

# **F<sup>2</sup>MC-16LX/16FX FAMILY**

## **16-BIT MICROCONTROLLER**

### **ALL SERIES**

---

# **HOW TO USE ECLIPSE WITH FUJITSU MCUS**

APPLICATION NOTE



## Revision History

Date	Issue
2009-11-10	V1.0, CII, First Version

This document contains 23 pages.

## Warranty and Disclaimer

The use of **the deliverables** (e.g. software, application examples, target boards, evaluation boards, starter kits, schematics, engineering samples of IC's etc.) is subject to the conditions of Fujitsu Microelectronics Europe GmbH ("FME") as set out in (i) the terms of the License Agreement and/or the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials.

Please note that the deliverables are intended for and must only be used in an evaluation laboratory environment.

The software deliverables are provided without charge and therefore provided on an as-is basis. The software deliverables are to be used exclusively in connection with FME products.

Regarding hardware deliverables, FME warrants that they will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

Should a hardware deliverable turn out to be defect, FME's entire liability and the customer's exclusive remedy shall be, at FME's sole discretion, either return of the purchase price and the license fee, or replacement of the hardware deliverable or parts thereof, if the deliverable is returned to FME in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to FME, or abuse or misapplication attributable to the customer or any other third party not relating to FME or to unauthorised decompiling and/or reverse engineering and/or disassembling.

FME does not warrant that the deliverables does not infringe any third party intellectual property right (IPR). In the event that the deliverables infringe a third party IPR it is the sole responsibility of the customer to obtain necessary licenses to continue the usage of the deliverable.

In the event the software deliverables include the use of open source components, the provisions of the governing open source license agreement shall apply with respect to such software deliverables.

To the maximum extent permitted by applicable law FME disclaims all other warranties, whether express or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the deliverables are not designated.

To the maximum extent permitted by applicable law, FME's liability is restricted to intention and gross negligence. FME is not liable for consequential damages.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

The contents of this document are subject to change without a prior notice, thus contact FME about the latest one.

## Contents

<b>REVISION HISTORY .....</b>	<b>2</b>
<b>WARRANTY AND DISCLAIMER .....</b>	<b>3</b>
<b>CONTENTS .....</b>	<b>4</b>
<b>1 INTRODUCTION .....</b>	<b>6</b>
1.1 Overview .....	6
1.2 Differences between Eclipse and Softune .....	7
<b>2 INSTALLATION .....</b>	<b>8</b>
2.1 Java Runtime Environment (JRE) .....	8
2.2 Eclipse .....	8
2.3 Fujitsu Plug-in for Eclipse .....	8
2.3.1 Online installation .....	9
2.3.2 Manual installation .....	10
2.4 Softune.....	10
2.5 MSYS.....	10
2.6 FLASHly.....	11
2.7 Set up the environment variable .....	11
<b>3 FIRST STEPS IN ECLIPSE.....</b>	<b>12</b>
3.1 Create a project.....	12
3.2 Configure flash programming .....	13
3.3 Add the Fujitsu controller files.....	14
3.4 Create a new main.c source file .....	15
3.5 Compile, link and if enabled flash the application .....	16
<b>4 ECLIPSE FEATURES.....</b>	<b>17</b>
4.1 Special terms used by Eclipse.....	17
4.2 Views .....	18
4.2.1 Project Explorer .....	18
4.2.2 Problems .....	18
4.2.3 Outline .....	19
4.2.4 Console .....	19
4.3 Panes.....	19
4.4 Code editor.....	19
4.4.1 Code folding and line numbers .....	19
4.4.2 Mark occurrences .....	20

---

4.4.3	Code completion.....	20
4.4.4	Code formatting .....	20
4.4.5	Refactoring .....	20
4.5	Menu bar / Toolbar .....	22
4.6	Project settings.....	22
<b>5</b>	<b>APPENDIX.....</b>	<b>23</b>
5.1	Links to Fujitsu .....	23
5.2	Links to third party sites.....	23

# 1 Introduction

Eclipse is a well known development environment developed by a large open source community including companies like IBM. Since its architecture is based on plug-ins it allows a wide variety of features to be added as in this case support for the Softune compiler tool chain for Fujitsu microcontrollers.

This application note describes how to set up Eclipse so you can use it to develop your Fujitsu microcontroller applications. It will also highlight the important differences between Softune Workbench and Eclipse and give a short introduction in Eclipse in general.

This application note is referring to version 1.1.3 of the Fujitsu Eclipse plug-in with Eclipse Galileo (v3.5).

Please note that most software presented in this application note is not developed or owned by Fujitsu. Therefore Fujitsu can not take any responsibilities or provide support for any problems you encounter by using it.

## 1.1 Overview

In order to be able to use Eclipse to build Fujitsu MCU targeted applications the following software components are required:

- Eclipse for C/C++ (i.e. Eclipse with the C Development Tools plug-in)
- Java Runtime Environment 6  
Required to run Eclipse
- Fujitsu plug-in for Eclipse  
Provides project dialogs specific for Fujitsu microcontrollers and the interface to the Softune tool chain
- Softune compiler tool chain (Softune C Compiler, Softune Assembler Pack)  
Primary component to build applications for Fujitsu microcontrollers
- MSYS  
Optional; allows to use the external builder to build the projects as it provides the commands used in the process of compiling applications, i.e. make (handles the build process itself), echo (prints messages on the console) and rm (deletes files); alternatively the internal builder can be used which sometimes requires to recompile the whole project and not only the changed and/or new files
- FLASHly  
Optional; allows flashing of the target MCU directly controlled by Eclipse

Eclipse does provide a lot of other plug-ins which can improve workflows, e.g. integration with Subversion and CVS or modelling tools.

## 1.2 Differences between Eclipse and Softune

Feature	Softune	Eclipse
Tool chain help file integration	Yes	No
Simulator	Yes	No
Debugger	Yes	No <sup>1</sup>
Dedicated Error/Warning list	No	No <sup>1,2</sup>
Project management	+	++
Integrated flash programming	Yes <sup>3</sup>	Yes <sup>3</sup>
Integration with code documentation (Doxygen)	No	Yes
Syntax highlighting	Yes (basic)	Yes (advanced)
Code completion	No	Yes
Code formatter	No	Yes
Refactoring	No	Yes
Outline view	No	Yes
Code folding	No	Yes

<sup>1</sup>: Planned for future revisions

<sup>2</sup>: Not fully functional

<sup>3</sup>: External tool can be controlled by the development environment

## 2 Installation

---

Installation of the different software components required to work with Eclipse.

---

In order to use Eclipse properly some additional software packages are required. This chapter will guide you through the installation steps.

The following software besides Softune is neither developed nor owned by Fujitsu. Thus Fujitsu can not guarantee that the given addresses will remain correct. If you encounter an invalid address please report it to Fujitsu ([microcontroller\\_info@fme.fujitsu.com](mailto:microcontroller_info@fme.fujitsu.com)).

The installation steps for a full set up including all optional components are:

1. Java Runtime Environment
2. Eclipse
3. Eclipse Plug-in
4. Softune
5. MSYS
6. FLASHly
7. Set up the PATH environment variable

### 2.1 Java Runtime Environment (JRE)

Install the current JRE from [java.sun.com/javase/downloads/index.jsp](http://java.sun.com/javase/downloads/index.jsp).

### 2.2 Eclipse

Open [www.eclipse.org/downloads/](http://www.eclipse.org/downloads/), download the “Eclipse IDE for C/C++ Developers” and unpack it.

### 2.3 Fujitsu Plug-in for Eclipse

There are two ways to get the plug-in in your local copy of Eclipse:

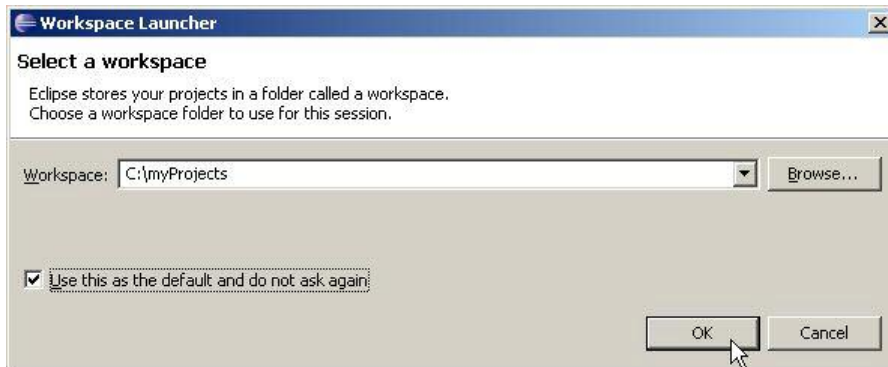
- Online installation through Eclipse’s software updater
- Manual installation

Depending on your environment you can go either way. Online installation has the advantage that the plug-in can automatically be updated when there is a new version whereas manual installation does not require an internet connection on the target machine when installing.

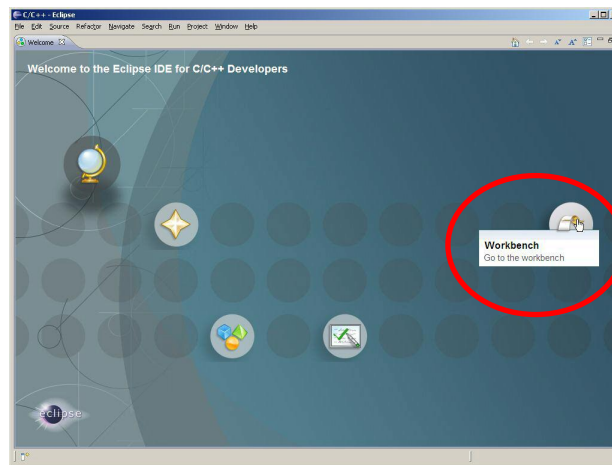
The whole Eclipse folder including the installed plug-in can also be copied to other machines without any problems so you can integrate the plug-in in Eclipse once and then copy Eclipse to other machines.

### 2.3.1 Online installation

Run Eclipse. When asked to select a workspace select the path where the projects should be stored later on.

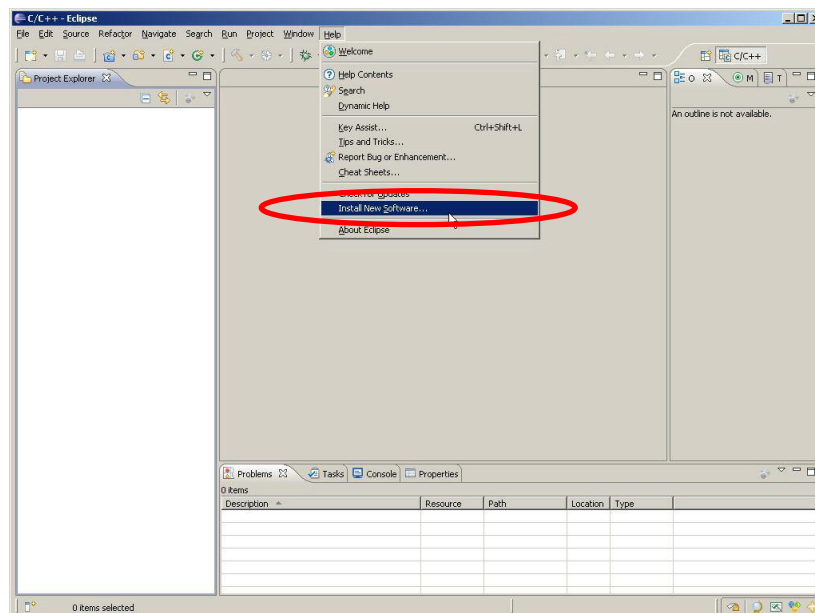


Go to the Eclipse Workbench:

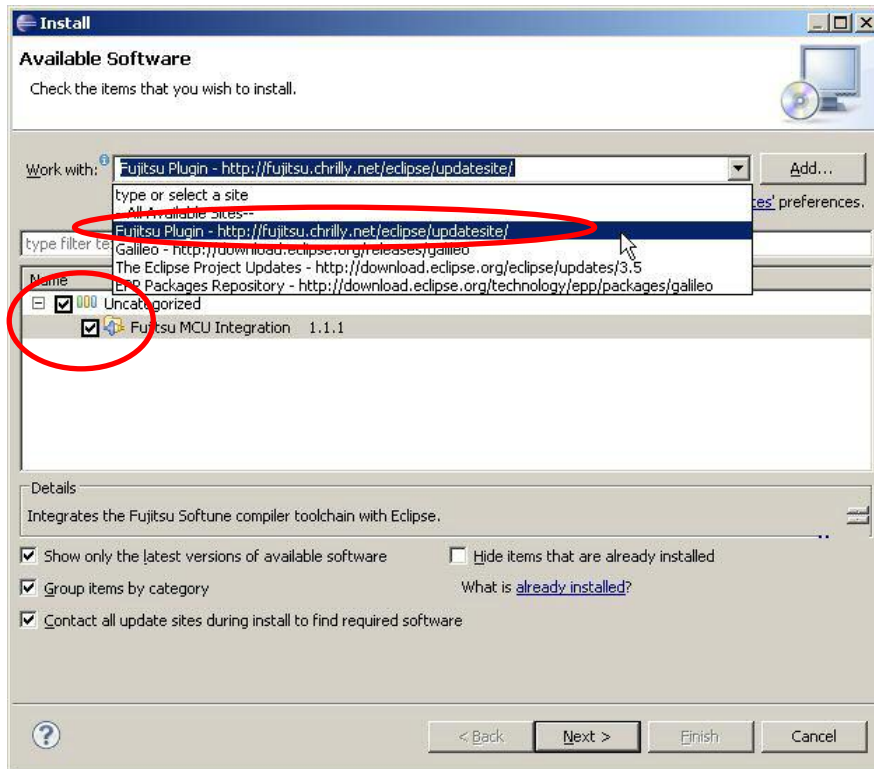


Note: If your workstation is connected to the internet through a proxy you have to set it up first (menu entry *Window – Preferences*, tree location *General – Network Connections*. *Active Provider* has to be set to *Manual* in this case).

Select *Help – Install New Software...* from the menu:



Click on *Add...* and enter a name for the new update site (e.g. “Fujitsu Plug-in”) and the location of the update site “<http://fujitsu.chrilly.net/eclipse/updatesite/>”. Close the dialog by clicking OK. Select the newly entered update site from the dropdown box:



Expand the item “Uncategorized” and check the box in front of the “Fujitsu MCU Integration” then click next and follow the dialogs. Ignore the warning about the plug-in being unsigned content and when asked to restart Eclipse confirm it.

### 2.3.2 Manual installation

Open the plug-in package listing website (<http://fujitsu.chrilly.net/eclipse/updatesite/plugins/>) in a browser and download the latest version of the plug-in. Copy the plug-in archive into the *plugins* subfolder of your Eclipse installation.

## 2.4 Softune

In the package selection menu of the Softune installer only “SOFTUNE C Compiler” and “SOFTUNE Assembler Pack” have to be selected for use with Eclipse. For information regarding the installation of Softune refer to its website ([http://mcu.emea.fujitsu.com/mcu\\_tool/detail/SWB\\_\(F2MC-16\)\\_V3.htm](http://mcu.emea.fujitsu.com/mcu_tool/detail/SWB_(F2MC-16)_V3.htm)).

Note: Softune is not provided via Internet but only available on CD/DVD available from your distributor.

## 2.5 MSYS

Go to the download page of MinGW ([www.mingw.org](http://www.mingw.org)). Open the “MSYS Base System” – “Current Release” (as of writing this document the current release is 1.0.11). Download *msysCORE-\*-bin.tar.gz* and unpack it (e.g. to C:\MSYS).

Alternatively download the stripped down MSYS which includes only the required files from [fujitsu.chrilly.net/eclipse](http://fujitsu.chrilly.net/eclipse).

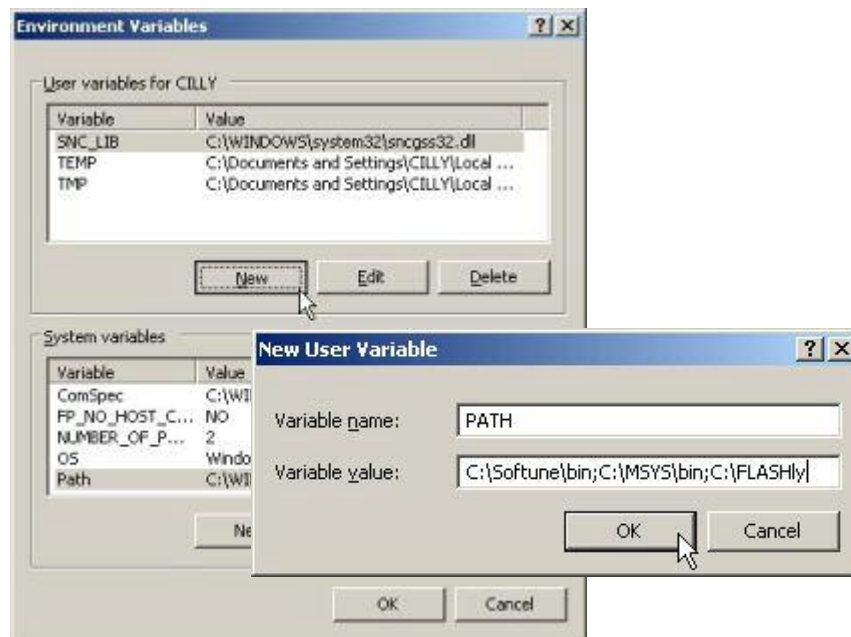
## 2.6 FLASHly

Get FLASHly from [www.holgerium.de/elektronik/index.htm](http://www.holgerium.de/elektronik/index.htm) and unpack it (e.g. to C:\Program Files\FLASHly).

## 2.7 Set up the environment variable

In order for Eclipse to be able to find the binaries from Softune, MSYS and FLASHly each of them have to be added to the environment variable *PATH* in Windows.

1. Open the *System* dialog (either through the Control Panel or right click on the *My Computer* icon – *Properties*).
2. Select the *Advanced* tab.
3. Click on *Environment Variables*.
4. Check whether there already is a “PATH” entry in the upper list. If so select it and click on *Edit* otherwise click on *New*.
5. Enter “PATH” as *Variable name*. As *Variable value* enter the path names to the binaries of Softune (C:\Softune\bin), MSYS (C:\MSYS\bin) and FLASHly (C:\Program Files\FLASHly), each entry separated by a semicolon. If there was a value before just add the new paths to the end of the old values.
6. Close all dialogs with *OK*.



## 3 First steps in Eclipse

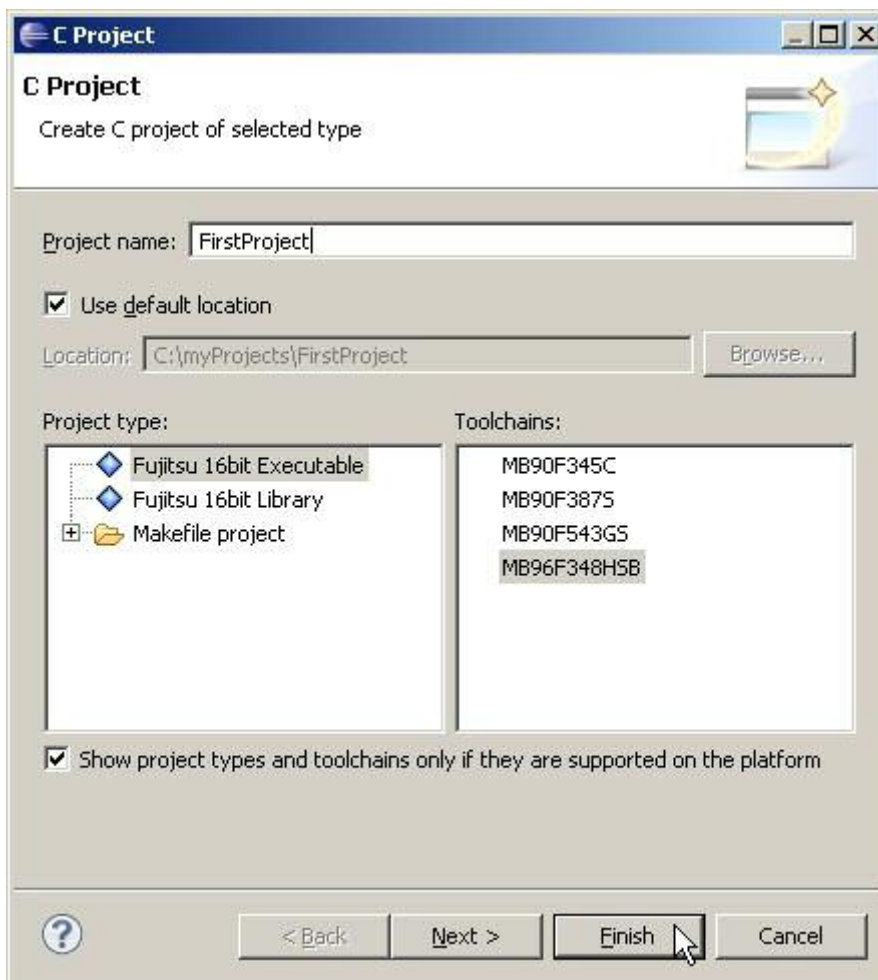
Basic steps to the first project.

In the following a standard development process is described:

- Create a project
- Configure flash programming
- Add Fujitsu controller files
- Create source files
- Compile, link and program the flash

### 3.1 Create a project

Do a right click in the *Project Explorer* on the left of the main Eclipse window and select *New – C Project*. Enter a name for the project and select “Fujitsu 16bit Executable” as *Project type* and the appropriate microcontroller name as *Toolchain*.

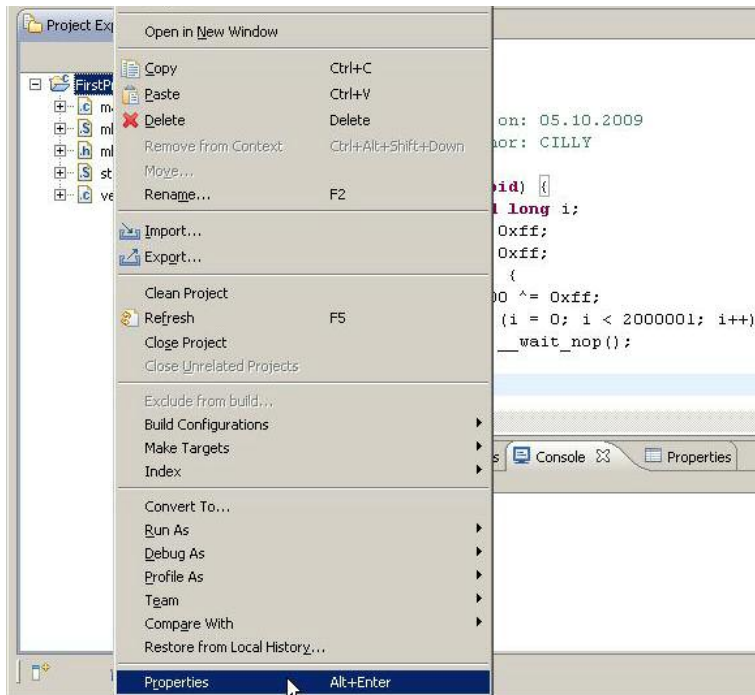


Note: If your microcontroller is not available from the list refer to the documentation of the plug-in on its web site on how to add new devices or contact the developer of the plug-in.

Click *Next >* to add a second compiler configuration besides *Release* (e.g. *Debug*) or just *Finish* to create the project with a *Release* configuration only.

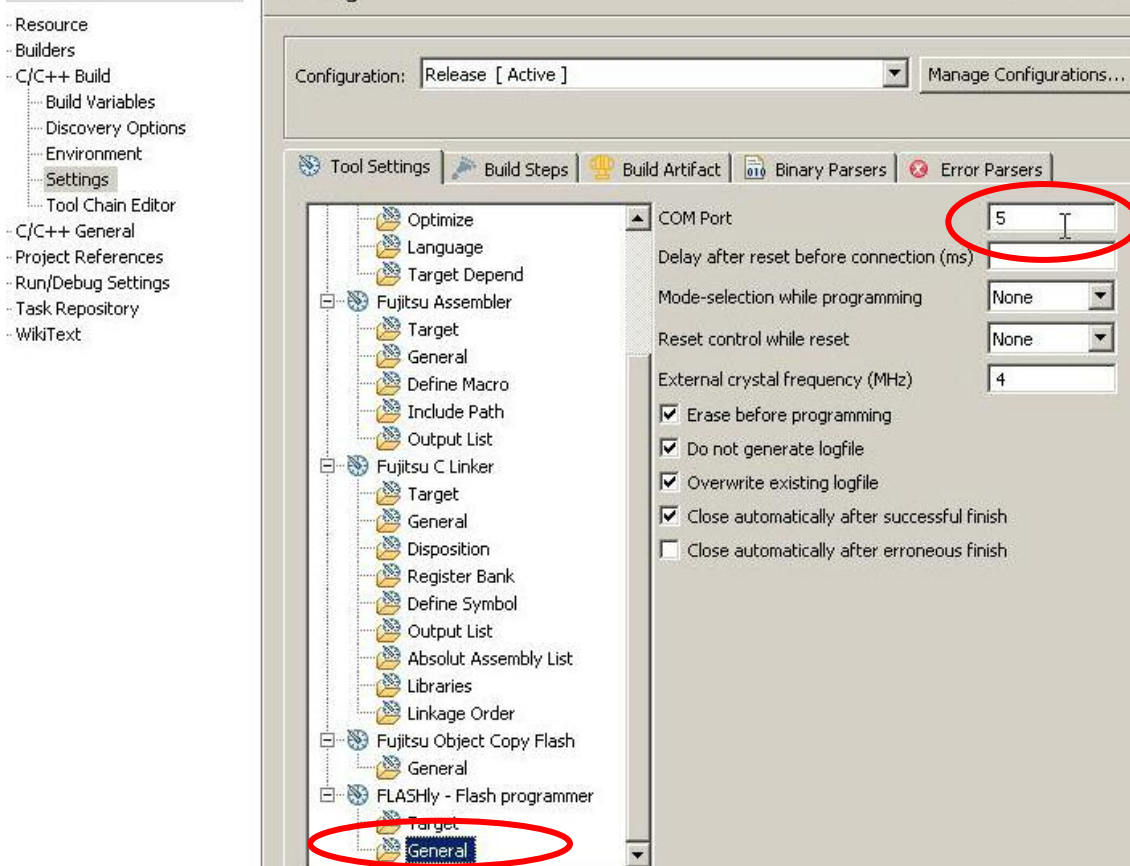
### 3.2 Configure flash programming

Open the project properties through the menu (*Project – Properties*) or through the context menu of the project in the *Project Explorer – Properties*:

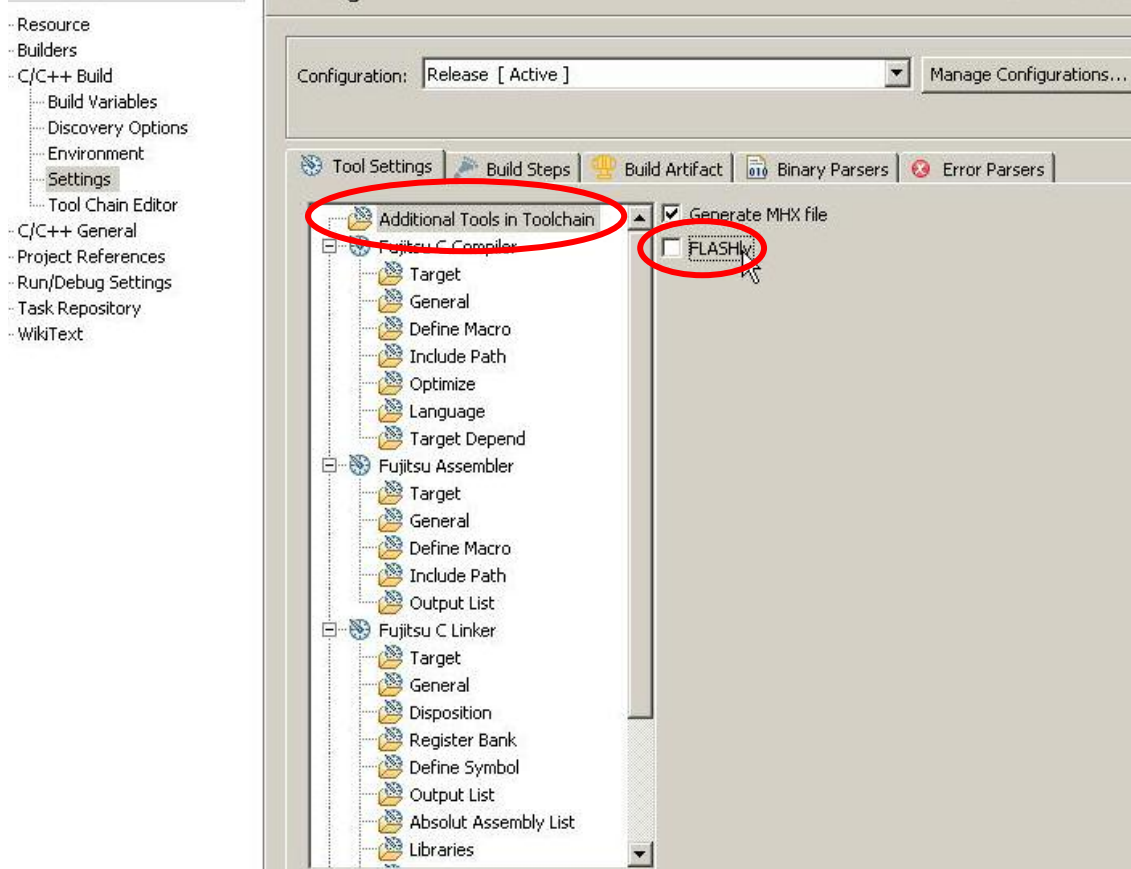


On the left side go to *C/C++ Build – Settings*.

- If you have a microcontroller which is supported by FLASHly navigate to *FLASHly – General* and enter the COM port the board is connected to in the appropriate field (also update the other fields if needed by your hardware, e.g. the crystal frequency):



- If FLASHly does not support your microcontroller uncheck the box *FLASHly* on the page *Additional Tools in Toolchain*:

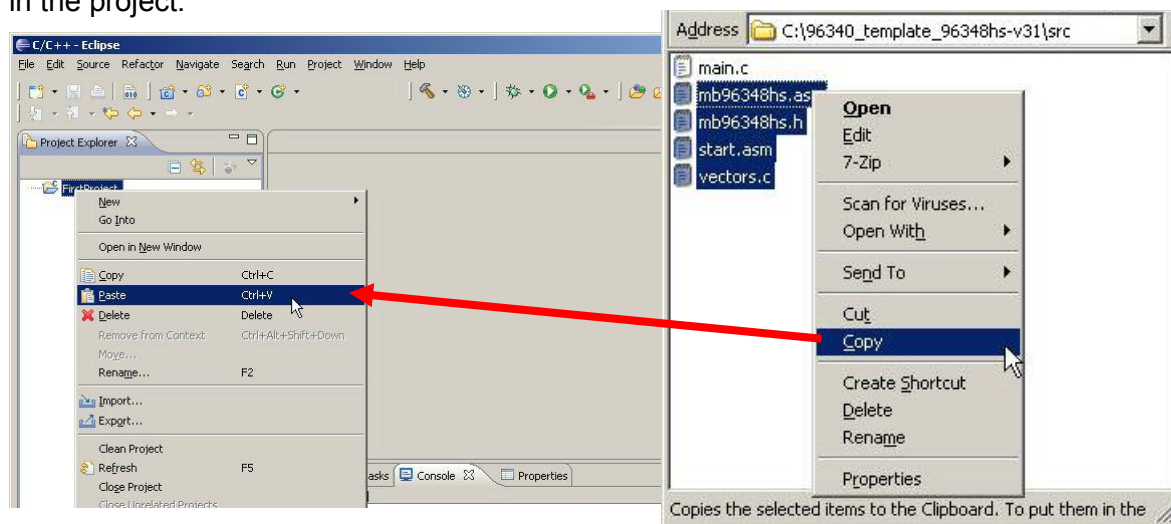


Close the dialog by clicking on OK.

### 3.3 Add the Fujitsu controller files

As with Softune it is recommended to use the template files for the used microcontroller series from Fujitsu, including the microcontroller resource definitions (e.g. *mb96f348hs.h*, *mb96f348hs.asm*), the start-up assembler code (*start.asm*) and the vector table (*vectors.c*).

The *Project Explorer* supports drag and drop so files can directly be dragged in the project from the system's file navigator. Alternatively they can be copied to the clipboard and pasted in the project.



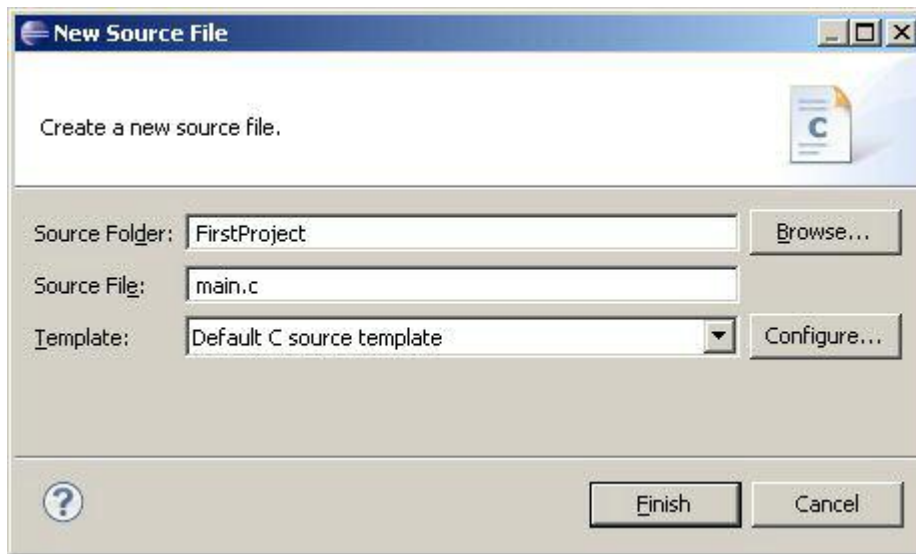
### 3.4 Create a new main.c source file

Note: Normally you would copy the *main.c* from the template just like the other files. This is just an example on how to create new source files.

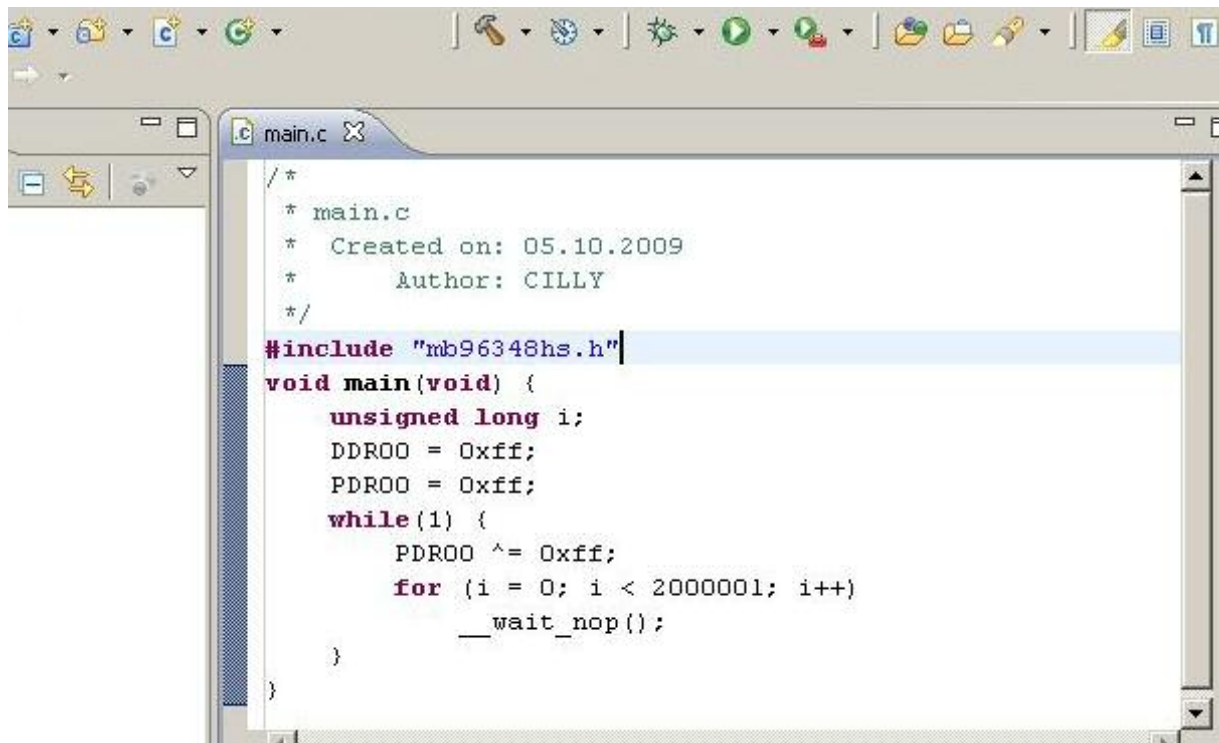
Click on *New C/C++ Source File* in the toolbar:



Enter "main.c" as name for the source file and click on *Finish*:



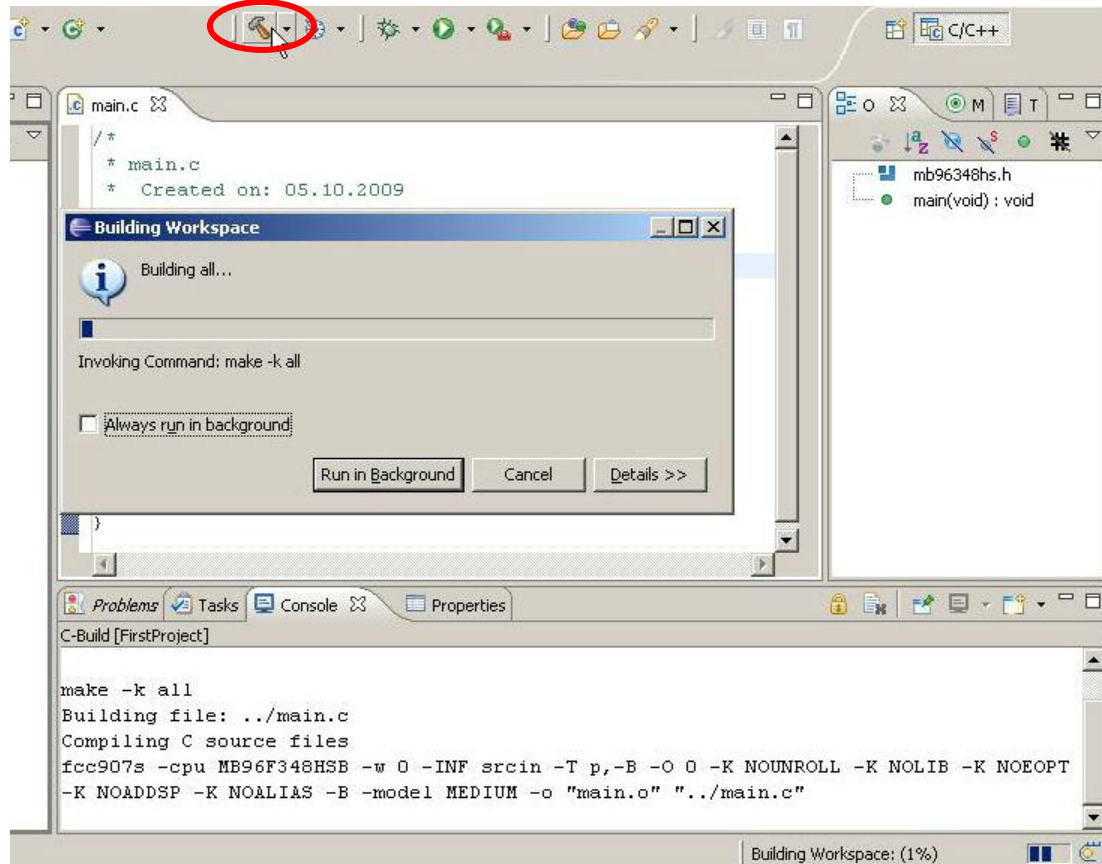
Open the new file and enter some code to test the setup (example code is for the SK-16FX-EUROSCOPE starter kit):



Save the file.

### 3.5 Compile, link and if enabled flash the application

Click on *Build project*. If everything was setup correctly the tool invocations should scroll through the console window on the bottom pane of Eclipse.



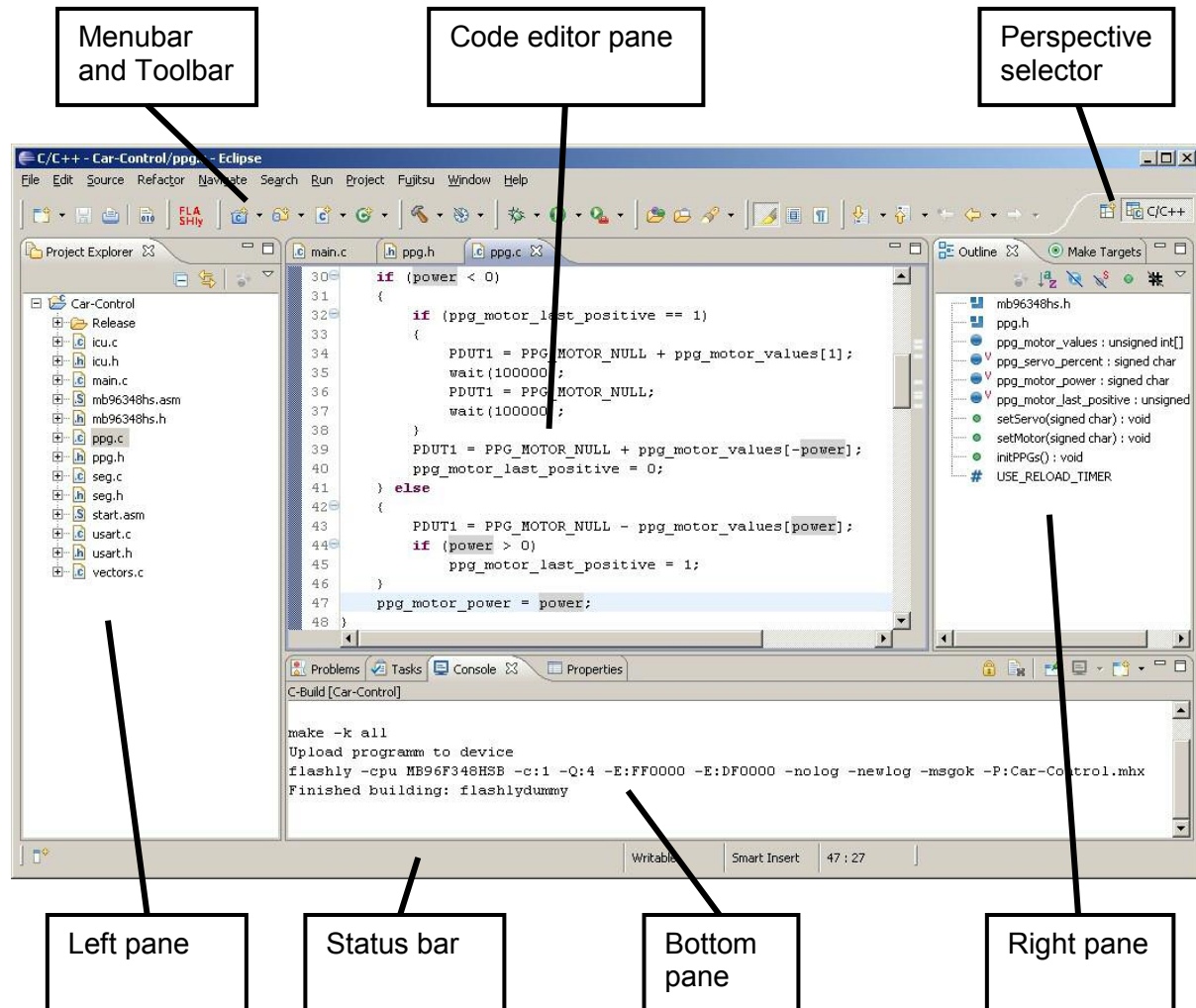
If the code compiled fine and programming with FLASHly is enabled the programmer should automatically start. Depending on flash size it will take some time to erase the chip and flash the new application:



If everything went fine FLASHly will automatically close itself.

## 4 Eclipse features

This chapter will give you a short introduction on the features Eclipse offers you while developing applications.

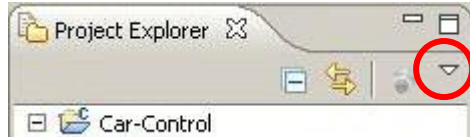


### 4.1 Special terms used by Eclipse

- Perspective
  - Describes a set of views, their sizes and positions selected by the user to be shown in an environment for a development purpose. The current perspective can be switched in the perspective selector. Examples:
    - C/C++-Perspective shows views for code, structural overview, compilation
    - SVN Repository Exploring shows views for Subversion repositories, version history, Subversion file properties
- View
  - A functional entity which can be docked to a pane. Will be explained in chapter 4.2. Examples: Project Explorer, Outline, Console
- Pane
  - Sub window in eclipse providing a tabbed environment to dock views to.

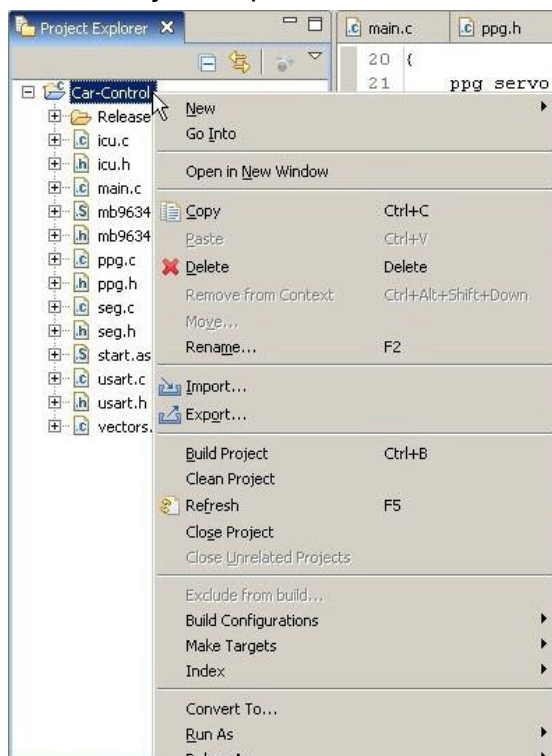
## 4.2 Views

New views can be opened through the menu point *Window – Show View*. Views can be moved to other locations inside a pane, different panes and to new panes by dragging the tab header of the view with the mouse. Many views provide settings through a menu which can be opened by clicking on the white triangle:



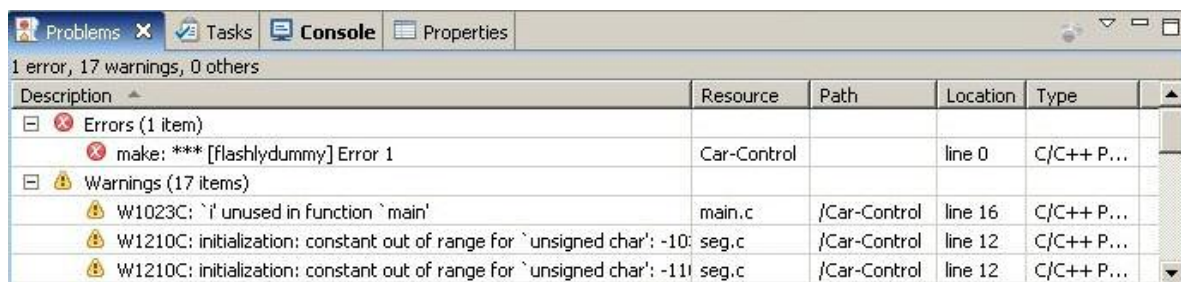
The important views for application development in C are already opened by default in new installations.

### 4.2.1 Project Explorer



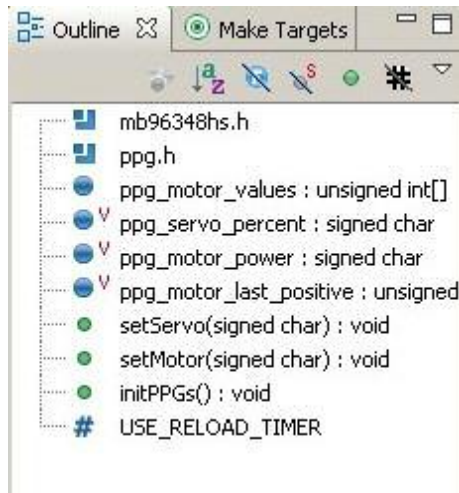
Shows the projects and their files in the currently opened workspace. The context menu of elements provides lots of functions for files and projects.

### 4.2.2 Problems



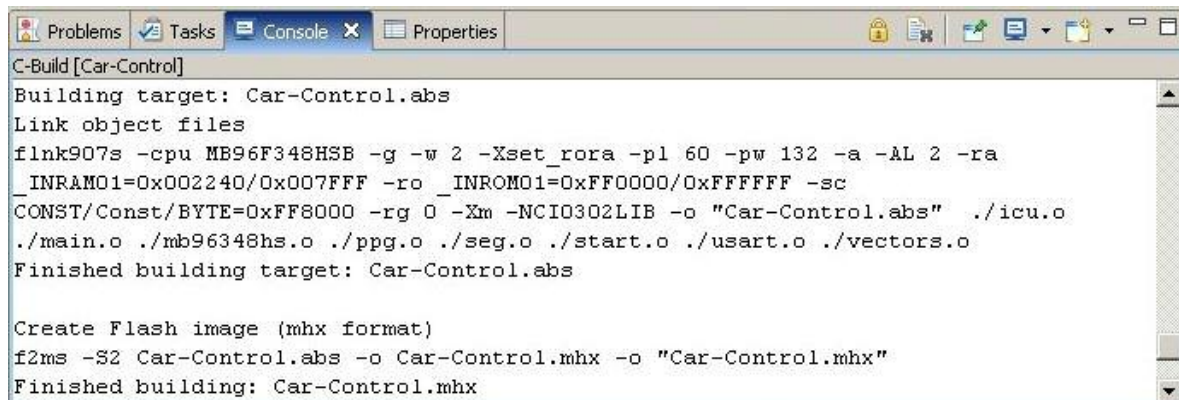
Shows the warnings and errors output by the compiler, assembler, linker. If there was a reference to a source file and line number you can directly jump to the problem by double clicking on an entry.

### 4.2.3 Outline



Provides a structural overview on the source file currently visible in the editor. By default it shows all declarations of includes, defines, variables and methods.

### 4.2.4 Console



The Console view shows the output of the tools which are run by the plug-in when building the application. Here can be seen what arguments were passed to the tools or what errors the tools may have thrown.

## 4.3 Panes

Each of the panes in Eclipse is a tabbed area where views can be docked. The code editor pane is a special pane in that it does not allow docking any view but only shows the primary elements of the currently opened perspective. A pane can be maximized to the whole Eclipse window by double clicking on its tab section.

## 4.4 Code editor

The code editor is the main part of the development environment. It includes a lot of features which can reduce the time spent on non-programming stuff like formatting.

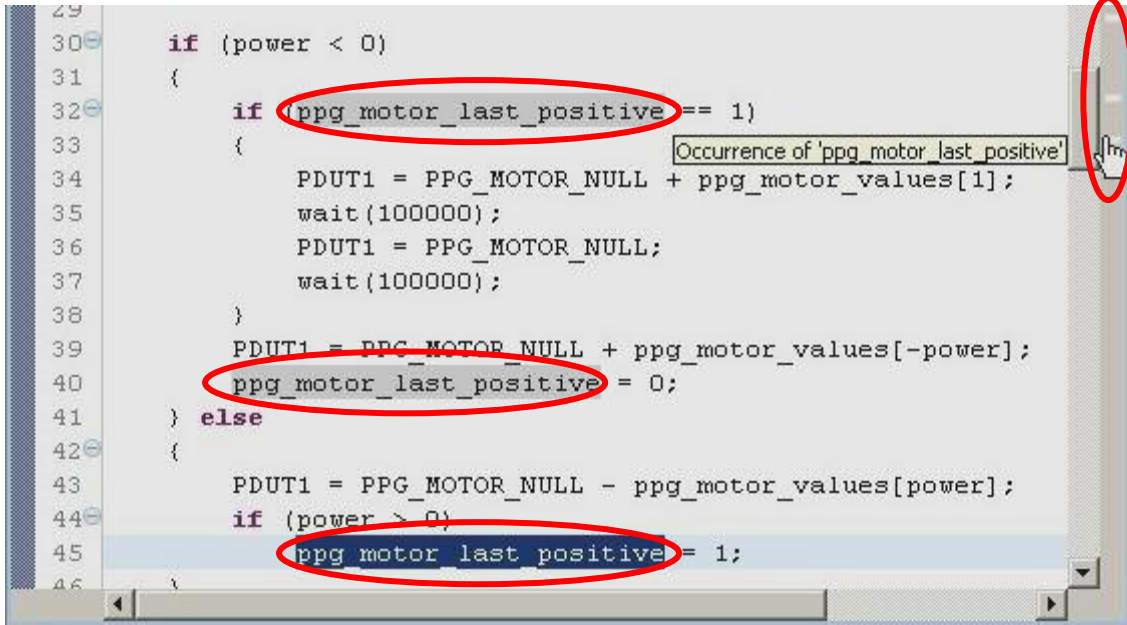
### 4.4.1 Code folding and line numbers

Neither code folding nor display of line numbers are enabled by default. To enable them open the preferences (Menu – *Window* – *Preferences*) and go to *General* – *Editors* – *Text*

*Editors.* On the right check the box *Show line numbers*. Then go to the section *C/C++ – Editor – Folding* and check the box *Enable folding when opening a new editor* and any other option you might want. Apply the changes and close the preferences dialog.

Next time a new editor tab is opened there will be small plus and minus signs in front of methods and, if selected, in front of conditional statements. By clicking on a minus sign the code section inside the block will be folded and by clicking on a plus it will be unfolded.

#### 4.4.2 Mark occurrences



If the cursor is in an identifier the editor automatically highlights occurrences of the same identifier in the file. It also shows the occurrences by putting small white dashes on the right bar of the editor to easily jump to them in the whole file.

#### 4.4.3 Code completion

When typing an identifier press *Ctrl + Space* to get a box showing all identifiers which begin with the already typed characters. You can select one with the cursor keys and enter or the mouse to complete the identifier.

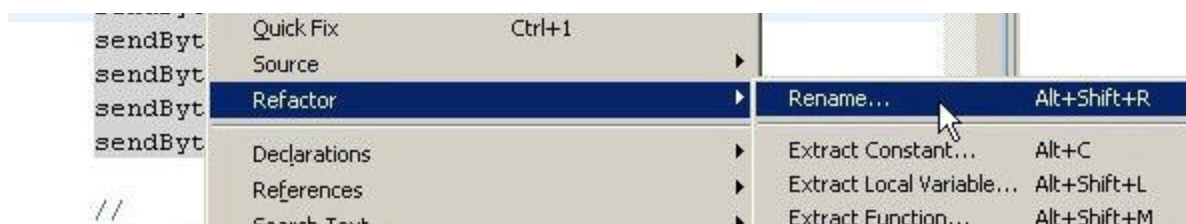
#### 4.4.4 Code formatting

If the currently opened source file should be completely formatted press *Ctrl + Shift + F*.

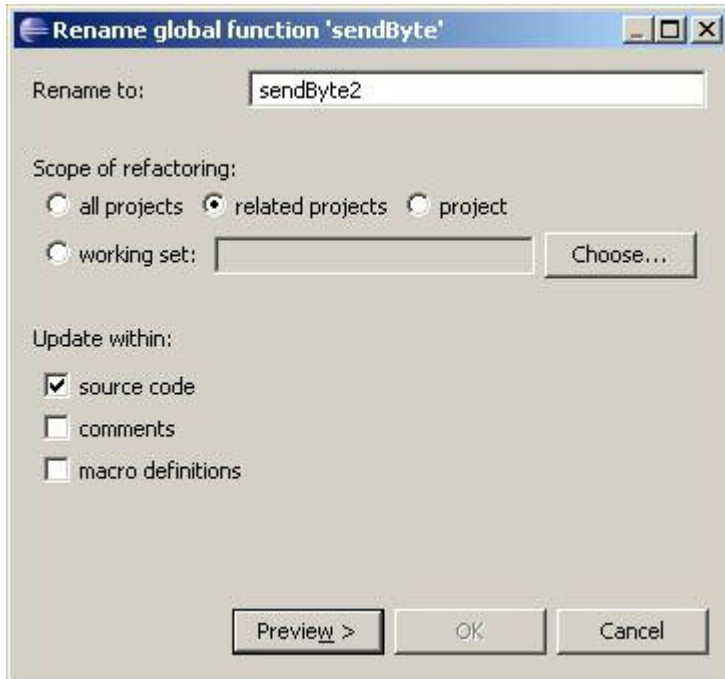
Note: The rules for the formatter can be configured in the preferences (*Menu – Window – Preferences*) under *C/C++ – Code Style*. On the right select *Edit* to get to the formatting preferences editor. The name of the profile has to be changed to save the changes.

#### 4.4.5 Refactoring

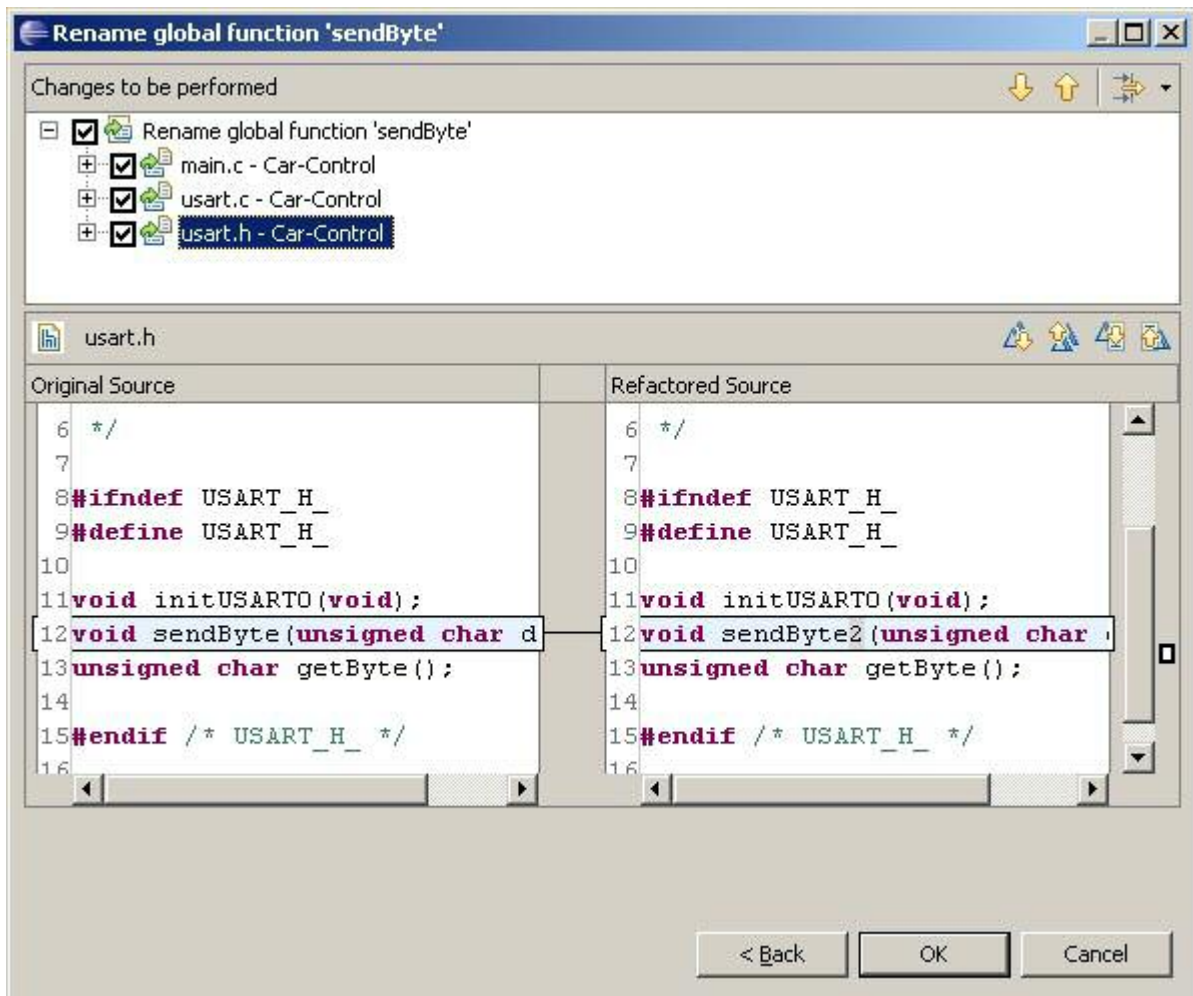
Refactoring allows for renaming identifiers in the whole project without breaking references to them. Right click on the identifier you want to rename, go to *Refactor – Rename*.



Enter a new name for the identifier and click on *Preview*.



Now the changes that would be made to the source code will be shown. If you are ok with them press *OK*.



## 4.5 Menu bar / Toolbar



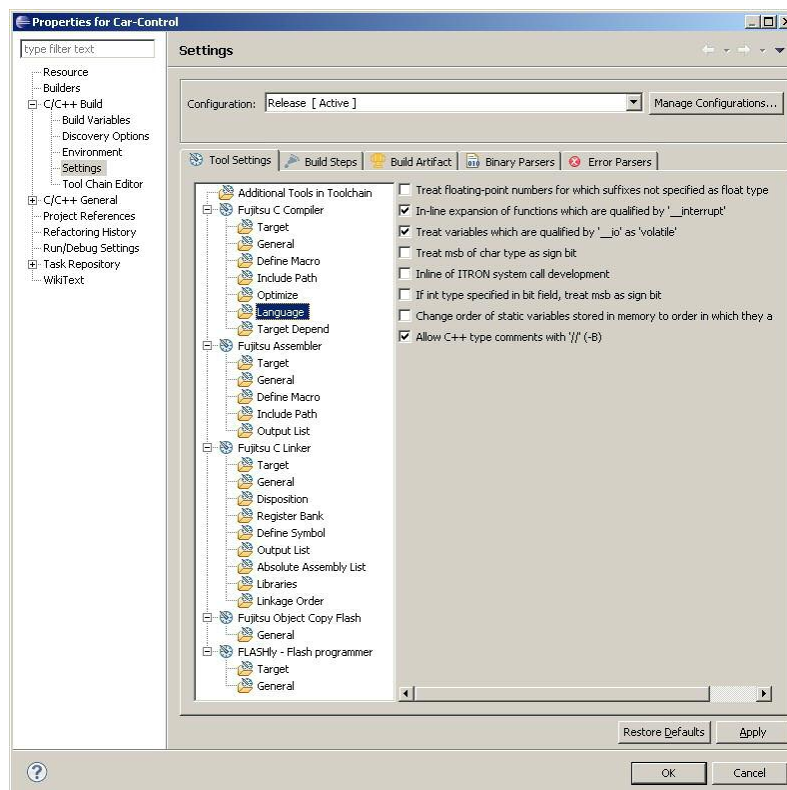
The toolbar consists of multiple sections. They are from left to right:

- File operations (New, Save, Print, Build all)
- FLASHly invocation
- File creation (New Project, New Folder, New File, New Class)
- Build options (Build project, Manage Configurations)
- Run commands (Debug, Run, Run tool)
- Search functions (Open Element, Open Task, Search)
- Editor options (Mark Occurrences, Block selection, Show special characters)
- Navigation (Go to annotations, last edited locations, backward/forward)

Many functions in Eclipse can also be reached with hotkeys which can also be defined by the user in the preferences under *General – Keys*.

## 4.6 Project settings

The project properties dialog can be opened by right clicking on a project in the *Project Explorer* and clicking on *Properties* (or the menu *Project – Properties*). The project properties dialog controls many project related options, most importantly the build process. The settings for the different tools can be found under *C/C++ Build – Settings*. The tree to the right resembles the project settings of Softune where the top-level elements of the tree correspond to the tabs in Softune and the second-level elements correspond to the categories of the drop-down boxes in Softune. The options themselves are mostly exactly the same.



## 5 Appendix

### 5.1 Links to Fujitsu

- Softune Workbench:  
[http://mcu.emea.fujitsu.com/mcu\\_tool/detail/SWB\\_\(F2MC-16\)\\_V3.htm](http://mcu.emea.fujitsu.com/mcu_tool/detail/SWB_(F2MC-16)_V3.htm)

### 5.2 Links to third party sites

- MinGW / MSYS: <http://www.mingw.org>
- Stripped MSYS package: <http://fujitsu.chrilly.net/eclipse>
- FLASHly: <http://www.holgerium.de/elektronik/index.htm>
- Java Runtime Environment: <http://java.sun.com/javase/downloads/index.jsp>
- Eclipse: <http://www.eclipse.org/downloads/>
- Fujitsu plug-in update site: <http://fujitsu.chrilly.net/eclipse/updatesite>
- Fujitsu plug-in packages for manual installation:  
<http://fujitsu.chrilly.net/eclipse/updatesite/plugins/>