

KF32 系列

编译工具使用说明 V1.2

2021 年 08 月

目录

| | |
|---|----|
| 目录 | 2 |
| 修改记录 | 3 |
| 1. 工具的介绍 | 4 |
| 1.1 发布形式 | 4 |
| 1.2 文件结构 | 4 |
| 1.3 主要工具与使用说明 | 4 |
| 1.3.1 <i>gmake.exe</i> | 5 |
| 1.3.2 <i>kf32-g++.exe</i> | 5 |
| 1.3.3 <i>kf32-gcc.exe</i> | 5 |
| 1.3.4 <i>kf32-as.exe</i> | 5 |
| 1.3.5 <i>kf32-ld.exe</i> | 5 |
| 1.3.6 <i>kf32-objdump.exe</i> | 6 |
| 1.3.7 <i>kf32-objcopy.exe</i> | 6 |
| 1.3.8 <i>kf32-readelf.exe</i> | 6 |
| 1.3.9 <i>kf32-size.exe</i> | 6 |
| 1.3.10 <i>kf32-ar.exe</i> | 6 |
| 1.3.11 <i>kf32-gdb.exe</i> | 7 |
| 2. 基于 KF32 IDE 环境的应用举例 | 8 |
| 2.1 MAKEFILE: | 8 |
| 2.2 OBJECTS.MK | 10 |
| 2.3 SOURCES.MK | 10 |
| 2.4 SUBDIR.MK | 10 |
| 2.5 USB-FS-DEVICE_DRIVER /SUBDIR.MK | 11 |
| 2.6 DEBUG 与 RELEASE 选择差异 | 13 |
| 2.7 KUNGFU32 差异化参数应用举例说明 | 13 |
| 2.7.1 编译器 <i>kf32-gcc</i> : | 13 |
| 2.7.2 汇编器 <i>kf32-as</i> : | 14 |
| 2.7.3 链接器 <i>kf32-ld</i> : | 14 |
| 2.7.4 库管理 <i>kf32-ar</i> : | 15 |

修改记录

| 序号 | 日期 | 版本 | 变更及说明 | 其他 |
|----|------------|------|---|----|
| 1 | 2020-09-12 | V1.0 | 初稿 | |
| 2 | 2021-8-23 | V1.1 | 细节更新 | |
| 3 | 2022-5-5 | V1.2 | 文件结构更新 更新 gdb 调用串口参数 工具章节修正 部分其他细节调整 6-29 部分错别字修正 | |
| 4 | 2026-6-11 | V1.3 | 工具链组织关系更新 | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |

1. 工具的介绍

1.1 发布形式

跟随 ChipON IDE KF32 软件部分更新，位于安装目录下 ChiponCC32 文件夹内，或基于 Zip 压缩包交付。该文档基于 Windows 版本进行说明，Linux 版本的使用仍然适用(路径格式差异)。

1.2 文件结构

| | |
|---------------------------|--|
| ccr1_issue | 当前发布型号版本，编号为 r1 |
| ccr1_issue\lib | 存放芯片算法库与系统最小相关库，如 libmath.a 或 基于标准 newlibs 和 libstdc++-v3 的库 libc libm libstdc++ libsupc++ |
| ccr1_issue\scripting | 对于版本下链接器脚本文件 |
| ccr1_issue\bin | 基本工具路径，如 kf32-gcc.exe kf32-ar.exe |
| ccr1_issue\kf32\bin | 汇编链接等工具组的工具，如 ar.exe as.exe ld.exe |
| chipregister | 服务 IDE 软件的芯片寄存器格式描述文件集合 |
| include | 芯片 C 语言和汇编语言编写的寄存器操作头文件 |
| include\Sys | 最小适配系统相关头文件，如 math、malloc 等 |
| include\Std_Sys | 基于标准 newlibs 组织的 c 头文件 |
| include\Std_Sys\c++\4.7.0 | 基于标准 libstdc++-v3 的 c++头文件 |
| common | 通用工具路径，如 gmake, rm 等 |

在 include\Sys 和 ccr1_issue\lib 提供基于最小需要的标准库与运行库
在 include\Std_Sys 和 ccr1_issue\lib 提供基于 newlibs 与 libstdc++-v3 的嵌入式适配标准 c 库和 c++库支持。

最小库打印基于 usart 映射的轻量级打印,标准库支持系统级的功能和 c++容器类支持,打印基于文件系统适配，同时需要使用 malloc 方法支持。

1.3 主要工具与使用说明

更多工具的选项可见 gnu 工具描述。gcc 可参考 gcc4.7.0, binutils 可参考 2.24, gdb 可参考 7.8。

1.3.1 gmake.exe

基于 makefile 管理编译的工具，实际即为 make 的重命名，当前使用版本使用 3.81.

1.3.2 kf32-g++.exe

C++编译器

使用格式 `{COMMAND} {FLAGS} {INPUTS} {OUTPUT_FLAG} {OUTPUT}`

如 `kf32g++ -I"C:\Program Files (x86)\ChipON IDE\KungFu32\ChiponCC32\include" -I"C:\Program Files (x86)\ChipON IDE\KungFu32\ChiponCC32\include\Sys" -save-temps -fno-builtin-printf -c -funsigned-char -Wa,--kf32-arch=kf32r, -I"C:\Program Files (x86)\ChipON IDE\KungFu32\ChiponCC32\include" -ffunction-sections -fdata-sections -D"KF32F130GQS" -std=gnu++11 -O2 ../main.cpp -o "main.o"`

1.3.3 kf32-gcc.exe

C 编译器驱动，也可用于汇编和链接输出目标文件。

格式: `{COMMAND} {FLAGS} {INPUTS} {OUTPUT_FLAG} {OUTPUT}`

如 `kf32cc -I"C:\Program Files (x86)\ChipON IDE\KungFu32\ChiponCC32\include" -I"C:\Program Files (x86)\ChipON IDE\KungFu32\ChiponCC32\include\Sys" -save-temps -fno-builtin-printf -c -funsigned-char -Wa,--kf32-arch=kf32r, -I"C:\Program Files (x86)\ChipON IDE\KungFu32\ChiponCC32\include" -ffunction-sections -fdata-sections -D"KF32F130GQS" -std=gnu++11 -O2 ../main.c -o "main.o"`

1.3.4 kf32-as.exe

汇编器

格式: `{COMMAND} {FLAGS} {INPUTS} {OUTPUT_FLAG} {OUTPUT}`

如 `kf32-as -MD ../main.d --kf32-arch=kf32r -I"d:/workspace32/demo" -I"C:\Program Files (x86)\ChipON IDE\KungFu32\ChiponCC32\include" --defsym KF32A250GQS=1 "../main.asm" -o "main.o"`

1.3.5 kf32-ld.exe

链接器，

格式: `{COMMAND} {INPUTS} {FLAGS} -o "${ProjName}.elf" -Map "${ProjName}.map"`

如: `kf32-ld ./_config/startup.o ./_config/vector.o ./kf_it.o ./main.o -L"C:\Program Files (x86)\ChipON IDE\KungFu32\ChipONCC32\ccr1_issue\lib" -L"d:/workspace32/demo" -lIqmath-R1 -lmath -lio -lstring -lstdlib -lctype -lcrtv1 -T"C:\Program Files (x86)\ChipON IDE\KungFu32\ChipONCC32\ccr1_issue\scripting\KF32F350MQV.ld"`

```
--kf32-autoihex --kf32-arch=kf32r --kf32-z --gc-sections -o "demo.elf" -Map "demo.map"
```

注：链接器的输出文件为 elf 格式的目标机文件，--kf32-autoihex 选项实现自动调用 objcopy 的输出可烧录 hex 文件。

库文件应在编译汇编输出 o 文件后面添加，若库使用其他库方法，该库应优先添加。

若链接使用 kf2-gcc，默认会传递库，这些库在路径\lib\gcc\kf32\4.7.0 下的 libgcc.a ccr0.o crti.o crtn.o crtbegin.o crtend.o，若不需要可添加选项 -nostdlib（推荐）。使用 kf32-gcc 下链接器专有选项请使用 -Wl,option 或 -Xlinker option，

最小系统依赖库分布在 -lmath -lio -lstring -lstdlib -lctype -lcrtvx 中。

系统级支持的库分布在 -lgcc -lc -lm -lstdc++ -lsupc++ 中。

库传达具有依赖关系，推荐使用如 --start-group -lgcc -lc -lm -lstdc++ -lsupc++ --end-group 的建立组。

1.3.6 kf32-objdump.exe

elf 文件反汇编程序，如：

```
kf32-objdump -G TestElf1.elf > TestElf1.lst
```

```
kf32-objdump -S -l [-z] [--kf32-arch=kf32r] --section=.text --section=.data  
--section=.bss project.elf > project.lst
```

1.3.7 kf32-objcopy.exe

elf 文件输出可执行下载格式程序。

如：kf32-objcopy -O ihex project.elf project.hex

支持输出格式 ihex、binary 和 srec (S19)。链接器可通过参数的自动调用该程序实现 elf 文件到 hex 文件的输出。

1.3.8 kf32-readelf.exe

elf 文件或 o 文件的信息查看器。

1.3.9 kf32-size.exe

统计项目的 flash 和 ram 的使用情况，更多信息见软件适配输出。

如：kf32-size XXX.elf

输出：

| <u>text</u> | <u>data</u> | <u>bss</u> | <u>eep</u> | <u>dec</u> | <u>hex</u> | <u>filename</u> |
|-------------|-------------|------------|------------|------------|------------|-----------------|
| 8684 | 0 | 0 | 4096 | 12780 | 31ec | XXX.elf |

1.3.10 kf32-ar.exe

归档文件管理程序，即库程序管理程序。

kf32-ar rcs libA.a libA.o [*o ...] 基于对象文件集创建或添加到库

kf32-ar d libA.a libA.o [*o ...] 删除库中对象文件

kf32-ar t libA.a 查看库内容

主要用于编译汇编输出对象文件 o 的库发布管理，库文件后缀为 a，以 lib 开头的库的链接器添加中可以简写，如库 libFunc.a 的 -lFunc 或 -llibFunc.a

1.3.11 kf32-gdb.exe

硬件仿真调试器。

1、启动 gdb.exe XX.elf

2、适配配置选项

```
set confirm off
set pagination off
set step-mode off
set print elements 1024
set print repeats unlimited
```

3、启动硬件仿真目标机：

--kft-code=[234]对应 kungfu32 芯片内部编码，--kfd-code=[0123] 启动调试的方法，0 复位并运行到 main，1 复位并运行到 startup，2 关闭看门狗的当前程序暂停与调试，3 预留。编程调试器设备有效端口号为 COM1-COM256

示例:target kf32remote

COM11

--kfvoltage=3.0 --kfd-speed=2 --kft-code=2 --kfd-code=0

--kfinspect=kf32r --kfprint

--arch="C:\Program Files (x86)\ChipON IDE\KungFu32\ChipONCC32\ccr1_issue\scripting\KF32F350MQV.def"

软件仿真不具有外设功能，仅支持进行代码逻辑的调试处理，请软件仿真的命令行为：

```
target kf32sim --arch="C:\Program Files (x86)\ChipON IDE\KungFu32\ChipONCC32\ccr1_issue\scripting\KF32F350MQV.def" --kfinspect=kf32r --kfprint
```

注：当前不需要框架文件的初始化差异配置，即--arch=给取任意空文件即可。

4、加载目标机

load

辅助工具获取程序的 pid 码，可供发送暂停用，如 kill.exe -2 8892，即 Ctrl+C 的暂停，另可“taskkill /f /t /im kf32-gdb.exe”强制退出调试器。

6、断点 break main

7、运行到断点：r，[后续运行使用命令 c，即 continue]

8、其他调试支持命令

调试中断允许与禁止：intctl on intctl off 集成在单步与运行功能下

查看通用寄存器：info registers info registers r0 r1

修改通用寄存器：set var \$r0=0x1234567

查看指定内存地址：x /9w 0x500

查看变量起始地址的值：x /9w arr

修改指定地址的值: `set *0x10000000=0x12345678 set *(short *)0x10000000=0x1234`
查看函数的 flash 空间: `x /16w main`
查看全局变量: `print variable`
查看局部变量: `info locals`
修改局部变量: `set x=0x1`
函数参数查询: `info args`
断点设定: `b main.c:144`
查询断点: `info breakpoint`
删除断点: `delete delete 2`
GDB 通过获取 `ctrl+C` 信号来识别用户暂停命令。
运行: `continue`
单步: `stepi step`
单步跳过: `nexti next`
单步返回: `finish`
复位: `kfreset`
退出: `quit`

2. 基于 KF32 IDE 环境的应用举例

举例项目 KungFu32_ProDebug_WinUSB, 工具软件放置在 `C:/Program Files (x86)/ChipON IDE/KungFu32` 目录下的 `ChiponCC32`。

文档文件夹结构:

User, User\inc,
StdPeriph_Driver, StdPeriph_Driver\inc,
_config,
USB-FS-Device_Driver, USB-FS-Device_Driver\inc.

基于 IDE 生成 `gmake` 需要的依赖关系与参数应用说明如下, 具体参加实际项目输出。

注 1: 若文件结构不再调整, 可使用 `ide` 预输出 `makefile` 及相关文件, 随后通过 `common` 目标下脚本 `bat` 文件维护后进行调用, 也可使用基于 `python` 的脚步生成 `makfile` (需维护内容) 并构建。

注 2: 命令行机制的 `makefile` 针对参数的长度具有约束, `cmd` 命令行为 8192 字符, 通过 `createprocess` 如 `make` 的长度限制为 32767 字符, 通过 `shellexecute` 约 2048 字符。

2.1 Makefile:

```
#####
# 自动生成的文件。或手动维护结果!
#####
```



```
-include ../makefile.init

RM := rm -rf

# All of the sources participating in the build are defined here
-include sources.mk
-include _config/subdir.mk
-include User/subdir.mk
-include USB-FS-Device_Driver/subdir.mk
-include StdPeriph_Driver/subdir.mk
-include subdir.mk
-include objects.mk

-include $(C_DEPS)

-include ../makefile.defs

# Add inputs and outputs from these tool invocations to the build variables

# 所有目标
all: KungFu32_ProDebug_WinUSB.hex

# 工具调用
KungFu32_ProDebug_WinUSB.hex: $(OBJS) $(USER_RELS)
    @echo 'Building target: $@'
    @echo 'Invoking: C Linker Release'
    kf32-ld $(OBJS) $(USER_RELS) $(LIBS) -L"C:/Program Files (x86)/ChipON
IDE/KungFu32/ChiponCC32/lib/ccr1_issue"
    -L"D:\workspace32\KungFu32_ProDebug_WinUSB" -lIQmath-R1
    -lSeriesDIServices -lmath -lio -lstring -lstdlib -lctype -lcrtv1
    -T"../KF32F341IQS_9KWinUSB.ld" --gc-sections -o
    "KungFu32_ProDebug_WinUSB.elf" -Map "KungFu32_ProDebug_WinUSB.map"
    @echo 'Finished building target: $@'
    @echo ' '

# 其他目标
clean:
    -$(RM) $(OBJS) $(EXECUTABLES) $(C_DEPS) KungFu32_ProDebug_WinUSB.hex
    -@echo ' '
```

```
.PHONY: all clean dependents
.SECONDARY:

    -include ../makefile.targets
```

2.2 objects.mk

```
#####
# 自动生成的文件。或手动维护结果！
#####

USER_OBJS :=

LIBS :=
```

2.3 sources.mk

```
#####
# 自动生成的文件。或手动维护结果！
#####

O_SRCS :=
C_SRCS :=
OBJ_SRCS :=
C_DEPS :=
OBJS :=
EXECUTABLES :=

# Every subdirectory with source files must be described here
SUBDIRS := \
. \
_config \
User \
USB-FS-Device_Driver \
StdPeriph_Driver \
```

2.4 subdir.mk

```
#####
# 自动生成的文件。或手动维护结果！
```

```
#####
```

```
# Add inputs and outputs from these tool invocations to the build variables
```

```
C_SRCS += \  
../flash.c \  
../main.c \  
../system_init.c
```

```
OBJS += \  
./flash.o \  
./main.o \  
./system_init.o
```

```
C_DEPS += \  
./flash.d \  
./main.d \  
./system_init.d
```

```
# Each subdirectory must supply rules for building sources it contributes
```

```
%.o: ../%.c
```

```
    @echo 'Building file: $<'
```

```
    @echo 'Invoking: C Compiler Release'
```

```
    kf32-gcc -MMD -I"D:\workspace32\KungFu32_ProDebug_WinUSB"  
-I"D:\workspace32\KungFu32_ProDebug_WinUSB\USB-FS-Device_Driver\inc"  
-I"D:\workspace32\KungFu32_ProDebug_WinUSB\StdPeriph_Driver\inc"  
-I"D:\workspace32\KungFu32_ProDebug_WinUSB\User\inc" -I"C:/Program  
Files (x86)/ChipON IDE/KungFu32/ChiponCC32/include" -I"C:/Program Files  
(x86)/ChipON IDE/KungFu32/ChiponCC32/include/Sys" -save-temps  
-fno-builtin-printf -fno-builtin-fprintf -fno-builtin-fputs -c  
-funsigned-char -fsigned-bitfields -Wa,--kf32-arch=kf32r,-I"C:/Program  
Files (x86)/ChipON IDE/KungFu32/ChiponCC32/include"  
-ffunction-sections -fdata-sections -D"KF32F341IQS"  
-Wno-packed-bitfield-compat -std=gnu99 -O2 $< -o "$@"  
    @echo 'Finished building: $<'
```

2.5 USB-FS-Device_Driver /subdir.mk

```
#####
```

```
# 自动生成的文件。或手动维护结果！
```

```
#####
```

```
# Add inputs and outputs from these tool invocations to the build variables
C_SRCS += \
../USB-FS-Device_Driver/usb_core.c \
../USB-FS-Device_Driver/usb_init.c \
../USB-FS-Device_Driver/usb_int.c \
../USB-FS-Device_Driver/usb_mem.c \
../USB-FS-Device_Driver/usb_regs.c \
../USB-FS-Device_Driver/usb_sil.c

OBJS += \
../USB-FS-Device_Driver/usb_core.o \
../USB-FS-Device_Driver/usb_init.o \
../USB-FS-Device_Driver/usb_int.o \
../USB-FS-Device_Driver/usb_mem.o \
../USB-FS-Device_Driver/usb_regs.o \
../USB-FS-Device_Driver/usb_sil.o

C_DEPS += \
../USB-FS-Device_Driver/usb_core.d \
../USB-FS-Device_Driver/usb_init.d \
../USB-FS-Device_Driver/usb_int.d \
../USB-FS-Device_Driver/usb_mem.d \
../USB-FS-Device_Driver/usb_regs.d \
../USB-FS-Device_Driver/usb_sil.d

# Each subdirectory must supply rules for building sources it contributes
USB-FS-Device_Driver/%.o: ../USB-FS-Device_Driver/%.c
    @echo 'Building file: $<'
    @echo 'Invoking: C Compiler Release'
    kf32-gcc -MMD -I"D:\workspace32\KungFu32_ProDebug_WinUSB"
-I"D:\workspace32\KungFu32_ProDebug_WinUSB\USB-FS-Device_Driver\inc"
-I"D:\workspace32\KungFu32_ProDebug_WinUSB\StdPeriph_Driver\inc"
-I"D:\workspace32\KungFu32_ProDebug_WinUSB\User\inc" -I"C:/Program
Files (x86)/ChipON IDE/KungFu32/ChiponCC32/include" -I"C:/Program Files
(x86)/ChipON IDE/KungFu32/ChiponCC32/include/Sys" -save-temps
-fno-builtin-printf -fno-builtin-fprintf -fno-builtin-fputs -c
-funsigned-char -fsigned-bitfields -Wa,--kf32-arch=kf32r,-I"C:/Program
Files (x86)/ChipON IDE/KungFu32/ChiponCC32/include"
-ffunction-sections -fdata-sections -D"KF32F341IQS"
-Wno-packed-bitfield-compat -std=gnu99 -O2 $< -o "$@"
```

```
@echo 'Finished building: $<'
```

2.6 debug 与 release 选择差异

差异体现在编译器选项，即调试下`-O0`，正式下`-O2`，均传递`-gstabs+`。即默认任意模式均可调试。

2.7 KungFu32 差异化参数应用举例说明

2.7.1 编译器 `kf32-gcc`:

添加`-MMD`基于源文件名输出依赖文件，如输出`main.d`。内容如：

```
main.o: ../main.c ../system_init.h \  
C:/Program Files (x86)/ChipON\  
IDE/KungFu32/ChiponCC32/include/Sys/string.h \  
C:/Program Files (x86)/ChipON\  
IDE/KungFu32/ChiponCC32/include/Sys/stdint.h \  
C:/Program Files (x86)/ChipON\  
IDE/KungFu32/ChiponCC32/include/Sys/stddef.h \  
  
D:\workspace32\KungFu32_ProDebug_WinUSB\StdPeriph_Driver\inc/KF32F_BA  
SIC.h ...
```

添加头文件搜索路径

`-I"D:\workspace32\KungFu32_ProDebug_WinUSB"`，指向主项目路径，可以不引入子路径下相对路径访问实现

`-I"C:/Program Files (x86)/ChipON
IDE/KungFu32/ChiponCC32/include"` 型号头文件目录所在

`-I"C:/Program Files (x86)/ChipON
IDE/KungFu32/ChiponCC32/include/Sys"` 系统头文件目录所在

添加`-fno-builtin-printf -fno-builtin-fprintf -fno-builtin-fputs`

即对于的打印不映射到 `gcc` 的函数格式，意味着采用明面函数名的 KF32 库方法实现，如 `put` 方法不调用为 `fwrite` 方法。

添加`-Wa, --kf32-arch=kf32r`

即选择对应的芯片指令集，当前选择 `kf32r`。

`-save-temps` 保存过程文件

即预处理后的 `i` 文件和编译的汇编文件，如 `main.s`

2.7.2 汇编器 kf32-as:

输出依赖文件

-MD \$(dir \$@)\$(basename \$(notdir \$@)).d

指定芯片内核

--kf32-arch=kf32r

添加头文件搜索路径

-I"C:/Program Files (x86)/ChipON
IDE/KungFu32/ChiponCC32/include " 型号头文件目录所在

2.7.3 链接器 kf32-ld:

添加库路径

-L"C:/Program Files (x86)/ChipON
IDE/KungFu32/ChiponCC32/lib/ccr1_issue" 系统库路径
-L"D:\workspace32\KungFu32_ProDebug_WinUSB" 当前
项目下主目录下库路径

基本 KungFu 最小系统库

-lIQmath-R1 定点浮点库
-lSeriesDIServices 串口交换协议接口
-lmath -lio -lstring -lstdlib -lctype 系统相关的实现库
-lcrtv1 型号相关的必须库

指定脚本

-T"./KF32F341IQS_9KWinUSB.ld" 项目主目录的自定义
脚本
或
-T"C:/Program Files (x86)/ChipON
IDE/KungFu32/ChiponCC32/scripting/ccr1_issue/KF32F130GQ
T.ld 工具目录发布的默认型号脚本

死段优化选项

--gc-sections

指定输出

-o XXX.elf 机器可执行程序格式文件
-Map XXX.map 编译段与符号信息

辅助选项

--kf32-nodisassemble 不执行链接后反汇编动作
--kf32-z 反汇编连续的 0 也仍然输出。

校验和功能

基于范围计算并输出校验和值，详细见校验和使用说明文档
--with-checksum-fill=0xFF

```
--with-checksum-fill=0x00
--with-checksum-fill=0x55ff
--with-checksum-fill=0x5500    // 4 种模式选 1
支持 1、2、3 共 3 组计算配置
--with-checksumX=CRC-32
--with-checksumX=CRC-32/MPEG-2
--with-checksumX=SUM32
--with-checksumX=SIG-CODE      // 支持的算法选项

--with-checksum-addressX=Where1
--with-checksum-sizeX=Bytes
--with-checksum-out-addressX=Where2
```

2.7.4 库管理 **kf32-ar**:

创建并添加或添加库文件: kf32-ar.exe rcs libA.a libA.o

【libB.o】

删除指定库文件: kf32-ar.exe d libA.a libA.o 【libB.o】

查看更多选项: kf32-ar.exe --h