

**JBLopen**

Embedded Software Insight

---

# IAR Embedded Workbench Eclipse Setup Guide

WP0002 Version 2

March 2, 2021

**Copyright**

© 2017-2021 JBLOpen Inc.

All rights reserved. No part of this document and any associated software may be reproduced, distributed or transmitted in any form or by any means without the prior written consent of JBLOpen Inc.

**Disclaimer**

While JBLOpen Inc. has made every attempt to ensure the accuracy of the information contained in this publication, JBLOpen Inc. cannot warrant the accuracy or completeness of such information. JBLOpen Inc. may change, add or remove any content in this publication at any time without notice.

All the information contained in this publication as well as any associated material, including software, scripts, and examples are provided “as is”. JBLOpen Inc. makes no express or implied warranty of any kind, including warranty of merchantability, noninfringement of intellectual property, or fitness for a particular purpose. In no event shall JBLOpen Inc. be held liable for any damage resulting from the use or inability to use the information contained therein or any other associated material.

**Trademark**

JBLOpen, the JBLOpen logo, TREEspan™ and BASEplatform™ are trademarks of JBLOpen Inc. All other trademarks are trademarks or registered trademarks of their respective owners.



---

# Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
1.1	About IAR Embedded Workbench . . . . .	1
1.2	About Eclipse . . . . .	1
<b>2</b>	<b>Tools Setup</b>	<b>2</b>
2.1	Note About the Java Runtime Environment . . . . .	2
2.2	Note About MSYS2 and Makefile Projects . . . . .	2
2.3	IAR Embedded Workbench Installation . . . . .	2
2.4	Eclipse Installation . . . . .	9
2.5	IAR Embedded Workbench for Eclipse Installation . . . . .	14
2.6	MSYS2 Installation . . . . .	21
2.7	Environment Variable Setup . . . . .	30
2.7.1	PATH Environnement Variable Setup . . . . .	30
<b>3</b>	<b>Managed Build Project Setup</b>	<b>34</b>
3.1	IAR Eclipse Managed Build Project Setup . . . . .	34
3.2	IAR Eclipse Managed Build Project Debugging . . . . .	41
<b>4</b>	<b>Makefile Project Setup</b>	<b>47</b>
4.1	IAR Eclipse Makefile Project . . . . .	47
4.2	IAR Eclipse Makefile Project Debugging . . . . .	59
<b>5</b>	<b>Document Revision History</b>	<b>65</b>

---

# Overview

This document provides a series of step-by-step guides to help readers install and integrate the IAR Embedded Workbench® toolchain within the Eclipse IDE on Microsoft Windows®. This document will go over the installation and configuration required to use the IAR Embedded Workbench for Eclipse plugin which is developed and distributed by IAR. The instructions written therein are primarily for the IAR ARM Embedded Workbench (EWARM) which supports both building and debugging from within Eclipse. Other editions of IAR for various targets are also supported but only for building. This guide is also not aimed at Linux users. A separate installation of IAR is available to support building on Linux but this is not covered in this guide. Note that there are many combinations of Eclipse and IAR versions as well as more than one way of installing the IAR Eclipse plugin. This document is the recommended way by JBLopen but customers are free to modify their development environment to suit their needs.

Readers are encouraged to also look at the official installation instructions for the IAR Eclipse Plugin which can be found at <http://eclipse-update.iar.com/>.

## 1.1 About IAR Embedded Workbench

IAR Embedded Workbench is a complete set of compiler toolchain, debugger and IDE for embedded software development targeting multiple CPU architectures and device manufacturers. Additional information about IAR Embedded Workbench can be found on the [IAR website](#).

A free code size limited edition of the IAR Embedded Workbench for ARM is available from the IAR website. Additionally, users can activate a one time, full featured, 30-day trial licence. Both trial editions can be used to test the steps included in this guide.

## 1.2 About Eclipse

Eclipse is a cross-platform IDE that supports multiple programming languages. This document will mainly use the Eclipse CDT environment for C and C++. Additional information about the Eclipse IDE can be found on the [Eclipse.org website](http://Eclipse.org).

## Tools Setup

---

This section will go over the Eclipse IDE and the IAR toolchain setup. The following instructions will go through the installation and configuration of the following components in order:

- Eclipse – 2020.12 – eclipse-inst-jre-win64.exe  
[www.eclipse.org](http://www.eclipse.org)
- IAR Embedded Workbench for ARM – 8.50.9 – EWARM-CD-8509-33462.exe  
[www.iar.org](http://www.iar.org)
- MSYS2 (Optional) – 20200903 – msys2-x86\_64-20200903.exe  
[www.msys2.org](http://www.msys2.org)

The versions cited above are those that were used when writing this guide. Installation instructions for newer versions, if available, should be similar.

### 2.1 Note About the Java Runtime Environment

Eclipse is now packaged with a suitable JRE. However using the JRE over a JDK may cause some dependency issue when installing the IAR Eclipse Plugin. This guide will use JDK version 14. To ensure that the JDK is used it must be installed prior to installing Eclipse and must be selected on the Eclipse install dialog as shown in the guide.

### 2.2 Note About MSYS2 and Makefile Projects

The following guide includes instructions on how to install MSYS2. This is optional to use the IAR Embedded Workbench within Eclipse and is only useful when creating Makefile projects which require GNU Make to be available. Readers who are not interested in creating Makefile projects using IAR do not need to install MSYS2 for the purpose of this guide.

### 2.3 IAR Embedded Workbench Installation

The first package to install is the IAR Embedded Workbench for ARM. Running the IAR installer, EWARM-CD-8509-33462.exe in this example, the following dialog should appear.

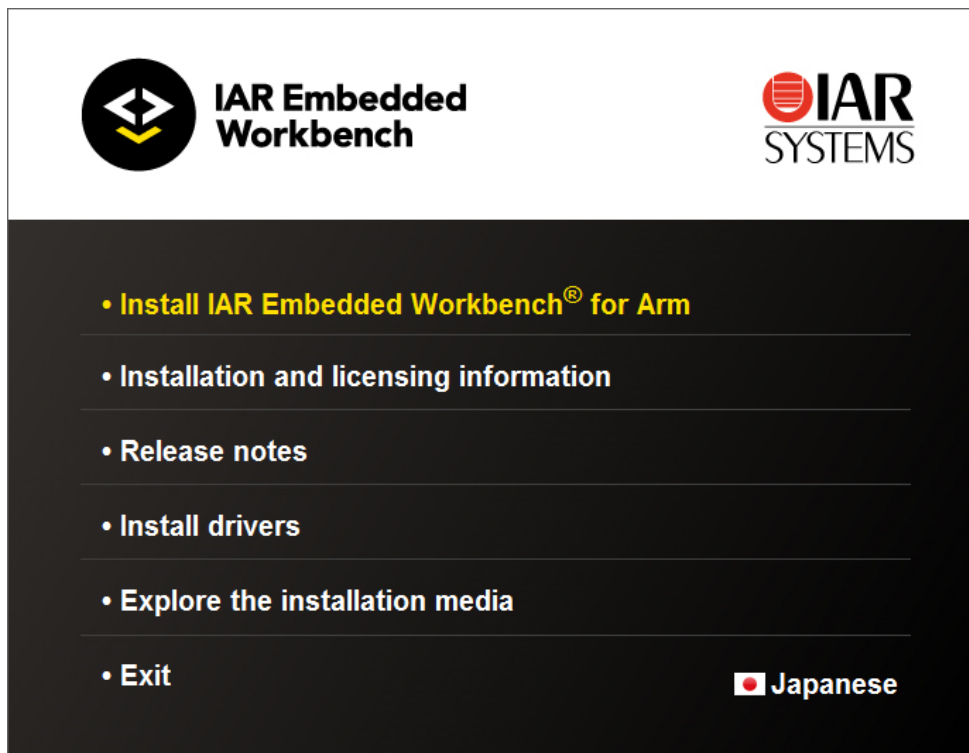


Figure 1 – IAR installer

On the installer menu select "Install IAR Embedded Workbench for ARM". This will start the IAR EWARM installation wizard proper.

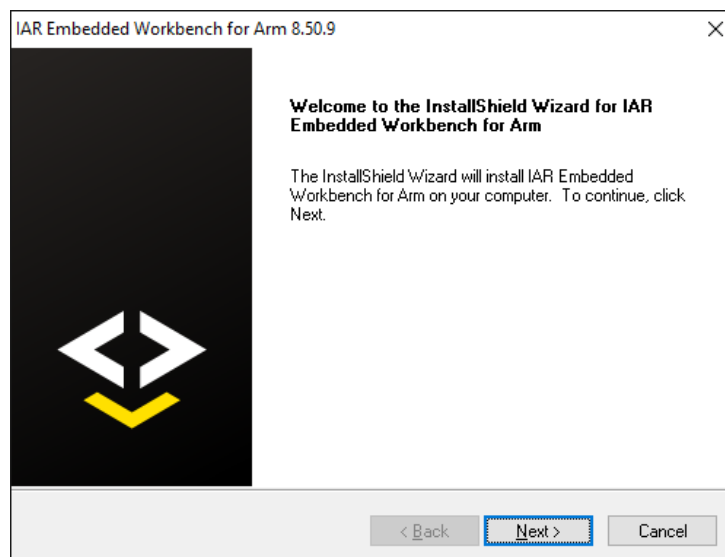
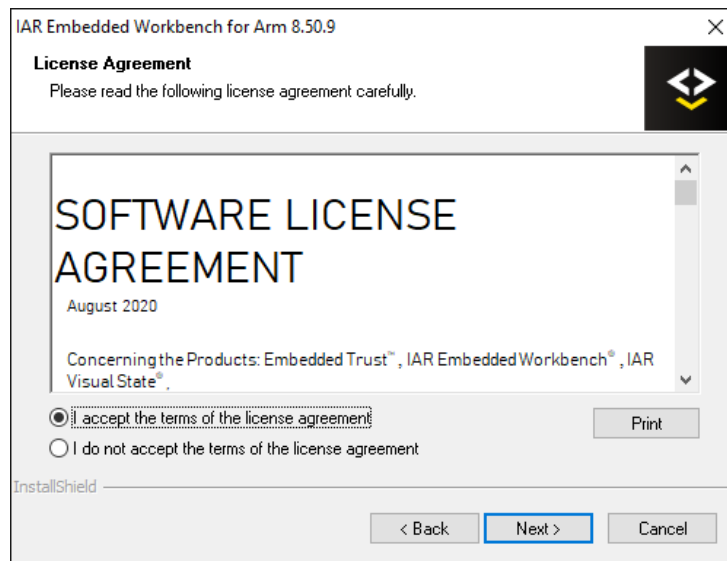


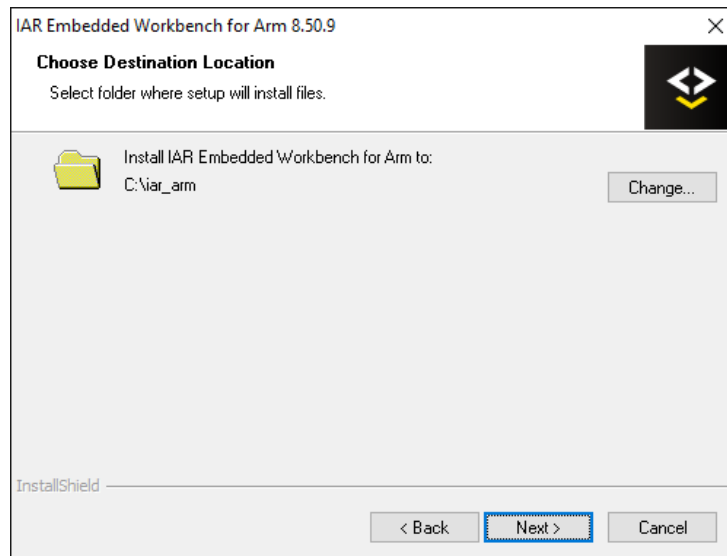
Figure 2 – IAR EWARM installation welcome dialog

Click "Next" to display the license agreement.



**Figure 3** – License agreement

Accept the license agreement and click "Next" again to move to the installation path selection screen.



**Figure 4** – Installation path selection

The wizard will install the IDE within the usual "Program Files" directory. If, however, you intend to use the IAR toolchain with custom Makefiles it is recommended to install IAR within a different directory with a short path containing no spaces or special characters. In this example, C:/iar\_arm will be used as the installation directory.

Click "Next" to continue the installation wizard and display the drivers installation selection screen.

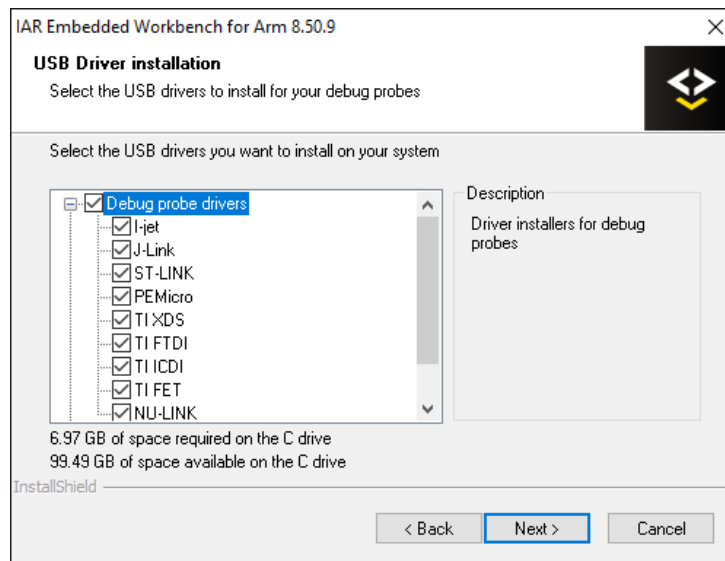


Figure 5 – Debug probe and dongle driver selection

By default all the debug probe drivers will be selected. Drivers that aren't needed can be removed from the selection if desired. Click "Next" to continue.

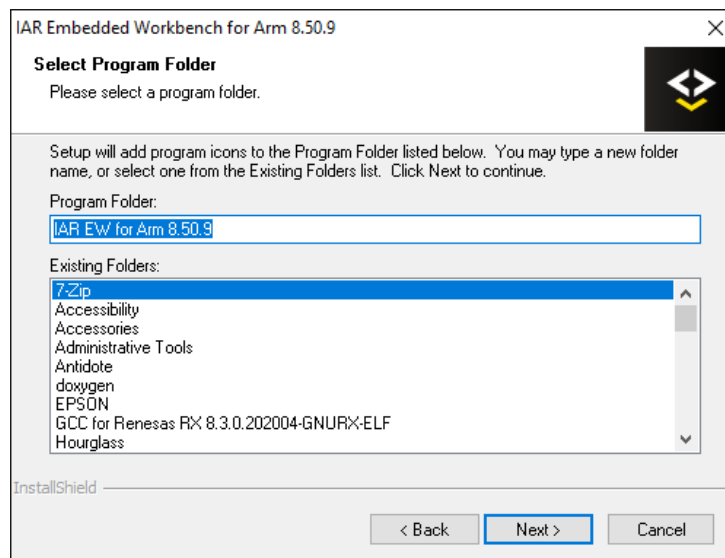


Figure 6 – Windows start menu item dialog

Change the start menu entry if desired and click "Next" again.



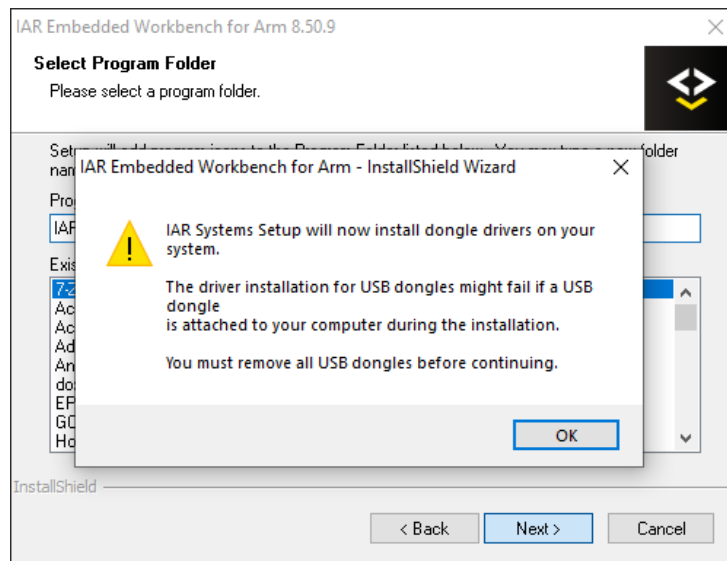


Figure 7 – Dongle driver USB warning

If the dongle driver was selected in the driver selection screen, a warning dialog will be displayed instructing the user to remove any IAR dongles present on the host machine before proceeding. Click "OK" to dismiss the warning.

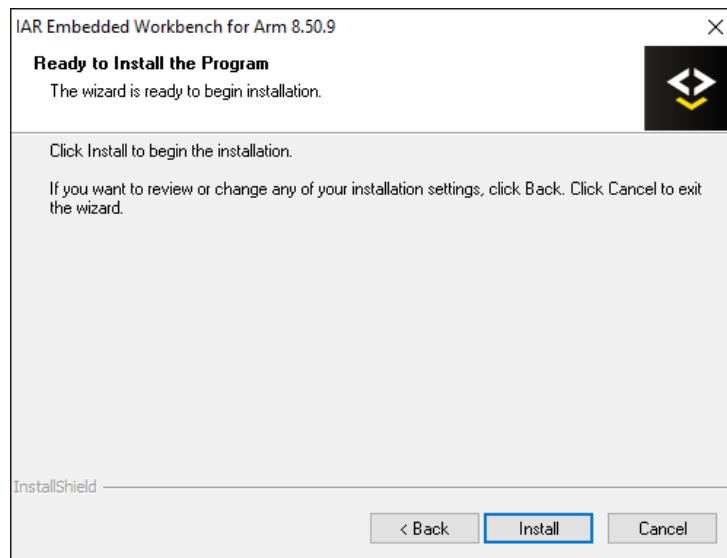


Figure 8 – EWARM installation ready

Click "Install" to begin the installation process which may take a few minutes.

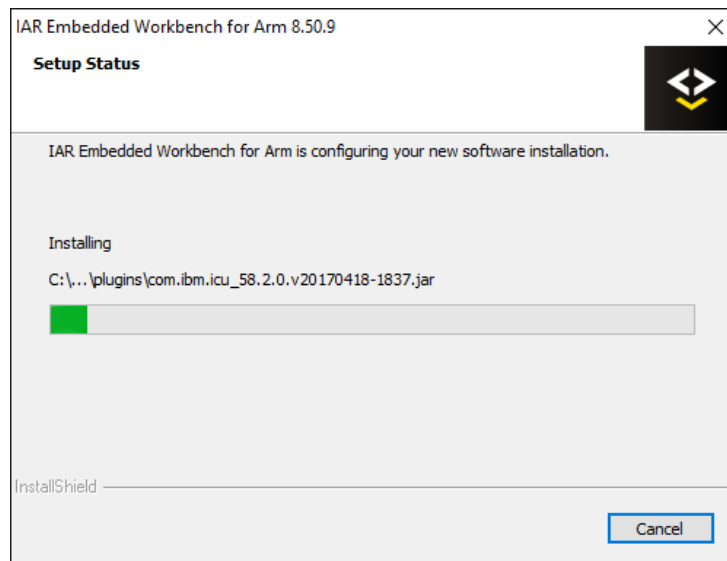


Figure 9 – EWARM installation underway

The installation will proceed and once finished the installation completed screen should appear.

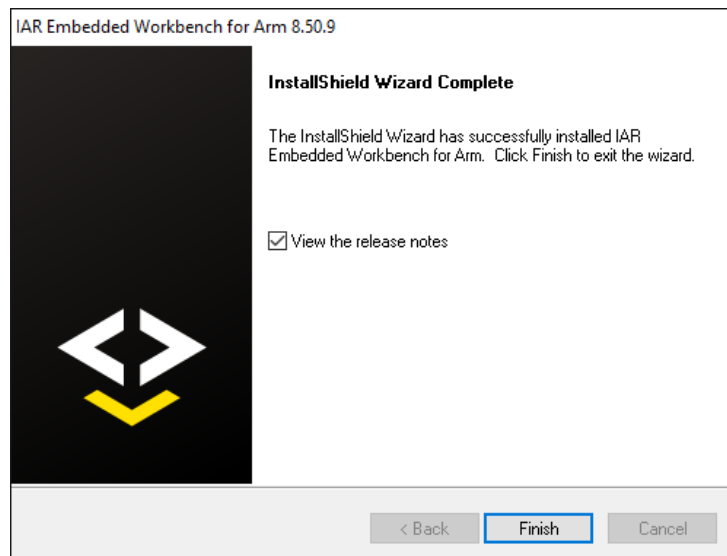


Figure 10 – EWARM installation completed

On the installation completed screen, you have the option to view the release note which will open in the default web browser. Otherwise click "Finish" to end the installation.

From the created start menu entry, launch the IAR Embedded Workbench that was just installed.

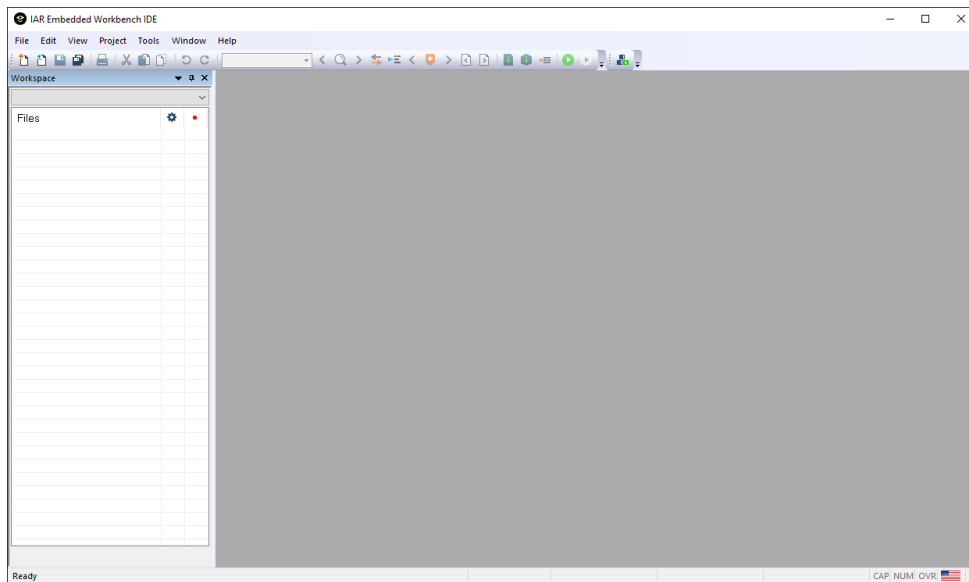


## IAR Embedded Workbench



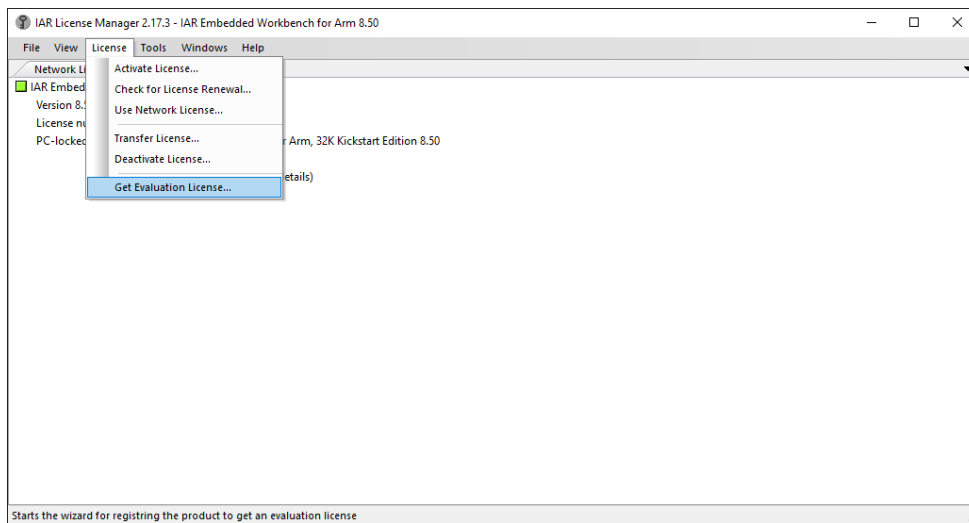
**Figure 11** – IAR Embedded Workbench for ARM splash screen

A splash screen will be displayed briefly while the IDE loads, then the IDE should appear.



**Figure 12** – EWARM IDE

If there is no valid license installed on the host computer, the IAR License Manager will start automatically. From there you can either install an available purchased license or request a code size limited or time-limited evaluation license.



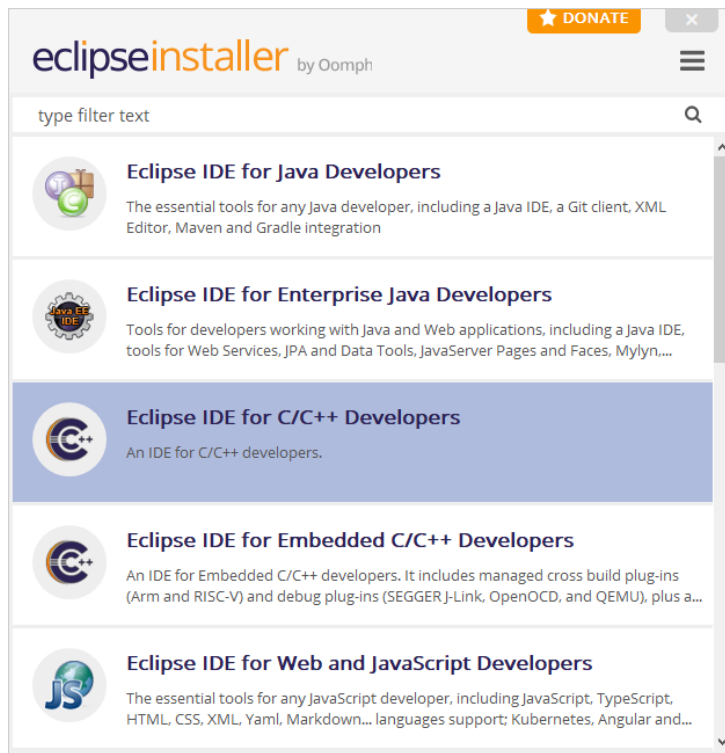
**Figure 13 – IAR License Manager**

Follow the instructions on screen to install the selected license type.

Once done with the IAR installation and license setup close the IAR IDE for the moment to proceed with the Eclipse installation.

## 2.4 Eclipse Installation

The next package to install is the Eclipse IDE. Begin by executing the installer file. Note that if possible the 64-bit version of Eclipse is strongly recommended. In this example `eclipse-inst-jre-win64.exe`, the following screen should appear.



**Figure 14** – Eclipse installation welcome screen

Select "Eclipse IDE for C/C++ Developers" by clicking it. Note that it's also possible to use the "Eclipse IDE for Embedded C/C++ Developers" but the extra features provided by the embedded developers edition aren't useful when using the IAR Eclipse plugin.

The installation path and JVM selection dialog should appear.



**Figure 15** – Installation path and JVM selection

In the first selection box, the Java Virtual Machine to be used by Eclipse must be selected. By default the installer will automatically select a suitable JRE or JDK if one is found on the host machine. Otherwise the bundled JRE will be used. As mentioned earlier in this guide, JDK version 14 or later is recommended to prevent dependency issues when installing the IAR plugin. This guide uses JDK 14.0.2.

For the installation directory, it is recommended to use a unique directory that is short with no spaces in the name as the eclipse install directory. In this example we will use `C:/eclipse_iar` as the install directory. The eclipse installer will automatically create an eclipse subdirectory inside the specified installation path. Finally, it is possible to create a start menu entry and/or a desktop shortcut if desired.

Click the "Install" button to begin the installation process.



**Figure 16** – Installation progress dialog

By default the installer will download Eclipse during the installation process which may take longer on slow internet connections. An offline installer is available from [eclipse.org](http://eclipse.org) if necessary.

The installation should eventually finish.

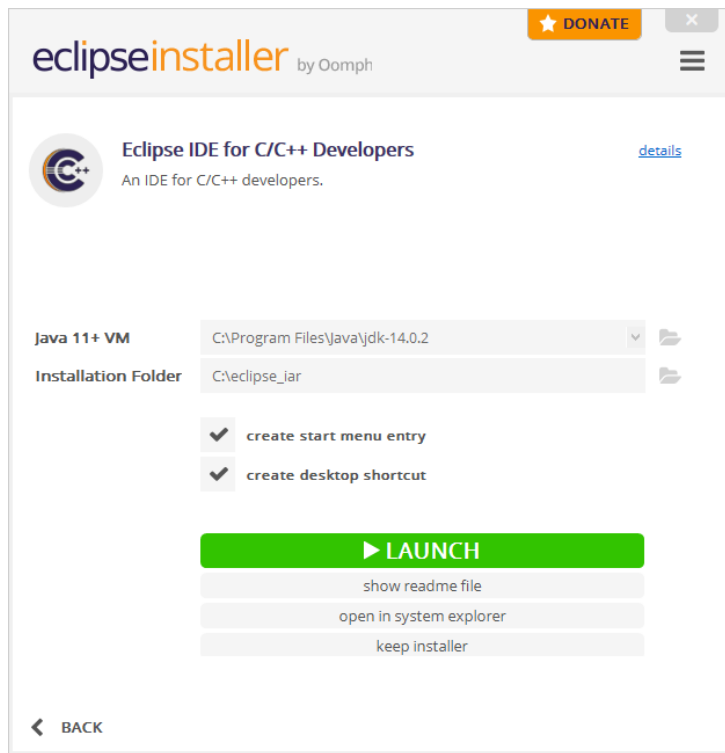


Figure 17 – Installation successful

Click "Launch" to check that the Eclipse installation works properly. The workspace selection dialog should appear.

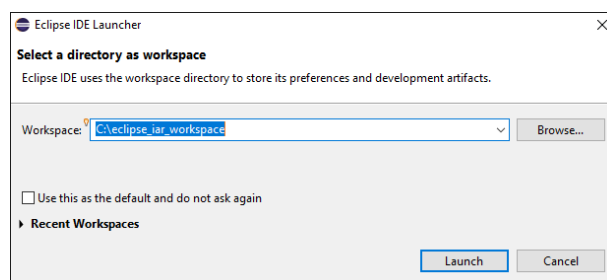


Figure 18 – Workspace selection dialog

It is strongly recommended to create a unique workspace that will be used exclusively for the projects related to the IAR toolchain that will be installed in this guide. The workspace selected should not have a path name that is excessively long and it is recommended not to use spaces in the path name either. For the purpose of this guide C:/eclipse\_iar\_workspace will be used as the workspace.

Click "Launch".





Figure 19 – Eclipse splash screen

After a few seconds, the Eclipse welcome screen should appear.

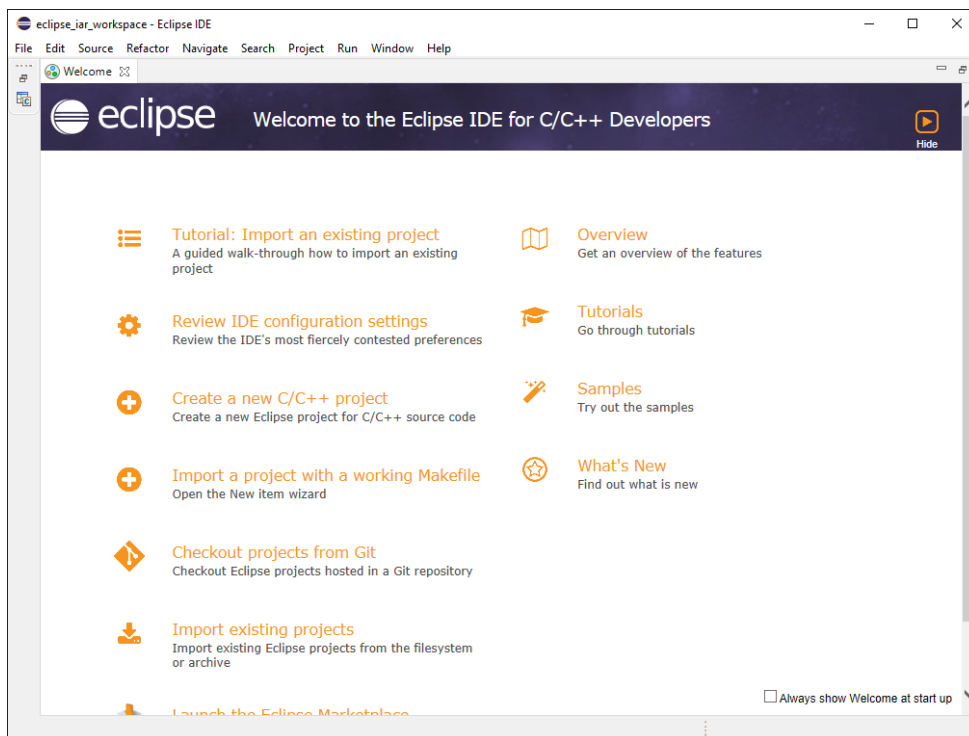
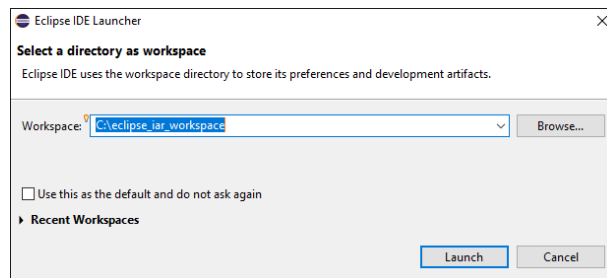


Figure 20 – Eclipse welcome screen

## 2.5 IAR Embedded Workbench for Eclipse Installation

After having installed Eclipse and the IAR Embedded Workbench it is now time to install the IAR Eclipse plugin manager and register the IAR ARM toolchain installed previously.

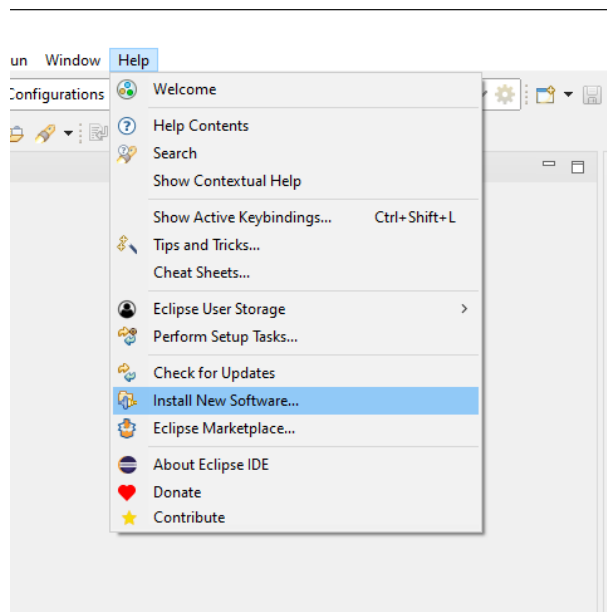
Launch Eclipse if it's not already started. The workspace selection dialog should appear.



**Figure 21** – Workspace selection dialog

If one wasn't created earlier, you must select a directory for the Eclipse workspace. It is strongly recommended to create a unique workspace that will be used exclusively for the projects related to the IAR toolchain that will be installed in this guide. The workspace selected should not have a path name that is excessively long and it is recommended not to use spaces in the path name either. For the purpose of this guide `C:/eclipse_iar_workspace` will be used as the workspace. Click "Launch" to open the Eclipse IDE with the selected workspace.

First things first, find the well hidden "Install New Software" from the Help menu.



**Figure 22** – Install New Software menu item

Clicking it will open the Eclipse update site manager.

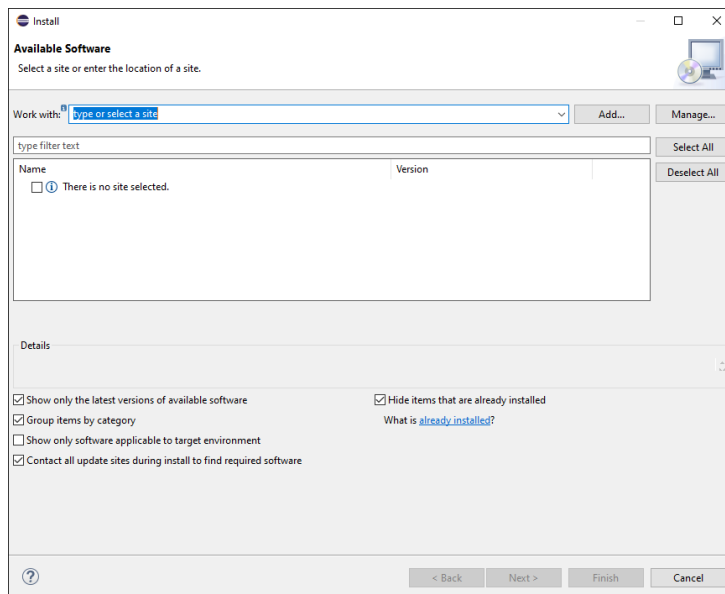


Figure 23 – Install New Software dialog

Type `http://eclipse-update.iar.com/plugin-manager/1.0` in the "Work with:" field and press enter. The screen should update to show the available software from the IAR update site.

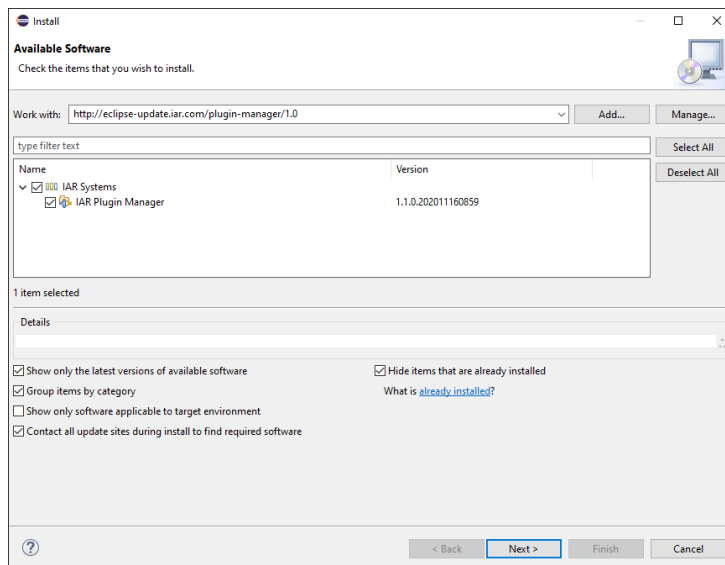


Figure 24 – Install New Software selection

Expand the "IAR Systems" line item and click the checkbox next to "IAR Plugin Manager". Then click "Next" to be taken to the installation summary screen.

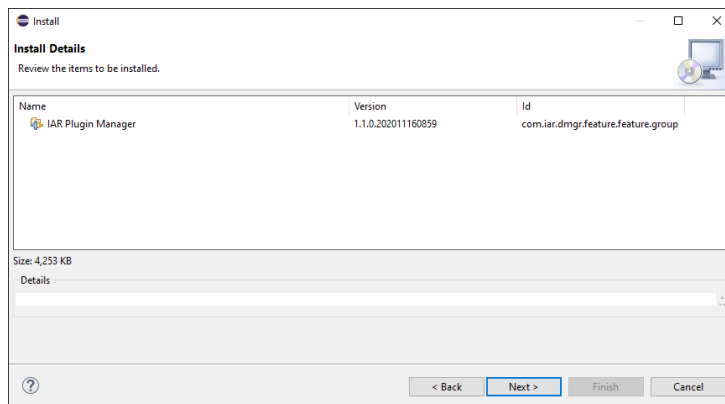


Figure 25 – Installation summary screen

Click "Next" again to be taken to a license agreement screen.

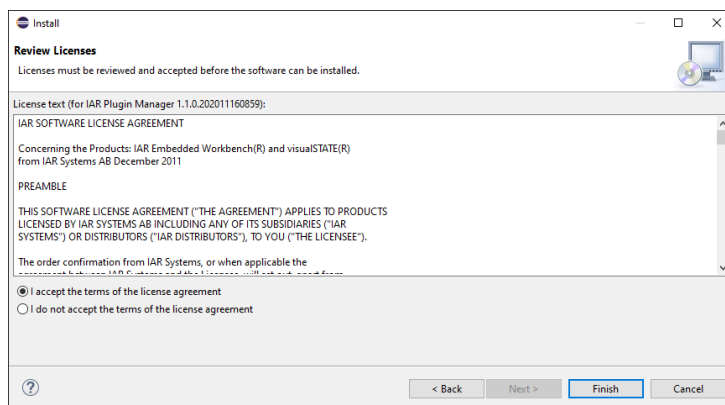


Figure 26 – Plugin license agreement

Make sure to accept the license and click "Finish". The installation will proceed with the progress displayed in the right bottom corner of the IDE. Do not close Eclipse until the installation completes.

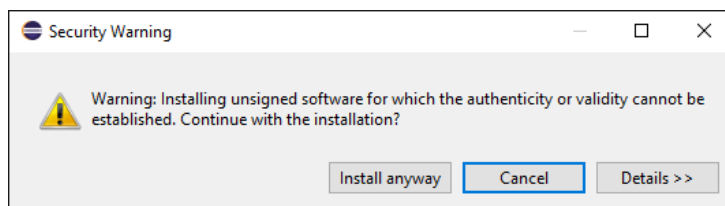
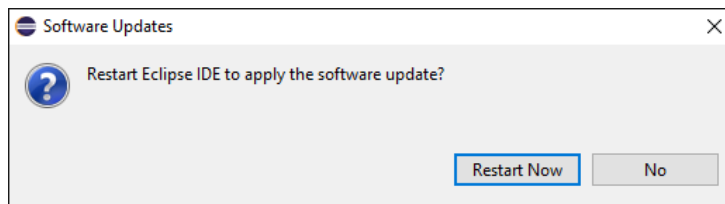


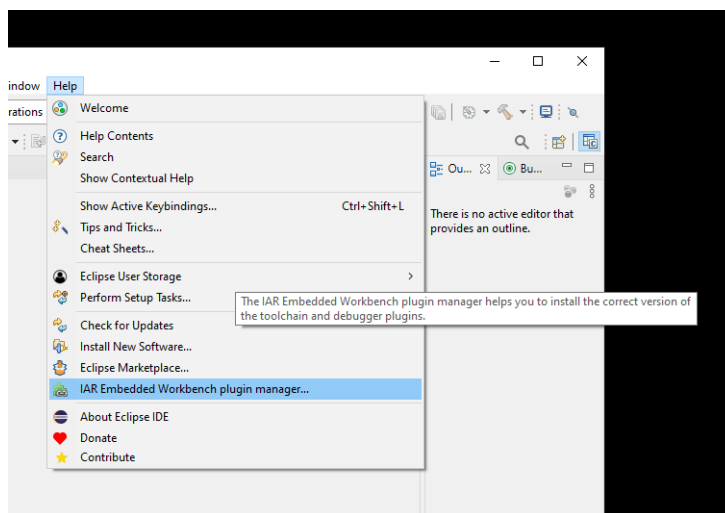
Figure 27 – Plugin installation security warning

It's possible that a warning dialog appears warning against installing unsigned content. Click "Install anyway" to progress with the plugin installation.



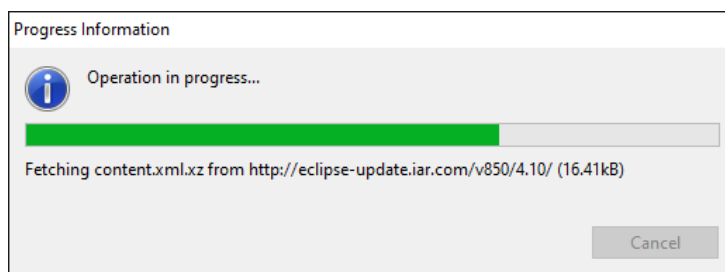
**Figure 28** – Restart prompt after plugin installation

Eventually, when the installation has completed, a dialog prompt will ask to restart Eclipse to enable the newly installed plugin. Do so by clicking "Restart Now".



**Figure 29** – IAR plugin manager menu item

After the IDE restarted, there will be a new menu item under the "Help" menu named "IAR Embedded Workbench plugin manager". Click it to open the manager.



**Figure 30** – IAR plugin manager loading

Loading the manager for the first time might take a few seconds as it queries the updated plugin online.

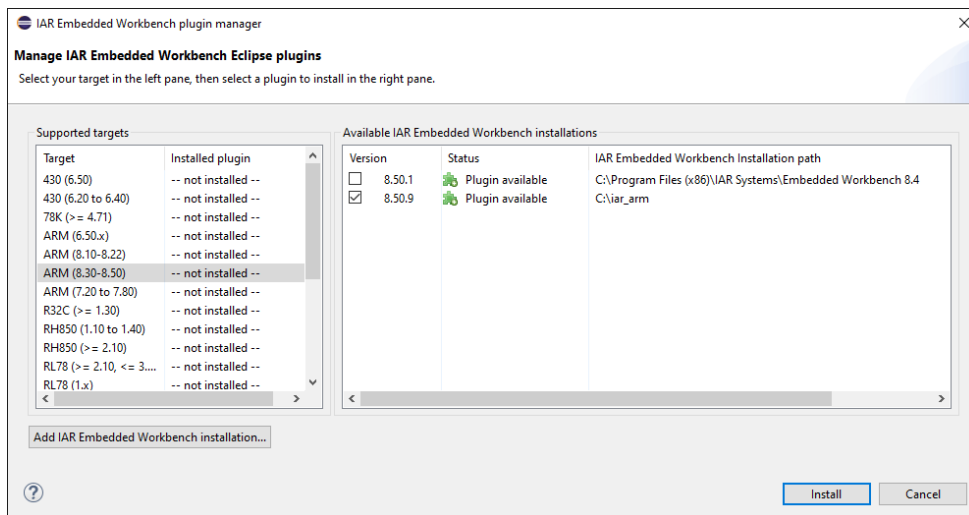


Figure 31 – IAR plugin manager

Once fully loaded the manager will display all the different versions and targets supported by the IAR Eclipse plugin. Clicking on "ARM (8.30-8.50)" should display the toolchain installed earlier in this guide.

Select the installed toolchain, here in this example it is version 8.50.9 installed in the iar\_arm directory. Note that any other existing installation will also be displayed as shown in the example there was an existing installation of IAR 8.50.1 in the program files directory. If it is the case that more than one toolchain is displayed make sure you select the one you want to use within Eclipse. After selecting the toolchain click "Install".

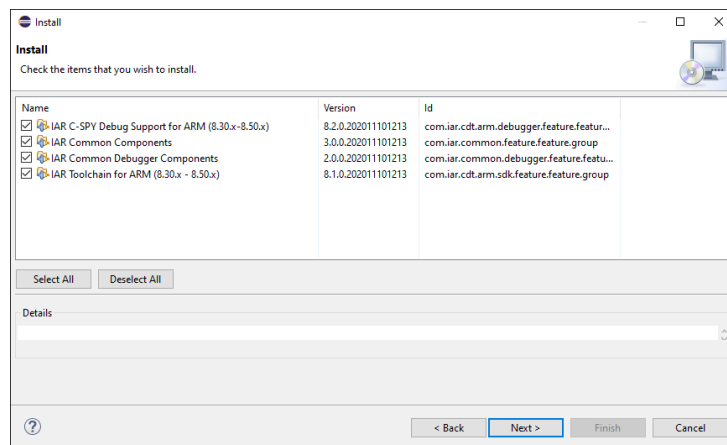


Figure 32 – IAR plugin manager install selection

In the Install screen make sure to select all the available components to install. While it may be possible to remove one of the components if it's not needed, this can cause issues later. Click "Next" to proceed.

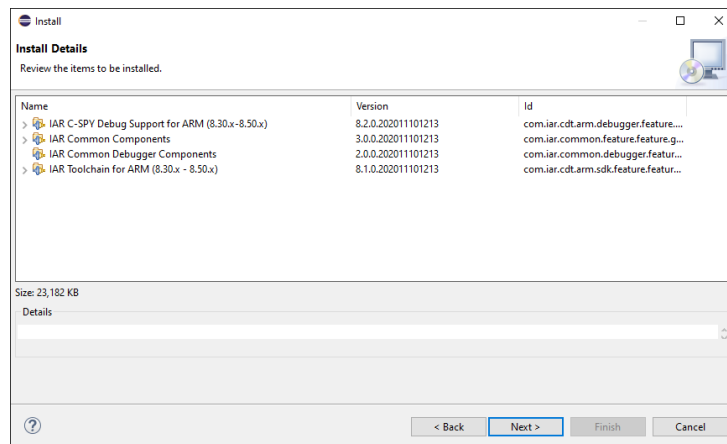


Figure 33 – IAR plugin manager install summary

On the installation summary screen click "Next" again to be taken to another license agreement screen.

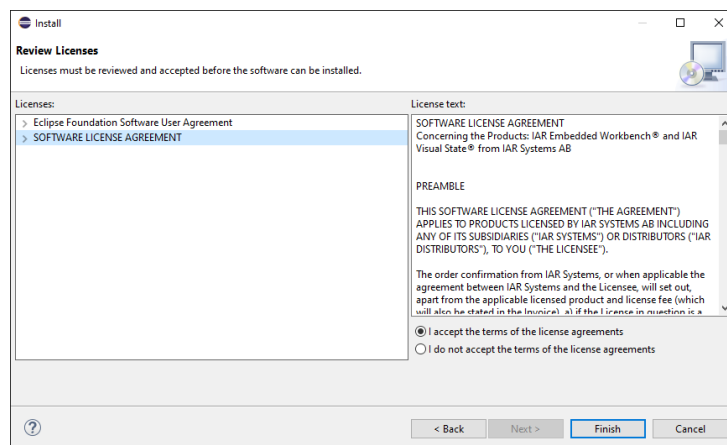


Figure 34 – IAR plugin manager license agreement

Make sure to accept the various licenses and click "Finish". The installation will proceed, again the progress is displayed in the lower right corner of the IDE. Once finished another prompt to restart Eclipse will appear.

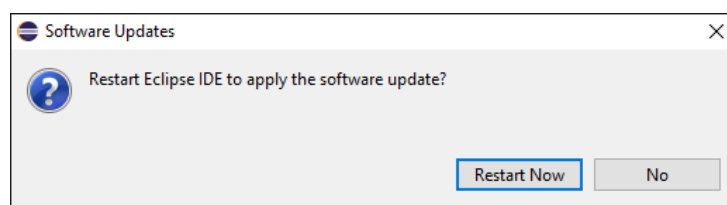


Figure 35 – IAR plugin manager restart prompt

Click "Restart Now" to restart Eclipse. If all went well, the IAR toolchain is now usable within Eclipse.

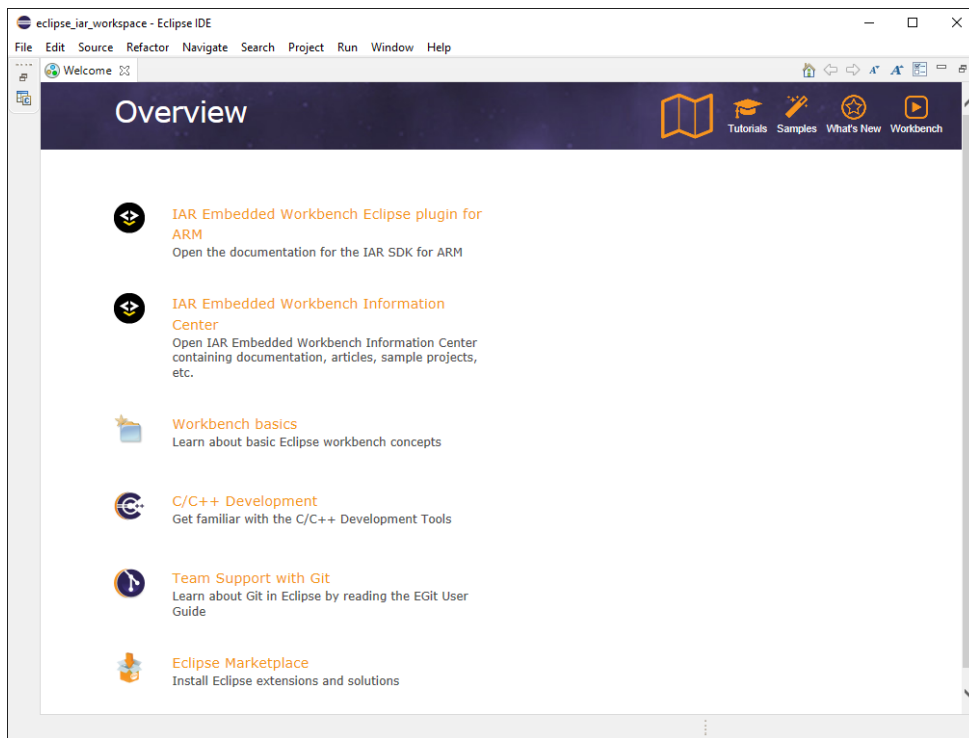


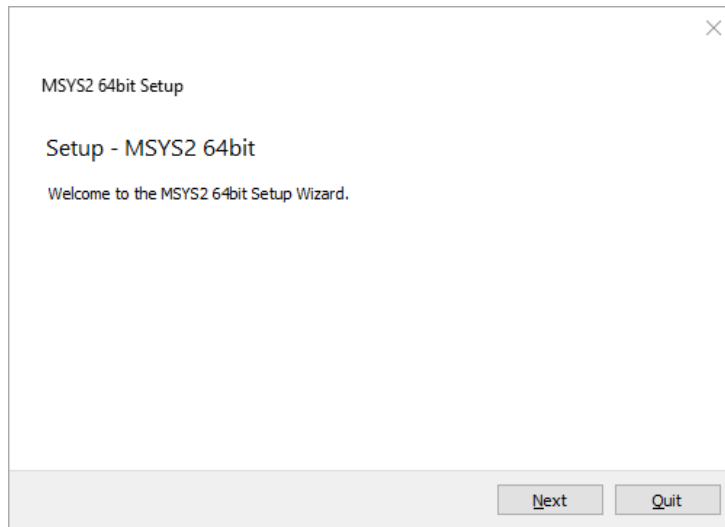
Figure 36 – IAR Eclipse welcome screen

## 2.6 MSYS2 Installation

MSYS2 is used to provide the GNU Make utility and is not necessary to use the IAR Eclipse Plugin but can be useful when writing Makefile projects. There are multiple options when it comes to installing a basic UNIX environment in Windows, including the well-known Cygwin. This guide will use MSYS2 since it offers superior build performance and good compatibility with native Windows paths. If compatible with the host system, the 64-bit version of MSYS2 is recommended, which is labelled x86\_64.

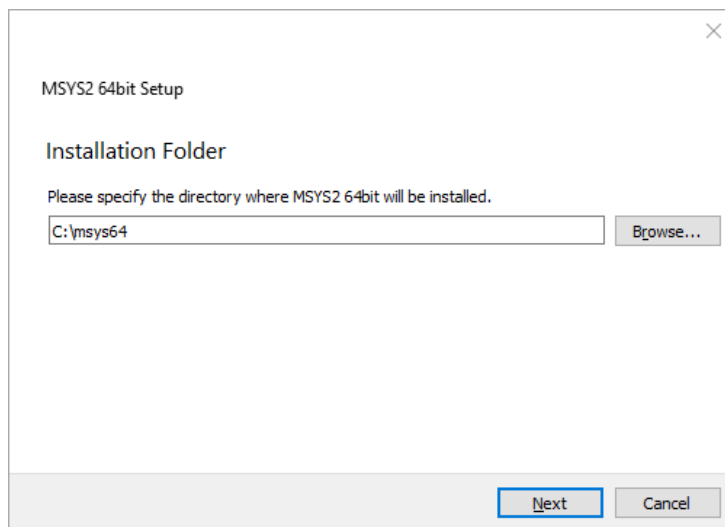
The installation can be started by executing the executable installation file, in this example `msys2-x86_64-20180531.exe`, the following screen should appear:





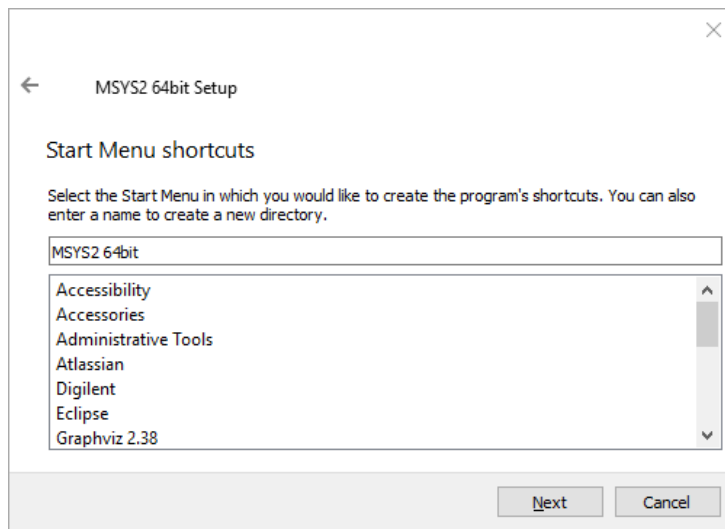
**Figure 37** – MSYS2 installation welcome prompt

Click Next.



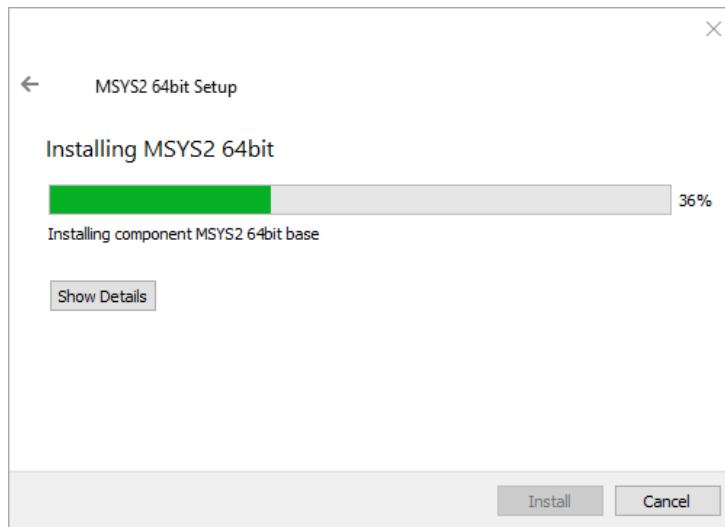
**Figure 38** – MSYS2 installation directory prompt

Enter the desired installation directory, the installation directory should be short with no spaces in the path. This guide will assume that the default installation directory of C : /msys64 is used. Click Next.



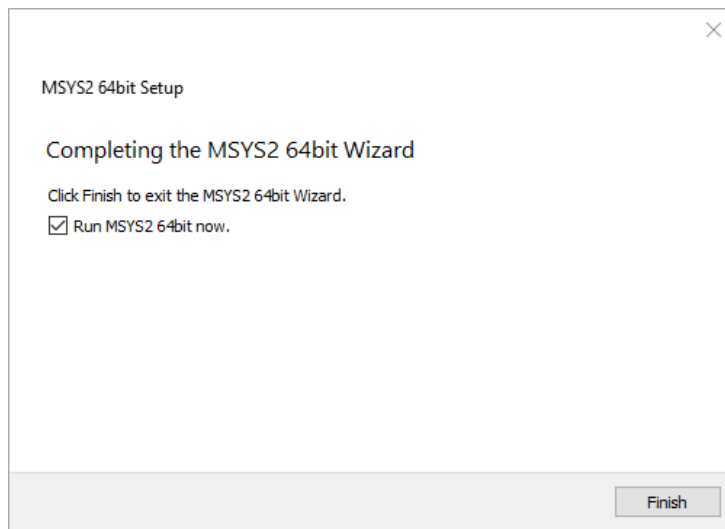
**Figure 39** – MSYS2 installation start menu shortcut prompt

The installer will now prompt for the name of the start menu shortcut. It is recommended to use the default name. Click Next.



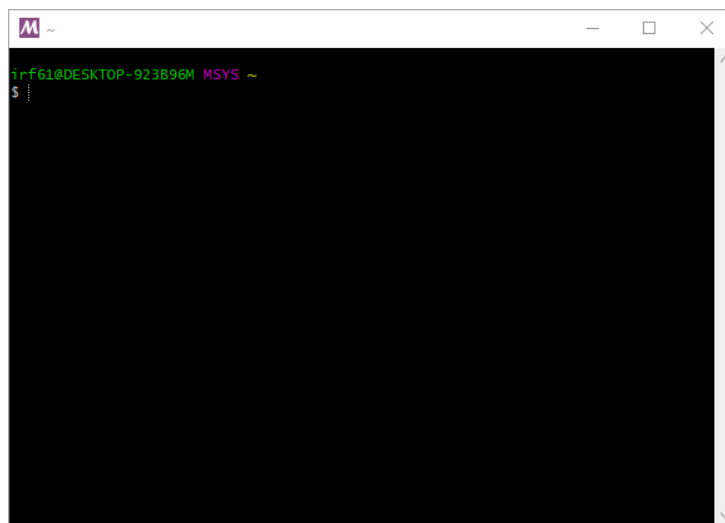
**Figure 40** – MSYS2 installation progress screen

Installation should begin.



**Figure 41** – MSYS2 installation finished screen

The installation should finish successfully. Make sure that "Run MSYS2 64 bit now." is checked.

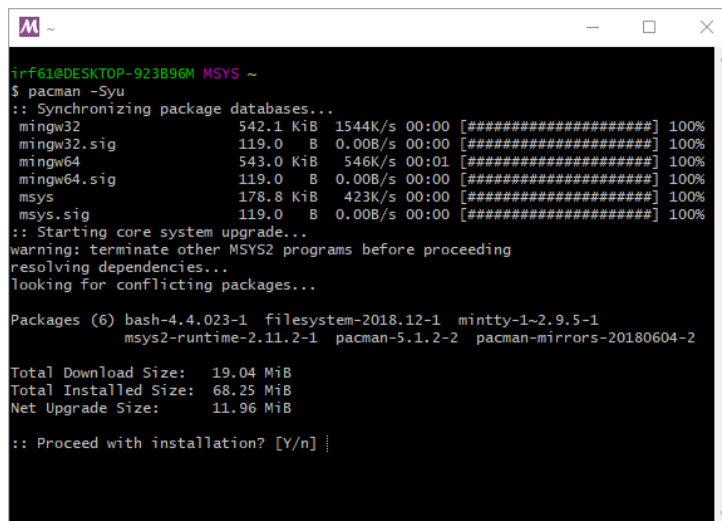


**Figure 42** – MSYS2 console

The MSYS2 console should open. It is now necessary to perform an update of the MSYS package manager by typing:

```
pacman -Syu
```

And press enter.



```
irrf61@DESKTOP-923B96M MSYS ~
$ pacman -Syu
:: Synchronizing package databases...
mingw32                542.1 KiB  1544K/s  00:00 [#####] 100%
mingw32.sig             119.0   B   0.00B/s  00:00 [#####] 100%
mingw64                543.0 KiB   546K/s  00:01 [#####] 100%
mingw64.sig             119.0   B   0.00B/s  00:00 [#####] 100%
msys                   178.8 KiB   423K/s  00:00 [#####] 100%
msys.sig                119.0   B   0.00B/s  00:00 [#####] 100%
:: Starting core system upgrade...
warning: terminate other MSYS2 programs before proceeding
resolving dependencies...
looking for conflicting packages...

Packages (6) bash-4.4.023-1  filesystem-2018.12-1  mintty-1~2.9.5-1
               msys2-runtime-2.11.2-1  pacman-5.1.2-2  pacman-mirrors-20180604-2

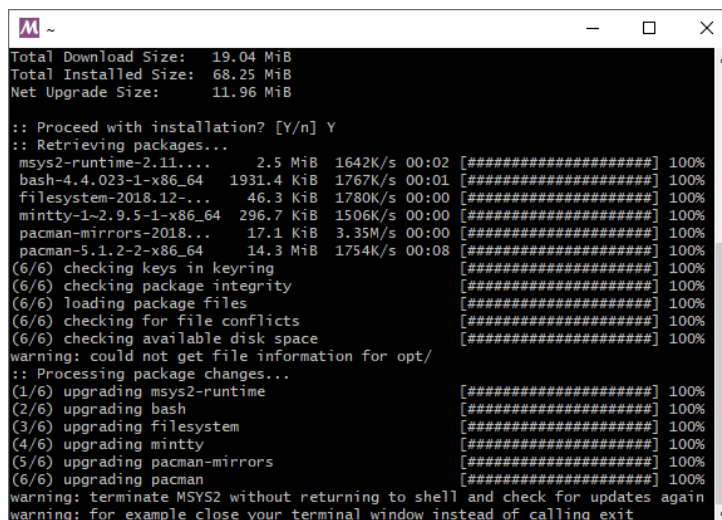
Total Download Size:  19.04 MiB
Total Installed Size: 68.25 MiB
Net Upgrade Size:     11.96 MiB

:: Proceed with installation? [Y/n]
```

Figure 43 – MSYS2 package manager update prompt

After downloading the update information, a prompt will appear to ask for permission to proceed with the installation. Type Y and press enter to continue.

Download and installation will proceed, this can take several minutes depending on network connection and computer performance.



```
Total Download Size:  19.04 MiB
Total Installed Size: 68.25 MiB
Net Upgrade Size:     11.96 MiB

:: Proceed with installation? [Y/n] Y
:: Retrieving packages...
msys2-runtime-2.11...  2.5 MiB  1642K/s  00:02 [#####] 100%
bash-4.4.023-1-x86_64 1931.4 KiB 1767K/s 00:01 [#####] 100%
filesystem-2018.12-...  46.3 KiB 1780K/s 00:00 [#####] 100%
mintty-1~2.9.5-1-x86_64 296.7 KiB 1506K/s 00:00 [#####] 100%
pacman-mirrors-2018... 17.1 KiB  3.35M/s 00:00 [#####] 100%
pacman-5.1.2-2-x86_64 14.3 MiB 1754K/s 00:08 [#####] 100%
(6/6) checking keys in keyring [#####] 100%
(6/6) checking package integrity [#####] 100%
(6/6) loading package files [#####] 100%
(6/6) checking for file conflicts [#####] 100%
(6/6) checking available disk space [#####] 100%
warning: could not get file information for opt/
:: Processing package changes...
(1/6) upgrading msys2-runtime [#####] 100%
(2/6) upgrading bash [#####] 100%
(3/6) upgrading filesystem [#####] 100%
(4/6) upgrading mintty [#####] 100%
(5/6) upgrading pacman-mirrors [#####] 100%
(6/6) upgrading pacman [#####] 100%
warning: terminate MSYS2 without returning to shell and check for updates again
warning: for example close your terminal window instead of calling exit
```

Figure 44 – MSYS2 package update restart request

It is likely that a warning message is displayed at this point asking to return to the shell and restart the update process. When this occurs exit the MSYS2 console by pressing the X button on the top right corner of the window. It is important to quit the console that way otherwise the warning may reappear in the next invocation.

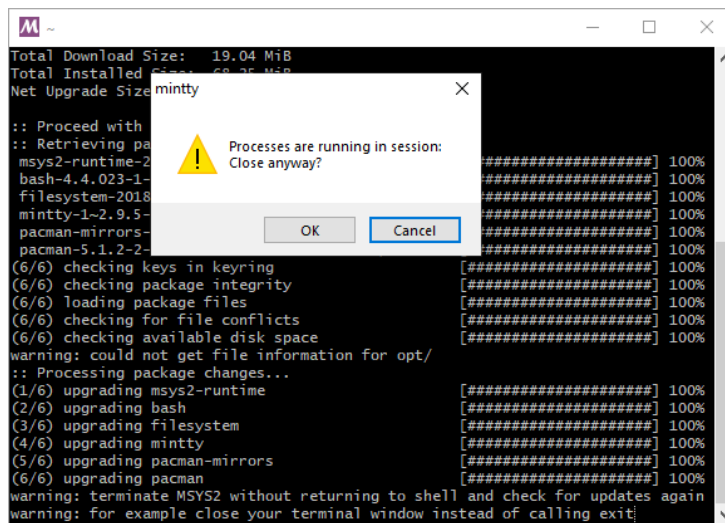


Figure 45 – MSYS2 window close confirmation

A warning message will be displayed, press OK to proceed with the close.

Open the MSYS2 console again by navigating to the start menu item for MSYS2 and clicking MSYS2 MSYS.

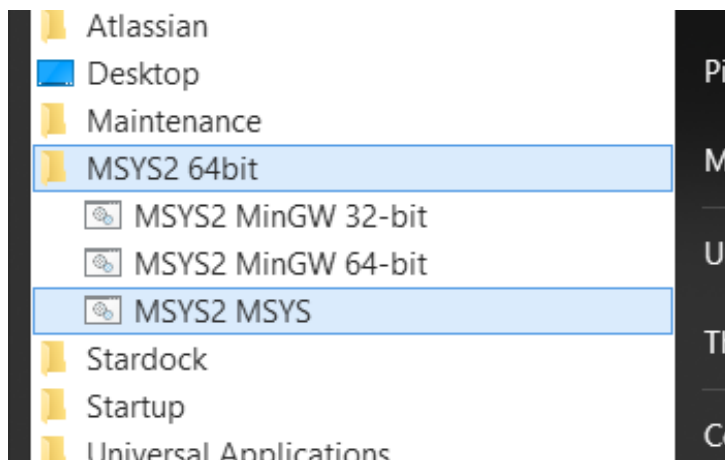
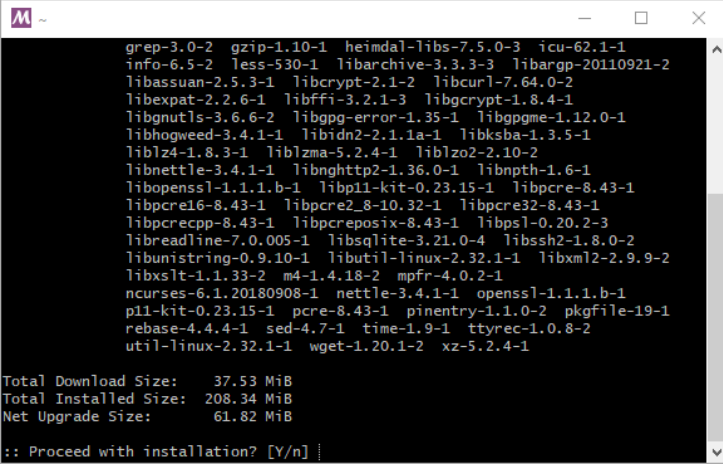


Figure 46 – MSYS2 start menu entry

Once opened type:

`pacman -Su`

And press enter to continue the update process.



```

grep-3.0-2  gzip-1.10-1  heimdal-libs-7.5.0-3  icu-62.1-1
info-6.5-2  less-530-1  libarchive-3.3.3-3  libargp-20110921-2
libassuan-2.5.3-1  libcrypto-2.1-2  libcurl-7.64.0-2
libexpat-2.2.6-1  libffi-3.2.1-3  libgpg-error-1.8.4-1
libgpgme-1.12.0-1  libhogweed-3.4.1-1  libidn2-2.1.1a-1  libksba-1.3.5-1
libl24-1.8.3-1  libl2ma-5.2.4-1  libl2o-2.10-2
libnettle-3.4.1-1  libnhttp2-1.36.0-1  libnph-1.6-1
libopenssl-1.1.1.b-1  libp11-kit-0.23.15-1  libpcre-8.43-1
libpcre16-8.43-1  libpcre2_8-10.32-1  libpcre32-8.43-1
libpcrecpp-8.43-1  libpcreposix-8.43-1  libpsl-0.20.2-3
libreadline-7.0.005-1  libsqlite-3.21.0-4  libssh2-1.8.0-2
libunistring-0.9.10-1  libutil-linux-2.32.1-1  libxml2-2.9.9-2
libxslt-1.1.33-2  m4-1.4.18-2  mpfr-4.0.2-1
ncurses-6.1.20180908-1  nettle-3.4.1-1  openssl-1.1.1.b-1
p11-kit-0.23.15-1  pcre-8.43-1  pinentry-1.1.0-2  pkgfile-19-1
rebase-4.4.4-1  sed-4.7-1  time-1.9-1  ttyrec-1.0.8-2
util-linux-2.32.1-1  wget-1.20.1-2  xz-5.2.4-1

Total Download Size: 37.53 MiB
Total Installed Size: 208.34 MiB
Net Upgrade Size: 61.82 MiB

:: Proceed with installation? [Y/n]

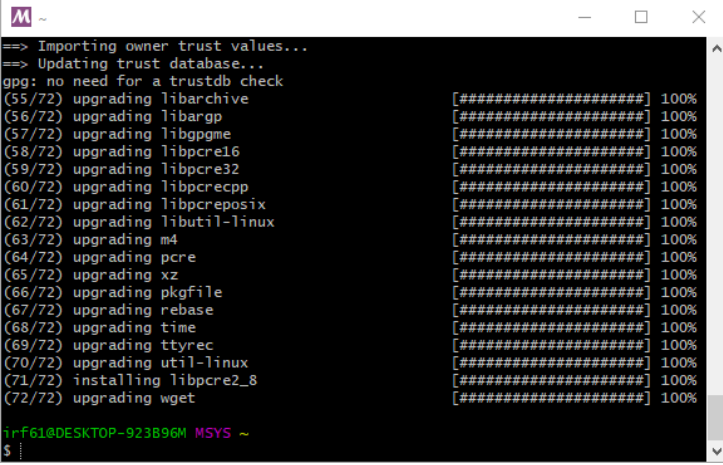
```

Figure 47 – MSYS2 package manager update prompt

A prompt asking for permission to continue will appear again. Type Y and press enter to continue.

Download and installation will proceed, like the previous step this can take several minutes depending on network and computer performance.

Installation should complete without any error or warning messages.



```

==> Importing owner trust values...
==> Updating trust database...
gpg: no need for a trustdb check
(55/72) upgrading libarchive [#####] 100%
(56/72) upgrading libargp [#####] 100%
(57/72) upgrading libgpgme [#####] 100%
(58/72) upgrading libpcre16 [#####] 100%
(59/72) upgrading libpcre32 [#####] 100%
(60/72) upgrading libpcrecpp [#####] 100%
(61/72) upgrading libpcreposix [#####] 100%
(62/72) upgrading libutil-linux [#####] 100%
(63/72) upgrading m4 [#####] 100%
(64/72) upgrading pcre [#####] 100%
(65/72) upgrading xz [#####] 100%
(66/72) upgrading pkgfile [#####] 100%
(67/72) upgrading rebase [#####] 100%
(68/72) upgrading time [#####] 100%
(69/72) upgrading ttyrec [#####] 100%
(70/72) upgrading util-linux [#####] 100%
(71/72) installing libpcre2_8 [#####] 100%
(72/72) upgrading wget [#####] 100%

irff61@DESKTOP-923B96M MSYS ~
$

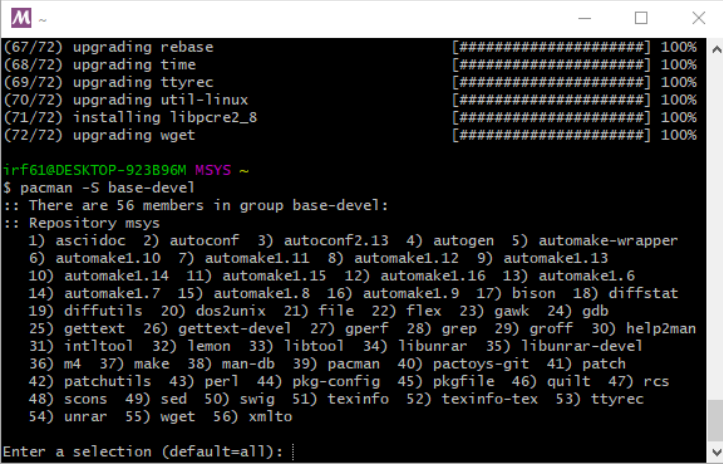
```

Figure 48 – MSYS2 package manager update done

With the package manager updated and synced it is time to install the basic development utilities. This can be achieved by typing:

```
pacman -S base-devel
```

A list of packages to be installed will appear.



```

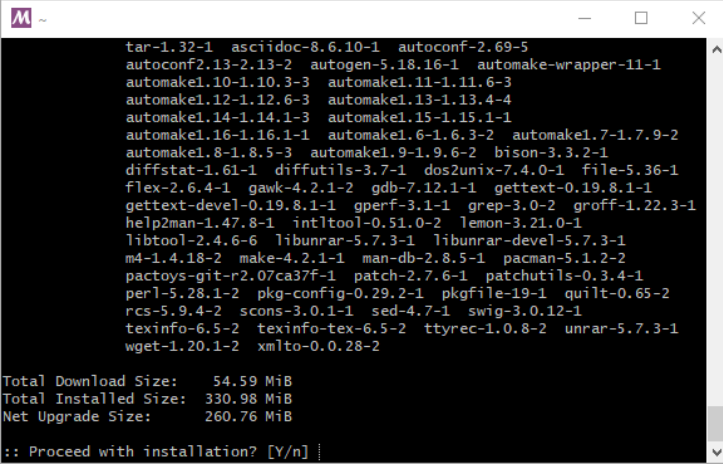
(67/72) upgrading rebase [#####] 100%
(68/72) upgrading time [#####] 100%
(69/72) upgrading ttyrec [#####] 100%
(70/72) upgrading util-linux [#####] 100%
(71/72) installing libpcr2_8 [#####] 100%
(72/72) upgrading wget [#####] 100%

irrf61@DESKTOP-923B96M MSYS ~
$ pacman -S base-devel
:: There are 56 members in group base-devel:
:: Repository msys
 1) asciidoc 2) autoconf 3) autoconf2.13 4) autogen 5) automake-wrapper
 6) automake1.10 7) automake1.11 8) automake1.12 9) automake1.13
10) automake1.14 11) automake1.15 12) automake1.16 13) automake1.6
14) automake1.7 15) automake1.8 16) automake1.9 17) bison 18) diffstat
19) diffutils 20) dos2unix 21) file 22) flex 23) gawk 24) gdb
25) gettext 26) gettext-devel 27) gperf 28) grep 29) groff 30) help2man
31) intltool 32) lemon 33) libtool 34) libunrar 35) libunrar-devel
36) m4 37) make 38) man-db 39) pacman 40) pautoys-git 41) patch
42) patchutils 43) perl 44) pkg-config 45) pkgfile 46) quilt 47) rcs
48) scons 49) sed 50) swig 51) texinfo 52) texinfo-tex 53) ttyrec
54) unrar 55) wget 56) xmlto
Enter a selection (default=all):

```

Figure 49 – MSYS2 base-level prompt

Press Enter to continue.



```

tar-1.32-1  asciidoc-8.6.10-1  autoconf-2.69-5
autoconf2.13-2.13-2  autogen-5.18.16-1  automake-wrapper-11-1
automake1.10-1.10.3-3  automake1.11-1.11.6-3
automake1.12-1.12.6-3  automake1.13-1.13.4-4
automake1.14-1.14.1-3  automake1.15-1.15.1-1
automake1.16-1.16.1-1  automake1.6-1.6.3-2  automake1.7-1.7.9-2
automake1.8-1.8.5-3  automake1.9-1.9.6-2  bison-3.3.2-1
diffstat-1.61-1  diffutils-3.7-1  dos2unix-7.4.0-1  file-5.36-1
flex-2.6.4-1  gawk-4.2.1-2  gdb-7.12.1-1  gettext-0.19.8.1-1
gettext-devel-0.19.8.1-1  gperf-3.1-1  grep-3.0-2  groff-1.22.3-1
help2man-1.47.8-1  intltool-0.51.0-2  lemon-3.21.0-1
libtool-2.4.6-6  libunrar-5.7.3-1  libunrar-devel-5.7.3-1
m4-1.4.18-2  make-4.2.1-1  man-db-2.8.5-1  pacman-5.1.2-2
pautoys-git-r2.07ca37f-1  patch-2.7.6-1  patchutils-0.3.4-1
perl-5.28.1-2  pkg-config-0.29.2-1  pkgfile-19-1  quilt-0.65-2
rcs-5.9.4-2  scons-3.0.1-1  sed-4.7-1  swig-3.0.12-1
texinfo-6.5-2  texinfo-tex-6.5-2  ttyrec-1.0.8-2  unrar-5.7.3-1
wget-1.20.1-2  xmlto-0.0.28-2

Total Download Size: 54.59 MiB
Total Installed Size: 330.98 MiB
Net Upgrade Size: 260.76 MiB

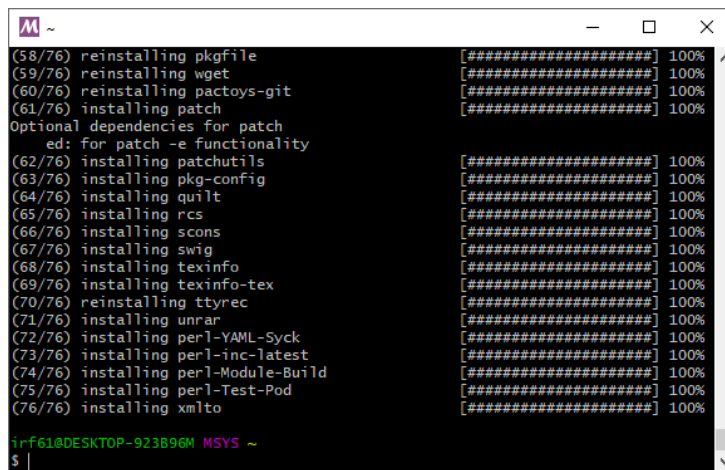
:: Proceed with installation? [Y/n]

```

Figure 50 – MSYS2 base-level installation confirmation prompt

Type Y and then Enter to continue.

Download and installation will proceed, this can take several minutes depending on network and computer performance.



```
~
(58/76) reinstalling pkgfile [#####] 100%
(59/76) reinstalling wget [#####] 100%
(60/76) reinstalling pactois-git [#####] 100%
(61/76) installing patch [#####] 100%
Optional dependencies for patch
ed: for patch -e functionality
(62/76) installing patchutils [#####] 100%
(63/76) installing pkg-config [#####] 100%
(64/76) installing quilt [#####] 100%
(65/76) installing rcs [#####] 100%
(66/76) installing scons [#####] 100%
(67/76) installing swig [#####] 100%
(68/76) installing texinfo [#####] 100%
(69/76) installing texinfo-tex [#####] 100%
(70/76) reinstalling ttyrec [#####] 100%
(71/76) installing unrar [#####] 100%
(72/76) installing perl-YAML-Syck [#####] 100%
(73/76) installing perl-inc-latest [#####] 100%
(74/76) installing perl-Module-Build [#####] 100%
(75/76) installing perl-Test-Pod [#####] 100%
(76/76) installing xmlto [#####] 100%

irf61@DESKTOP-923B96M MSYS ~
$ |
```

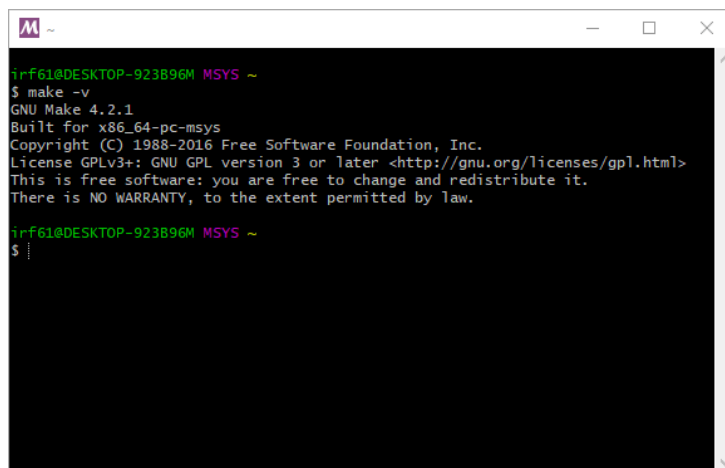
Figure 51 – MSYS2 base-level installation completed

The installation should complete without any error or warning messages.

To test that the installation is functional, let's run a simple test by invoking the make utility as such:

```
make -v
```

The make utility should output its version information in the console.



```
~
irf61@DESKTOP-923B96M MSYS ~
$ make -v
GNU Make 4.2.1
Built for x86_64-pc-msys
Copyright (C) 1988-2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

irf61@DESKTOP-923B96M MSYS ~
$ |
```

Figure 52 – MSYS2 make test output

MSYS2 and the required utilities are now installed, the console window can be closed.

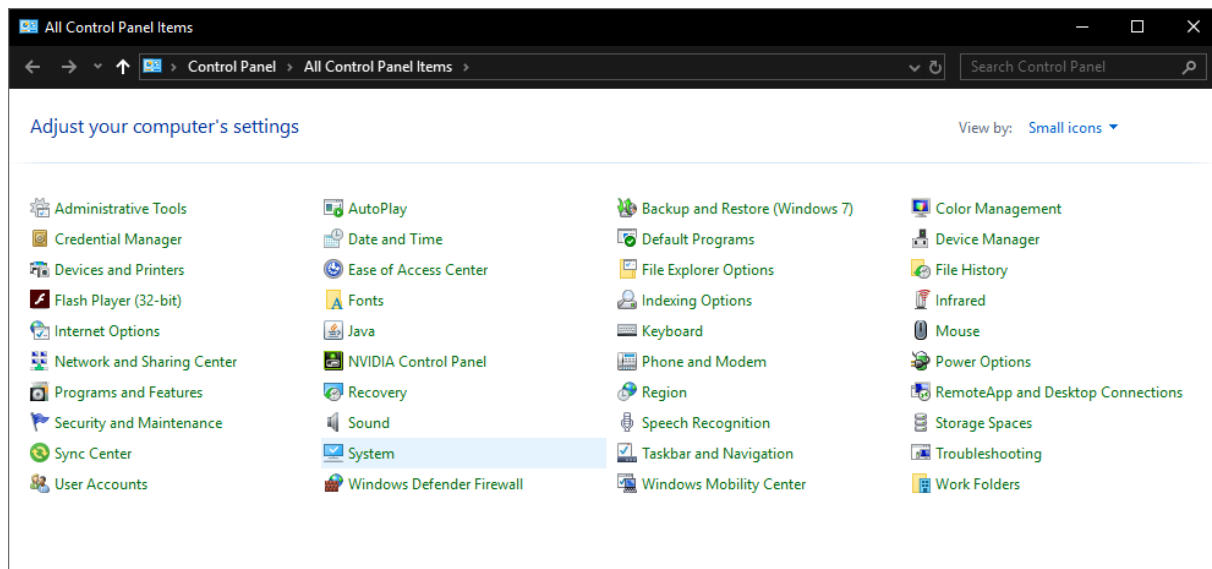


## 2.7 Environment Variable Setup

For the MSYS2 executables to be easily accessible by Eclipse, Make and other utilities it is preferable to have the toolchain binary location in the PATH environment variable. Note that these steps are not required if MSYS wasn't installed.

### 2.7.1 PATH Environment Variable Setup

Start by opening the Windows Control Panel.



**Figure 53** – Windows Control Panel

Click "System".

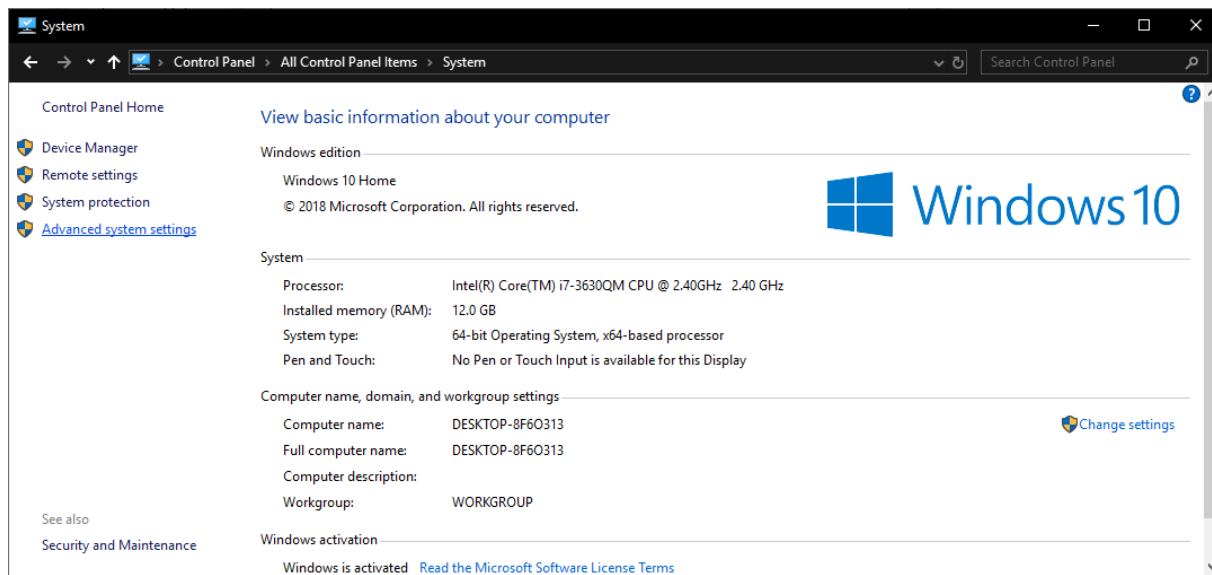


Figure 54 – Windows system settings

Click "Advanced System Settings".

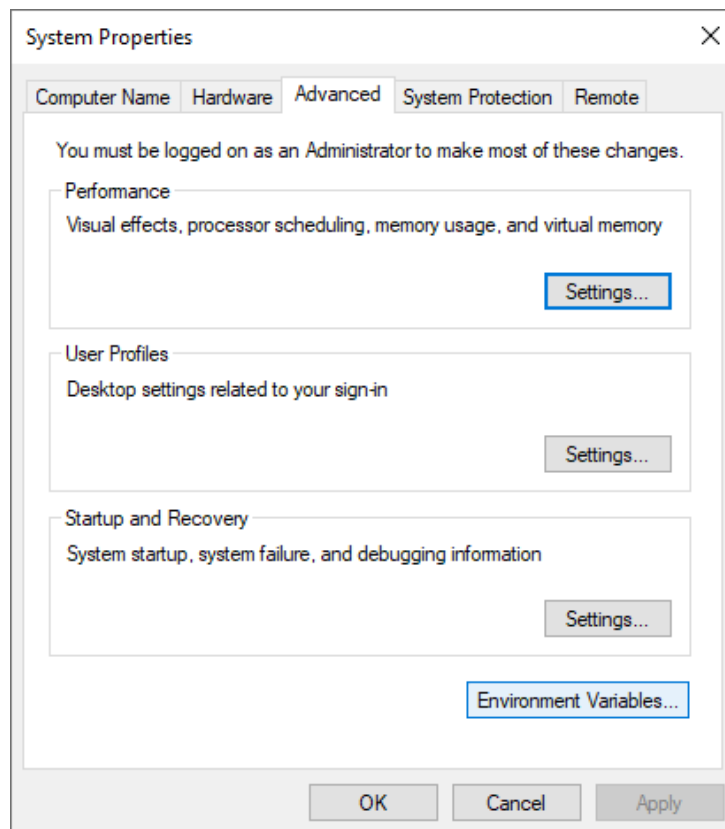
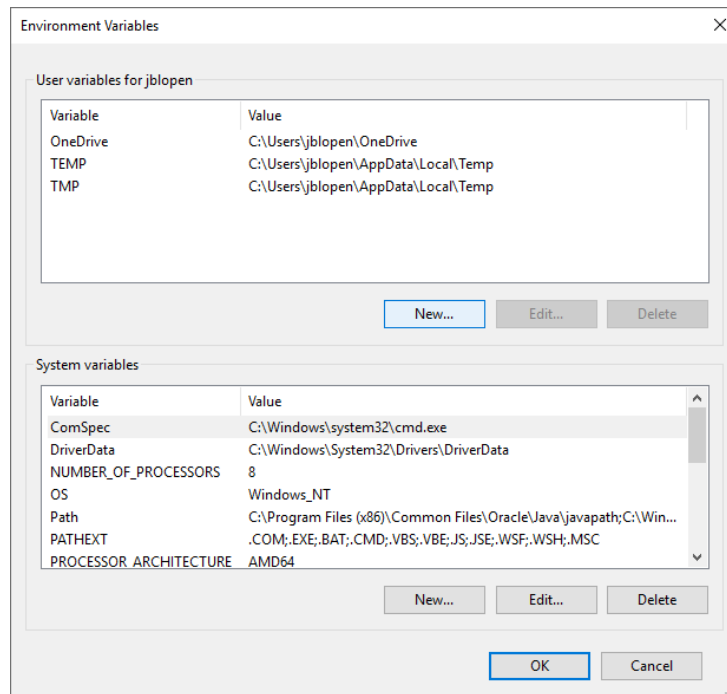


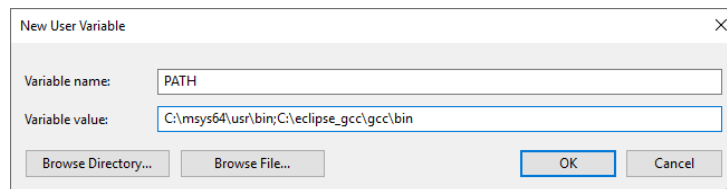
Figure 55 – Windows Advanced System Settings

Click "Environment Variables".



**Figure 56** – Windows environment variable configuration panel

In this screen, if an existing PATH environment variable exists, click it and then click "Edit..." otherwise click "New..."

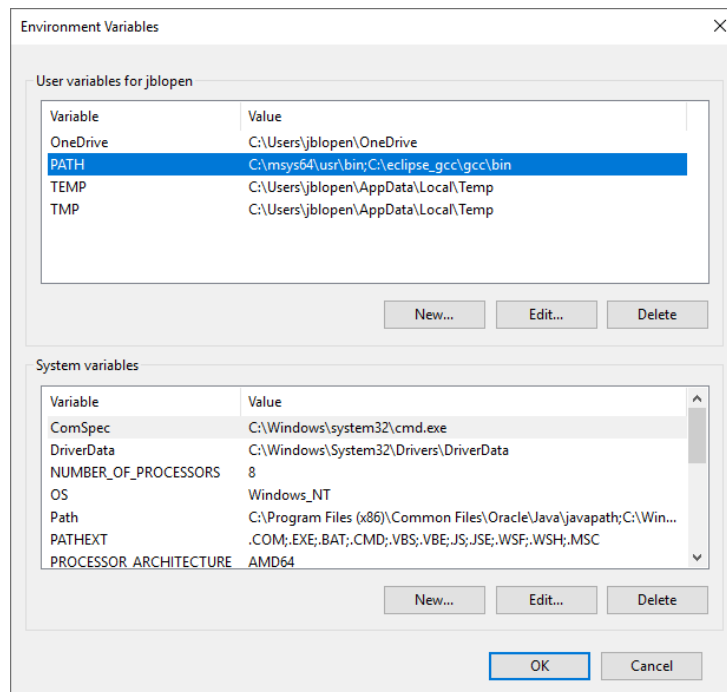


**Figure 57** – Windows new environment variable dialog

The path should be added either to the existing PATH variable or a new one. The `usr/bin` subdirectory of the MSYS2 installation should be used. For the install directory chosen for this example, this results in the following path:

```
C:/msys64/usr/bin;
```

Then click OK.



**Figure 58** – Configured environment variables

Then OK again to exit the Environment Variables window.

---

## Managed Build Project Setup

Broadly speaking there are two types of projects that can be created with IAR Eclipse. The first is a managed build project where the IDE is responsible for the various build steps. The second is a Makefile project where the user is responsible for providing a suitable Makefile to perform the building and linking of the project source code. The latter is discussed in a later section of this manual while this section will go over the steps necessary to create a managed build project. The project created in this section will be used afterward to demonstrate debugging a managed build project. Note that these instructions are generic and the steps required may be slightly different depending on the target platforms.

### 3.1 IAR Eclipse Managed Build Project Setup

After having opened Eclipse with a suitable workspace as discussed in the previous sections, creating a new managed build project can be accomplished first by invoking the "File->New->C/C++ Project" menu item.

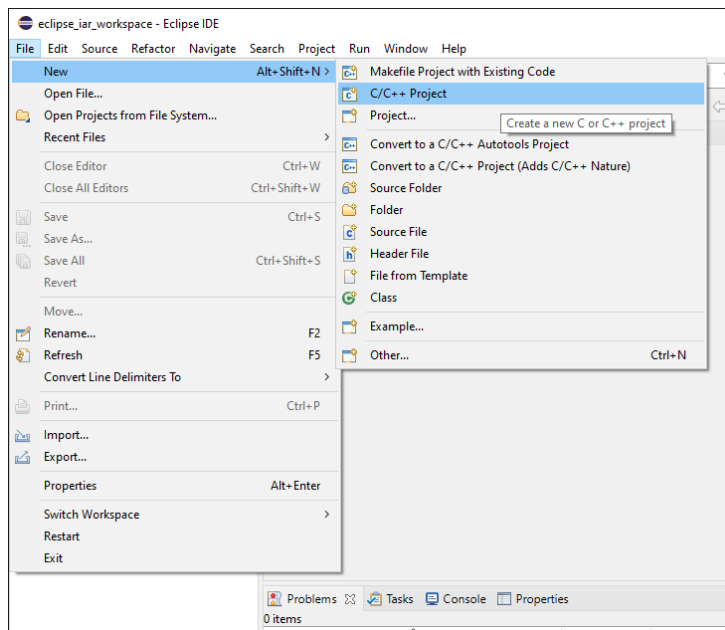


Figure 59 – Create New C/C++ Project menu item

This should open the "New C/C++ Project" dialog.

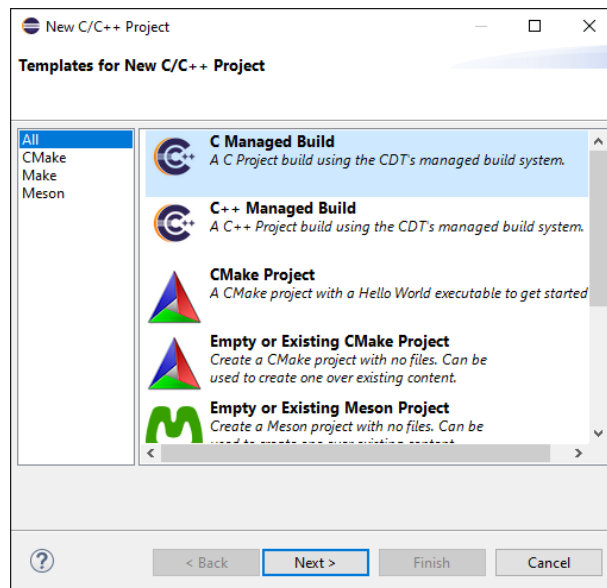
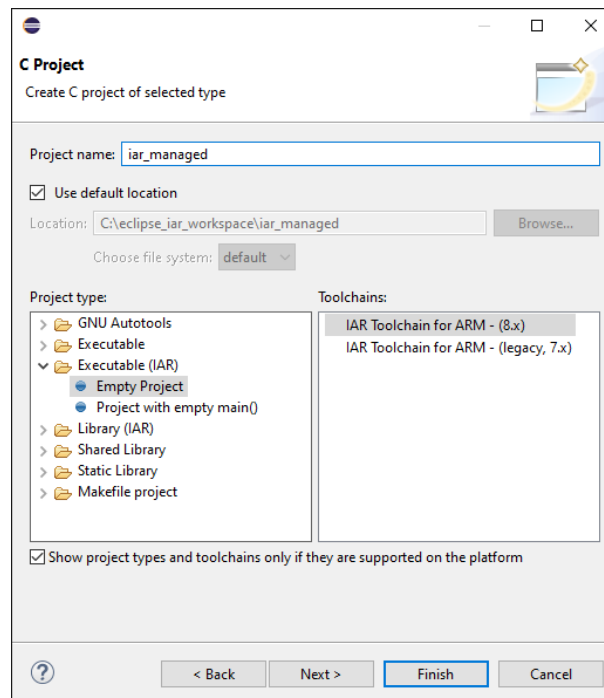


Figure 60 – New C/C++ Project dialog

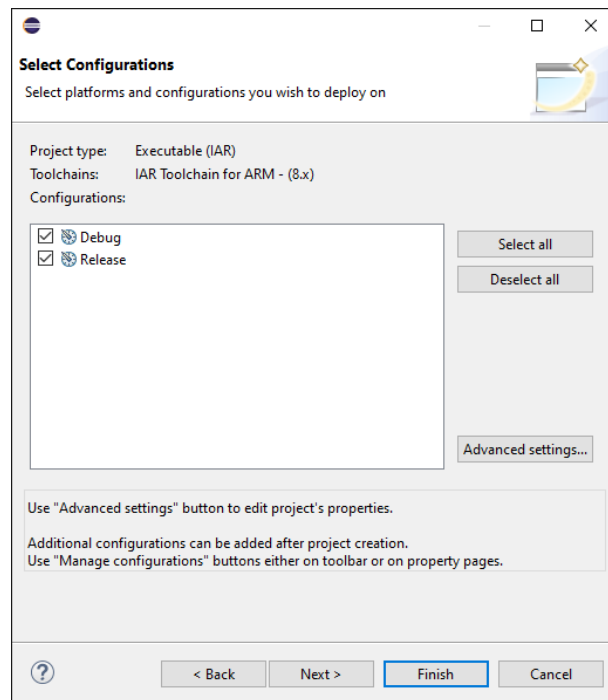
Select "C Managed Build" and then click the "Next" button.



**Figure 61** – New project name and type selection

In this next panel, the project name and type must be selected. First enter the desired project name. Then select the "Empty Project" item from under the "Executable (IAR)" category. Next, make sure that the "IAR Toolchain for ARM - (8.x)" option is selected unless you are using an older, legacy version of the IAR toolchain.

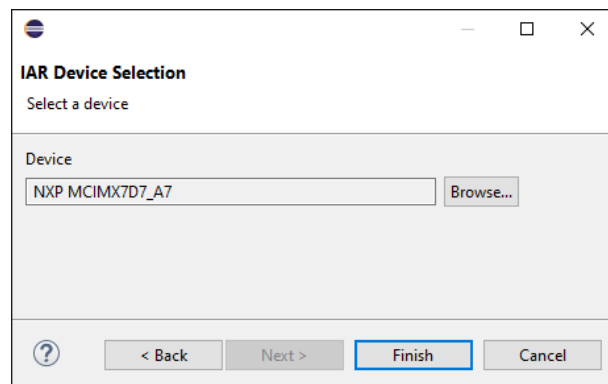
Click "Next" to continue with the wizard. Note that it is possible to exit the wizard by clicking "Finish" but this will skip the device selection step.



**Figure 62** – New project build configurations

This next panel shows the build configurations for the project to be created. For the purpose of this guide, it is recommended to keep the default values.

Click "Next" yet again to continue further to the device selection dialog.



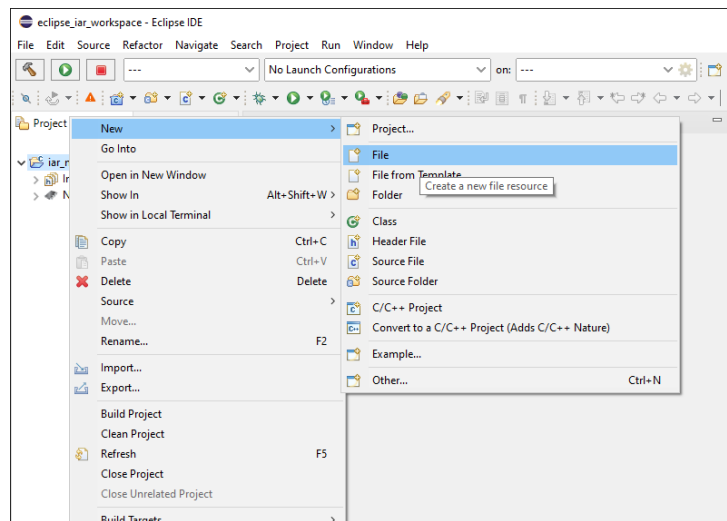
**Figure 63** – IAR Device Selection dialog

In the device selection dialog select the target device for the new project. This example uses the A7 core of the NXP i.MX7 SoC but any of the supported targets can be used.

Click "Finish" to exit the wizard and create the project. The newly created project should appear in the workspace on the left panel of the IDE.

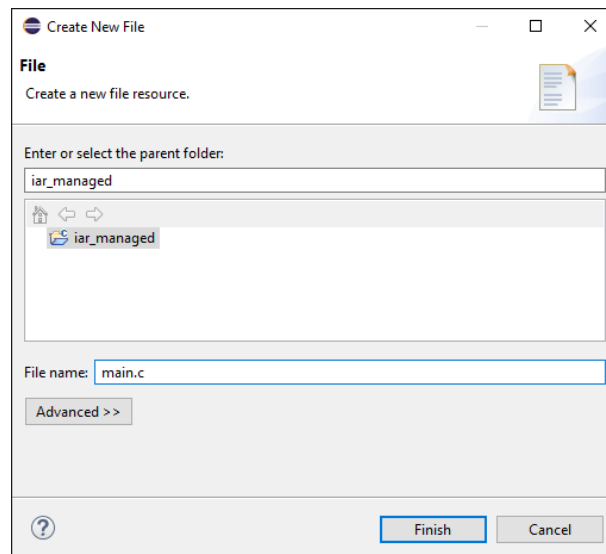


To demonstrate building and debugging this example uses an empty main function.



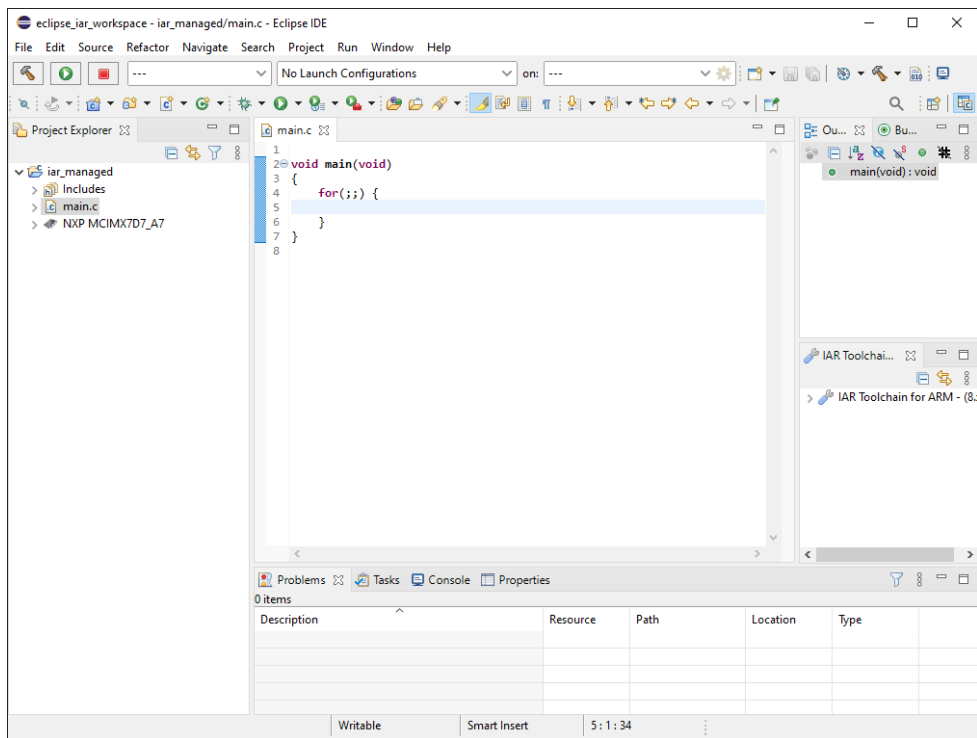
**Figure 64** – Add new file to project context menu item

To add the example file to the project right-click on the project root in the workspace and select "New->File". This opens the new file dialog.



**Figure 65** – Create new file dialog

Name the new file `main.c` and click "Finish".



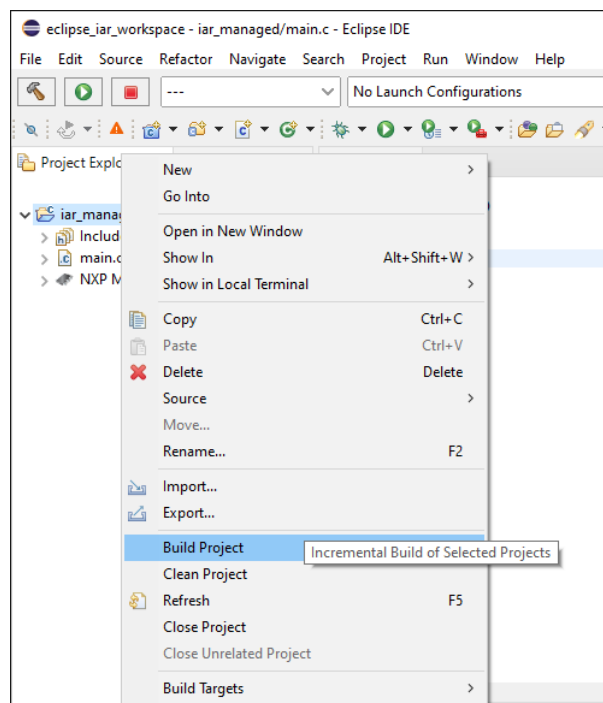
**Figure 66** – Empty main function

The new file is now showing in the project and should be open in the IDE. Type the following empty function into the newly created file before continuing.

```
void main(void)
{
    for(;;) {

    }
}
```

**Listing 1** – Example main function with infinite loop



**Figure 67** – Build project context menu item

Now right-click again on the root of the project and select "Build Project". This should start building the project. Note that you might get errors if a valid IAR licence isn't available for the IAR toolchain. See the previous sections on how to request an evaluation licence.

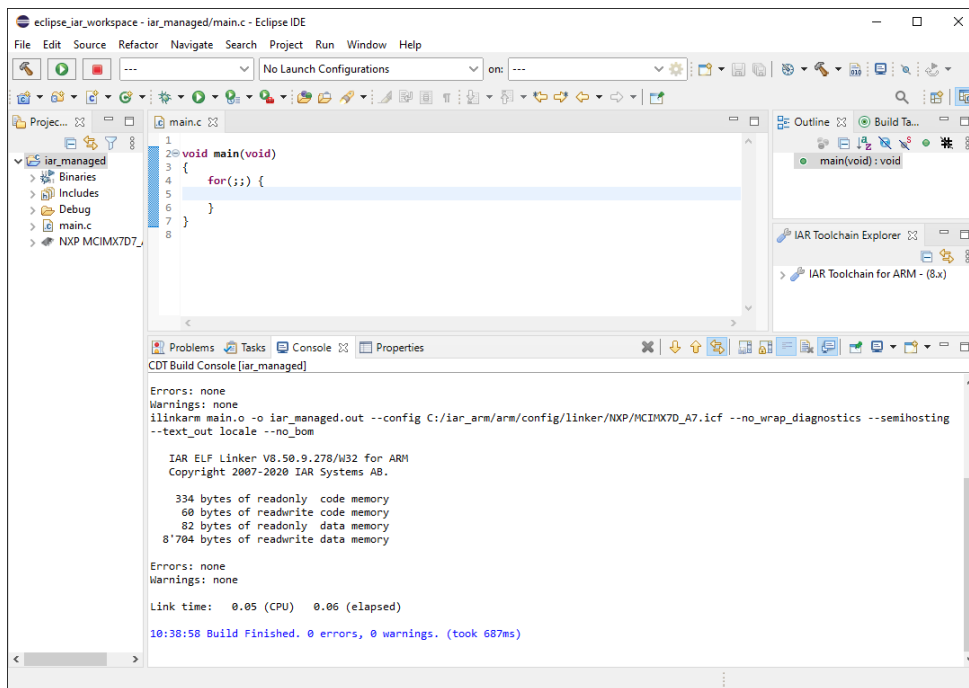


Figure 68 – Example project built

Before long the project should be built and is now ready to be loaded to a target for debugging. The next section will cover debugging a managed build project.

## 3.2 IAR Eclipse Managed Build Project Debugging

With a built project, it is now time to setup a debug configuration for the project and launch a debug session.

Start by searching for and clicking the "Run->Debug Configurations" menu. Not that while it is possible to create a debug configuration by right-clicking the project root and selecting "Debug as," this will launch a debug session without having the chance to configure it first which is almost certainly not the desired outcome.

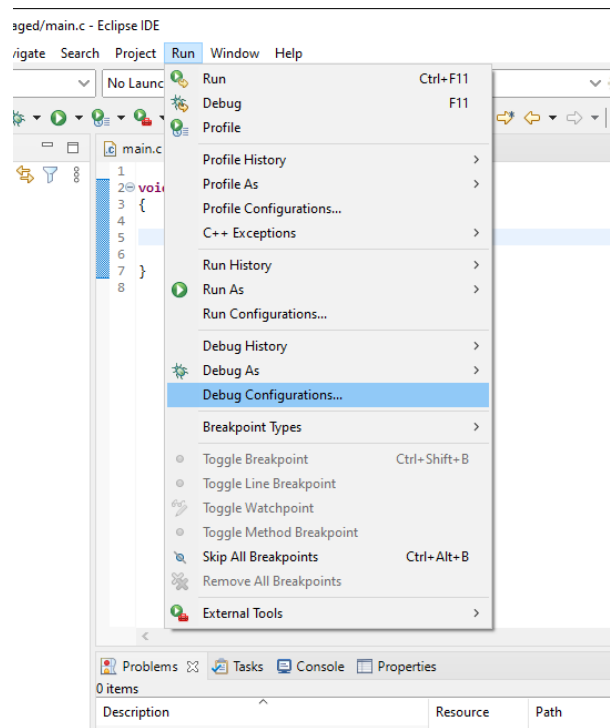


Figure 69 – Debug Configurations menu item

The Debug Configurations panel should appear.

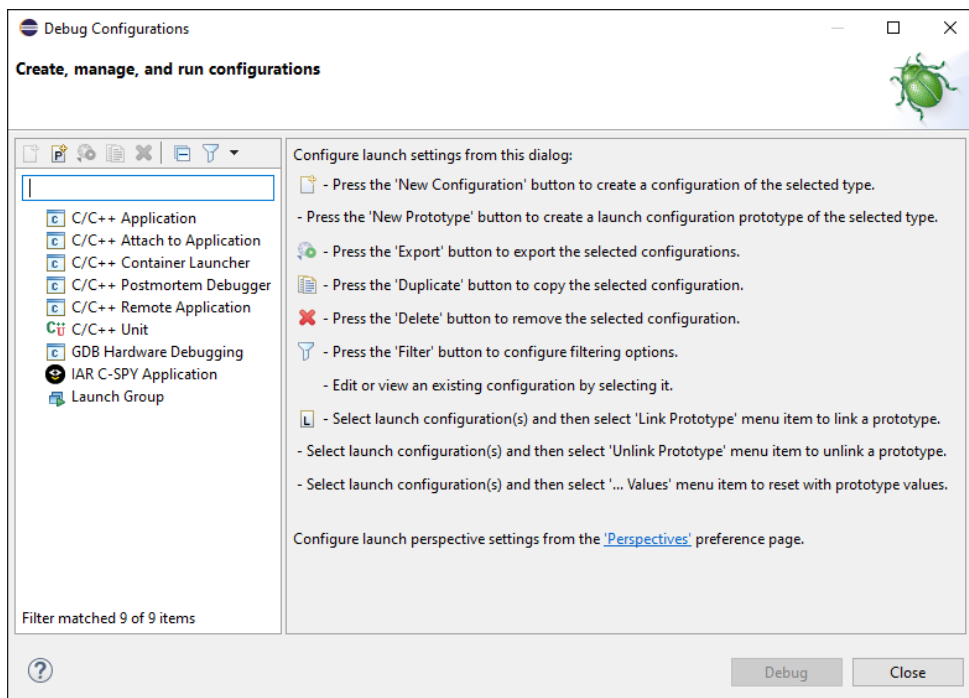
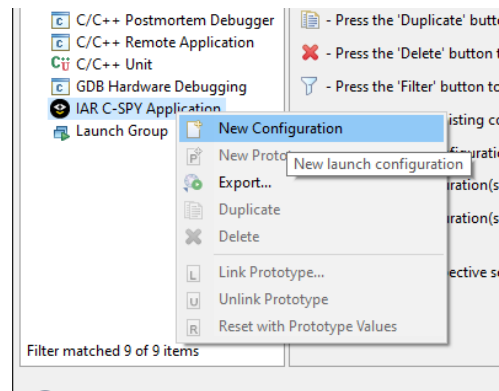


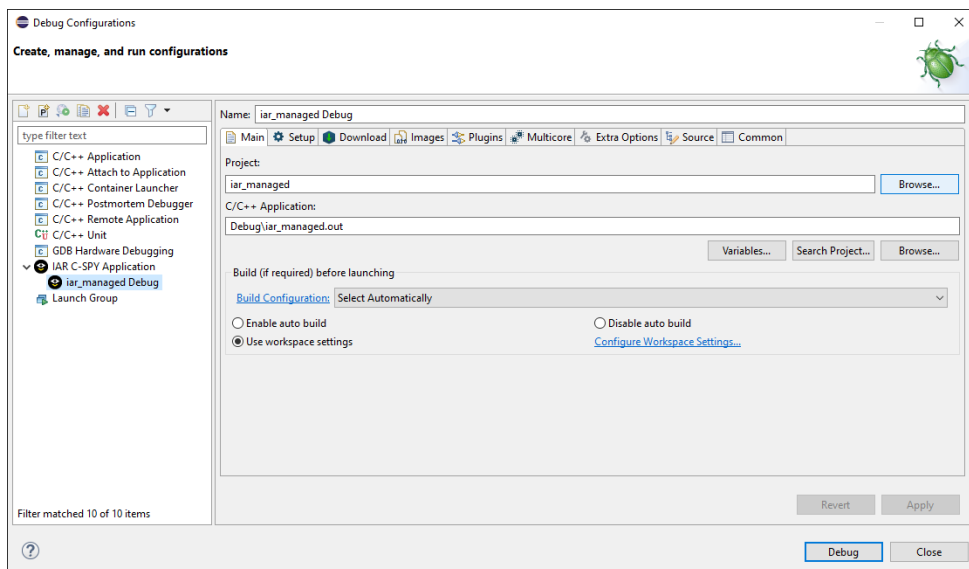
Figure 70 – Debug Configurations panel

On this panel right-click on "IAR C-SPY Application".



**Figure 71** – New Configuration context menu item

From the context menu select "New Configuration". This should create a new debug configuration and display it in the right pane.



**Figure 72** – IAR C-SPY debug configuration

By default the newly created configuration will be associated with the active project. In our example it is also the only project in the workspace. However, if for some reason the wrong project or no project is displayed in the "Project:" field click the "Browse" button to select the correct project from a list of projects available in the workspace.

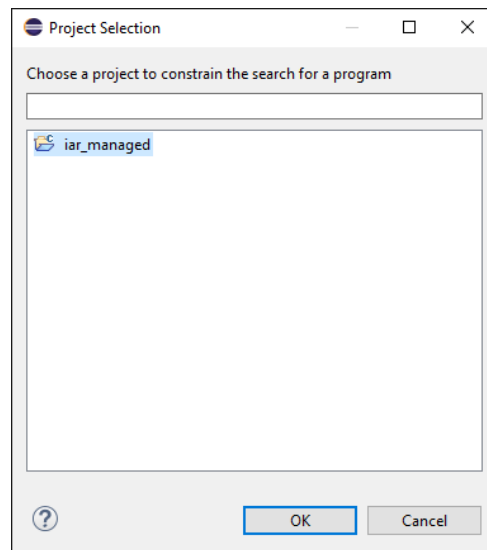


Figure 73 – IAR C-SPY debug project selection

Select the desired project for this debug configuration and click "OK" to return to the debug configuration panel.

Next let's look at the debug connection setup by selecting the "Setup" pane.

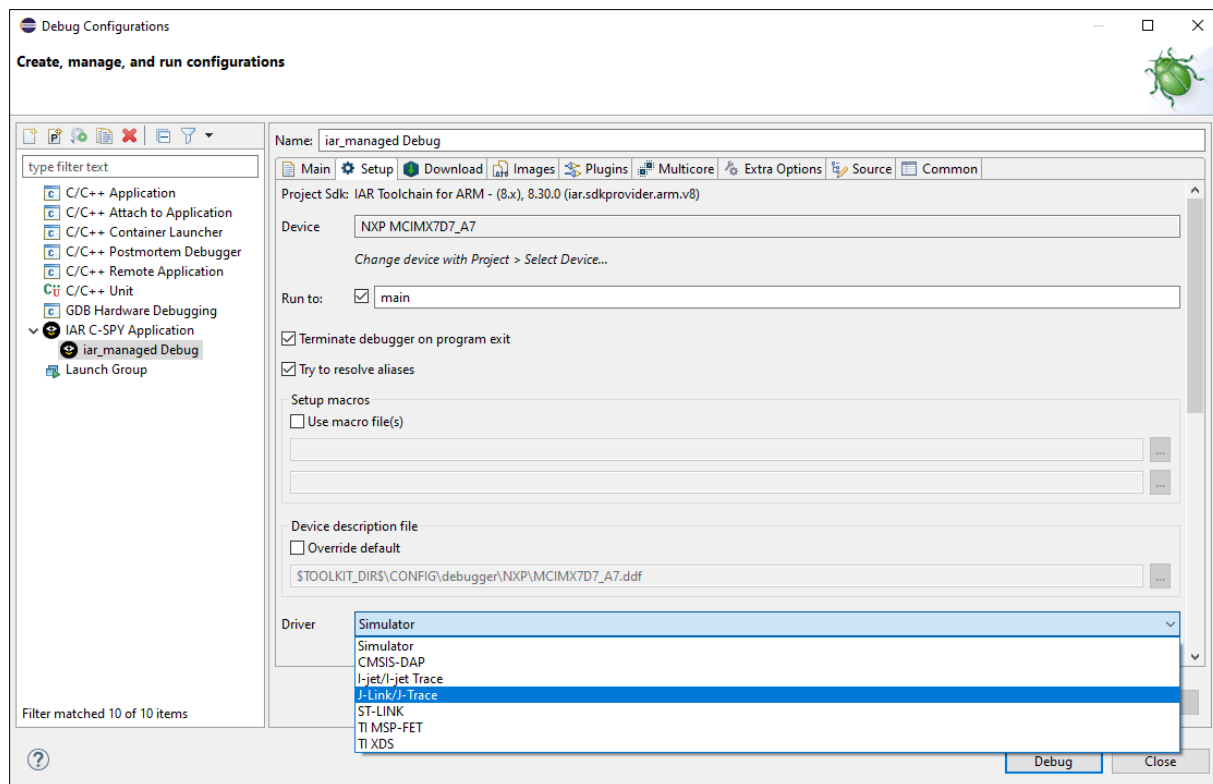


Figure 74 – IAR C-SPY debug setup

By default IAR will launch a debug session into the IAR Simulator. To use a real hardware target a suitable debug probe must be selected and configured. The exact configuration required for each target and debug probe combination is beyond the scope of this guide. As an example this guide will use the J-Link debug probe.

Once the proper debug probe is selected click "Debug" to launch the debug session.

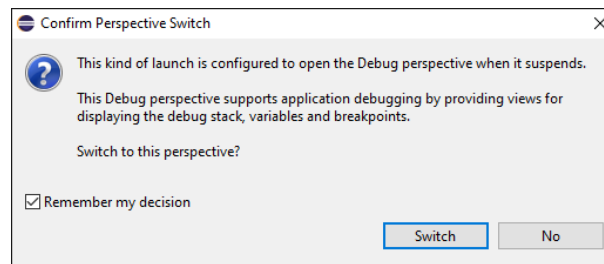


Figure 75 – Eclipse switch perspective prompt

Eclipse will probably warn that launching a debug session is associated with the "Debug" perspective. Clicking "Switch" will rearrange the selection of panels within Eclipse to show the views relevant to debugging. Checking "Remember my decision" will stop Eclipse from asking the same question each time a debug session is launched.

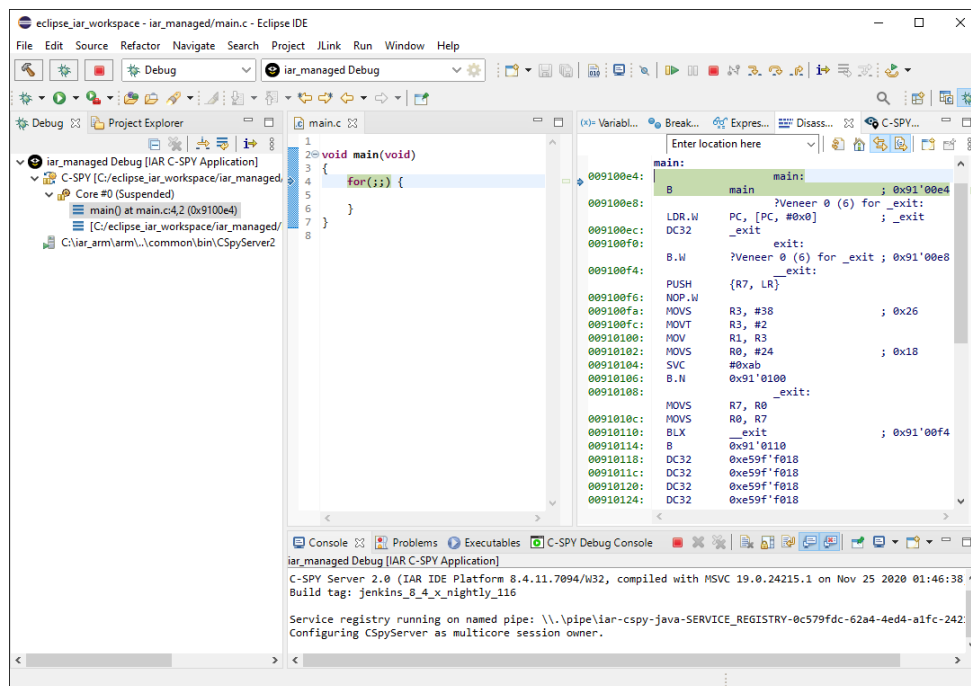


Figure 76 – IAR Eclipse C-SPY debug session

If all goes well, the debug session should now be active with the application halted at the main function created in the previous section. Pressing the stop button will terminate the current debug session. It is



also possible to return to the C/C++ perspective by clicking the icon next to the small green bug near the top right corner of the IDE.

---

# Makefile Project Setup

Contrary to a managed build project where the IDE is responsible for compiling the source code and linking the application binary, a Makefile project delegate those tasks to the user. It is possible to create a Makefile project using the IAR Eclipse plugin. However the steps are slightly different compared to the usual Eclipse Makefile project.

This section will focus on the steps that must be performed within Eclipse to get a Makefile project working. Additional information on how to write a Makefile for the IAR tools can be found on our website [here](#).

## 4.1 IAR Eclipse Makefile Project

The first step is to create a managed build project in the exact same way as was done in the previous section. Then we'll modify the project configuration to use a custom Makefile instead of auto-generating one.

Start by opening the new C/C++ Project dialog from the "File->New->C/C++ Project" menu item.

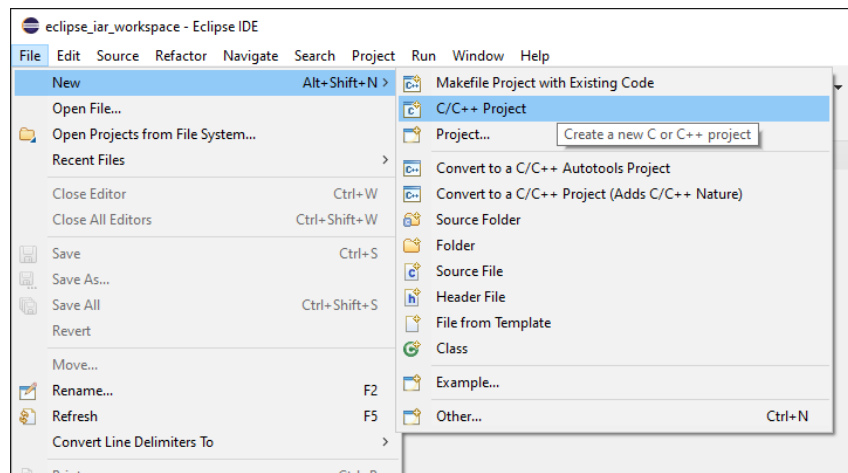


Figure 77 – New C/C++ project menu item

This should open the new project dialog.

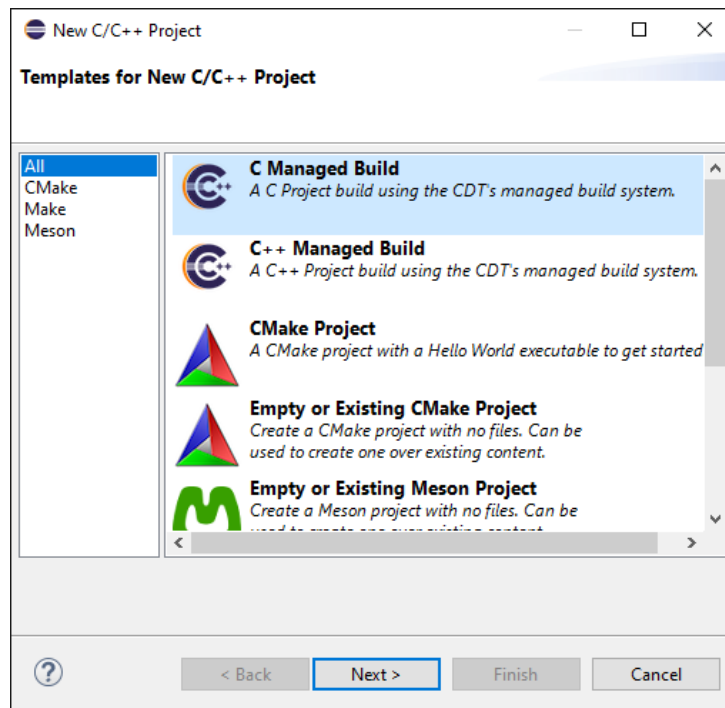
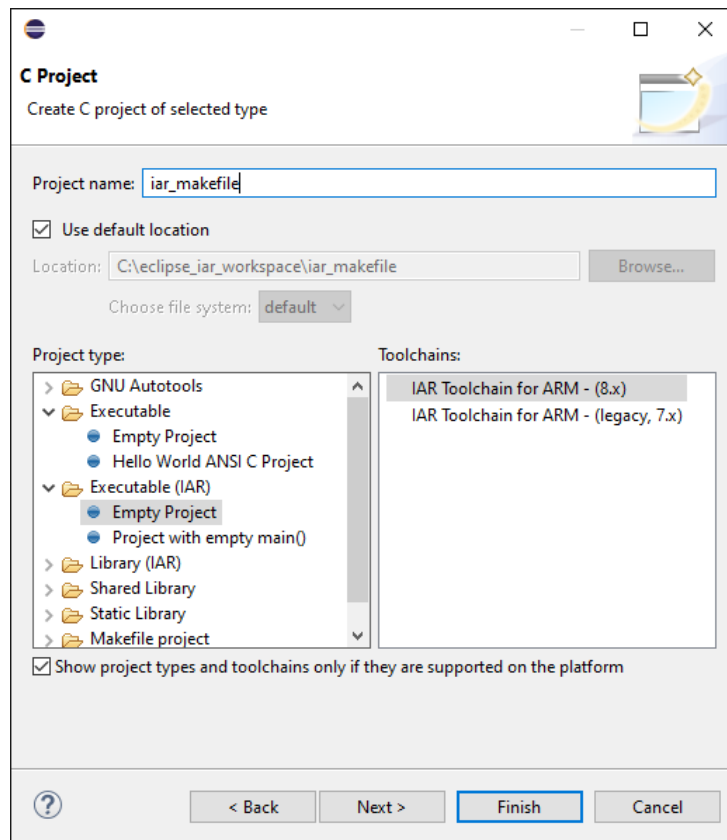


Figure 78 – New C/C++ project dialog

Select "C Managed Build" and click "Next" to go to the project type and name configuration.



**Figure 79** – New project name and type

Type a name for the new project, this example uses "iar\_makefile" as a project name.

In the project type panel (the left pane) expand the "Executable (IAR)" category and select "Empty Project". In the right pane make sure to select "IAR Toolchain for ARM - (8.x)".

Click "Next" to continue the wizard. The project build configurations panel should be displayed.

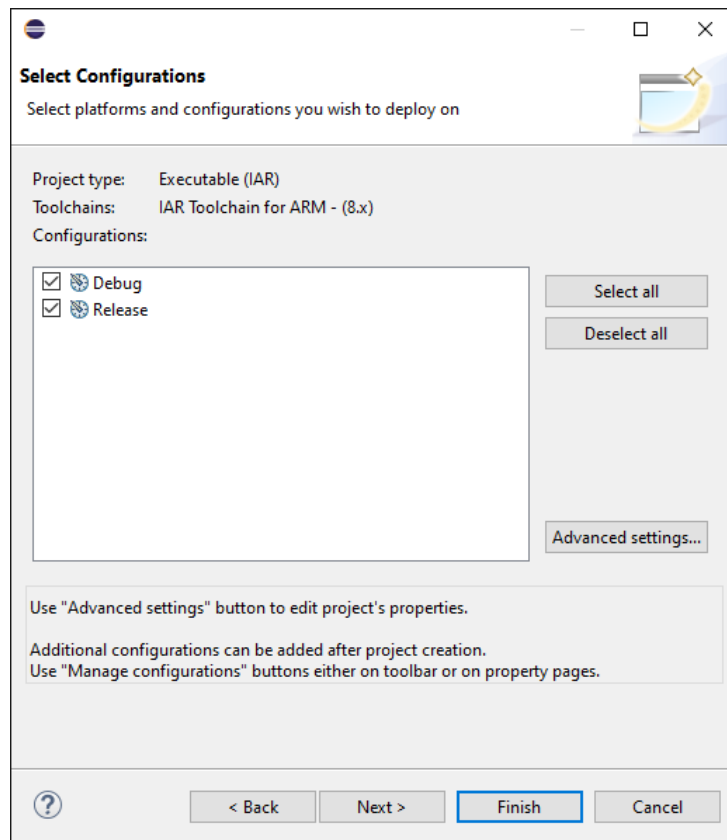


Figure 80 – New project build configurations

No changes should be performed in this panel so click "Next" again to continue.

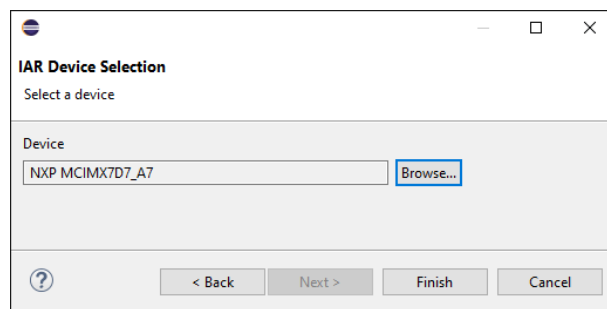
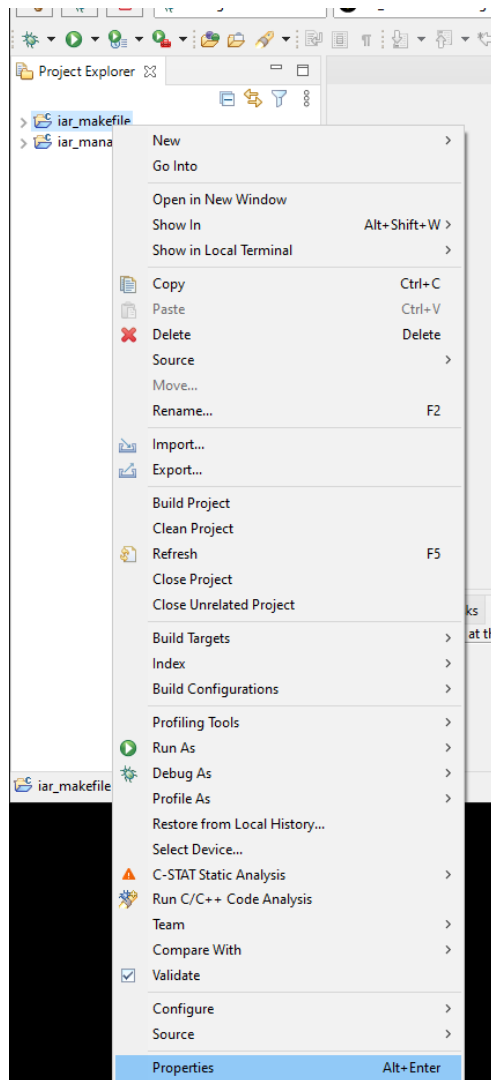


Figure 81 – New project device selection dialog

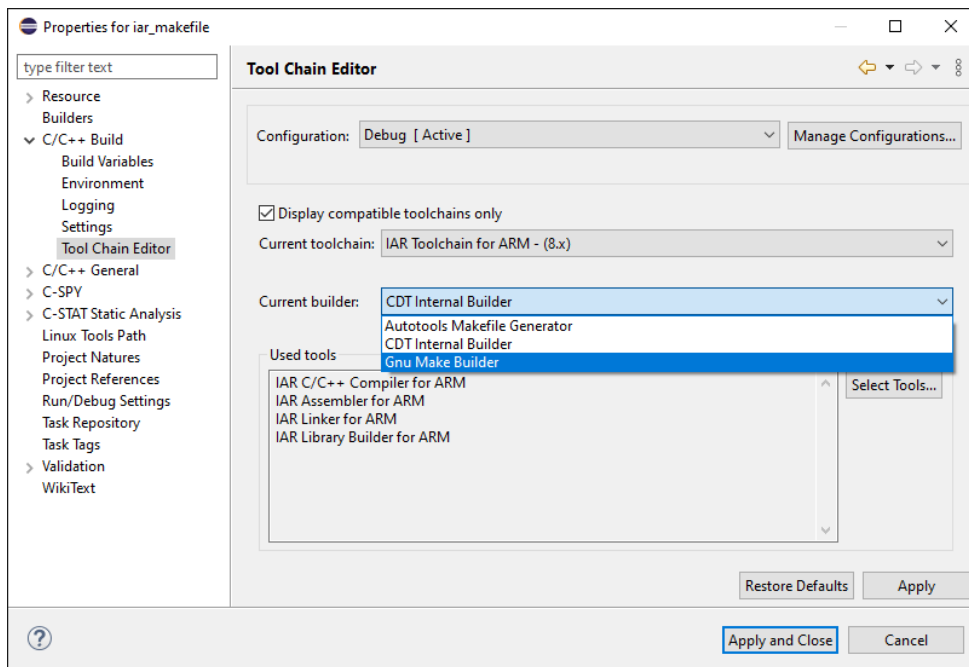
From the device selection dialog click "Browse" to select the MCU or SoC to target with this project. This example will use the NXP i.MX 7 SoC but any other supported MCU can be used.

Click "Finish" to end the wizard and create the new project. The newly created project should appear in the workspace.



**Figure 82** – Project properties context menu item

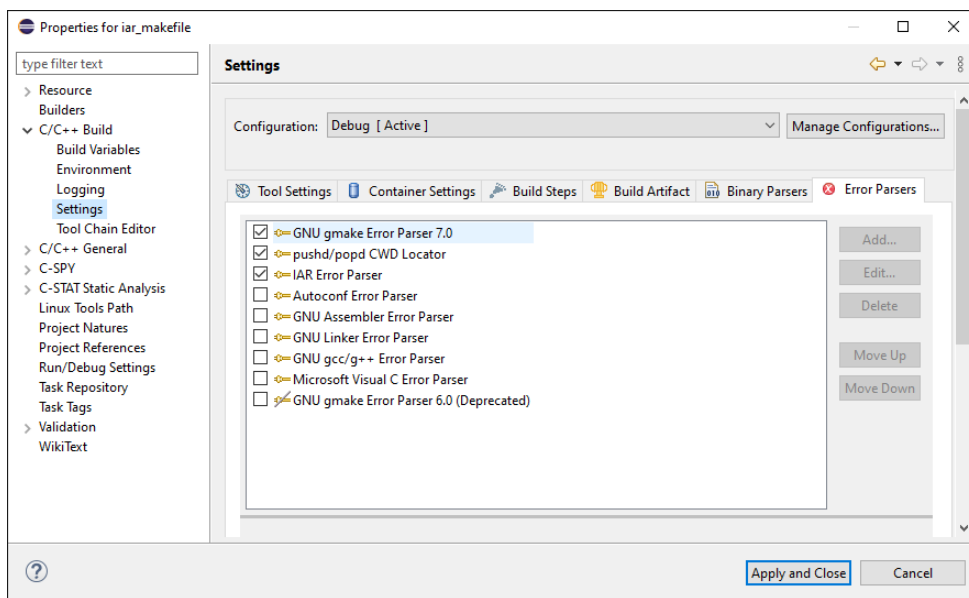
Right click the newly created project and select the "Properties" item from the context menu.



**Figure 83** – Project properties tool chain editor

In the project properties screen navigate to the "C/C++ Build->Tool Chain Editor" category. In the displayed panel change the "Current builder:" selection to "Gnu Make Builder".

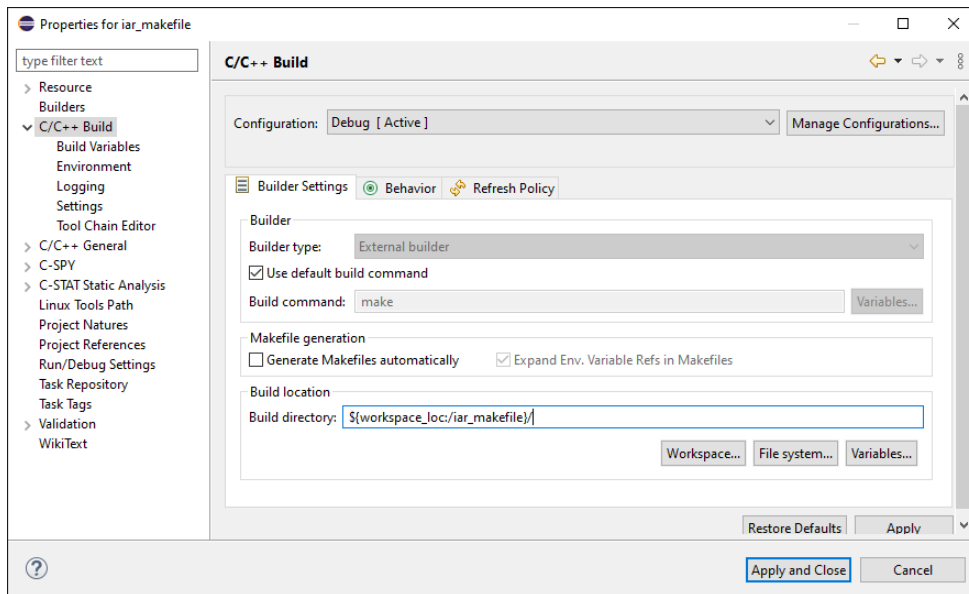
Now open the "Settings" category and open the "Error Parsers" pane. It's possible that the pane isn't displayed in which case two arrows will be displayed next to the panes tabs that can be used to navigate through the tabs.



**Figure 84** – Project properties error parsers settings

In the Error Parsers panel make sure that "GNU gmake Error Parser 7.0" and "IAR Error Parser" are checked. This will help CDT parse the build errors correctly and enable jumping to the source of an error from the console by double clicking on a displayed build error.

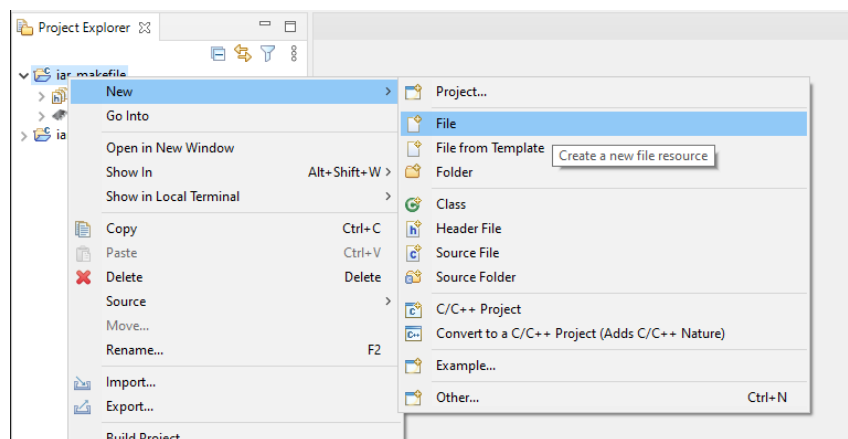
Now open the "C/C++ Build" root category.



**Figure 85** – Project properties build settings

From the "Builder Settings" pane uncheck "Generate Makefile automatically". Also modify the build location to point to the root of the project created earlier in this section. For the example project that was named "iar\_makefile" the resulting path should be "\${workspace\_loc:/iar\_makefile}/".

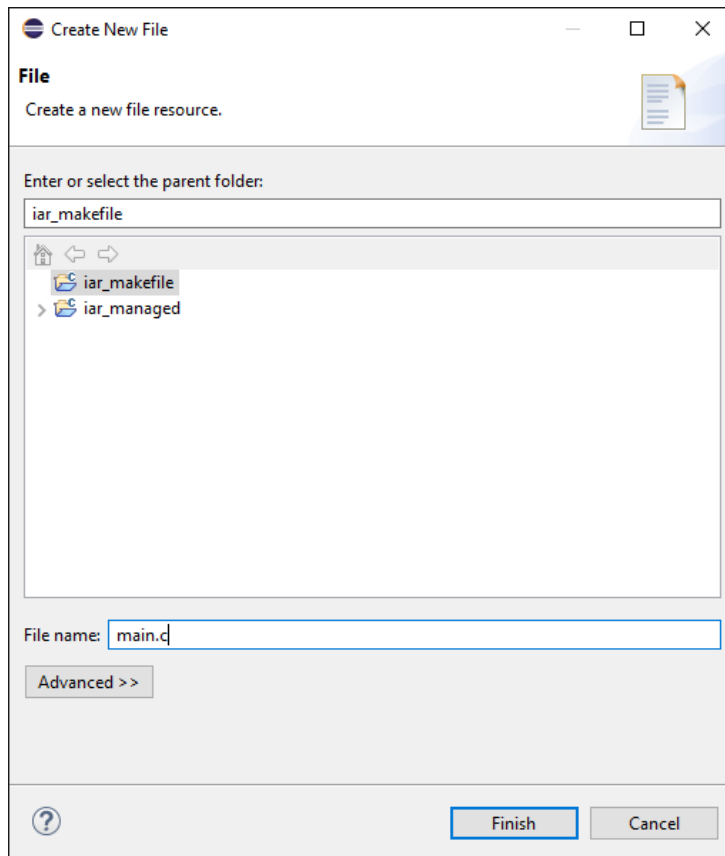
Click "Apply and Close" to save the settings and return to the workspace.



**Figure 86** – Add new file context menu item

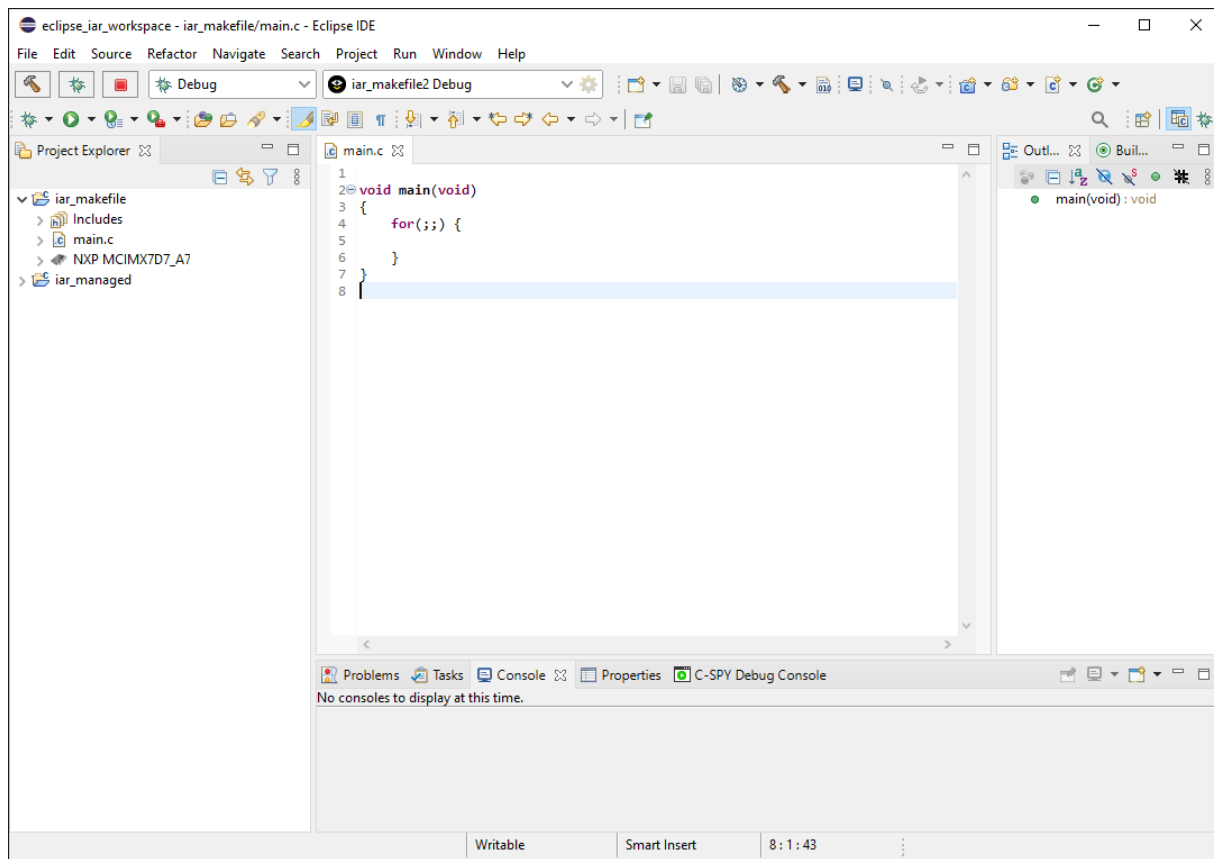


Now let's create a dummy source file to build. Right click on the project and select the "New->File" context menu item.



**Figure 87** – Add new file dialog

Type the name "main.c" as the source file name and click "Finish". The new empty file should open in the IDE.



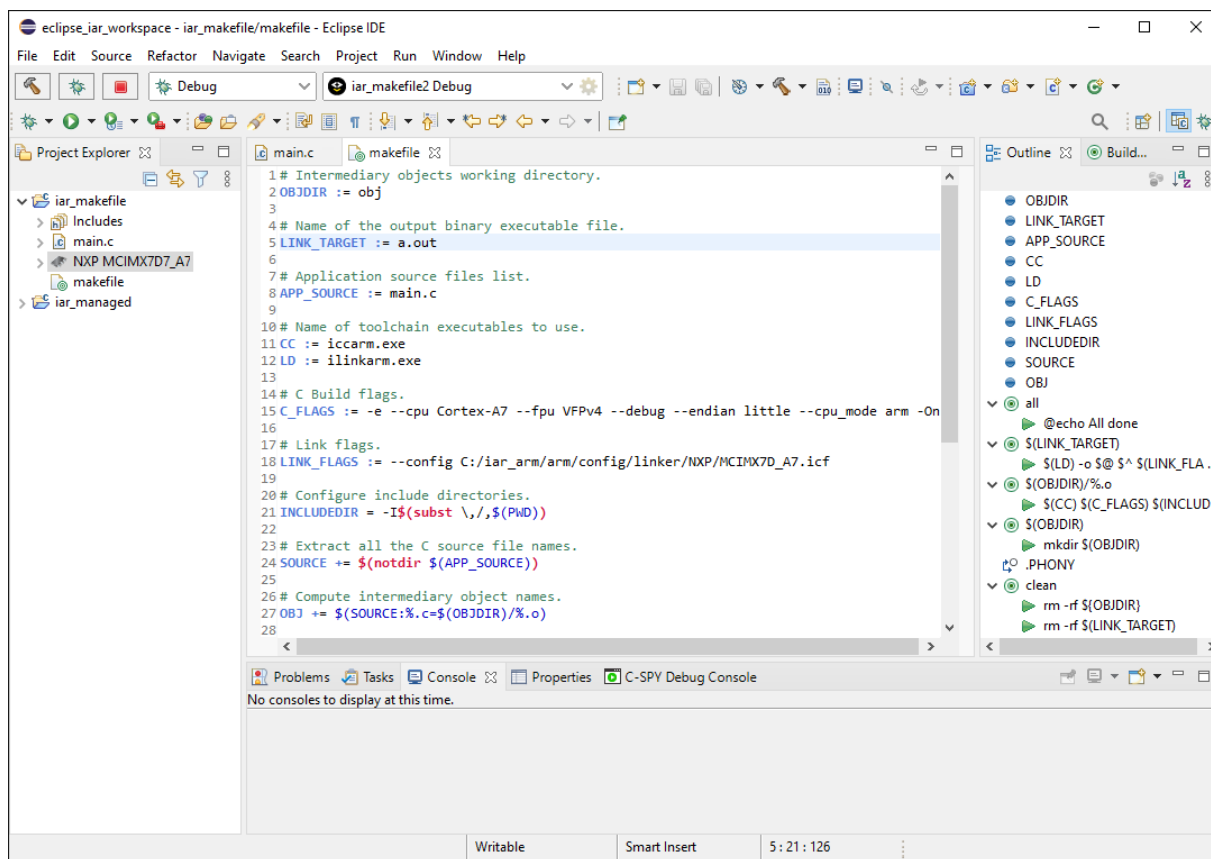
**Figure 88** – Example main with infinite loop

For the purpose of this example let's populate the source file with a dummy main function with an infinite loop.

```
void main(void)
{
    for(;;) {

    }
}
```

**Listing 2** – Example main function with infinite loop



**Figure 89** – Example makefile

Next add another file to the project named either "Makefile" or "makefile" and copy the content of the example makefile below. Additional information on writing a Makefile using the IAR tools can be found on our website [here](http://www.jblopen.com).

```
# Intermediary objects working directory.
OBJDIR := obj

# Name of the output binary executable file.
LINK_TARGET := a.out

# Application source files list.
APP_SOURCE := main.c

# Name of toolchain executables to use.
CC := iccarm.exe
LD := ilinkarm.exe

# C Build flags.
C_FLAGS := -e --cpu Cortex-A7 --fpu VFPv4 --debug --endian little --cpu_mode arm
          -On

# Link flags.
LINK_FLAGS := --config C:/iar_arm/arm/config/linker/NXP/MCIMX7D_A7.icf

# Configure include directories.
INCLUDEDIR = -I$(subst \,/,$(PWD))

# Extract all the C source file names.
SOURCE += $(notdir $(APP_SOURCE))

# Compute intermediary object names.
OBJ += $(SOURCE:%.c=$(OBJDIR)/%.o)

# Build everything.
all : $(OBJDIR) $(LINK_TARGET)
    @echo All done

# Firmware binary linking.
$(LINK_TARGET) : $(OBJ)
    $(LD) -o $@ $^ $(LINK_FLAGS)

# C files build.
$(OBJDIR)/%.o : %.c | $(OBJDIR)
    $(CC) $(C_FLAGS) $(INCLUDEDIR) -o $@ -c $<;

$(OBJDIR) :
    mkdir $(OBJDIR)

.PHONY: clean
clean:
    rm -rf ${OBJDIR}
    rm -rf $(LINK_TARGET)
```

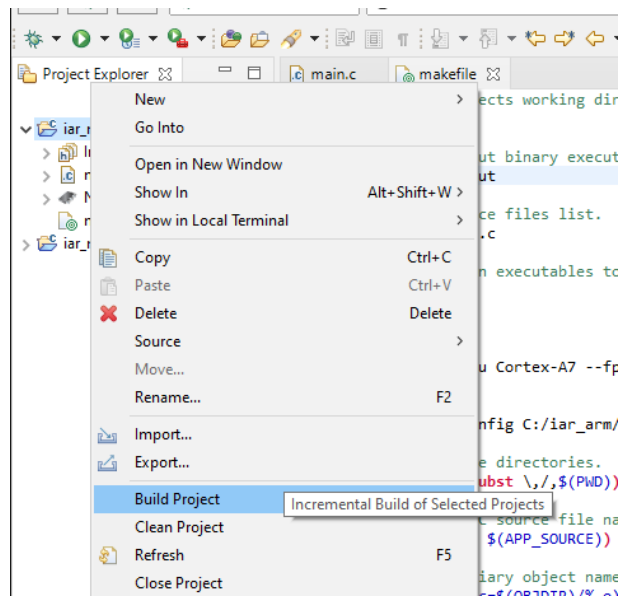


Figure 90 – Build project context menu item

With everything set up right-click again on the project and select "Build Project" from the context menu.

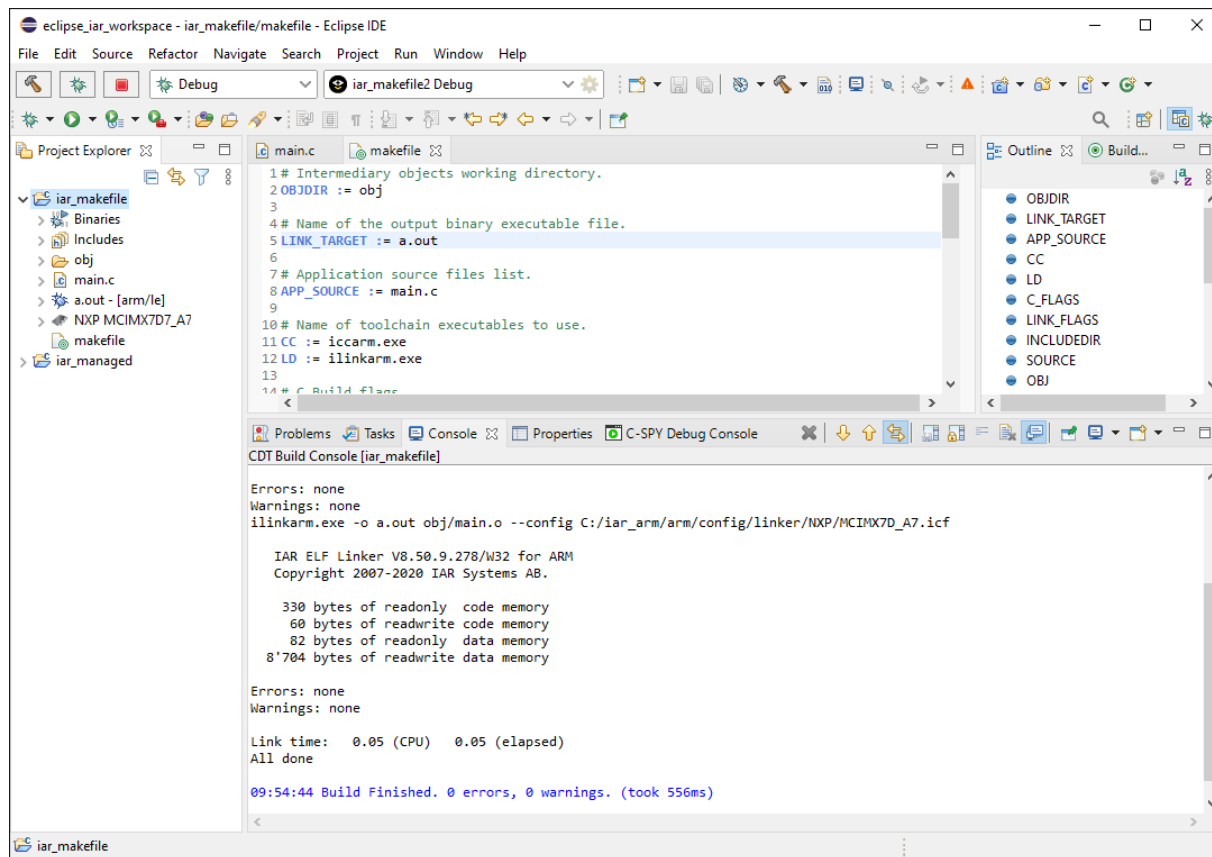


Figure 91 – Project built successfully

If all went well the project should now be built and ready to be loaded onto a target and debugged.

## 4.2 IAR Eclipse Makefile Project Debugging

With the Makefile project properly set up and built, it's now time to create a debug configuration. The instructions are very similar to creating a debug configuration for a managed build project except that it is important to properly set the project and binary file to debug as they may not be detected automatically.

Start by searching for and clicking the "Run->Debug Configurations" menu. Note that while it is possible to create a debug configuration by right-clicking the project root and selecting "Debug as" this will launch a debug session without having the chance to configure it first which is almost certainly not the desired outcome.

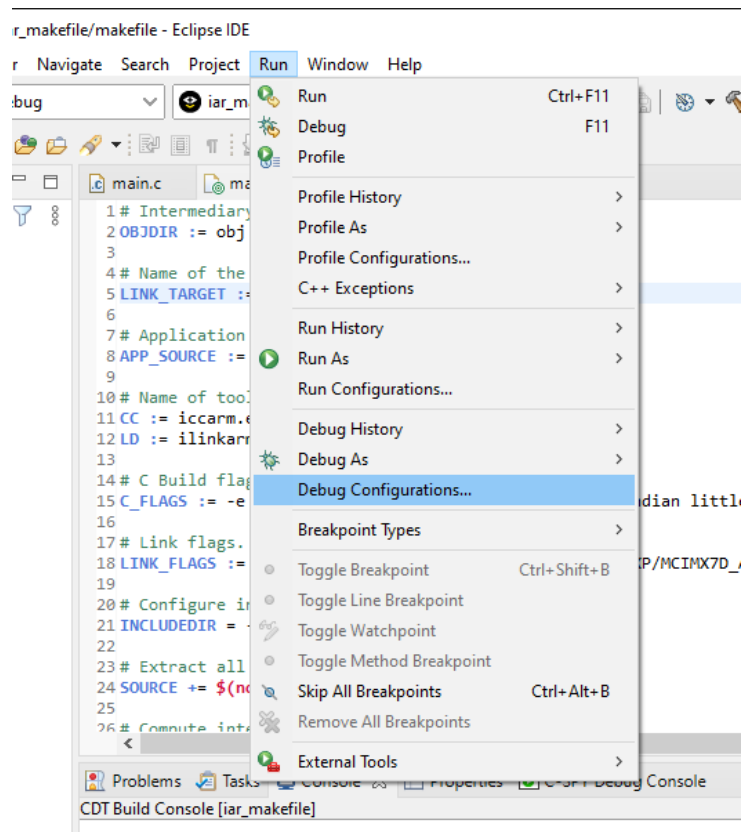


Figure 92 – Debug configurations menu item

The "Debug Configurations" panel should appear.

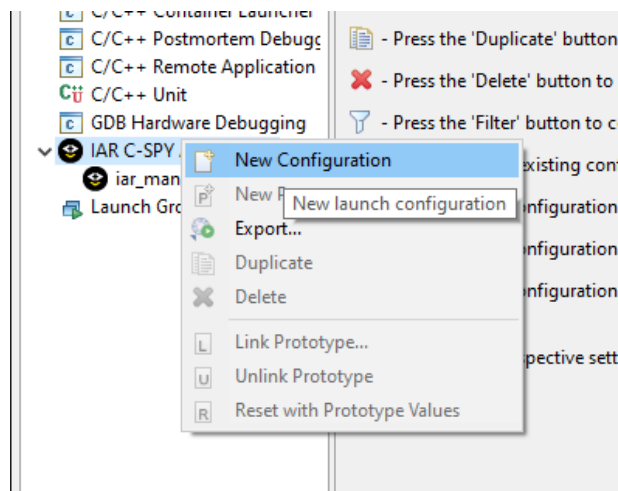
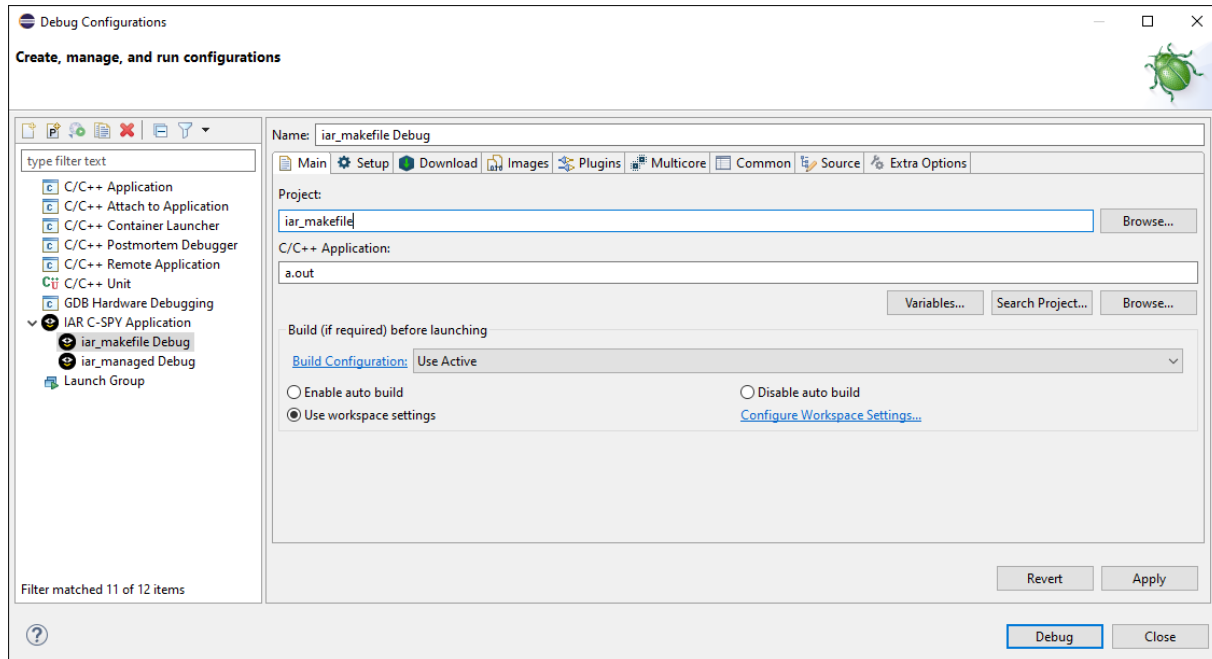


Figure 93 – New IAR C-SPY debug configuration context menu item

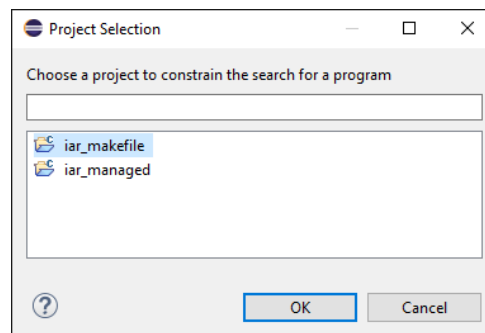
On this panel right-click "IAR C-SPY Application".

From the context menu select "New Configuration". This should create a new configuration which should be displayed on the right panel.



**Figure 94** – New IAR C-SPY debug configuration

By default the new configuration is associated with the active project. If the project named in the "Project:" field is not the correct one, or the field is empty, click "Browse..." next to it to select a project.

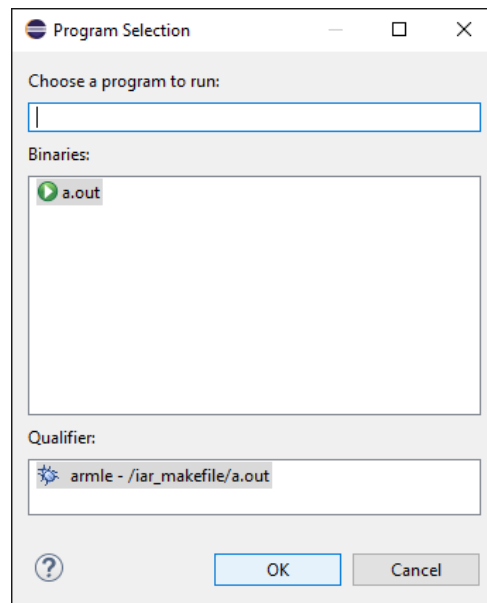


**Figure 95** – Debug project selection dialog

Select the desired project from the "Project Selection" dialog and click "OK".

In a similar fashion, verify that the "C/C++ Application:" field points to the correct executable binary to debug. If it is incorrect or left empty click "Browse..." underneath the field to select a binary file.





**Figure 96** – Debug program selection dialog

From the selection dialog select the binary file to debug. If nothing is displayed make sure that the project was built correctly. Also if the binary file has an unknown extension, other than ".out" or ".axf" for example, it may be necessary to manually type the name of the file instead. Once done click "OK".

Now select the "Setup" tab to display the debug setup configuration panel.

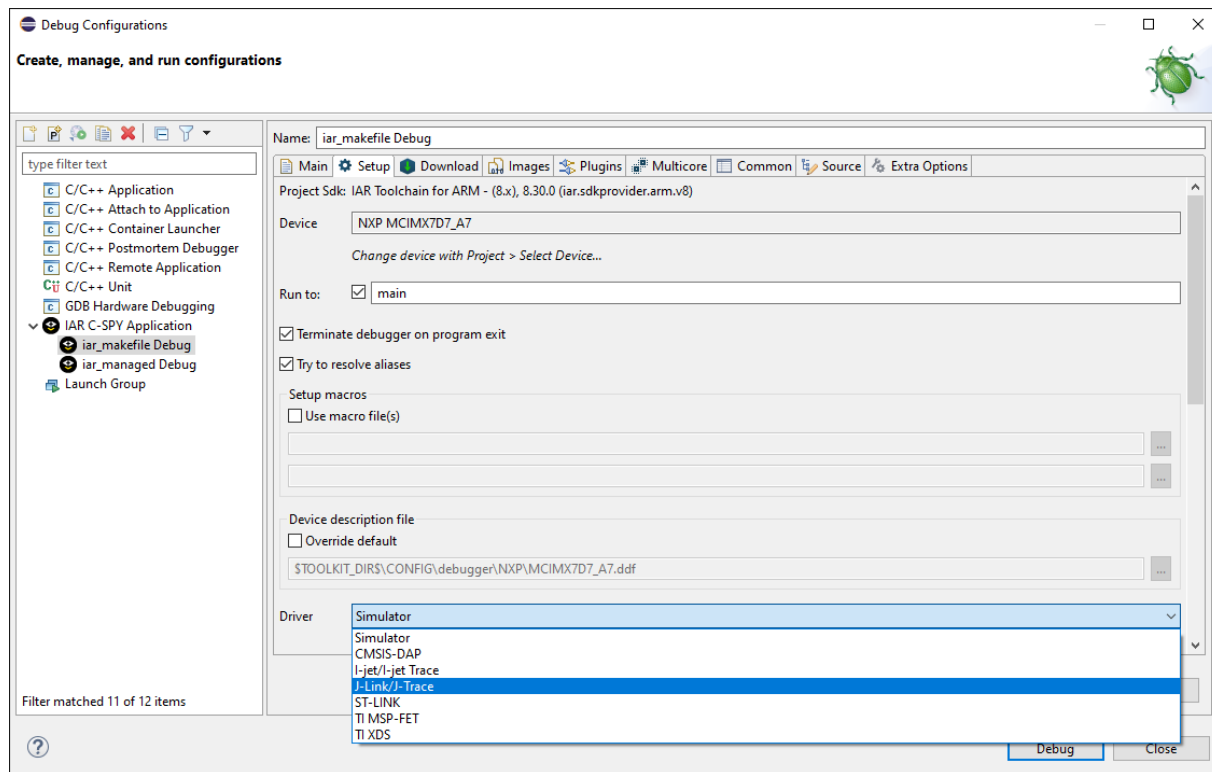


Figure 97 – Debug setup panel

By default IAR will launch the executable file using the IAR simulator. To debug a hardware target make sure to select the correct debug probe. This example uses the Segger J-Link debug probe but any supported probe can be used.

Also make sure that the correct device is mentioned in the device field. If the incorrect device is displayed or the field is empty, it is possible to select a device for the current project using the "Project->Select Device..." menu item from the main IDE screen.

If all is well click "Debug" to launch the debug session.

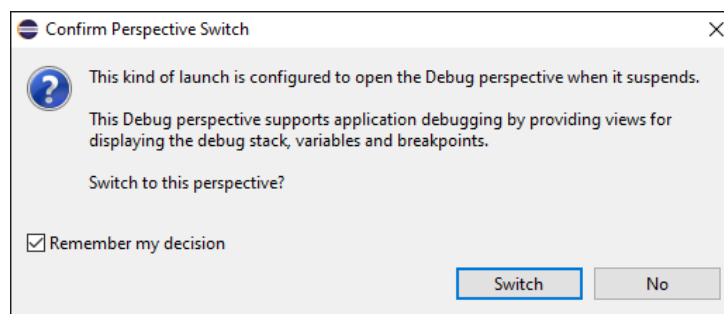


Figure 98 – Eclipse switch perspective prompt

Eclipse will probably warn that launching a debug session is associated with the "Debug" perspective. Clicking switch will rearrange the selection of panels within Eclipse to show the panels relevant to debugging. Checking "Remember my decision" will stop Eclipse from asking the same question each time a debug session is launched.

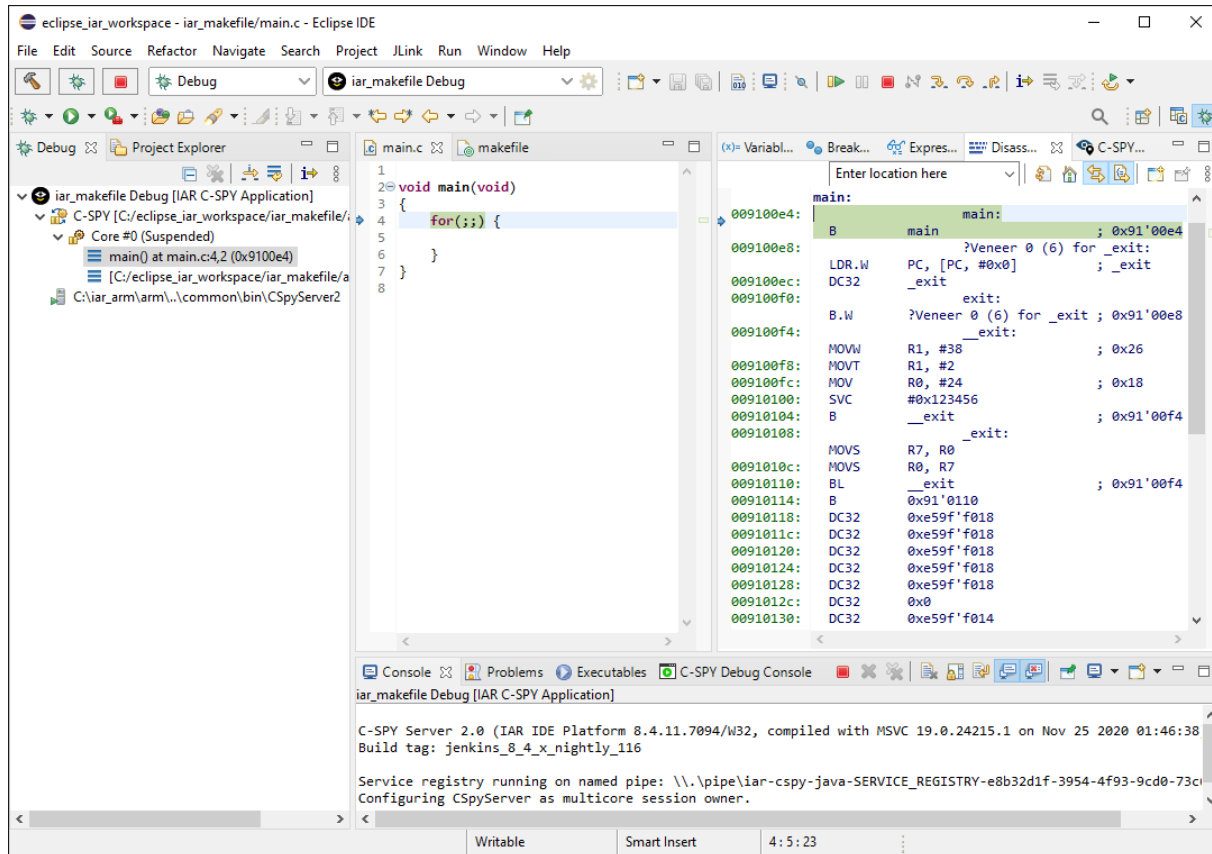


Figure 99 – Debug session

If all went well, the project should now be loaded on that target and the debugger halted at the main loop.

## Document Revision History

Ver- sion	Release Date	Description
1	2021-02-01	- Initial release.
2	2021-03-02	- Minor fixes. - Added link to additional information on using Makefiles with IAR.

**Table 1** – Document Revision History