

# WinAVR und Eclipse

Da es für mich immer wieder Thema ist, wie ich eine Toolchain - hier für AVR - zum Laufen bekomme, werd ich das hier nun einmal dokumentieren.

Eine gute Zusammenfassung für WinAVR selbst ist auf [mikrocontroller.net](http://mikrocontroller.net)

## Downloads

Ich verwende [WinAVR-201001](#) und [Eclipse CDT Kepler](#) auf Windows 7

## Umgebungsvariablen

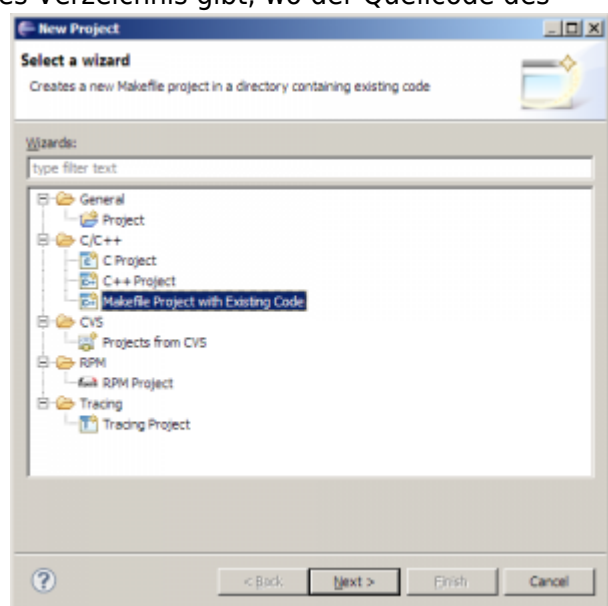
Damit avr-gcc erkannt wird, müssen im System-Pfad die Verzeichnisse für avr-gcc und für die bin-utils eingetragen werden:

```
PATH =  
[...];D:\Programme\WinAVR_20100110\bin;D:\Programme\WinAVR_20100110\utils\bin
```

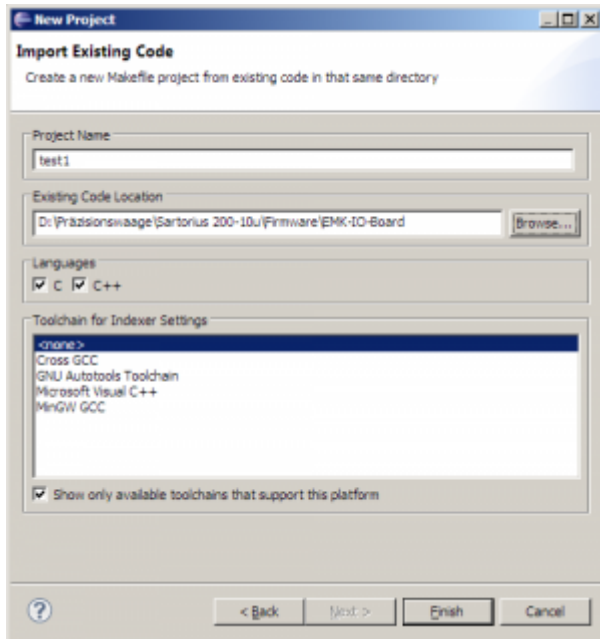
Nach dem Ändern ist ein erneutes Einloggen in Windows notwendig (Benutzer abmelden).

## Projekt in Eclipse erstellen

Ich gehe davon aus, dass es bereits ein existierendes Verzeichnis gibt, wo der Quellcode des



Projektes bereits vorhanden ist - inklusive Makefile.



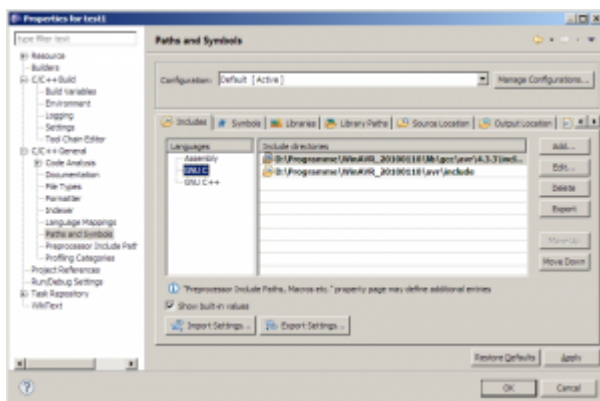
## Projekt-Einstellungen

Der C/CIndexer, der sehr hilfreich ist beim Browsen durch den Code (STRG+Klick auf Variable/Funktion/Define/...), funktioniert Anfangs noch nicht richtig: [\ { :winavr:indexer-errors.png?direct&300 } }](#)

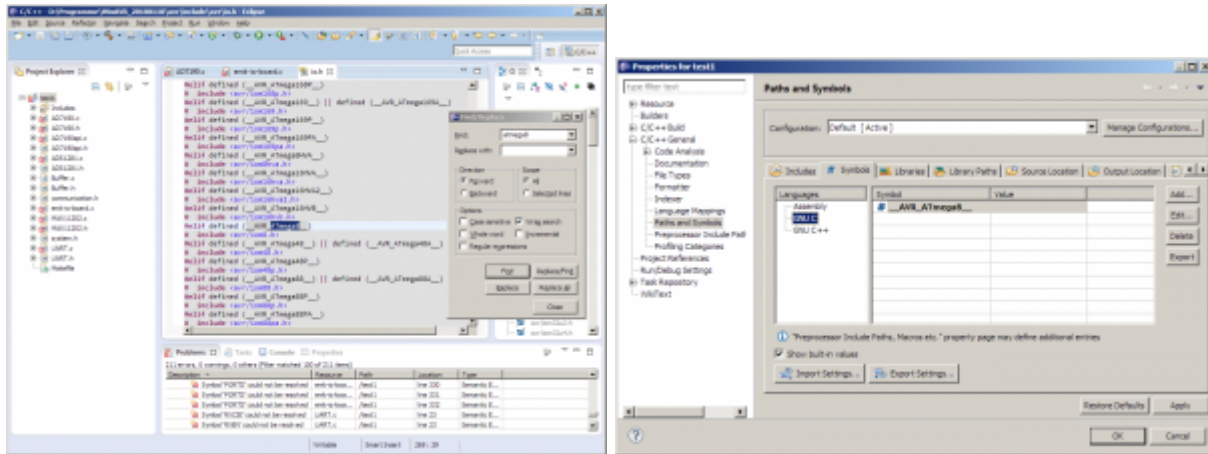
Damit der C/C Indexer in Eclipse richtig funktioniert, müssen die AVR-Includepaths im Projekt eingetragen werden: Mit Rechtsklick auf das Projekt und Eigenschaften geht man zum Unterpunkt „C/C++ General -> Paths and Symbols“.

Im Reiter Includes werden unter Language „GNU C“ folgende (File-System-)Pfade hinzugefügt:

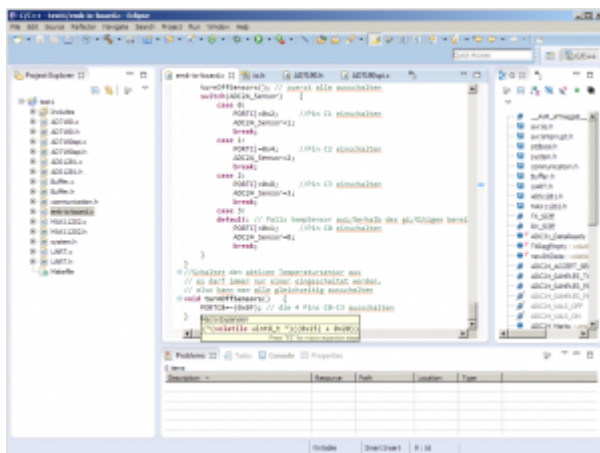
```
D:\Programme\WinAVR_20100110\lib\gcc\avr\4.3.3\include
D:\Programme\WinAVR_20100110\avr\include
```



Damit die Definitionen in <avr/io.h> richtig aufgelöst werden, braucht es noch die Definition des Mikrocontroller-Typs. Im Reiter „Symbols“ wird der entsprechende Mikrocontroller eingetragen. In meinem Fall ist es der ATMEGA8. Wie das Define genau heißt, findet man in der Header-Datei <avr/io.h>



Nun wird der Index erneut erstellt (wird beim Schließen des Properties-Fenster automatisch vorgeschlagen). Öffnet man nun die mit Fehler-Markern gekennzeichneten Dateien erneut, verschwinden die Fehlermaker.



Man kann den Index auch manuell erneut erstellen lassen, indem man auf das Projekt rechtsklickt und auf „Index -> Rebuild“ bzw. „Index -> Freshen All Files“ klickt.

## Makefile

Hier als Beispiel das verwendete Makefile

### Makefile

```
# Hey Emacs, this is a -*- makefile -*-
#
# WinAVR Sample makefile written by Eric B. Weddington, Jürg 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 program = Download the hex file to the device, using avrdude.
Please
#               customize the avrdude settings below first!
#
# 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
#FORMAT = binary

# Target file name (without extension).
TARGET = emk-io-board

# List C source files here. (C dependencies are automatically
generated.)
SRC = $(TARGET).c Buffer.c UART.c ADS1281.c MAX11202.c AD7190.c
AD7190spi.c
#readraw.c mmc_spi.c fat.c dos.c dir.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 =

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

# Debugging format.
# Native formats for AVR-GCC's -g are stabs [default], or dwarf-2.
# AVR (extended) COFF requires stabs, plus an avr-objcopy run.
DEBUG = stabs

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

# Compiler flag to set the C Standard level.
# c89 - "ANSI" C
# gnu89 - c89 plus GCC extensions
# c99 - ISO C99 standard (not yet fully implemented)
# gnu99 - c99 plus GCC extensions
CSTANDARD = -std=gnu99

# Place -D or -U options here
CDEFS =

# Place -I options here
CINCS =

# Compiler flags.
# -g*: generate debugging information
# -O*: optimization level
# -f...: tuning, see GCC manual and avr-libc documentation
# -Wall...: warning level
# -Wa,...: tell GCC to pass this to the assembler.
# -adhlns...: create assembler listing
CFLAGS = -g$(DEBUG)
CFLAGS += $(CDEFS) $(CINCS)
CFLAGS += -O$(OPT)
CFLAGS += -fsigned-char -funsigned-bitfields -fpack-struct -fshort-
enums
CFLAGS += -Wall -Wstrict-prototypes
#CFLAGS += -Wa,-adhlns=$(<:.c=.lst)
CFLAGS += $(patsubst %, -I%, $(EXTRINC_DIRS))
CFLAGS += $(CSTANDARD)

```

```
# 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

#Additional libraries.

# Minimalistic printf version
PRINTF_LIB_MIN = -Wl,-u,vfprintf -lprintf_min

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

PRINTF_LIB =

# Minimalistic scanf version
SCANF_LIB_MIN = -Wl,-u,vfscanf -lscanf_min

# Floating point + %[ scanf version (requires MATH_LIB = -lm below)
SCANF_LIB_FLOAT = -Wl,-u,vfscanf -lscanf_flt

SCANF_LIB =

MATH_LIB = -lm

# External memory options

# 64 KB of external RAM, starting after internal RAM (ATmega128!),
# used for variables (.data/.bss) and heap (malloc()).
#EXTMEMOPTS = -Wl,-Tdata=0x801100,--defsym=__heap_end=0x80ffff

# 64 KB of external RAM, starting after internal RAM (ATmega128!),
# only used for heap (malloc()).
#EXTMEMOPTS = -Wl,--defsym=__heap_start=0x801100,--
defsym=__heap_end=0x80ffff

EXTMEMOPTS =
```

```
# 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
LDLFLAGS += $(EXTMEMOPTS)
LDLFLAGS += $(PRINTF_LIB) $(SCANF_LIB) $(MATH_LIB)

# Programming support using avrdude. Settings and variables.

# Programming hardware: alf avr910 avrisp bascom bsd
# dt006 pavr picoweb pony-stk200 sp12 stk200 stk500
#
# Type: avrdude -c ?
# to get a full listing.
#
# AVRDUDE_PROGRAMMER = pony-stk200
AVRDUDE_PROGRAMMER = avrispmkII

# com1 = serial port. Use lpt1 to connect to parallel port.
AVRDUDE_PORT = usb

AVRDUDE_WRITE_FLASH = -U flash:w:$(TARGET).hex
#AVRDUDE_WRITE_EEPROM = -U eeprom:w:$(TARGET).eep

# Uncomment the following if you want avrdude's erase cycle counter.
# Note that this counter needs to be initialized first using -Yn,
# see avrdude manual.
#AVRDUDE_ERASE_COUNTER = -y

# Uncomment the following if you do /not/ wish a verification to be
# performed after programming the device.
#AVRDUDE_NO_VERIFY = -V

# Increase verbosity level. Please use this when submitting bug
# reports about avrdude. See
# <http://savannah.nongnu.org/projects/avrdude>
# to submit bug reports.
#AVRDUDE_VERBOSE = -v -v

AVRDUDE_FLAGS = -p $(MCU) -P $(AVRDUDE_PORT) -c $(AVRDUDE_PROGRAMMER)
AVRDUDE_FLAGS += $(AVRDUDE_NO_VERIFY)
AVRDUDE_FLAGS += $(AVRDUDE_VERBOSE)
AVRDUDE_FLAGS += $(AVRDUDE_ERASE_COUNTER)
```

```
# -----  
-----  
  
# Define directories, if needed.  
DIRAVR = D:/Programme/WinAVR_20100110  
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  
NM = avr-nm  
AVRDUDE = avrdude  
REMOVE = rm -f  
COPY = cp  
  
# 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)
```



```
# Compiler flags to generate dependency files.
GENDEPFLAGS = -Wp,-M,-MP,-MT,$(*F).o,-MF,.dep/$(@F).d

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

# Default target.
all: begin gccversion sizebefore build sizeafter finished end

build: elf hex

elf: $(TARGET).elf
hex: $(TARGET).hex
eep: $(TARGET).eep
lss: $(TARGET).lss
sym: $(TARGET).sym

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

# Program the device.
program: $(TARGET).hex $(TARGET).eep
    $(AVRDUDE) $(AVRDUDE_FLAGS) $(AVRDUDE_WRITE_FLASH)
    $(AVRDUDE_WRITE_EEPROM)

# 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) $@
    $(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)
$(REMOVE) .dep/*
```

*# Include the dependency files.*

```
-include $(shell mkdir .dep 2>/dev/null) $(wildcard .dep/*)
```

*# Listing of phony targets.*

```
.PHONY : all begin finish end sizebefore sizeafter gccversion \
build elf hex eep lss sym coff extcoff \
clean clean_list program
```

From:

<http://www.zeilhofer.co.at/wiki/> - Verschiedenste Artikel von Karl Zeilhofer

Permanent link:

[http://www.zeilhofer.co.at/wiki/doku.php?id=winavr\\_und\\_eclipse&rev=1393848444](http://www.zeilhofer.co.at/wiki/doku.php?id=winavr_und_eclipse&rev=1393848444)

Last update: **2014/03/03 13:07**

