1:　　用户参数 1

*p2:　　用户参数 2

返回 1。

17.2.2　重新初始化屏幕：mtjt_reinit( )

函数原型：int mtjt_reinit(void)

函数参数：

返回 1。

第十八章　系统绘制机制概述

系统使用 HDC 前景颜色绘制点、线、弧；使用 HDC 背景颜色进行填充操作；使用 HDC 文本前景颜色显示文本颜色，如果是覆盖模式的话，同时使用 HDC 文本背景颜色填充背景；在显示图像时，如果是透明模式的话，则使用 HDC 背景颜色填充背景。

一次绘制序列遵循以下步骤：

1）获取 HDC 句柄

如果绘制 Basic 版功能，则使用 hdc_get_basic()函数获取 HDC 句柄；如果绘制 Window 区域（用户不常用），则使用 hdc_get_window()函数获取 HDC 句柄；如果绘制 Client 区域，则使用 hdc_get_client()函数获取 HDC 句柄。

2）具体绘制操作

在 Basic 版中，用户调用 2D 绘制函数、文本显示函数、图像显示函数等进行绘制操作。

在 Window 版中，窗口的绘制由系统完成。但是如果改变窗口的显示外观或者绘制用户 GUI，用户需要调用相关的绘制函数来完成。同时用户可以向窗口发送 MSG_PAINT 消息、或者调用窗口无效函数，以便达到重绘一次窗口目的。

该绘制操作步骤可以连续多次执行。

3）释放 HDC 句柄

用户完成一次绘制序列后，需要释放 HDC 句柄。

如果第一步使用 hdc_get_basic()函数，则采用 hdc_release_basic()函数释放 HDC；如果第一步使用 hdc_get_window()函数，则采用 hdc_release_window()函数释放 HDC；如果第一步使用 hdc_get_client()函数，则采用 hdc_release_client()函数释放 HDC。

第三部分　系统 Basic 版本

第十九章　Basic 版概述

Basic 版相对简易，用户直接在显示屏幕上进行绘制 GUI。在相同的区域中，后一次的绘制覆盖前一次的绘制。其逻辑坐标和物理坐标重合。

Basic 版主要功能包括 2D 绘制、文本显示、图像显示。

第二十章　2D 绘制

20.1　写像素点

函数原型：int point_output_pixel(HDC hdc, int x, int y)
函数参数：
    hdc:　HDC 句柄
    x:　X 坐标
    y:　Y 坐标
成功返回 1，否则返回-1。

20.2　读像素点

函数原型：int point_input_pixel(HDC hdc, int x, int y, GUI_COLOR *color)
函数参数：
    hdc:　HDC 句柄
    x:　X 坐标
    y:　Y 坐标
    color:　读入的颜色地址
成功返回 1，否则返回-1。

20.3　画直线

函数原型：int line(HDC hdc, int x0, int y0, int x1, int y1)
函数参数：
    hdc:　HDC 句柄
    x0:　左端点 X 坐标
    y0:　左端点 Y 坐标
    x1:　右端点 X 坐标
    y1:　右端点 Y 坐标
成功返回 1，否则返回-1。
该函数可以画水平线或垂直线。

20.4　画曲线

画曲线分成两步：
    1）设置曲线的起始端点；
    2）将起始端点和终点连成一条直线。

15.4.1　设置曲线起始端点：move_to

函数原型：int move_to(HDC hdc, int x, int y)
成功返回 1，否则返回-1。

15.4.2　连接起始点和终点：line_to

函数原型：int line_to(HDC hdc, int x, int y)
成功返回 1，否则返回-1。

20.5　画矩形

函数原型：int int rect_frame(HDC hdc, int left, int top, int right, int bottom)
函数参数：
    hdc:　HDC 句柄
    left:　左上角 X 坐标
    top:　左上角 Y 坐标

right: 右下角 X 坐标
bottom: 右下角 Y 坐标
成功返回 1，否则返回-1。

20.6 填充矩形

函数原型：int int rect_fill(HDC hdc, int left, int top, int right, int bottom)
函数参数：
hdc: HDC 句柄
left: 左上角 X 坐标
top: 左上角 Y 坐标
right: 右下角 X 坐标
bottom: 右下角 Y 坐标
成功返回 1，否则返回-1。

20.7 画圆

函数原型：int circle(HDC hdc, int x, int y, unsigned int r)
函数参数：
hdc: HDC 句柄
x: 圆心 X 坐标
y: 圆心 Y 坐标
r: 圆半径
成功返回 1，否则返回-1。

20.8 填充实心圆

函数原型：int circle_fill(HDC hdc, int x, int y, unsigned int r)
函数参数：
hdc: HDC 句柄
x: 圆心 X 坐标
y: 圆心 Y 坐标
r: 圆半径
成功返回 1，否则返回-1。

第二十一章　文本显示

21.1　显示字符串

函数原型：　　int　text_out(HDC hdc, int x, int y, const TCHAR *str, int code_counter)

函数参数：

hdc:　　HDC 句柄

x:　　X 坐标

y:　　Y 坐标

*str:　　字符串文本

code_counter:　字符串文本长度

成功返回 1，否则返回-1。

第二十二章　图像显示

22.1　图像显示概述

系统通过图像转换工具软件将图像文件转换成 C 语言格式文件而进行显示。图像文件转换后，分别形成.h 和.c 文件，.h 文件是声明图像数据变量文件，.c 文件是图像数据定义文件，将.h 和.c 添加到应用开发项目后，就能通过图像数据变量进行图像显示。

Bitmap 图像需要宏_LG_BITMAP_支持、Icon 图像需要宏_LG_ICON_支持、Gif 图像需要宏_LG_GIF_支持。

22.2　图像转换工具软件

该图像转换工具软件 LguiTool.exe 是 PC Windows 下应用软件（位于源码包/tools 目录下），其作用将 Bitmap 图像、Icon 图像、Gif 图像转换成 LearningGUI 能够识别的 C 语言格式文件。启动 LguiTool.exe 后，按照相应的菜单操作，就能够完成图像转换。LguiTool 启动后，图像装换菜单如图 22-1 所示。

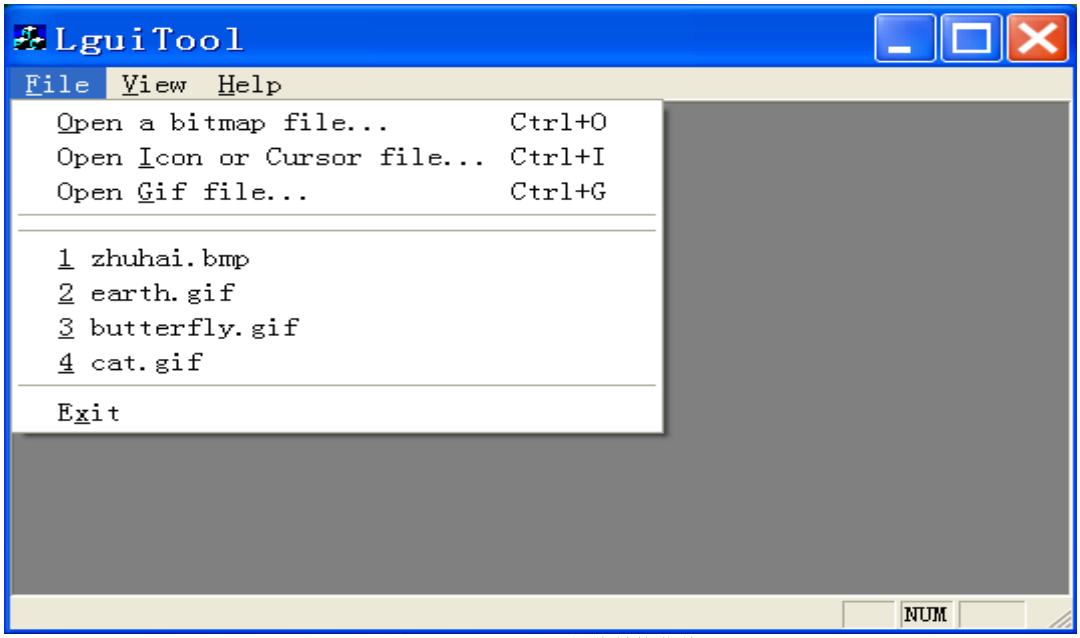

图 22-1　LguiTool 图像转换菜单

打开相应的 Bitmap 后，显示界面类似于图 22-2 所示。

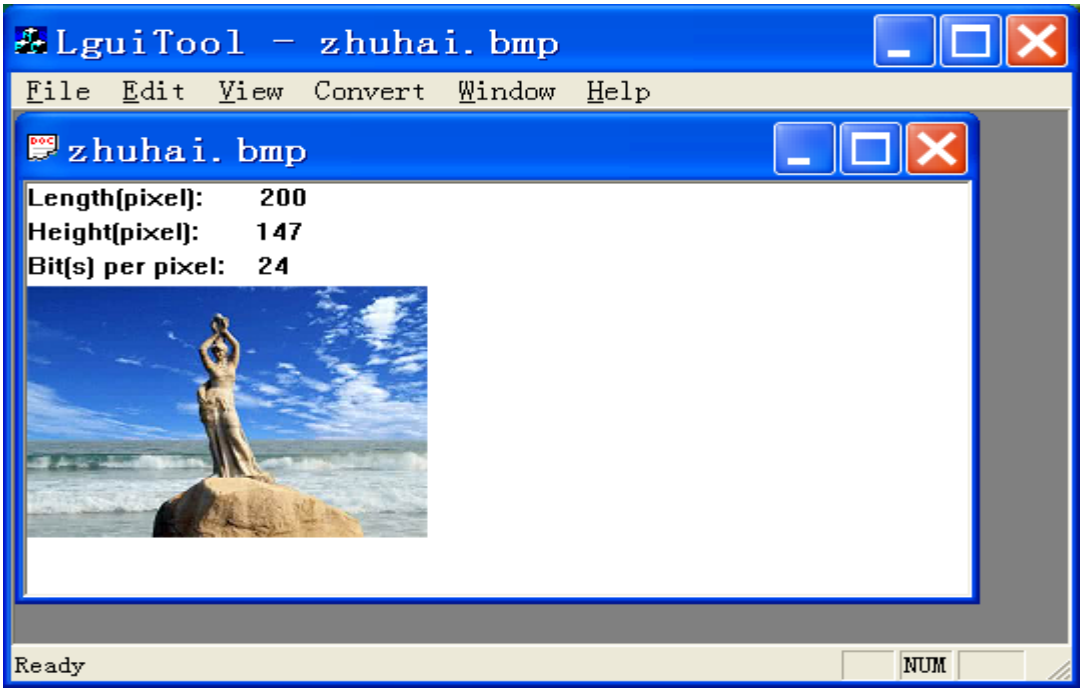

图 22-2　LguiTool 显示 bitmap 图像

点击 Convert 菜单项后，出现如图 22-3 所示的 Bitmap 转换界面。

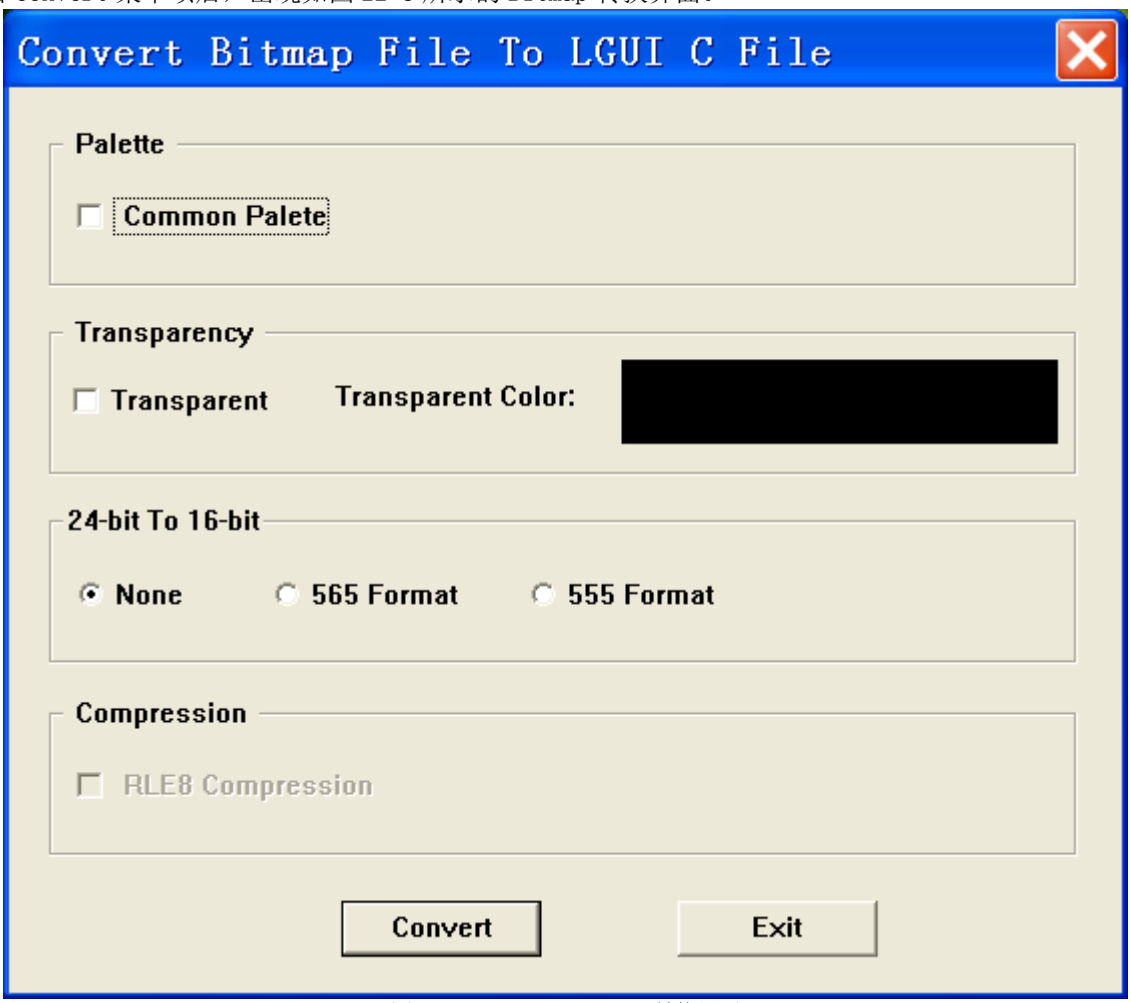

图 22-3 LguiTool bitmap 转换界面

一般直接点击 Convert 按钮，就能在当前目录生成相应的.h 和.c 文件，之后，在应用中能够使用相应的图像变量。

同理，可以进行 Icon 图像、Gif 图像转换。

22.3 Bitmap 图像显示

系统定义 GUI_BITMAP 结构体来表示 Bitmap 图像。GUI_BITMAP 定义如下。

```
struct  _GUI_BITMAP
{
          unsigned int    width;
          unsigned int    height;
          unsigned int    bits_per_pixel;
          unsigned int    bytes_per_line;
          GUI_COLOR       transparent_color;
          unsigned int    is_transparent;
          unsigned int    bit16_format;
          unsigned int    is_rle8_format;
    const GUI_PALETTE    *palette;
    const unsigned char  *data;
};
typedef  struct  _GUI_BITMAP   GUI_BITMAP;
```

GUI_BITMAP 结构体各个成员描述如表 22-1 所示。

结构体成员	成员描述
width	图像宽度(像素单位)
height	图像宽度(像素单位)

bits_per_pixel	每像素所占位数
bytes_per_line	每行（线）所占字节数目
transparent_color	透明颜色
is_transparent	是否做透明处理
bit16_format	16 位颜色格式
is_rle8_format	是否是 RLE8 压缩格式
*palette	调色板指针
*data	图像数据

表 22-1 GUI_BITMAP 结构体成员描述表

22.3.1 显示 Bitmap 图像：bitmap_fill( )

函数原型：int bitmap_fill(HDC hdc, int x, int y, const GUI_BITMAP *bitmap)

函数参数：

hdc: HDC 句柄

x: X 坐标

y: Y 坐标

*bitmap: GUI_BITMAP 变量指针

成功返回 1，否则返回-1。

22.4 Icon 图像显示

系统定义 GUI_ICON 结构体来表示 Icon 图像。GUI_ICON 定义如下。

```
struct  _GUI_ICON
{
          unsigned int    type;
          unsigned int    width;
          unsigned int    height;
          unsigned int    bits_per_pixel;
          unsigned int    bytes_per_line;
          unsigned int    left;
          unsigned int    top;
          unsigned int    bit16_format;
          unsigned int    is_rle8_format;
    const GUI_PALETTE    *palette;
          unsigned char   panes_left;
          unsigned char   bpp_top;
    const unsigned char  *xor_data;
    const unsigned char  *and_data;
};
typedef  struct  _GUI_ICON   GUI_ICON;
```

GUI_ICON 结构体各个成员描述如表 22-2 所示。

结构体成员	成员描述
type	ICON 类型，取值下述之一： ICON_TYPE CURSOR_TYPE
width	图像宽度(像素单位)
height	图像宽度(像素单位)
bits_per_pixel	每像素所占位数
bytes_per_line	每行（线）所占字节数目
left	左偏移
top	上偏移
bit16_format	16 位颜色格式

is_rle8_format	是否是 RLE8 压缩格式
panes_left	位面板数
bpp_top	每像素所占的位数
*palette	调色板指针
*xor_data	异或数据
*and_data	与数据

表 22-2 GUI_ICON 结构体成员描述表

22.4.1 显示 Icon 图像：icon_fill( )

函数原型：int icon_fill(HDC hdc, int x, int y, const GUI_ICON *icon)

函数参数：

hdc: HDC 句柄

x: X 坐标

y: Y 坐标

*bitmap: GUI_ICON 变量指针

成功返回 1，否则返回-1。

22.5 Gif 图像显示

Gif 图像相关的数据结构定义如下：

```
/* Gif 逻辑屏幕结构体 */
struct  _GIF_LOGICAL_SCREEN
{
    UINT16   width;
    UINT16   height;
    UINT8    packed_fields;
    UINT8    back_color_index;
    UINT8    pixel_aspect_ration;
};
typedef  struct  _GIF_LOGICAL_SCREEN   GIF_LOGICAL_SCREEN;

/* Gif 控制扩展结构体 */
struct  _GIF_CONTROL_EXTENSION
{
    UINT8    packed_fields;
    UINT16   delay_time;
    UINT8    trans_color_index;
};
typedef  struct  _GIF_CONTROL_EXTENSION   GIF_CONTROL_EXTENSION;

/* Gif 图像描述结构体 */
struct  _GIF_IMAGE_DESCRIPTOR
{
    UINT16         left;
    UINT16         top;
    UINT16         width;
    UINT16         height;
    UINT8          packed_fields;
    GUI_PALETTE    *local_palette;
    UINT8          data_type;
    UINT8          lzw_code_size;
    UINT           len;
    unsigned char  *data;
};
typedef  struct  _GIF_IMAGE_DESCRIPTOR   GIF_IMAGE_DESCRIPTOR;

/* Gif 文本扩展结构体 */
struct  _GIF_PLAIN_TEXT_EXTENSION
{
    UINT16         text_left;
    UINT16         text_top;
```

```
    UINT16          text_width;
    UINT16          text_height;
    UINT8           cell_width;
    UINT8           cell_height;
    UINT8           foreground_color_index;
    UINT8           back_color_index;
    UINT            len;
    unsigned char  *text;
};
typedef  struct  _GIF_PLAIN_TEXT_EXTENSION   GIF_PLAIN_TEXT_EXTENSION;

/* Gif 控制块结构体 */
struct  _GIF_BLOCK
{
          unsigned char           type;
    const GIF_CONTROL_EXTENSION  *control;
    const void                   *block;
};
typedef  struct  _GIF_BLOCK   GIF_BLOCK;

/* GUI_GIF 结构体 */
struct  _GUI_GIF
{
    const GIF_LOGICAL_SCREEN  *screen;
    const GUI_PALETTE         *global_palette;
          UINT                 block_num;
    const GIF_BLOCK           *block_list;
};
typedef  struct  _GUI_GIF   GUI_GIF;
```

GIF_LOGICAL_SCREEN 结构体各个成员描述如表 22-3 所示。

结构体成员	成员描述
width	图像宽度
height	图像宽度
packed_fields	内置域字段
back_color_index	背景颜色索引
pixel_aspect_ration	像素长宽比

表 22-3 GIF_LOGICAL_SCREEN 结构体成员描述表

GIF_CONTROL_EXTENSION 结构体各个成员描述如表 22-4 所示。

结构体成员	成员描述
packed_fields	内置域字段
delay_time	延时
trans_color_index	透明颜色索引

表 22-4 GIF_CONTROL_EXTENSION 结构体成员描述表

GIF_IMAGE_DESCRIPTOR 结构体各个成员描述如表 22-5 所示。

结构体成员	成员描述
left	左偏移
top	上偏移
width	图像宽度
height	图像宽度
packed_fields	内置域字段
*local_palette	本地调色板
data_type	数据类型，取值下列之一： GIF_UNCOMPRESS_DATA

	GIF_COMPRESS_DATA
lzw_code_size	LZW 码大小
len	数据长度
*data	数据指针

表 22-5 GIF_IMAGE_DESCRIPTOR 结构体成员描述表

GIF_PLAIN_TEXT_EXTENSION 结构体各个成员描述如表 22-6 所示。

结构体成员	成员描述
text_left	左偏移
text_top	上偏移
text_width	字符串宽度
text_height	字符串宽度
cell_width	所占单元格宽度
cell_height	所占单元格高度
fore_color_index	前景颜色索引
back_color_index	背景颜色索引
len	字符串长度
*text	字符串数据指针

表 22-6 GIF_PLAIN_TEXT_EXTENSION 结构体成员描述表

GIF_BLOCK 结构体各个成员描述如表 22-7 所示。

结构体成员	成员描述
type	数据块类型，取值下列之一： GIF_IMAGE_TYPE GIF_TEXT_TYPE
*control	GIF_CONTROL_EXTENSION 控制扩展块指针
*block	数据块指针

表 22-7 GIF_BLOCK 结构体成员描述表

GUI_GIF 结构体各个成员描述如表 22-8 所示。

结构体成员	成员描述
*screen	GIF_LOGICAL_SCREEN 结构指针
*global_palette	GUI_PALETTE 全局调色板指针
block_num	GIF_BLOCK 数据块数目
*block_list	GIF_BLOCK 数据块指针

表 22-8 GUI_GIF 结构体成员描述表

22.5.1 手动播放 Gif 帧：gif_man_play( )

函数原型：int gif_man_play(HDC hdc, int x, int y, const GUI_GIF *gui_gif, unsigned int frame_id)

函数参数：

hdc: HDC 句柄

x: X 坐标

y: Y 坐标

* gui_gif: GUI_GIF 变量指针

frame_id: 帧编号

成功返回 1，否则返回-1。

22.5.2 自动播放 Gif 动画：gif_auto_play( )

函数原型：int gif_auto_play(HDC hdc, int x, int y, const GUI_GIF *gui_gif, unsigned int *frame_id)
函数参数：

hdc: HDC 句柄
x: X 坐标
y: Y 坐标
* gui_gif: GUI_GIF 变量指针
*frame_id: 全局变量指针

成功返回 1，否则返回-1。
该函数需要在 MSG_COUNTER 或者 MSG_TIMER 消息中调用，系统自动播放 Gif 动画。

第二十三章　Basic 版演示程序概述

Basic 演示程序位于 examples/Baisic 目录下。主要演示字库使用、文本输出和图像显示。子目录名称一般能表示演示的主题。

图 23-1 演示 GB2312-80 字库的使用。

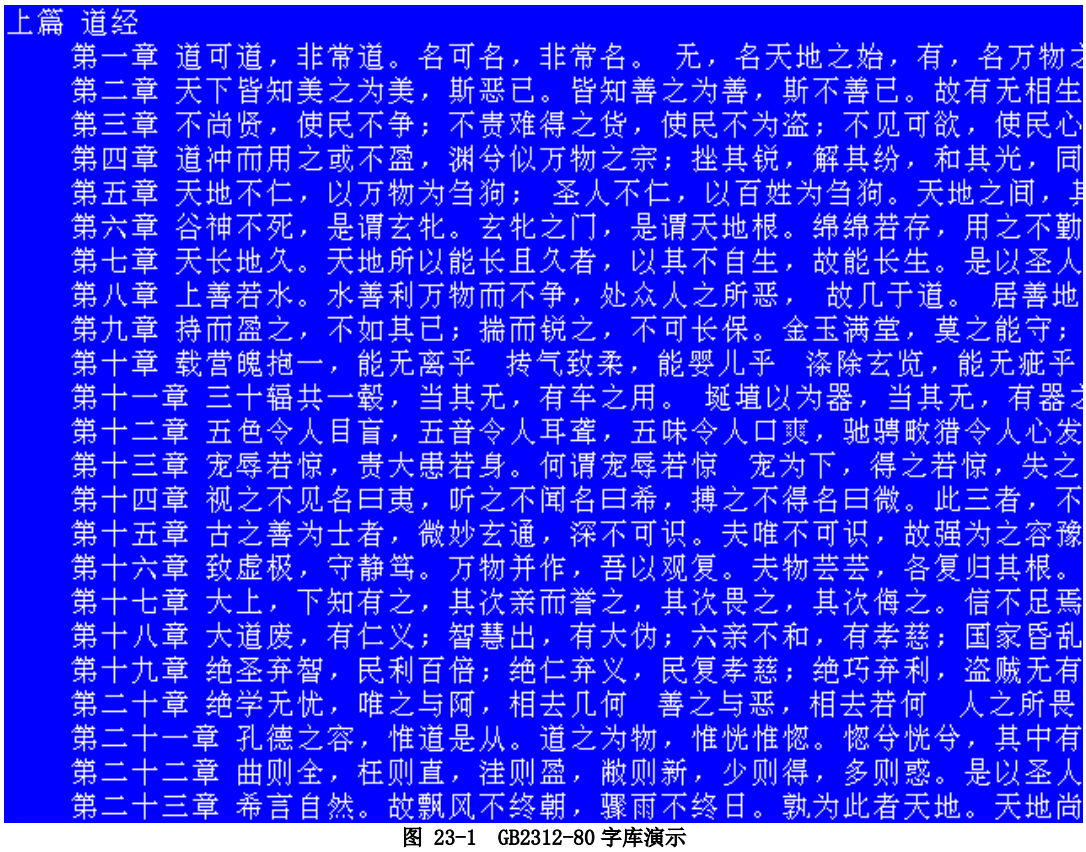

图 23-1　GB2312-80 字库演示

图 23-2 演示 ASCII 字库和 GB2312-80 字库的混合使用。

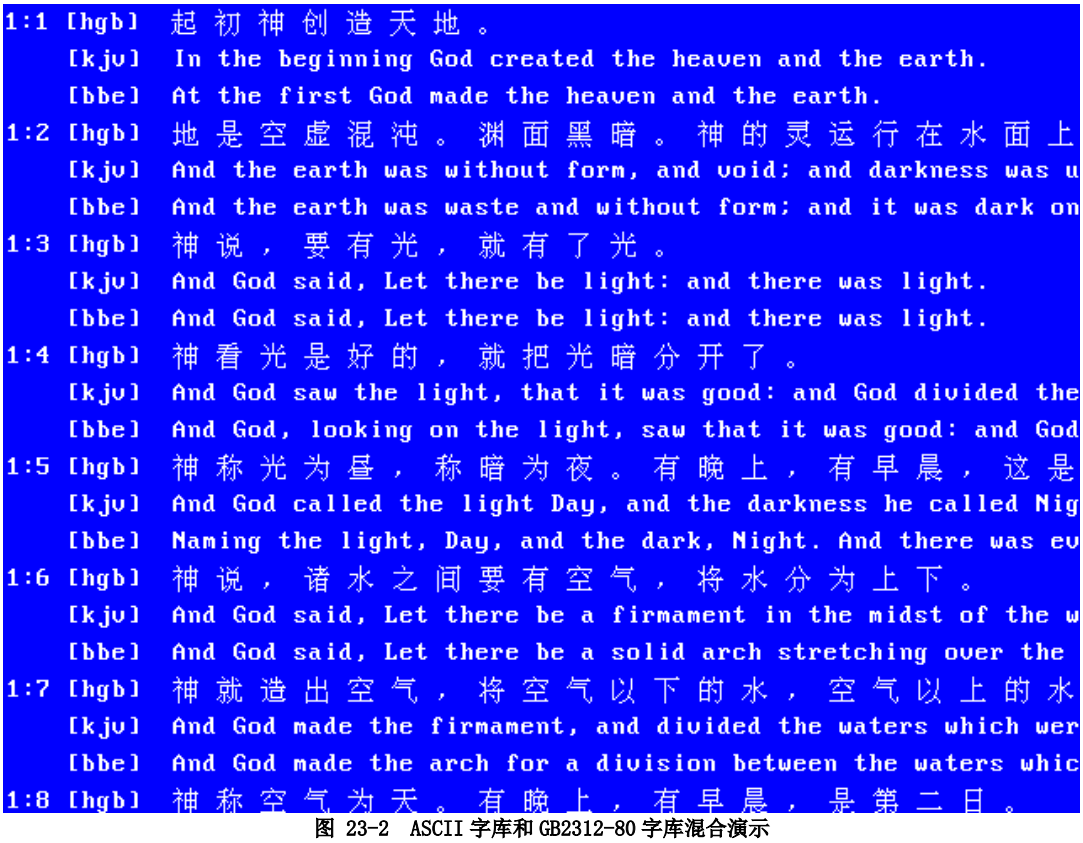

图 23-2　ASCII 字库和 GB2312-80 字库混合演示

图 23-3 演示自定义 ASCII 字库的使用。

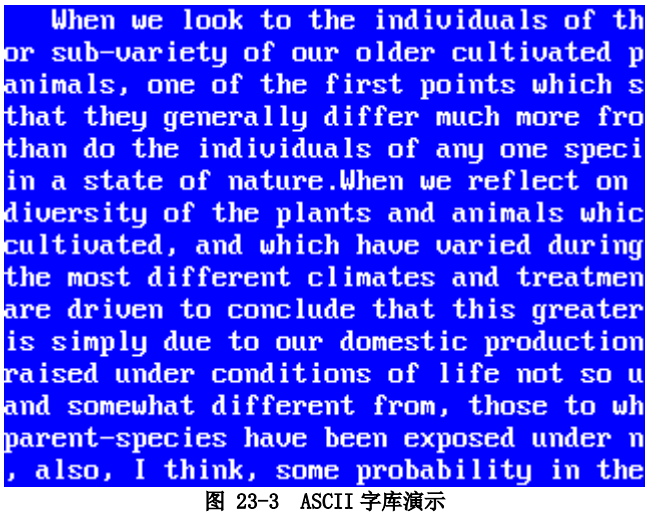

图 23-3 ASCII 字库演示

图 23-4 演示自定义 ASCII 字库和自定义小汉字库的混合使用。

图 23-4 ASCII 字库和用户自定义小汉字库演示

图 23-5 演示自定义 ASCII 字库和自定义小汉字库的混合使用。

图 23-5 ASCII 字库和用户自定义小汉字库演示

图 23-6 演示自定义 ASCII 字库和自定义小汉字库的混合使用。

图 23-6 ASCII 字库和用户自定义小汉字库演示

图 23-7 演示自定义数字小字库的使用。

图 23-7 数字小字库演示

图 23-8 演示 Unicode 字库的使用。

图 23-8 Unicode 字库演示

图 23-9 演示 Bitmap 图片显示。

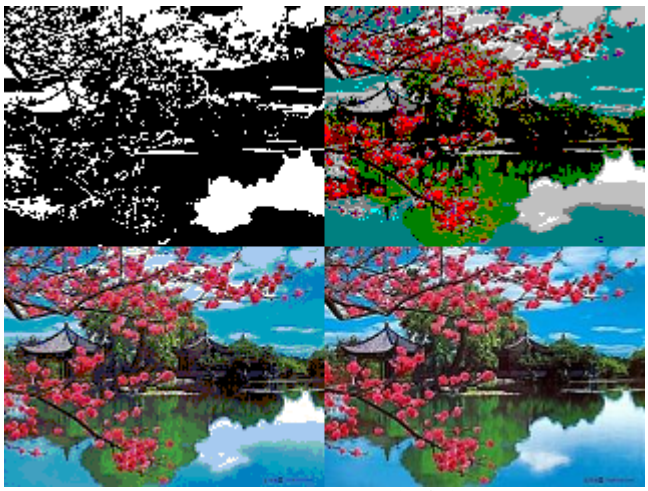

图 23-9 Basic Bitmap 图片演示

图 23-10 演示 Icon 图片显示。

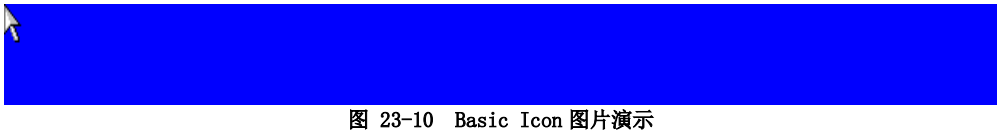

图 23-10 Basic Icon 图片演示

图 23-11 演示 Gif 动画播放。

图 23-11 Basic Gif 动画播放

Basic 模拟项目演示程序位于 examples/project/basic_ecg_monitor 目录下，模拟病人心电监护仪，屏幕快照如图 23-12 所示。

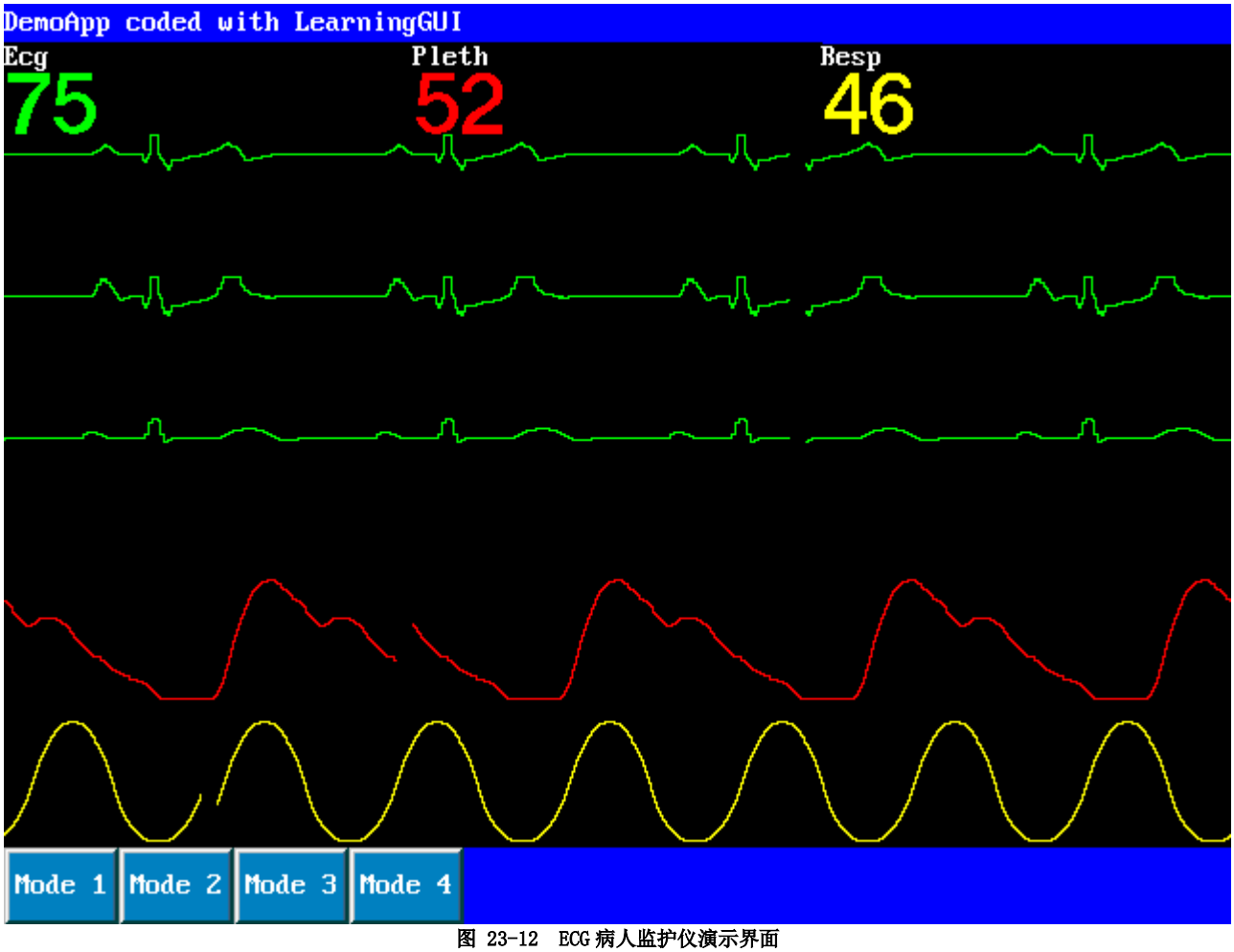

图 23-12 ECG 病人监护仪演示界面

第四部分　系统 Windows 版本

第二十四章　Windows 版概述

系统 Windows 版是建立在 Basic 版之上的可选部分。Windows 主要目的是窗口（小部件 widget）管理。其主要内容是管理窗口的创建、显示、窗口自动绘制、Windows 消息处理和窗口删除（窗口关闭）。窗口都是动态创建。窗口创建后，使用 HWND 句柄来标示、操作窗口。从用户角度来看，Windows 版主要功能围绕 HWND 句柄进行相关的操作。

在 Windows 初始化后，系统自动创建了桌面窗口。

Windows 自绘过程实际上是调用 Basic 版的相关函数。

除非特殊说明，Windows 坐标是以窗口左上角为原点的相对坐标。

24.1　窗口之间关系

根据窗口创建的节点，系统将窗口和窗口之间关系划分为父子窗口、兄弟窗口。

父窗口是子窗口的父亲，子窗口是父窗口的儿子，一旦父窗口刷新，系统自动刷新其下所有子窗口（如果子窗口是可视的话）；同时如果删除父窗口的话，系统自动删除其下所有的子窗口。

兄弟窗口是指拥有相同父窗口的窗口。兄弟窗口根据创建的先后顺序通过链表连接在一起，具备前后关系。

24.2　窗口显示和重叠顺序

系统先绘制父窗口，后绘制子窗口。子窗口重叠在父窗口上面，并且显示区域不超出父窗口区域。兄弟窗口依据创建顺序进行先后绘制和重叠（如果有重叠区域的话），后创建的窗口重叠在先创建窗口的上面。

通过设置焦点或者函数调用，可以改变窗口重叠顺序（创建顺序）。

24.3　窗口分类

根据父窗口，系统将窗口划分以下级别：

1）0 级窗口

不存在父窗口的窗口。桌面窗口是 0 级窗口，系统使用 HWND_DESKTOP 来标示，由系统创建和管理，应用不能删除该窗口，但是应用可以通过 HWND_DESKTOP 发送消息和进行相关的设置。

2）1 级窗口

父窗口是桌面窗口的窗口。通常所指系统主窗口。

3）2 级窗口

父窗口是 1 级窗口的窗口。通常所指主窗口中小部件窗口。

4）n 级窗口

父窗口是(n-1)级窗口的窗口。

24.4　缺省窗口

缺省窗口是指如果下次得到 GUI_KEY_ENTER 输入的话，那么该窗口设置成焦点的窗口。

24.5　幽灵（ghost）窗口

幽灵窗口是指正常的窗口，能接受鼠标、键盘输入（如果是焦点窗口的话），响应消息，但是不会在显示屏幕上显示出外观。主要应用在隐性触摸场合（隐性触摸键），方便显示背景图像。这样一来，用户可以设计操作性强、漂亮的背景图像。

24.6　窗口活跃状态

24.6.1　焦点窗口

焦点窗口也称之为当前窗口、活动窗口，是指即能接受键盘输入又能接受鼠标输入的窗口。焦点窗口使用焦点颜色组来绘制。

24.6.2　非活动窗口

非活动窗口不接受键盘输入的窗口、但是能接受鼠标输入的窗口。非活动窗口使用非活动颜色组来绘制。

24.6.3 禁止窗口

禁止窗口即不接受键盘输入的窗口、也不接受鼠标输入窗口。禁止窗口使用禁止颜色组来绘制。

无论窗口处于什么样的活跃状态，都接受用户程序控制，活跃状态都可以通过输入或者程序改变。

24.7 窗口显示状态

24.7.1 正常显示状态

窗口以窗口所定义的区域来显示，这就是正常显示状态。

24.7.2 极大化状态

在窗口所定义的区域不是整个显示屏幕的情况下，窗口充满整个显示屏幕，这种状态为极大化状态。

24.7.2 极小化状态

窗口影藏的状态为极小化状态。

同理，无论窗口处于什么样的显示状态，都接受用户程序控制，显示状态都可以通过输入或者程序改变。

24.8 窗口无效区域

窗口无效区域是指系统需要重绘的区域。应用可以设置窗口的无效区域。

24.9 窗口全局功能键

系统内置了如表 24-1 所示的全局功能键。

键码	功能
GUI_KEY_PAGE_UP	将上一个 1 级窗口设置为焦点窗口
GUI_KEY_PAGE_DOWN	将下一个 1 级窗口设置为焦点窗口
GUI_KEY_BACK_TAB	将上一个 1 级窗口设置为缺省窗口
GUI_KEY_TAB	将下一个 2 级窗口设置为缺省窗口

表 24-1 系统全局功能键

第二十五章　窗口的创建

25.1　HWND 句柄概述

HWND 代表一个窗口指针，在窗口（小部件）被创建后生成。严格禁止用户直接操作 HWND，必须通过系统提供的函数操作 HWND 句柄。

25.2　创建窗口的通用方法

每个 widget 窗口都由专用函数来创建。

创建函数名称规则是：widget 名称_create，如 frame_create、label_create 等。

创建函数参数有三个参数：

参数 1：表示父窗口 HWND。如标示为 NULL，则表示父窗口是桌面窗口。

参数 2：表示窗口通用参数结构指针。

参数 3：表示窗口专用参数结构指针。

创建窗口通用函数声明如下：

```
HWND  xxx_create(HWND parent, GUI_COMMON_WIDGET *p1, GUI_XXX *p2)
```

其中 xxx 表示具体小部件名称。

如果创建成功，返回有效的 HWND 句柄，否则返回无效句柄（NULL）。

如果不再需要使用 HWND 句柄时，需要关闭该窗口。

25.3　窗口通用参数结构体

从应用角度来看，窗口通用结构体 GUI_COMMON_WIDGET 定义如下：

```
struct  _GUI_COMMON_WIDGET
{
    unsigned int  id;

    int           left;
    int           top;
    int           right;
    int           bottom;

    UINT          style;
    UINT          ext_style;

    BINT          is_app_callback;
    int           (*app_callback)(void *msg);

    #ifdef  _LG_WINDOW_BACKGROUND_IMAGE_
    BINT          bimage_flag;
    BINT          bimage_type;
    BINT          bimage_align;
    const  void  *pimage;
    #endif

    #ifdef  _LG_WIDGET_USER_DATA_
    unsigned int  max_data_bytes;
    unsigned int  data_bytes;
    void         *pdata;
    #endif
};
typedef   struct    _GUI_COMMON_WIDGET  GUI_COMMON_WIDGET;
```

GUI_COMMON_WIDGET 结构成员描述如表 25-1 所示。

成员	描述
id	窗口 ID
left	左上角 X 相对坐标

top	左上角 Y 相对坐标
right	右下角 X 相对坐标
bottom	右下角 Y 相对坐标
style	窗口式样。取值范围如下（或操作）： VISUAL_STYLE： 可视 ENABLE_STYLE： 允许 BORDER_STYLE： 边界 如果该属性有效的话： BORDER_3D_STYLE 表示 3D 边界式样 WINBAR_STYLE： 具备 Windows Bar 如果该属性有效的话： WINBAR_CLOSE_BTN_STYLE 表示具备 CLOSE 按钮 WINBAR_MAX_BTN_STYLE 表示具备 MAX 按钮 WINBAR_MIN_BTN_STYLE 表示具备 MIN 按钮
ext_style	扩展式样。不同的小部件，扩展式样不同。
is_app_callback	应用回调函数标志
(*app_callback)(GUI_MESSAGE *msg)	应用回调函数
bimage_flag	背景图像标志
bimage_type	背景图像类型
bimage_align	背景图像对齐标志
pimage	背景图像数据指针
max_data_bytes	用户区域最大数据
data_bytes	用户区域实际数据
pdata	用户数据指针

表 25-1 GUI_COMMON_WIDGET 结构体成员表

25.4 窗口专用参数结构体

窗口专用参数结构体定义与具体的小部件相关。

第二十六章　窗口标题栏

26.1　Windows Bar 概述

窗口标题栏 Windows Bar 不是独立的 Widget 小部件，而是属于 Widget 小部件的一个扩展功能属性项。

26.2　Windows Bar 式样

窗口标题栏 Windows Bar 具备如表 26-1 所示式样。

式样	描述
WINBAR_CLOSE_BTN_STYLE	关闭按钮
WINBAR_MAX_BTN_STYLE	极大化按钮
WINBAR_MIN_BTN_STYLE	极小化按钮

表 26-1　Windows Bar 式样列表

26.3　Windows Bar 操作接口

26.3.1　获取窗口标题栏文本：win_get_window_bar_text( )

函数原型：int win_get_window_bar_text(HWND hwnd, TCHAR *text, unsigned int *text_len)

函数参数：

hwnd:　　HWND 句柄

text:　　存放返回窗口标题栏文本缓冲区

text_len:　　作为输入参数时，表示缓冲区长度

　　作为输出参数时，表示返回文本长度

成功返回 1；否则返回-1。

26.3.2　设置窗口标题栏文本：win_set_window_bar_text( )

函数原型：int win_set_window_bar_text(HWND hwnd, TCHAR *text, unsigned int text_len)

函数参数：

hwnd:　　HWND 句柄

text:　　窗口标题栏文本缓冲区

text_len:　　缓冲区文本长度

成功返回 1；否则返回-1。

第二十七章　窗口通用操作

27.1　窗口通用操作概述

除桌面窗口 HWND_DESKTOP 句柄外，HWND 句柄遵循相同的操作。

27.2　窗口背景图像数据结构体

窗口背景图像 GUI_BACKGROUND_IMAGE 结构体定义如下：

```
struct  _GUI_BACKGROUND_IMAGE
{
    BINT          bimage_flag;
    BINT          bimage_type;
    BINT          bimage_align;
    const  void  *pimage;
};
typedef   struct    _GUI_BACKGROUND_IMAGE  GUI_BACKGROUND_IMAGE;
```

GUI_BACKGROUND_IMAGE 结构成员描述如表 27-1 所示。

成员	描述
bimage_flag	背景图像标志
bimage_type	背景图像类型
bimage_align	背景图像对齐标志
pimage	背景图像数据指针

表 27-1　GUI_BACKGROUND_IMAGE 结构体成员表

27.3　窗口通用操作

27.3.1　显示窗口：win_show( )

函数原型：int win_show(HWND hwnd)
函数参数：
　　hwnd:　　HWND 句柄
成功返回 1；否则返回-1。

27.3.2　隐藏窗口：win_hide( )

函数原型：int win_hide(HWND hwnd)
函数参数：
　　hwnd:　　HWND 句柄
成功返回 1；否则返回-1。

27.3.3　关闭窗口：win_close( )

函数原型：int win_close(HWND hwnd)
函数参数：
　　hwnd:　　HWND 句柄
成功返回 1；否则返回-1。

27.3.4　窗口是否可视：win_is_visual( )

函数原型：int win_is_visual(HWND hwnd)
函数参数：
　　hwnd:　　HWND 句柄
如果可视的话，返回 1；否则返回-1。

27.3.5　允许窗口：win_enable( )

函数原型：int win_enable(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.6 禁止窗口：win_disable( )

函数原型：int win_disable(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.7 窗口是否允许：win_is_enable( )

函数原型：int win_is_enable(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
如果允许的话，返回 1；否则返回-1。

27.3.8 窗口是否禁止：win_is_disable( )

函数原型：int win_is_disable(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
如果禁止的话，返回 1；否则返回-1。

27.3.9 窗口是否为幽灵窗口：win_is_ghost( )

函数原型：int win_is_ghost(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
如果是的话，返回 1；否则返回-1。

27.3.10 极大化窗口：win_maxize( )

函数原型：int win_maxize(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.11 设置焦点窗口：win_set_focus( )

函数原型：int win_set_focus(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.12 获取焦点窗口：win_get_focus( )

函数原型：HWND win_get_focus(void)
函数参数：
    无
成功返回有效 HWND；否则返回无效 NULL。

27.3.13 是否是焦点窗口：win_is_focus( )

函数原型：int win_is_focus(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.14 设置缺省窗口：win_set_default( )

函数原型：int win_set_default(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.15 获取缺省窗口：win_get_default( )

函数原型：HWND win_get_default(void)
函数参数：
    无
成功返回有效 HWND；否则返回无效 NULL。

27.3.16 是否是缺省窗口：win_is_default( )

函数原型：int win_is_default(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.17 允许焦点缺省窗口：win_enable_default_focus( )

函数原型：int win_enable_default_focus(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.18 禁止焦点缺省窗口：win_disable_default_focus( )

函数原型：int win_disable_default_focus(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.19 窗口置前：win_bring_to_top( )

函数原型：int win_bring_to_top(const HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.20 窗口置前：win_get_hwnd_by_id( )

函数原型：HWND win_get_hwnd_by_id(const HWND hwnd, unsigned int id)
函数参数：
    hwnd: 起点 HWND 句柄
    id: 窗口 ID
成功返回有效 HWND；否则返回无效 NULL。
函数说明：
    从 hwnd 窗口开始，遍历所有子窗口，返回第一个匹配 id 的窗口句柄。

27.3.21 允许擦除窗口背景：win_enable_erase_back( )

函数原型：int win_enable_erase_back(HWND hwnd)
函数参数：
    hwnd: HWND 句柄
成功返回 1；否则返回-1。

27.3.22 禁止擦除窗口背景：win_disable_erase_back( )

函数原型：int win_disable_erase_back(HWND hwnd)
函数参数：
    hwnd: HWND 句柄

成功返回 1；否则返回-1。

27.3.23 设置窗口背景图像：win_set_background_image( )

函数原型：int win_set_background_image(HWND hwnd, GUI_BACKGROUND_IMAGE *gui_background_image)

函数参数：

hwnd: HWND 句柄

gui_background_image：GUI_BACKGROUND_IMAGE 背景图像数据指针

成功返回 1；否则返回-1。

27.3.24 清除窗口背景图像：win_clear_background_image( )

函数原型：int win_clear_background_image(HWND hwnd)

函数参数：

hwnd: HWND 句柄

成功返回 1；否则返回-1。

27.3.25 设置窗口无效窗口区域：win_invalidate_window_rect( )

函数原型：int win_invalidate_window_rect(HWND hwnd, const GUI_RECT *rect)

函数参数：

hwnd: HWND 句柄

rect: GUI_RECT 无效区域

成功返回 1；否则返回-1。

27.3.26 设置窗口无效窗口：win_invalidate( )

函数原型：int win_invalidate(HWND hwnd)

函数参数：

hwnd: HWND 句柄

成功返回 1；否则返回-1。

第二十八章　Desktop 桌面窗口

桌面窗口是一特殊窗口，由系统创建和管理，是 1 级窗口的父窗口，应用不能删除桌面窗口，但应用可以向桌面窗口发送消息、进行相关的设置。桌面窗口的 HWND 句柄是 HWND_DESKTOP。

第二十九章　Frame 窗口框架小部件

29.1　Frame 窗口框架小部件概述

一般地，Frame 作为主窗口，容纳其它的小部件。同时，Frame 支持 Windows Bar 属性。

29.2　创建 Frame 小部件

系统定义 GUI_FRAME 结构体来描述 Frame 创建专用参数：
GUI_FRAME 结构体定义如下：

```
struct  _GUI_FRAME
{
    TCHAR         text[MAX_WIN_TEXT_LEN+1];
    unsigned int  len;
};
typedef   struct     _GUI_FRAME  GUI_FRAME;
```

GUI_FRAME 结构成员描述如表 29-1 所示。

成员	描述
text	Windows Bar 文本缓冲区 MAX_WIN_TEXT_LEN 是系统定义的缺省 Windows Bar 文本长度
len	Windows Bar 文本长度

表 29-1　GUI_FRAME 结构体成员表

创建 Frame 函数声明：

HWND frame_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_FRAME *gui_frame)

如果创建 Frame 成功，返回有效的 Frame 句柄；否则返回无效的 NULL 句柄。
Frame 小部件效果图如图 29-1 所示。

图 29-1　Frame 小部件效果图

第三十章　GroupBox 分组小部件

30.1　GroupBox 分组小部件概述

GroupBox 作为分组窗口，容纳其它的小部件。

30.2　创建 GroupBox 小部件

系统定义 GUI_GROUP_BOX 结构体来描述 GroupBox 创建专用参数：
GUI_FRAME 结构体定义如下：

```
struct  _GUI_GROUP_BOX
{
    TCHAR         text[MAX_GROUP_BOX_TEXT_LEN+1];
    unsigned int  len;
    unsigned int  left_offset;
};
typedef   struct     _GUI_GROUP_BOX  GUI_GROUP_BOX;
```

GUI_GROUP_BOX 结构成员描述如表 30-1 所示。

成员	描述
text	GroupBar 文本缓冲区 MAX_GROUP_BOX_TEXT_LEN 是系统定义的缺省 GroupBar 文本长度
len	GroupBox 文本长度
left_offset	文本起始位置左偏移

表 30-1　GUI_GROUP_BOX 结构体成员表

创建 GroupBox 函数声明：

HWND　group_box_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_GROUP_BOX *gui_group_box)

如果创建 GroupBox 成功，返回有效的 GroupBox 句柄；否则返回无效的 NULL 句柄。
要求 GroupBox 先于放置其内部的小部件创建。
GroupBox 小部件效果图如图 30-1 所示。

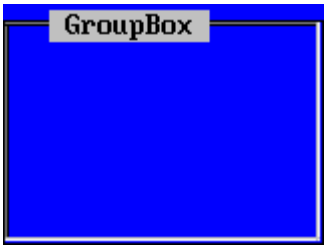

图 30-1　GroupBox 小部件效果图

30.3　GroupBox 分组小部件操作接口函数

30.3.1　获取 GroupBox 文本：group_box_get_text( )

函数原型：int　group_box_get_text(HWND hwnd, TCHAR *text, unsigned int *text_len)
函数参数：

hwnd:　　HWND 句柄
text:　　存放返回 GroupBox 文本缓冲区
text_len:　　作为输入参数时，表示缓冲区长度
　　作为输出参数时，表示返回文本长度

成功返回 1；否则返回-1。

30.3.2　设置 GroupBox 文本：group_box_set_text( )

函数原型：int　group_box_set_text(HWND hwnd, TCHAR *text, unsigned int　text_len)
函数参数：

hwnd: HWND 句柄
text: 文本缓冲区
text_len: 缓冲区文本长度

成功返回 1；否则返回-1。

第三十一章　Cell 单元小部件

31.1　Cell 单元小部件概述

系统只是绘制 Cell 单元小部件的基本外观，Cell 面板部分需要由应用绘制。这给应用提供了一个自绘小部件。

31.2　创建 Cell 小部件

系统定义 GUI_CELL 结构体来描述 Cell 创建专用参数：
GUI_CELL 结构体定义如下：

```
struct  _GUI_CELL
{
    BINT  unused;
};
typedef   struct    _GUI_CELL  GUI_CELL;
```

系统出于兼容性目的，定义 GUI_CELL 结构体，该结构体无实际意义。
创建 Cell 函数声明：

```
HWND  cell_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_CELL *gui_cell)
```

Gui_cell 参数可以为 NULL。
如果创建 Cell 成功，返回有效的 Cell 句柄；否则返回无效的 NULL 句柄。
Cell 小部件效果图如图 31-1 所示。

图 31-1　Cell 小部件效果图

第三十二章　Label 标签小部件

32.1　Label 标签小部件概述

Label 小部件提供静态文本显示空间。

32.2　创建 Label 小部件

系统定义 GUI_LABEL 结构体来描述 Label 创建专用参数：
GUI_LABEL 结构体定义如下：

```
struct  _GUI_LABEL
{
    TCHAR       text[MAX_LABEL_TEXT_LEN+1];
    unsigned int len;
    BINT         no_back_flag;
};
typedef   struct    _GUI_LABEL  GUI_LABEL;
```

GUI_LABEL 结构成员描述如表 32-1 所示。

成员	描述
text	Label 文本缓冲区 MAX_LABEL_TEXT_LEN 是系统定义的缺省 Label 文本长度
len	Label 文本长度
no_back_flag	是否绘制 Label 背景标记

表 32-1　GUI_LABEL 结构体成员表

创建 Label 函数声明：

HWND label_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_LABEL *gui_label)

如果创建 Label 成功，返回有效的 Label 句柄；否则返回无效的 NULL 句柄。
Label 小部件效果图如图 32-1 所示。

图 32-1　Label 小部件效果图

32.3　Label 小部件操作接口函数

32.3.1　获取 Label 文本：label_get_text( )

函数原型：int　label_get_text(HWND hwnd, TCHAR *text, int *text_len)
函数参数：

hwnd:	HWND 句柄
text:	存放返回 Label 文本缓冲区
text_len:	作为输入参数时，表示缓冲区长度 作为输出参数时，表示返回文本长度

成功返回 1；否则返回-1。

32.3.2　设置 Label 文本：label_set_text( )

函数原型：int　label_set_text(HWND hwnd, TCHAR *text, int text_len)
函数参数：

hwnd:	HWND 句柄
text:	文本缓冲区
text_len:	缓冲区文本长度

成功返回 1；否则返回-1。

第三十三章　PushButton 按钮小部件

33.1　PushButton 小部件概述

PushButton 小部件提供按钮功能。

33.2　创建 PushButton 小部件

系统定义 GUI_PUSH_BUTTON 结构体来描述 PushButton 创建专用参数：
GUI_PUSH_BUTTON 结构体定义如下：

```
struct  _GUI_PUSH_BUTTON
{
    TCHAR         text[MAX_PUSH_BUTTON_TEXT_LEN+1];
    unsigned int  len;
    BINT          is_ghost_flag;
};
typedef   struct     _GUI_PUSH_BUTTON  GUI_PUSH_BUTTON;
```

GUI_PUSH_BUTTON 结构成员描述如表 33-1 所示。

成员	描述
text	PushButton 文本缓冲区 MAX_PUSH_BUTTON_TEXT_LEN 是系统定义的缺省 PushButton 文本长度
len	PushButton 文本长度
is_ghost_flag	是否是幽灵窗口标记

表 33-1　GUI_PUSH_BUTTON 结构体成员表

创建 PushButton 函数声明：

HWND push_button_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_PUSH_BUTTON *gui_push_button)

如果创建 PushButton 成功，返回有效的 PushButton 句柄；否则返回无效的 NULL 句柄。
PushButton 小部件效果图如图 33-1 所示。

图 33-1　PushButton 小部件效果图

33.3　PushButton 小部件操作接口函数

33.3.1　获取 PushButton 文本：push_button_get_text( )

函数原型：int　push_button_get_text(HWND hwnd, TCHAR *text, unsigned int *text_len)
函数参数：

hwnd:	HWND 句柄
text:	存放返回 PushButton 文本缓冲区
text_len:	作为输入参数时，表示缓冲区长度 作为输出参数时，表示返回文本长度

成功返回 1；否则返回-1。

33.3.2　设置 PushButton 文本：push_button_set_text( )

函数原型：int　push_button_set_text(HWND hwnd, TCHAR *text, unsigned int text_len)
函数参数：

hwnd:	HWND 句柄
text:	文本缓冲区
text_len:	缓冲区文本长度

成功返回 1；否则返回-1。

33.3.3 是否是幽灵窗口：push_button_is_ghost( )

函数原型：int push_button_is_ghost(HWND hwnd)

函数参数：

hwnd: HWND 句柄

如果是幽灵窗口的话，返回 1；否则返回-1。

33.3.4 设置 PushButton 为幽灵窗口：push_button_set_ghost( )

函数原型：int push_button_set_ghost(HWND hwnd)

函数参数：

hwnd: HWND 句柄

成功返回 1；否则返回-1。

第三十四章　WidgetGroup 管理部件

34.1　WidgetGroup 管理部件概述

WidgetGroup 管理部件不是 GUI 小部件，而是管理多选一小部件的管理部件。首先需要创建该管理部件，其次，依次将需要连接的多选一小部件连接到该管理部件上，这样就能进行多选一的管理工作；如果不需要进行连接管理，则断开管理部件的连接，并关闭删除该管理部件。

34.2　WidgetGroup 管理部件操作接口函数

34.2.1　创建 WidgetGroup 管理部件：widget_group_create( )

函数原型：HWND　widget_group_create(HWND parent)

函数参数：

parent:　　父窗口句柄

创建成功的话，返回有效的 WidgetGroup 句柄；否则返回无效的句柄 NULL。

34.2.2　删除 WidgetGroup 管理部件：widget_group_close( )

函数原型：int　widget_group_close(HWND hwnd)

函数参数：

hwnd:　　WidgetGroup 句柄

成功返回 1，否则返回-1。

34.2.3　连接 Widget 小部件：attach_widget( )

函数原型：int　attach_widget(HWND hwnd, HWND widget_group)

函数参数：

hwnd:　　待连接的小部件句柄

widget_group:　WidgetGroup 管理部件句柄

连接成功的话，返回 1；否则返回-1。

该函数用于将多选一小部件连接到 WidgetGroup 管理部件上，以便 WidgetGroup 管理部件对多选一小部件进行多选一管理。

34.2.4　断开连接 Widget 小部件：attach_widget( )

函数原型：int　detack_widget(HWND hwnd)

函数参数：

hwnd:　　已经连接的 hwnd 小部件句柄

成功返回 1；否则返回-1。

第三十五章　RadioButton 多选一小部件

35.1　RadioButton 小部件概述

RadioButton 小部件是一个多选一小部件。在使用之前必须使用 WidgetGroup 管理部件进行分组管理。

35.2　创建 RadioButton 小部件

系统定义 GUI_RADIO_BUTTON 结构体来描述 RadioButton 创建专用参数：
GUI_RADIO_BUTTON 结构体定义如下：

```
struct  _GUI_RADIO_BUTTON
{
    BINT  radius_offset;
};
typedef   struct     _GUI_RADIO_BUTTON  GUI_RADIO_BUTTON
```

GUI_RADIOU_BUTTON 结构成员描述如表 35-1 所示。

成员	描述
radius_offset	外圆周至内填充圆周间的半径偏移

表 35-1　GUI_RADIO_BUTTON 结构体成员表

创建 RadioButton 函数声明：

HWND　　radio_button_create(HWND　parent,　GUI_COMMON_WIDGET　*gui_common_widget, GUI_RADIO_BUTTON *gui_radio_button)

如果创建 RadioButton 成功，返回有效的 RadioButton 句柄；否则返回无效的 NULL 句柄。
RadioButton 小部件选中效果图如图 35-1 所示。

图 35-1　RadioButton 小部件选中效果图

35.3　RadioButton 小部件操作接口函数

35.3.1　获取 RadioButton 状态：radio_button_get_state( )

函数原型：int　radio_button_get_state(HWND hwnd)
函数参数：
hwnd:　　HWND 句柄
成功 RadioButton 状态；
1 表示选中状态
0 表示非选中状态

35.3.2　设置 RadioButton 状态：radio_button_set_state( )

函数原型：int　radio_button_set_state(HWND hwnd, unsigned char state)
函数参数：
hwnd:　　HWND 句柄
state:　　设置状态：1 表示设置选中，0 表示设置非选中
成功返回 1；否则返回-1。

第三十六章　CheckBox 单选小部件

36.1　CheckBox 小部件概述

CheckBox 小部件是二选一小部件。

36.2　创建 CheckBox 小部件

系统定义 GUI_CHECK_BOX 结构体来描述 CheckBox 创建专用参数：
GUI_CHECK_BOX 结构体定义如下：

```
struct  _GUI_CHECK_BOX
{
    unsigned char  state;
};
typedef   struct     _GUI_CHECK_BOX  GUI_CHECK_BOX;
```

GUI_CHECK_BOX 结构成员描述如表 36-1 所示。

成员	描述
state	状态

表 36-1　GUI_CHECK_BOX 结构体成员表

创建 CheckBox 函数声明：

```
HWND   check_box_create(HWND  parent,  GUI_COMMON_WIDGET  *gui_common_widget,  GUI_CHECK_BOX *gui_check_box)
```

如果创建 CheckBox 成功，返回有效的 CheckBox 句柄；否则返回无效的 NULL 句柄。
CheckBox 小部件选中效果图如图 36-1 所示。

图 36-1　CheckBox 小部件选中效果图

36.3　CheckBox 小部件操作接口函数

36.3.1　获取 CheckBox 状态：check_box_get_state( )

函数原型：int　check_box_get_state(HWND hwnd)
函数参数：
hwnd:　　HWND 句柄
成功 CheckBox 状态；
1 表示选中状态
0 表示非选中状态

36.3.2　设置 CheckBox 状态：check_box_set_state( )

函数原型：int　check_box_set_state(HWND hwnd, unsigned char state)
函数参数：
hwnd:　　HWND 句柄
state:　　设置状态：1 表示设置选中，0 表示设置非选中
成功返回 1；否则返回-1。

第三十七章　字符定位符

37.1　字符定位符概述

字符定位符 caret 是指在应用输入时指示下次字符输入的位置。字符定位符功能属于可选择功能，需要 _LG_CARET_宏支持。

字符定位符闪烁间隔是指主循环的次数。字符定位符宽度单位是像素。

37.2　字符定位符操作接口函数

37.2.1　显示字符定位符：caret_show( )

函数原型：int caret_show(void)

返回 1。

37.2.2　隐藏字符定位符：caret_hide( )

函数原型：int caret_hide(void)

返回 1。

37.2.3　获取字符定位符位置：caret_get_position( )

函数原型：int caret_get_position(void)

返回字符定位符位置。

37.2.4　设置字符定位符位置：caret_set_position( )

函数原型：int caret_set_position(unsigned int index)

返回 1。

37.2.5　获取字符定位符闪烁间隔：caret_get_interval( )

函数原型：int caret_get_interval(void)

返回字符定位符闪烁间隔。

37.2.6　设置字符定位符闪烁间隔：caret_set_interval( )

函数原型：int caret_set_interval(unsigned int interval)

返回 1。

37.2.7　获取字符定位符宽度：caret_get_width( )

函数原型：int caret_get_interval(void)

返回字符定位符闪烁宽度。

37.2.8　设置字符定位符宽度：caret_set_width( )

函数原型：int caret_set_width(unsigned int width)

返回 1。

第三十八章　LineEdit 单行编辑小部件

38.1　LineEdit 小部件概述

LineEdit 小部件提供应用输入文本窗口。

38.2　创建 LineEdit 小部件

系统定义 GUI_LINE_EDIT 结构体来描述 LineEdit 创建专用参数：
GUI_LINE_EDIT 结构体定义如下：

```
struct  _GUI_LINE_EDIT
{
    TCHAR         text[MAX_LINE_EDIT_TEXT_LEN+1];
    unsigned int  len;
};
typedef   struct     _GUI_LINE_EDIT  GUI_LINE_EDIT;
```

GUI_LINE_EDIT 结构成员描述如表 38-1 所示。

成员	描述
Text	LineEdit 初始文本 MAX_LINE_EDIT_TEXT_LEN 是系统定义的缺省 LineEdit 文本长度
Len	LineEdit 文本长度

表 38-1　GUI_LINE_EDIT 结构体成员表

创建 LineEdit 函数声明：

HWND　line_edit_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_LINE_EDIT *gui_line_edit)

如果创建 LineEdit 成功，返回有效的 LineEdit 句柄；否则返回无效的 NULL 句柄。
LineEdit 小部件效果图如图 38-1 所示。

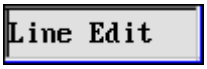

图 38-1　LineEdit 小部件效果图

38.3　LineEdit 小部件操作接口函数

38.3.1　获取 LineEdit 文本：line_edit_get_text( )

函数原型：int　line_edit_get_text(HWND hwnd, TCHAR *text, unsigned int *text_len)
函数参数：

hwnd:　HWND 句柄
text:　存放返回 LineEdit 文本缓冲区
text_len:　作为输入参数时，表示缓冲区长度
　　作为输出参数时，表示返回文本长度

成功返回 1；否则返回-1。

38.3.2　设置 Line 文本：line_edit_set_text( )

函数原型：int　line_edit_set_text(HWND hwnd, TCHAR *text, unsigned int text_len)
函数参数：

hwnd:　HWND 句柄
text:　文本缓冲区
text_len:　缓冲区文本长度

成功返回 1；否则返回-1。

第三十九章　ListBox 列表部件

39.1　ListBox 小部件概述

ListBox 小部件提供应用列表显示功能，同时能选中列表项。

39.2　创建 ListBox 小部件

系统定义 GUI_LIST_BOX 结构体来描述 ListBox 创建专用参数：
GUI_LIST_BOX 结构体定义如下：

```
struct  _GUI_LIST_BOX
{
    unsigned int     multi_flag;
};
typedef   struct     _GUI_LIST_BOX  GUI_LIST_BOX;
```

GUI_LIST_BOX 结构成员描述如表 39-1 所示。

成员	描述
multi_flag	多选标志

表 39-1　GUI_LIST_BOX 结构体成员表

创建 ListBox 函数声明：

```
HWND    list_box_create(HWND  parent,  GUI_COMMON_WIDGET  *gui_common_widget,  GUI_LIST_BOX
*gui_list_box)
```

如果创建 ListBox 成功，返回有效的 ListBox 句柄；否则返回无效的 NULL 句柄。
ListBox 小部件效果图如图 39-1 所示。

图 39-1　ListBox 小部件效果图

39.3　ListBox 小部件操作接口函数

39.3.1　插入项目条：list_box_insert_item( )

函数原型：int　list_box_insert_item(HWND hwnd, int index, TCHAR *text, unsigned int text_len)
函数参数：

hwnd:	HWND 句柄
index:	插入位置索引号（从 0 开始）
text:	项目条文本缓冲区
text_len:	项目条文本长度

成功返回 1；否则返回-1。

39.3.2　增加项目条：list_box_add_item( )

函数原型：int　list_box_add_item(HWND hwnd, TCHAR *text, unsigned int text_len)
函数参数：

hwnd:	HWND 句柄
text:	项目条文本缓冲区
text_len:	项目条文本长度

成功返回 1；否则返回-1。
在 ListBox 最后处增加一条项目。

39.3.3 获取项目条文本：list_box_get_item( )

函数原型：int list_box_get_item(HWND hwnd, unsigned int index, TCHAR *text, unsigned int *text_len)

函数参数：

hwnd:	HWND 句柄
index:	项目条索引号
text:	文本缓冲区
text_len:	作为输入参数时，表示缓冲区长度 作为输出参数时，表示返回文本长度

成功返回 1；否则返回-1。

39.3.4 删除项目条：list_box_delete_item( )

函数原型：int list_box_delete_item(HWND hwnd, unsigned int index)

函数参数：

hwnd:	HWND 句柄
index:	项目条索引号

成功返回 1；否则返回-1。

39.3.5 设置项目条选择模式：list_box_set_select_mode( )

函数原型：int list_box_set_select_mode(HWND hwnd, unsigned int mode)

函数参数：

hwnd:	HWND 句柄
mode:	选择模式：1 表示多行选择，0 表示单行选择

成功返回 1；否则返回-1。

39.3.6 获取项目条选择模式：list_box_get_select_mode( )

函数原型：int list_box_get_select_mode(HWND hwnd, unsigned int *mode)

函数参数：

hwnd:	HWND 句柄
mode:	返回选择模式指针

成功返回 1；否则返回-1。

39.3.7 设置选择项目条：list_box_set_selected_index( )

函数原型：int list_box_set_selected_index(HWND hwnd, unsigned int index)

函数参数：

hwnd:	HWND 句柄
index:	所设置的项目索引号

成功返回 1；否则返回-1。

39.3.8 是否选择了某个项目条：list_box_is_selected_index( )

函数原型：int list_box_is_selected_index(HWND hwnd, unsigned int index)

函数参数：

hwnd:	HWND 句柄
index:	项目索引号

如果 index 被选择了，则返回 1；否则返回-1。

39.3.9 获取选择项目条索引号：list_box_get_selected_index( )

函数原型：int list_box_get_selected_index(HWND hwnd, unsigned int start_index, int *index)

函数参数：

hwnd:	HWND 句柄
start_index:	开始计数的索引号
index:	所选择的项目索引号指针

如果存在被选择的条目，那么返回 1；否则返回-1。

对于多选模式，返回第一个匹配的索引号。

39.3.10 设置高亮度项目条：list_box_set_lighted_index( )

函数原型：int list_box_set_lighted_index(HWND hwnd, unsigned int index)
函数参数：
hwnd: HWND 句柄
index: 所设置的项目索引号
成功返回 1；否则返回-1。

39.3.11 是否高亮度了某个项目条：list_box_is_lighted_index( )

函数原型：int list_box_is_lighted_index(HWND hwnd, unsigned int index)
函数参数：
hwnd: HWND 句柄
index: 项目索引号
如果 index 被高亮度了，则返回 1；否则返回-1。

39.3.12 获取高亮度项目条索引号：list_box_get_lighted_index( )

函数原型：int list_box_get_lighted_index(HWND hwnd, unsigned int start_index, int *index)
函数参数：
hwnd: HWND 句柄
start_index: 开始计数的索引号
index: 所高亮度的项目索引号指针
如果存在被高亮度的条目，那么返回 1；否则返回-1。

39.3.13 获取项目条总数：list_box_get_item_counter( )

函数原型：int list_box_get_item_counter(HWND hwnd, unsigned int *counter)
函数参数：
hwnd: HWND 句柄
counter: 存放项目条总数指针
成功返回 1；否则返回-1。

39.3.14 ListBox 是否只读：list_box_is_read_only( )

函数原型：int list_box_is_read_only(HWND hwnd)
函数参数：
hwnd: HWND 句柄
如果 ListBox 是只读的，那么返回 1；否则返回-1。

39.3.15 设置 ListBox 只读属性：list_box_set_read_only( )

函数原型：int list_box_set_read_only(HWND hwnd, unsigned int read_only)
函数参数：
hwnd: HWND 句柄
read_only: 只读属性：1 表示只读，0 表示非只读属性
成功返回 1；否则返回-1。

第四十章　ComBox 组合框小部件

40.1　ComBox 组合框小部件概述

ComBox 组合框小部件主要由 LineEdit 小部件和 ListBox 小部件组成，具备二者的属性和操作方法。其弹出方向可以下拉，也可以下拉，弹出框长度由应用定义。

40.2　创建 ComBox 小部件

系统定义 GUI_COM_BOX 结构体来描述 ComBox 创建专用参数：
GUI_COM_BOX 结构体定义如下：

```
struct  _GUI_COM_BOX
{
    BUINT        line_edit_style;
    BUINT        line_edit_ext_style;
    BUINT        list_box_style;
    BUINT        list_box_ext_style;
    BUINT        open_dir;
    unsigned int open_len;
};
typedef   struct     _GUI_COM_BOX  GUI_COM_BOX;
```

GUI_COM_BOX 结构成员描述如表 40-1 所示。

成员	描述
line_edit_style	LineEdit 式样
line_edit_ext_style	LineEdit 扩展式样
list_box_style	ListBox 式样
list_box_ext_style	ListBox 扩展式样
open_dir	弹出方向： 0：下拉 1：上拉
open_len	弹出框长度

表 40-1　GUI_COM_BOX 结构体成员表

创建 ComBox 函数声明：

HWND　com_box_create(HWND　parent,　GUI_COMMON_WIDGET　*gui_common_widget,　GUI_COM_BOX *gui_com_box)

如果创建 ComBox 成功，返回有效的 ComBox 句柄；否则返回无效的 NULL 句柄。
ComBox 小部件效果图如图 40-1 所示。

图 40-1　ComBox 小部件效果图

40.3　ComBox 小部件操作接口函数

40.3.1　获取 LineEdit 句柄：com_box_get_line_edit_hwnd( )

函数原型：int　com_box_get_line_edit_hwnd(HWND hwnd, HWND *hline_edit)
函数参数：

hwnd:　　ComBox 句柄
hline_edit:　　存放 LineEdit 句柄

成功返回 1；否则返回-1。
应用获取 LineEdit 句柄后，可以通过该句柄使用 LineEdit 接口函数来操作。

40.3.2　获取 ListBox 句柄：com_box_get_list_box_hwnd( )

函数原型：int com_box_get_list_box_hwnd(HWND hwnd, HWND *hlist_box)
函数参数：
    hwnd: ComBox 句柄
    hlist_box: 存放 ListBox 句柄
成功返回 1；否则返回-1。
应用获取 ListBox 句柄后，可以通过该句柄使用 ListBox 接口函数来操作。

40.3.3 获取 ComBox 状态：com_box_get_state( )

函数原型：int com_box_get_state(HWND hwnd, unsigned int *opened)
函数参数：
    hwnd: ComBox 句柄
    opened: 存放 ListBox 状态指针
成功返回 1；否则返回-1。
如果是弹出状态，则*opened 为 1，否则为 0。

40.3.4 设置 ComBox 状态：com_box_set_state( )

函数原型：int com_box_set_state(HWND hwnd, unsigned int opened)
函数参数：
    hwnd: ComBox 句柄
    opened: 设置 ComBox 状态：1 表示弹出状态，0 表示关闭状态。
成功返回 1；否则返回-1。

40.3.5 设置 ComBox 所显示的文本索引号：com_box_set_index_item( )

函数原型：int com_box_set_index_item(HWND hwnd, unsigned int index)
函数参数：
    hwnd: ComBox 句柄
    index: ListBox 中索引号。
成功返回 1；否则返回-1。

以下句柄为 LineEdit 句柄

40.3.6 获取显示文本：com_box_line_edit_get_text( )

函数原型：int com_box_line_edit_get_text(HWND hwnd, TCHAR *text, unsigned int *text_len)
函数参数：
    hwnd: LineEdit 句柄
    text: 文本缓冲区
    text_len: 作为输入参数时，表示缓冲区长度
                    作为输出参数时，表示返回文本长度
成功返回 1；否则返回-1。

40.3.7 设置显示文本：com_box_line_edit_set_text( )

函数原型：int com_box_line_edit_set_text(HWND hwnd, TCHAR *text, unsigned int text_len)
函数参数：
    hwnd: LineEdit 句柄
    text: 文本缓冲区
    text_len: 表示缓冲区长度
成功返回 1；否则返回-1。

以下句柄为 ListBox 句柄

40.3.9 插入项目条：com_box_list_box_insert_item( )

函数原型：int com_box_list_box_insert_item(HWND hwnd, int index, TCHAR *text, unsigned int text_len)
函数参数：
    hwnd: ListBox 句柄
    index: 插入位置索引号（从 0 开始）

text: 项目条文本缓冲区
text_len: 项目条文本长度
成功返回 1；否则返回-1。

40.3.10 增加项目条：com_box_list_box_add_item( )

函数原型：int com_box_list_box_add_item(HWND hwnd, TCHAR *text, unsigned int text_len)
函数参数：
hwnd: ListBox 句柄
text: 项目条文本缓冲区
text_len: 项目条文本长度
成功返回 1；否则返回-1。
在 ListBox 最后处增加一条项目。

40.3.11 获取项目条文本：com_box_list_box_get_item( )

函数原型：int com_box_list_box_get_item(HWND hwnd, unsigned int index, TCHAR *text, unsigned int *text_len)
函数参数：
hwnd: ListBox 句柄
index: 项目条索引号
text: 文本缓冲区
text_len: 作为输入参数时，表示缓冲区长度
作为输出参数时，表示返回文本长度
成功返回 1；否则返回-1。

40.3.12 删除项目条：com_box_list_box_delete_item( )

函数原型：int com_box_list_box_delete_item(HWND hwnd, unsigned int index)
函数参数：
hwnd: ListBox 句柄
index: 项目条索引号
成功返回 1；否则返回-1。

40.3.13 设置项目条选择模式：com_box_list_box_set_select_mode( )

函数原型：int com_box_list_box_set_select_mode(HWND hwnd, unsigned int mode)
函数参数：
hwnd: ListBox 句柄
mode: 选择模式：1 表示多行选择，0 表示单行选择
成功返回 1；否则返回-1。

40.3.14 获取项目条选择模式：com_box_list_box_get_select_mode( )

函数原型：int com_box_list_box_get_select_mode(HWND hwnd, unsigned int *mode)
函数参数：
hwnd: ListBox 句柄
mode: 返回选择模式指针
成功返回 1；否则返回-1。

40.3.15 设置选择项目条：com_box_list_box_set_selected_index( )

函数原型：int com_box_list_box_set_selected_index(HWND hwnd, unsigned int index)
函数参数：
hwnd: ListBox 句柄
index: 所设置的项目索引号
成功返回 1；否则返回-1。

40.3.16 是否选择了某个项目条：com_box_list_box_is_selected_index( )

函数原型：int com_box_list_box_is_selected_index(HWND hwnd, unsigned int index)

函数参数：
hwnd: ListBox 句柄
index: 项目索引号
如果 index 被选择了，则返回 1；否则返回-1。

40.3.17 获取选择项目条索引号：com_box_list_box_get_selected_index( )

函数原型：int com_box_list_box_get_selected_index(HWND hwnd, unsigned int start_index, int *index)
函数参数：
hwnd: ListBox 句柄
start_index: 开始计数的索引号
index: 所选择的项目索引号指针
如果存在被选择的条目，那么返回 1；否则返回-1。
对于多选模式，返回第一个匹配的索引号。

40.3.18 设置高亮度项目条：com_box_list_box_set_lighted_index( )

函数原型：int com_box_list_box_set_lighted_index(HWND hwnd, unsigned int index)
函数参数：
hwnd: ListBox 句柄
index: 所设置的项目索引号
成功返回 1；否则返回-1。

40.3.19 是否高亮度了某个项目条：com_box_list_box_is_lighted_index( )

函数原型：int com_box_list_box_is_lighted_index(HWND hwnd, unsigned int index)
函数参数：
hwnd: ListBox 句柄
index: 项目索引号
如果 index 被高亮度了，则返回 1；否则返回-1。

40.3.20 获取高亮度项目条索引号：com_box_list_box_get_lighted_index( )

函数原型：int com_box_list_box_get_lighted_index(HWND hwnd, unsigned int start_index, int *index)
函数参数：
hwnd: ListBox 句柄
start_index: 开始计数的索引号
index: 所高亮度的项目索引号指针
如果存在被高亮度的条目，那么返回 1；否则返回-1。

40.3.21 获取项目条总数：com_box_list_box_get_item_counter( )

函数原型：int com_box_list_box_get_item_counter(HWND hwnd, unsigned int *counter)
函数参数：
hwnd: ListBox 句柄
counter: 存放项目条总数指针
成功返回 1；否则返回-1。

40.3.22 ListBox 是否只读：com_box_list_box_is_read_only( )

函数原型：int com_box_list_box_is_read_only(HWND hwnd)
函数参数：
hwnd: ListBox 句柄
如果 ListBox 是只读的，那么返回 1；否则返回-1。

40.3.23 设置 ListBox 只读属性：com_box_list_box_set_read_only( )

函数原型：int list_box_set_read_only(HWND hwnd, unsigned int read_only)
函数参数：
hwnd: ListBox 句柄
read_only: 只读属性：1 表示只读，0 表示非只读属性

成功返回 1；否则返回-1。

第四十一章　ProgressBar 进度条小部件

41.1　ProgressBar 进度条小部件概述

ProgressBar 小部件主要用来直观表示进度。

41.2　创建 ProgressBar 小部件

系统定义 GUI_PROGRESS_BAR 结构体来描述 ProgressBar 创建专用参数：
GUI_PROGRESS_BAR 结构体定义如下：

```
struct  _GUI_PROGRESS_BAR
{
    int     min_value;
    int     max_value;
    int     current_value;
    BUINT   is_display_text;
    BUINT   display_style;
    BUINT   decimal_digits;
};
typedef   struct     _GUI_PROGRESS_BAR  GUI_PROGRESS_BAR;
```

GUI_PROGRESS_BAR 结构成员描述如表 41-1 所示。

成员	描述
min_value	最小值
max_value	最大值
current_value	当前值
is_display_text	是否显示文本
display_style	显示式样
decimal_digits	小数位数 或者表示百分比放大系数

表 41-1　GUI_PROGRESS_BAR 结构体成员表

创建 ProgressBar 函数声明：

HWND progress_bar_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_PROGRESS_BAR *gui_progress_bar)

如果创建 ProgressBar 成功，返回有效的 ProgressBar 句柄；否则返回无效的 NULL 句柄。
ProgressBar 小部件水平显示效果图如图 41-1 所示。

图 41-1　ProgressBar 小部件效果图

41.3　ProgressBar 小部件操作接口函数

41.3.1　获取当前值：progress_bar_get_value( )

函数原型：int　progress_bar_get_value(HWND hwnd, int *value)
函数参数：

hwnd:　　ProgressBar 句柄
value:　　存放放回值指针

成功返回 1；否则返回-1。

41.3.2　获取当前百分比值：progress_bar_get_percent( )

函数原型：nt　progress_bar_get_percent(HWND hwnd, unsigned int *percent)
函数参数：

hwnd:　　ProgressBar 句柄
percent:　　存放返回百分比值指针

成功返回 1；否则返回-1。

41.3.3 设置当前值：progress_bar_set_value( )

函数原型：int progress_bar_set_value(HWND hwnd, int value)

函数参数：

hwnd: ProgressBar 句柄

value: 设置值

成功返回 1；否则返回-1。

41.3.4 设置当前百分比值：progress_bar_set_percent( )

函数原型：int progress_bar_set_percent(HWND hwnd, unsigned int percent)

函数参数：

hwnd: ProgressBar 句柄

percent: 百分比值

成功返回 1；否则返回-1。

第四十二章　SliderBar 滑杆小部件

42.1　SliderBar 滑竿小部件概述

SliderBar 滑竿小部件直观表示设置数值。

42.2　创建 SliderBar 小部件

系统定义 GUI_SLIDER_BAR 结构体来描述 SliderBar 创建专用参数：
GUI_SLIDER_BAR 结构体定义如下：

```
struct  _GUI_SLIDER_BAR
{
            int  min_value;
            int  max_value;
            int  current_value;
    unsigned int  step_value;
    BUINT   decimal_digits;
    BUINT   ruler_height;
    BUINT   slot_height;
    BUINT   tick_width;
};
typedef   struct    _GUI_SLIDER_BAR  GUI_SLIDER_BAR;
```

GUI_SLIDER_BAR 结构成员描述如表 42-1 所示。

成员	描述
min_value	最小值
max_value	最大值
current_value	当前值
step_value	步长值
decimal_digits	小数位数 或者表示百分比放大系数
ruler_height	滑尺高度
slot_height	滑槽高度
tick_width	滑杆宽度

表 42-1　GUI_SLIDER_BAR 结构体成员表

创建 SliderBar 函数声明：

HWND　slider_bar_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_SLIDER_BAR *gui_slider_bar)

如果创建 SliderBar 成功，返回有效的 SliderBar 句柄；否则返回无效的 NULL 句柄。
SliderBar 小部件水平显示效果图如图 42-1 所示。

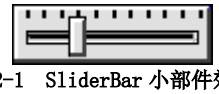

图 42-1　SliderBar 小部件效果图

42.3　SliderBar 小部件操作接口函数

42.3.1　获取当前值：slider_bar_get_value( )

函数原型：int　slider_bar_get_value(HWND hwnd, int *value)
函数参数：

hwnd:　　SliderBar 句柄
value:　　存放放回值指针

成功返回 1；否则返回-1。

42.3.2　获取当前百分比值：slider_bar_get_percent( )

函数原型：nt slider_bar_get_percent(HWND hwnd, unsigned int *percent)
函数参数：
hwnd: SliderBar 句柄
percent: 存放返回百分比值指针
成功返回 1；否则返回-1。

42.3.3 设置当前值：slider_bar_set_value( )

函数原型：int slider_bar_set_value(HWND hwnd, int value)
函数参数：
hwnd: SliderBar 句柄
value: 设置值
成功返回 1；否则返回-1。

42.3.4 设置当前百分比值：slider_bar_set_percent( )

函数原型：int slider_bar_set_percent(HWND hwnd, unsigned int percent)
函数参数：
hwnd: SliderBar 句柄
percent: 百分比值
成功返回 1；否则返回-1。

第四十三章　Image 图像小部件

43.1　Image 图像小部件概述

Image 图像小部件主要显示图像。包括 Bitmap 图像、Icon 图像、Gif 动画。

43.2　创建 Image 小部件

系统定义 GUI_IMAGE 结构体来描述 Image 创建专用参数：
GUI_IMAGE 结构体定义如下：

```
struct  _GUI_IMAGE
{
    BINT          image_type;
    BINT          image_align;
    unsigned int  frame_id;
    const void   *pimage;
};
typedef   struct     _GUI_IMAGE  GUI_IMAGE;
```

GUI_IMAGE 结构成员描述如表 43-1 所示。

成员	描述
image_type	图像类型
image_align	图像对齐方式
frame_id	系统保留
pimage	图像数据

表 43-1　GUI_IMAGE 结构体成员表

创建 Image 函数声明：

```
HWND  image_create(HWND parent, GUI_COMMON_WIDGET *gui_common_widget, GUI_IMAGE *gui_image)
```

如果创建 Image 成功，返回有效的 Image 句柄；否则返回无效的 NULL 句柄。
Image 小部件水平显示效果图如图 43-1 所示。

图 43-1　Image 小部件效果图

43.3　Image 小部件操作接口函数

43.3.1　获取当前图像：image_get_image( )

函数原型：int　image_get_image(HWND hwnd, GUI_IMAGE *gui_image)
函数参数：

hwnd:　　Image 句柄
gui_image:　　存放返回图像指针

成功返回 1；否则返回-1。

43.3.2　设置图像：image_set_image( )

函数原型：int　image_set_image(HWND hwnd, GUI_IMAGE *gui_image)
函数参数：

hwnd: Image 句柄
gui_image: 图像指针

成功返回 1；否则返回-1。

第四十四章　Windows 演示程序

Windows 演示程序位于 examples/Windows 目录下。主要演示 Windows 基本功能，子目录名称一般能表示演示的主题。

图 44-1 演示 Widget 背景图像的使用。

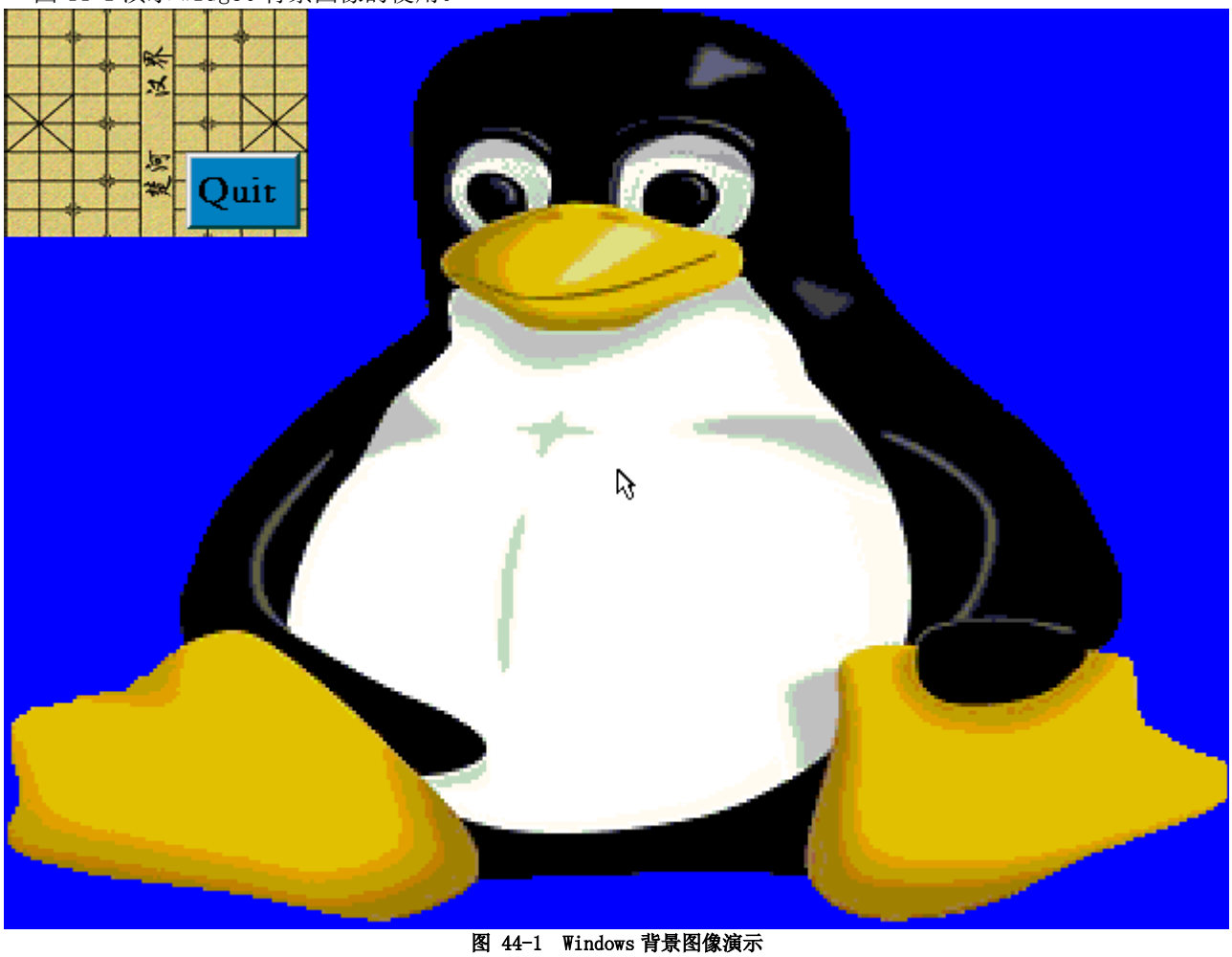

图 44-1　Windows 背景图像演示

图 44-2 演示 Image 小部件显示 Bitmap 图像。

图 44-2　Windows Bitmap 演示

图 44-3 演示 Image 小部件播放 Gif 动画。

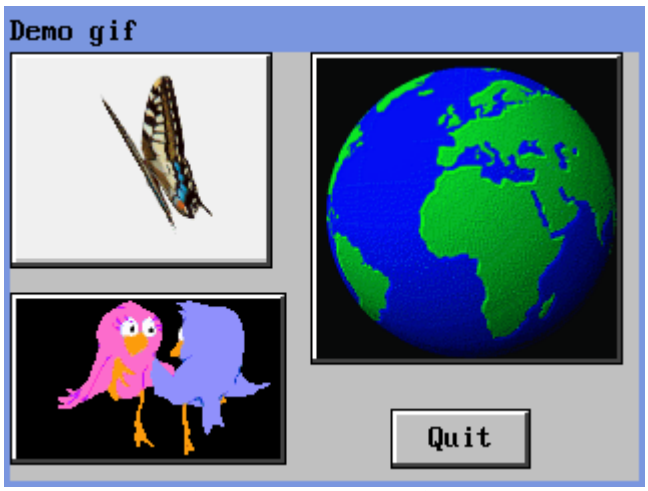

图 44-3 Windows Gif 演示

图 44-4 演示 Cell 小部件自绘功能。

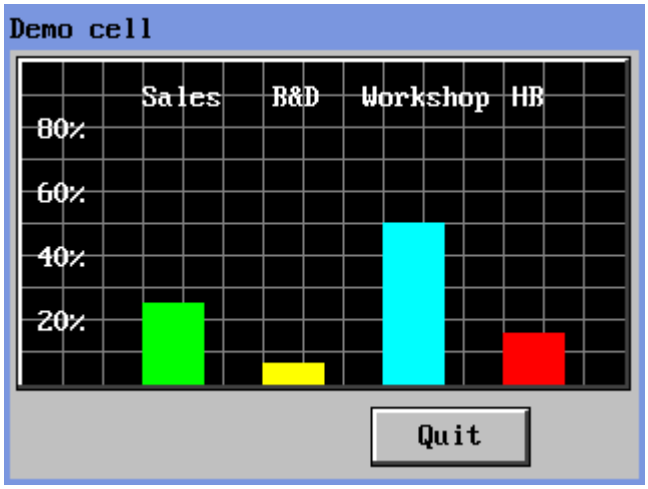

图 44-4 Cell 演示

图 44-5 演示幽灵小部件的使用。

图 44-5 Windows 幽灵小部件演示

图 44-6 演示 Windows 小部件综合使用。

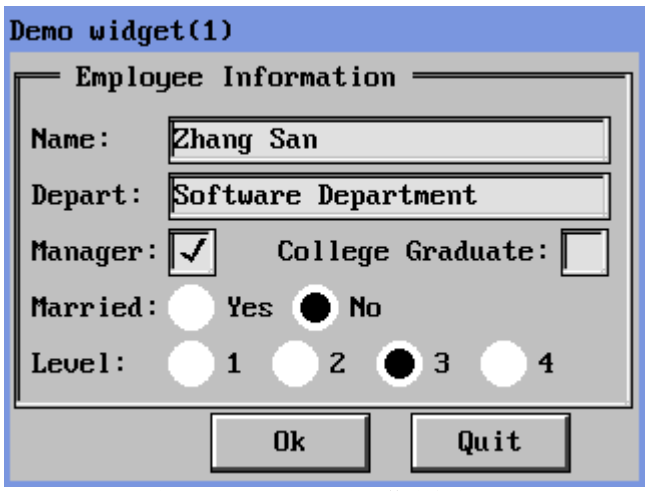

图 44-6 Windows 小部件综合演示

图 44-7 演示 Windows 小部件综合使用。

图 44-7 Windows 小部件综合演示

图 44-8 演示 Windows 小部件综合使用。

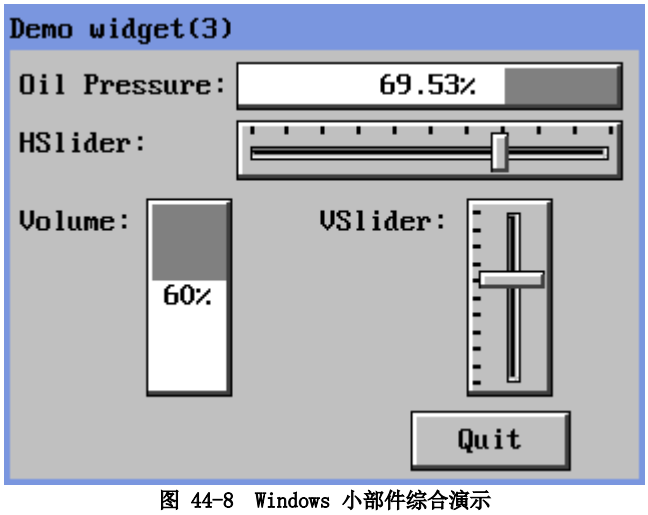

图 44-8 Windows 小部件综合演示

图 44-9 演示 Windows 多窗口使用。

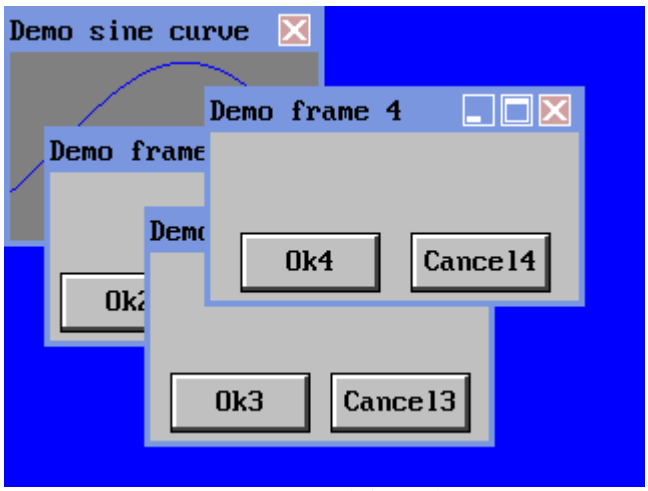

图 44-9 Windows 多窗口综合演示