```
# WinAVR Sample makefile written by Eric B. Weddington, Jörg Wunsch, et al.
# Released to the Public Domain
# Please read the make user manual!
#
# Additional material for this makefile was submitted by:
#  Tim Henigan
#  Peter Fleury
#  Reiner Patommel
#  Sander Pool
#  Frederik Rouleau
#  Markus Pfaff
#
# On command line:
#
# make all = Make software.
#
# make clean = Clean out built project files.
#
# make coff = Convert ELF to AVR COFF (for use with AVR Studio 3.x or VMLAB).
#
# make extcoff = Convert ELF to AVR Extended COFF (for use with AVR Studio
#                4.07 or greater).
#
# make filename.s = Just compile filename.c into the assembler code only
#
# To rebuild project do "make clean" then "make all".
#


# MCU name
MCU = atmega8

# Output format. (can be srec, ihex, binary)
FORMAT = ihex

# Target file name (without extension).
TARGET = main

# Optimization level, can be [0, 1, 2, 3, s]. 0 turns off optimization.
# (Note: 3 is not always the best optimization level. See avr-libc FAQ.)
OPT = 0

# List C source files here. (C dependencies are automatically generated.)
SRC = main.c

# If there is more than one source file, append them above, or modify and
# uncomment the following:
#SRC += foo.c bar.c

# You can also wrap lines by appending a backslash to the end of the line:
#SRC += baz.c \
#xyzzy.c

# List Assembler source files here.
# Make them always end in a capital .S.  Files ending in a lowercase .s
# will not be considered source files but generated files (assembler
# output from the compiler), and will be deleted upon "make clean"!
# Even though the DOS/Win* filesystem matches both .s and .S the same,
# it will preserve the spelling of the filenames, and gcc itself does
# care about how the name is spelled on its command-line.
ASRC =


# List any extra directories to look for include files here.
#     Each directory must be seperated by a space.
EXTRAINCDIRS =


# Optional compiler flags.
#  -g:        generate debugging information (for GDB, or for COFF conversion)
#  -O*:       optimization level
#  -f...:     tuning, see gcc manual and avr-libc documentation
#  -Wall...:  warning level
#  -Wa,...:   tell GCC to pass this to the assembler.
#    -ahlms:  create assembler listing
CFLAGS = -O$(OPT) \
```

```
-funsigned-char -funsigned-bitfields -fpack-struct -fshort-enums \
-Wall -Wstrict-prototypes \
-Wa,-adhlns=$(<:.c=.lst) \
$(patsubst %,-I%,$(EXTRAINCDIRS))


# Set a "language standard" compiler flag.
#   Unremark just one line below to set the language standard to use.
#   gnu99 = C99 + GNU extensions. See GCC manual for more information.
#CFLAGS += -std=c89
#CFLAGS += -std=gnu89
#CFLAGS += -std=c99
CFLAGS += -std=gnu99

# Optional assembler flags.
#  -Wa,...:   tell GCC to pass this to the assembler.
#  -ahlms:    create listing
#  -gstabs:   have the assembler create line number information; note that
#             for use in COFF files, additional information about filenames
#             and function names needs to be present in the assembler source
#             files -- see avr-libc docs [FIXME: not yet described there]
ASFLAGS = -Wa,-adhlns=$(<:.S=.lst),-gstabs

# Optional linker flags.
#  -Wl,...:   tell GCC to pass this to linker.
#  -Map:      create map file
#  --cref:    add cross reference to  map file
LDFLAGS = -Wl,-Map=$(TARGET).map,--cref

# Additional libraries

# Minimalistic printf version
#LDFLAGS += -Wl,-u,vfprintf -lprintf_min

# Floating point printf version (requires -lm below)
#LDFLAGS += -Wl,-u,vfprintf -lprintf_flt

# -lm = math library
LDFLAGS += -lm

# Define directories, if needed.
DIRAVR = c:/winavr
DIRAVRBIN = $(DIRAVR)/bin
DIRAVRUTILS = $(DIRAVR)/utils/bin
DIRINC = .
DIRLIB = $(DIRAVR)/avr/lib

# Define programs and commands.
SHELL = sh

CC = avr-gcc

OBJCOPY = avr-objcopy
OBJDUMP = avr-objdump
SIZE = avr-size

REMOVE = rm -f
COPY = cp

HEXSIZE = $(SIZE) --target=$(FORMAT) $(TARGET).hex
ELFSIZE = $(SIZE) -A $(TARGET).elf

# Define Messages
# English
MSG_ERRORS_NONE = Errors: none
MSG_BEGIN = -------- begin --------
MSG_END = --------  end  --------
MSG_SIZE_BEFORE = Size before:
MSG_SIZE_AFTER = Size after:
MSG_COFF = Converting to AVR COFF:
MSG_EXTENDED_COFF = Converting to AVR Extended COFF:
MSG_FLASH = Creating load file for Flash:
MSG_EEPROM = Creating load file for EEPROM:
MSG_EXTENDED_LISTING = Creating Extended Listing:
MSG_SYMBOL_TABLE = Creating Symbol Table:
MSG_LINKING = Linking:
```

```
MSG_COMPILING = Compiling:
MSG_ASSEMBLING = Assembling:
MSG_CLEANING = Cleaning project:

# Define all object files.
OBJ = $(SRC:.c=.o) $(ASRC:.S=.o)

# Define all listing files.
LST = $(ASRC:.S=.lst) $(SRC:.c=.lst)

# Combine all necessary flags and optional flags.
# Add target processor to flags.
ALL_CFLAGS = -mmcu=$(MCU) -I. $(CFLAGS)
ALL_ASFLAGS = -mmcu=$(MCU) -I. -x assembler-with-cpp $(ASFLAGS)

# Default target.
all: begin gccversion sizebefore $(TARGET).elf $(TARGET).hex $(TARGET).eep \
    $(TARGET).lss $(TARGET).sym sizeafter finished end

# Eye candy.
# AVR Studio 3.x does not check make's exit code but relies on
# the following magic strings to be generated by the compile job.
begin:
    @echo
    @echo $(MSG_BEGIN)

finished:
    @echo $(MSG_ERRORS_NONE)

end:
    @echo $(MSG_END)
    @echo


# Display size of file.
sizebefore:
    @if [ -f $(TARGET).elf ]; then echo; echo $(MSG_SIZE_BEFORE); $(ELFSIZE); echo; fi

sizeafter:
    @if [ -f $(TARGET).elf ]; then echo; echo $(MSG_SIZE_AFTER); $(ELFSIZE); echo; fi


# Display compiler version information.
gccversion :
    @$(CC) --version


# Convert ELF to COFF for use in debugging / simulating in
# AVR Studio or VMLAB.
COFFCONVERT=$(OBJCOPY) --debugging \
    --change-section-address .data-0x800000 \
    --change-section-address .bss-0x800000 \
    --change-section-address .noinit-0x800000 \
    --change-section-address .eeprom-0x810000


coff: $(TARGET).elf
    @echo
    @echo $(MSG_COFF) $(TARGET).cof
    $(COFFCONVERT) -O coff-avr $< $(TARGET).cof


extcoff: $(TARGET).elf
    @echo
    @echo $(MSG_EXTENDED_COFF) $(TARGET).cof
    $(COFFCONVERT) -O coff-ext-avr $< $(TARGET).cof

# Create final output files (.hex, .eep) from ELF output file.
%.hex: %.elf
    @echo
    @echo $(MSG_FLASH) $@
    $(OBJCOPY) -O $(FORMAT) -R .eeprom $< $@

%.eep: %.elf
    @echo
    @echo $(MSG_EEPROM) $@
```

```
    -$(OBJCOPY) -j .eeprom --set-section-flags=.eeprom="alloc,load" \
    --change-section-lma .eeprom=0 -O $(FORMAT) $< $@

# Create extended listing file from ELF output file.
%.lss: %.elf
    @echo
    @echo $(MSG_EXTENDED_LISTING) $@
    $(OBJDUMP) -h -S $< > $@

# Create a symbol table from ELF output file.
%.sym: %.elf
    @echo
    @echo $(MSG_SYMBOL_TABLE) $@
    avr-nm -n $< > $@

# Link: create ELF output file from object files.
.SECONDARY : $(TARGET).elf
.PRECIOUS : $(OBJ)
%.elf: $(OBJ)
    @echo
    @echo $(MSG_LINKING) $@
    $(CC) $(ALL_CFLAGS) $(OBJ) --output $@ $(LDFLAGS)


# Compile: create object files from C source files.
%.o : %.c
    @echo
    @echo $(MSG_COMPILING) $<
    $(CC) -c $(ALL_CFLAGS) $< -o $@

# Compile: create assembler files from C source files.
%.s : %.c
    $(CC) -S $(ALL_CFLAGS) $< -o $@


# Assemble: create object files from assembler source files.
%.o : %.S
    @echo
    @echo $(MSG_ASSEMBLING) $<
    $(CC) -c $(ALL_ASFLAGS) $< -o $@

# Target: clean project.
clean: begin clean_list finished end

clean_list :
    @echo
    @echo $(MSG_CLEANING)
    $(REMOVE) $(TARGET).hex
    $(REMOVE) $(TARGET).eep
    $(REMOVE) $(TARGET).obj
    $(REMOVE) $(TARGET).cof
    $(REMOVE) $(TARGET).elf
    $(REMOVE) $(TARGET).map
    $(REMOVE) $(TARGET).obj
    $(REMOVE) $(TARGET).a90
    $(REMOVE) $(TARGET).sym
    $(REMOVE) $(TARGET).lnk
    $(REMOVE) $(TARGET).lss
    $(REMOVE) $(OBJ)
    $(REMOVE) $(LST)
    $(REMOVE) $(SRC:.c=.s)
    $(REMOVE) $(SRC:.c=.d)

# Automatically generate C source code dependencies.
# (Code originally taken from the GNU make user manual and modified
# (See README.txt Credits).)
#
# Note that this will work with sh (bash) and sed that is shipped with WinAVR
# (see the SHELL variable defined above).
# This may not work with other shells or other seds.
#
%.d: %.c
    set -e; $(CC) -MM $(ALL_CFLAGS) $< \
    | sed 's,\(.*\)\.o[ :]*,\1.o \1.d : ,g' > $@; \
    [ -s $@ ] || rm -f $@
```

```
# Remove the '-' if you want to see the dependency files generated.
-include $(SRC:.c=.d)


# Listing of phony targets.
.PHONY : all begin finish end sizebefore sizeafter gccversion coff extcoff \
    clean clean_list program
```