

# KF32 系列

## 库生成与使用说明 V1.1

2022 年 3 月

## 目录

库生成与使用说明 V1.1.....	1
目录.....	2
1. 概述.....	3
2. 库生成.....	3
2.1 库源码开发.....	3
2.2 编译输出库文件.....	3
2.2.1 集成开发环境 ChipON IDE KF32 功能输出库.....	4
2.2.2 工具命令实现输出库.....	5
3. 库使用.....	6
3.1 工具层库配置使用.....	7
3.2 工具链层库特殊关系说明.....	9
3.3 代码层引入声明示例.....	9
3.4 代码层调用示例.....	9
4. 附录 A: KF32-AR 库管理程序选项.....	10
5. 附录 B: KF32-STRIP 信息移除程序选项.....	10
6. 附录 C: 命令行脚本参考源码.....	11

## 1. 概述

本文档介绍 KungFu32 系列工具链的生成自定义库以及使用自定义库。

源码编译默认输出为后缀".o"文件，可作为库输入，也可直接提供其他项目使用；将.o文件添加(包括全新创建)到默认后缀“.a”的库文件，用于其他项目使用。

注：输出“.o”，“.a”的文件应为同目标机平台的编译环境产生，这里专指 ChipON KungFu32, 即库是运行平台专有化产物。

## 2. 库生成

### 2.1 库源码开发

库的源码组成文件可以是汇编文件，也可以是.c文件，其可以包含头文件。库可提供变量和函数方法供其他项目使用。

源码编写可以使用宏定义，但宏展开编译，即不存在2次修改宏控制代码分支或改变参数。即不建议内部宏编写入提供用户的头文件中。

不建议存放与库服务不相关的对象，如变量或函数，若源码存在可以static修饰使能外部不可见。

**建议保持库的独立性。**若当前库文件调用其他库文件中方法或变量，传递库时该当前库应优先传递；若其调用的其他库文件同时存在调用当前库文件方法或变量时，需使用--start-group archives --end-group的格式将该系列库作为一个组，具体见使用说明。

### 2.2 编译输出库文件

当库发布时，需要提供库配套的头文件供其他项目包含，该头文件中应包括全局对外变量和函数的声明。建议库以lib为库文件名前缀(库传递可省略输入)。

KungFu32 系列芯片具有完整的工具链，其包括编译器cc、汇编器as、链接器ld、库管理器ar。可以使用集成开发环境IDE执行包含库的项目的编译，通过"编译后执行"界面输

入命令的伴随编译的完成库制作，也可以通过外部命令行实现库制作。支持可以通过编写命令行脚步调用 KungFu32 工具链执行源码的编译与制作库文件。

### 2.2.1 集成开发环境 ChipON IDE KF32 功能输出库

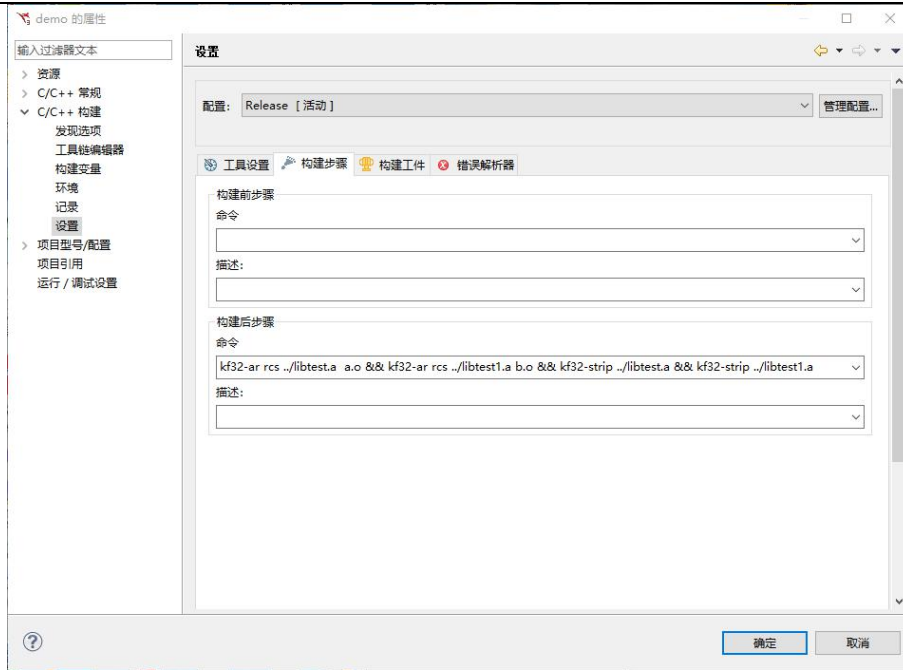
从项目资源管理器中选择对应的项目，右键进入"项目属性"页面。

在左侧"C/C++构建"下面选择"设置"。

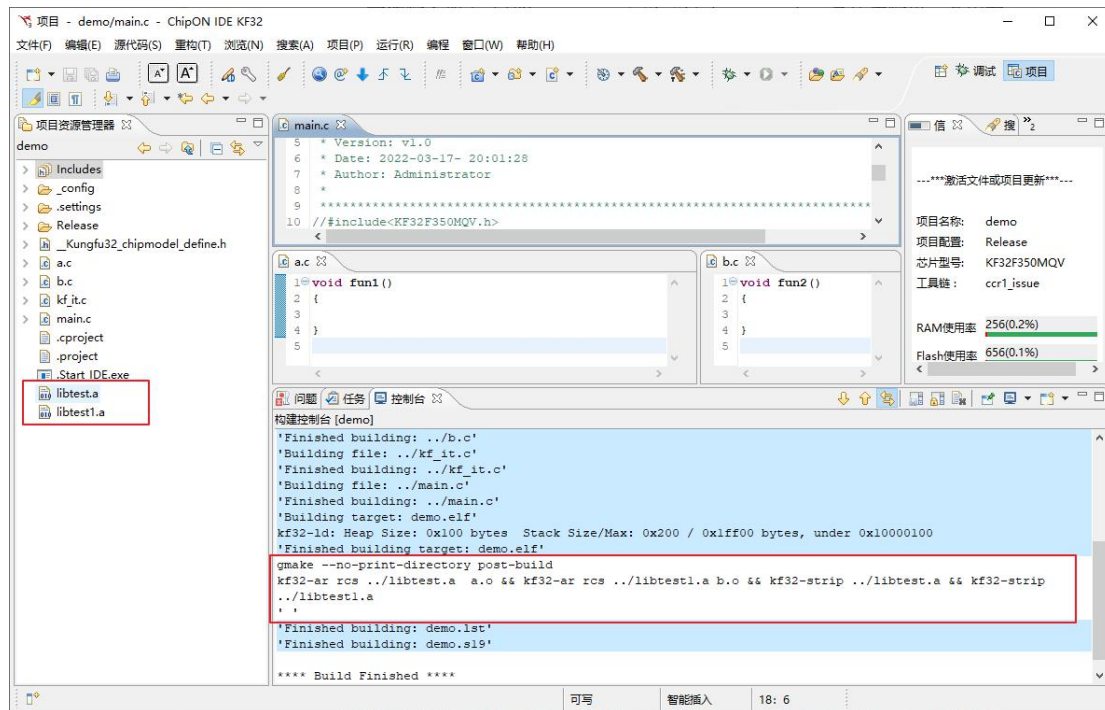
**配置模式具有独立的可配置属性**，因此选择你需要的配置，如 Debug（默认不优化编译）和 Release（默认优化等级 2）。当前页面选择构建步骤，在"构建步骤"的"构建后步骤"的"命令"栏下面输入输出库命令，可以使用“;”或“&&”（推荐）间隔添加多组命令。

示例输入如下 `kf32-ar rcs ../outputlib.a a.o b.o` 完成库输出，其中参数 `rsc` 意味着替换、`randlib` 格式、不提示库文件是否存在（不存在时创建）。更多选项可以输出 `kf32-ar --help` 查看。示例中的“../”对应系统的上一级文件路径，即构建项目时设定当前路径为当前的编译模式输出文件夹下，如 Release，此时对于这项目的根目录下。若对应的库的源文件存放在子文件夹中，输入库文件时应使用相对文件路径，如 `./bsp/a.o`，当源文件在项目根目录时可以直接使用文件名或相对文件名，如 `b.o` 或 `./b.o`。

需要注意的是，当前默认 Debug 或 Release 便于调试目的均使能了调试选项 `-gstab+`，因此可以制作库时关闭，或使用命令后期移除调试信息。示意配置如下，输入样例“`kf32-ar rcs ../libtest.a a.o && kf32-ar rcs ../libtest1.a b.o && kf32-strip -g ../libtest.a && kf32-strip -g ../libtest1.a`”。



注：使用集成环境开发库时，只有编译正常通过时才出发构建后步骤，因此，该库源码所在项目应该是完整的，若非原始项目的独立库开发，可优先利用 IDE 创建一个最小样例项目，然后添加库相关文件并编译。构建后步骤的执行会输出在控制台中指示，如下：



## 2.2.2 工具命令实现输出库

这里假定用于制作库文件已加工完成，即根据需保留调试信息。可通过 kf32-ar 程序执行库文档创建、更新、移除等操作。可以在工具所在路径运行，或注册工作路径后运行。

常用的命令包括：添加库对象文件到库、删除库中某库对象文件、查看库中库对象文件。

```
kf32-ar rcs libA.a libA.o *.o
```

```
kf32-ar d libA.a libA.o *.o
```

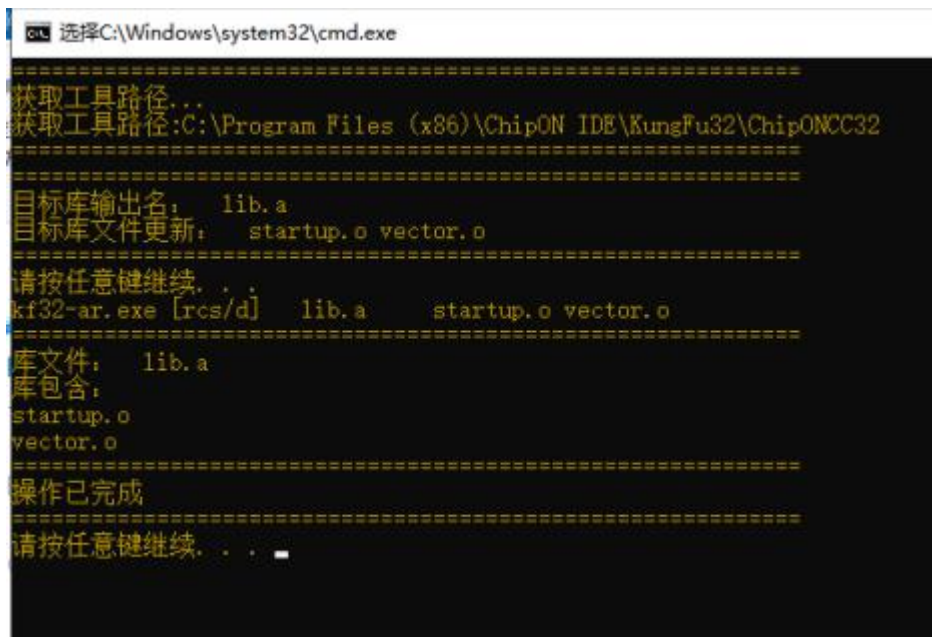
```
kf32-ar t libA.a
```

```
for %%I in (*.o) do kf32-ar rcs %%~n1.a %%~n1.o
```

ChipON IDE KF32 工具提供对应的脚本文件 **“Drag get library and update all object file in path.bat”**。该文件存在于 ChipON IDE KF32 安装路径下，也可从附录中复制原型并建立批处理文件并运行，若为 Win7 及以上操作系统，应该配置管理员权限运行，否则可能获取工具路径失败。

脚步使用方法为将对应要制作库的库对象文件与其放在同一级文件路径下，默认运行输出 lib.a 的库，如果需要往库中添加库对象内容，将对应的库也拷贝到当前路径下。

拖动当前库到对应的批处理命令上执行，也可以拖动当前路径的一个库文件对象文件启动批处理命令，此时将输出对应库文件对象名关联的库文件，如拖动对象为 a.o，目标库文件为 lib.a。示例拷贝如下文件并拖动单个 o 文件到脚本运行参考如下：



```
选择C:\Windows\system32\cmd.exe
=====
获取工具路径...
获取工具路径:C:\Program Files (x86)\ChipON IDE\KungFu32\ChipONCC32
=====
目标库输出名: lib.a
目标库文件更新: startup.o vector.o
=====
请按任意键继续. . .
kf32-ar.exe [rcs/d] lib.a startup.o vector.o
=====
库文件: lib.a
库包含:
startup.o
vector.o
=====
操作已完成
=====
请按任意键继续. . .
```

注：根据需要，你可以修改脚本添加指定的库文件，或优先处理移除库文件调试信息后再调用制作库命令。

### 3. 库使用

库的使用支持汇编项目和 C 项目，但应提供满足其语言的头文件。将库与库头文件复制到项目根目录与目录下。

项目属性下配置链接器选项的库路径与库传递。

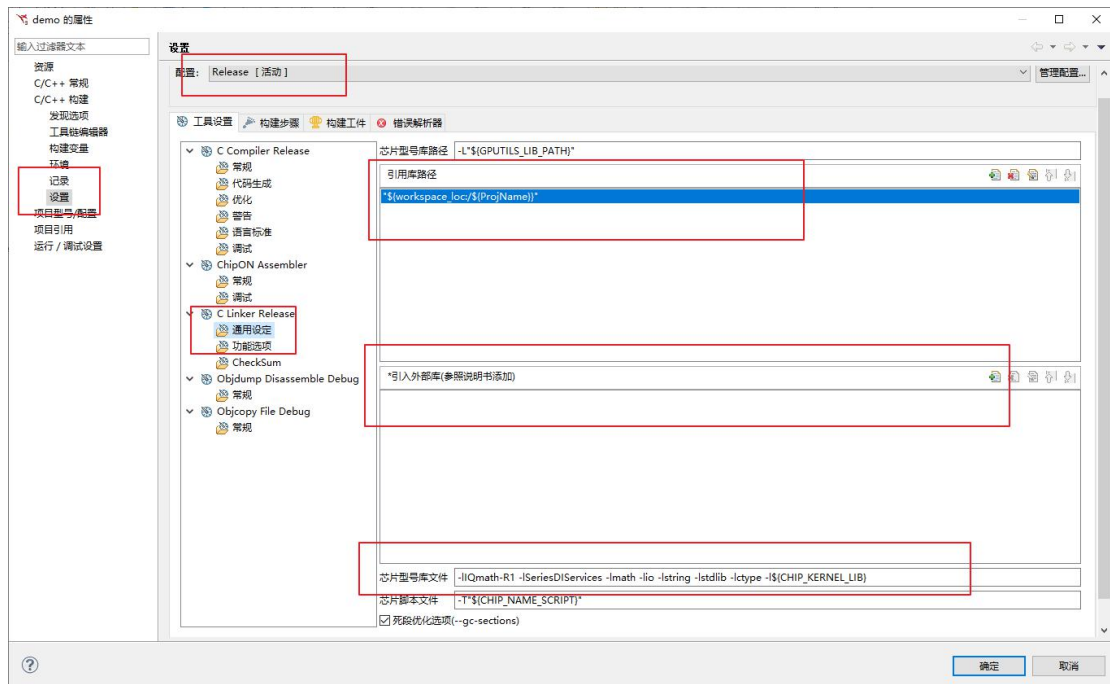
在使用库的项目下开发代码时，其应包括库声明的头文件。

### 3.1 工具层库配置使用

从"项目资源管理器"中选择对应的项目，右键进入项目属性。

在 C/C++ 构建下面选择设置。

➤ 应分别选定配置模式 Debug 和 Release 进行库引用配置。不可选择所有配置，即 Debug 和 Release 模式存在差异化配置选项（否则所有工具选项将同步为一致，如优化等级）。在对应模式下，选项“链接”工具下的通用设定下，引用库路径配置和引入外部库配置或执行芯片型号库文件输入框下添加。



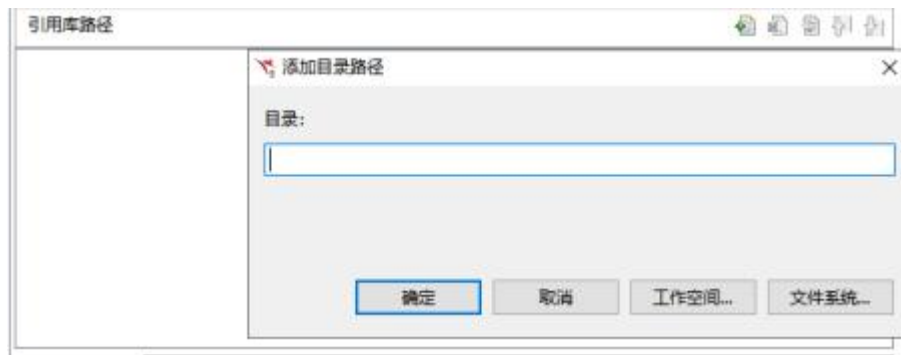
➤ 库的引入应注意先后顺序，即独立库放在后面，使用其他库的库应放在前面。如 A. a 方法使用 B. a 的方法，A 库应该优先配置。如果编写库使用了标准库，如软件浮点、IO 打印、字符转换等

➤ 因为工具设计先包括芯片型号库文件配置的库，在包括引入外部库中的文件，因此即使为外部库文件，仍需要在芯片型号库文件栏进行库添加。

添加库的参数选项为 -l，后面的库文件可以是完整的库文件名，也可以是简写的形式，但库名必须为 lib 为前缀。



- ❖ **配置自定库库所在搜索路径，配置引用库路径**，点击添加图标进行路径添加，如果库位于当前项目下，建议通过“工作空间”进行路径的选定，如果在主机指定位置可以通过“文件系统”使用绝对路径，也可以通过复制粘贴直接在文本框中输入，通过确定完成当前路径的添加。



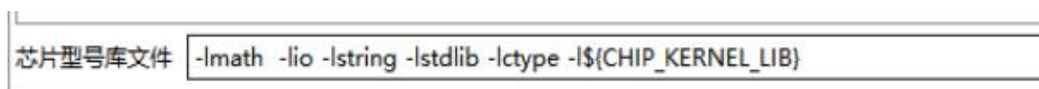
- ❖ **配置引入外部库**

通过“工作空间”选择空间下项目的对应库文件，或“文件系统”的绝对位置。也可以直接根据格式特效实际输入并确定，如库 libComm.a 的直接输入为 libComm.c 或 Comm，即该选项下的输入会自动添加前缀“-l”。当使用工作空间时输入结果显示其资源管理器的相对空间，实际工具将处理为仅传递其本身，如-l1libComm.o 对应如下输入：



- ❖ **在系统库前面添加自定义库**

**默认引入外部库优先传递**，这里作为最小系统需要的工具库。也可以在这里添加实现库扩展，建议直接在其最前面添加对应的库，即使当前库使用了系统库中的函数，其需要的库函数也能正常识别，若当前库为独立运行库，则不局限其位置顺序，输入如-l1Comm 或-l1libComm.a，注这里需要-l 的库选项格式，工具默认标准库包括如图所示：



即数学方法库、串口操作库、字符串处理库、系统模式相关库、类型库、芯片系列编译运算库。



## 3.2 工具链层库特殊关系说明

基于 binutils 的链接器 ld 考虑内存与效率，识别库时仅保留之前对象文件 o 或前面库需要的变量和函数，否则不给予保留。

假定 libB 需要 libA 中方法，传递路径时应将 libB 放置在前面，若 -lA -lB，其中在 libB 的对象中找不到符号（在 libA 中定义），按照 -lB -lA 则可以通过，但若 A 库中同时也使用 B 库中方法，可以考虑为 -lA -lB -lA 的冗余传递（不推荐）。建议设计独立关系库，若需要其他库方法时，推荐使用库组的概念，即格式化如下：“`--start-group -lA -lB --end-group`”，若在 IDE 工具环境下，需在芯片型号栏输入库传递，如“`--start-group -lA -lB -lIQmath-R1 -lSeriesDIServices -lmath -lio -lstring -lstdlib -lctype -l${CHIP_KERNEL_LIB} --end-group`”。

## 3.3 代码层引入声明示例

如示例使用串口打印函数，代码中包含其头文件：

```
#include "stdio.h"
```

## 3.4 代码层调用示例

需要输出使用的位置直接调用其函数：

```
printf("文件%s:%d\r\n",__FUNCTION__,__LINE__);
fprintf(USART0_STREAM,"文件%s:%d\r\n",__FUNCTION__,__LINE__);
sprintf(USART_Array_Tansmit,"F:%s:%d\r\n",__FUNCTION__,__LINE__);
printf("abcdefghijklmnopqrst1234567890\r\nC1CC\r\n");
fprintf(USART0_STREAM,"abcdefghijklmnopqrst1234567890\r\nC1CC\r\n");
puts("ABBB\t\r\n");
fputs("ABBB\t\r\n", USART0_STREAM);
putchar('A');
fputc('\n', USART0_STREAM);
Receive_flag=getchar();
```

## 4. 附录 A: kf32-ar 库管理程序选项

```
Usage: kf32-ar [emulation options] [-]{dmpqrstx}[abcDfilMNoPsSTuvV]
[--plugin <name>] [member-name] [count] archive-file file...
      kf32-ar -M [<mri-script>]
commands:
  d          - delete file(s) from the archive
  m[ab]      - move file(s) in the archive
  p          - print file(s) found in the archive
  q[f]       - quick append file(s) to the archive
  r[ab][f][u] - replace existing or insert new file(s) into the archive
  s          - act as ranlib
  t          - display contents of archive
  x[o]       - extract file(s) from the archive
command specific modifiers:
  [a]        - put file(s) after [member-name]
  [b]        - put file(s) before [member-name] (same as [i])
  [D]        - use zero for timestamps and uids/gids
  [U]        - use actual timestamps and uids/gids (default)
  [N]        - use instance [count] of name
  [f]        - truncate inserted file names
  [P]        - use full path names when matching
  [o]        - preserve original dates
  [u]        - only replace files that are newer than current archive
contents
generic modifiers:
  [c]        - do not warn if the library had to be created
  [s]        - create an archive index (cf. ranlib)
  [S]        - do not build a symbol table
  [T]        - make a thin archive
  [v]        - be verbose
  [V]        - display the version number
  @<file>    - read options from <file>
  --target=BFDNAME - specify the target object format as BFDNAME
optional:
  --plugin <p> - load the specified plugin
emulation options:
  No emulation specific options
```

## 5. 附录 B: kf32-strip 信息移除程序选项

```
Usage: kf32-strip <option(s)> in-file(s)
Removes symbols and sections from files
The options are:
  -p --preserve-dates          Copy modified/access timestamps to the
output
  -D --enable-deterministic-archives
                                Produce deterministic output when
stripping archives
  -U --disable-deterministic-archives
```

	Disable -D behavior (default)
<b>-R --remove-section=&lt;name&gt;</b>	<b>Remove section &lt;name&gt; from the output</b>
<b>-s --strip-all</b>	Remove all symbol and relocation information
<b>-g -S -d --strip-debug</b>	<b>Remove all debugging symbols &amp; sections</b>
--strip-dwo	Remove all DWO sections
--strip-unneeded	Remove all symbols not needed by relocations
--only-keep-debug	Strip everything but the debug information
<b>-N --strip-symbol=&lt;name&gt;</b>	Do not copy symbol <name>
<b>-K --keep-symbol=&lt;name&gt;</b>	Do not strip symbol <name>
--keep-file-symbols	Do not strip file symbol(s)
<b>-w --wildcard</b>	Permit wildcard in symbol comparison
<b>-x --discard-all</b>	Remove all non-global symbols
<b>-X --discard-locals</b>	Remove any compiler-generated symbols
<b>-v --verbose</b>	List all object files modified
<b>-V --version</b>	Display this program's version number
<b>-h --help</b>	Display this output
--info	List object formats & architectures supported
<b>-o &lt;file&gt;</b>	<b>Place stripped output into &lt;file&gt;</b>

## 6. 附录 C：命令行脚本参考源码

```
@echo off
cd /d %~dp0

echo =====
echo 获取工具路径...
:ADMIN
if /I "%PROCESSOR_ARCHITECTURE%"=="AMD64" goto AMD64
if /I "%PROCESSOR_ARCHITECTURE%"=="AMD64" goto AMD64
rem =====
FOR /F "tokens=2*" %A IN ('reg query HKEY_LOCAL_MACHINE\SOFTWARE\ChipON\ChipONCC32
/v Path ') DO (set pInstallDir=%B)
PATH=%pInstallDir%\ccr1_issue\bin;%pInstallDir%\kf32\bin;%PATH%
rem =====
goto END
:AMD64
rem =====
FOR /F "tokens=2*" %A IN ('reg query
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ChipON\ChipONCC32 /v Path ') DO (set
pInstallDir=%B)
PATH=%pInstallDir%\ccr1_issue\bin;%pInstallDir%\kf32\bin;%PATH%
rem
#####
:END
rem kf32-ar rcs libA.a libA.o *.o 手动常规添加使用
rem kf32-ar d libA.a libA.o *.o 删除指定库文件
rem kf32-ar -h 查看更多选项
rem for %I in (*.o) do kf32-ar rcs %~n1.a %~n1.o 遍历单个文件单个库使用
rem
#####
echo 获取工具路径:%pInstallDir%
```

```
set CYGWIN=nodosfilewarning
echo =====
set libName= lib~nl.a
echo 目标库输出名: %libName%

setlocal enabledelayedexpansion

for %%i in (*.o) do kf32-strip -g %%i

for %%i in (*.o) do set libFiles=!libFiles! %%i
echo 目标库文件更新: %libFiles%
echo =====
pause
rem if exist %libName% del %libName% rem 不存在时创建, 存在时仅对对应的 o 文件进行替换。
全新时可以先删除的语句

echo kf32-ar [rcs/d] %libName% %libFiles%
REM 更新命令
kf32-ar rcs %libName% %libFiles%
REM 删除名
rem if exist %libName% kf32-ar d %libName% %libFiles%

echo =====
if exist %libName% echo 库文件: %libName%

echo 库包含:
if exist %libName% kf32-ar t %libName%
echo =====
echo 操作已完成
echo =====
pause
```