```
$(TARGETNAME).elf: $(LDFILE_MAIN)
    @$(PRINTRELEASE) $(TARGETNAME)_$@$(VARIANTSTR)
    @echo creating $@
    @$(ARMLINK) $(LDFLAGS_MAIN) $(ARFLAGSOPTION) $(LD_LINKLIB) $(LDLIBS) -o $@
    @$(PRINTRELEASE) /$(TARGETNAME)_$@$(VARIANTSTR)
```

生成tx.elf

> $@(ARMLINK)$(LDFLAGS_MAIN) $(ARFLAGSOPTION)$(LD_LINKLIB) $(LDLIBS) -o@

也就是

> arm-marvell-eabi-gcc -Wl,-Map,map.txt,-O,–sort-alignment -Wl,–cref -L ./Hal/6220/linker_common -Wl,–gc-sections -T ./Hal/6220/build/memory.ld @Application/ar.flags base_lib.a -lm -lc -o tx.elf

这里的链接的文件list是 `@Application/ar.flags`

见ar的manual

```
@file
   Read command-line options from file.  The options read are inserted in place of the original @file option.
   If file does not exist, or cannot be read, then the option will be treated literally, and not removed.

   Options in file are separated by whitespace.  A whitespace character may be included in an option by
   surrounding the entire option in either single or double quotes.  Any character (including a backslash)
   may be included by prefixing the character to be included with a backslash.  The file may itself contain
   additional @file options; any such options will be processed recursively.
```

gcc and gas都支持类似功能。

`@Application/ar.flags` 包含

```
@/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/ar.flags
@/home/walterzh/work2/LSP/ccsgit/r4-freertos/Hal/ar.flags
@/home/walterzh/work2/LSP/ccsgit/r4-freertos/Netstack/ar.flags
@/home/walterzh/work2/LSP/ccsgit/r4-freertos/Os/ar.flags
@/home/walterzh/work2/LSP/ccsgit/r4-freertos/Sys/ar.flags
```

而 `@/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/ar.flags`

```
@/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/ar.flags
@/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/init/ar.flags
```

而 `@/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/ar.flags`

```
/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/IgmpNetconn.o
/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/TcpEchoSrv.o
/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/TelnetSrv.o

/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/UdpEchoSrv.o
/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/WebServer.o

/home/walterzh/work2/LSP/ccsgit/r4-freertos/ApplicationMrvl/Test/monitor.o
```

这些ar.flags是怎么generated？

in Makefile-rules

```
FLAGSFILE    =   .flags
ASFLAGSFILE =   as$(FLAGSFILE)
CCFLAGSFILE =   cc$(FLAGSFILE)
ARFLAGSFILE =   ar$(FLAGSFILE)
```

generate `as.flags` , `cc.flags` and `ar.flags`

generated file	description
as.flags	用于gas
cc.flags	gcc options
ar.flags	传递给ar的obj file list

in Makefile-os

```
    ASFLAGSOPTION    =   @$(ASFLAGSFILE)
    CCFLAGSOPTION    =   @$(CCFLAGSFILE)
    ARFLAGSOPTION    =   @$(ARFLAGSPATH)$(ARFLAGSFILE)
```

in Makefile-rules

```
%.o :   %.asm $(ASFLAGSFILE)
    @echo AS $<
    @$(ARMASM) $(ASFLAGSOPTION) $< -o $@

%.o :   %.s $(ASFLAGSFILE)
    @echo AS $<
    @$(ARMASM) $(ASFLAGSOPTION) $< -o $@

%.o :   %.c $(CCFLAGSFILE)
    @echo CC $<
    @$(ARMCC) $(CCFLAGSOPTION) $< -o $@
```

比如.c ==> .o

```
gcc $(CCFLAGSOPTION) $< -o $@
gcc @$(CCFLAGSFILE) $< -o $@
```

```
%.o :   %.c $(CCFLAGSFILE)
    gcc @cc.flags $< -o $@
```

即当前目录下的.c文件使用本目录下的cc.flags文件中的compiler option来编译！

比如Os/freertos/cc.flags

```
-DOS_freertos -DNETSTACK_lwipv6 -DSTATUS_BMU_ISR_MODE            -DSYSTEM_HE
AP_SIZE="(40*1024)" -DAPP_HEAP_SIZE="(4*1024)" -DSYSTEM_STACK_SIZE="(1600)"
-DAPP_STACK_SIZE="(0x5000)" -DSTATE_CHANGE_ACTION_U0=1       -DSTATE_CHANGE_
ACTION_03=2      -DSTATE_CHANGE_ACTION_05=2      -DSTATE_CHANGE_ACTION_30=1
     -DSTATE_CHANGE_ACTION_50=1      -DNUM_ARP_OFFLOAD_ENTRIES=1 -DNUM_NS_OF
FLOAD_ENTRIES=2 -DNUM_WAKE_PATTERN_ENTRIES=32 -DNUM_IPV4_TCP_SYN_PATTERN_ENT
RIES=12 -DNUM_IPV6_TCP_SYN_PATTERN_ENTRIES=8 -DREPORT_WAKE_REASON=1 -DENABLE
_WAKE_ON_MAGIC_PACKET=1 -DENABLE_WAKE_ON_LINK_CONNECT=1 -DENABLE_WAKE_ON_LIN
K_DISCONNECT=1 -DENABLE_SECOND_FILTER_STAGE -DMRVL_BSD_SOCKETS_ENABLED -DTX_
ENABLE_FIQ_SUPPORT -DMV_OSNET_LAYER -DENABLE_IP_V6_SUPPORT -DLWIP_IPV6=1 -DE
NABLE_IGMP_SUPPORT -DARCH_6220 -DMODE_ -I../../Os/freertos/include -I../../O
s/freertos/portable/GCC/6220 -I../../Hal/6220/src -I../../Hal/inc -I../../mi
sc -I../../Sys/debug/ -I../../Sys/debug/ -I../../Sys/fifo/ -I../../Sys/filte
r/ -I../../Sys/filter/ -I../../Sys/hostcmd/ -I../../Sys/init/ -I../../Sys/jt
ag/ -I../../Sys/lan/ -I../../Sys/memory/ -I../../Sys/memory/ -I../../Sys/mem
ory/ -I../../Sys/memory/ -I../../Sys/memory/ -I../../Sys/memory/ -I../../Sys
/memory/ -I../../Sys/memory/ -I../../Sys/memory/ -I../../Sys/memory/ -I../..
/Sys/memory/ -I../../Sys/memory/ -I../../Sys/sdk/ -I../../Sys/sdk/ -I../../S
ys/sdk/ -I../../Sys/socket/ -I../../Sys/socket/ -I../../Sys/system/ -I../../
Sys/system/ -I../../Sys/system/ -I../../Sys/system/ -I../../Sys/system/ -I..
/../Sys/thread/ -I../../Sys/thread/ -I../../Sys/timer/ -I../../Sys/timer/ -I
../../Sys/time/ -I../../Sys/time/ -I. -I../../misc -O0 -mcpu=cortex-r4
 -g -ftabstop=4  -DDEBUG=1 -Wunreachable-code -Winit-self -Wimplicit -Wfloat
-equal -Wshadow -Wall -Wextra -Wpadded  -Wswitch-default -Wswitch-enum -O0
-Wno-unknown-pragmas -fno-builtin-memcpy -mlittle-endian -std=iso9899:1999 -
mno-unaligned-access -ffunction-sections -fdata-sections -MD -c -march=armv7
-r -mtune=cortex-r4 -mthumb -mthumb-interwork -DTHUMB_INTERWORK -c
```

generate ar.flags

```
$(ARFLAGSFILE):    $(TL_ARFLAGS) $(SPLINT_LOGS) $(CLANG_LOGS)
    @$(PRINTRELEASE) $(TARGETNAME)_$@$(VARIANTSTR)
    @echo creating $@
    @echo \
        $(patsubst %, $(VIADIR)/%/$@, $(filter $(filter-out $(IGNORESUBMODULES), $(ALLSUBMODULES)),$^)) \
        $(patsubst %.o, "$(CURDIR)"/%.o, $(filter $(ALLOBJS),$^)) \
        > $@
    @$(PRINTRELEASE) /$(TARGETNAME)_$@$(VARIANTSTR)
```

上面的 `> $@` 就是生成ar.flags

generate as.flags and cc.flags

```
$(ASFLAGSFILE):    $(DEPEND_MAKEFILES)
    @$(PRINTRELEASE) $(TARGETNAME)_$@$(VARIANTSTR)
    @echo creating $(ASFLAGSFILE)
    @$(PRINTCMD) AS flags: $(ASBINFLAGS)
    @echo $(ASBINFLAGS) > $(ASFLAGSFILE)
    @$(PRINTRELEASE) /$(TARGETNAME)_$@$(VARIANTSTR)

$(CCFLAGSFILE):    $(DEPEND_MAKEFILES)
    @$(PRINTRELEASE) $(TARGETNAME)_$@$(VARIANTSTR)
    @echo creating $(CCFLAGSFILE)
    @$(PRINTCMD) CC flags: $(CCBINFLAGS)
    @echo '$(CCBINFLAGS) $(CC_COMPILEONLY)' > $(CCFLAGSFILE)
    @$(PRINTRELEASE) /$(TARGETNAME)_$@$(VARIANTSTR)
```

echo $(ASBINFLAGS) > as.flagsecho'(CCBINFLAGS) $(CC_COMPILEONLY)' > cc.flags

in Makefile-rules

```
################################################################################
###
# set up default ASBINFLAGS & CCBINFLAGS if not set
################################################################################
###
ifndef  ASBINFLAGS
ASBINFLAGS  =   $(ASM_FLAGS)
endif   # ASBINFLAGS

ifndef  CCBINFLAGS
CCBINFLAGS  =   $(CC_FLAGS)
endif   # CCBINFLAGS
```

这是默认的设置

in Makefile-rules

```
1.  ################################################################################
    ###
2.  # setup default asm & cc flags
3.  ################################################################################
    ###
4.  ASM_COMMON  +=      \
5.      $(ASM_DEBUG)   \
6.      $(ENDIAN_AS_PARAM) \
7.      $(CPUTYPE)                # default ASM flags for all cpu modes
8.
9.  ASM_FLAGS   =       \
10.     $(ASM_COMMON) -march=armv7-r              # default ASM flags for auto
    cpu mode
11.
12. ASM_FLAGS_NORMAL_FORCED =   \
13.     $(ASM_NORMALMODE)      \
14.     $(ASM_COMMON)             # default ASM flags for normal cpu mode
15.
16. ASM_FLAGS_THUMB_FORCED  =   \
17.     $(ASM_THUMBMODE)       \
18.     $(ASM_COMMON)             # default ASM flags for thumb cpu mode
19.
20. CC_FLAGS    +=          \ASBINFLAGS
21.     $(CC_DFLAGS)       \
22.     $(CC_IFLAGS)       \
23.     $(CC_OFLAGS)       \
24.     $(CPUTYPE)         \
25.     $(CC_DEBUG)            \
26.     $(ENDIAN_CC_PARAM)     \
27.     $(CC_CMODE)            \
28.     $(CC_AUTODEP)      \
29.     $(CC_COMPILEONLY)      \
30.     -march=armv7-r -mtune=cortex-r4
```

CC_DFLAGS ==> -DXXX, define MACRO

CC_IFLAGS ==> -IXXX, include header file directory

CC_OFLAGS ==> -OX, OPTIMIZATION

CPUTYPE

in Makefile-os

```
1.  CPUTYPE     =   -mcpu=$(TARGETCPU)
```

CC_DEBUG ==> debug, control debug level in Makefile-os

ENDIAN_CC_PARAM ==> ENDIAN_CC_PARAM = -mlittle-endian in Makefile-os

CC_CMODE ==> c compiler's other options

```
    # empty for compiler defaults
    C_MODE          ?=  c99

ifeq ($(C_MODE), ansi)
    CC_CMODE        =   -ansi
else ifeq ($(C_MODE), c90)
    CC_CMODE        =   -std=iso9899:1990
else ifeq ($(C_MODE), c99)
    CC_CMODE        =   -std=iso9899:1999
else ifeq ($(C_MODE), c11)
    CC_CMODE        =   -std=iso9899:2011
else
    # proceed with compiler default settings
    CC_CMODE        =
endif

    # extend c mode
#   CC_CMODE        +=  --param case-values-threshold=9999
    CC_CMODE        +=  -mno-unaligned-access

 CC_CMODE       +=  -ffunction-sections -fdata-sections
```

CC_AUTODEP = -MD
"used to generate a dependency output file"

CC_COMPILEONLY = -c
只是生成.o文件

Makefile.rules中的macro很多在Makefile-os中定义，所以Makefile.os用于定制化Makefile.rules
而各个source directory中的Makefile-cfg则用于定制特定与该目录的code generating.

for example
in r4-freertos/Hal/6220/src/Makefile

```
# set the relative path to the project root dir
PROJECTDIR  :=  ../../..

# include global project config
include $(PROJECTDIR)/Makefile-cfg            ①

# include local module project config
include $(PROJECTDIR)/Hal/$(ARCH)/Makefile-cfg    ②

# include the global rule set
include $(PROJECTDIR)/Makefile-rules           ③
```

①
Global cfg

②
使用local cfg中的macros setting修改global cfg中的setting
③
build rules中用到的macros就是上面①②中的macros.

Makefile.rules include Makefile-os
即修改Makefile-os中的macros就可以customize Makefile.rules

那些source文件会被编译?

in Makefile-rules

```
1.  ################################################################################
    ###
2.  # set up default SRC_C, SRC_ASM & SRC_S if not set
3.  ################################################################################
    ###
4.  ifndef  SRC_C
5.  SRC_C   =   $(wildcard *.c)
6.  endif   # SRC_C
7.
8.  ifndef  SRC_ASM
9.  SRC_ASM =   $(wildcard *_$(TOOLTYPE).asm)
10. endif   # SRC_ASM
11.
12. ifndef  SRC_S
13. SRC_S   =   $(wildcard *_$(TOOLTYPE).s)
14. endif   # SRC_S
```

即如果没有指定SRC_C macro,则compile当前目录下所有.c file

```
1.  ################################################################################
    ###
2.  # set up default OBJ_C, OBJ_ASM & OBJ_S if not set
3.  ################################################################################
    ###
4.  ifndef  OBJ_C
5.  OBJ_C   =   $(sort $(SRC_C:.c=.o))
6.  endif   # OBJ_C
7.
8.  ifndef  OBJ_ASM
9.  OBJ_ASM =   $(sort $(SRC_ASM:.asm=.o))
10. endif   # OBJ_ASM
11.
12. ifndef  OBJ_S
13. OBJ_S   =   $(sort $(SRC_S:.s=.o))
14. endif   # OBJ_S
```

```
1.  ##############################################################################
    ###
2.  # set up default ALLOBJS if not set
3.  ##############################################################################
    ###
4.  ifndef  ALLOBJS
5.  ALLOBJS =   $(OBJ_ASM) $(OBJ_S) $(OBJ_C)
6.  endif   # ALLOBJS
```

这样在对应的source directory下的Makefile有2种情况

1. 不指定 `SRC_C` macro
   for example

in Hal/6220/88pa6220/a0/config/Makefile

```
1.  # set the relative path to the project root dir
2.  PROJECTDIR  :=  ../../../../..
3.
4.  # include global project config
5.  include $(PROJECTDIR)/Makefile-cfg
6.
7.  # include local module project config
8.  include $(PROJECTDIR)/Hal/$(ARCH)/Makefile-cfg
9.
10. # include the global rule set
11. include $(PROJECTDIR)/Makefile-rules
```

没有指定 `SRC_C` macro，所以该目录下的all .c files都编译成了.o files

1. 在Makefile中指定 `SRC_C` macro

for example
in Hal/6220/cpu/cortex_r4-r1p3/src

```
1.  # set the relative path to the project root dir
2.  PROJECTDIR  :=  ../../../../..
3.
4.  # include global project config
5.  include $(PROJECTDIR)/Makefile-cfg
6.
7.  # include local module project config
8.  include $(PROJECTDIR)/Hal/$(ARCH)/Makefile-cfg
9.
10. # include the global rule set
11. include $(PROJECTDIR)/Makefile-rules
12.
13. SRC_C   =                           \
14.     cortex_r4.o
```