How to compile XD-2031

December 21, 2012

Contents

1 About

This document describes how to compile the XD-2031 software. It should give you some hints if you haven't worked with GNU make / GCC / Eclipse before. If you're lucky enough to have this practice, you won't find much if any news here.

Instructions contained herein cover both compiling the firmware and the server software under Linux and OS X. Sorry, there isn't a Windows port at the time.

You can keep your favorite editor and compile at the command line with GNU make, but the usage of the integrated development environment *Eclipse* will be explained also.

Both, firmware and server, are written in plain C.

For any hints, how this software works, please have a look at the */docsrc* and */doc* directories and read *firmware overview.odp* (Libre Office document) and *wireformat.txt*.

You also won't find detailed explanations on how to contribute patches although they are welcome. Cloning the git repository and attempting a *pull request* would be the best way. Simply drop a mail in that case.

Good luck and may the Force be with you...

2 Prerequisites

The following software packages are required to compile XD-2031:

Package	Description	Version*
GNU make	build tool	3.81
GCC	GNU C compiler	4.2.1
cURL	cmdline tool for transferring files	7.26.0
AVR-GCC	GNU C compiler for AVR	4.5.1
AVR Libc	C library for AVR	1.8.0
AVRDUDE	AVR Downloader/UploaDEr	5.11.1

The following packages are optional:

Package	Description	Version*
git	Version control system	1.7.11.2
Doxygen	Documentation generator	1.8.1.1
Eclipse IDE for C/C++	IDE	Juno SR 1
LaTeX	Typesetting program	TeX Live 2012

*These version numbers reflect the versions used here at the time of writing. Other versions may also work, but that hasn't been tested.

2.1 Linux

Freedom causes variety, by consequence the names of the required packages and the commands to install these packages vary from distribution to distribution. The following instructions should be suitable for a Debian / Ubuntu style Linux. You are of course welcome to contribute commands for your preferred distribution.

Open a terminal and copy the following line into it:

```
sudo apt-get install build-essential gcc-avr avr-libc
avrdude avrdude-doc
```

Make sure to copy this into a single command line, then hit ENTER and type in your password to install the packages. This may take a while...

If you get a message, that you are not in the sudoers list, you may gain root rights with `su <your-root-password>` and copy the rest of the line after sudo.

The usage of git is strongly recommended, but not necessary if you want to compile only a specific release version of XD-2031. If git isn't installed at your system by default, you may add it with:

```
sudo apt-get install git git-gui
```

The installation of Eclipse for your specific system is out of the scope of this document. Please refer to the Eclipse website and have a look at the section "Installing Eclipse". Make sure to download "Eclipse for C/C++ Developers".

http://www.eclipse.org/downloads/

2.2 OS X

If you haven't already done before, start with installing XCode. This huge software environment including a larger number of applications not needed for compiling XD-2031 is virtually the only way to get a proper GCC for the Mac.

If you still are on Snow Leopard (or even before that), XCode is included in your installation disks. You can not install XCode using the App Store since that version requires Lion at least.

If you are on Lion, Mountain Lion or whatever cat to come next (are there still any cats left?), install XCode via the App Store:

`https://itunes.apple.com/us/app/xcode/id497799835?mt=12&uo=4`

Next, install the AVR toolchain "CrossPack for AVR". Grab and install it from:

`http://www.obdev.at/products/crosspack`

Starting with Mountain Lion, you may have to (at least temporarily) disable Gatekeeper to be able to install software from beyond the App Store. Instructions can be found at `https://answers.uchicago.edu/page.php?id=25481` or on YouTube at `http://youtu.be/H5DZXA4uxeQ`.

The usage of git is strongly recommended, but not necessary if you want to compile only a specific release version of XD-2031. Don't think further, go for it:

`http://git-scm.com/download`

If you'd like to use the Eclipse IDE, visit their website and make sure to download the "Eclipse IDE for C/C++ Developers". Install it as usual.

`http://www.eclipse.org/downloads`

3 Download Sources

XD-2031 is still under heavy construction, so you might want to get the latest developer sources with git. However, each commit may introduce new bugs, so you may consider the use of a release version or a snapshot taken a few days ago as well.

3.1 Releases / Snapshots

Release versions and git snapshots are available from the URL given below. Download and unzip your desired source in a directory of your choice.

http://xd2031.petsd.net

3.2 Latest developer sources

To clone the repository, run the following command. It will create the directory *XD2031* inside the current directory, download the latest source and place it there. This step is required only once. If you later decide to move this directory, you can do so. It is not necessary to delete and clone it again somewhere else.

```
git clone https://github.com/fachat/XD2031
```

This tutorial can not and doesn't want to be an introduction into git, and using git is not necessary for compiling and playing around if you do not intend sharing your results. On the other hand, a version control system is a great help and if you haven't worked with a VCS before or are new to git, it is definitely worth doing further reading. Start here: http://git-scm.com/doc

4 Command line tools

4.1 Firmware

4.1.1 Declare default device

Compilation is controlled by the build control tool 'GNU make'. If you haven't worked with Makefiles before, this section gives you some hints on usage.

In most cases you'll want to compile for your specific device only. You do so by declaring exactly one device as your default device inside the Makefile. Open the file *Makefile* with your favorite editor. You will find:

```
# Uncomment one of the following lines to select your device:
DEVICE=xs1541
#DEVICE=petSD
```

Comments start with a '#', so all device definitions starting with a '#' are in fact only comments and thus disabled. If you want to change the default device, delete the '#' before its line and don't forget to add another '#' at the line, that has been the default device until then. If you'd like to compile for a petSD, your Makefile should therefore look like this (notice how the '#' moved from petSD to xs1541):

```
# Uncomment one of the following lines to select your device:
#DEVICE=xs1541
DEVICE=petSD
```

If you edit the Makefile and save your changes, make sure your editor does not replace tabs with spaces or the Makefile won't work any longer.

4.1.2 Compile

Once you defined your default device, compilation is made by typing simply make without parameters. The binaries are placed in *XD2031/firmware/bin*.

```
cd XD2031/firmware
make
```

4.1.3 Compile for all devices

You can compile for all devices with a single command. GNU Make will run all necessary steps and compile each target one by one. The binaries are then placed in XD2031/firmware/bin.

```
cd XD2031/firmware
make zoo
```

4.1.4 Upload firmware

Edit *firmware/Makefile* suitable for your device and programmer:

```
# This defines the serial-over-USB port to use when loading the
# firmware into the device with "make load"
# For AVR (AtMega) needs avrdude - see avr/Makefile for details
SERIAL=/dev/ttyUSB0

# If you do ISP programming with "make flash", select your programmer/port:
DUDE_PROGRAMMER = avrispmkii
# DUDE_PROGRAMMER = stk200

DUDE_PORT = usb
# DUDE_PORT = /dev/parport0
# DUDE_PORT = lpt1

# If you didn't find your programmer, have a took at the list given from:
# avrdude -c ?
```

To upload the firmware on a XS-1541 via bootloader / USB, enter at your console without pressing ENTER:

```
make load
```

Reset the XS-1541 and hit ENTER within the next three seconds.

When using an ISP programmer (like an AVR ISP mkII), simply type:

```
make flash
```

4.2 Server

4.2.1 Compile

To compile the Server, change into the directory *XD2031/pcserver* and run the Makefile just the way you did to compile the firmware:

```
cd XD2031/pcserver
make
```

Your binary, let's say your program *fsser*, will be placed in the current directory *XD2031/pcserver*.

4.2.2 Install

If you'd like to run the Server with its default configuration (without parameters), installation is required to move some files to places where the Server expects them. To permanently install the Server, run:

```
sudo make install
```

To check respectively edit what goes where, have a look at the head of *Makefile* inside the pcserver directory. Watch out for:

```
PREFIX=/usr/local
BINDIR=bin
DOCDIR=xd2031/doc
SAMPLEDIR=xd2031/sample
TOOLSDIR=xd2031/tools
```

The full path for binaries, documents, samples and tools is generated by appending the specific directory path to PREFIX. As an example:

```
PREFIX=/usr/local
DOCDIR=xd2031/doc
→ /usr/local/xd2031/doc
```

Please be aware of the difference between the two document directories:

XD2031/doc should contain documents relevant while using XD-2031. These files are copied by the installer during installation.

The opposite is true for *XD2031/docsrc*: this directory should contain files that are of interest during development only but not for a regular user, so the installer ignores those files.

4.2.3 Uninstall

To uninstall the Server, run:

```
sudo make uninstall
```

This will remove not only the Server but also all its directories that may contain user-generated files, such as */usr/local/xd2031/samples*. So please be careful and check first, if you want to save some of your files first, before uninstalling.

On OS X, the auto-generated uninstall script from xd2031.dmg behaves slightly different: it will remove the Server and all the files it came with. It will *not* delete any files you created by yourself, e.g. by SAVEing to a sample directory.

4.2.4 Privileged Server Start (Linux)

On OS X, the device (XS-1541 / petSD) is always accessible with user rights. This is not the case for Linux.

Some Linux systems are very restrictive and doesn't allow a program running with user rights to communicate with an attached device, e.g. a XS-1541 showing up as /dev/ttyUSB0. Enabling this can be challenging, so by default the Server starts with root rights to ensure access to the device. Once opened, it drops his privileges and continues running with user rights to make sure not to harm anything.

That's the reason, why you will be asked for your password when compiling. Sudo is required to enable the Server to start with root rights.

If you want to configure the server to always use user rights only, make sure it can access the device. As a hint:

```
ls -l /dev/ttyUSB0
```

to get the group the device was assigned too. This could be *dialout*. Then add your user to this group. In most cases, you should be able to access the device now with user rights only.

To compile for "user rights only", edit *pcserver/Makefile* and change PRIVILEGED from 'y' to 'n'. Your resulting section of the Makefile should look like this:

```
# Open the serial device with root or user rights
PRIVILEGED=n
```

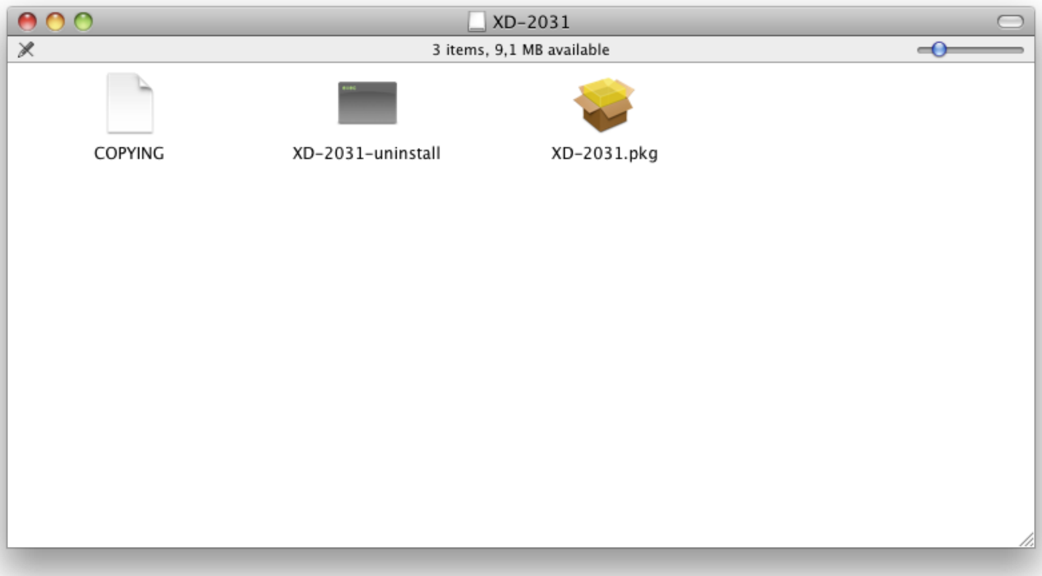
Another possibility without editing the Makefile is passing this case-sensitive parameter when calling make:

```
make PRIVILEGED=n
```

4.2.5 Generate dmg for binary distribution (OS X)

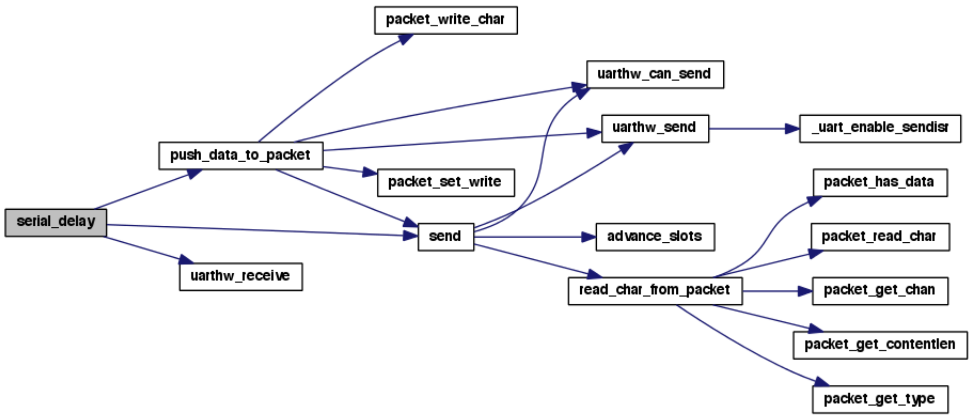
If you want to use your self-compiled Server on more than a single Mac, you can generate a nice xd2031.dmg file:

```
make dmg
```



4.2.6 Doxygen

Doxygen is a documentation generator. Its a great help to get an overview over all functions, data types etc. And last but not least, it can even draw some nice figures:



To generate the documentation for the firmware, change into the firmware directory and invoke make.

```
cd XD2031/firmware
make doc
```

Open `XD2031/docsrc/doxygen/firmware/html/index.html` with a browser (Firefox, Safari...) to view the generated documentation.

Generating the documentation for the Server is just the same.

```
cd XD2031/pcserver
make doc
```

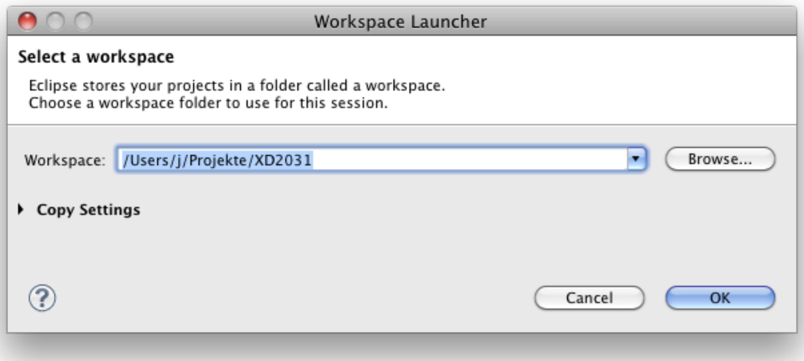
Open `XD2031/docsrc/doxygen/pcserver/html/index.html`.

5 Configure Eclipse

5.1 Workspace

Click *File* → *Switch workspace* → *Other...*

Select the topmost XD2031 directory as your workspace and click OK.


Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

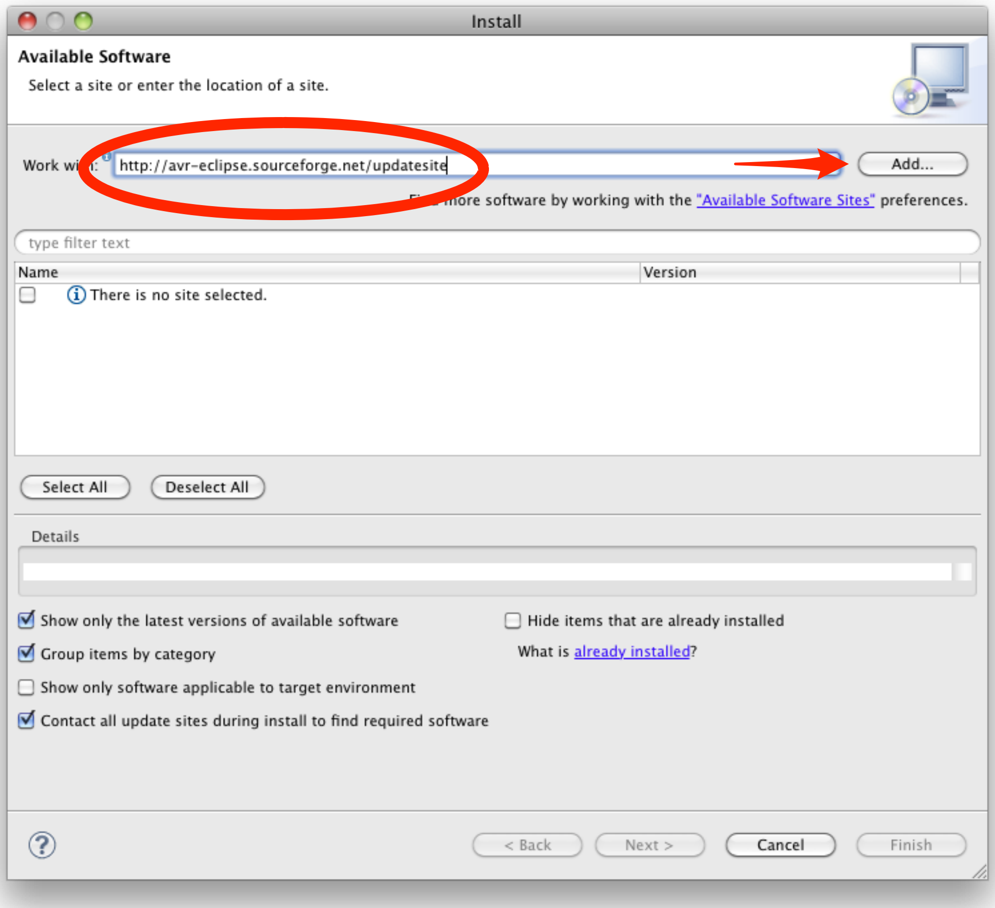
Workspace: /Users/j/Projekte/XD2031

5.2 Install AVR plugin

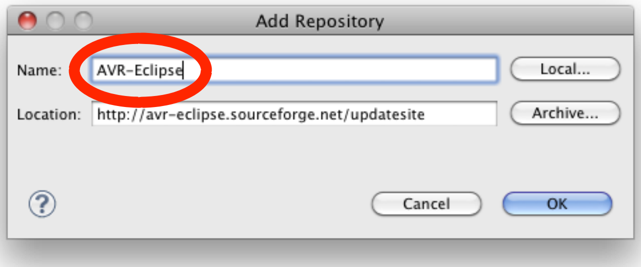
Click *Help* → *Install New Software...*

Fill the field *Work with* with **http://avr-eclipse.sourceforge.net/updatesite**

Click *Add...*.

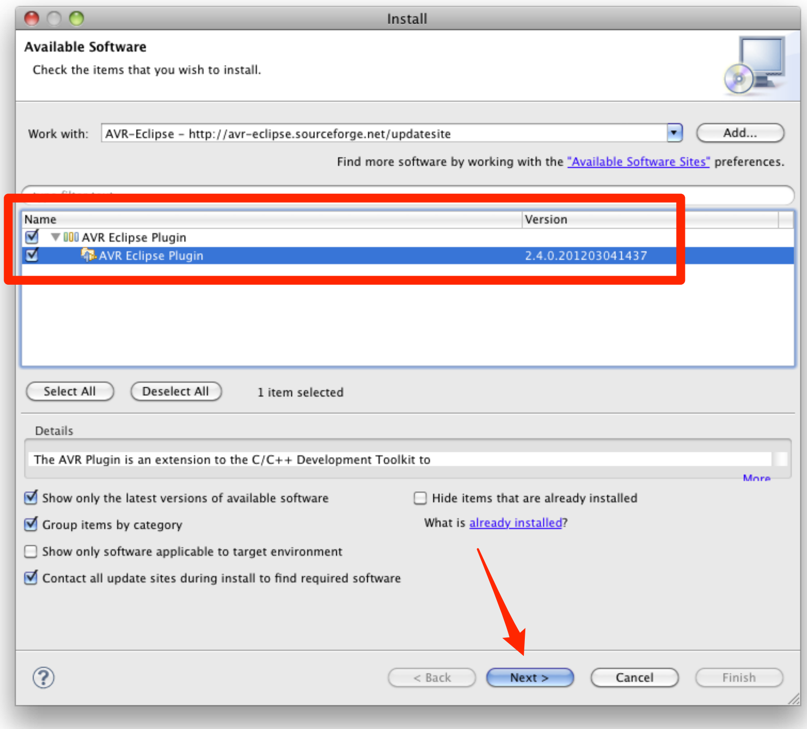

Install
Available Software
Select a site or enter the location of a site.
Work with:
http://avr-eclipse.sourceforge.net/updatesite
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
There is no site selected.
Select All
Deselect All
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Cancel
Finish

Give a name, e.g. **AVR-Eclipse**, then click OK.



Name: AVR-Eclipse
Location: http://avr-eclipse.sourceforge.net/updatesite

Select the newest version. Make sure to checkmark it. Then click *Next* a couple of times to install the plugin.



Available Software

Check the items that you wish to install.

Work with: AVR-Eclipse - http://avr-eclipse.sourceforge.net/updatesite

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☑ AVR Eclipse Plugin	
☑ AVR Eclipse Plugin	2.4.0.201203041437

1 item selected

Details

The AVR Plugin is an extension to the C/C++ Development Toolkit to

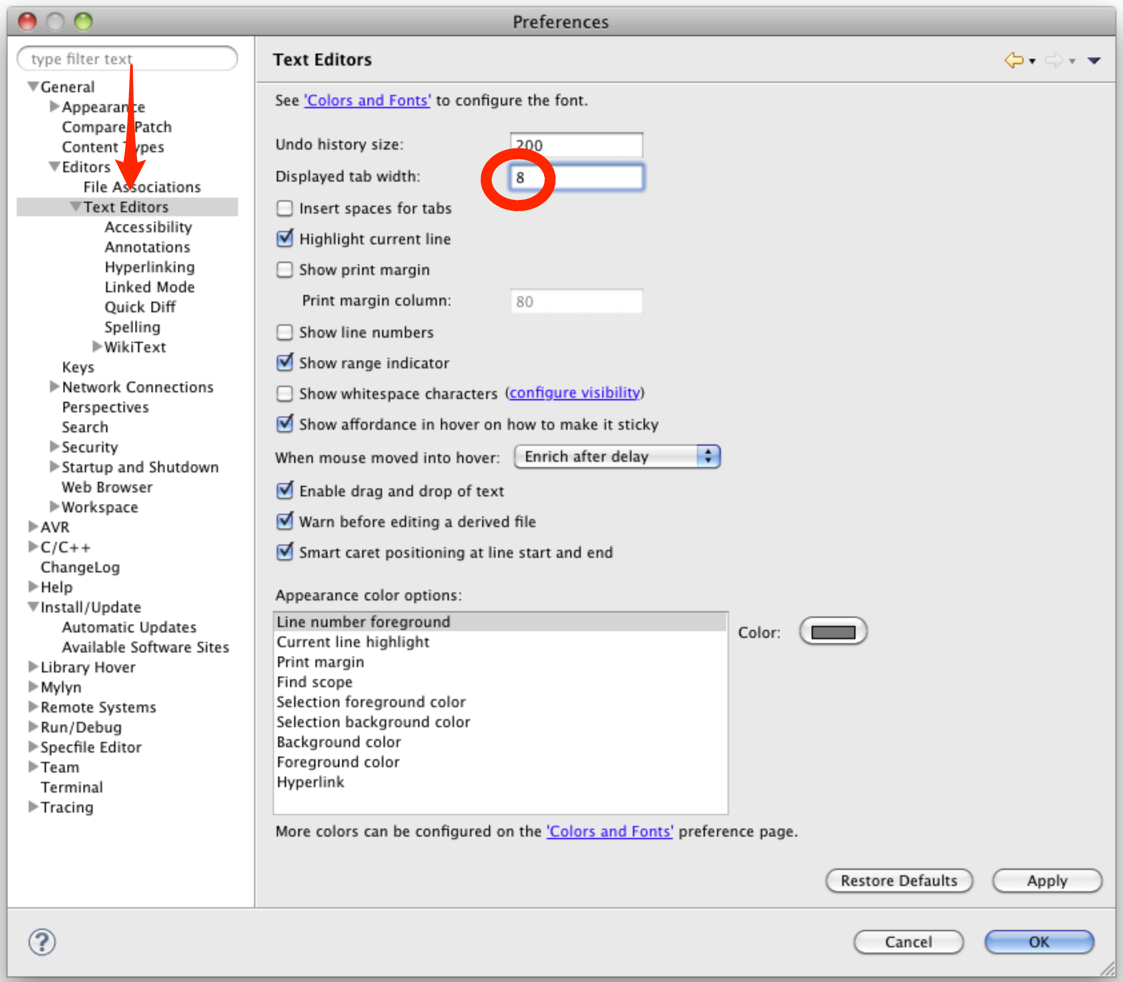
☑ Show only the latest versions of available software
☐ Hide items that are already installed
☑ Group items by category
What is already installed?
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software

5.3 Editor preferences

If you're on OS X, click at *Eclipse* → *Preferences...* On Linux or Windows, this should be under *File* → *Preferences...*

Expand *General* → *Editors* → *Text Editors* at the left side of the window.

Set tab size to **8**, then click *Apply*.



Text Editors

See 'Colors and Fonts' to configure the font.

Undo history size: 200
Displayed tab width: 8

- [ ] Insert spaces for tabs
- [✓] Highlight current line
- [ ] Show print margin
  - Print margin column: 80
- [ ] Show line numbers
- [✓] Show range indicator
- [ ] Show whitespace characters (configure visibility)
- [✓] Show affordance in hover on how to make it sticky

When mouse moved into hover: Enrich after delay

- [✓] Enable drag and drop of text
- [✓] Warn before editing a derived file
- [✓] Smart caret positioning at line start and end

Appearance color options:

Line number foreground
Current line highlight
Print margin
Find scope
Selection foreground color
Selection background color
Background color
Foreground color
Hyperlink

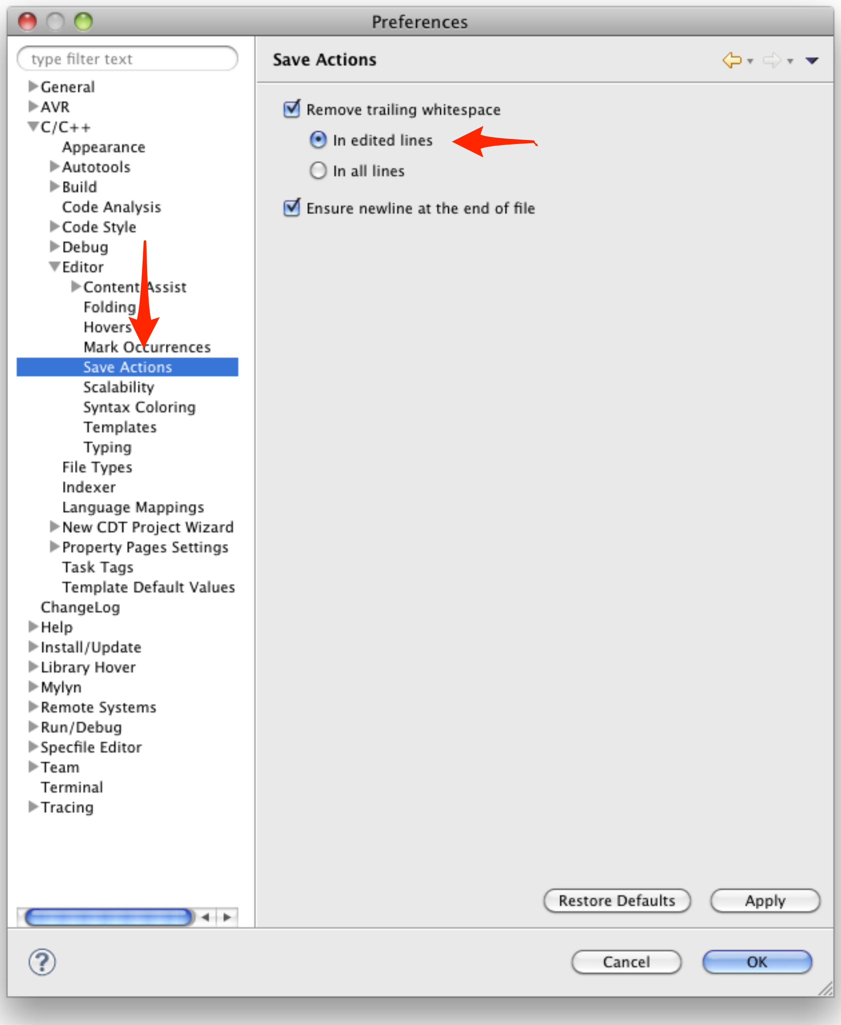
Color: ▬

More colors can be configured on the 'Colors and Fonts' preference page.

Expand *C/C++* → *Editor* → *Save Actions* at the left side of the window.

Make sure to have removing whitespaces *In edited lines* selected, or completely disable this function. Do not select *In all lines* or you'll end up with a lot of whitespace only commits when contributing with git.

Click *Apply*. You may check other options now and make them suitable for your needs, then click OK.



Save Actions

[✓] Remove trailing whitespace

[● In edited lines]

○ In all lines

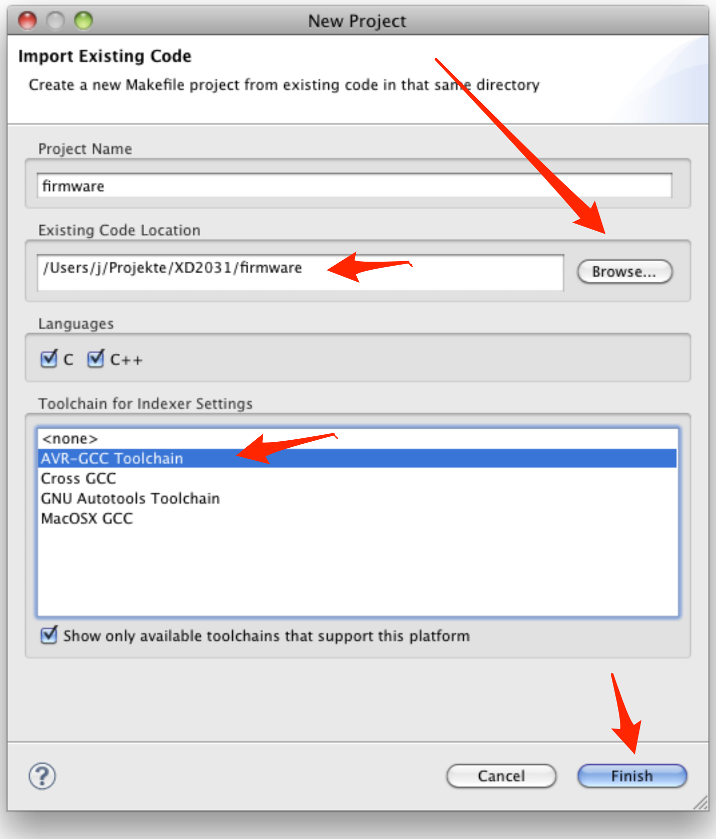
[✓] Ensure newline at the end of file

5.4 Firmware Project

Click *File* → *New* → *Makefile Project with Existing Code...*

Click *Browse...* and select the firmware directory of your previous downloaded XD-2031 source tree. The *Project Name* will automatically get filled with *firmware*, what is just fine.

Select the *AVR-GCC Toolchain*, then click *Finish*.


New Project

Import Existing Code

Create a new Makefile project from existing code in that same directory

Project Name: firmware

Existing Code Location: /Users/j/Projekte/XD2031/firmware

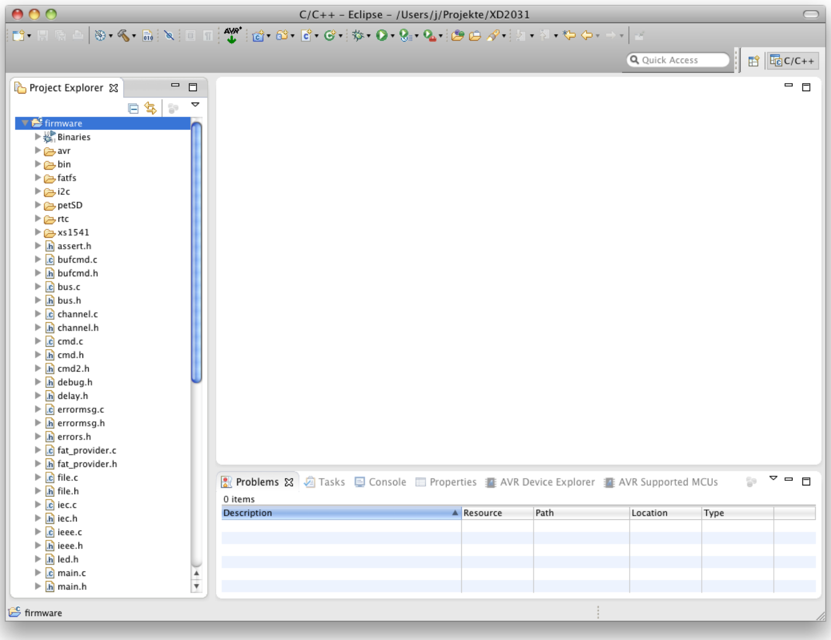
Languages: ☑ C ☑ C++

Toolchain for Indexer Settings:
- <none>
- AVR-GCC Toolchain
- Cross GCC
- GNU Autotools Toolchain
- MacOSX GCC

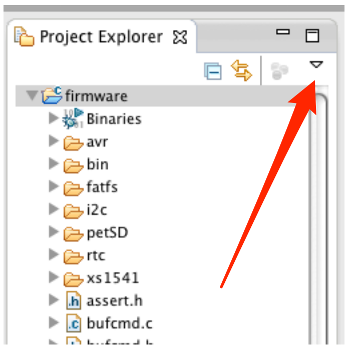
☑ Show only available toolchains that support this platform

In order to be able to upload the firmware with a single mouse-click, we have to switch our new project into an AVR project. There's still some magic required to do this...

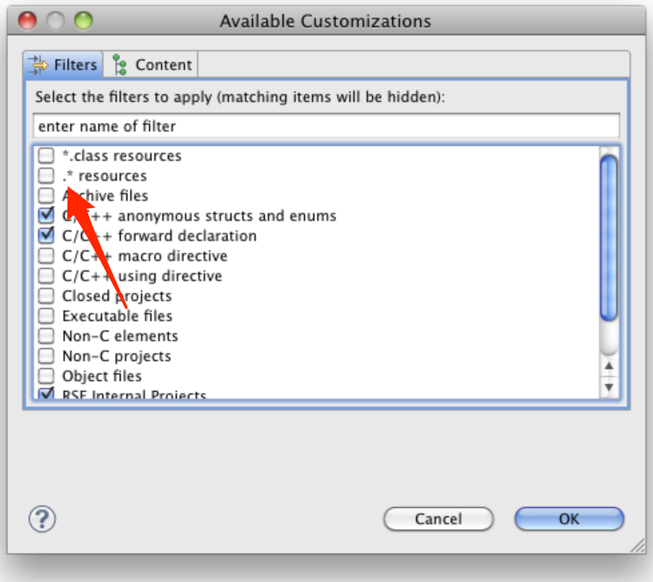
Click *Window* → *Show View* → *Project Explorer*. Double click at *firmware* to expand the files list. You should see a number of source files and directories now:



Click at the triangle icon to expand the view menu. Click *Customize View...*



Remove the checkmark at *.* resources* to show hidden files. Click OK.



Select the filters to apply (matching items will be hidden):

enter name of filter

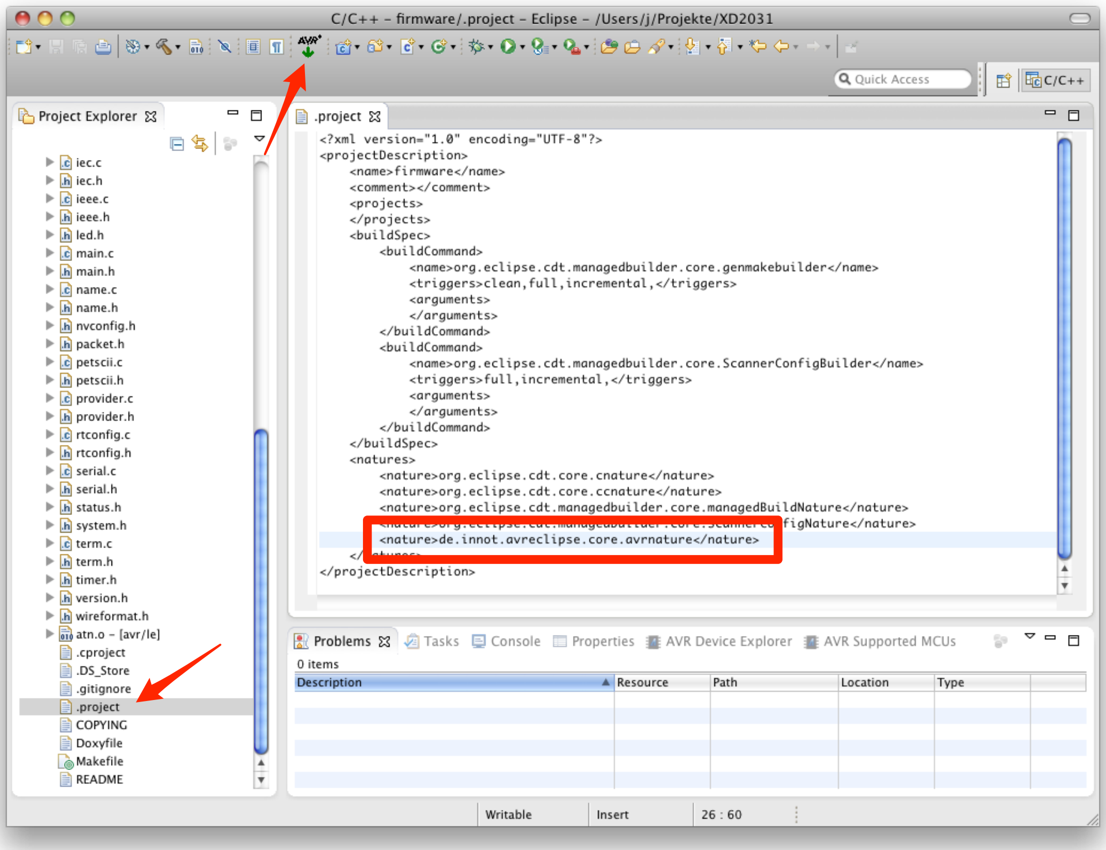
- [ ] *.class resources
- [ ] .* resources
- [ ] Archive files
- [x] C/C++ anonymous structs and enums
- [x] C/C++ forward declaration
- [ ] C/C++ macro directive
- [ ] C/C++ using directive
- [ ] Closed projects
- [ ] Executable files
- [ ] Non-C elements
- [ ] Non-C projects
- [ ] Object files
- [x] RSE Internal Projects

At the end of the file list, some hidden dot-files are visible now. You may have to scroll down to see them. Double-click at *.project* to open the file.

Add the following line to the <natures>-Block:

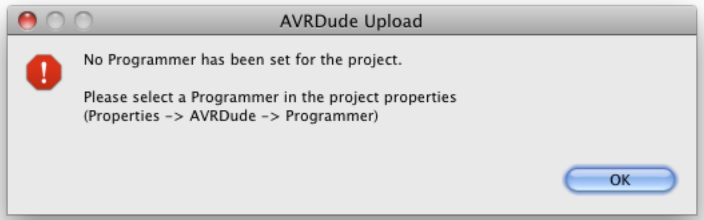
```
<nature>de.innot.avreclipse.core.avrnature</nature>
```

Save the file by clicking *File* → *Save*.



```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
    <name>firmware</name>
    <comment></comment>
    <projects>
    </projects>
    <buildSpec>
        <buildCommand>
            <name>org.eclipse.cdt.managedbuilder.core.genmakebuilder</name>
            <triggers>clean,full,incremental,</triggers>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>
            <name>org.eclipse.cdt.managedbuilder.core.ScannerConfigBuilder</name>
            <triggers>full,incremental,</triggers>
            <arguments>
            </arguments>
        </buildCommand>
    </buildSpec>
    <natures>
        <nature>org.eclipse.cdt.core.cnature</nature>
        <nature>org.eclipse.cdt.core.ccnature</nature>
        <nature>org.eclipse.cdt.managedbuilder.core.managedBuildNature</nature>
        <nature>org.eclipse.cdt.managedbuilder.core.ScannerConfigNature</nature>
        <nature>de.innot.avreclipse.core.avrnature</nature>
    </natures>
</projectDescription>
```

Click at the AVR Upload Icon. You should get a message *No programmer has been set for the project* if you successfully switched your project into an AVR project. Click *File* → *Restart* then to restart Eclipse. You should get the same view again.



AVRDude Upload

No Programmer has been set for the project.

Please select a Programmer in the project properties
(Properties -> AVRDude -> Programmer)

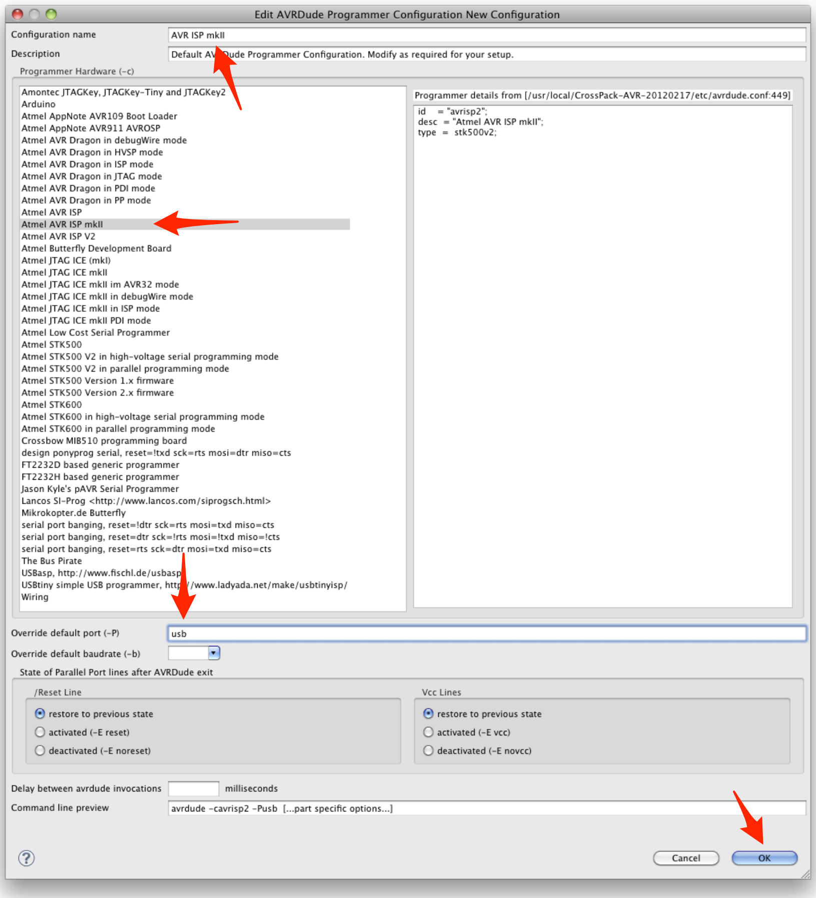
5.5 AVRDUDE

Click *Project* → *Properties* at the menu, and expand *AVR* → *AVRDude* at the left side of the *Properties for firmware* window.

Click *New...* in the section *Programmer configuration*.

The following configuration fits for a AVR ISP mkII. If you're still in need of a programmer, don't waste time and money with toys, go and buy a tool like this.

Name the configuration, e.g. **AVR ISP mkII**. Select your appropriate programmer, **Atmel AVR ISP mkII** here. Enter **usb** at the field *Override default port (-P)*.



Edit AVRDude Programmer Configuration New Configuration

Configuration name: AVR ISP mkII

Description: Default AVRDude Programmer Configuration. Modify as required for your setup.

Programmer Hardware (-c)

- Amontec JTAGKey, JTAGKey-Tiny and JTAGKey2
- Arduino
- Atmel AppNote AVR109 Boot Loader
- Atmel AppNote AVR911 AVROSP
- Atmel AVR Dragon in debugWire mode
- Atmel AVR Dragon in HVSP mode
- Atmel AVR Dragon in ISP mode
- Atmel AVR Dragon in JTAG mode
- Atmel AVR Dragon in PDI mode
- Atmel AVR Dragon in PP mode
- Atmel AVR ISP
- [✓ Atmel AVR ISP mkII]
- Atmel AVR ISP V2
- Atmel Butterfly Development Board
- Atmel JTAG ICE (mkI)
- Atmel JTAG ICE mkII
- Atmel JTAG ICE mkII im AVR32 mode
- Atmel JTAG ICE mkII in debugWire mode
- Atmel JTAG ICE mkII in ISP mode
- Atmel JTAG ICE mkII PDI mode
- Atmel Low Cost Serial Programmer
- Atmel STK500
- Atmel STK500 V2 in high-voltage serial programming mode
- Atmel STK500 V2 in parallel programming mode
- Atmel STK500 Version 1.x firmware
- Atmel STK500 Version 2.x firmware
- Atmel STK600
- Atmel STK600 in high-voltage serial programming mode
- Atmel STK600 in parallel programming mode
- Crossbow MIB510 programming board
- design ponyprog serial, reset=!txd sck=rts mosi=dtr miso=cts
- FT2232D based generic programmer
- FT2232H based generic programmer
- Jason Kyle's pAVR Serial Programmer
- Lancos SI-Prog <http://www.lancos.com/siprogsch.html>
- Mikrokopter.de Butterfly
- serial port banging, reset=!dtr sck=rts mosi=txd miso=cts
- serial port banging, reset=dtr sck=!rts mosi=!txd miso=!cts
- serial port banging, reset=rts sck=dtr mosi=txd miso=cts
- The Bus Pirate
- USBasp, http://www.fischl.de/usbasp/
- USBtiny simple USB programmer, http://www.ladyada.net/make/usbtinyisp/
- Wiring

Programmer details from [/usr/local/CrossPack-AVR-20120217/etc/avrdude.conf:449]

```
id    = "avrisp2";
desc  = "Atmel AVR ISP mkII";
type  =  stk500v2;
```

Override default port (-P): usb

Override default baudrate (-b):

State of Parallel Port lines after AVRDude exit

/Reset Line: [✓ restore to previous state] activated (-E reset) deactivated (-E noreset)

Vcc Lines: [✓ restore to previous state] activated (-E vcc) deactivated (-E novcc)

Delay between avrdude invocations: milliseconds

Command line preview: avrdude -cavrisp2 -Pusb [...part specific options...]

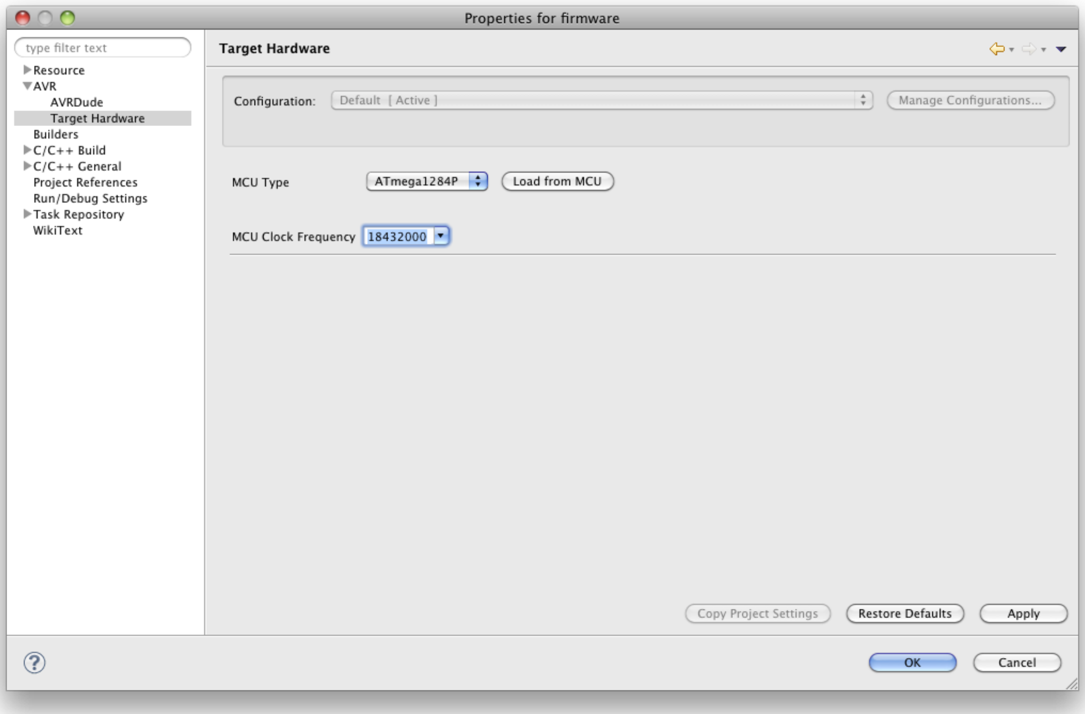
Click OK to get back to the window *Properties for firmware*.

Click *AVR* → *Target Hardware* at the left side of the *Properties for firmware* window.

Select the MCU Clock Frequency appropriate for your device:

Device	MCU Clock Frequency
XS-1541	14745600
petSD	18432000

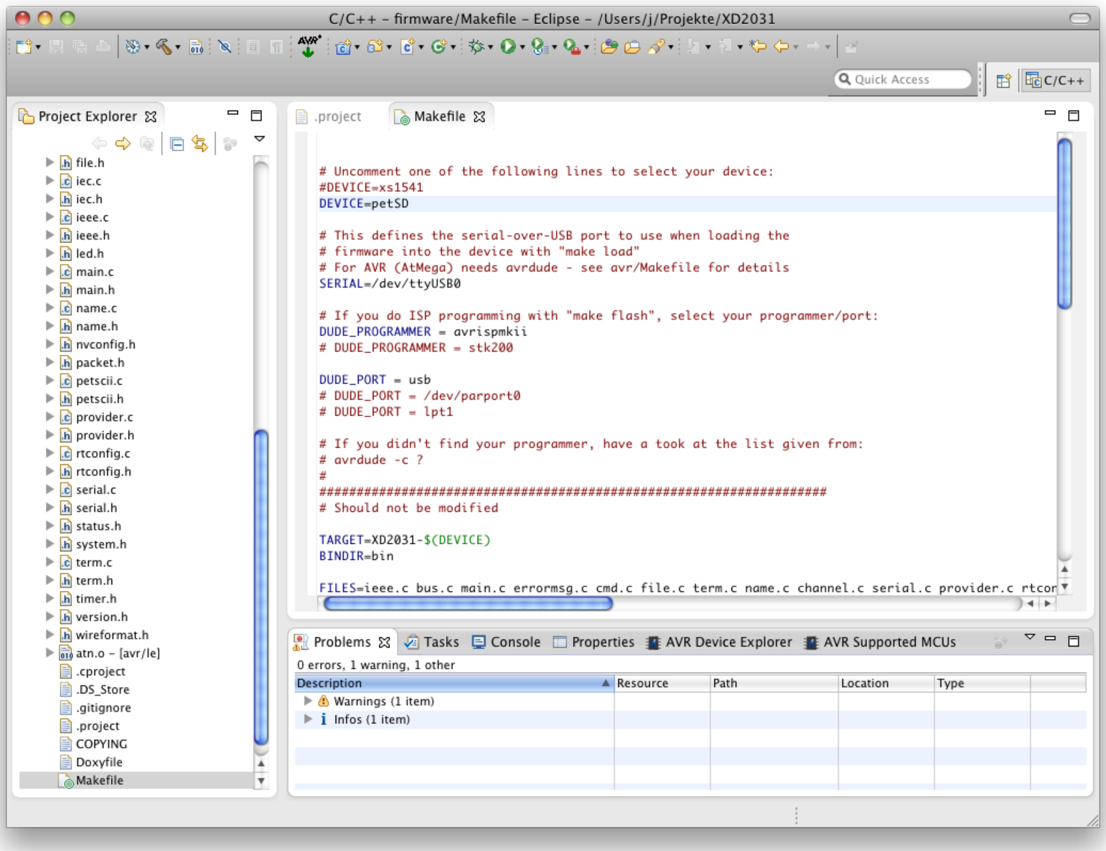
At *MCU Type* above, click *Load from MCU* to auto detect it. This is also a test to verify the communication with the programmer. You should get one of those ATmega644-types for an XS-1541 and ATmega1284P for a petSD.


Configuration: Default [Active]
MCU Type: ATmega1284P
MCU Clock Frequency: 18432000

Click OK to close the properties window.

At the *Project Explorer*, double click at *Makefile* to open the file. Uncomment one of the DEVICE definitions to select your device. Make sure to comment a previous uncommented device, since only a single device is allowed at once.

Click *File* → *Save* to save the edited Makefile.



```
# Uncomment one of the following lines to select your device:
#DEVICE=xs1541
DEVICE=petSD

# This defines the serial-over-USB port to use when loading the
# firmware into the device with "make load"
# For AVR (AtMega) needs avrdude - see avr/Makefile for details
SERIAL=/dev/ttyUSB0

# If you do ISP programming with "make flash", select your programmer/port:
DUDE_PROGRAMMER = avrispmkii
# DUDE_PROGRAMMER = stk200

DUDE_PORT = usb
# DUDE_PORT = /dev/parport0
# DUDE_PORT = lpt1

# If you didn't find your programmer, have a took at the list given from:
# avrdude -c ?
#
######################################################################
# Should not be modified

TARGET=XD2031-$(DEVICE)
BINDIR=bin

FILES=ieee.c bus.c main.c errormsg.c cmd.c file.c term.c name.c channel.c serial.c provider.c rtcor
```

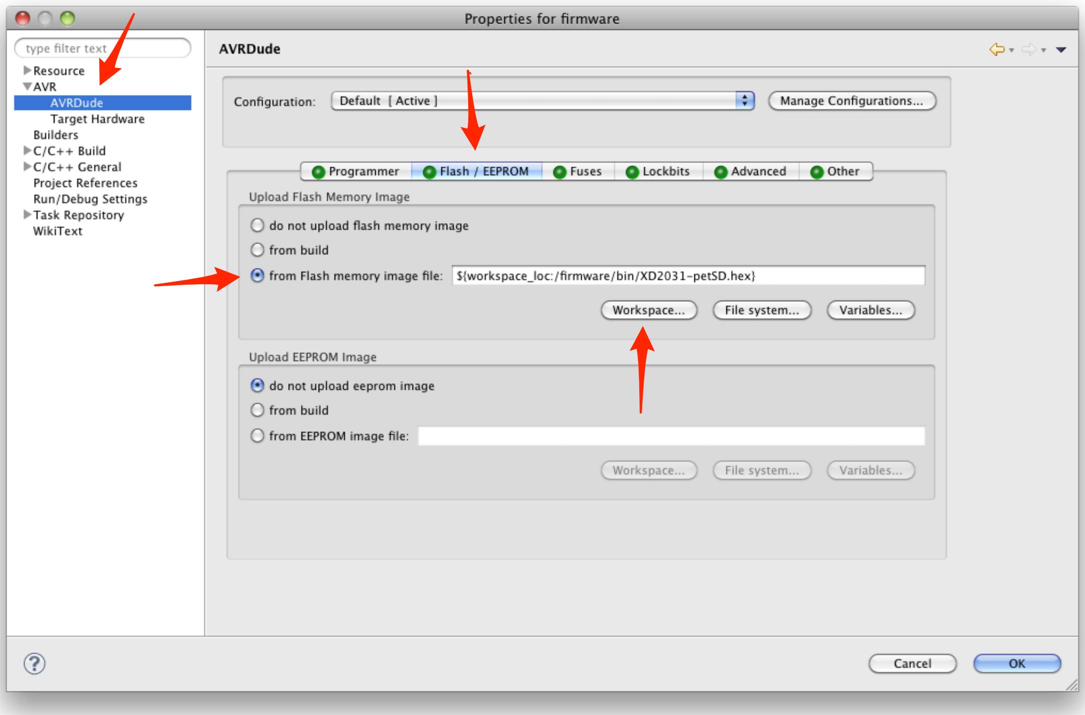
0 errors, 1 warning, 1 other

Description	Resource	Path	Location	Type
Warnings (1 item)				
Infos (1 item)				

Now its time to compile for the first time: click *Project* → *Build All*. The compiler output is shown in the Console window. Now that we have binaries in the directory *bin*, we can configure the AVRDude Plugin to flash those.

Click *Project* → *Properties* and expand *AVR* → *AVRDude* at the left side of the *Properties for firmware* window.

Open the section *Flash / EEPROM*. At the section *Upload Flash Memory Image*, select the radio-box *from Flash memory image file*. Click at *Workspace...*. A file selection window pops up then. Expand the directory firmware to see all files, go to the bin directory and choose the hex-file appropriate for your device.



Configuration: Default [Active]

Upload Flash Memory Image

- ( ) do not upload flash memory image
- ( ) from build
- (●) from Flash memory image file: ${workspace_loc:/firmware/bin/XD2031-petSD.hex}

Upload EEPROM Image

- (●) do not upload eeprom image
- ( ) from build
- ( ) from EEPROM image file:

Click OK to close the properties window.

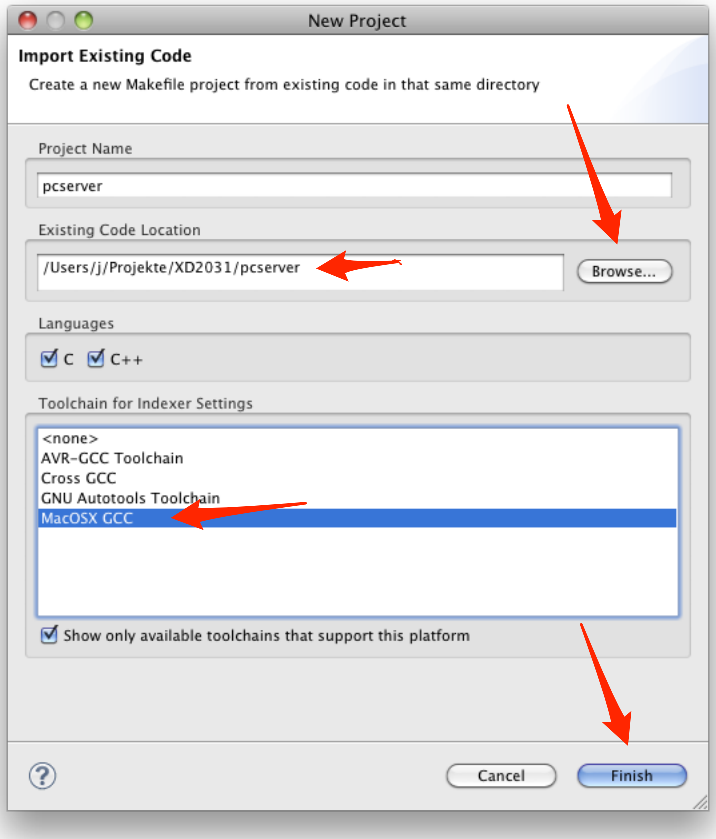
Congrats, you should be able to upload the firmware now simply by clicking the AVR Upload Icon. If you prefer using the menu bar, click *AVR* → *Upload Project to Target Device*.

5.6 Server Project

Configuring the Server Project is much like the firmware project. Click *File* → *New* → *Makefile Project with Existing Code...*

Click *Browse...* and select the directory *pcserver* of your previous downloaded XD-2031 source tree. The *Project Name* will automatically get filled with *pcserver*, what is just fine.

Select the toolchain appropriate for compiling for your computer, *MacOSX GCC* here, then click *Finish*.


New Project

Import Existing Code

Create a new Makefile project from existing code in that same directory

Project Name: pcserver

Existing Code Location: /Users/j/Projekte/XD2031/pcserver

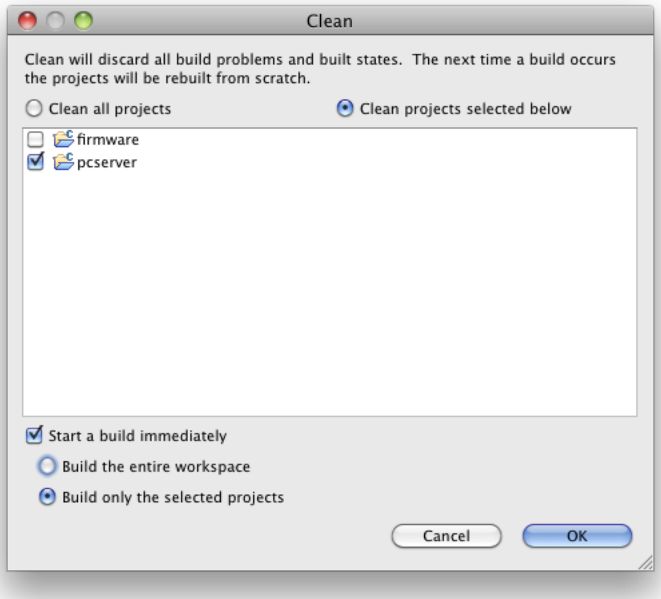
Languages: ☑ C ☑ C++

Toolchain for Indexer Settings:
- <none>
- AVR-GCC Toolchain
- Cross GCC
- GNU Autotools Toolchain
- MacOSX GCC

☑ Show only available toolchains that support this platform

Now click *Project* → *Clean...*

Select *Clean projects below*. Checkmark only *pcserver*. Checkmark *Start a build immediately* and select *Build only the selected projects*. Then click OK to compile the Server.


Clean will discard all build problems and built states. The next time a build occurs the projects will be rebuilt from scratch.

( ) Clean all projects [✓ Clean projects selected below]

☐ firmware
✓ pcserver

✓ Start a build immediately

( ) Build the entire workspace
[✓ Build only the selected projects]

Have a look at the Console Log and check, if the compilation was successful. If you read *fsser opens the serial device...* everything went fine and you can skip the rest of this chapter.

If you configured the Makefile to compile the Server starting with root rights, you may find:

```
sudo chown root fsser
sudo: no tty present and no askpass program specified
```

You should have read the chapter Privileged Server Start (Linux) at page 10 first to understand why your password is required at compile time.

There are two workarounds:

First you may edit *pcserver/Makefile* and replace sudo with gksudo (GNOME, Xfce, LXDE) or kdesudo (KDE). On the downside, you may have to enter your password twice at each compilation.

Another possibility is allowing the execution of `chmod` and `chown` with root rights without the need of entering the password.

If you disable the sudo password for your account, you will seriously compromise the security of your computer. Anyone sitting at your unattended, logged in account will have complete Root access, and remote exploits become much easier for malicious crackers.

You have been warned...

Start with getting the absolute paths of chown and chmod by entering the commands:

```
whereis chown
whereis chmod
```

This gives `/usr/sbin/chown` and `/bin/chmod` here.

Add the following lines with `sudo visudo` at the end of your /etc/sudoers file:

```
<your-user-name> ALL = NOPASSWD: /usr/sbin/chown
<your-user-name> ALL = NOPASSWD: /bin/chmod
```

Replace <your-user-name> with (you guessed it) your user name and omit the <>. And don't forget to give the paths to chown and chmod suitable for your system, that's what you asked whereis for.