Mini6410 上移植 SDL 游戏的详细步骤

2011-01-11

(本手册适用于 Mini6410/Tiny6410)

简　介

本手册由广州友善之臂计算机科技有限公司(简称“友善之臂”)创建和维护，并作为标准用户手册的一个补充，仅供嵌入式爱好者学习参考使用，友善之臂目前并不对本手册的内容提供任何解释和解答服务，用户可以在论坛中反馈你所遇到的问题和疑问，我们将在以后的更新中修正或者采纳您的建议，本手册主要以首页日期为版本标志。

本手册由友善之臂软件开发工程师编写制作，以 Mini6410 和 Tiny6410 作为开发平台，通过讲解《仙剑奇侠传》和《超级马里奥战争(Super Mario War)》这两个游戏的移植步骤，实例演示如何将基于 SDL 编写的游戏移植到 Mini6410 上运行，非常适合喜欢动手的用户进行练习和参考，用户可以参考此文档，移植更多其它的 SDL 接口的游戏到 Mini6410 与大家分享，在学习的同时享受游戏所带来的乐趣。

Mini6410 是一款十分精致的低价高品质一体化 ARM11 开发板，由广州友善之臂设计、生产和发行销售。它采用三星 S3C6410 作为主处理器，在设计上承袭了 Mini2440“精于心，简于形”的风格，而且布局更加合理，接口更加丰富，十分适用于开发 MID、汽车电子、工业控制、导航系统、媒体播放等终端设备；也可适用于 高校教学、嵌入式培训、个人研究学习和 DIY 等。

Tiny6410 是由友善之臂出品的一款以 ARM11 芯片(三星 S3C6410)作为主处理器的嵌入式核心板，它采用高密度 6 层板设计，尺寸为 64 x 50mm，它集成了 128M DDR RAM，256M/1GB SLC Nand Flash 存储器，采用 5V 供电，在板实现 CPU 必需的各种核心电压转换，还带有专业复位芯片，通过 2.0mm 间距的排针，引出各种常见的接口资源，以供不打算自行设计 CPU 板的开发者进行快捷的二次开发使用。

因为 Mini6410 和 Tiny6410 的硬件接口和资源都是基本相同的，所以本手册完全适用于 Tiny6410 开发板。

为了方便用户，本手册的所用到的软件包放在光盘 A 的“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\源代码”目录下，用户可以不需要自己去下载。

在移植之前需要想先体验一下效果的用户，光盘上提供了现成的可执行文件，存放在以下目录下：“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\Bin”，使用方法请参考文件夹内的 readme.txt 说明文件,同时也提供了现成可烧写的 ROM，位于光盘 B 的目录 “Images\Games” 下。

友善之臂公司网址：http://www.arm9.net

本手册由ARM9 之家论坛(http://www.arm9home.net)发布，转载请注明出处，手册内难免有遗漏和不足之处，欢迎大家提出宝贵意见，请发邮件至：qt_friendlyarm@163.com。

本手册内容参考了 mini6410 用户 bluedrum 大侠编写的关于仙剑移植的博客，文章链接：
http://blog.chinaunix.net/u3/105675/showart_2325128.html

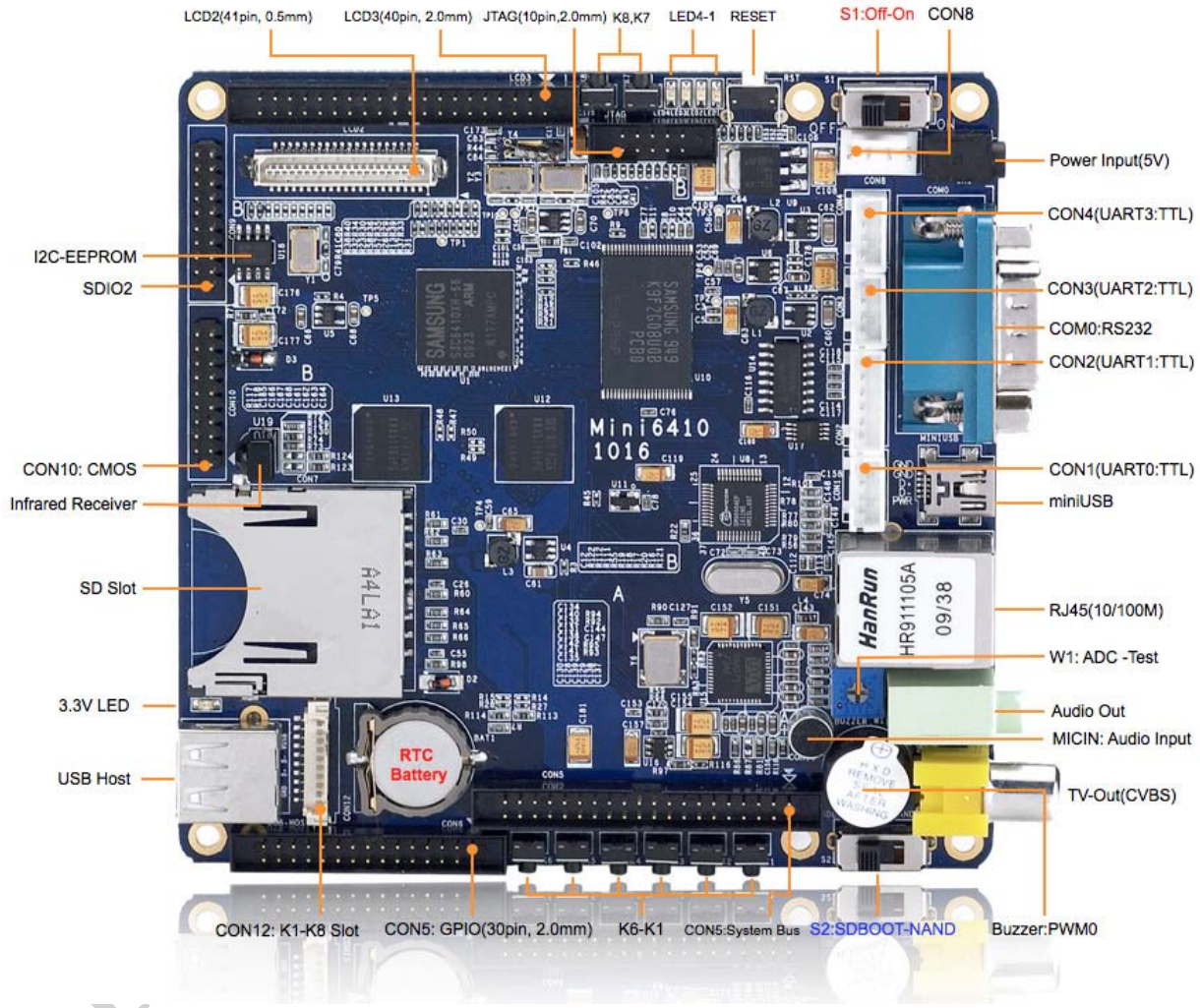
更新说明

2011-01-11	本手册第一次发布

目　录

第一章　mini6410 和tiny6410 开发板简介

1.1　Mini6410 开发板



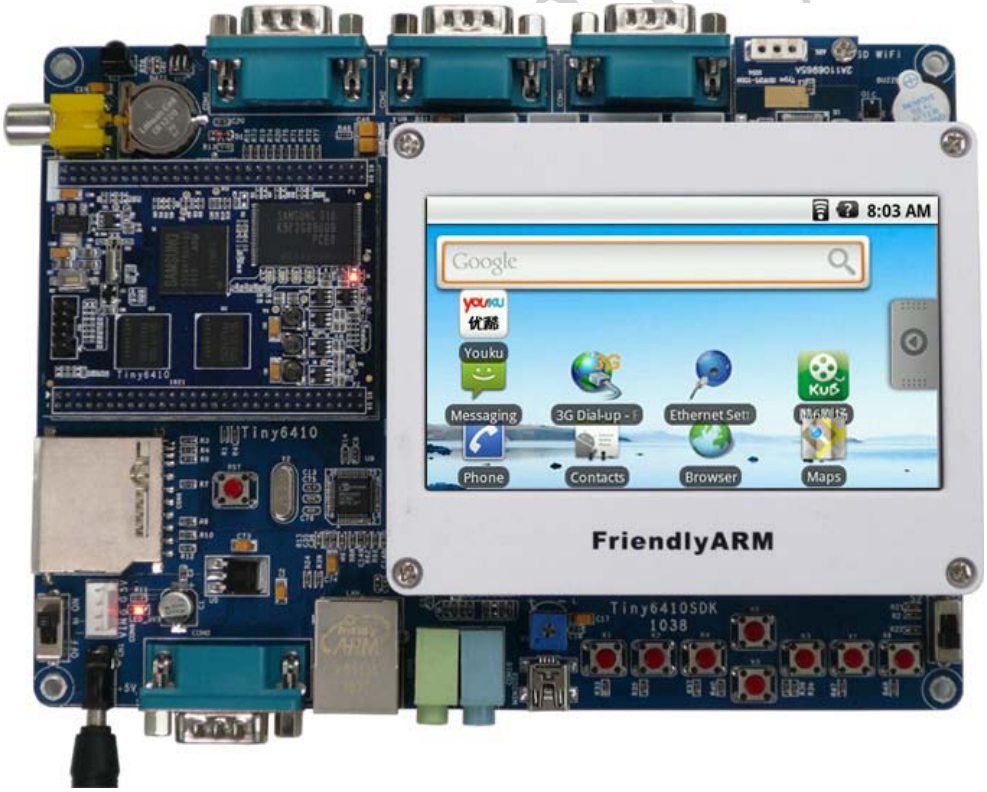
Mini6410 是一款十分精致的低价高品质一体化 ARM11 开发板，由广州友善之臂设计、生产和发行销售。它采用三星 S3C6410 作为主处理器，在设计上承袭了 Mini2440“精于心，简于形”的风格，而且布局更加合理，接口更加丰富，十分适用于开发 MID、汽车电子、工业控制、导航系统、媒体播放等终端设备；也可适用于高校教学、嵌入式培训、个人研究学习和 DIY 等。

具体而言，Mini6410 具有双 LCD 接口、4 线电阻触摸屏接口、100M 标准网络接口、标准 DB9 五线串口、Mini USB 2.0-OTG 接口、USB Host 1.1、3.5mm 音频输出口、在板麦克风、标准 TV-OUT 接口、弹出式 SD 卡座、红外接收等常用接口；另外还引出 4 路 TTL 串口、CMOS Camera 接口、40pin 总线接口、30pin GPIO 接口(可复用为 SPI、I2C、中断等，另含 3 路 ADC、1 路 DAC)、SDIO2 接口(可接 SD WiFi)、10pin Jtag 接口等；在板的还有蜂鸣器、I2C-EEPROM、备份电池、AD 可调电阻、8 按键(可引出)、4LED 等；所有这些，都极大地方便了开发者的评 估和使用，再加上我们按照 Mini6410 尺寸专门定制的 4.3″LCD 模块，真正做到让您“一手掌握所有”！

我们还充分地发挥了 6410 支持 SD 卡启动这一特性，使用我们精心研制的 superboot，无需连接电脑，只要把目标文件拷贝到 SD 卡中(可支持大于 2G 的高速大容量卡)，你就可以在开发板上极快极简单地自动安装或运行各种嵌入式系统(WindowsCE6/Linux/Android/Ubuntu 等)；甚至无需烧写，就可以在 SD 卡上直接运行它们，这一切，简直太酷了！

要了解 Mini6410 开发板的详细信息，请访问：http://www.arm9.net/mini6410-feature.asp

1.2 Tiny6410 开发板



Tiny6410 是一款以 ARM11 芯片(三星 S3C6410)作为主处理器的嵌入式核心 板，该 CPU 基于 ARM1176JZF-S 核设计，内部集成了强大的多媒体处理单元，支持 Mpeg4, H.264/H.263 等格式的视频文件硬件编解码，可同时输出至 LCD 和 TV 显示；它还并带有 3D 图形硬件加速器，以实现 OpenGL ES 1.1 & 2.0 加速渲染，另外它还支持 2D 图形图像的平滑缩放，翻转等操作。

Tiny6410 采用高密度 6 层板设计，尺寸为 64 x 50mm，它集成了 128M DDR RAM，256M/1GB SLC Nand Flash 存储器，采用 5V 供电，在板实现 CPU 必需的各种核心电压转换，还带有专业复位芯片，通过 2.0mm 间距的排针，引出各种常见的接口资源，以供 不打算自行设计 CPU 板的开发者进行快捷的二次开发使用。

Tiny6410SDK 是采用 Tiny6410 核心板的一款参考设计底板，它主要帮助开发者以此为参考进行核心板的功能验证以及扩展开发。该底板具有三 LCD 接口、4 线电阻触摸屏接口、100M 标准网络接口、标准 DB9 五线串口、Mini USB 2.0 接口、USB Host 1.1、3.5mm 音频输入输出口、标准 TV-OUT 接口、SD 卡座、红外接收等常用接口；另外还引出 4 路 TTL 串口，另 1 路 TV-OUT、SDIO2 接 口(可接 SD WiFi)接口等；在板的还有蜂鸣器、I2C-EEPROM、备份电池、AD 可调电阻、8 个中断式按键等。

在布局上安排上，我们尽量考虑把常用尺寸的 LCD 模块能够固定在底板上，比如 3.5”，4.3”LCD，7”LCD 等，这样用户在使用时不至于把各种电线搅在一起，更增加了开发套件的便携性。

我们还充分地发挥了 6410 支持 SD 卡启动这一特性，使用我们精心研制的 Superboot，无需连接电脑，只要把目标文件拷贝到 SD 卡中(可支持高达 32G 的高速大容量卡)，你就可以在开发板上极快极简单地自动安装各种嵌入式系统(WindowsCE6/Linux/Android/Ubuntu /uCos2 等)；甚至无需烧写，就可以在 SD 卡上直接运行它们！

要了解 Tiny6410 开发板的详细信息，请访问：http://www.arm9.net/tiny6410.asp

第二章　什么是SDL?

SDL（Simple DirectMedia Layer）是一套开放源代码的跨平台多媒体开发库，使用 C 语言写成。SDL 提供了数种控制图像、声音、输出入的函数，让开发者只要用相同或是相似的代码就可以开发出跨多个平台（Linux、Windows、Mac OS X 等）的应用软件。目前 SDL 多用于开发游戏、模拟器、媒体播放器等多媒体应用领域。

SDL 使用 GNU 宽通用公共许可证为授权方式，意指动态链接（dynamic link）其库并不需要开放本身的源代码。因此诸如《雷神之锤 4》（Quake 4）等商业游戏也使用 SDL 来开发。

虽然 SDL 时常被比较为‘跨平台的 DirectX’，然而事实上 SDL 是定位成以精简的方式来完成基础的功能，它大幅度简化了控制图像、声音、输出入等工作所需撰写的代码。但更高级的绘图功能或是音效功能则需搭配 OpenGL 和 OpenAL 等 API 来达成。另外它本身也没有方便创建图形用户界面的函数。

SDL 在结构上是将不同操作系统的库再包装成相同的函数，例如 SDL 在 Windows 平台上其实是 DirectX 的再包装，旧版本包装的是 DirectX 5，现时的版本（SDL 1.2）则是 DirectX 7。而在使用 X11 的平台上（包括 Linux），SDL 则是与 Xlib 库沟通来输出图像。

虽然 SDL 本身是使用 C 语言写成，但是它几乎可以被所有的编程语言所使用，例如：C++、Perl、Python（借由 pygame 库 、Pascal 等等，甚至是 Euphoria、Pliant 这类较不流行的编程语言也都可行。

SDL 库分为 Video、Audio、CD-ROM、Joystick 和 Timer 等若干子系统，除此之外，还有一些单独的官方扩充函数库。这些库由官方网站提供，并包含在官方文档中，共同组成了 SDL 的“标准库”：

* SDL_image—支持时下流行的图像格式：BMP、PPM、XPM、 PCX、GIF、JPEG、PNG、TGA。
* SDL_mixer—更多的声音输出函数以及更多的声音格式支持。
* SDL_net—网络支持。
* SDL_ttf—TrueType 字体渲染支持。
* SDL_rtf—简单的 RTF 渲染支持。

SDL 的官方网站：http://www.libsdl.org

第三章　准备工作

在开始之前，读者需要先做好以下工作：

1) 安装 Fedora9;
2) 安装友善之臂提供的 mini6410 交叉编译器;
3) 烧写最新 Linux 系统的 Image 到开发板;

3.1　安装Fedora9

本教程中的所有开发工作都是在发行版为Fedora9 的Linux 桌面环境下进行的,所以在开始之前，读者需要先在 PC 上安装 Fedora9，安装方法可以参考 FriendlyARM mini6410 光盘 A 中的用户文档中的“安装并设置 Fedora9”章节。

3.2　安装交叉编译器

从光盘上拷贝 mini6410 上的交叉编译器到 /opt 下，目前最新的 mini6410 交叉编译器是 arm-linux-gcc-v6-vfp-20101103.tgz， 如果你的 mini6410A 光盘中的版本比较旧，建议先从友善之臂官方网站 http://www.arm9.net 下载最新的 mini6410 光盘。

在光盘上找到交叉编译后，将其拷到到 /opt 下，然后输入以下命令进行解压：

```
# cd /
# tar xvzf /opt/arm-linux-gcc-v6-vfp-20101103.tgz
```

交叉编译器会解压到 /opt/FriendlyARM/toolschain/4.5.1/ 目录下，现在将编译器路径添加到 PATH 环境变量中：

```
# export /opt/FriendlyARM/toolschain/4.5.1/bin/:$PATH
```

执行以下命令能成功显示编译器版本信息表明编译器已安装成功。

```
# arm-linux-gcc -v
```

3.3 烧写最新Linux系统的Image到开发板

本手册并不介绍如何移植 Linux Kernel 和 Bootloader，因此，需要首先在 mini6410 开发板上烧写用于 Linux 系统的 Images。

不会烧写的读者，请参考友善之臂提供的光盘上的 mini6410 用户手册。

3.4 所需游戏装备说明(USB手柄和USB键盘)

本手册所移植的两个游戏《仙剑奇侠传》和《超级马里奥战争(Super Mario War)》要进行畅玩的话，需要以下运行条件：

游戏	USB 键盘	USB 手柄	4 寸屏	7 寸屏
《仙剑奇侠传》	支持	支持	支持	支持
《超级马里奥战争(Super Mario War)》	支持	支持，但需要 USB 键盘先进行设置。	不支持	支持

如上所述，《仙剑奇侠传》游戏同时支持 USB 手柄和键盘，而《超级马里奥战争(Super Mario War)》则需要 USB 键盘才能玩，而且，《超级马里奥战争(Super Mario War)》不支持 4 寸屏。

没有 USB 键盘，只有 PS/2 键盘的朋友可以购买一个 USB 转 PS/2 的转换头来达到目的。

下面是所需游戏装备的截图：

		
USB 手柄	USB 键盘	USB 转 PS/2 的转换头

第四章　移植SDL运行库到Mini6410 上

在移植游戏之前，我们首先需要将 SDL/SDL_mixer/SDL_ttf/SDL_image 这四个库移植到 Mini6410 上，另外还需要 iconv 库。

请从 Mini6410 光盘 A 中的“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\源代码”目录下找到以下文件，并**拷贝到 /tmp 目录下备用**：

文件名	说明	原始下载地址
libiconv-1.13.1.tar.gz	Iconv 库	http://ftp.gnu.org/pub/gnu/libiconv/
SDL-1.2.14.tar.gz	SDL 库	http://www.libsdl.org/download-1.2.php
SDL_image-1.2.8.tar.gz	SDL_image 库	http://www.libsdl.org/projects/SDL_image/release/
SDL_ttf-2.0.9.tar.gz	SDL_ttf 库	http://www.libsdl.org/projects/SDL_ttf/release/
SDL_mixer-1.2.9.tar.gz	SDL_mixer 库	http://www.libsdl.org/projects/SDL_mixer/release/

在 /opt/mini6410 目录下创建目录 games 来作为我们本次移植工作的工作目录，输入以下命令创建：

```
# mkdir –p /opt/mini6410/games
```

再在 games 目录下创建一个 output 目录，用来存放编译生成的可执行文件或者类库等：

```
# cd /opt/mini6410/games
# mkdir output
```

4.1　编译iconv库

输入以下命令编译：

```
# tar xvzf /tmp/libiconv-1.13.1.tar.gz
# cd libiconv-1.13.1
# ./configure --host=arm-linux --prefix=/opt/mini6410/games/output
# make
# make install
```

4.2 编译SDL库

输入以下命令编译：

```
# cd /opt/mini6410/games
# tar xvzf /tmp/SDL-1.2.14.tar.gz
# cd SDL-1.2.14
# ./configure --prefix=/opt/mini6410/games/output --disable-video-nanox -disable-video-qtopia
  --disable-video-photon --disable-video-ggi --disable-video-svga --disable-video-aalib
  --disable-video-dummy --disable-video-dga --disable-arts --disable-esd --disable-alsa
  --disable-video-x11 --disable-nasm --enable-joystick --enable-input-tslib
  --enable-video-fbcon --host=arm-linux --build=i386
# make
# make install
```

4.3 编译SDL_image库

输入以下命令编译：

```
# cd /opt/mini6410/games
# tar xvzf /tmp/SDL_image-1.2.8.tar.gz
# cd SDL_image-1.2.8
#       ./configure       --enable-shared       --enable-static       --host=arm-linux       --build=i386
--prefix=/opt/mini6410/games/output          --enable-bmp          --enable-gif          --enable-jpg
--enable-png                --enable-tif                --enable-pnm                --enable-xpm
--disable-sdltest                          SDL_CFLAGS="-I/opt/mini6410/games/output/include"
SDL_LIBS="-L/opt/mini6410/games/output/lib                                                    -lSDL"
LIBPNG_CFLAGS="-I/opt/mini6410/games/output/include"
LIBPNG_LIBS="-L/opt/mini6410/games/output/lib                                                 -lpng"
CPPFLAGS="-I/opt/mini6410/games/output/include/SDL      -I/opt/mini6410/games/output/include/"
LDFLAGS="-L/opt/mini6410/games/output/lib -ljpeg -lts -lpng -liconv"
# make
# make install
```

4.4 编译SDL_ttf库

输入以下命令生成 Makefile：

```
# cd /opt/mini6410/games
# tar xvzf /tmp/SDL_ttf-2.0.9.tar.gz
# cd SDL_ttf-2.0.9
# ./configure --host=arm-linux --prefix=/opt/mini6410/games/output --enable-shared --enable-static
--disable-sdltest
--with-freetype-prefix=/opt/FriendlyARM/toolschain/4.5.1/arm-none-linux-gnueabi/sys-root/usr/
--with-sdl-prefix=/opt/mini6410/games/output/ CFLAG="-I/opt/mini6410/games/output/include"
LDFLAGS="-L/opt/mini6410/games/output/lib -lSDL -lts -liconv -lfreetype"
```

在编译之前手工修改 Makefile,去掉 OpenGL 相关处理：

```
# vi Makefile
```

定位到 123 行，将以下内容中的“-I/usr/include -DHAVE_OPENGL”删除(红色字体部分)：

```
CFLAGS = -g -O2 -I/usr/include/freetype2 -I/opt/mini6410/games/output/include/SDL
-D_GNU_SOURCE=1 -D_REENTRANT   -I/usr/include -DHAVE_OPENGL
```

再定位到 143 行，将：

```
GL_LIBS =   -L/usr/lib -lGL
```

改成

```
GL_LIBS =
```

再定位到 266 行，将：

```
glfont_LDADD = libSDL_ttf.la   -L/usr/lib -lGL -lm
```

改成

```
glfont_LDADD = libSDL_ttf.la -lm
```

清空源代码文件 glfont.c 中的源代码 (先删除，再用 vi 创建)：

```
# rm glfont.c.-f
# vi glfont.c
```

在 vi 编辑器中输入以下内容并保存：

```
void main() {}
```

最好，输入命令编译并安装：

```
# make
# make install
```

4.5 编译SDL_mixer库

输入命令编译并安装：

```
# cd /opt/mini6410/games
# tar xvzf /tmp/SDL_mixer-1.2.9.tar.gz
# cd SDL_mixer-1.2.9
#  ./configure  --enable-music-mp3-mad-gpl  --enable-music-mp3=no  --host=arm-linux
--prefix=/opt/mini6410/games/output  --with-sdl-prefix=/opt/mini6410/games/output  --enable-shared
--enable-static
# make
# make install
```

4.6 将SDL库部置到Mini6410 上

通过以下命令打包 SDL 类库和其它相关的类：

```
# cd /opt/mini6410/games/output/
# tar cvzf sdl_setup.tar.gz bin lib share
```

将 sdl_setup.tar.gz 拷贝到 SD 卡中，然后将 SD 卡拷贝到 Mini6410 开发板上，在 Mini6410 根目录进行解压：

```
@# cd /
@# tar xvzf /sdcard/sdl_setup.tar.gz
```

至少，SDL 运行库已部置完成。

第五章　移植《仙剑奇侠传》到Mini6410

5.1 《仙剑奇侠传》是什么游戏？

《仙剑奇侠传》是一款被众多玩家公认为“旷世奇作”的经典中文 RPG 游戏，自 1995 年 7 月出片以来，即在各种游戏杂志的排行榜中名列前茅，至 1996 年 10 月为止，已蝉联 14 个月冠军。仙剑还荣获 1995 年 CEM STAR《最佳角色扮演游戏奖》以及 1995 年 KING TITLE《游戏类金袋奖》的肯定。后来系列第一部和第三部分别于 2004 年与 2008 年被改编成电视剧。如今，仙剑系列最新作《仙剑奇侠传五》正由软星科技（北京）有限公司接手研发。

本手册移植的《仙剑奇侠传》是一个非官方的跨平台版本，是开源的，使用 SDL（Simple DirectMedia Layer）类库编写的，SDL 是一套免费的跨平台多媒体应用编程接口，用于游戏、游戏开发工具、模拟器、样本演示、多媒体应用等。

《仙剑奇侠传》的游戏截图：

5.2 交叉编译《仙剑奇侠传》

请从 Mini6410 光盘 A 中的“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\源代码”目录下找到以下文件，并**拷贝到 /tmp 目录下备用**：

文件名	说明	下载地址
sdlpal-61376.zip	仙剑奇侠传非官方 SDL 版本的源代码	http://sdlpal.codeplex.com/

执下以下命令解压源代码，并定位到源代码目录：

```
# cd /opt/mini6410/games
# unzip /tmp/sdlpal-61376.zip
# cd sdlpal/
```

接着，请按以下步骤修改源代码和 Makefile：

5.2.1 修改main.c，更改屏幕分辨率

用文本编辑器打开 main.c 文件，定位到 535/536 行，将内容：

```
wScreenWidth = 640;
wScreenHeight = fFullScreen ? 480 : 400;
```

改成

```
getFBSize(&wScreenWidth,&wScreenHeight);
if(wScreenWidth> 640 || wScreenHeight > 480) {
    wScreenWidth = 640;
    wScreenHeight = 480;
}
```

再定位到 439 行，在 main 函数前面，插入以下函数，该函数用于动态获取屏幕分辨率：

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
```

```
void getFBSize(int *w, int *h)
{
	char buf[256];
	*w = 480;
	*h = 272;
	FILE *file=fopen("/sys/class/graphics/fb0/virtual_size","r");
	if (!file) {
		return ;
	}
	int tmp1=0,tmp2=0;
	if (fgets(buf, sizeof(buf), file)) {
		if (sscanf(buf, "%d,%d", &tmp1, &tmp2) == 2) {
			*w = tmp1;
			*h = tmp2;
		}
	}
	fclose(file);
	return ;
}
```

5.2.2　修改Makefile，使用交叉编译器编译

通过对主 Makefile 的分析，实际编译动作是 nbproject/Makefile-Debug.mk 来实现的，因此，用文本编辑器打开文件 nbproject/Makefile-Debug.mk，进行如下修改：

将 14 行的 RANLIB=ranlib 改成 **RANLIB=arm-linux-ranlib**
将 15 行的 CC=gcc 改成 **CC=arm-linux-gcc**
将 16 行的 CCC=g++改成 **CCC=arm-linux-g++**
将 17 行的 CXX=g++改成 **CXX=arm-linux-g++**
将 65 行的 CFLAGS=`sdl-config --cflags` 改成
CFLAGS="-I/opt/mini6410/games/output/include"
将 68 行的 CCFLAGS=`sdl-config --cflags`改成 **CCFLAGS = $(CFLAGS)**
将 69 行的 CXXFLAGS=`sdl-config --cflags`改成 **CXXFLAGS = $(CFLAGS)**
将 82 行的${LINK.cc} `sdl-config --libs` -Xlinker -Map=dist/Debug/GNU-Linux-x86/sdlpal.map -o dist/Debug/GNU-Linux-x86/sdlpal ${OBJECTFILES} ${LDLIBSOPTIONS}改为**${LINK.cc} -L/opt/mini6410/games/output/lib -lfreetype -liconv -lmad -lz -ljpeg -lpng -lSDL_mixer -lSDL_image -lSDL_ttf -lSDL -Xlinker -Map=dist/Debug/GNU-Linux-x86/sdlpal.map -o sdlpal_arm ${OBJECTFILES}**

把所有的 GNU-Linux-x86 替换成 **GNU-Linux-arm**
把所有的-I/usr/include/SDL 替换成**-I/opt/mini6410/games/output/include/SDL**

提醒：如果修改不成功，或者觉得修改太麻烦，Mini6410 光盘 A 中的“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\源代码”目录下有一个文件 sdlpal_mini6410_src.tar.gz 是我们修改好的源代码包，你可以直接拿来用。

5.2.3　修改input.c，修复手柄按键的问题

一般来说，用户可以跳过本步骤，直接编译并运行仙剑了，如果在运行时出现手柄操作不正常时，再回到参考本章节进行修改。

如果你是用 USB 键盘来操作游戏，则可直接跳过这个步骤。

5.2.3.1　修复北通神鹰(BETOP C036)手柄的按键问题

笔者使用北通神鹰(BETOP C036)手柄进行测试时，出现方向键只能操作左右两个方向，上下两个方向没有办法操作的问题，看了一下代码，发现处理手柄按键的源代码是在 input.c 文件里面，因此可以通过修改 input.c 来解决这个问题，请用编辑器打开 input.c，将其中的 PAL_JoystickEventFilter 函数体内的代码全部删除，替换成以下代码：

```
#ifdef PAL_HAS_JOYSTICKS
    BOOL keyDown = FALSE;

    switch (lpEvent->type)
    {
    case SDL_JOYAXISMOTION:
        //
        // Moved an axis on joystick
        //

        // printf("PAL_JoystickEventFilter: lpEvent->jaxis.axis: %d:    jaxis.value: %d\n", lpEvent->jaxis.axis,
lpEvent->jaxis.value );
        if (lpEvent->jaxis.axis == 0) {
            if (lpEvent->jaxis.value == 32767 && g_InputState.dir != kDirSouth && g_InputState.dir !=
kDirNorth)
```

```
            {
                if (g_InputState.dir != kDirEast)
                {
                    g_InputState.dwKeyPress |= kKeyRight;
                }
                g_InputState.prevdir = g_InputState.dir;
                g_InputState.dir = kDirEast;
            }
            else if (lpEvent->jaxis.value == -32768 && g_InputState.dir != kDirSouth && g_InputState.dir !=
kDirNorth)
            {
                if (g_InputState.dir != kDirWest)
                {
                    g_InputState.dwKeyPress |= kKeyLeft;
                }
                g_InputState.prevdir = g_InputState.dir;
                g_InputState.dir = kDirWest;
            }
            else if (lpEvent->jaxis.value == 255 && g_InputState.dir != kDirEast && g_InputState.dir !=
kDirWest)
            {
                if (g_InputState.dir != kDirSouth)
                {
                    g_InputState.dwKeyPress |= kKeyDown;
                }
                g_InputState.prevdir = g_InputState.dir;
                g_InputState.dir = kDirSouth;
            }
            else if (lpEvent->jaxis.value == 0 && g_InputState.dir != kDirEast && g_InputState.dir !=
kDirWest)
            {
                if (g_InputState.dir != kDirNorth)
                {
                    g_InputState.dwKeyPress |= kKeyUp;
                }
                g_InputState.prevdir = g_InputState.dir;
                g_InputState.dir = kDirNorth;
            }
            else
            {
```

```
                g_InputState.dir = kDirUnknown;
                g_InputState.prevdir = kDirUnknown;
            }
        }

        break;
    case SDL_JOYBUTTONDOWN:
        //
        // Pressed the joystick button
        //
        switch (lpEvent->jbutton.button & 1)
        {
        case 0:
            g_InputState.dwKeyPress |= kKeyMenu;
            break;

        case 1:
            g_InputState.dwKeyPress |= kKeySearch;
            break;
        }
        break;
    }
#endif
```

注意上述代码中用红色字体标出的内容，修改的含意是根据 lpEvent->jaxis.value 传过来的键值，重新转义为操作方向，参考红色字体标出的代码，例如，当 **lpEvent->jaxis.value == 32767 时，方向是向右。**

5.2.3.2 修复其它手柄的按键设置

可参考上一个章节对北通神鹰手柄的修改，根据 lpEvent->jaxis.value 传过来的键值，重新转义为游戏所认识的操作方向，参考红色字体标出的代码，例如，北通神鹰手柄当按键的值 lpEvent->jaxis.value == 32767 时，方向是向右。

如何知道手柄按键的键值呢？

SDL 库中带一个手柄的测试程序，可以通过它进行测试，该程序位于 /opt/mini6410/games/SDL-1.2.14/test/目录下，文件名为 testjoystick.c，使用以下命令交叉编译这个程序：

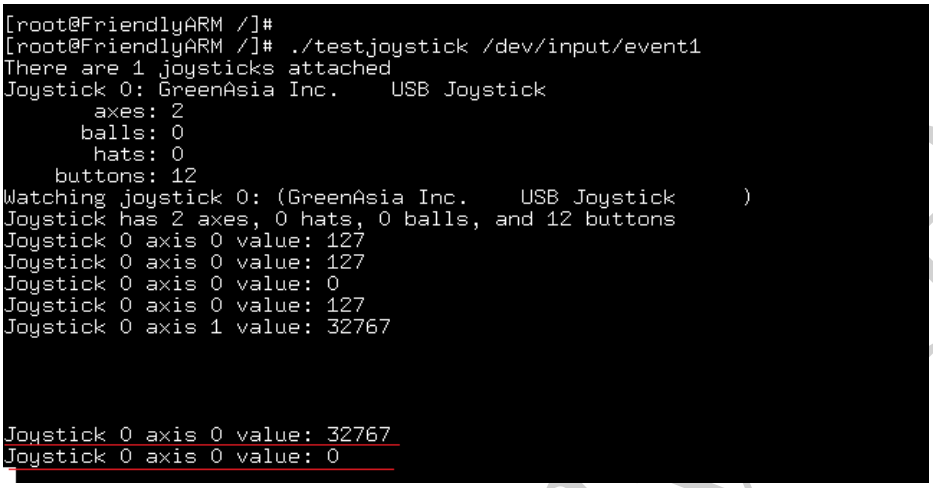
```
# arm-linux-gcc testjoystick.c -o testjoystick -L/opt/mini6410/games/output/lib
```

```
-I/opt/mini6410/games/output/include/SDL/ -lSDL
```

将 testjoystick 放到 Mini6410 开发板，输入如下命令运行它：

```
@# ./testjoystick /dev/input/event1
```

我接的手柄是北通神鹰，接一下方向右键，输出以下红色标出的两行信息，第一行表示按下的，第二行表示释放的，我们只取按下的值即可(如图中的 32767)，如下图所示：



如果 testjoystick 程序根据没有办法认出你的 USB 手柄，则可以尝试重新编译内核，加上更多手柄的驱动支持，相关的选项是：进入 Device Drivers -> Input device support，然后选中 Joystick interface（前面打*），再选中 Joysticks/Gamepads（前面打*),然后进入 Joysticks/Gamepads 里面，把所有的驱动都选中（前面打*），按 Exit 退出，进入 HID Devices，前所有选项选中（前面打*），再进入 Sepcial HID drivers，将里面的所有选项都选中（前面打*），然后重新编译并烧写内核试试。

5.2.4　输入make执行编译

在 sdlpal 目录下执行 make 开始编译：

```
# cd /opt/mini6410/games/sdlpal/
# make
```

编译完成后，在 sdlpal 目录下会生成可执行文件 sdlpal_arm。

5.3 在Mini6410 上部署《仙剑奇侠传》

光有可执行文件还不能运行游戏，还需要《仙剑奇侠传》原版的资源文件，请从 Mini6410 光盘 A 中的“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\源代码”目录下找到以下文件，并**拷贝到 /tmp 目录下备用：**

文件名	说明
xianjian_res.tar.gz	仙剑奇侠传的资源文件

执下以下命令，我们将创建一个目录 output_pal 用于存放仙剑的执行文件和资源，然后将资源解压到该目录下：

```
# cd /opt/mini6410/games/
# mkdir -p output_pal/
# mkdir -p output_pal/bin/
# cd output_pal
# tar xvzf /tmp/xianjian_res.tar.gz
```

将会在/opt/mini6410/games/output_pal 目录下生成 xianjian 目录。

接着，把我们编译出来的仙剑可执行文件拷到/opt/mini6410/games/output_pal/bin/目录下：

```
# cd /opt/mini6410/games/output_pal/
# cp ../sdlpal/sdlpal_arm bin/
```

执行以下命令，将仙剑的目录打包：

```
# cd /opt/mini6410/games/output_pal/
# tar cvzf xianjian_setup.tgz xianjian bin
```

将 xianjian_setup.tar.gz 拷贝到 SD 卡中，然后将 SD 卡拷贝到 mini6410 开发板上，在 mini6410 根目录进行解压：

```
@# cd /
@# tar xvzf /sdcard/xianjian_setup.tar.gz
```

注：在运行之前，别忘了先在 Mini6410 上部署 SDL 类库，请参考章节： 4.6。

5.4 在Mini6410 上运行《仙剑奇侠传》

在/bin 下创建一个脚本 run_xianjian.sh 用于运行仙剑：

```
@# vi /bin/run_xianjian.sh
```

在 vi 中输入如下内容：

```
#!/bin/sh
export SDL_NOMOUSE=1
export LD_LIBRARY_PATH=/lib:/usr/lib:/xianjian/lib:$LD_LIBRARY_PATH
cd /xianjian
sdlpal_arm
```

给脚本 run_xianjian.sh 加上可执行权限：

```
@# chmod +x /bin/run_xianjian.sh
```

修改 /etc/init.d/rcS，将以下内容：

```
/bin/qtopia &
```

改为：

```
/bin/run_xianjian.sh &
```

将 USB 游戏手柄， 或者 USB 键盘插上 Mini6410，重新开机即会自动运行仙剑奇侠传游戏，效果如下所示：

第六章　移植《超级马里奥战争(Super Mario War)》到Mini6410 上

6.1 《超级马里奥战争(Super Mario War)》是什么游戏？

Super Mario War 是一个多人的超级马里奥游戏，其目标是踩踏尽可能多的其他马里奥来赢得比赛。这是对任天堂和由 Samuele Poletto 创作的 Mario War 游戏的敬意，游戏使用来自任天堂游戏的作品和声音。

【游戏特点】

1. 多达 4 个玩家同时享受死斗的乐趣
2. 一大堆的游戏模式 (featuring GetTheChicken, Domination, CTF, …)
3. 配备了 leveleditor，您可以创建自己的地图，并且很多人也这样做了
4. 目前有超过 1000 个地图
5. 比起用棍子戳猴子有趣多了
6. 游戏的全部源代码是开放的
7. 使用 SDL，完全可移植到 Windows，Linux 或 Mac 系统平台
8. 由电脑控制的玩家
9. 会使你快乐，并给你一个模糊的感觉

【游戏截图】

【游戏攻略】

一份简单的图文游戏攻略可以在光盘 A 的以下目录下找到：“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\游戏攻略\超级马里奥战争-攻略.pdf”。

6.2 交叉编译《超级马里奥战争(Super Mario War)》

请从 Mini6410 光盘 A 中的“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\源代码”目录下找到以下文件，并**拷贝到 /tmp 目录下备用**：

文件名	说明	下载地址
smw_mini6410_src.tar.gz	**友善之臂修改过的 Super Mario War 官方原版的源代码**	
smw-1.8-beta2-src-org.tar.gz	Super Mario War 官方原版的源代码	http://code.google.com/p/supermariowar/

《超级马里奥战争》的移植工作比较简单，我只在原版源代码的基础上修正了一些编译错误，然后修改了一个 configuration 编译配置文件就可以编译通过了，因此，这里不再一步一步要说明修改步骤了，在上面我直接给出两份源代码，一份是我们修改过的，一份是原版本，你可以通过用 diff –r 命令比较两个目录，来获悉修改了哪些地方，下面我们将直接使用我们修改过的源代码 smw_mini6410_src.tar.gz 进行编译。

执下以下命令解压源代码，并定位到源代码目录：

```
# cd /opt/mini6410/games
# tar xvzf /tmp/ smw_mini6410_src.tar.gz
# cd smw-1.8-beta2-src /
```

执下以下命令编译并安装：

```
# make
# make install
```

游戏将编译成安装到 **/opt/mini6410/games/output_smw/** 目录下。

6.3 在Mini6410 上部署《超级马里奥战争》

执行以下命令，将超级马里奥战争目录打包：

```
# cd /opt/mini6410/games/output_smw/
# tar cvzf smw_setup.tar.gz usr
```

将 smw_setup.tar.gz 拷贝到 SD 卡中，然后将 SD 卡拷贝到 mini6410 开发板上，在 mini6410 根目录进行解压：

```
@# cd /
@# tar xvzf /sdcard/smw_setup.tar.gz
```

注：在运行之前，别忘了先在 Mini6410 上部署 SDL 类库，请参考章节： 4.6。

6.4 在Mini6410 上运行《超级马里奥战争》

在/bin 下创建一个脚本 run_smw.sh 用于运行仙剑：

```
@# vi /bin/run_smw.sh
```

在 vi 中输入如下内容：

```
#!/bin/sh
export SDL_NOMOUSE=1
export LD_LIBRARY_PATH=/lib:/usr/lib:/xianjian/lib:$LD_LIBRARY_PATH
cd /usr/games/
./smw
```

给脚本 run_smw.sh 加上可执行权限：

```
@# chmod +x /bin/run_smw.sh
```

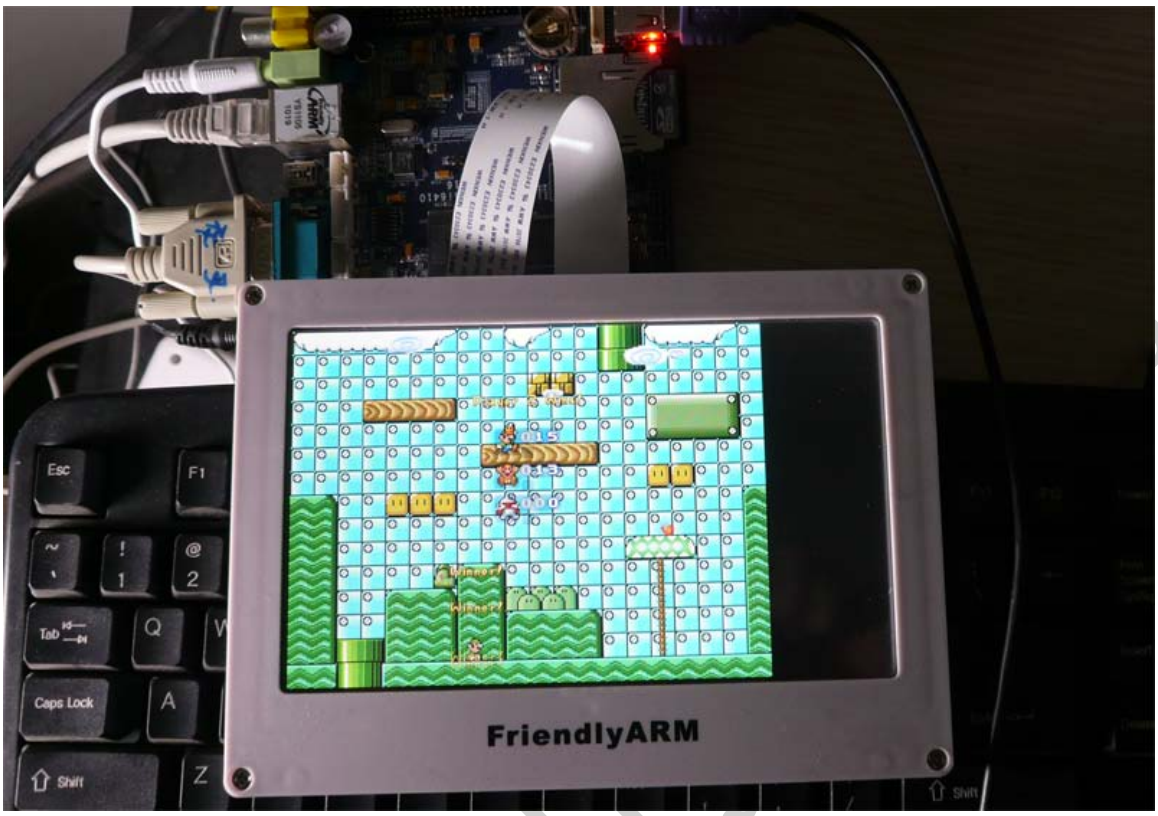
修改 /etc/init.d/rcS，将以下内容：

```
/bin/qtopia &
```

改为：

```
/bin/run_smw.sh &
```

将 USB 键盘插上 Mini6410，重新开机即会自动运行游戏，效果如下所示：



该游戏的界面选项太多，如果不知道怎么玩，一份简单的图文游戏攻略可以在光盘 A 的以下目录下找到“开发文档和教程\专题 03 Mini6410 上移植 SDL 游戏的详细步骤\游戏攻略\超级马里奥战争-攻略.pdf”。

第七章　更多游戏？

访问网址 http://www.libsdl.org/games.php，将会得到一份用SDL编写的开源的游戏清单，你可以尝试移植更多的游戏到Mini6410/Tiny6410 上，这也是友善之臂为大家编写本手册的初衷，希望广大学习用户的能力能在乐趣中成长提高。

更多开发相关的专题文档，敬请期待…

请密切留意我们的论坛(http://www.arm9home.net)上发布的最新消息。