TiniARM Development with Eclipse

By: James P. Lynch

Introduction

When I was a boy in the 1950's, I developed my interest in engineering and science through Heathkits. I built a shortwave radio first and listened to Radio Moscow during the Cold War. I then built Ham Radio gear, a couple televisions, the H8 computer and the Hero Jr. Robot over the years. Heathkit's devotion to customer support and clear and readable construction manuals made it possible for the little guy to build a fairly complex piece of electronics.

Unfortunately, the cost advantage of "building it yourself" faded away with automatic surface mount "pick and place" machines and Heathkit withdrew from the business. You can still see their current products (motion detector floodlights) for sale at Home Depot.

For a person today who wants to dabble in computer electronics, the stumbling block in the past has been the cost of the parts and the outlandish cost of the software development tools such as the compilers, linkers and editors. It's not unusual for a technology company to plunk down $20,000 for a software development system for a modern microcomputer.

Things are finally changing for the better. Serious embedded processors (the 32-bit machines with on chip RAM and FLASH) are suddenly affordable. There's no better example than the **TiniARM** computer offered by New Micros Inc. in Dallas, Texas.

This little postage stamp sized computer has the following specifications, as listed on the New Micros Inc. web site:

TiniARM Specifications

- Board Size: 1.0"(W) x 1.3"(L) x 0.4"(H)
- Weight: 0.2 Ounces
- Programming Language:
  - **C/C++ Eclipse/GNU Integrated Development Package (this tutorial)**
  - **In-System Programming Flash Utility** (free download)
- ARM7TDMI-S CPU, 32-bit microprocessor
- CPU operating range up to 60Mhz
- Real Time Clock
- On-chip Memory
  - 128KByte Program Flash
    - 10,000 erase and write cycles
    - 1ms programming time for up to a 512 byte line
    - Single sector erase (8KB), or the whole chip erase is done in 400ms

    - 64KByte Static Ram
-
- 16 General Purpose Digital I/O lines share functions with,
    - 4-wire SPI Interface
    - 4 PWM channels(3 on 24-pin connector, 1 shared with Jtag signal on other connector)
    - 7 Timers
    - 1 SCI (TTL) channel with the addition of a modem interface
    - 3 external Interrupts
- Two Serial Communication Interface (SCI)
    - UART0 is dedicate for RS-232 serial channel
    - UART1 is TTL signals and shared function with GPIO's
- I2C Serial Interface
- JTAG connection for flash programming/debugging
- Two low power modes, Idle and Power down
- WatchDog Timer
- Onboard three user's leds
- Onboard 5.0V, 3.3V, & 1.8V linear regulators
- 2x12 header pin connector for Power, Serial, and I/O's connection

The price of this little gem is $69.00. A development board with the TiniARM module, a power supply and RS-232 cable is just $95.00.

If the hardware can be made so affordable, can the same be so for the software required to develop applications? No layman is going to buy a $95.00 microcomputer and then spend $1500 and upwards on a software development system.

Can a good editor, compiler and linker for the TiniARM be acquired for nothing?

The answer is “Yes it can!”

In this tutorial, I will demonstrate in detail how to build the Eclipse open-source Integrated Development Environment (IDE) and then upgrade it for C/C++ development by installing the CDT plug-ins. I’ll then show how to download Cygwin on your computer to pave the way for the open-source (free) GNU software compiler suites. We’ll download a complete GNU software development package for Windows computers and then augment that with a GNU compiler suite specifically built for the ARM processors.

We’ll then download the Philips LPC2000 Flash Utility and install it into Eclipse so you can run it after building your application. Finally, I’ll show how to download the New Micros TiniARM test program and create it as an Eclipse project.

We’ll build the project, download it into the TiniARM’s flash memory and execute it. When completed, you will have a complete and modern IDE to develop code for the TiniARM in C and C++.

The only feature I didn’t address is debugging. This will take some more work since a resident monitor called RedBoot must be installed to support Eclipse/GNU’s GDB debugger. I hope to update this tutorial with that information soon.

Who knows, we may be able to bring back Heathkit or a version thereof some day!

1. Why Develop with Eclipse

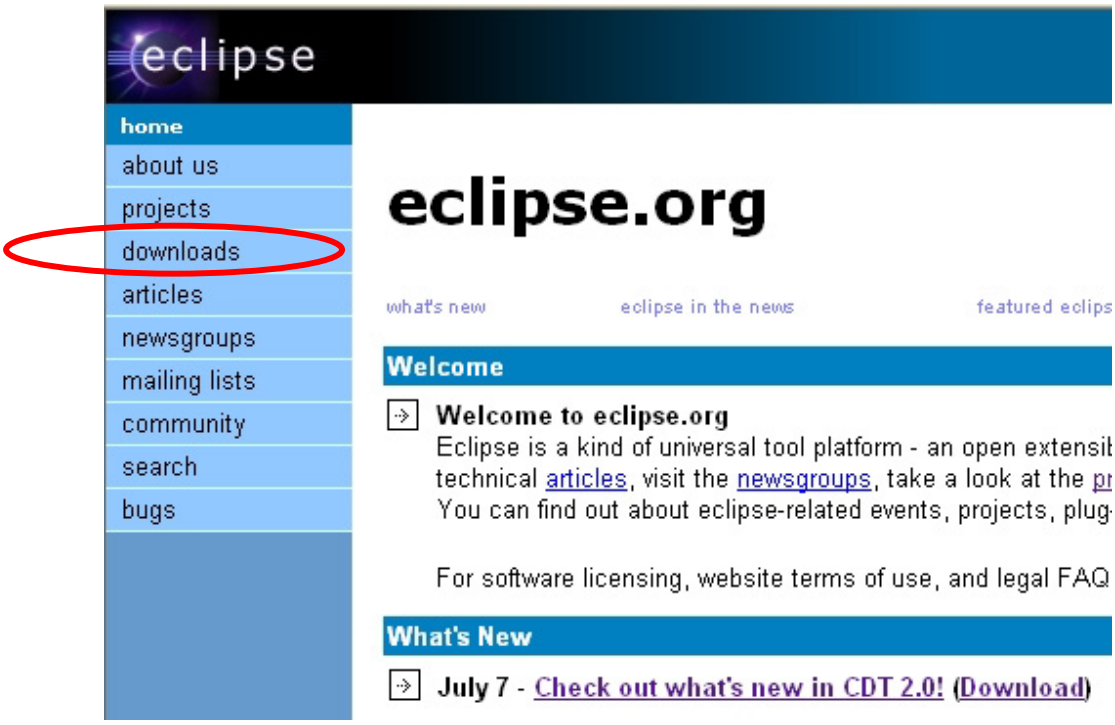
The simple answer is that Eclipse is a **free** open-source equivalent of Microsoft's Visual C++ development system. Eclipse was developed by IBM and released to the open-source community (namely hackers and professionals who know a good deal when they see it) and since then, it has been maintained and upgraded by IBM and a host of collaborators. While Eclipse was originally developed to create Java software, it can be extended to develop C/C++ software by adding the **CDT** (C++ Development Toolkit) plug-in.

The Eclipse/CDT together forms a professional IDE (Integrated Development Environment) to develop C and C++ software for the **TiniARM** computer. When you command Eclipse/CDT to build your software, it invokes a MAKE file. If you have the open-source **GNU** compiler tools installed on your computer, the GNU toolkit will run the MAKE utility and the compilers, linkers, etc. needed to build your downloadable hex file.

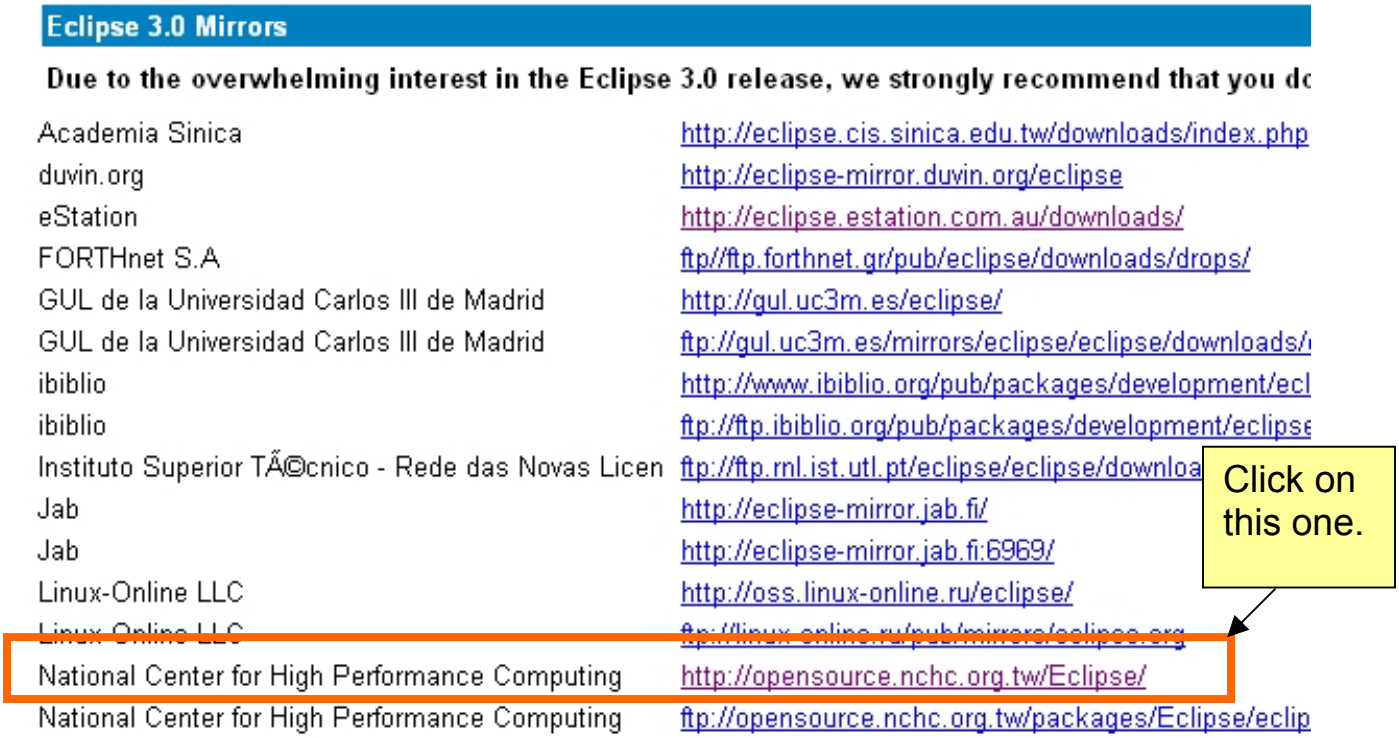
To be completely honest, setting all this up is a bit tedious and time consuming. So take a deep breath and let's get started.

2. Downloading Eclipse

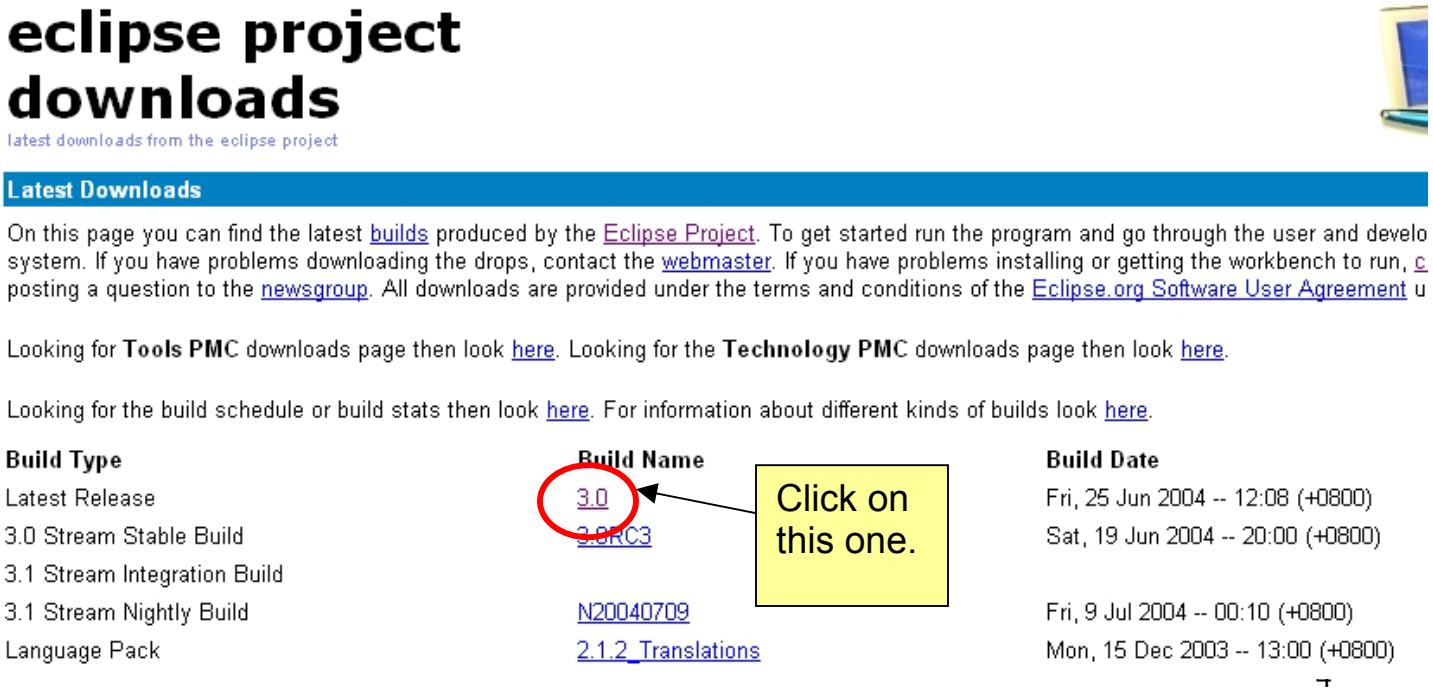
Using your internet browser, such as Internet Explorer, go to the Eclipse web site: **www.eclipse.org** You should see the following screen. Click on **downloads**.



First Eclipse will ask you to choose a download mirror. Now here's a real crap shoot. I picked the "**National Center for High Performance Computing**" because they must have some mucho teraflops computers there.

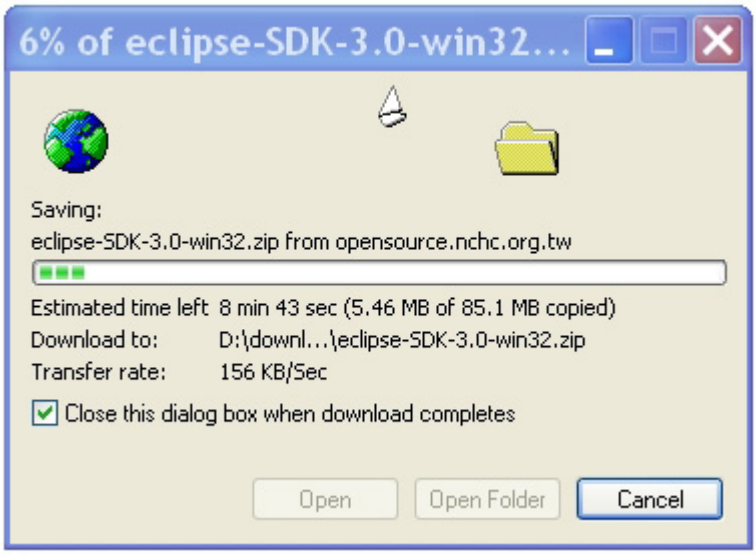


Now you may select which version of Eclipse to download. Stick to the "**most recent release**" as this is a version that has been thoroughly debugged. The nightly builds and stream builds are for Eclipse developers. In July of 2004, Eclipse **3.0** is the one to choose.



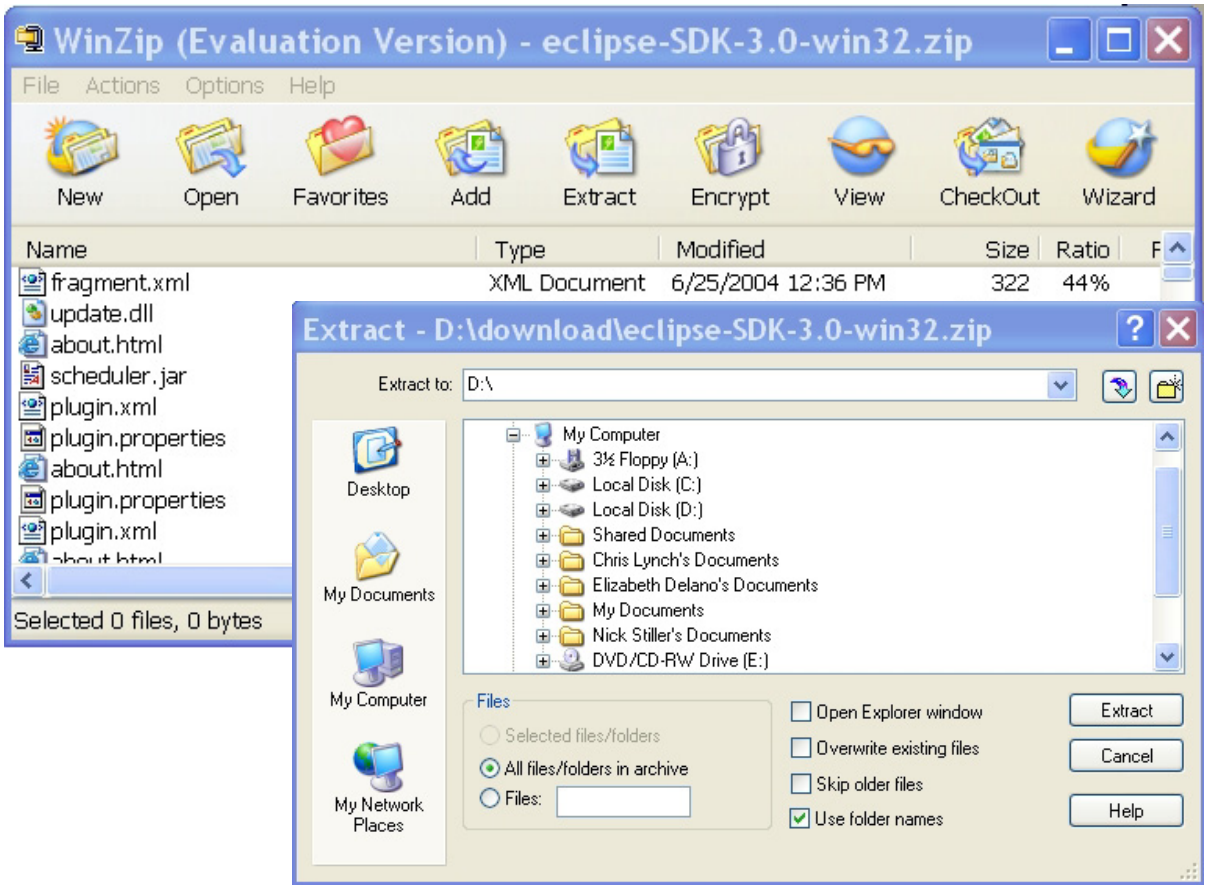
Since my computer is a Windows XP machine, I clicked on "**Windows 98/ME/2000/XP (http)**" to start the download.

Be forewarned that Eclipse is an 85.1 megabyte download, so even on a cable modem it will take 10 minutes. I directed it to download to an empty "download" directory on my hard drive (in my case, **d:\download**).

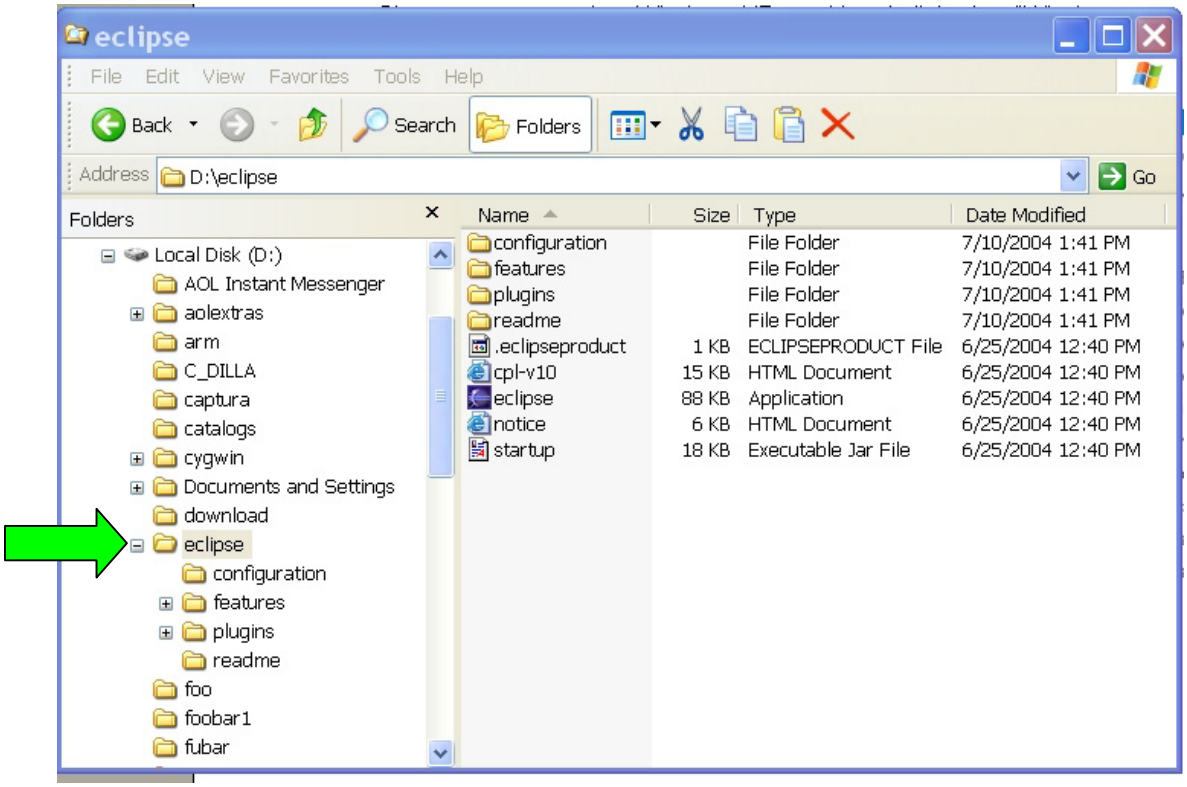


Eclipse is downloaded as a ZIP file, so you can use **WinZip** to extract it to your drive. In my case, I extracted it to the D: drive root and it created a **d:\eclipse** folder on that drive.

If you don't have WinZip, you can search for it using **Google** and the free evaluation version will suffice here.



Since I extracted **Eclipse** to my **D:** drive, here is where it put it.



If you double-click on the eclipse application (**eclipse.exe**) in the directory above, this will start up the eclipse application. Note that Eclipse is just an .exe file that runs, there are no entries in the Windows Registry made for it (what could be simpler).

If the following screens appear, we have successfully installed Eclipse!

Take the default on the above dialog and all your projects will go into the **d:\eclipse\workspace** directory.

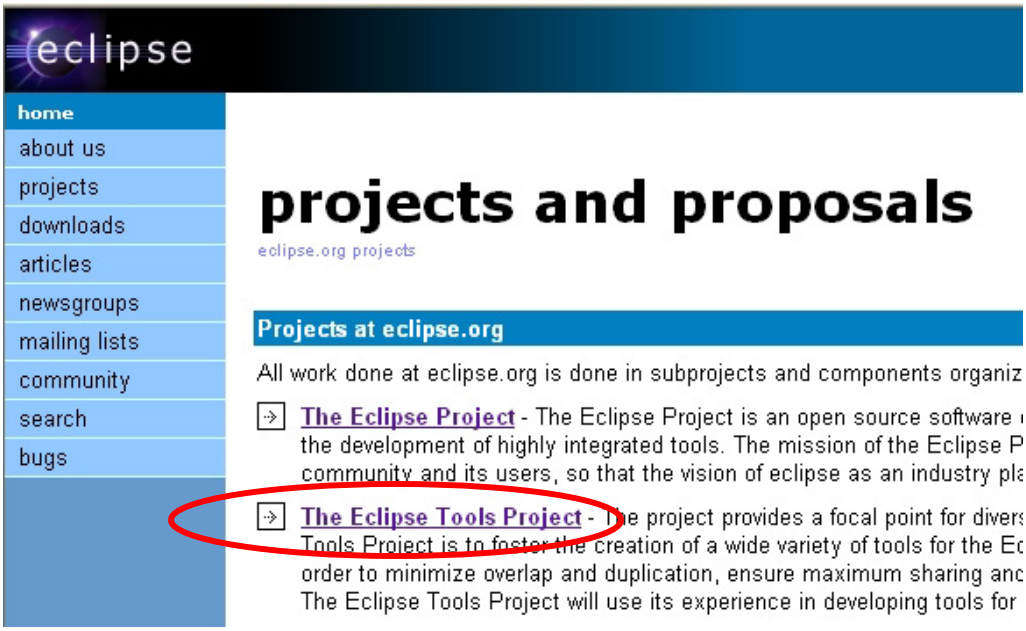
Normally at this point, Eclipse users will log onto the Sun Microsystems web site and download the latest and greatest JAVA development kit (which is free to play with but if you develop a commercial product with it, they want their cut).

Instead, we will download the **Eclipse CDT** plug-ins which will allow development of C/C++ programs.

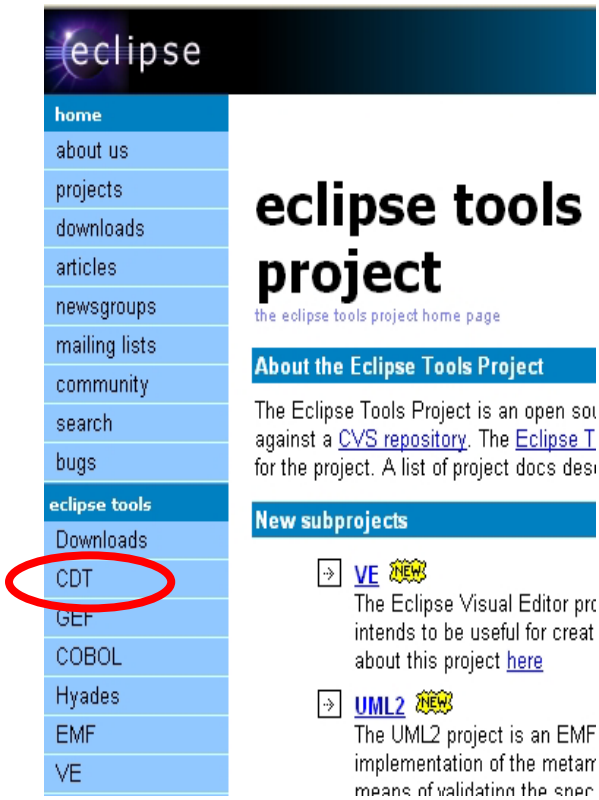
3. Downloading CDT

Using your internet browser, such as Internet Explorer, return to the **Eclipse** web site: **www.eclipse.org** You should see the following screen. Click on **projects**.

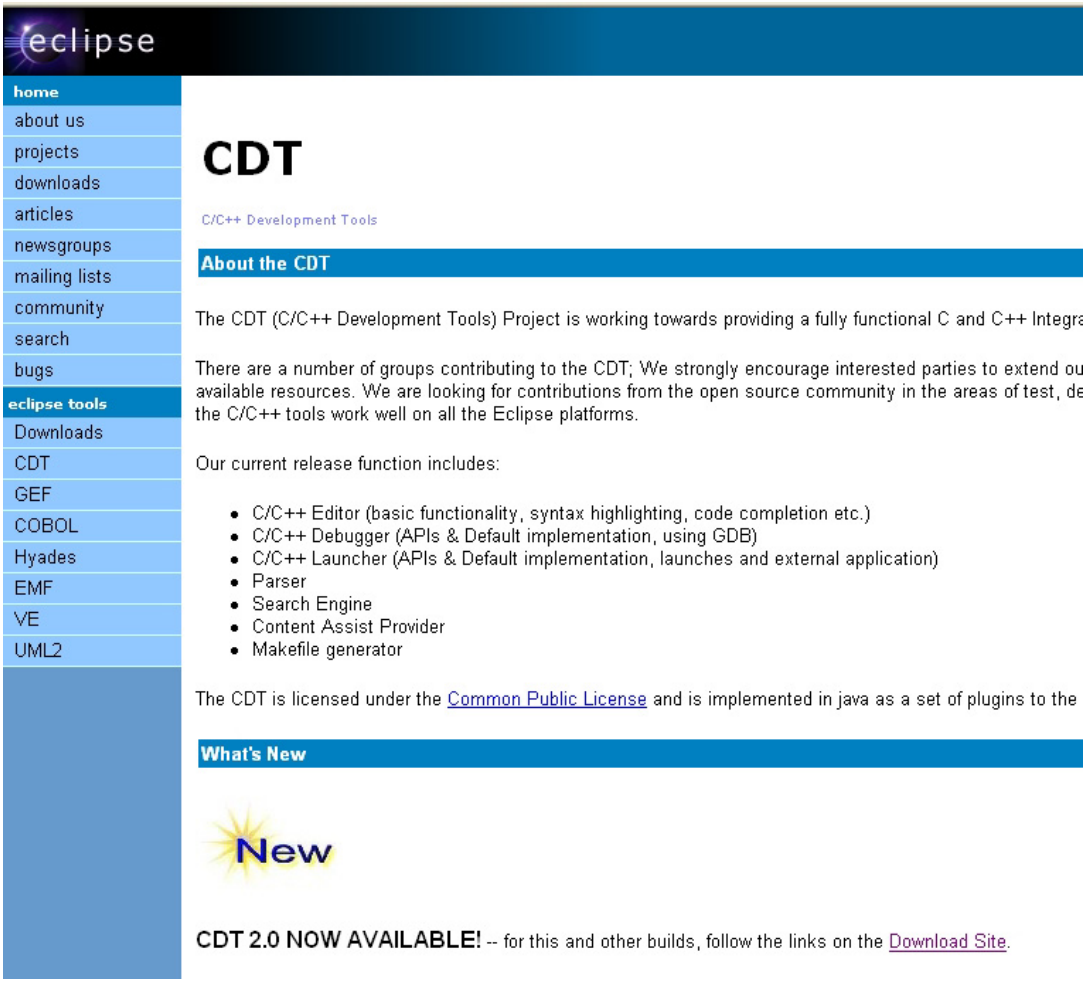
Now click on **The Eclipse Tools Project**.



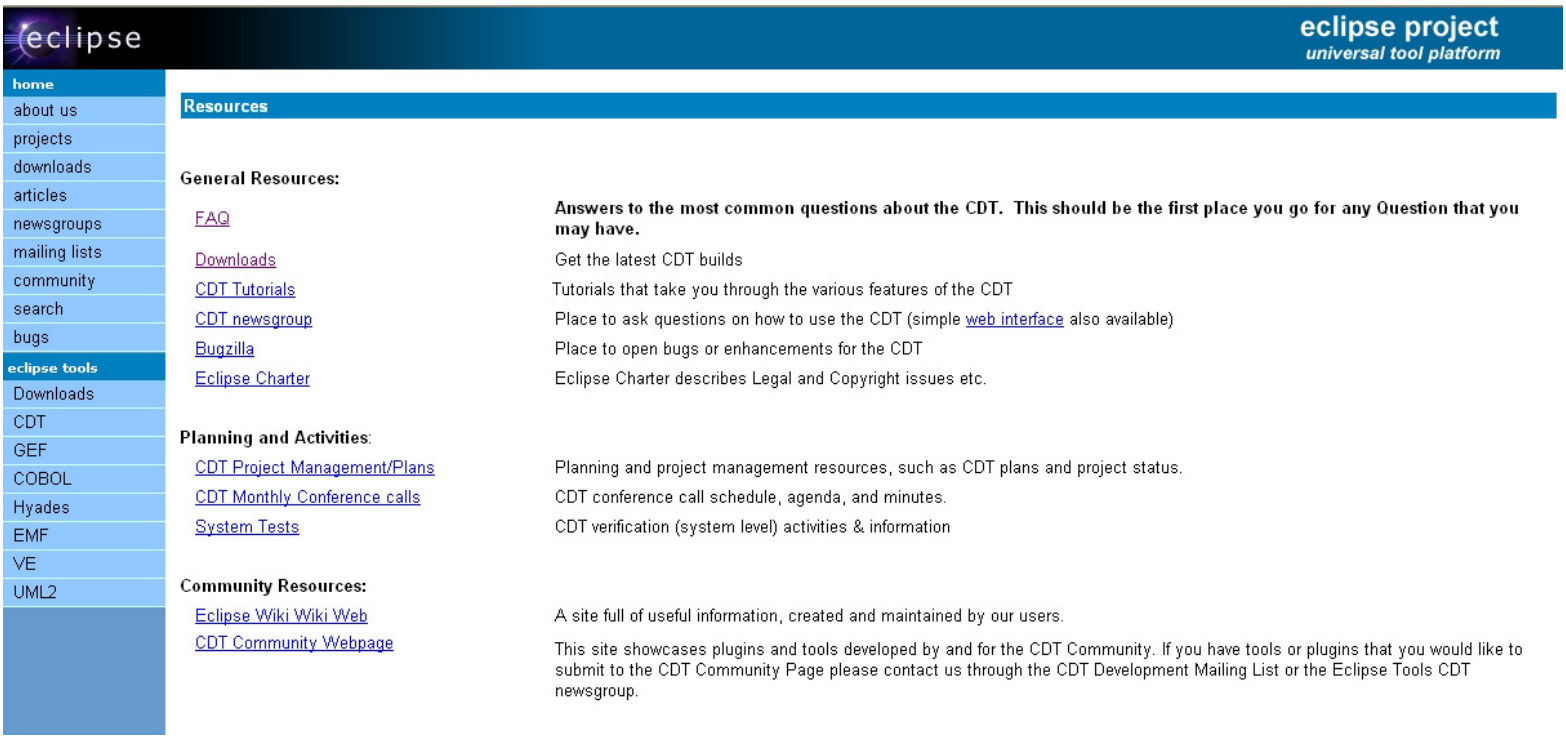
Now click on **CDT** to bring up the CDT Project.



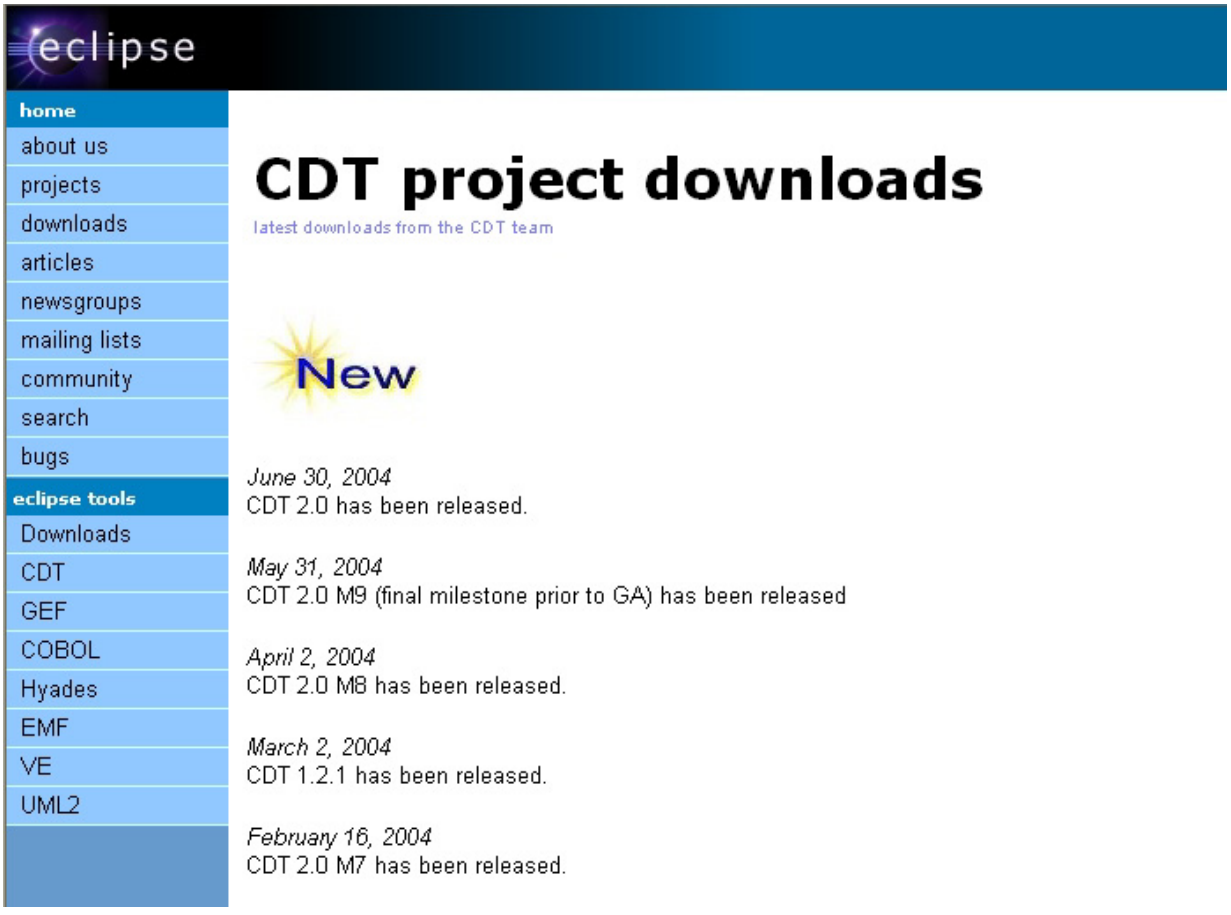
This is the CDT project screen.



If you scroll down a bit, you'll see a link to the CDT Download site.



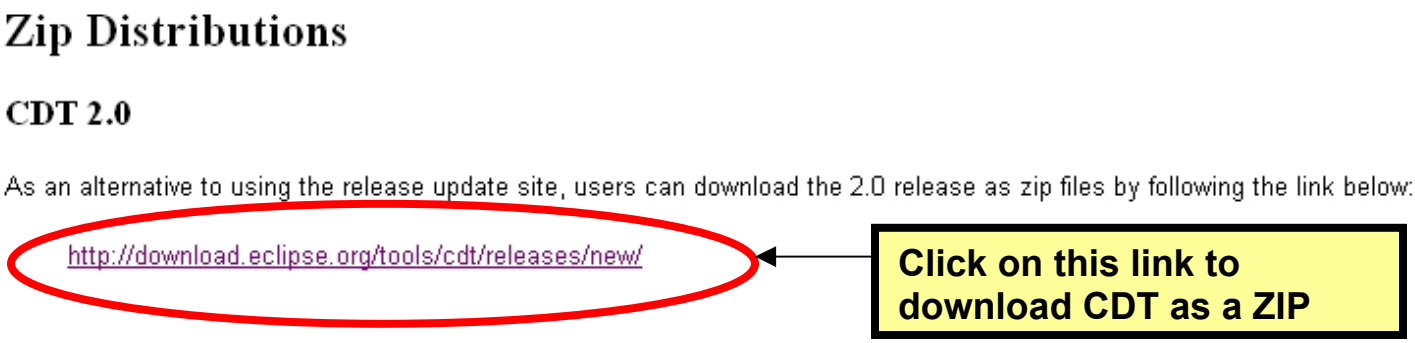
You should now see the CDT download screen, as shown directly below.



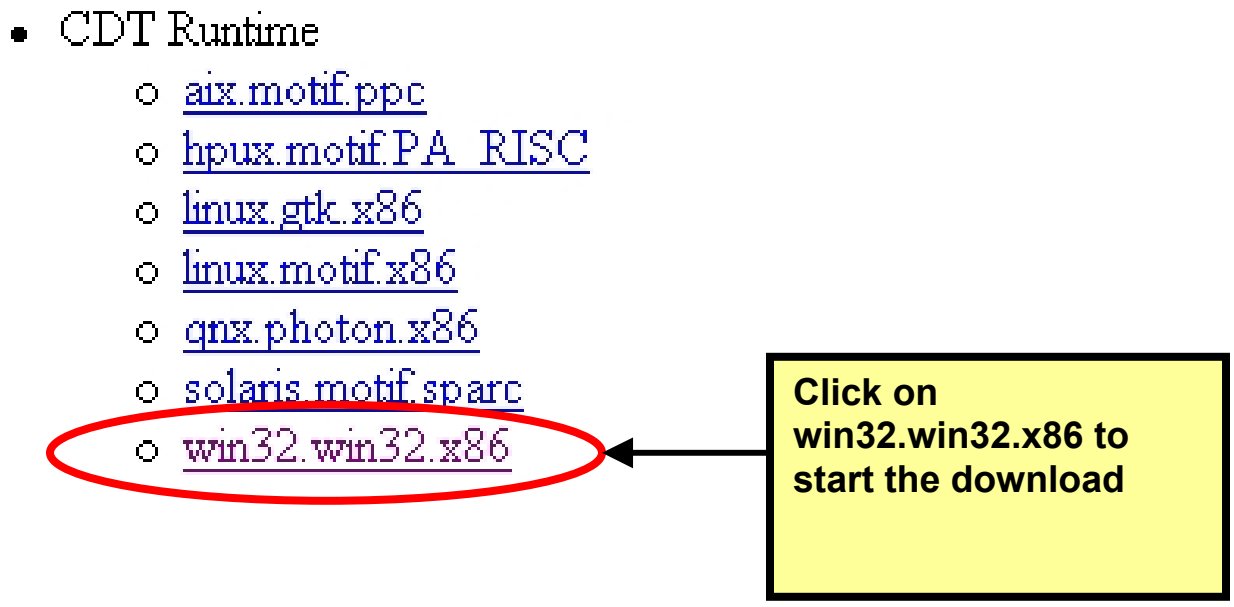
We must use **CDT 2.0** with **Eclipse 3.0**

If you scroll down a bit on the page, there will be a spot to select the CDT 2.0 Zip Distributions. There is an update manager technique that can install CDT hands off, but I've had problems with it.

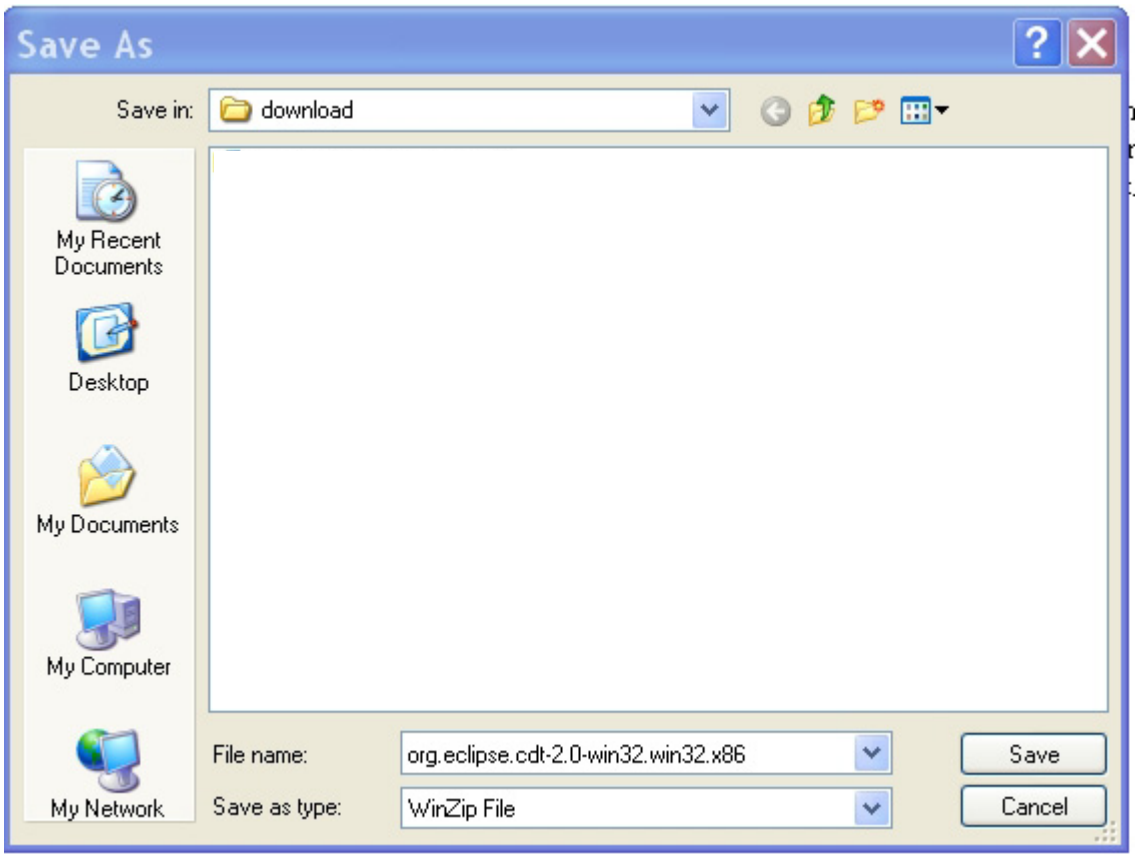
Note that this tutorial was prepared in July 2004. Eclipse and CDT are always in flux. Be sure to browse around the Eclipse web site and be sure that the CDT version that you choose is compatible with the Eclipse version you previously selected.



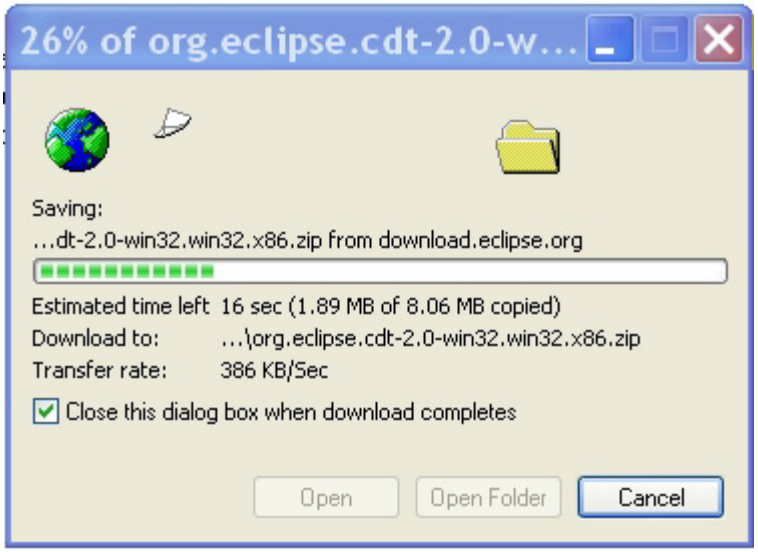
We finally select **win32.win32.x86** from the **CDT Runtime**. Selecting the **CDT SDK** brings in the source files which are little interest here.



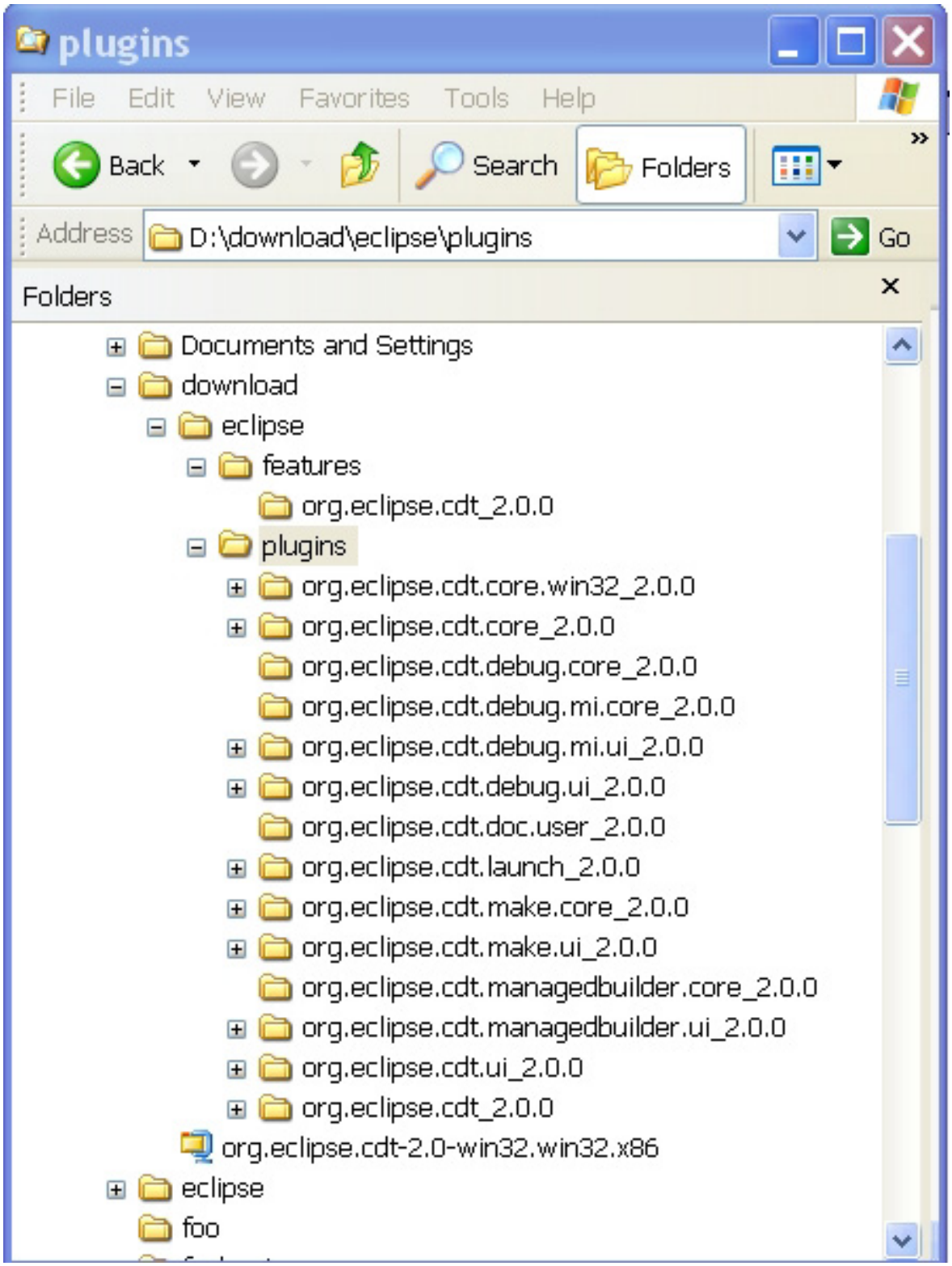
Once again, I downloaded to the empty scratch download directory.



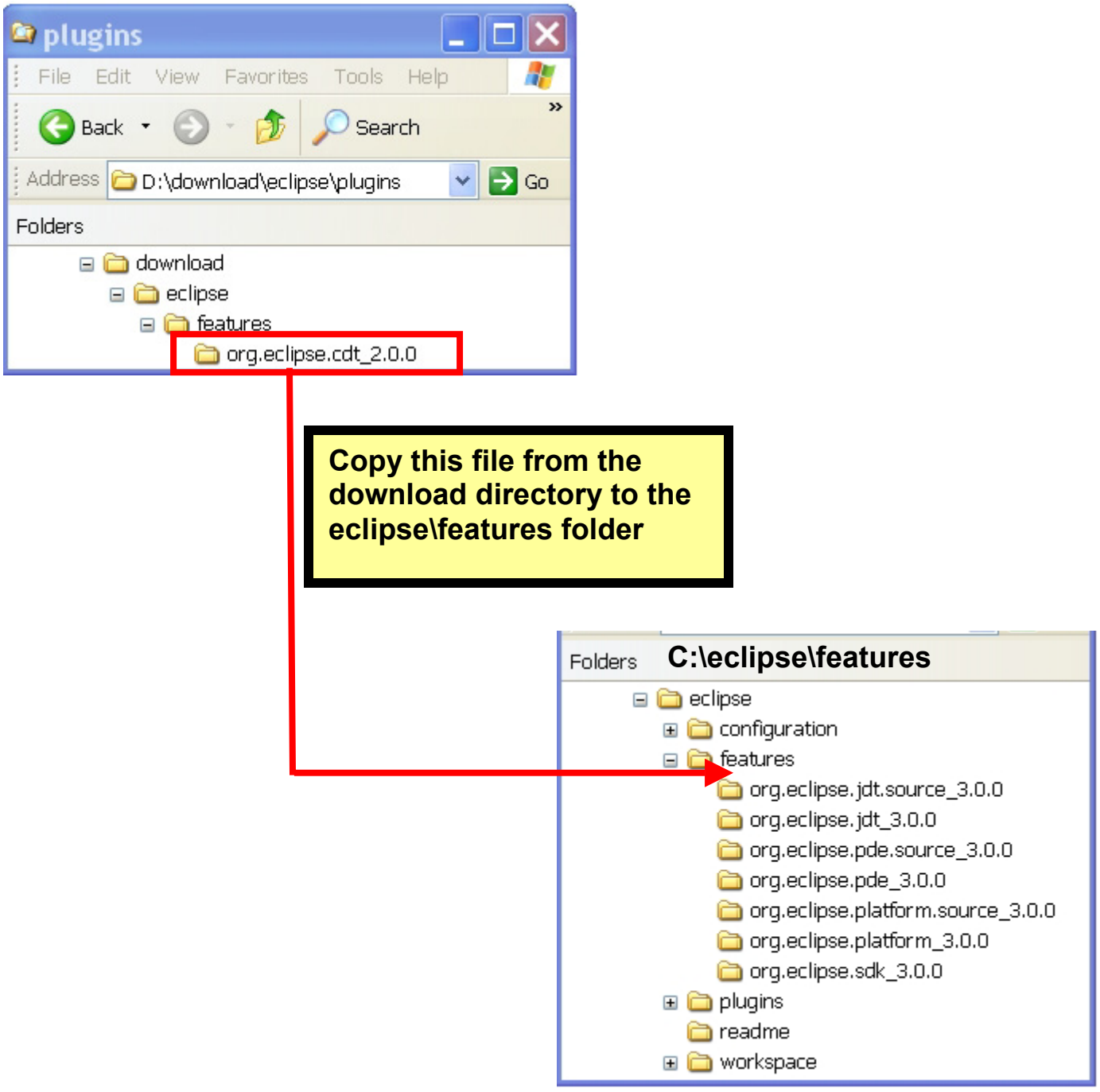
The **CDT** is an 8 Megabyte download; it takes about 20 seconds on a cable modem.



If you extract the CDT zip file to the same download directory, you will get the following result.

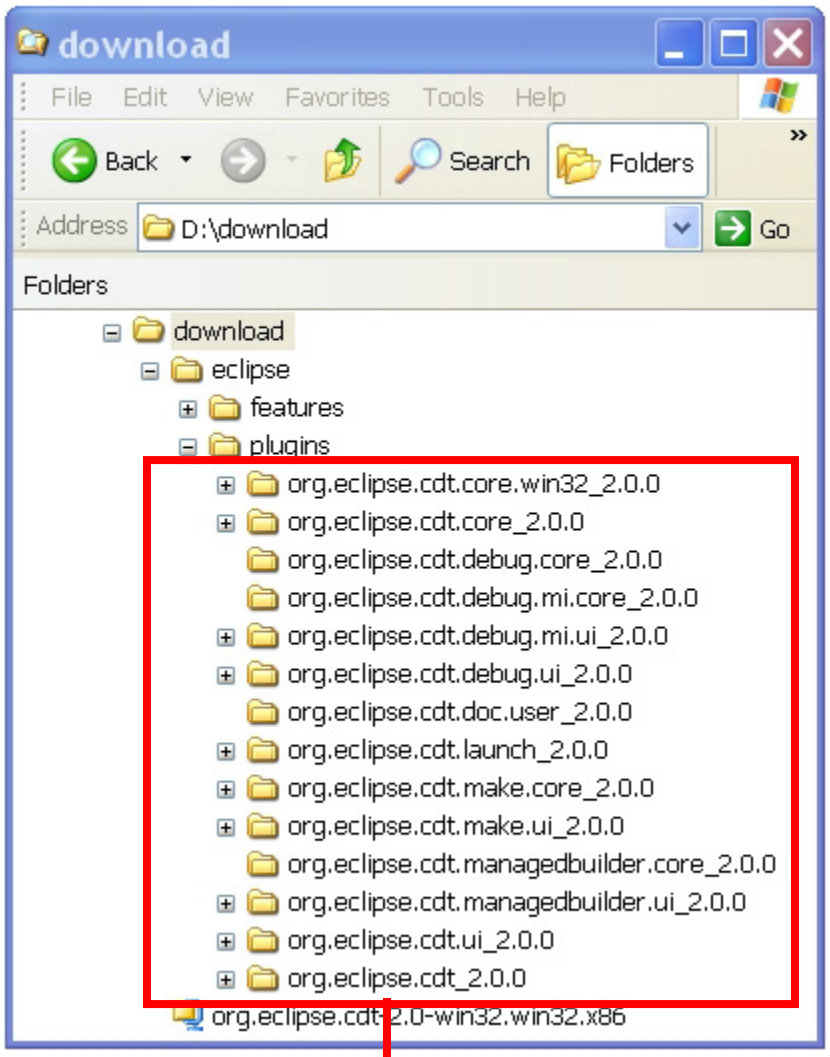


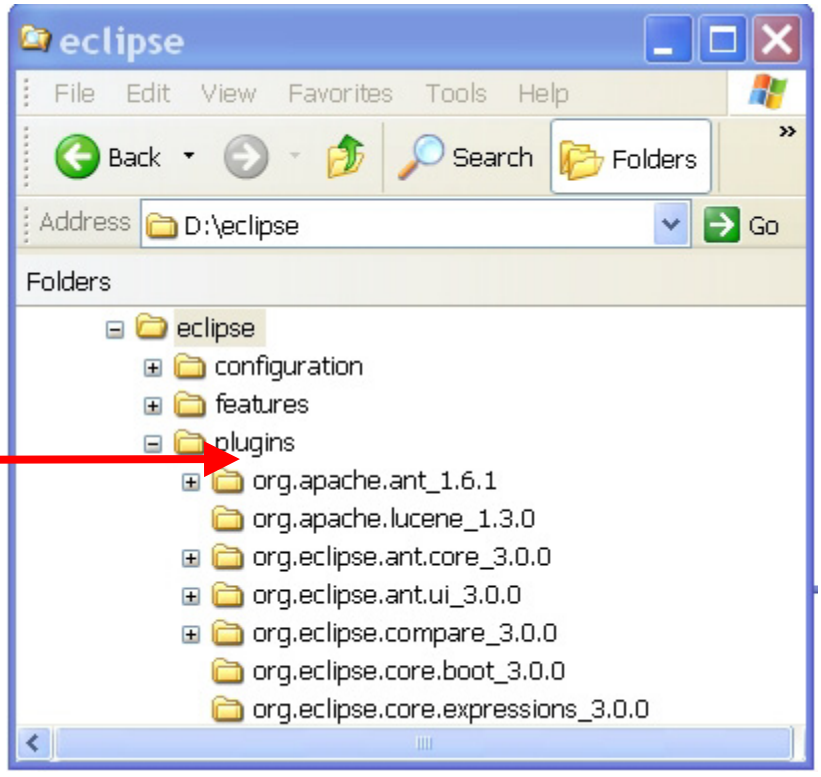
All we have to do to update **Eclipse** with the **CDT** plug-in is to copy the contents of the downloaded "**features**" folder to the same "**features**" folder in the Eclipse directory and do likewise for the "**plugins**" folders.



You can obviously do this with Windows Explorer and drag-and-drop techniques.

Likewise, copy **all** the folders in the **download\plugins** directory to the **eclipse plugins** directory.



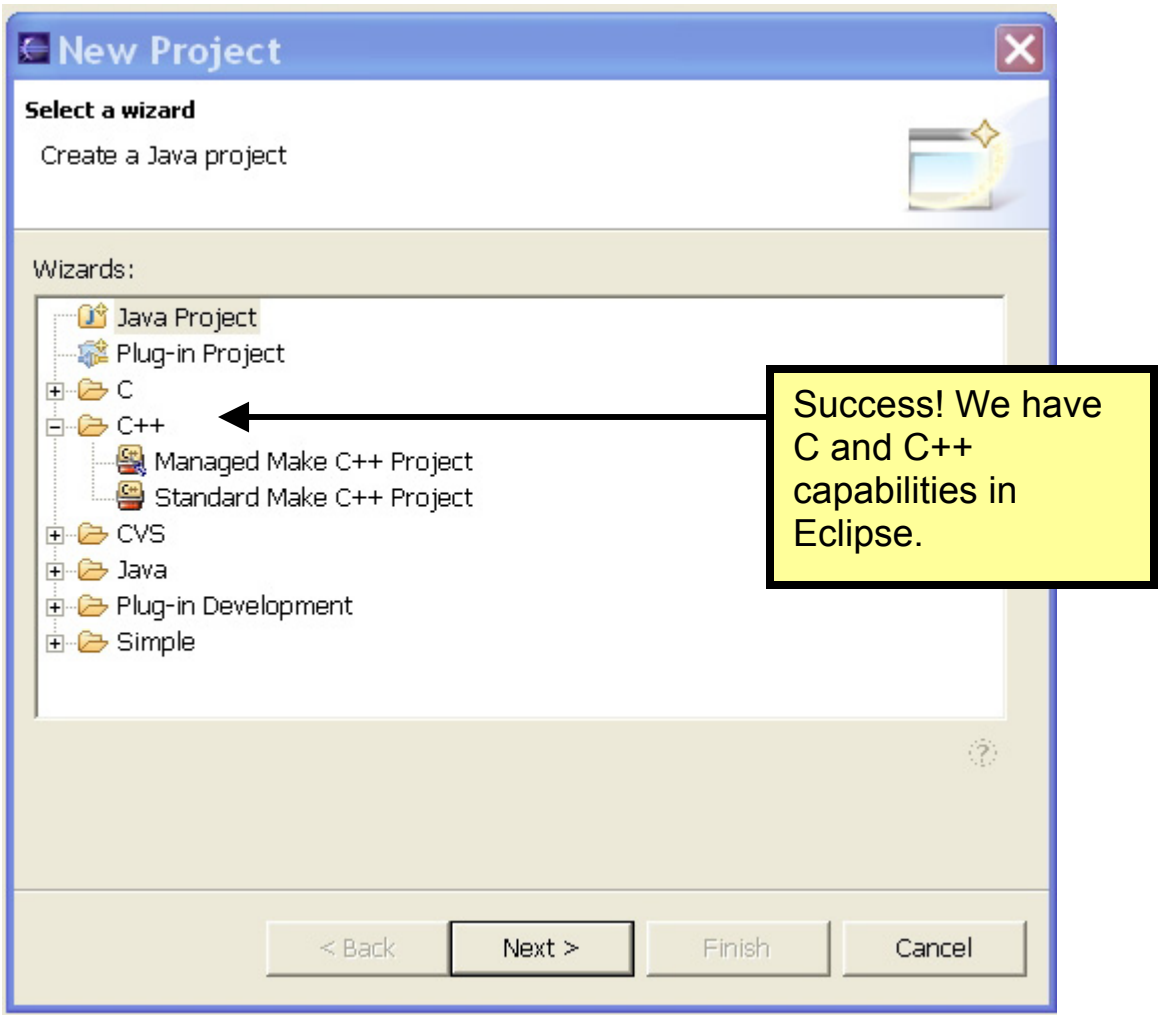


Copy these “plugins” folders from the download/plugins directory to the Eclipse plugins folder.

Now we can test if the Eclipse has been successfully updated with the CDT Plug-in. Before testing this, it behooves us to create a desktop icon that will run the Eclipse IDE.

The easy way to do this is to use Windows Explorer to find the Eclipse executable (application) in the Eclipse directory; right-click on it and select “Send to Desktop.” When finished, your desktop should have the follow icon placed. Clicking on it will start the Eclipse IDE.

Once Eclipse has started, select **File – New – Project** and you should see the following dialog. If the C and C++ project options are shown, then CDT has been successfully installed. Now exit from Eclipse and we’ll install the CYGWIN utilities.



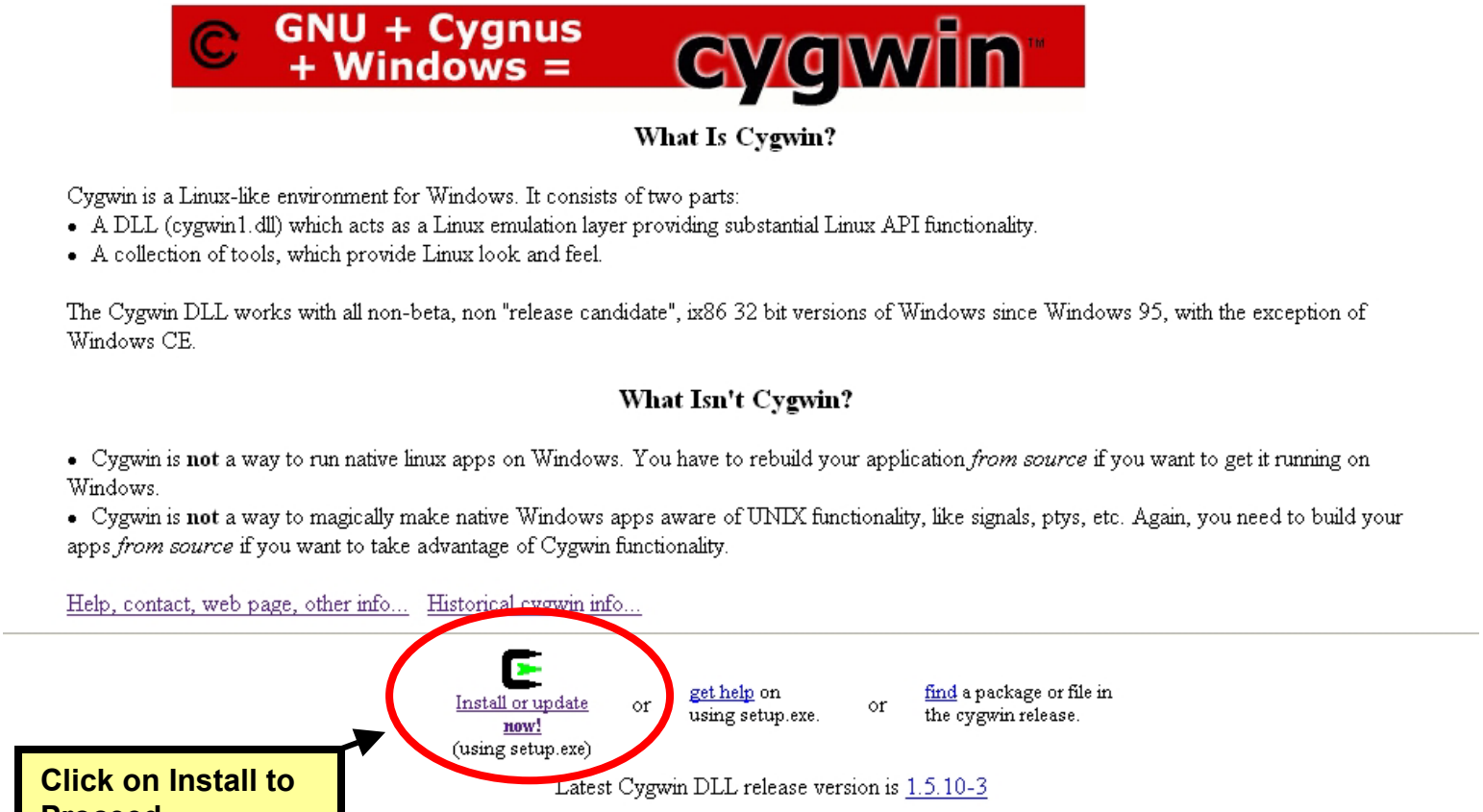
4. Downloading CYGWIN

To use the GNU compiler suite, we must first install the CYGWIN environment. Cygwin is a DLL which provides a UNIX emulation environment for Windows. The Cygwin environment also provides a complete port of such development utilities as gcc, binutils, gdb, make, etc., as well as a vast number of useful utilities. The CYGWIN download includes a GNU compiler suite for Windows/Intel Pentium targets. We will download later a GNU compiler suite specifically targeted for the ARM processor family and use that instead.

The Cygwin environment can be downloaded and installed from the internet. Click on the following link to proceed:

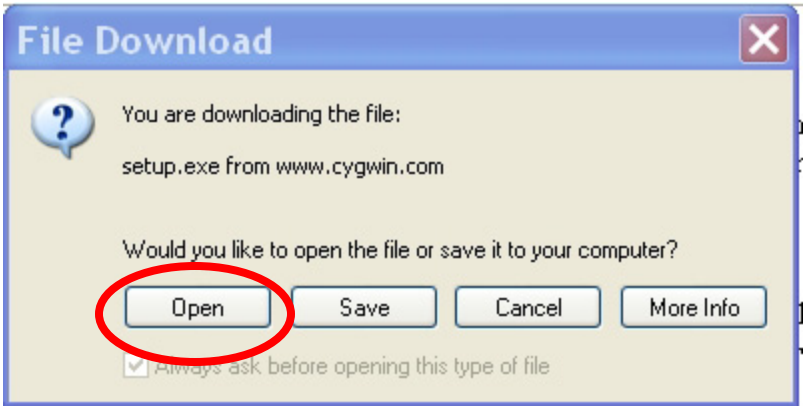
www.cygwin.com

This link leads to the CYGWIN home page, as shown below. Click on "**Install or Update New!**" to proceed.

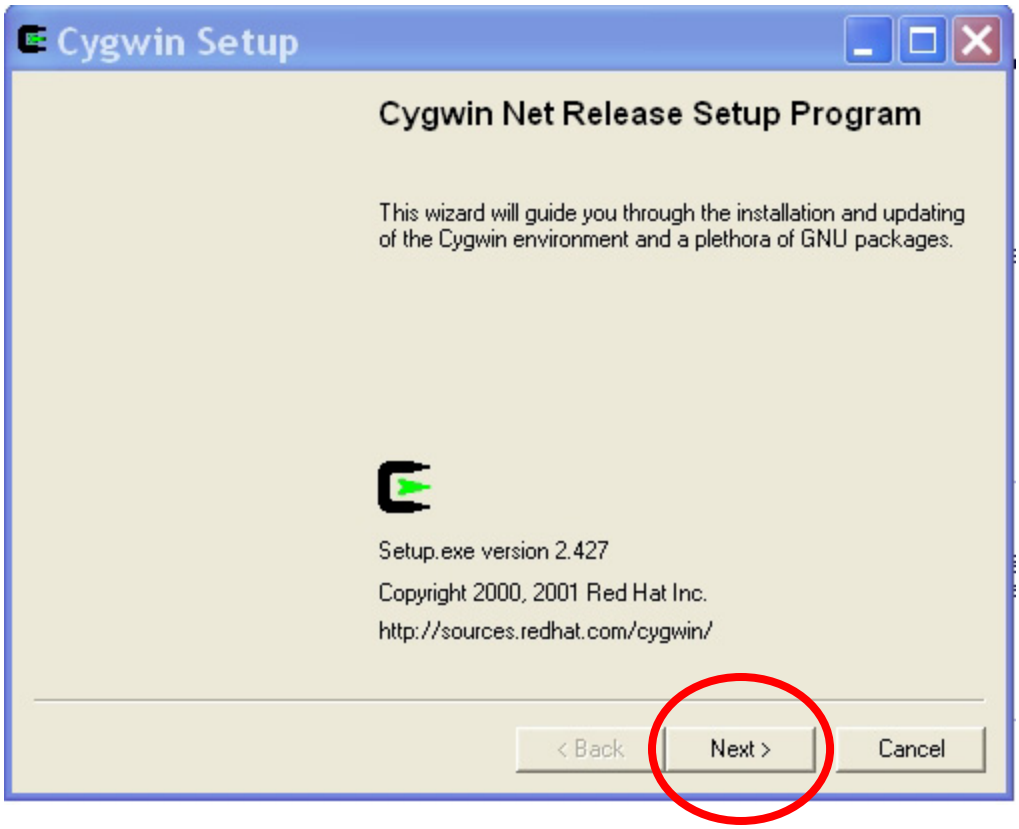


Click on Install to Proceed.

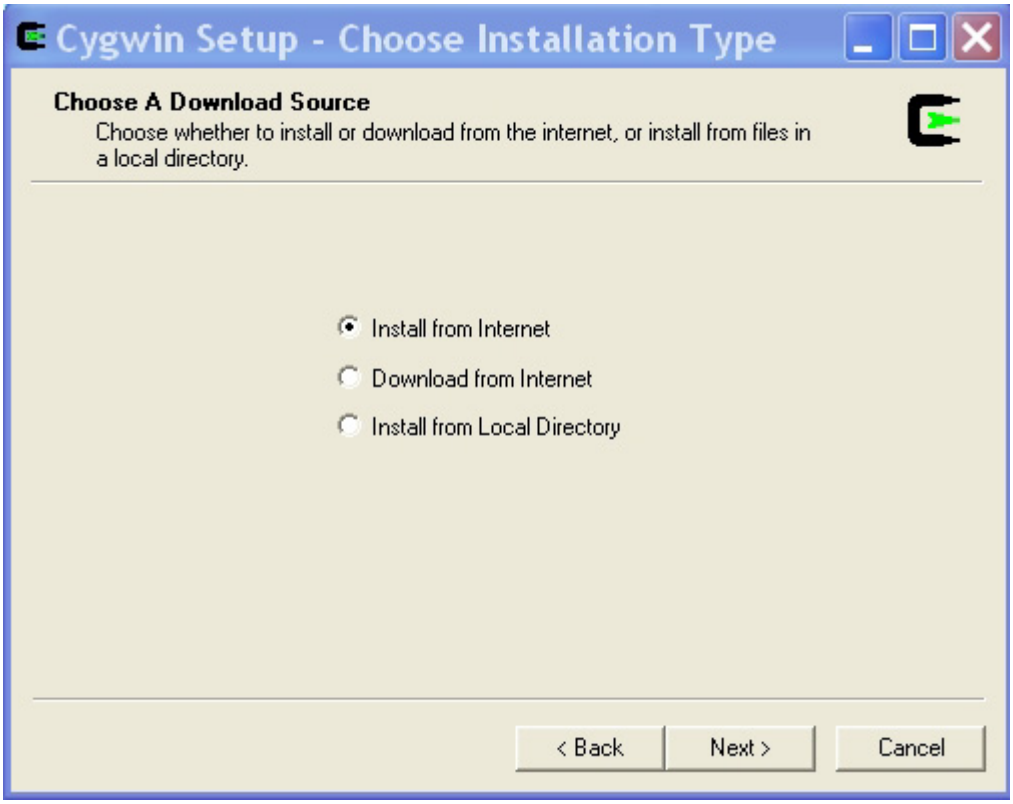
Now we will take the defaults on most of the following setup screens. In the screen below, select "**Open**" to allow the Cygwin web site to download and then install your Cygwin environment.



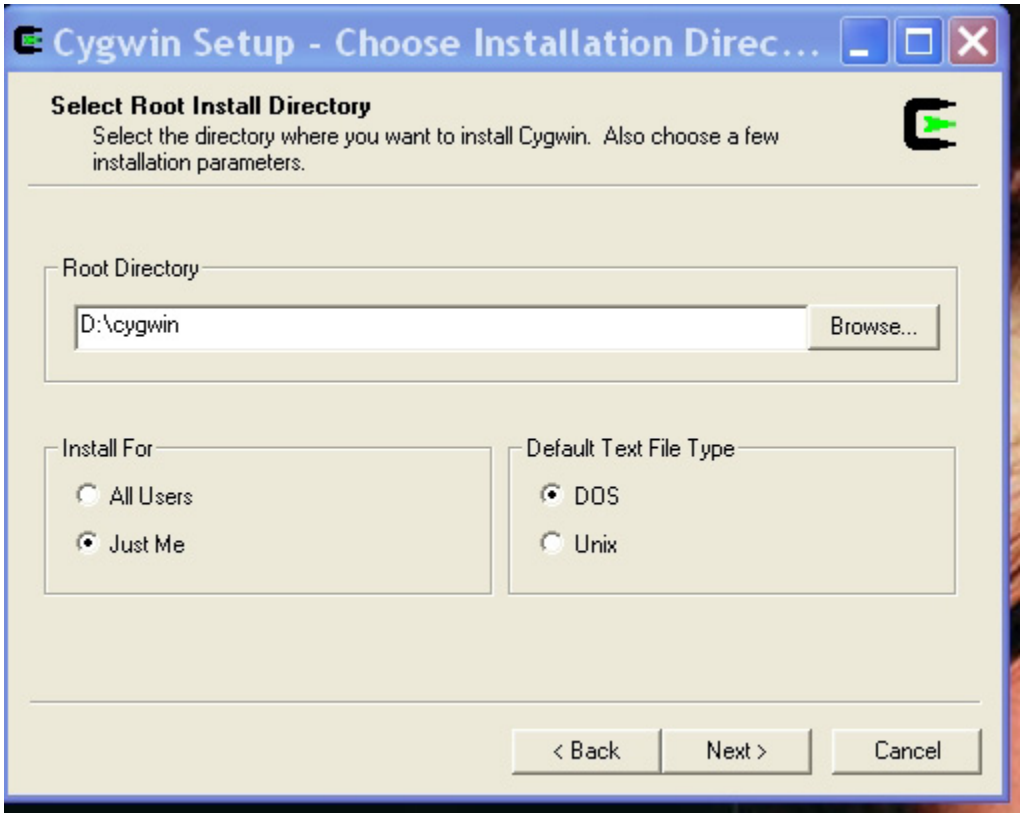
Now the Cygwin wizard will start up. Select "**Next**" to continue.



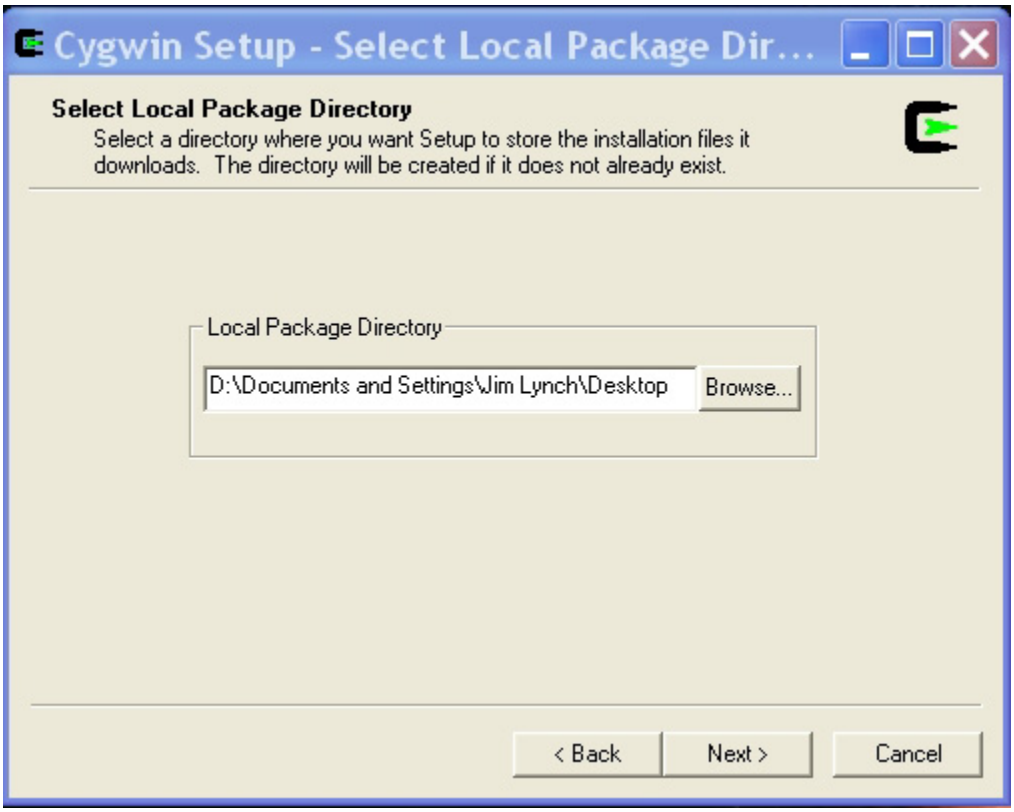
Choose “**Install from Internet**” and then click “**Next**.”



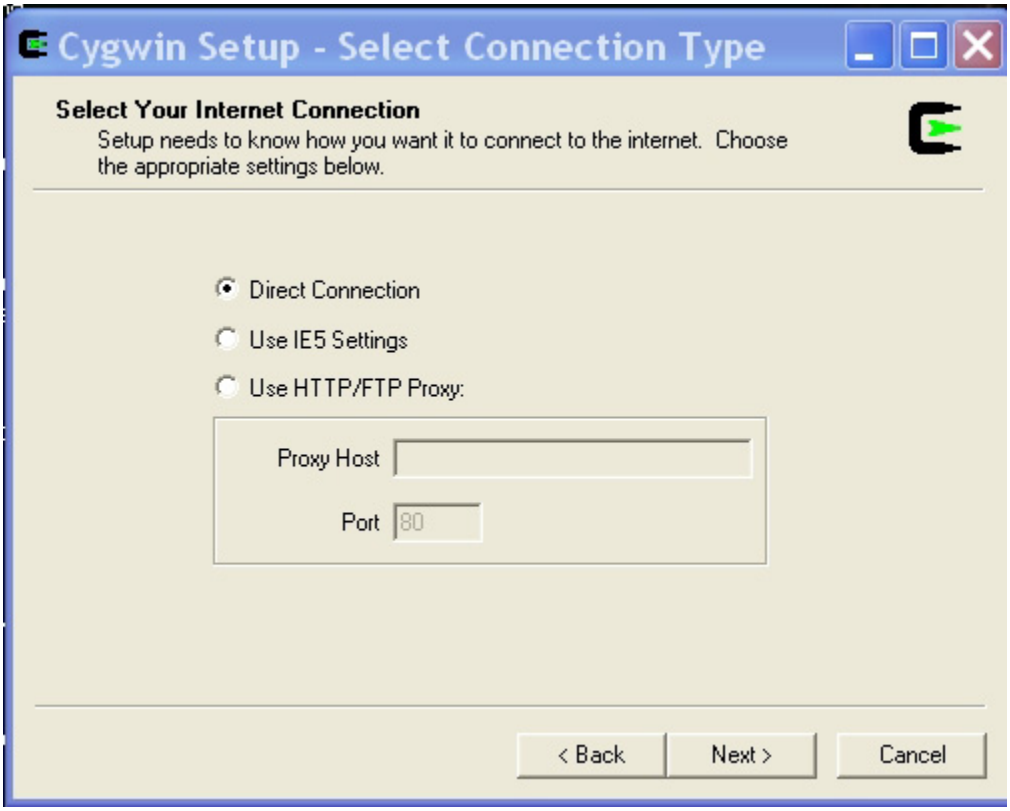
Now we specify the root directory where Cygwin will be installed (**d:\cygwin**). Click “**next**” to continue.



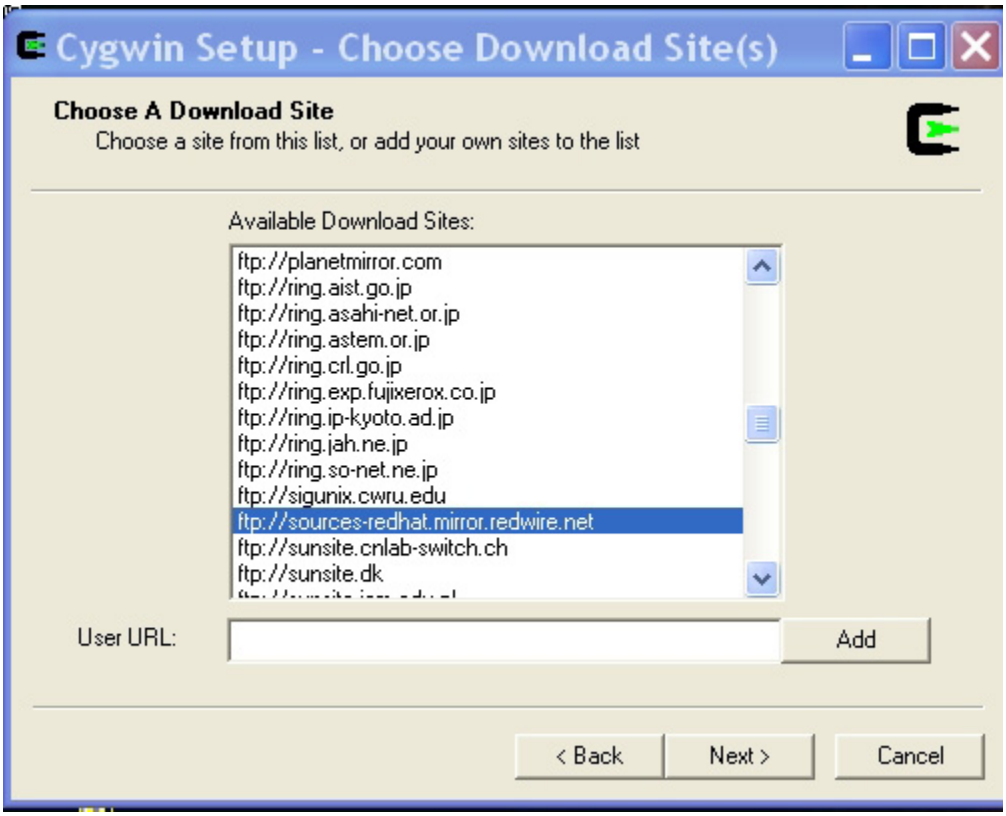
We can accept the default here since this specifies a temporary directory for downloading operations (actually on the desktop, you'll need to remove these from your desktop later). Click "**Next**" to continue.



Here I selected "**Direct Connection**" since I have a cable modem. If you have a dial-up, then "**Use IE5 Settings**" would be the proper choice. Click "**Next**" to proceed.



Here we are asked to select a download mirror site. As always, this is a bit of a gamble. Pick one, pray and click "**Next**" to proceed.



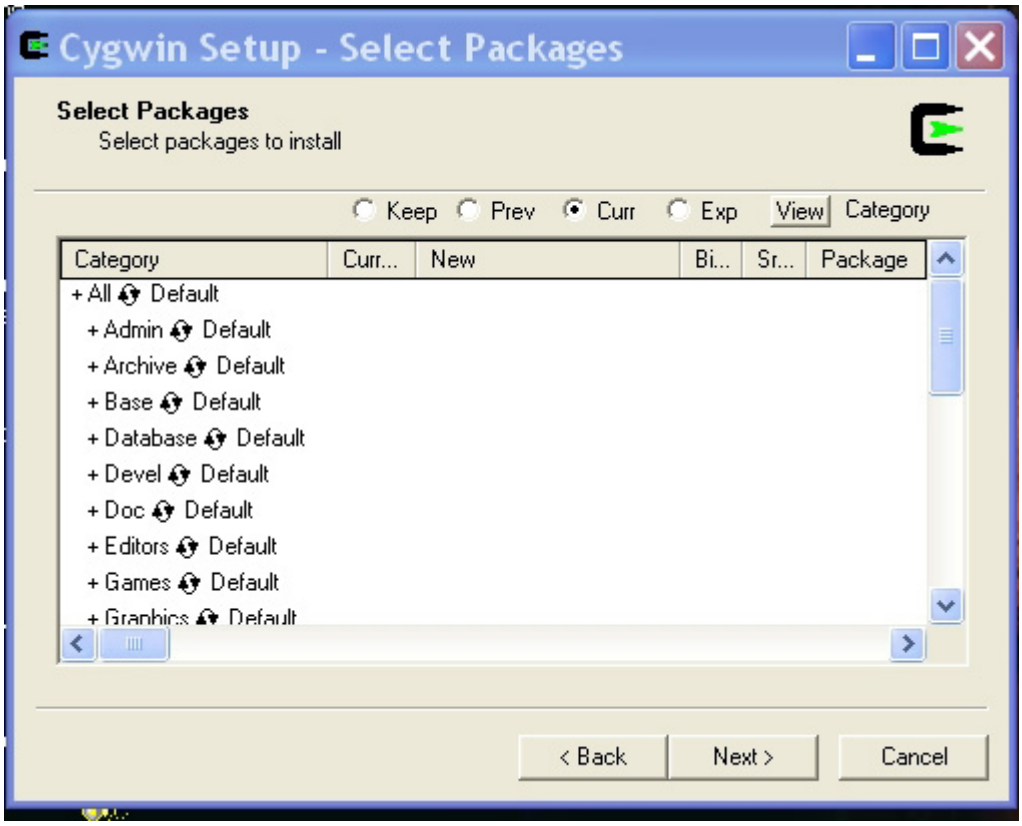
The next screen allows you to specify what GNU packages you wish to install.

Basically, we want the default installation that will allow us to compile for the Windows XP / Intel platform. This will allow us to use Eclipse to build Windows applications (not covered in this document).

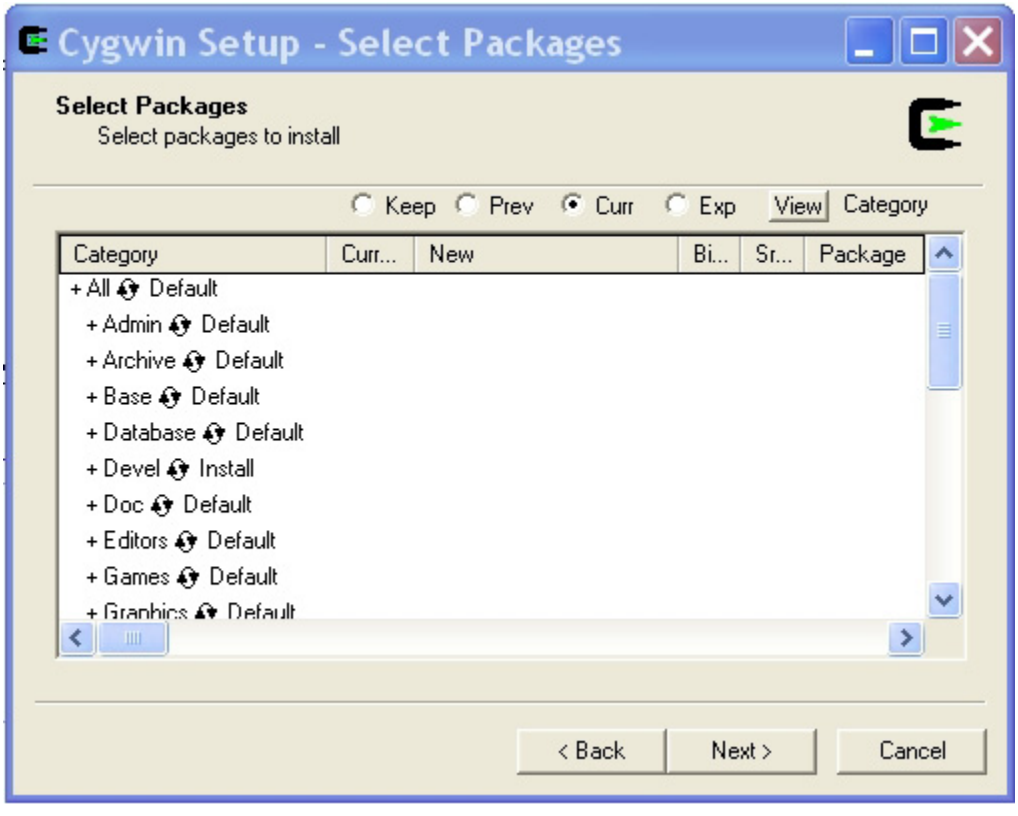
If you look at the Cygwin "Select Packages" screen below, you'll see the following line.

You must click on the little circle with the two arrowheads until the line changes to this:

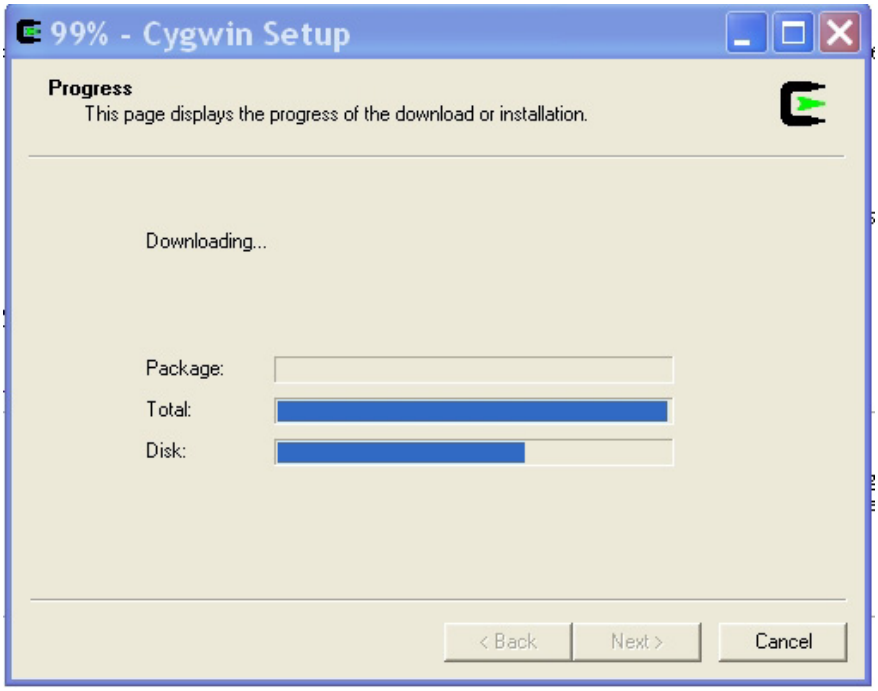
This will force installation of the default GNU compiler suite for Windows/Intel targets. Here's the "**Select Packages**" screen before clicking on the circle with arrowheads.



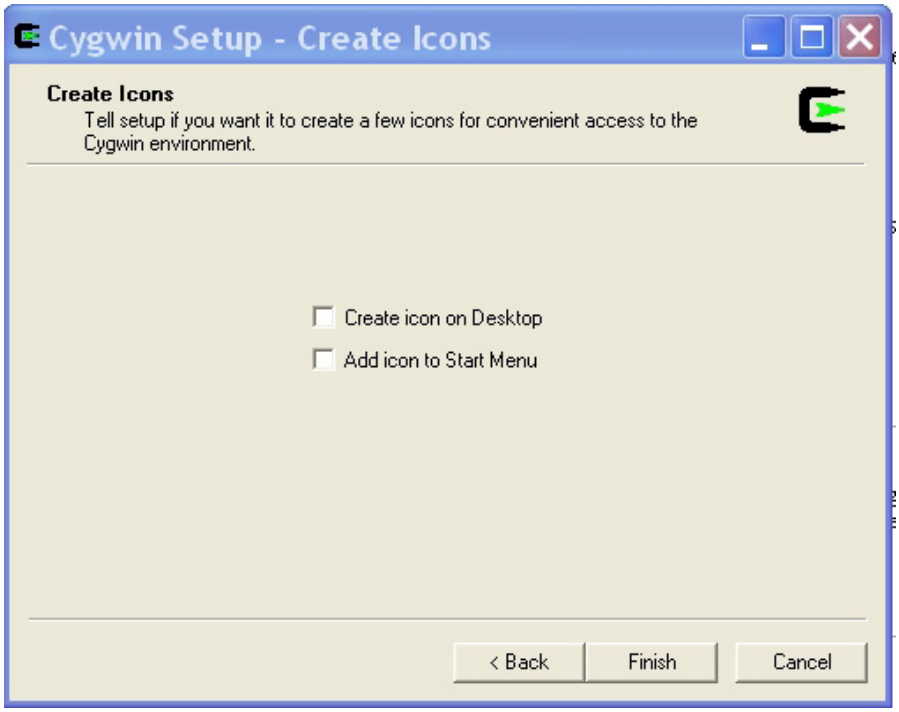
Here's the "Select Packages" screen after changing from **Devel – Default** to **Devel – Install.**



Now the Cygwin will start downloading. This creates a huge 700 Megabyte directory on your hard drive and takes 30 minutes to download and install using a cable modem.



When the installation completes, Cygwin will ask you if you want any desktop icons and start menu entries made. Say "**No**" to both. These icons allow you to bring up the BASH shell emulator (like the command prompt window in Windows XP). This would allow you do some Linux operations, but this capability is not necessary for our purposes here.

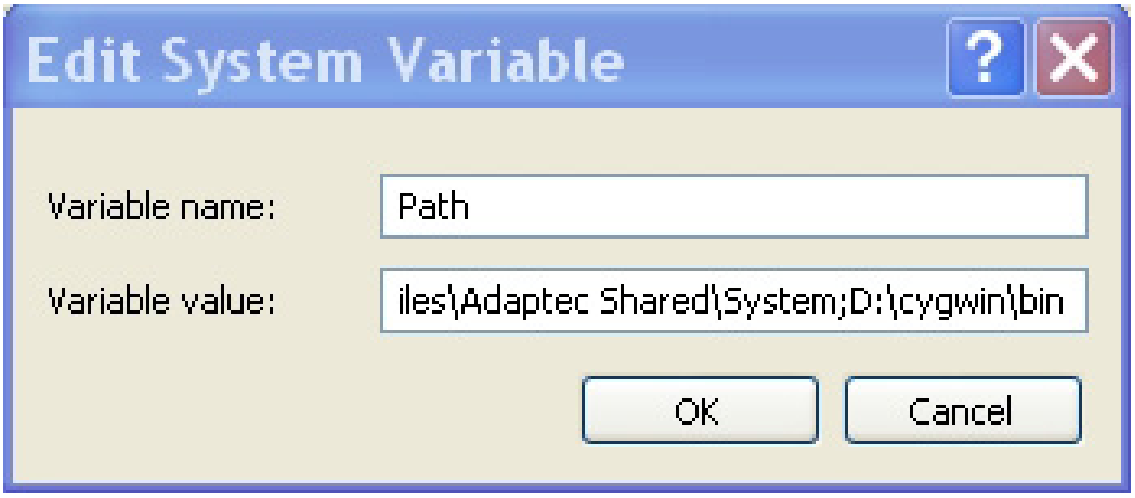


Now the Cygwin installation manager completes and shows the following result.

The directory **d:\cygwin\bin** must be added to the **Windows XP** path environment variable. This allows Eclipse to easily find the Make utility, etc.

Using the **Start Menu**, go to the **Control Panel** and click on the "**System**" icon.

Then click on the "**Advanced**" tab and select the "**Environment Variables**" icon. Highlight the "**Path**" line and hit the "**Edit**" button. Add the addition to the path as shown in the dialog box shown below (don't forget the semicolon separator). The Cygwin FAQ advises putting this path specification before all the others, but it worked for me sitting at the end of the list.



We are now finished with the CYGWIN installation. It runs silently in the background and you should never have to think about it again.

5. Downloading the GNUARM Compiler Suite

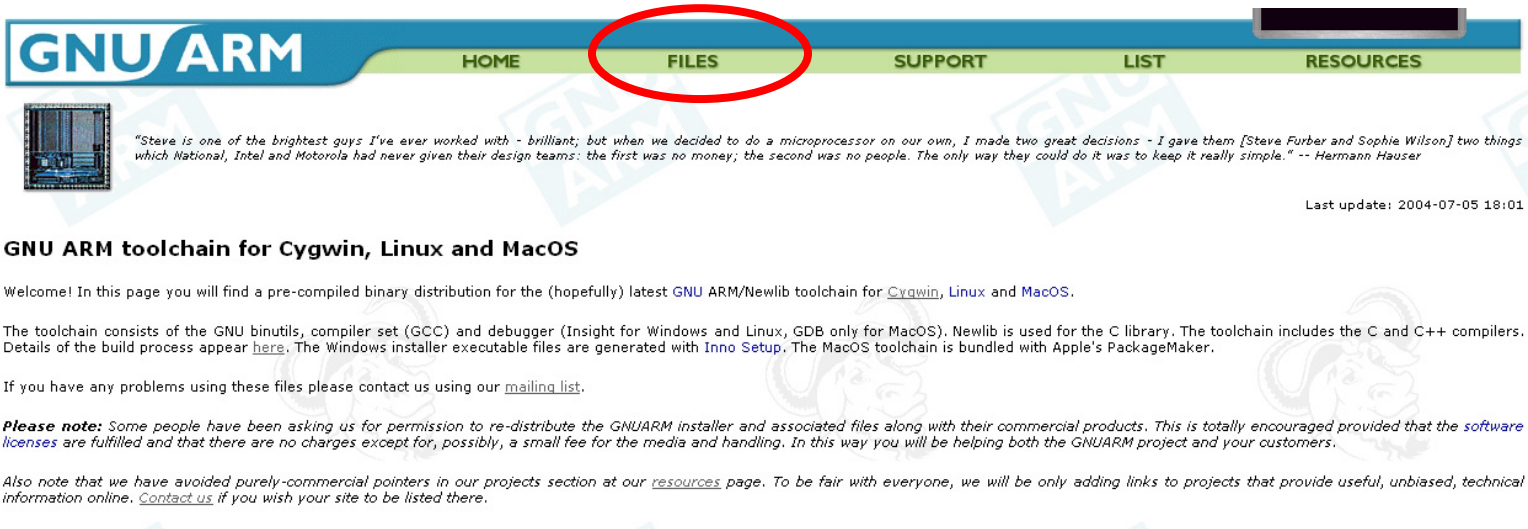
At this point, we have all the GNU tools needed to compile and link software for Windows/Intel computers. It is possible to use all this to build a custom GNU compiler suite for the ARM processor family. The very informative book "**Embedded System Design on a Shoestring**" by Lewin A.R.W. Edwards ©2003 describes how to do this and it is rather involved.

Fortunately, Pablo Bleyer Kocik and the people at **gnuarm.com** have come to the rescue with pre-built GNU compiler suite for the ARM processors. Just download it with the included installer and you're ready to go.

Click on the following link to download the GNUARM package.

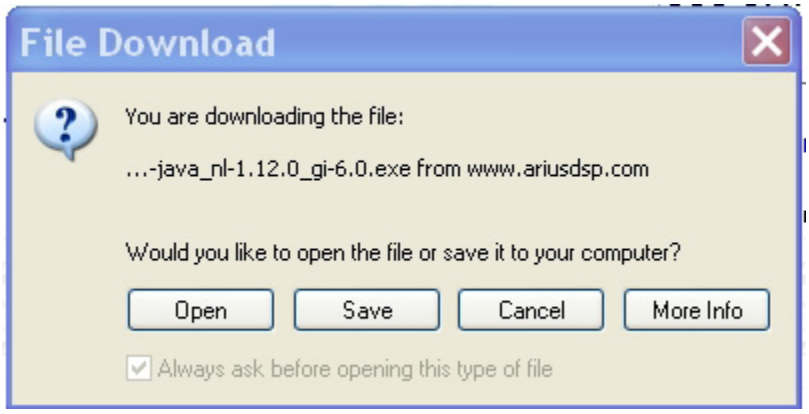
www.gnuarm.com

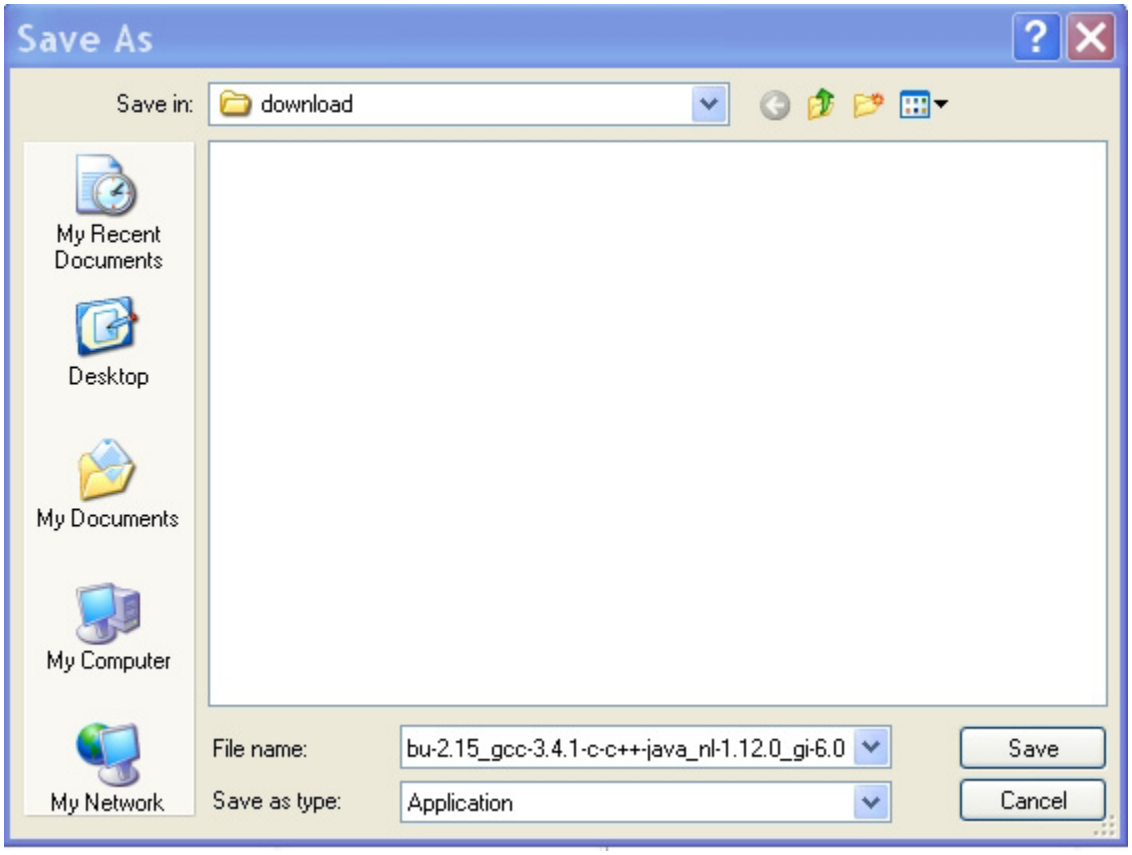
The GNUARM web site will display and you should click on the "**Files**" tab.



The correct package to download is **Binaries Cygwin– GCC-3.4 toolchain – GCC-3.4.1**

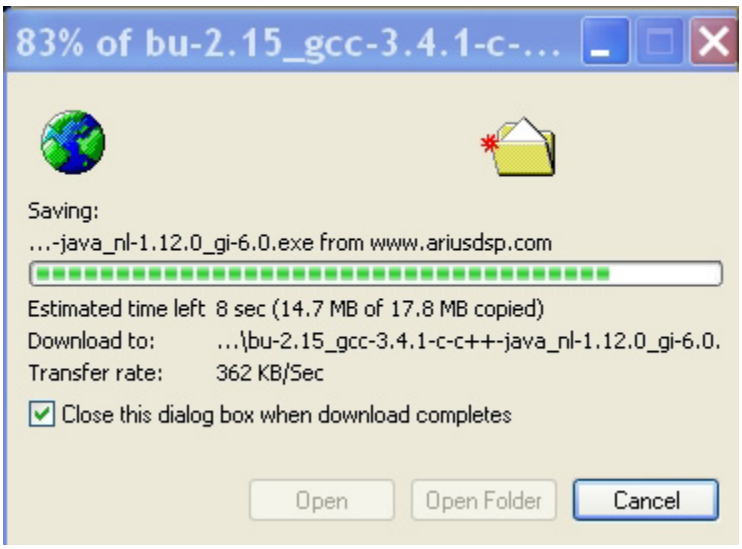
Just like all the other downloads we've done, we direct this one to our empty download directory on the hard drive. Here we click "**Save**" and then specify the download destination.





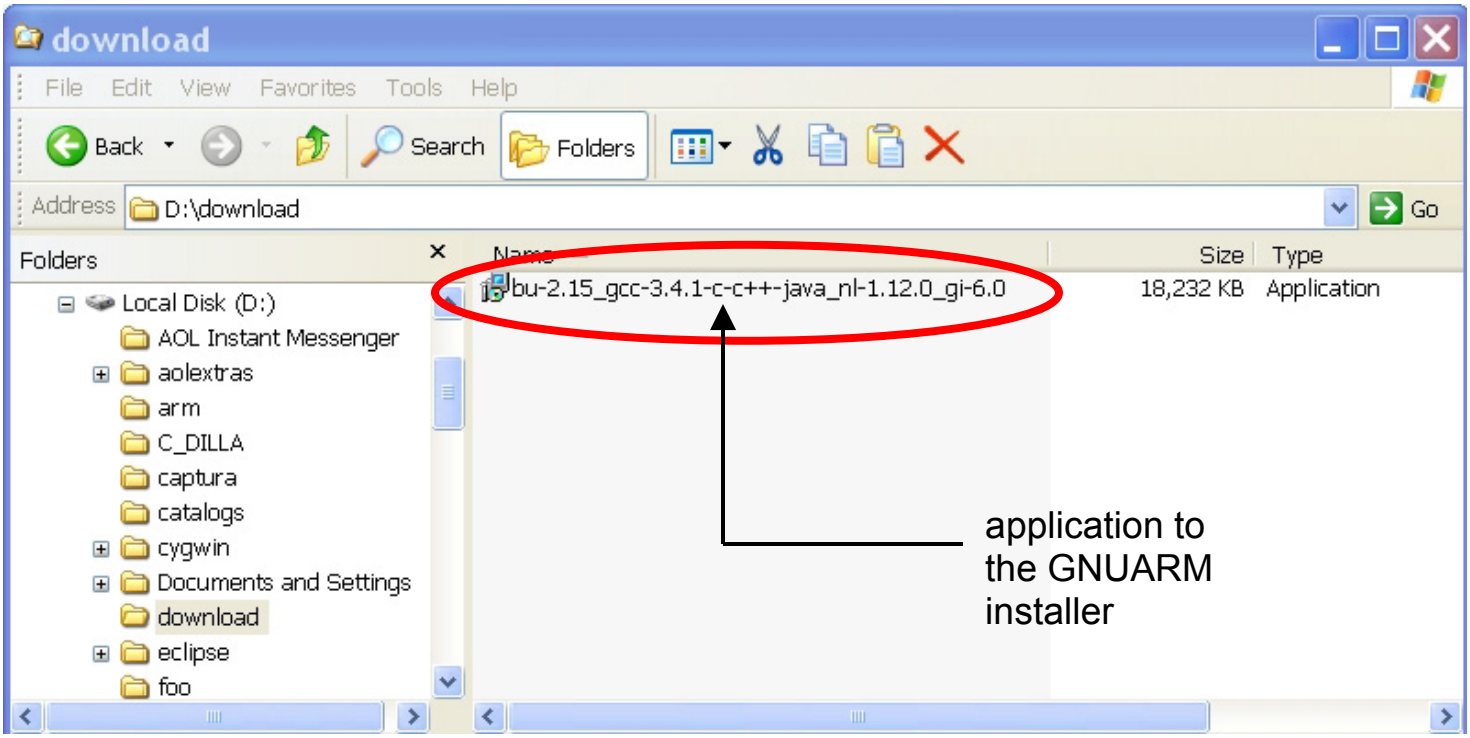
As you can see, this download has a very long name.

This download is a 18 megabyte file and takes 30 seconds on a cable modem.

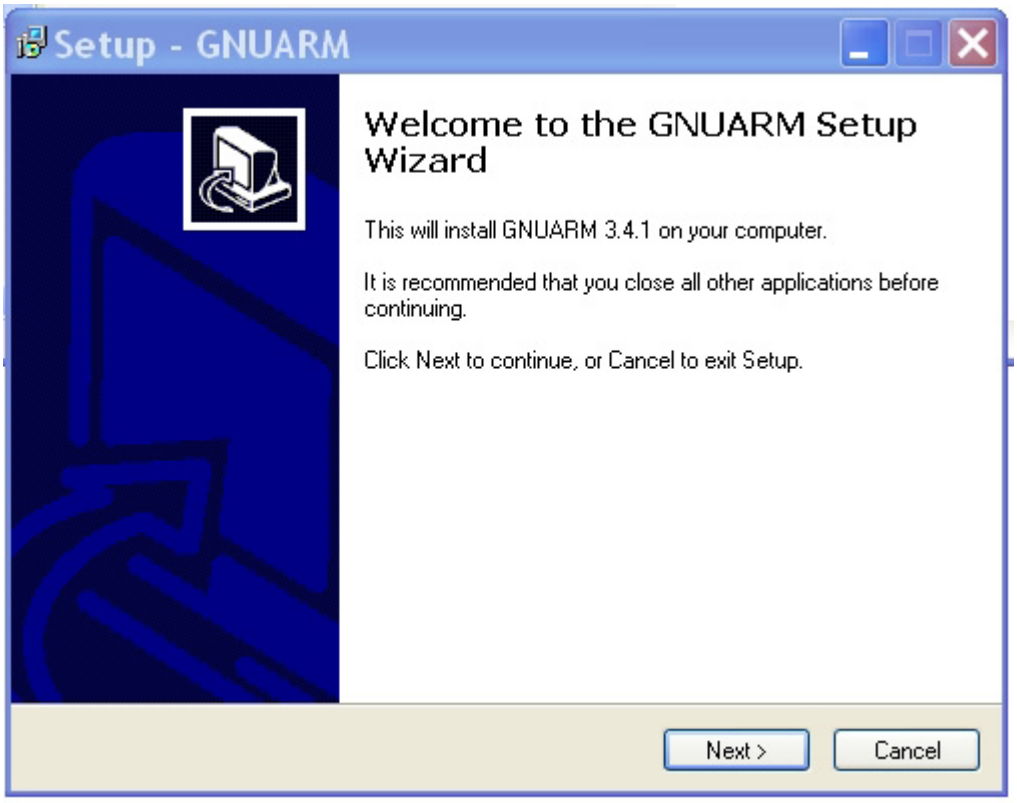


The download directory now has the following setup application with the following unintelligible filename: **bu-2.15_gcc-3.4.1-c-c++-java_nl-1.12.0_gi-6.0.exe**

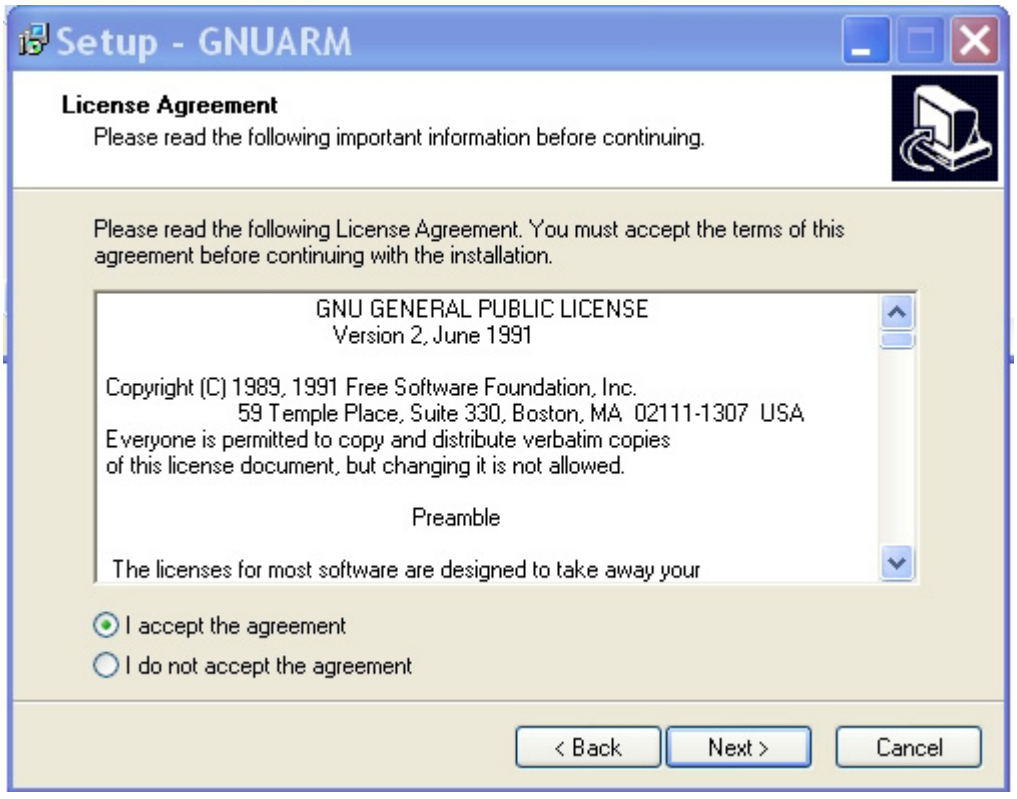
Click on that filename to start the installer.



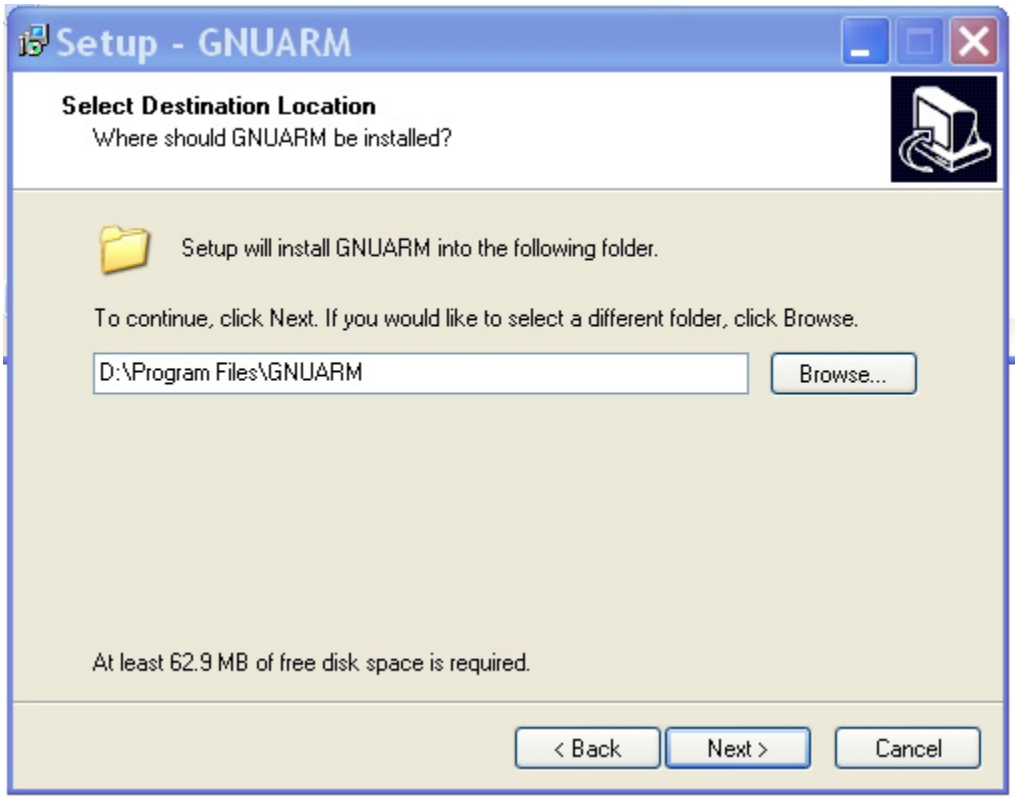
The GNUARM installer will now start. Click **"Next"** to continue.



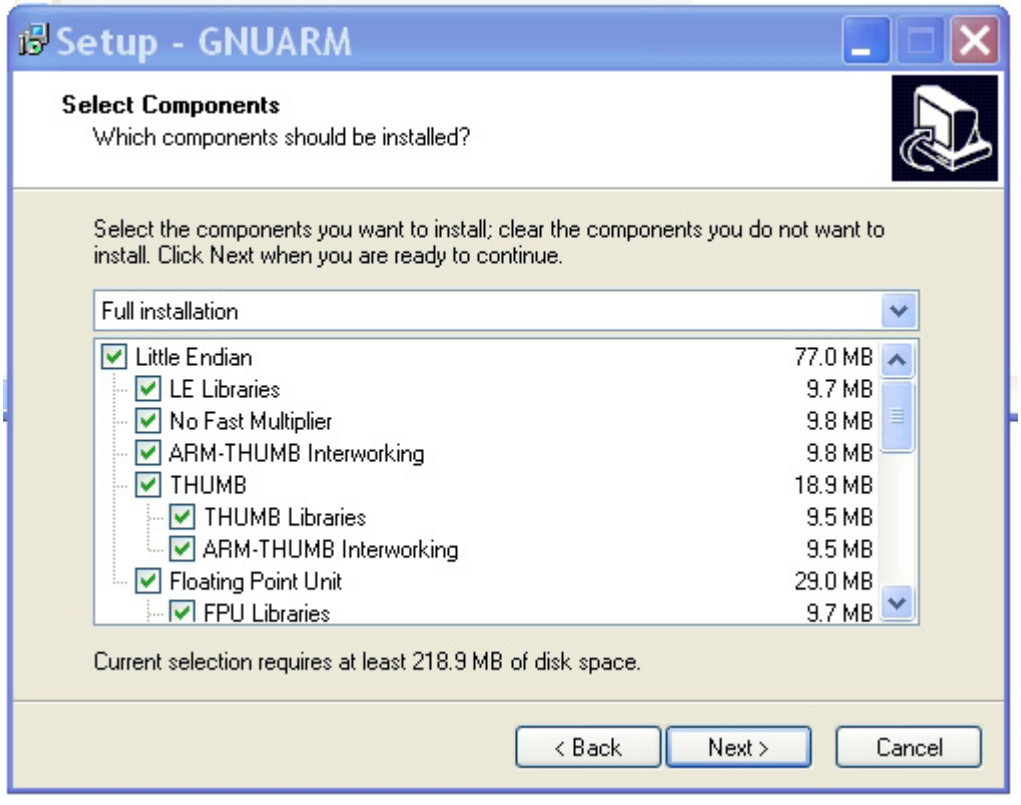
Accept the GNU license agreement – don't worry, it's still free. Click **"Next"** to continue.



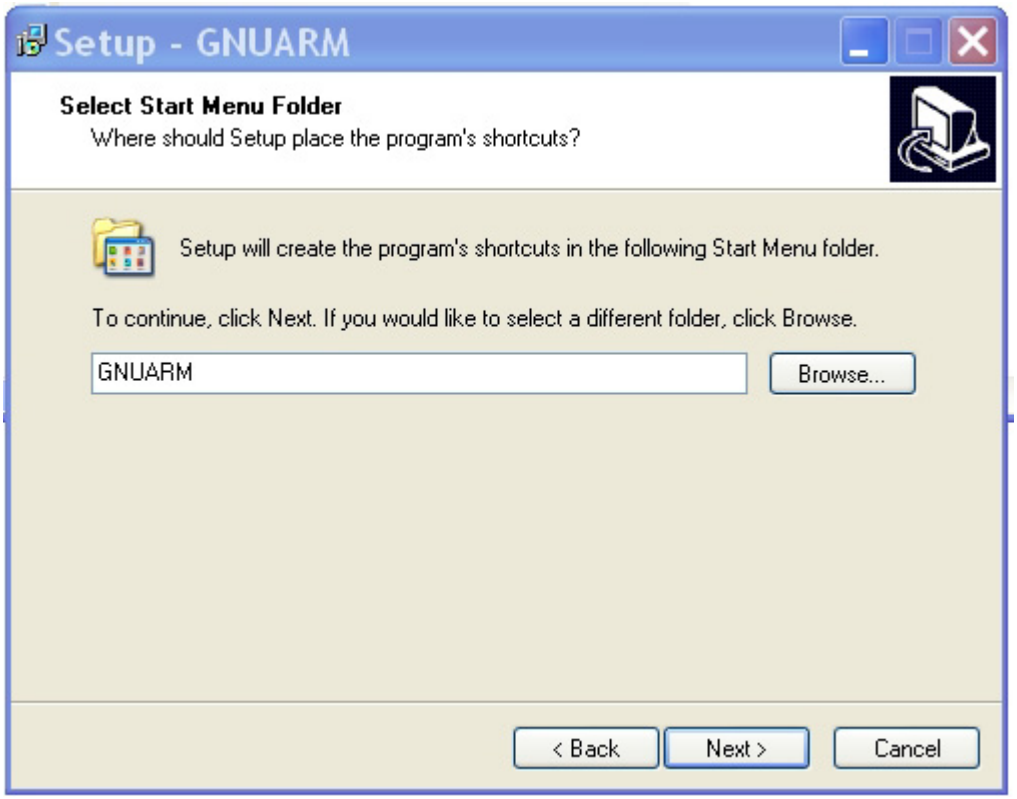
We'll take the default and let it install into the "**Program Files**" directory. Click **"Next"** to continue.



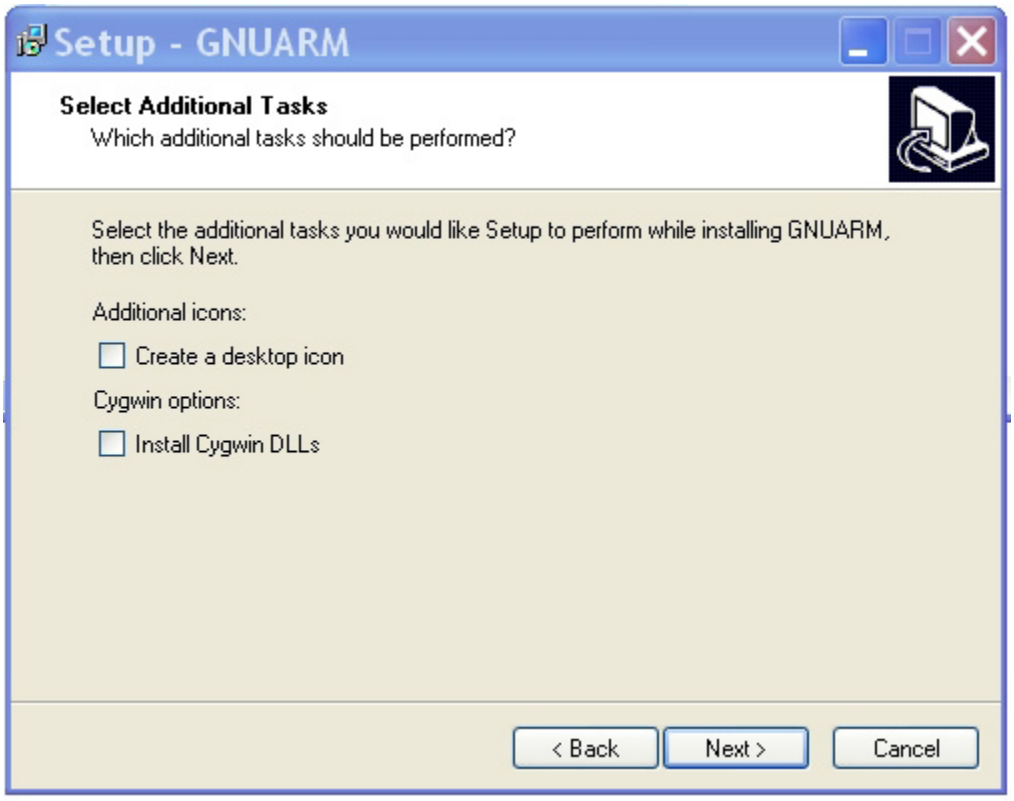
We'll also take the defaults on the "Select Components" window. Click **"Next"** to continue.



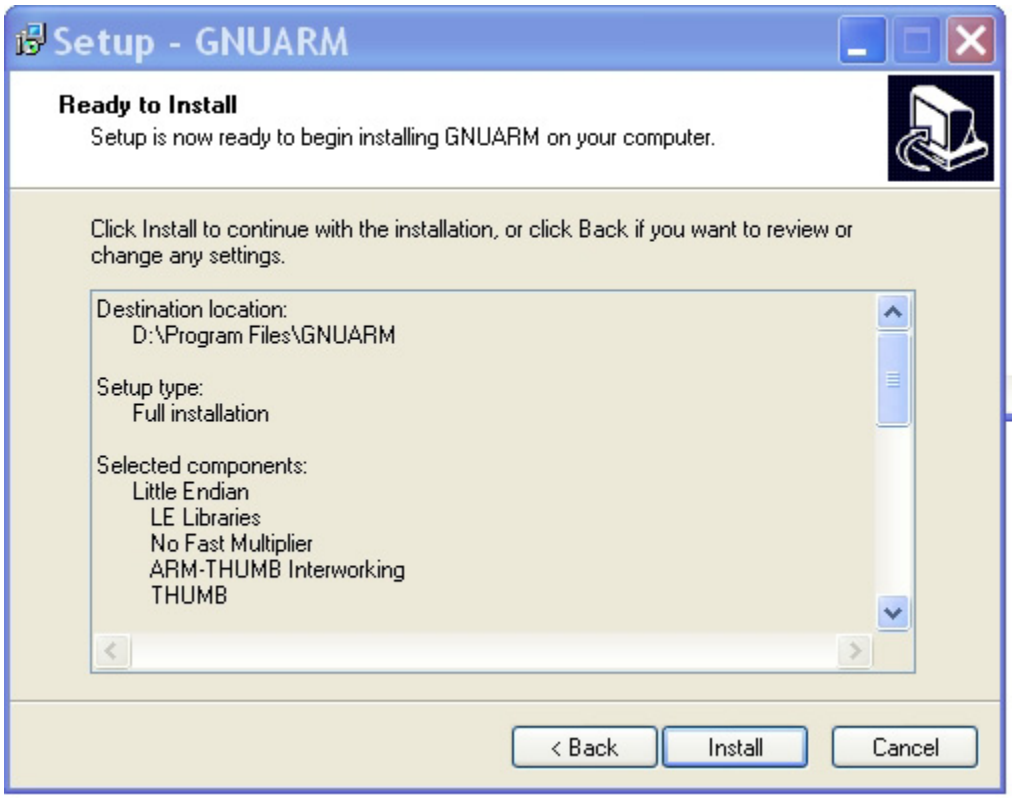
Take the default on this screen. Click **"Next"** to continue.



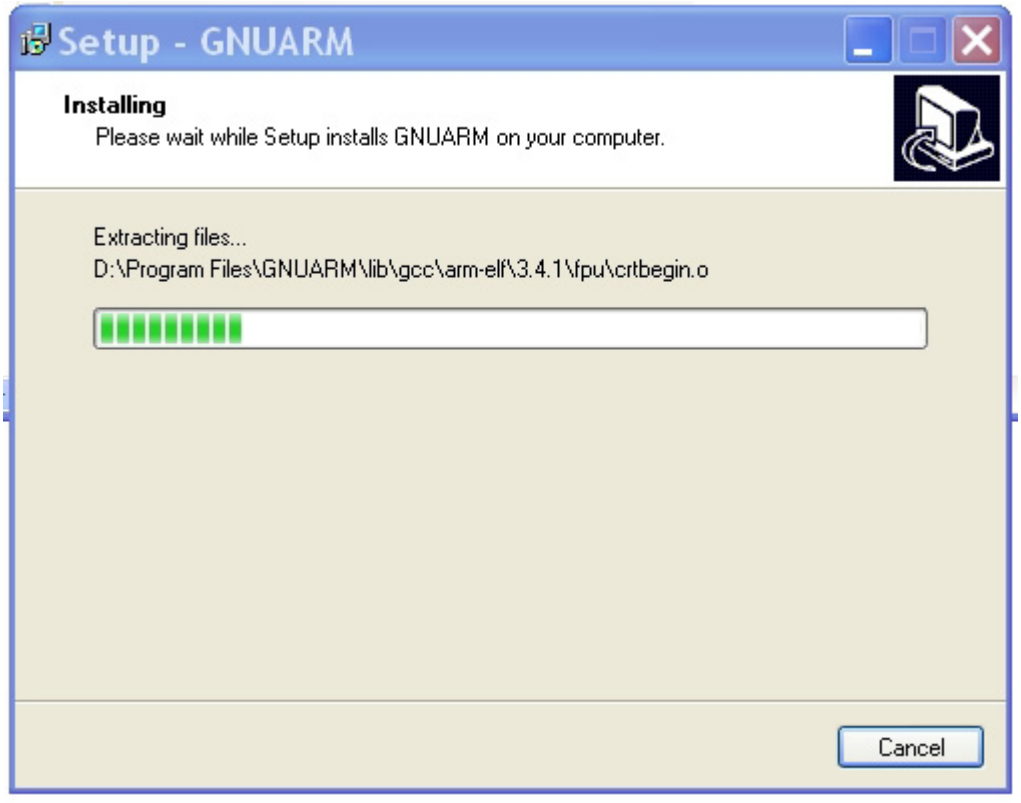
It's very important that you <u>don't check</u> **"Install Cygwin DLLs**". We already have the Cygwin DLL installed from our Cygwin environment installation. Since all operations are called from within Eclipse, we don't need a "**desktop icon**" either. Click **"Next"** to continue.



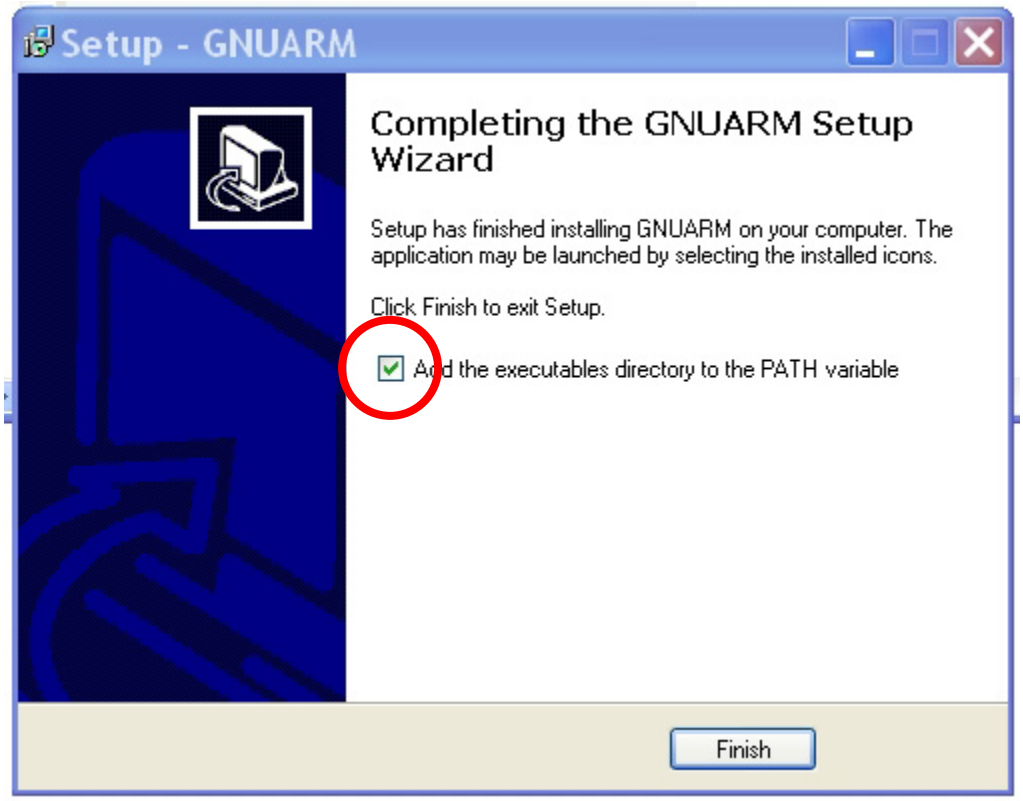
Click on “**Install**” to start the GNUARM installation.



Sit back and watch the GNUARM compiler suite install itself.



When it completes, the following screen is presented. Make sure that "**Add the executables directory to the PATH variable**" is checked. This is crucial.



This completes the installation of the compiler suites. Since Eclipse will call these components via the make file, you won't have to think about it again.

It's worth mentioning that the GNUARM web site has a nice Yahoo user group with other users posing and answering questions about GNUARM. Pay them a visit. The GNUARM web site also has links to all the ARM documentation you'll ever need.

6. Installing the Philips LPC2000 Flash Utility into Eclipse

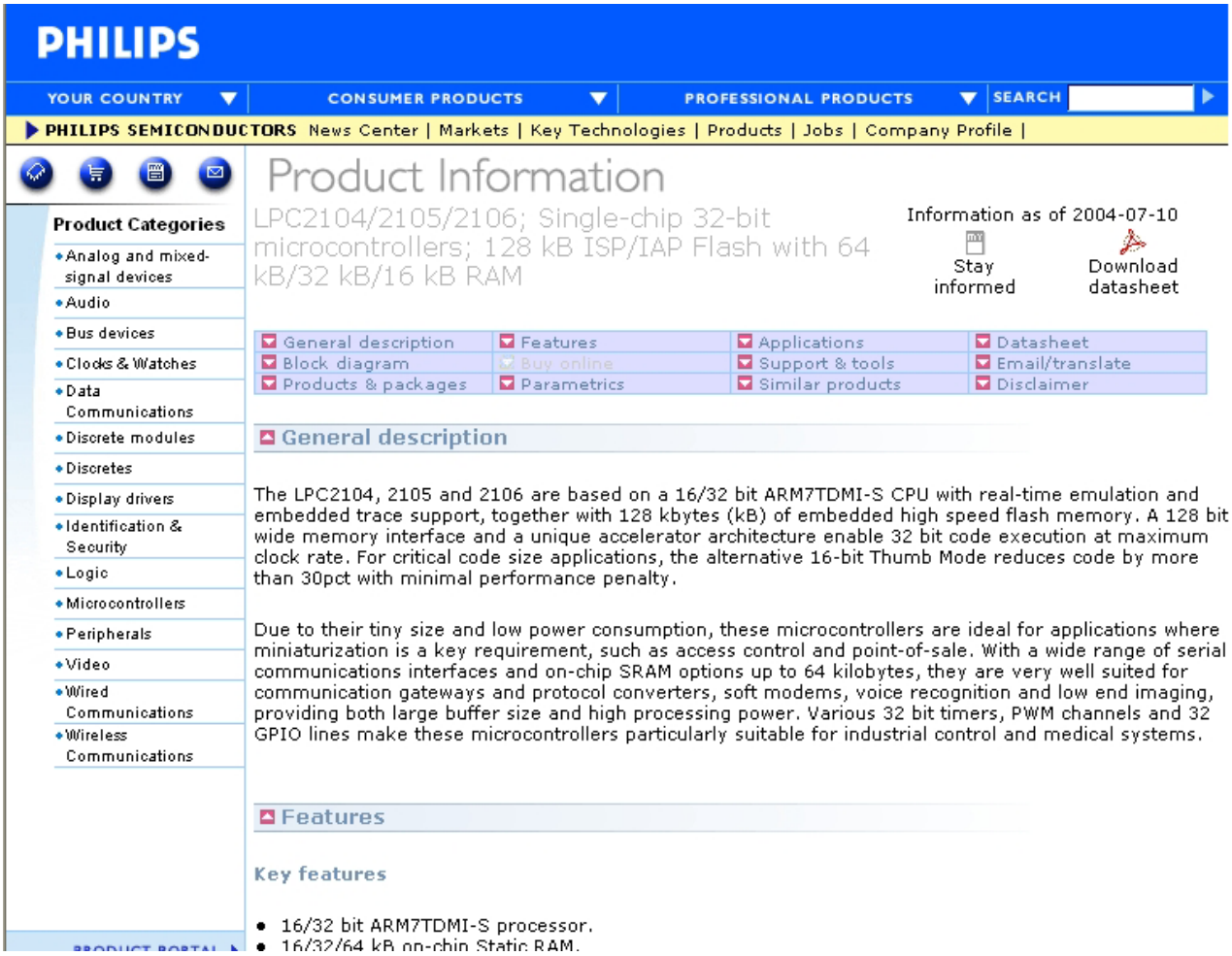
The Philips LPC2000 Flash Utility allows downloading of hex files from the COM1 port of the desktop computer to the TiniARM board's flash (or RAM) memory.

We need to download the latest version of this program from the Philips web site and unzip and install it into the **program files** directory. Then we will start Eclipse and add the LPC2000 Flash Utility as an external command to be invoked.

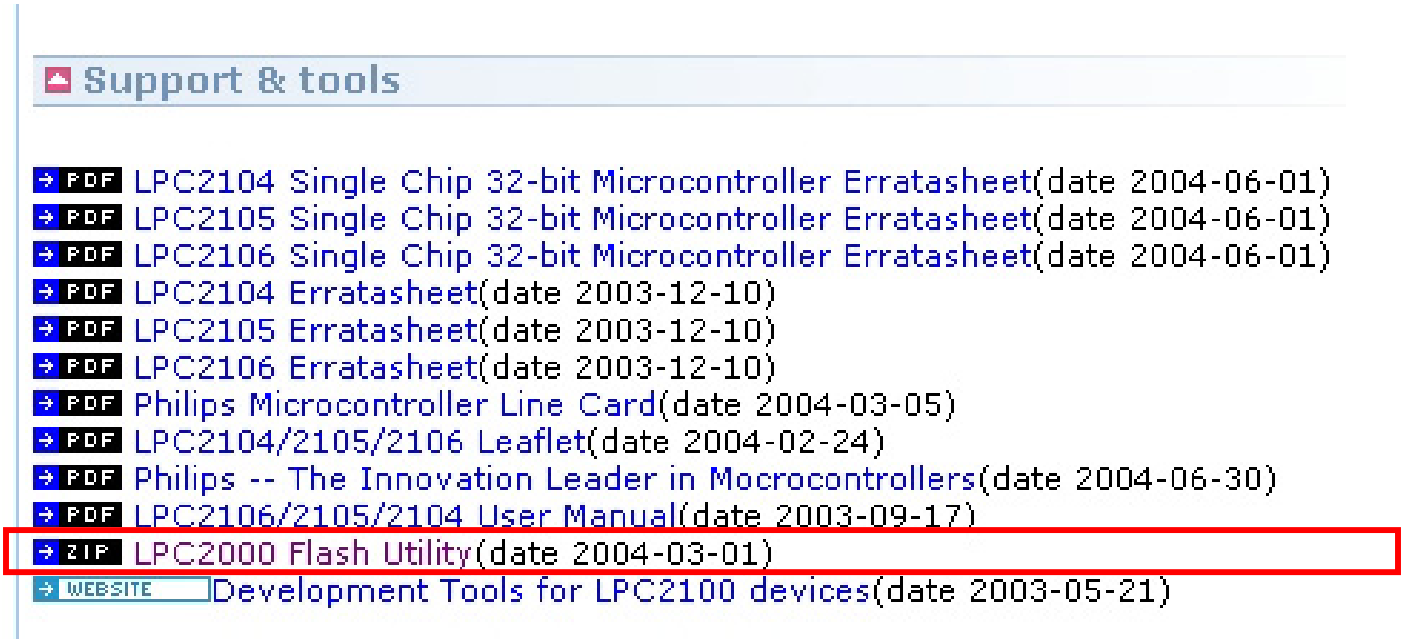
Click on the following link to access the Philips LPC2106 web page.

www.semiconductors.philips.com/pip/LPC2106.html

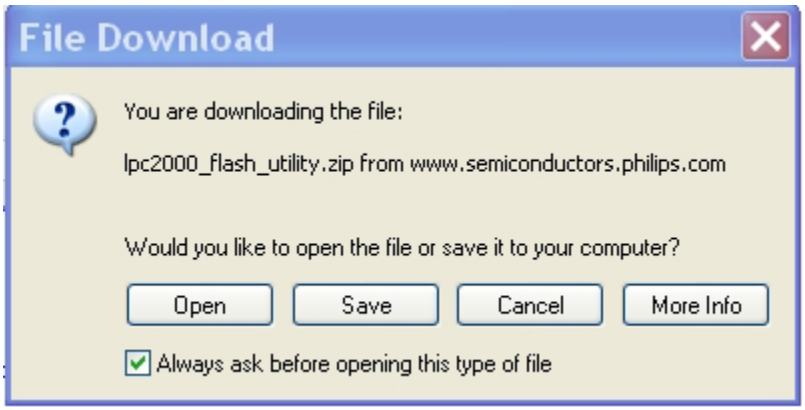
The following web page for the LPC2106 should open.

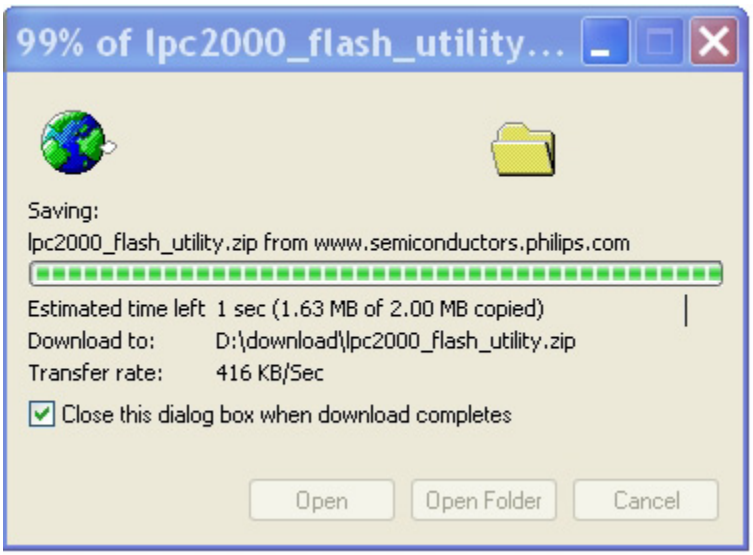


If you scroll down this page, you will see a link to the LPC2000 Flash Utility download. Click on the ZIP file LPC2000 Flash Utility (date 2004-03-01)

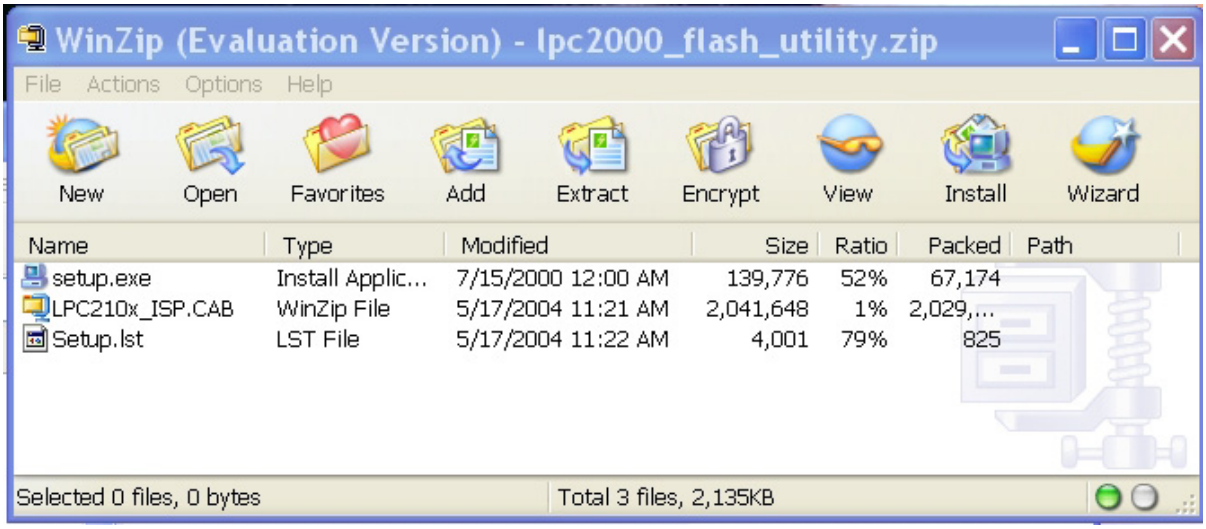


As before, we'll save the downloaded zip file in our empty download directory. This is a fairly short download, only about 2 megabytes.

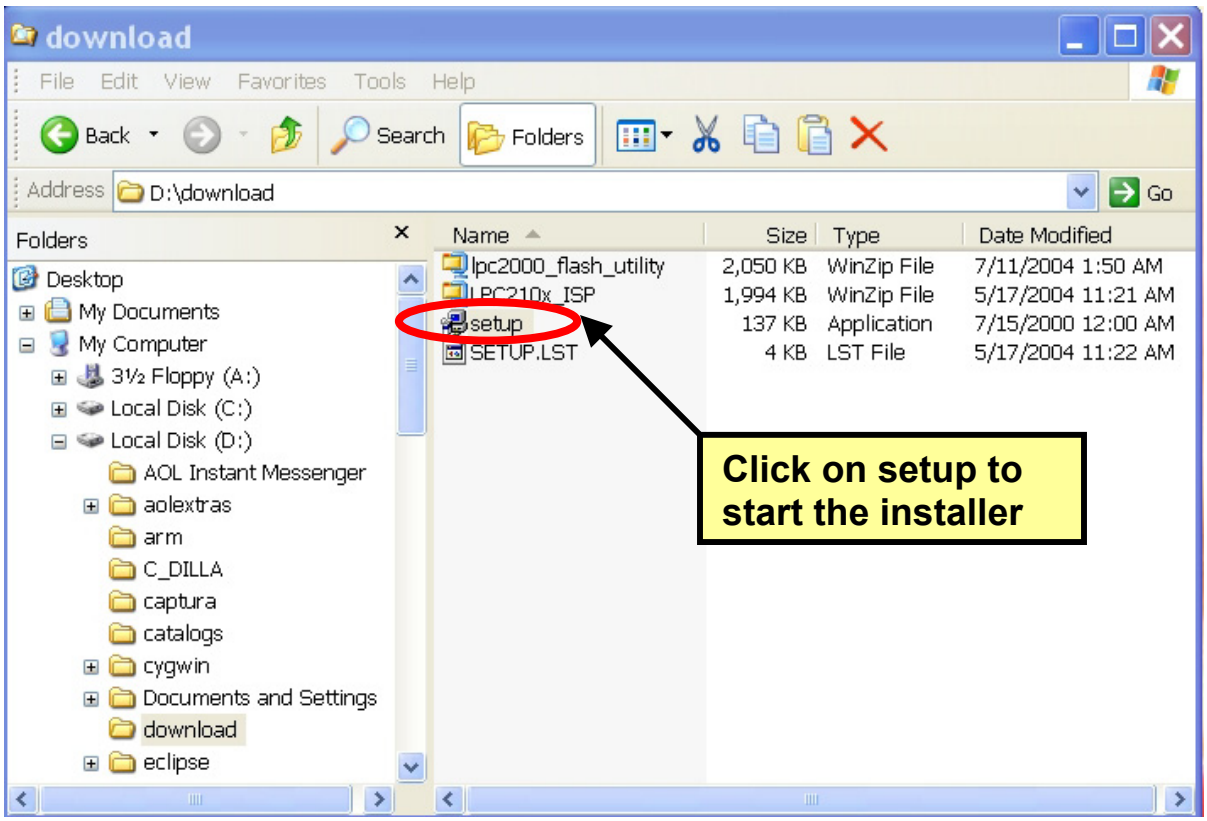




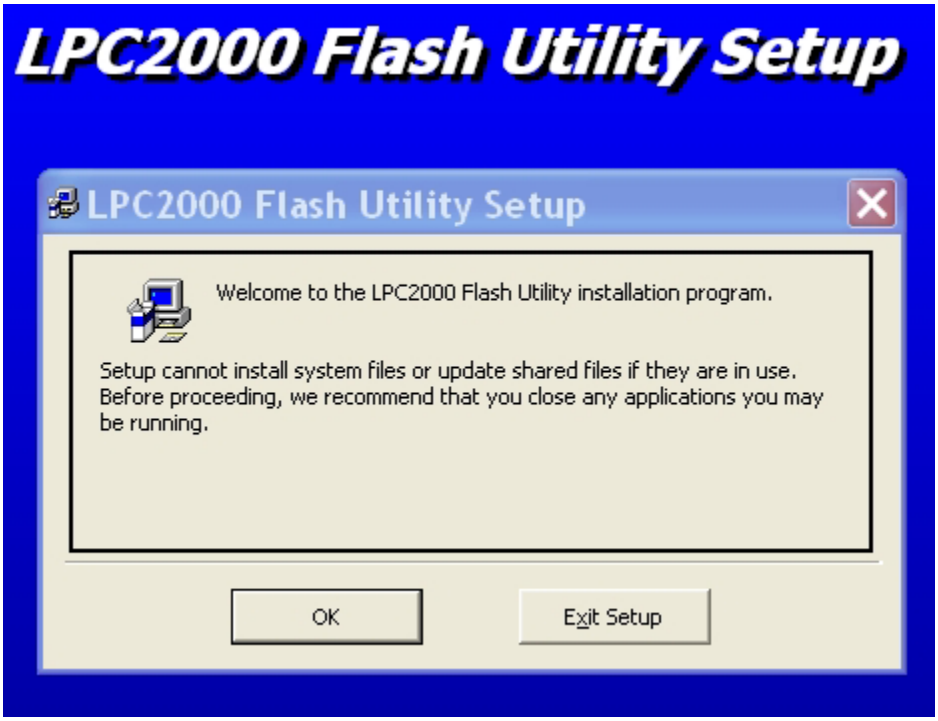
We'll use WinZip to unzip this into the download directory.



Now you can see that the download directory has a setup utility and another zip file containing the LPC2000 Hex Utility. Click on the **setup.exe** application to start the installer.

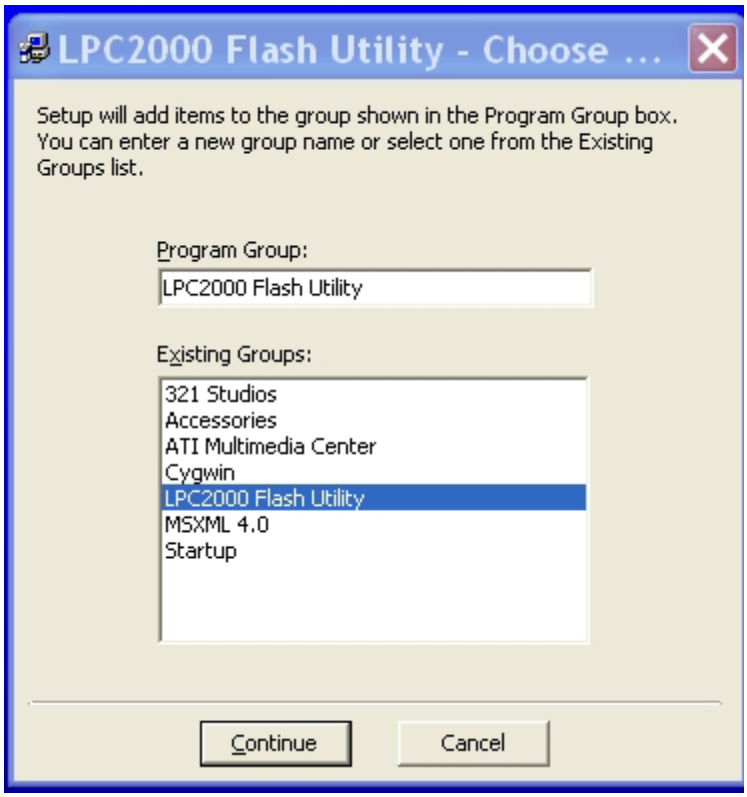


The LPC2000 Flash Utility setup now starts. Click on **OK** to proceed.



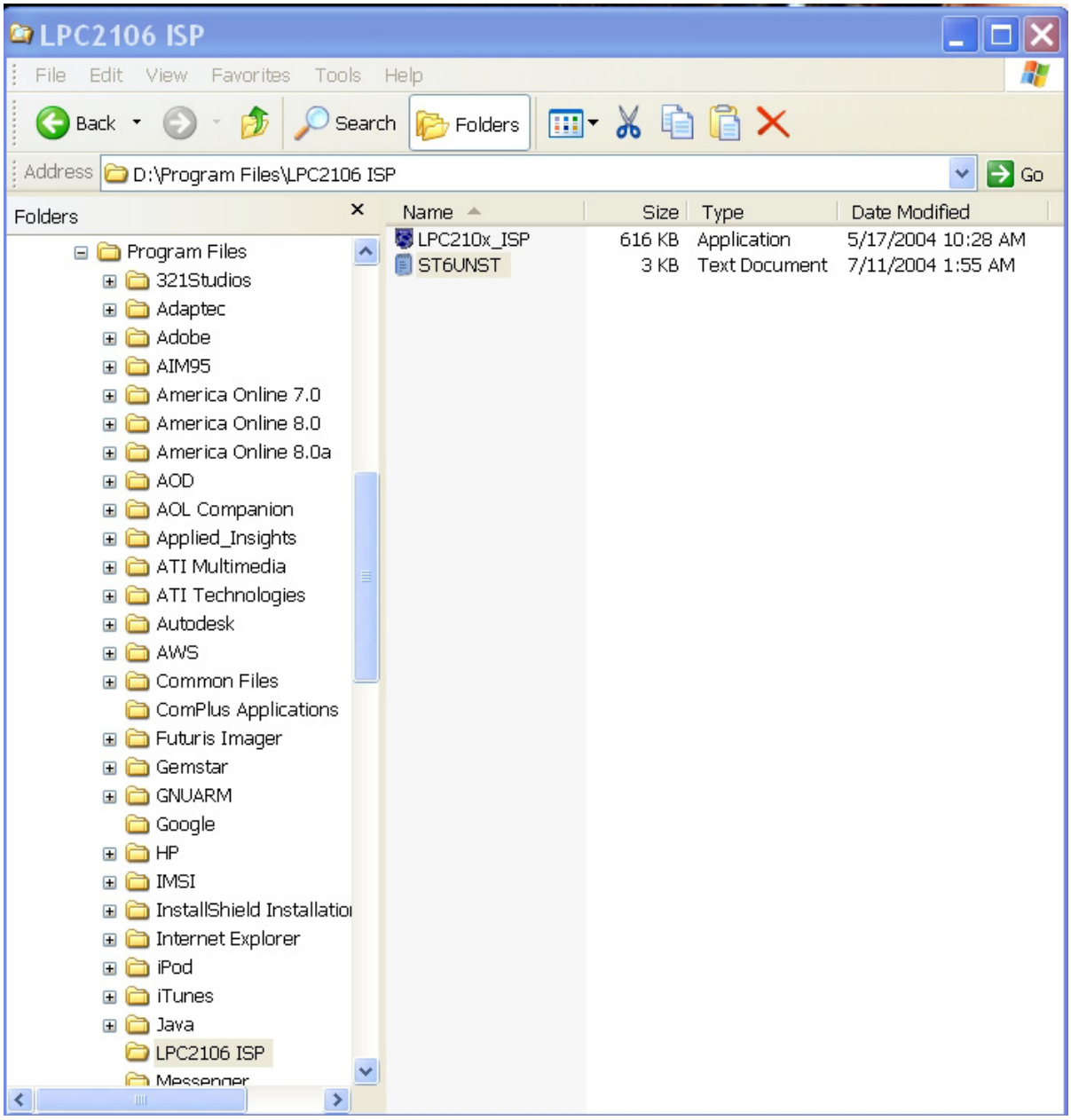
Take the default on this screen and let it install the LPC2000 Flash Utility into the Program Files directory.

Take the default on the screen below. Click on "**Continue**."



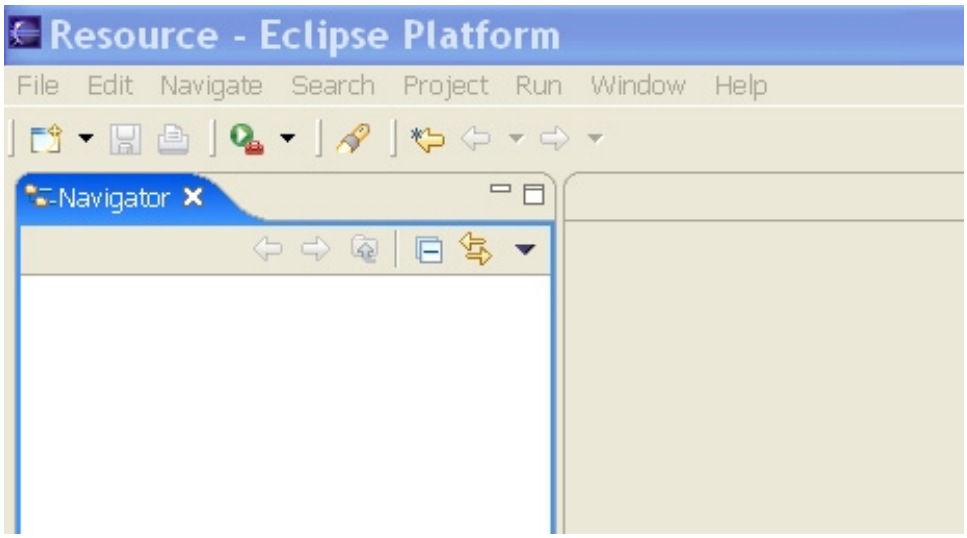
In a very few seconds, the installer will complete and you should see this screen.

Here we see the utility residing in the Program Files directory, just as promised.

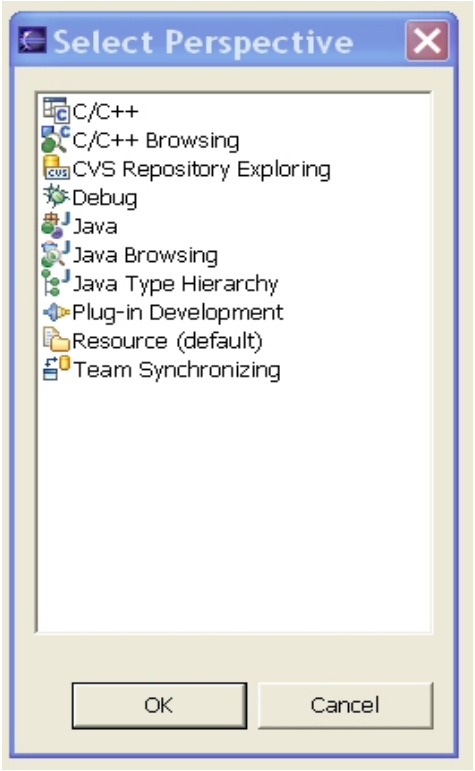


Now that the Philips LPC2000 Flash Utility is properly installed on our computer, we'd like to install it into Eclipse so that it can be invoked from the RUN pull-down menu under the "external tools" option. Start Eclipse by clicking on the desktop icon.

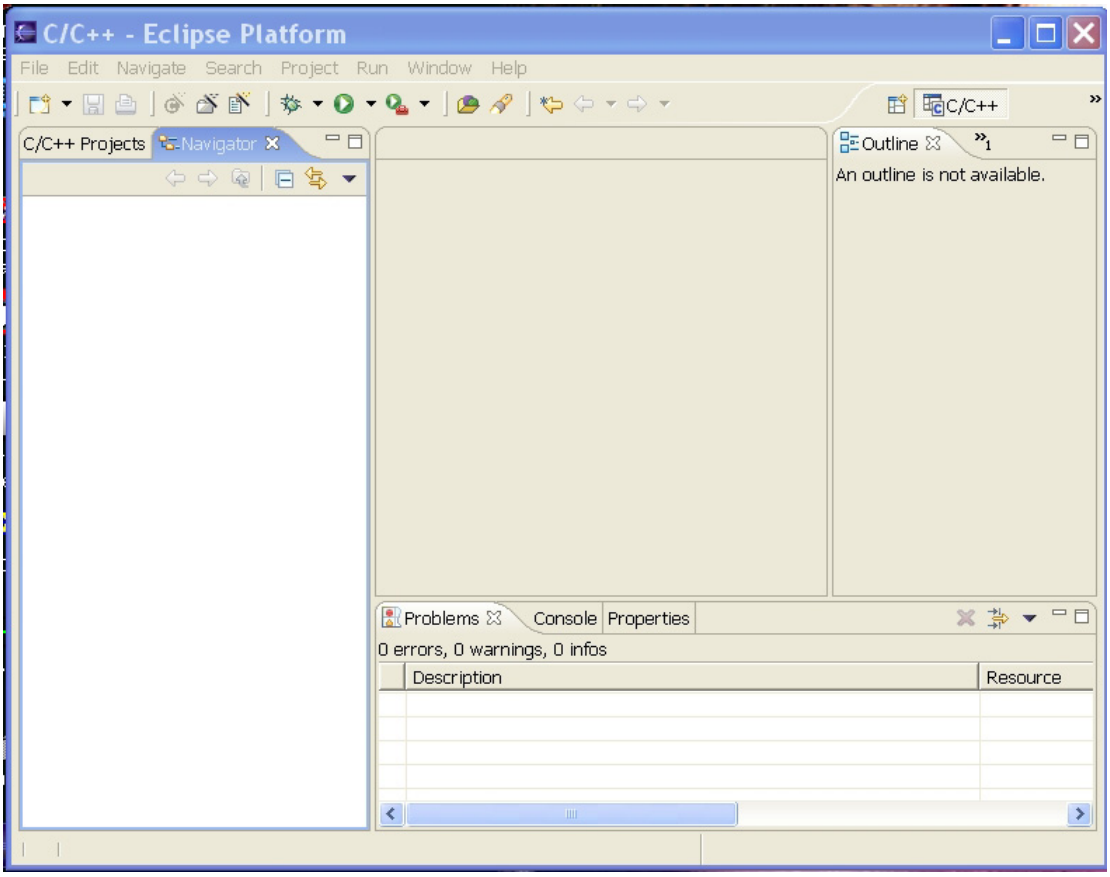
The layout of the Eclipse screen is called a "perspective." The default perspective is the "resource" perspective, as shown below.



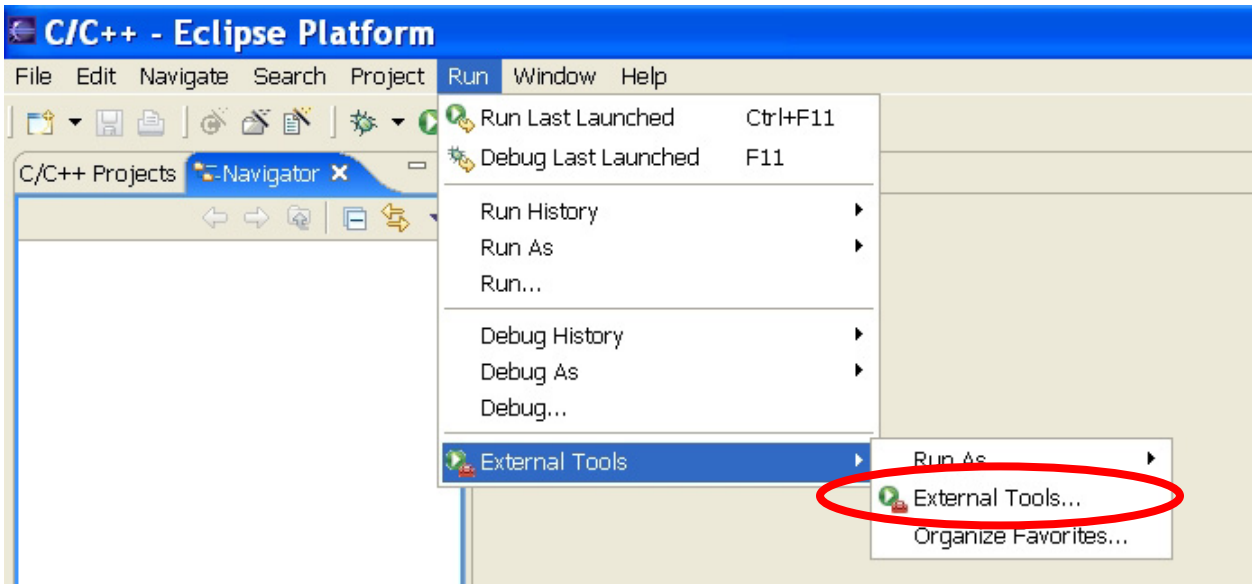
We need to change it into the C/C++ perspective. In the **Window** pull-down menu, select **Window – Open Perspective – Other – C/C++** and then click **OK**.



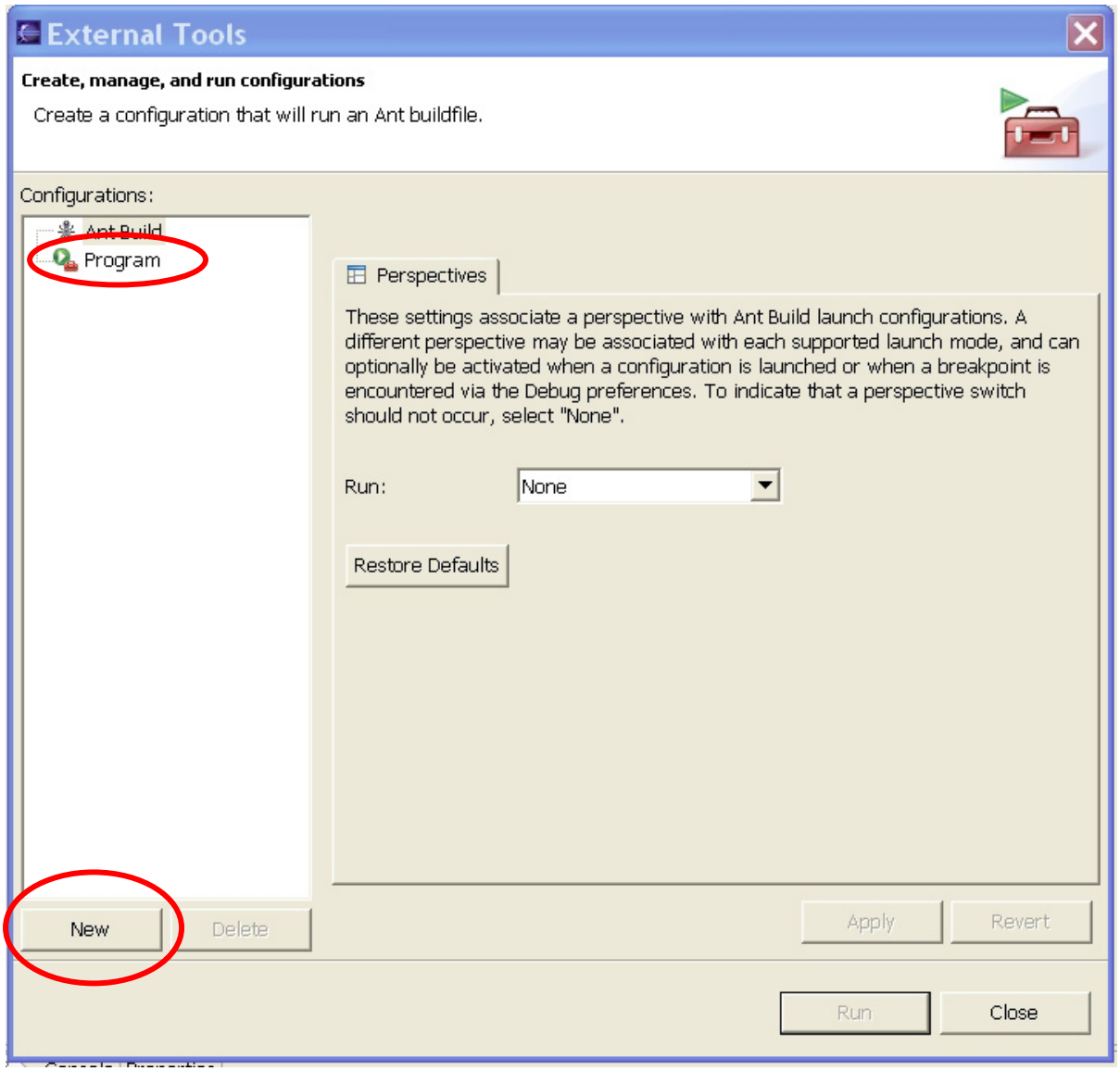
Eclipse will now switch to the **C/C++** perspective shown below and will remember it when you exit.



Now we want to add the Philips LPC2000 Flash Utility to the "**External Tools**" part of the **Run** pull-down menu. Select **RUN – External Tools – External Tools**.



We want to add a new program to the External Tools list, so click on **Program** and then **New**.

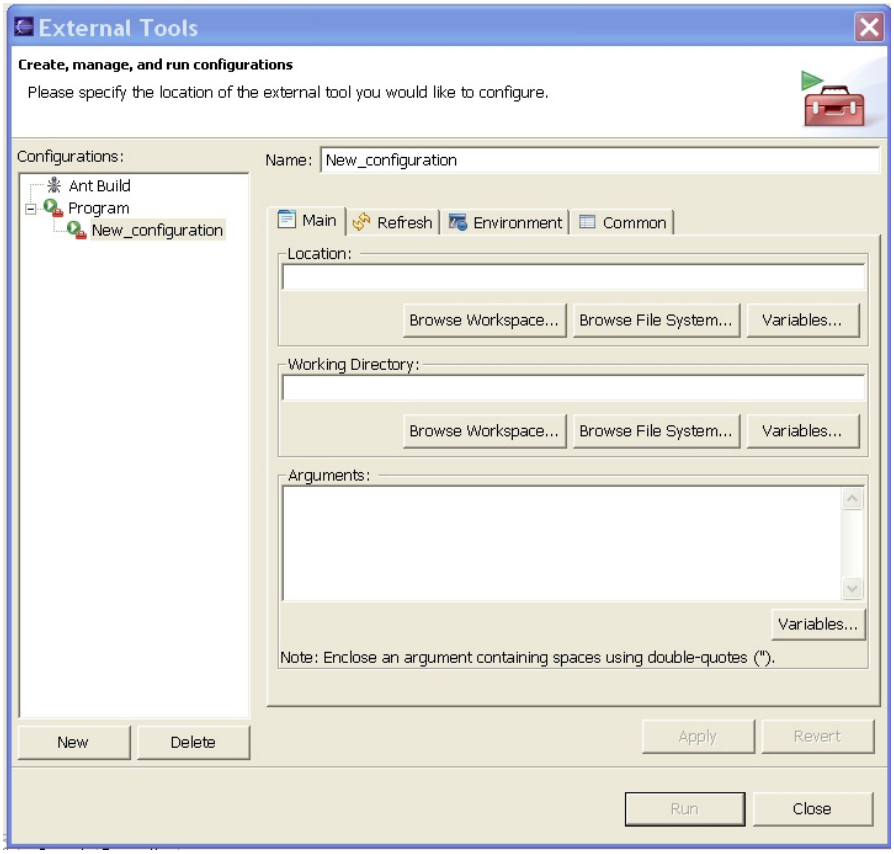


Note below that there's a new program under the "program" tree with the name **New_configuration** and there's no specifications as to what it is.

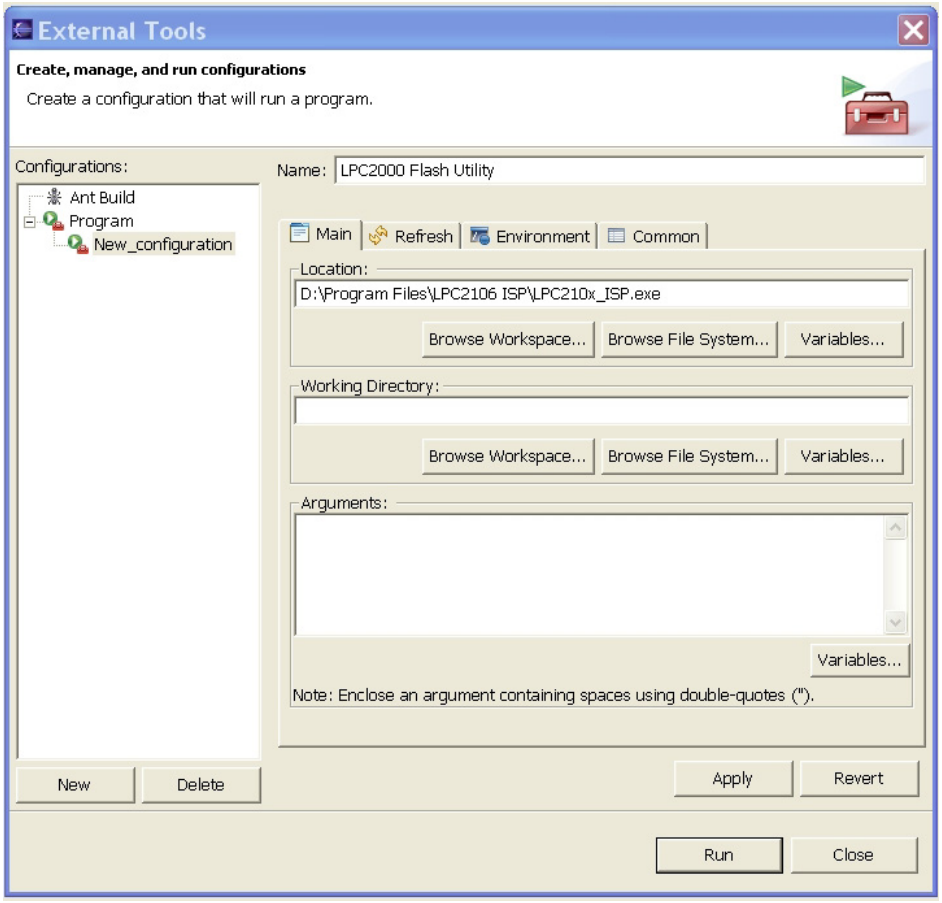
In the **Name** text box, replace **New-configuration** with **LPC2000 Flash Utility**.

In the **Location** text box, use the "**Browse File System**" tool to find the Philips LPC2000 Flash Utility in the Program Files directory. Its name is **LPC210x_IPC.exe**.

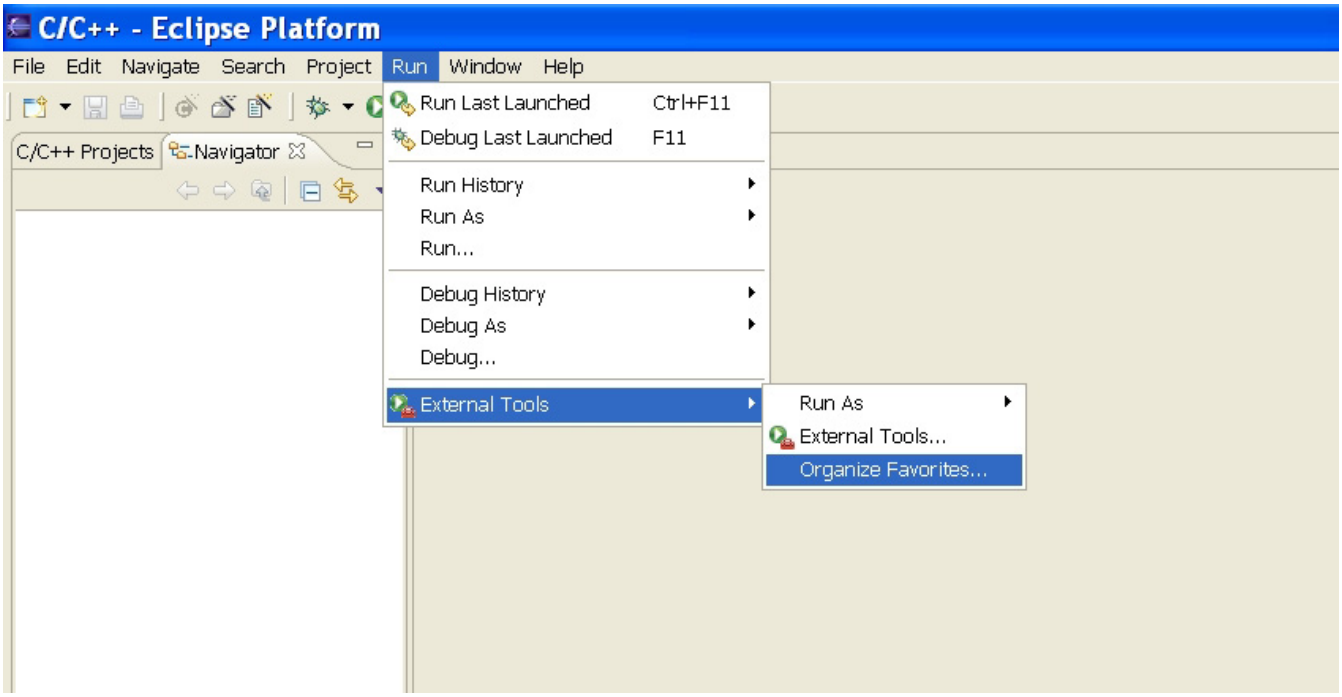
Here's the External Tools window before editing.



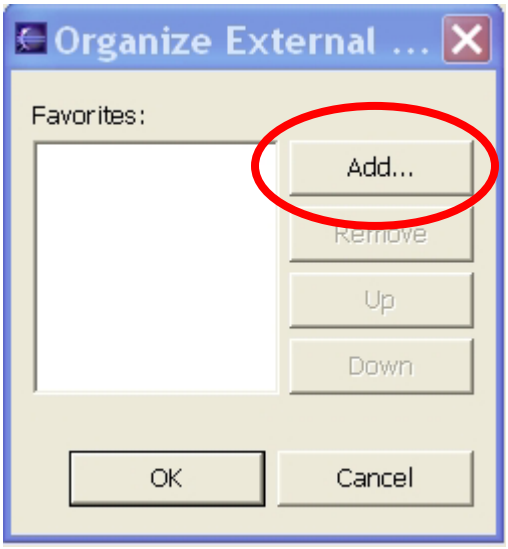
Here's the External Tools window after our modifications. Click on **Apply** to accept.



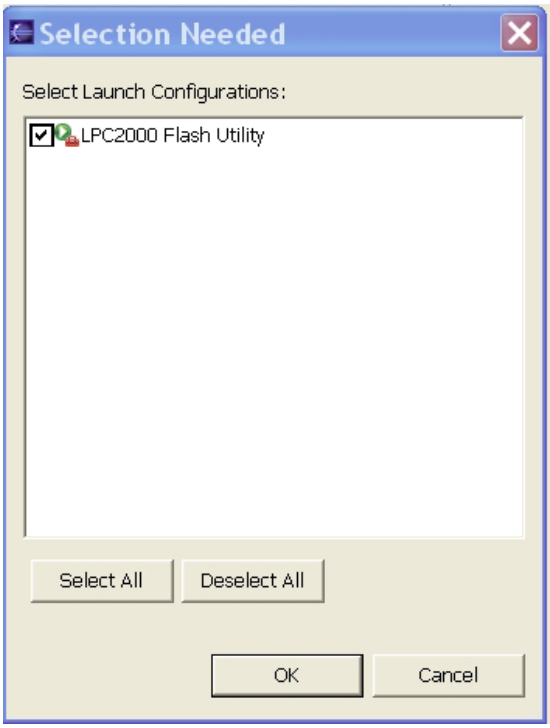
Close everything out and return to the **Run** pull-down menu. Select **Run – External Tools – Organize Favorites**.



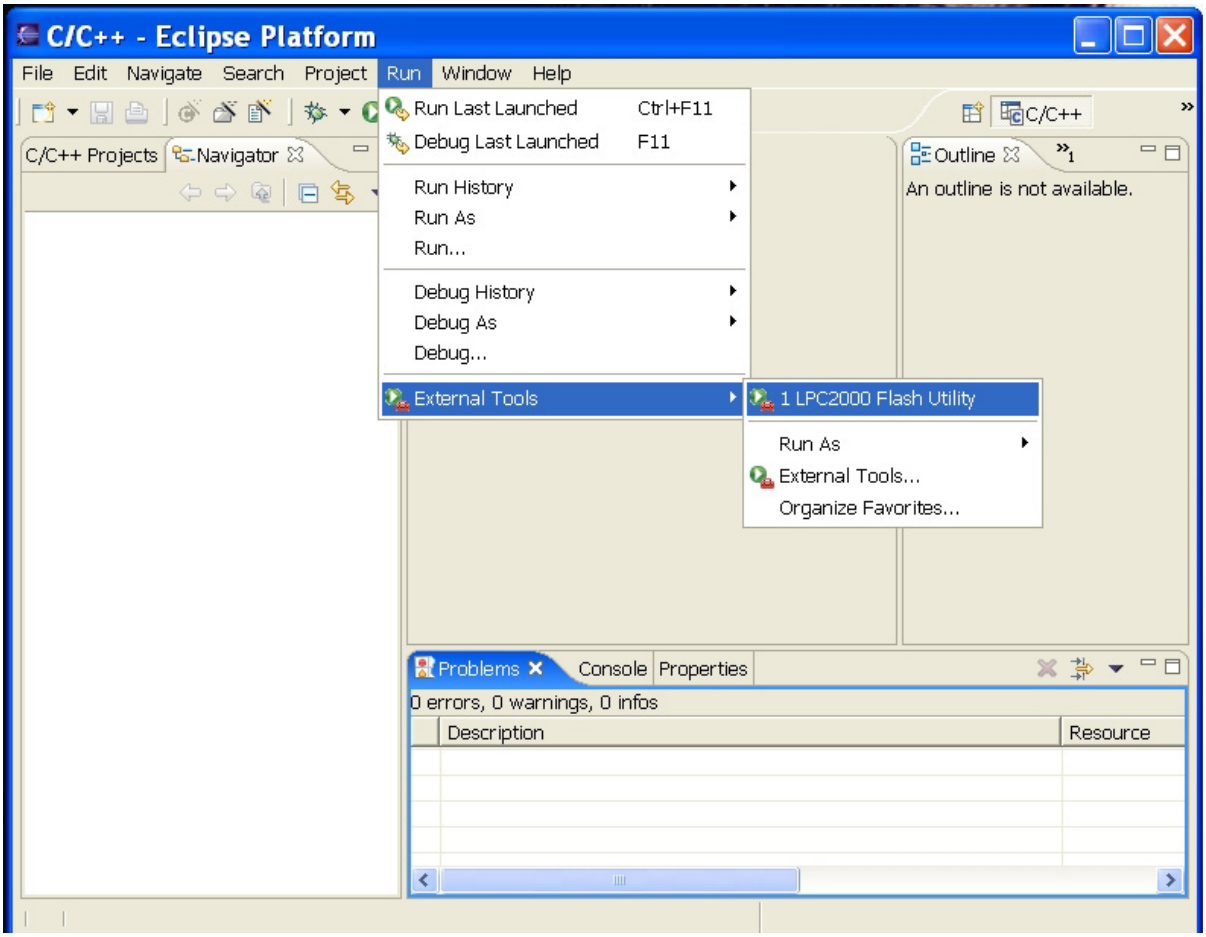
We're now going to put the Philips PLC2000 Flash Utility into the "favorites" list. Click on in the window below.



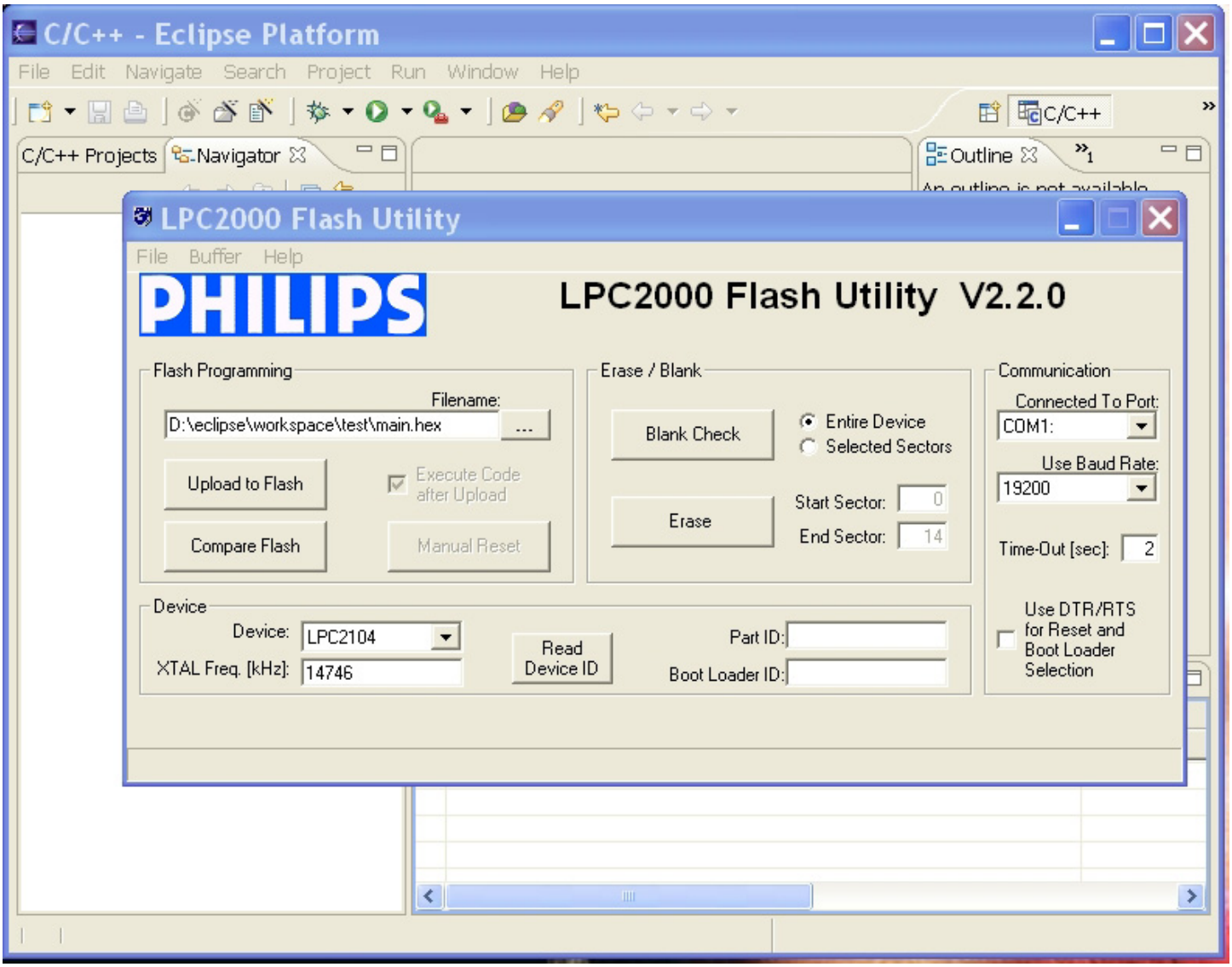
Click the selection box for LPC2000 Flash Utility. This will add it to the favorites list.



Now when we click on the Run pull-down menu and select "External Tools," we see the **LPC2000 Flash Utility** at the top of the list.



Click on LPC2000 Flash Utility to verify that it runs.



Now cancel the LPC2000 Flash Utility and quit Eclipse.

7. Downloading the New Micros TiniARM Example Program

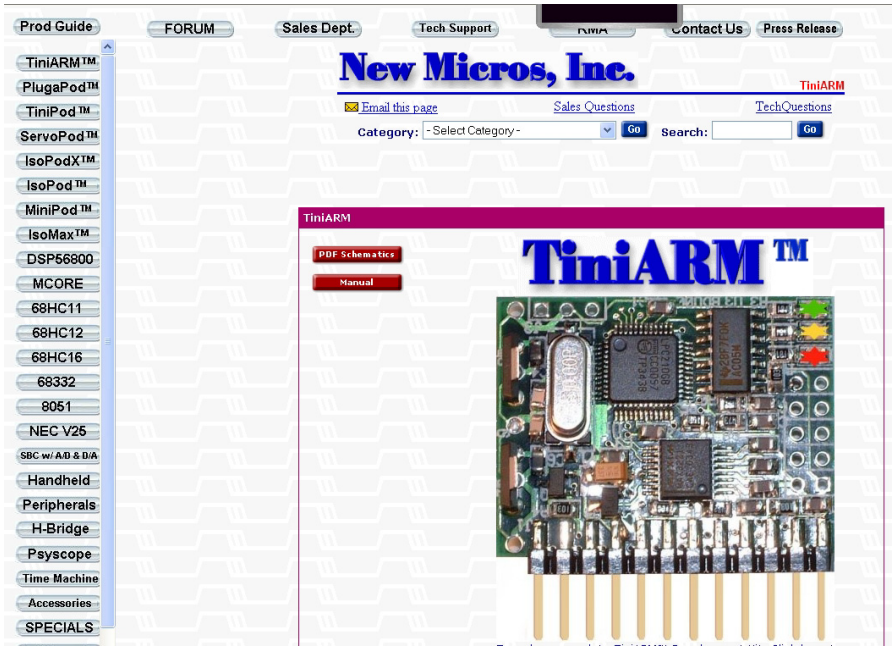
The example provided by New Micros for their **TiniARM** development system is a good place to start to check out our Eclipse system.

Click on the following link to go to the New Micros web site.

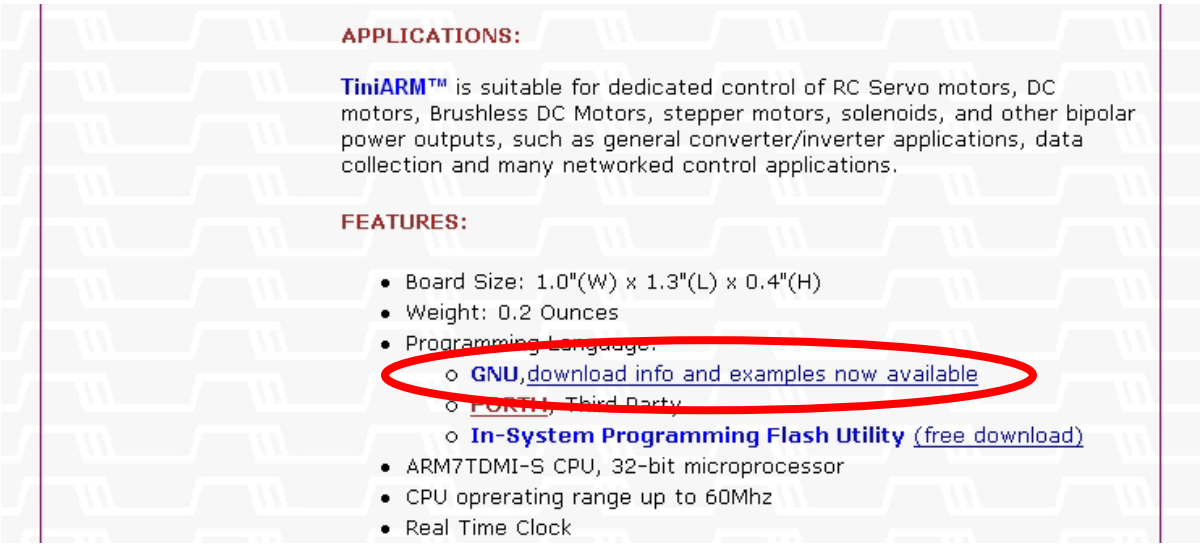
www.newmicros.com

When the New Micros web site appears, click on **TiniARM** to continue.

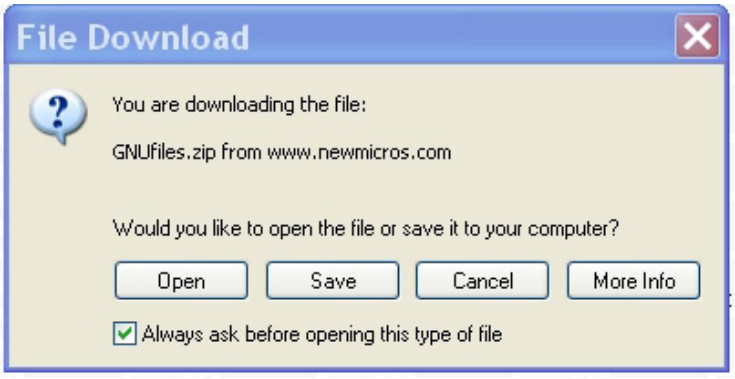
When the page for the **TiniARM** loads, scroll down towards the middle of the page.



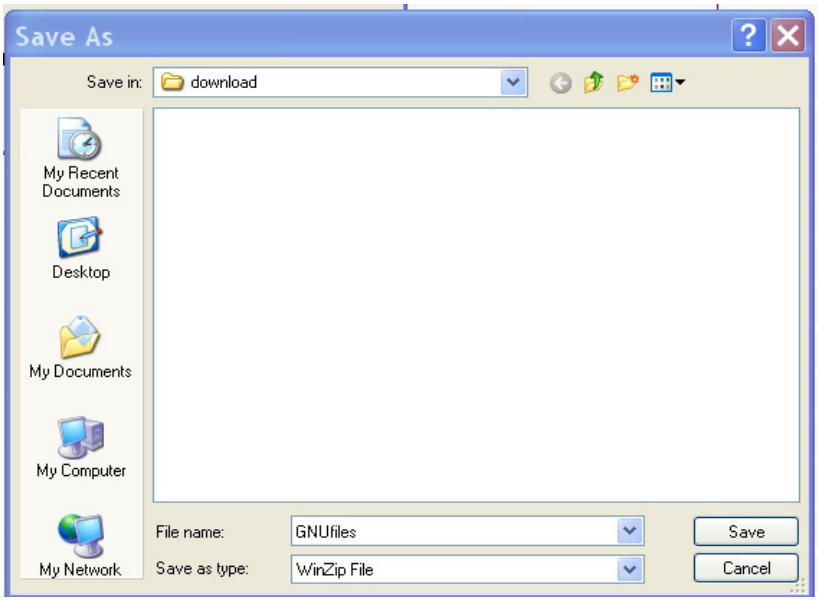
In the “Features” section, click on the link “**GNU, download info and examples now available.**”



As usual, we will download into our empty “download” scratch directory on the hard drive.

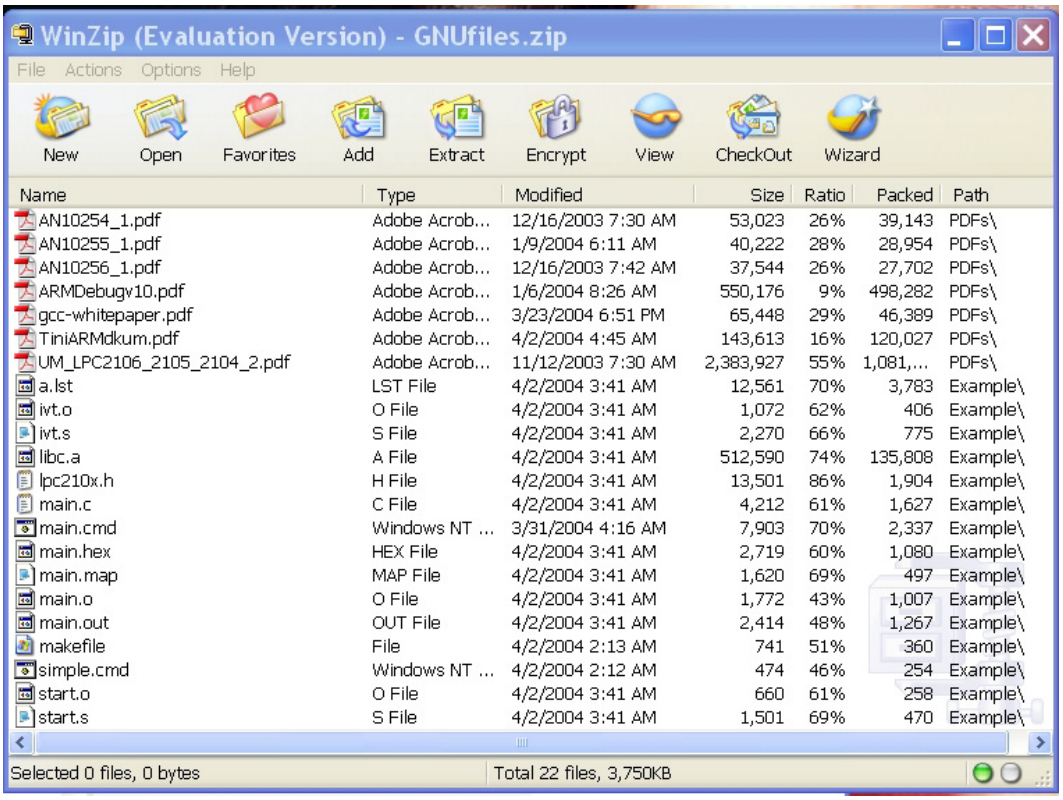


After

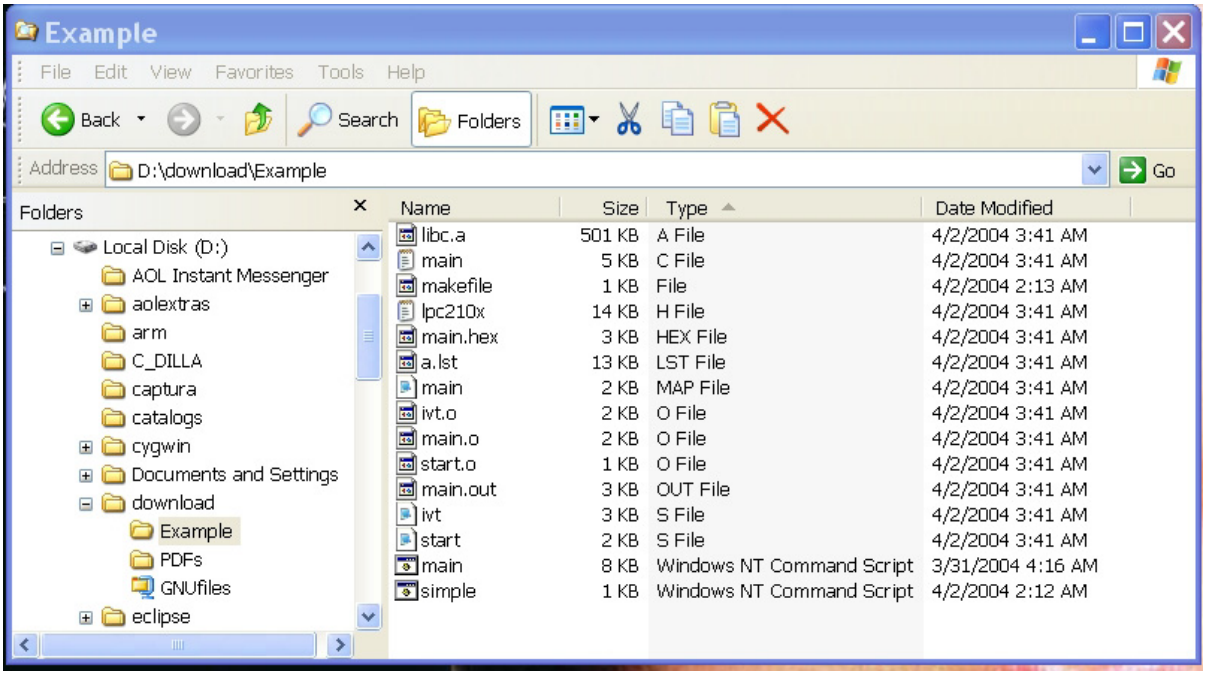


downloading the documentation and examples, we see that the download directory has a zip file named **GNUfiles.zip**.

Using WinZip, we extract this file into the **download** directory.



When unzipping is finished, we see the following files in the **download\example** directory.



When we create a **CDT** project to build the New Micros example, the following files from the **download\example** directory will be needed.

main.c	Test program in C (performs echo over serial port)
lpc210x.h	Standard Philips header file with all registers defined
ivt.s	Interrupt vector table in assembler language
start.s	Startup code in assembler language to set up TiniARM micro and jump to the main program
libc.a	Pre-compiled C library
simple.cmd	Simple Linker script
makefile	The script on how to compile and build the test application

The other files are the results on New Micros themselves building this application. We want to build it ourselves using the Eclipse system.

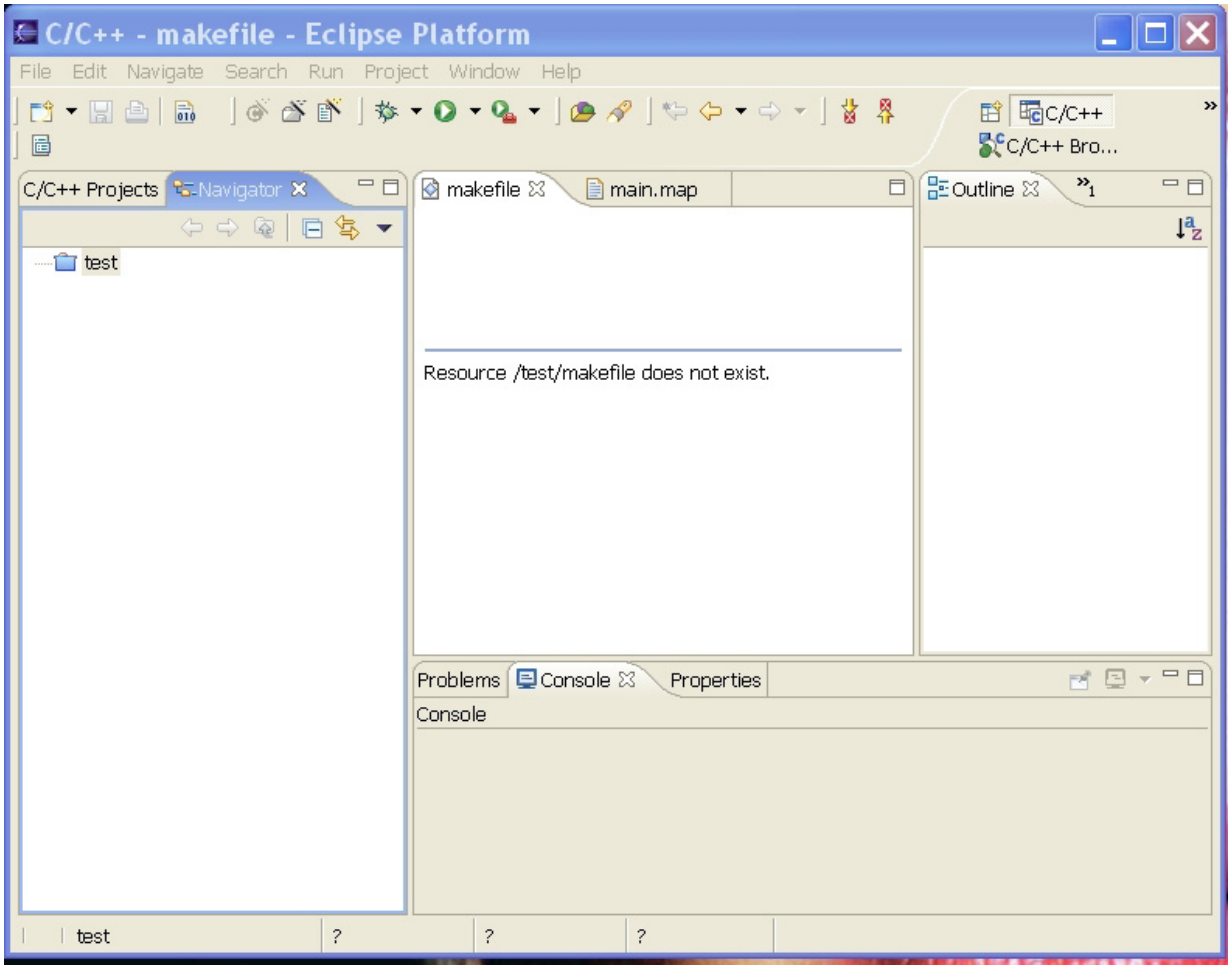
8. Creating a Eclipse Project

At this point, we have a fully functional Eclipse IDE capable of building C/C++ programs for the ARM microcomputer (specifically the TiniARM hardware).

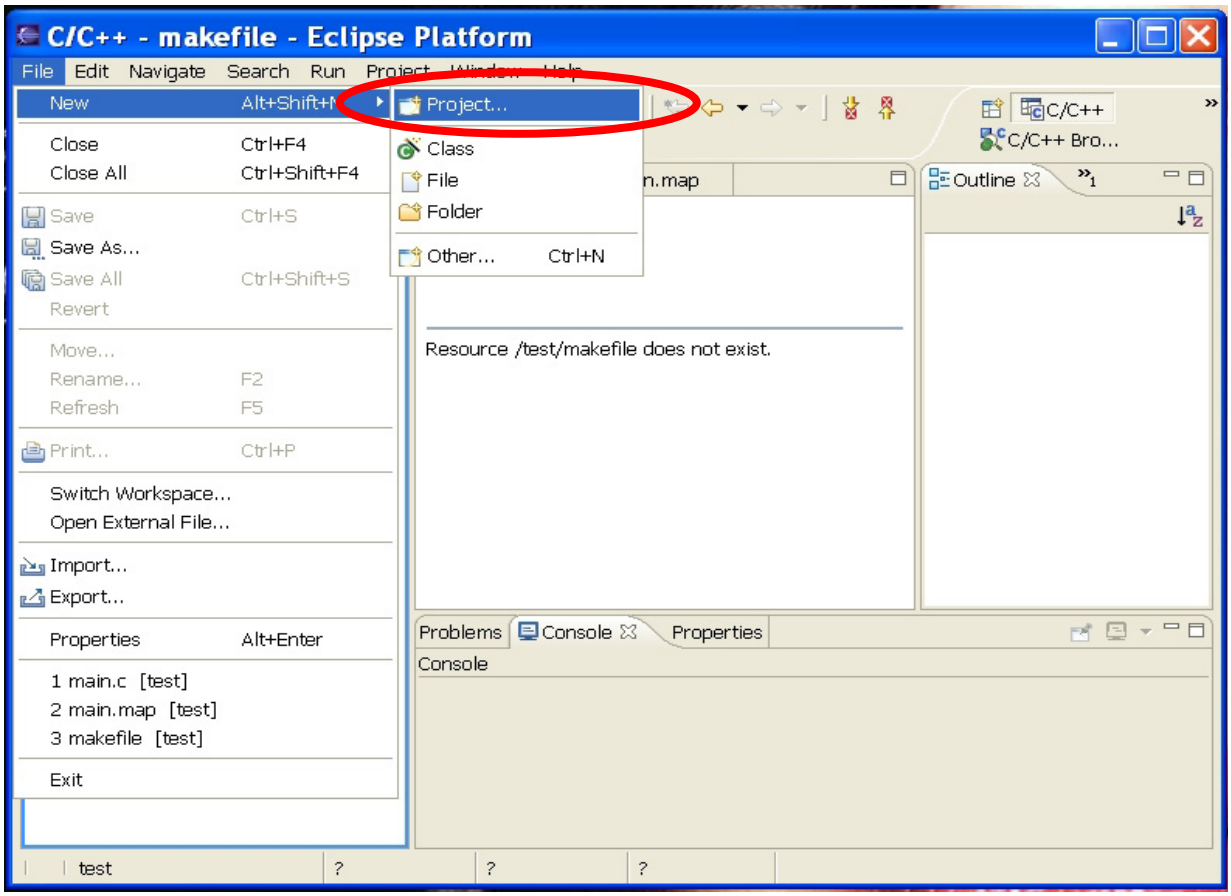
We will now create an Eclipse C++ project called “test” and import the New Micros TiniARM example test program into the project. It will work without change save for three simple lines added to the makefile to satisfy Eclipse.

Click on our Eclipse desktop icon to start Eclipse.

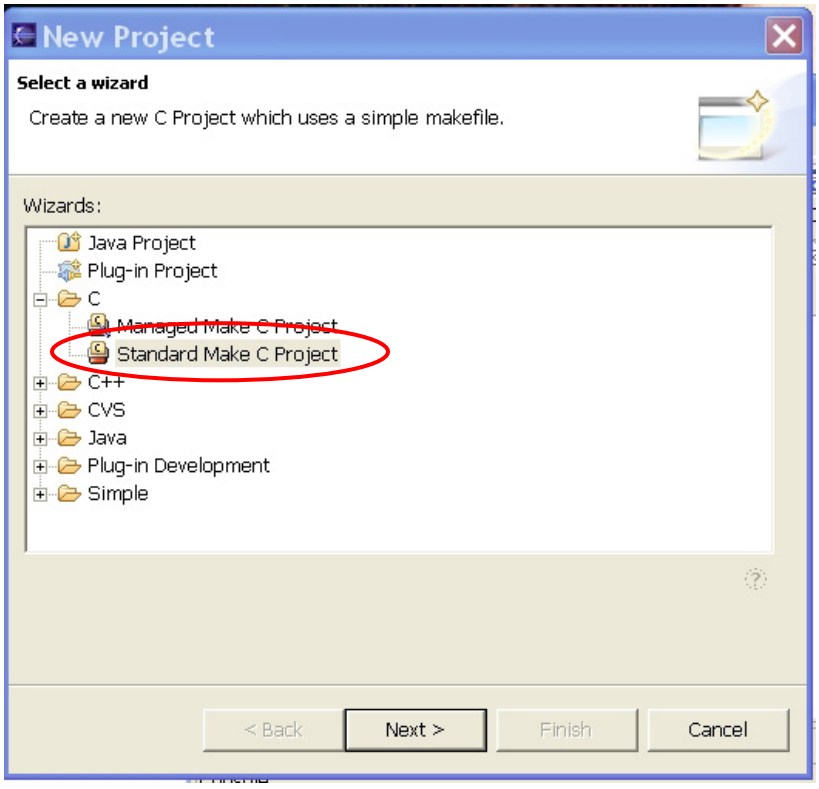
Eclipse should come up with the C/C++ perspective as shown below. If not, select “**Window – Open Perspective – Other – C/C++**” to change to the C++ perspective.



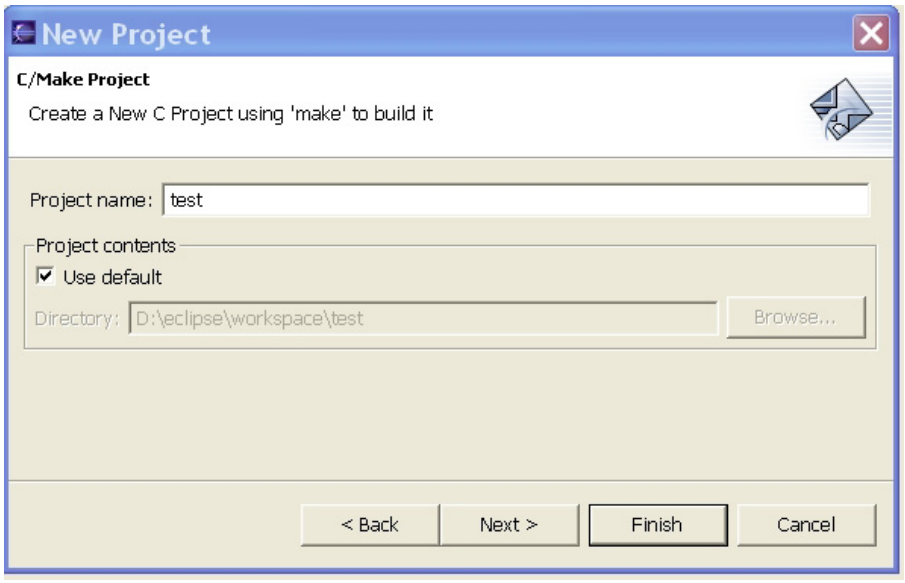
To create a C project, select **File – New - Project** from the **File** pull-down menu as shown below.



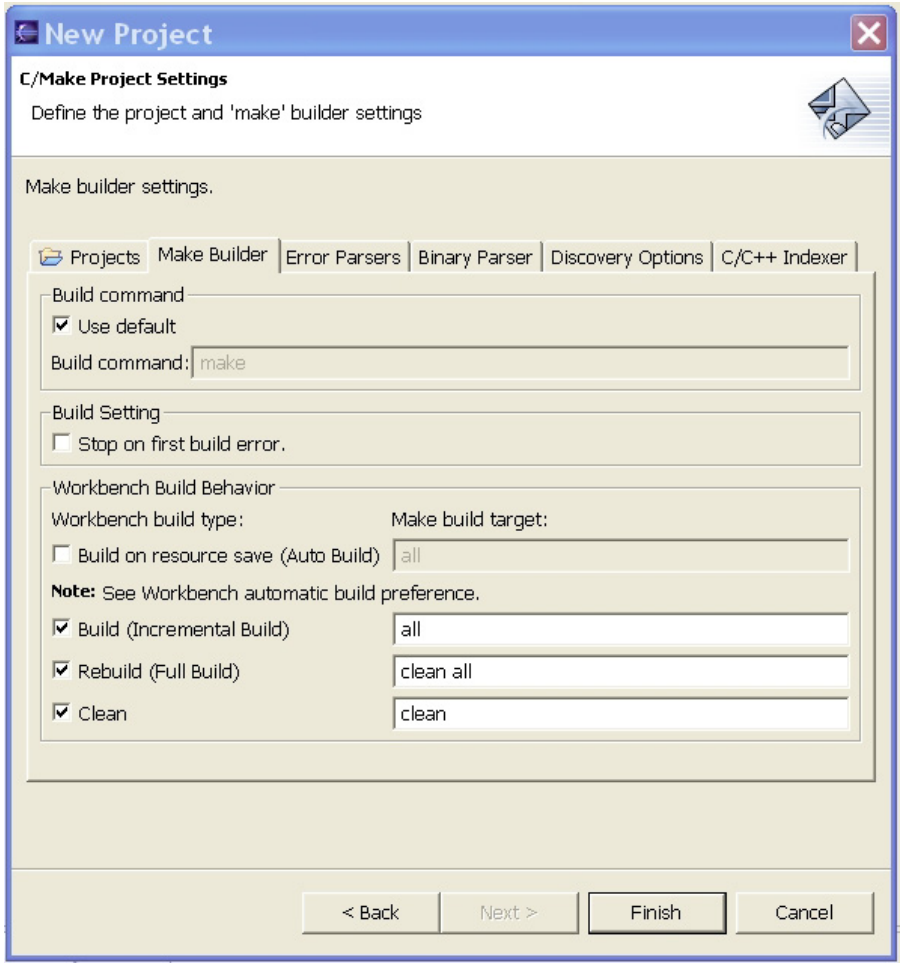
Select "**C – Standard Make C Project**" and then click "**Next**."



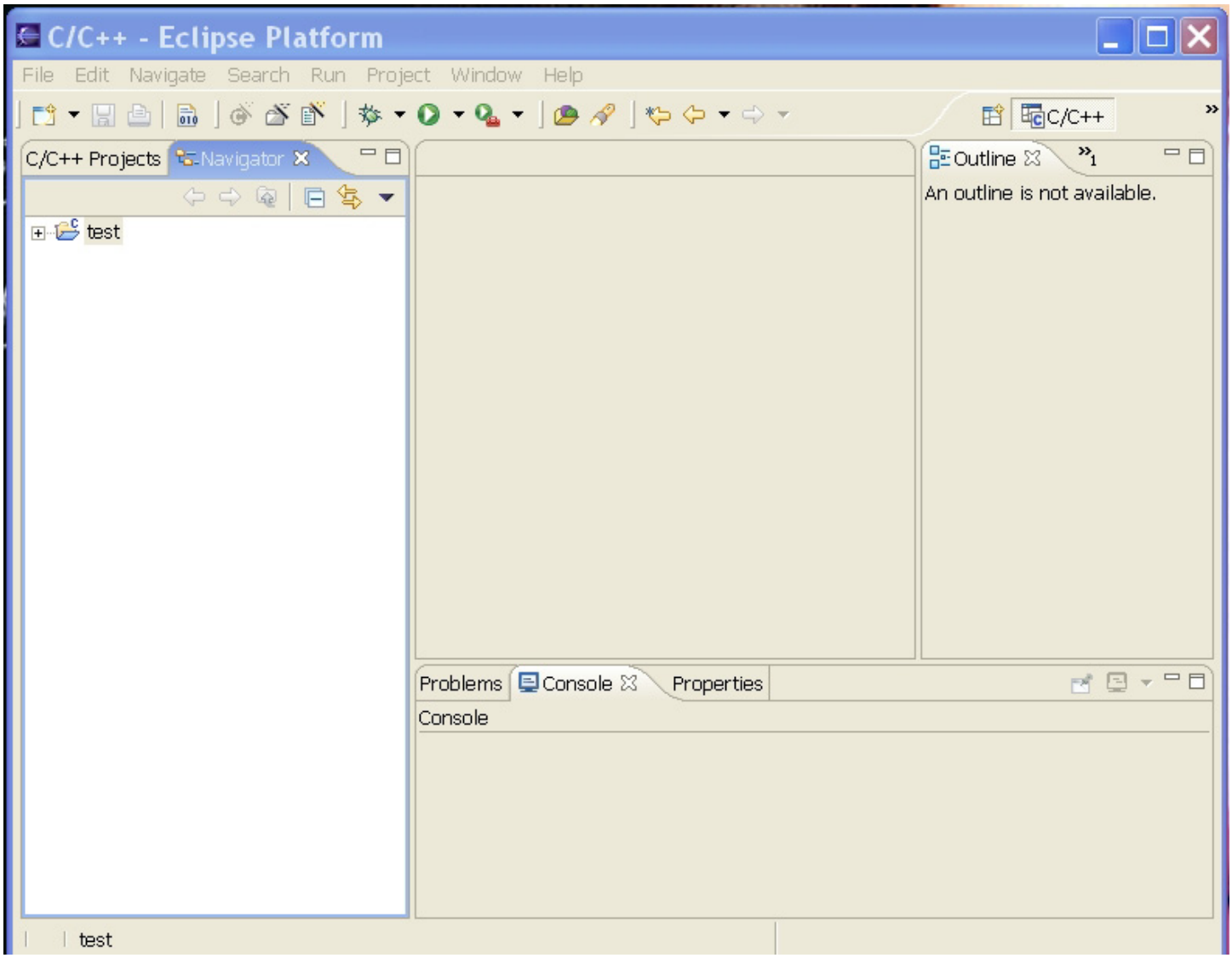
In the New Project window, we give the project the name "**test**." Click on "**Next**" to continue.



If you click on the "**Make Builder**" tab, you'll see that Eclipse invokes the command **Make All** for an incremental build, **Make Clean All** for a full build and **Make Clean** for the clean operation. The New Micros example makefile doesn't have the **All** and **Clean** targets, so we will have to modify the example makefile to bring it into compliance with Eclipse/CDT. Click "**Finish**" to complete the project specification.



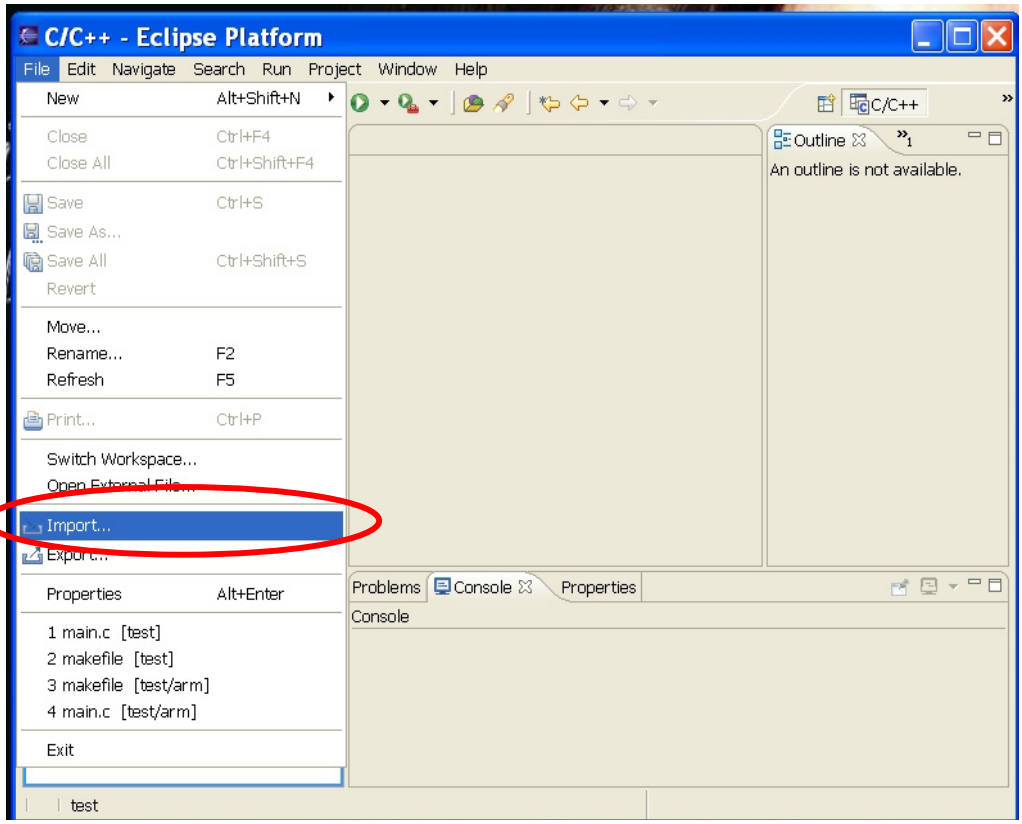
As the next screen shows, we have a legal C project named **test**, albeit with no files.



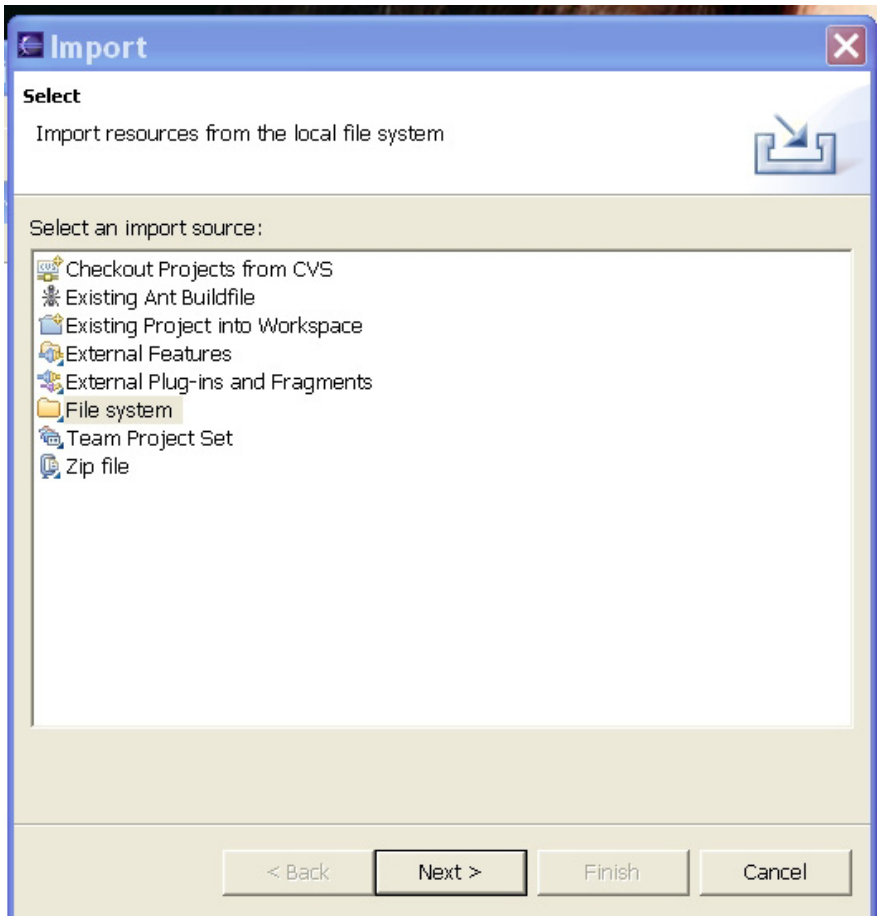
Remember from Section 7 above, we downloaded and unzipped the New Micros example test program into the directory **d:\download\example**.

We can use the **Import** facility under the File pull-down menu to fetch and insert these files into our Eclipse project.

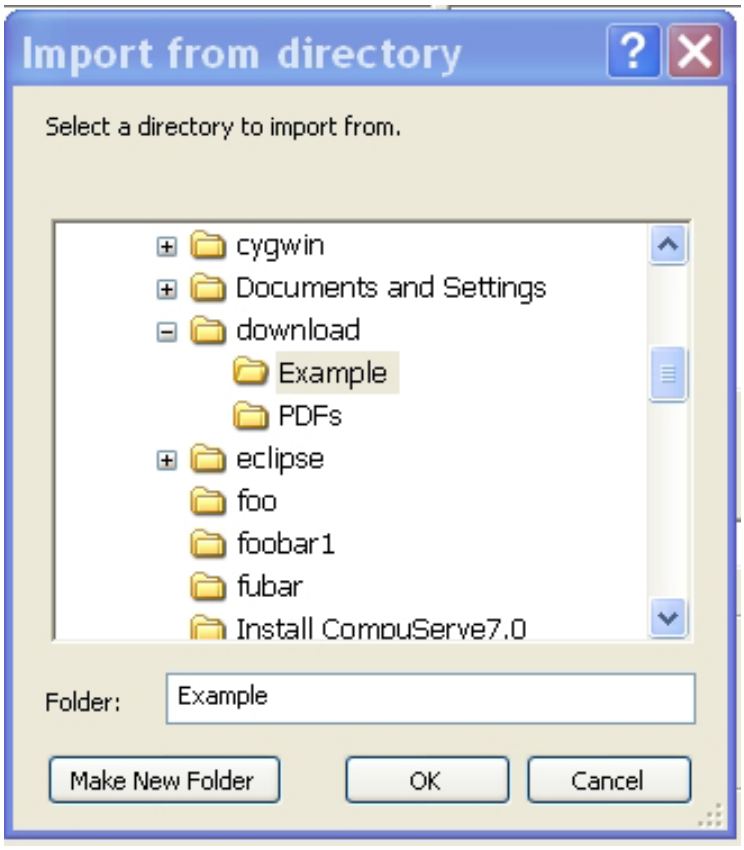
Click on **File – New – Import** to open the import facility, as shown below.



In this case, we'll want to import from the **File system**, as shown below.



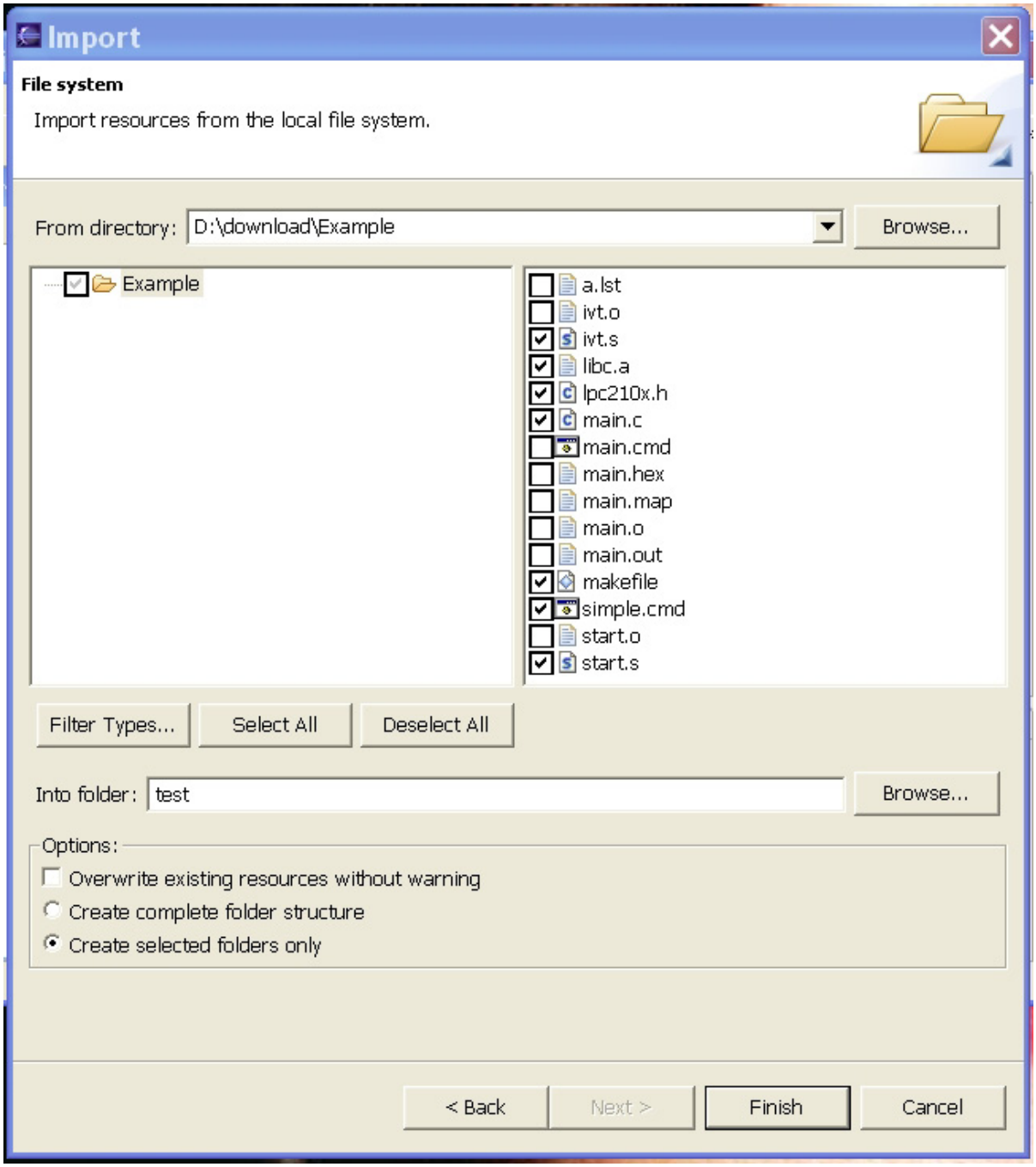
Now we browse to the desired directory: **d:\download\Example**. Click **OK** to continue.



There are 15 files in the Example directory. As mentioned before, we only need the following seven files to make up the complete project.

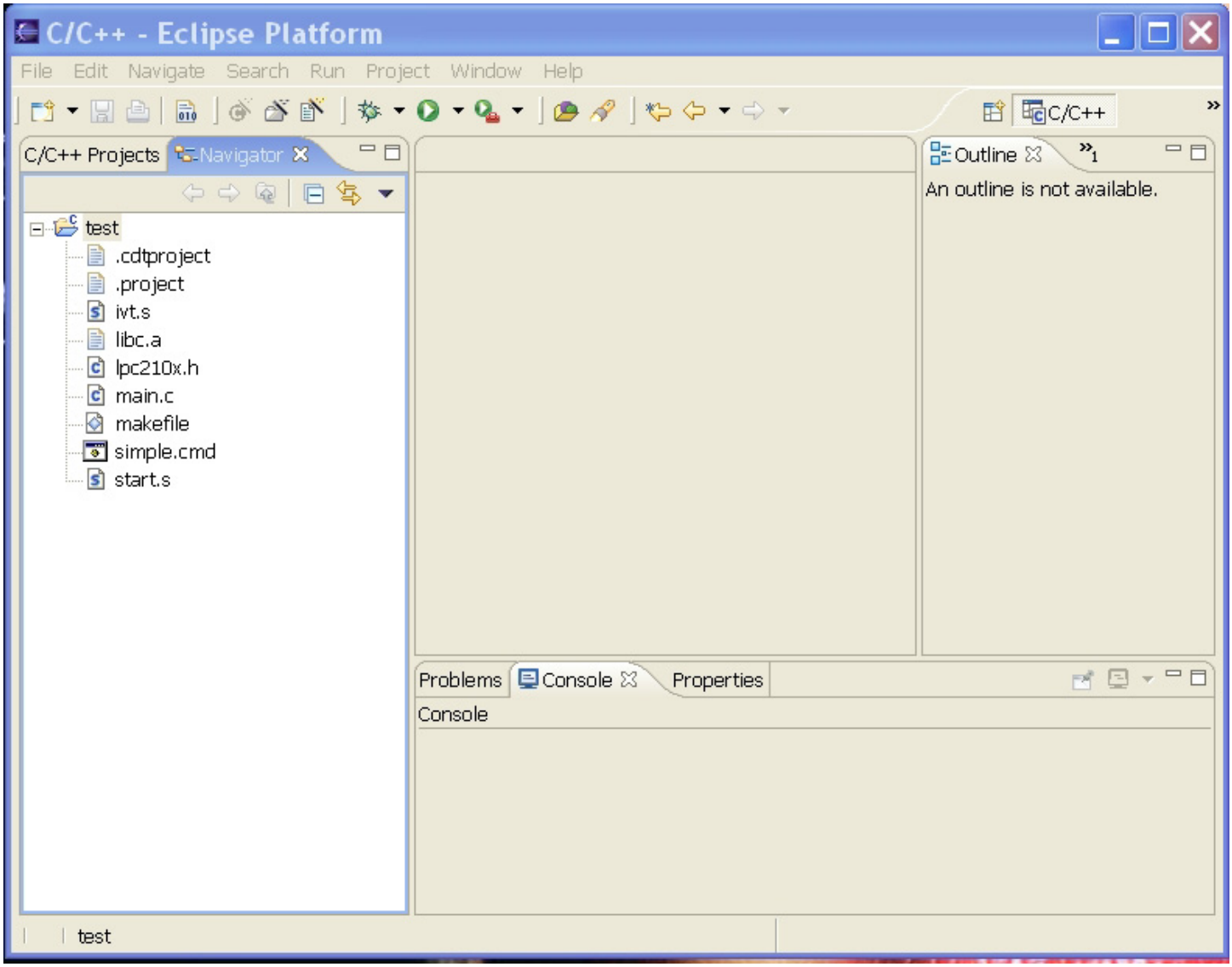
main.c	Test program in C (performs echo over serial port)
lpc210x.h	Standard Philips header file with all registers defined
ivt.s	Interrupt vector table in assembler language
start.s	Startup code in assembler language to set up TiniARM micro and jump to the main program
libc.a	Pre-compiled C library
simple.cmd	Simple Linker script
makefile	The script on how to compile and build the test application

Now we check only those files listed above for importation.



Now click **Finish** and these files will appear in our project.

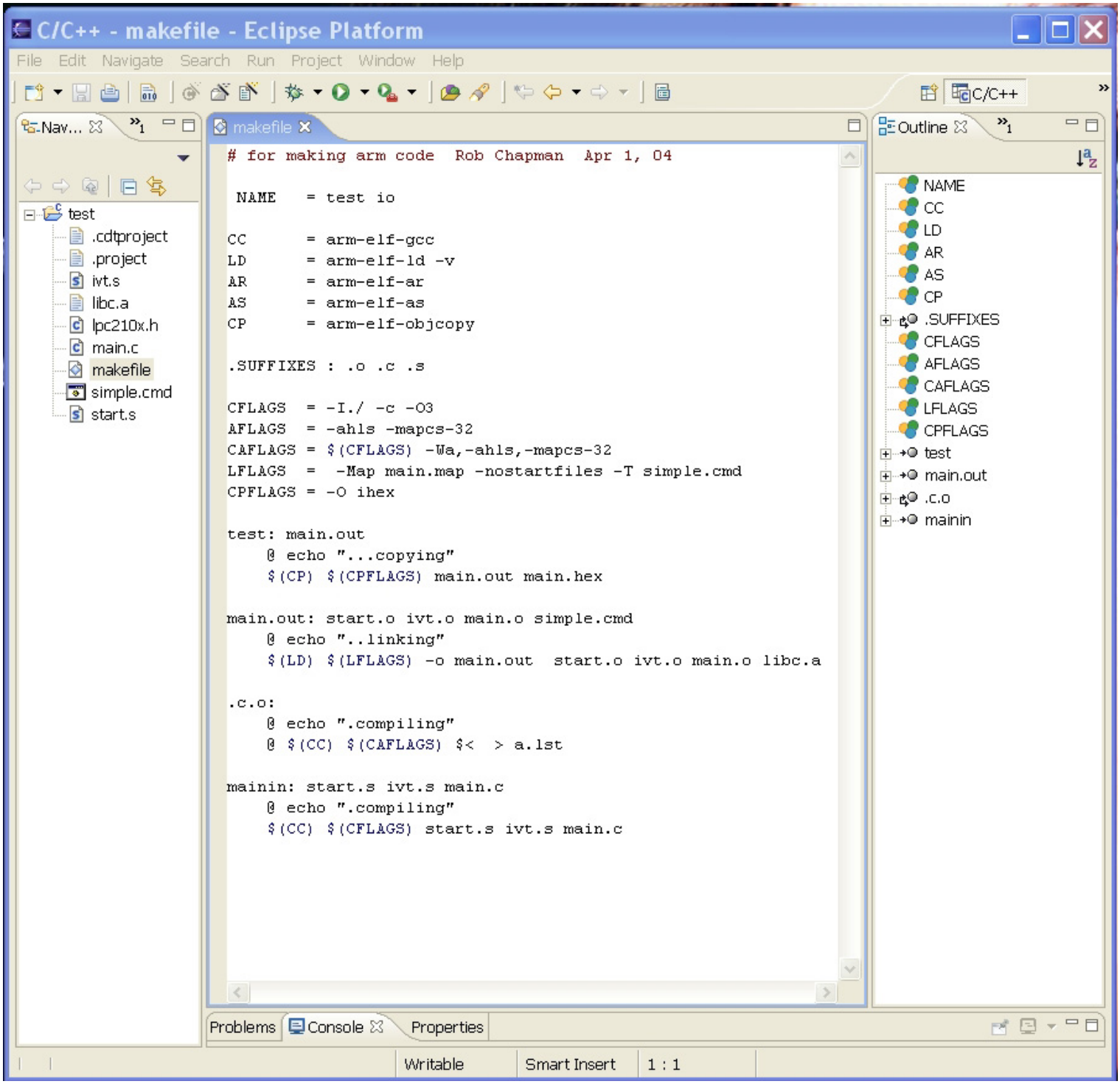
Notice that the Navigator pane shows the imported files.



Double-click on the makefile; it will pop up as an editor window.

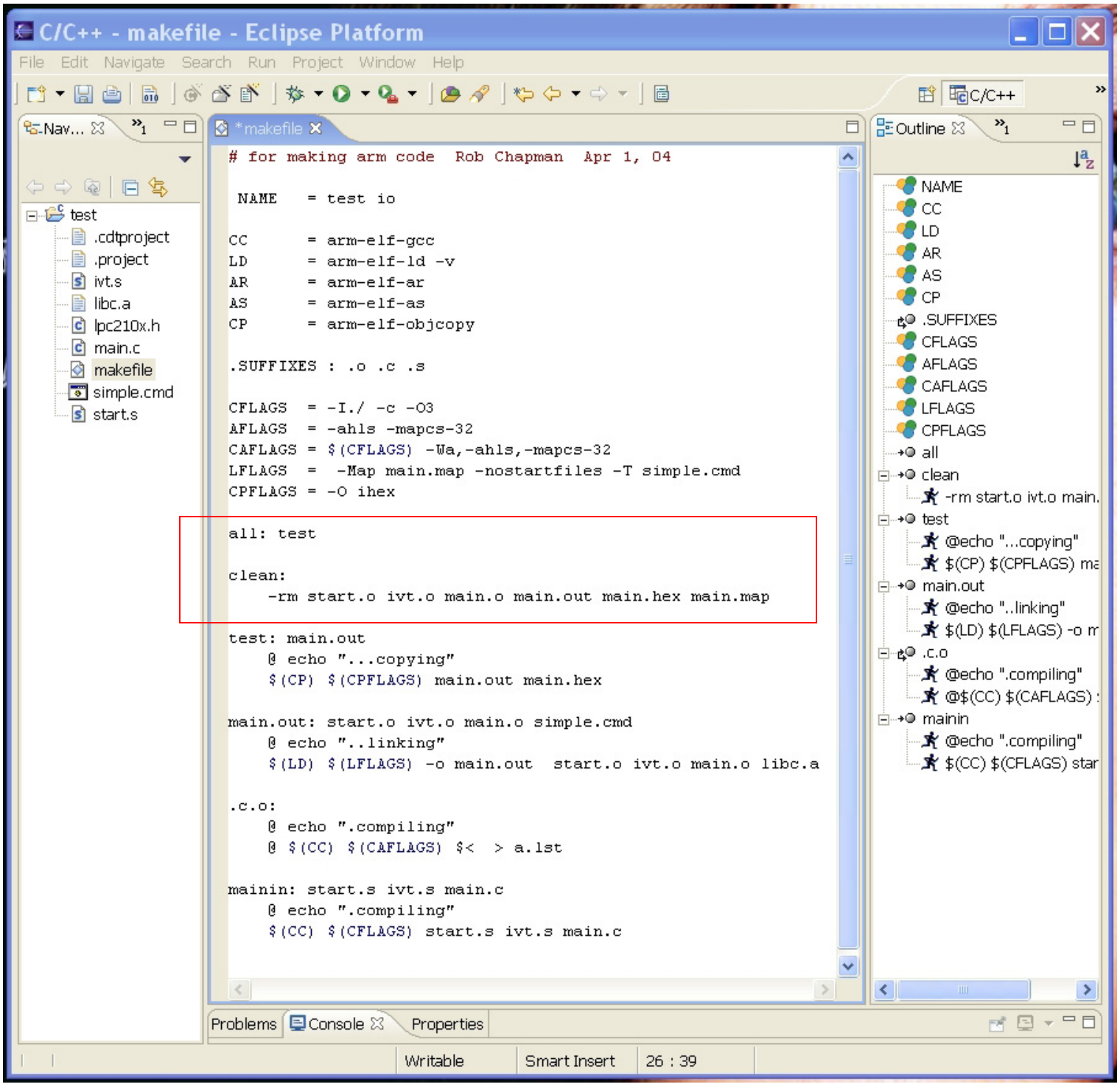
Eclipse requires that the makefile have the targets ALL and CLEAN. We will add three lines to satisfy this requirement.

This is the makefile as supplied by New Micros.



We'll add three lines to provide the missing targets. The updated makefile is shown below.

Notice the three new lines, outlined by a red box.

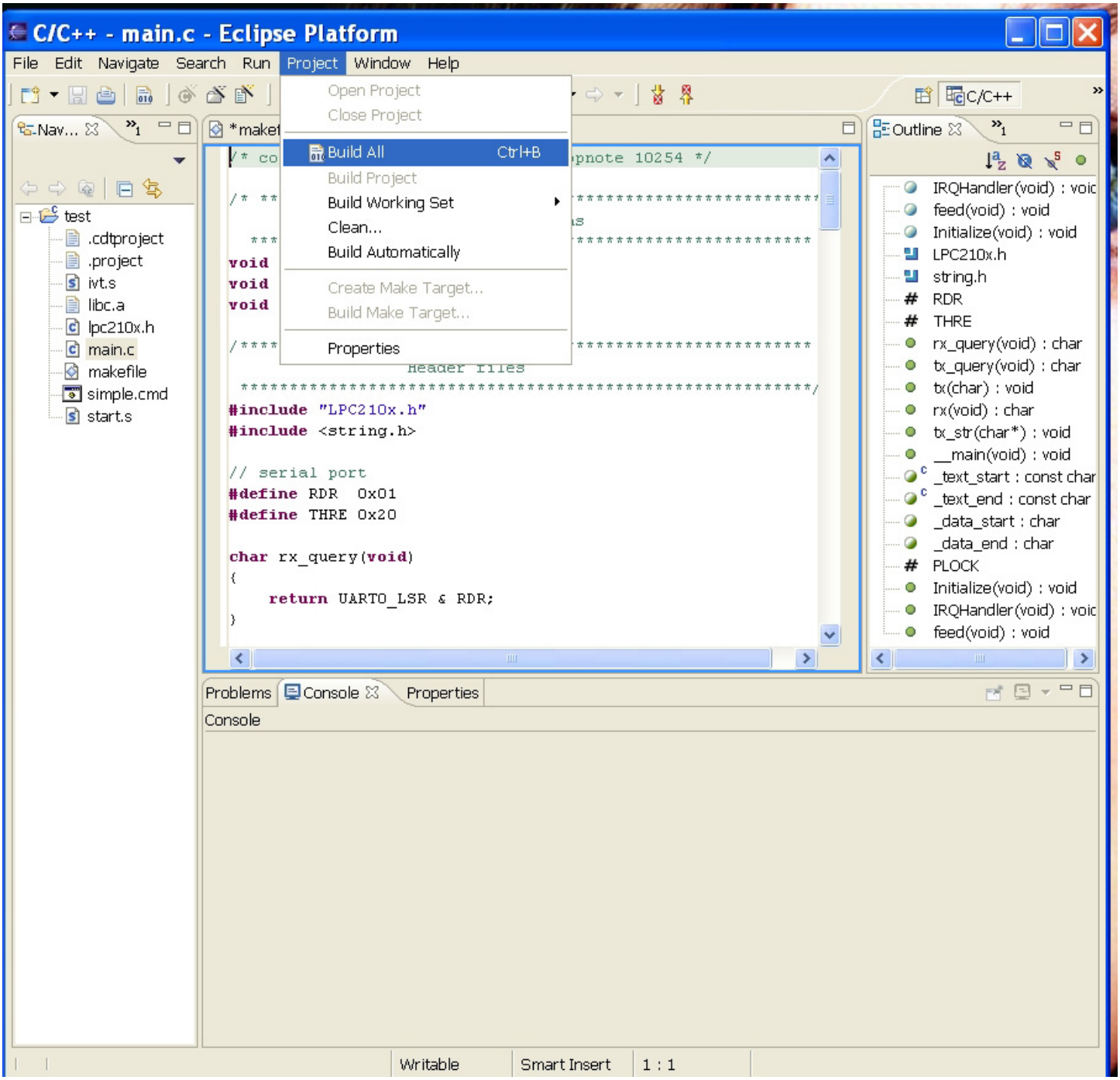


The first line "**all: test**" just runs the target test when Eclipse invokes the "**all**" target.

The second and third lines implement the "**clean**" target. The third line deletes the object files, the .out and .hex files and the map. Remember that the third line is indented by a **tab** character (not spaces). Forgetting the **tab** indent is the most common of all makefile mistakes.

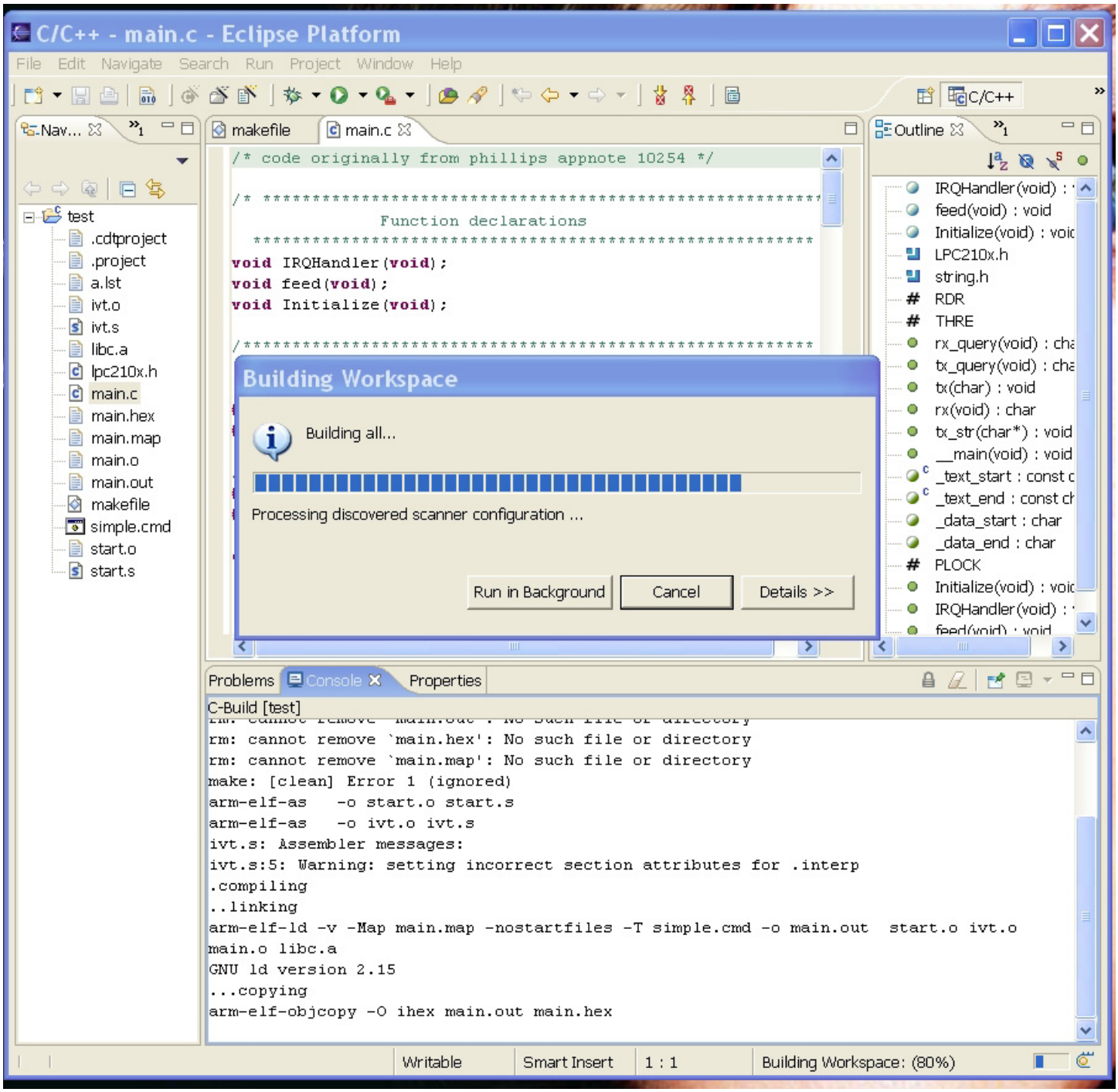
9. Building the Application

We can now build the project (compile, assemble and link all the source files) by selecting "**Project – Build All**".



When you invoke "**Build – All**", Eclipse will run the makefile as "**make clean all**." This means that it will erase the objects and load files and then compile, assemble and link everything.

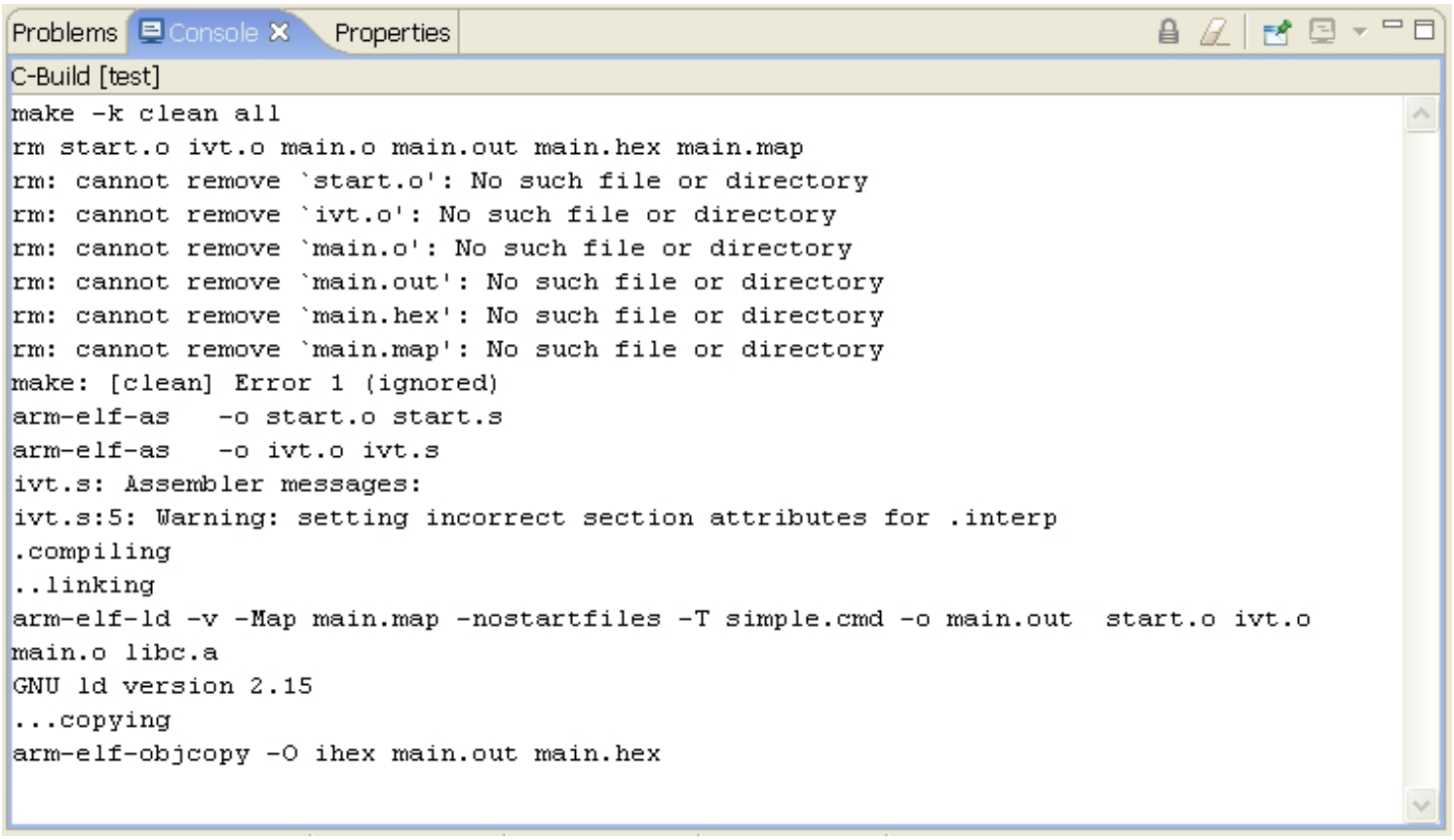
Here is a screen capture as Eclipse is building the ARM target software.



Here is the console display of the build process. Since we started with no object or load files, the **rm** operation indicated no such files to remove.

There was a mild assembler warning, but it did not stop the build process.

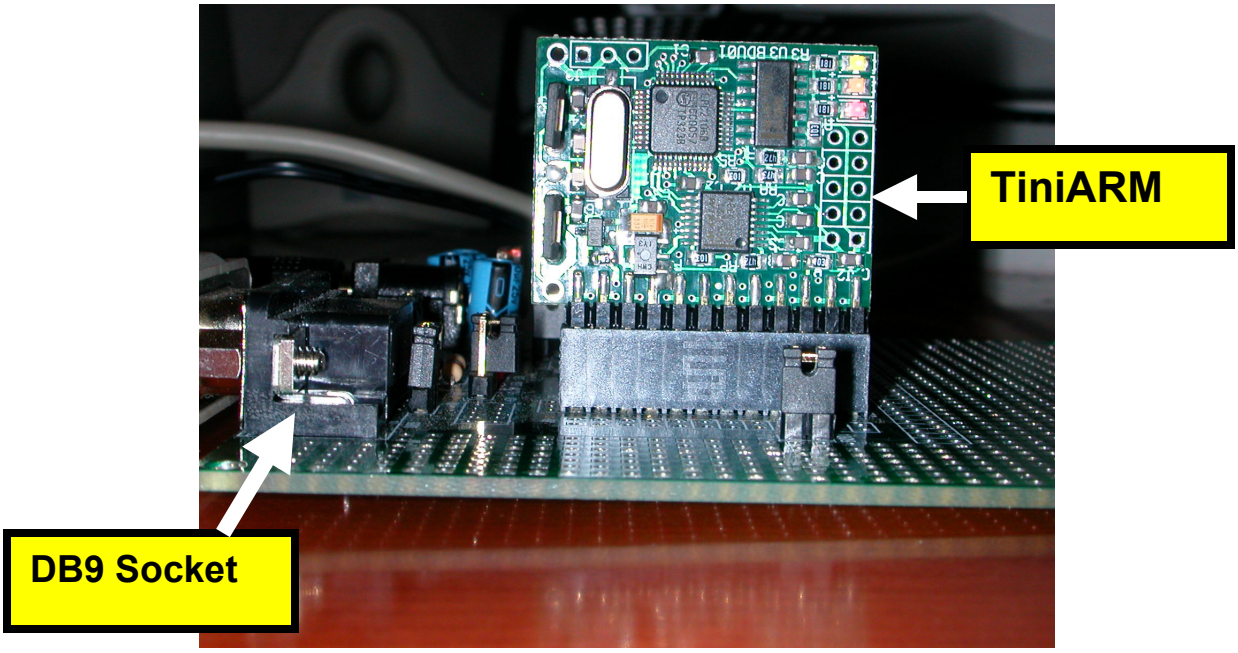
Note the last line, where the GNU utility **objcopy** is used to create a hex file suitable for the LPC2000 Flash Utility.



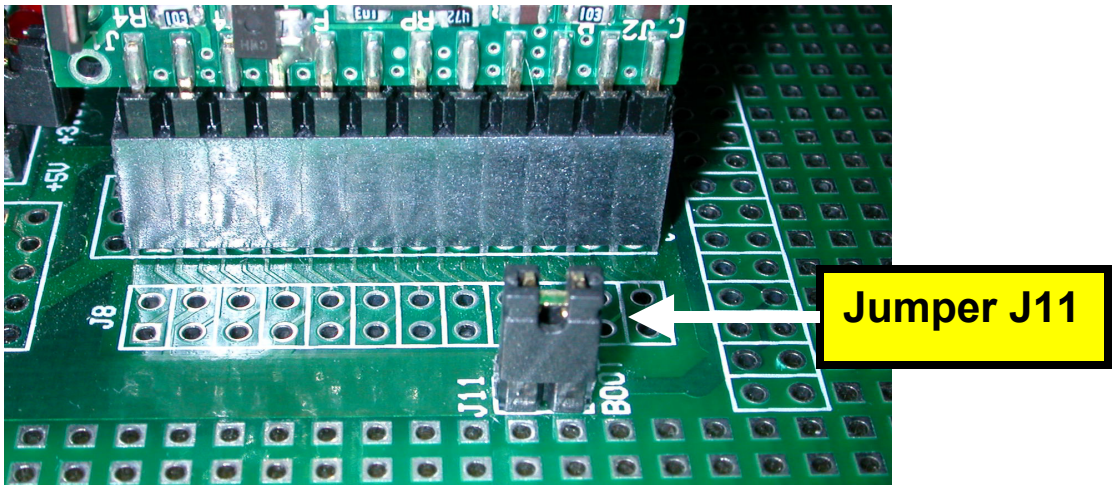
10. Downloading the Application to the TiniARM Board

Before starting the Philips lpc2000 Flash Utility, make sure that the hardware is properly hooked up and configured.

A. TiniARM board plugged into socket J1A with crystal closest to the 9-pin RS-232 socket.

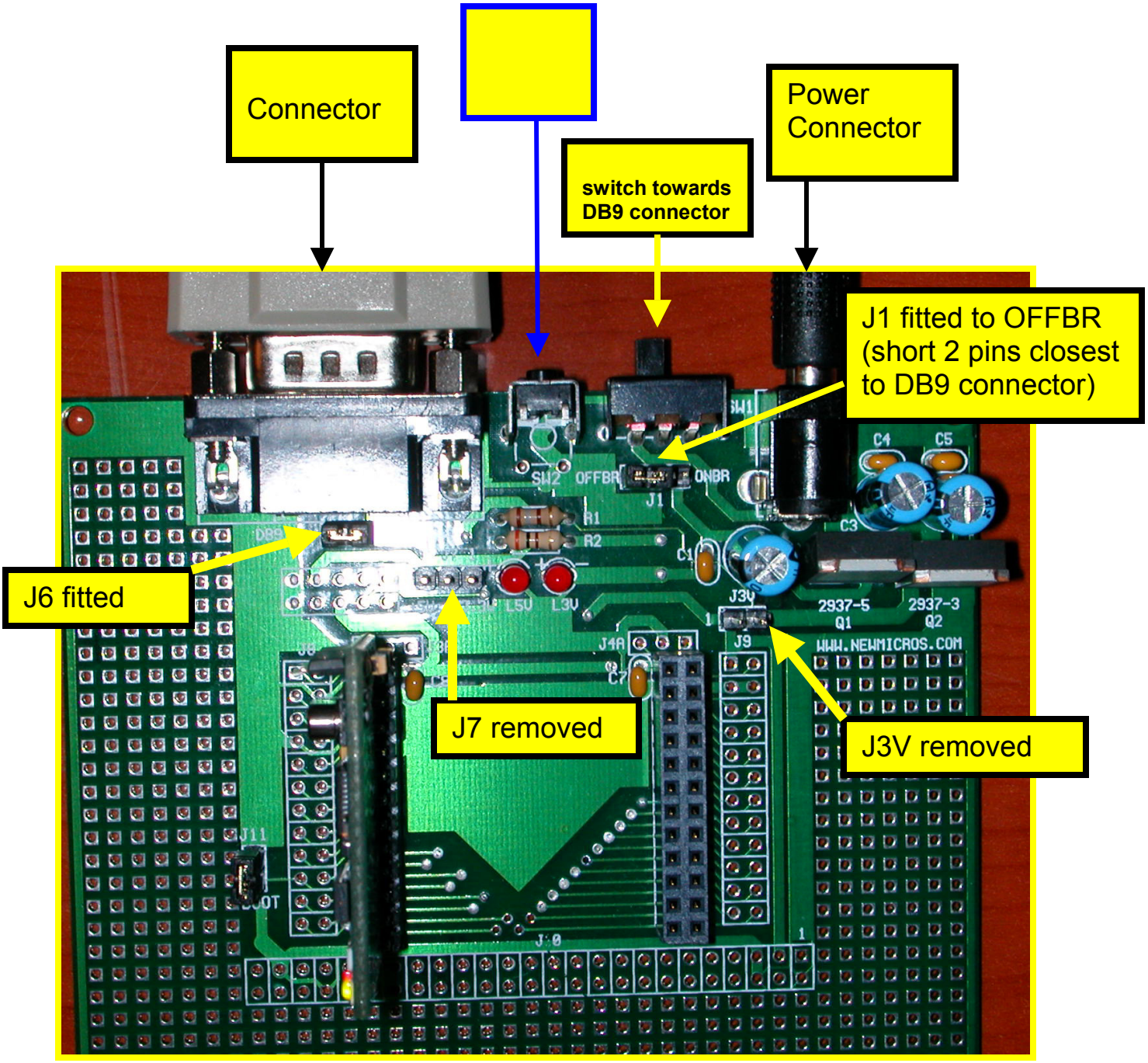


B. Jumper J11 installed for downloading



Install this jumper for download; **remove** this jumper for normal execution of the application.

A. Check all the other jumpers.



B. Attach Serial Cable

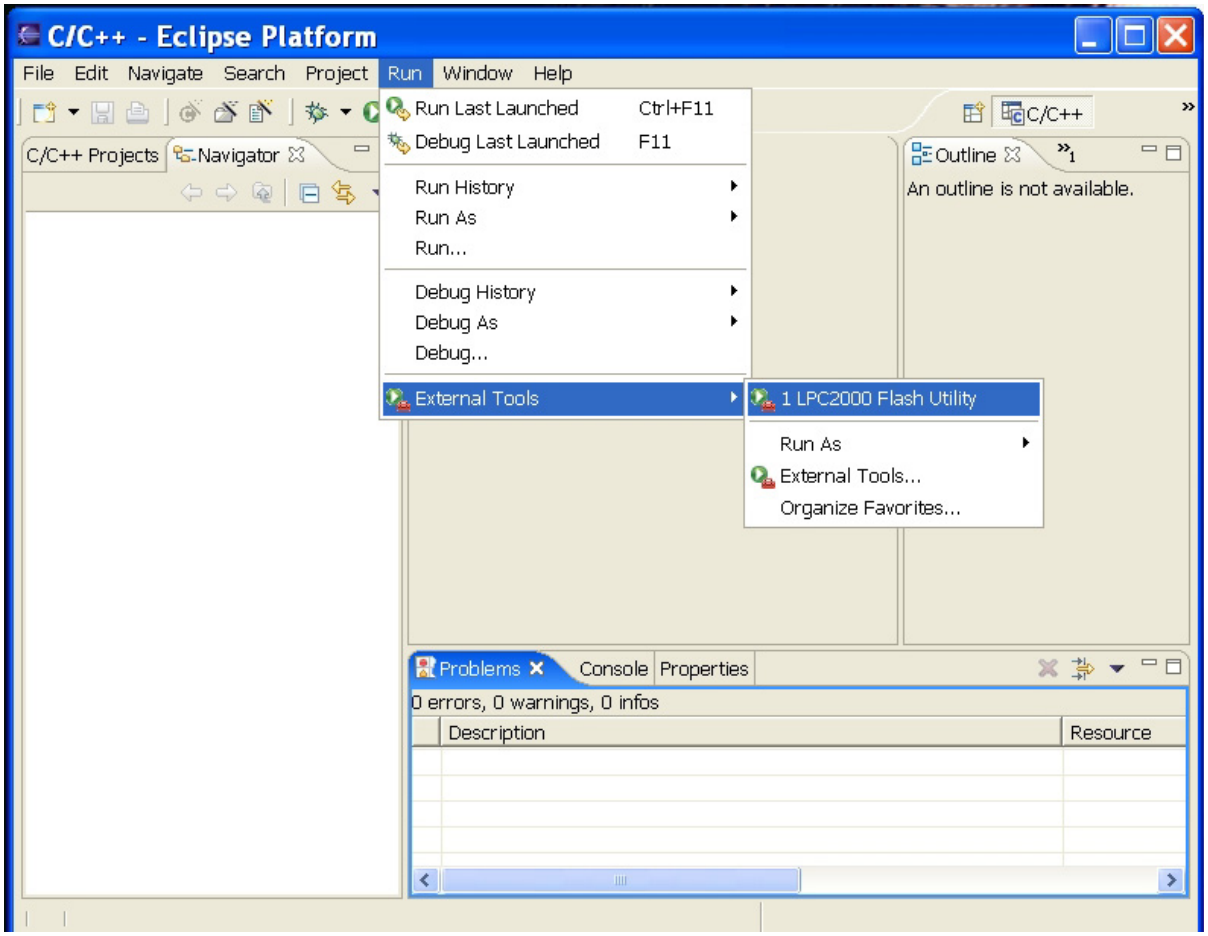
Attach the New Micros supplied 9-pin serial cable from COM1 serial port of your computer to the DB9 connector on the TiniARM Development Board.

E. Attach the 6-volt Power Supply

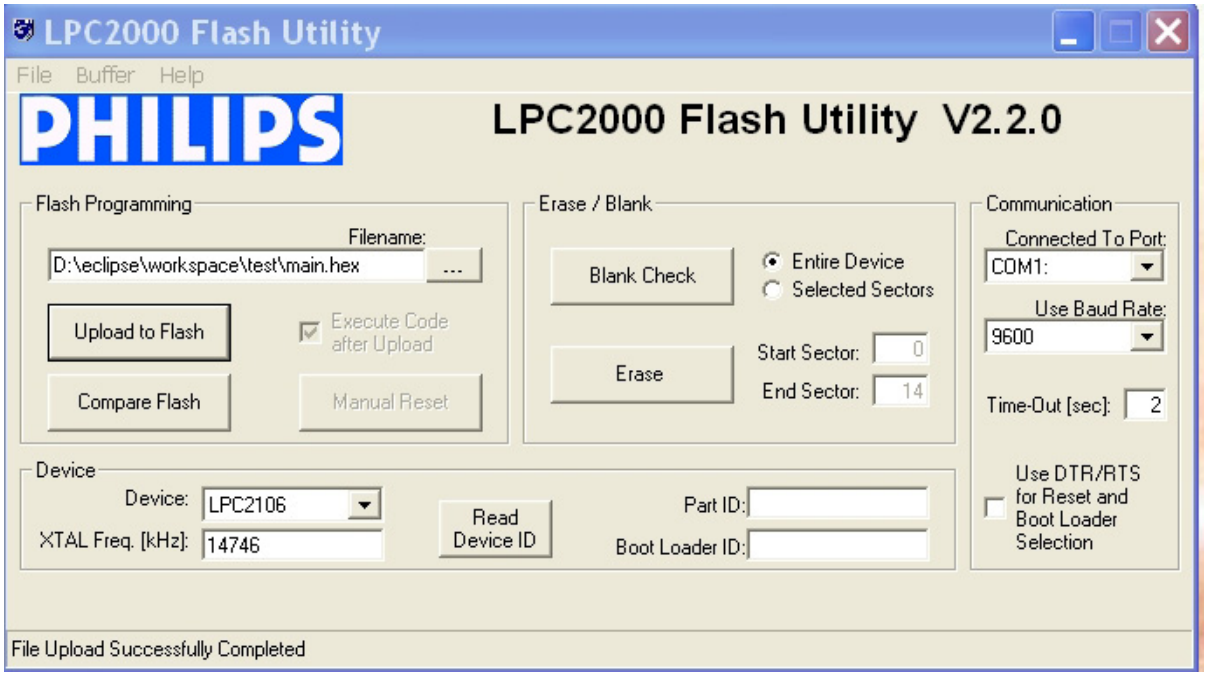
Plug in the “wall wart” power supply and observe that all three LEDs on the TiniARM are illuminated.

After double-checking that the “download” jumper J11 is installed, we can now attempt to download our hex file to the TiniARM board.

Select **Run – External Tools – LPC2000 Flash Utility** to start the Philips flash loader.



The Philips LPC2000 Flash Loader will start and appear on top of the Eclipse Screen, as shown below.



In the LPC2000 setup screen above, note that the correct hex file has been selected, thanks to our earlier setup work. Note also that the device is set to LPC2106 (very important) and the baud rate is set to 9600 baud; to match our Windows COM1 setup. You can change the baud rate to a faster one at your convenience.

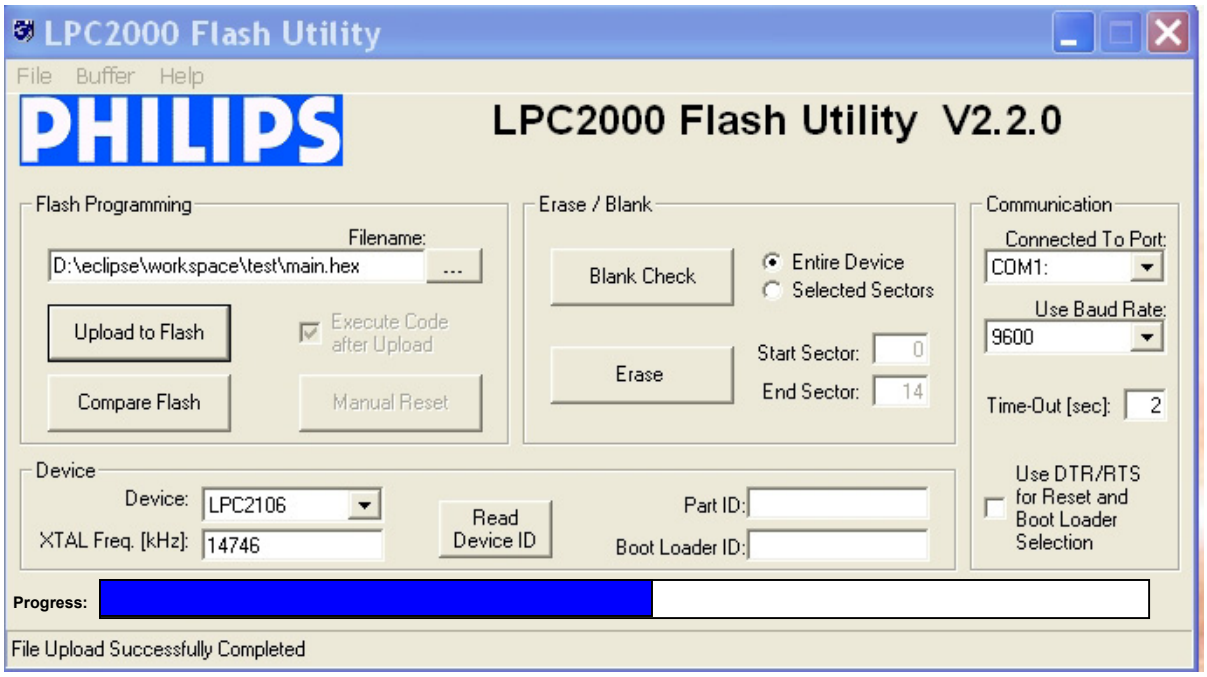
Push the RESET button **SW2** on the TiniARM board.

Click on "**Upload to Flash**" to start the download operation.

If you didn't press the RESET button SW2 you will get this reminder to do it.

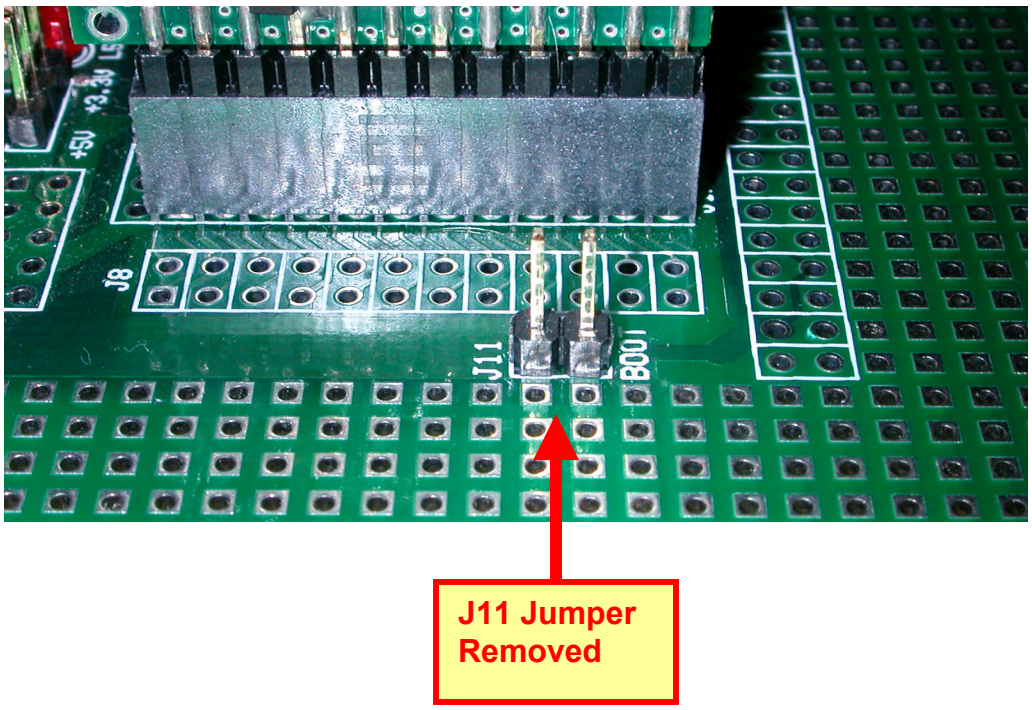
Now downloading into Flash should commence.

A progress bar will show it downloading into Flash.



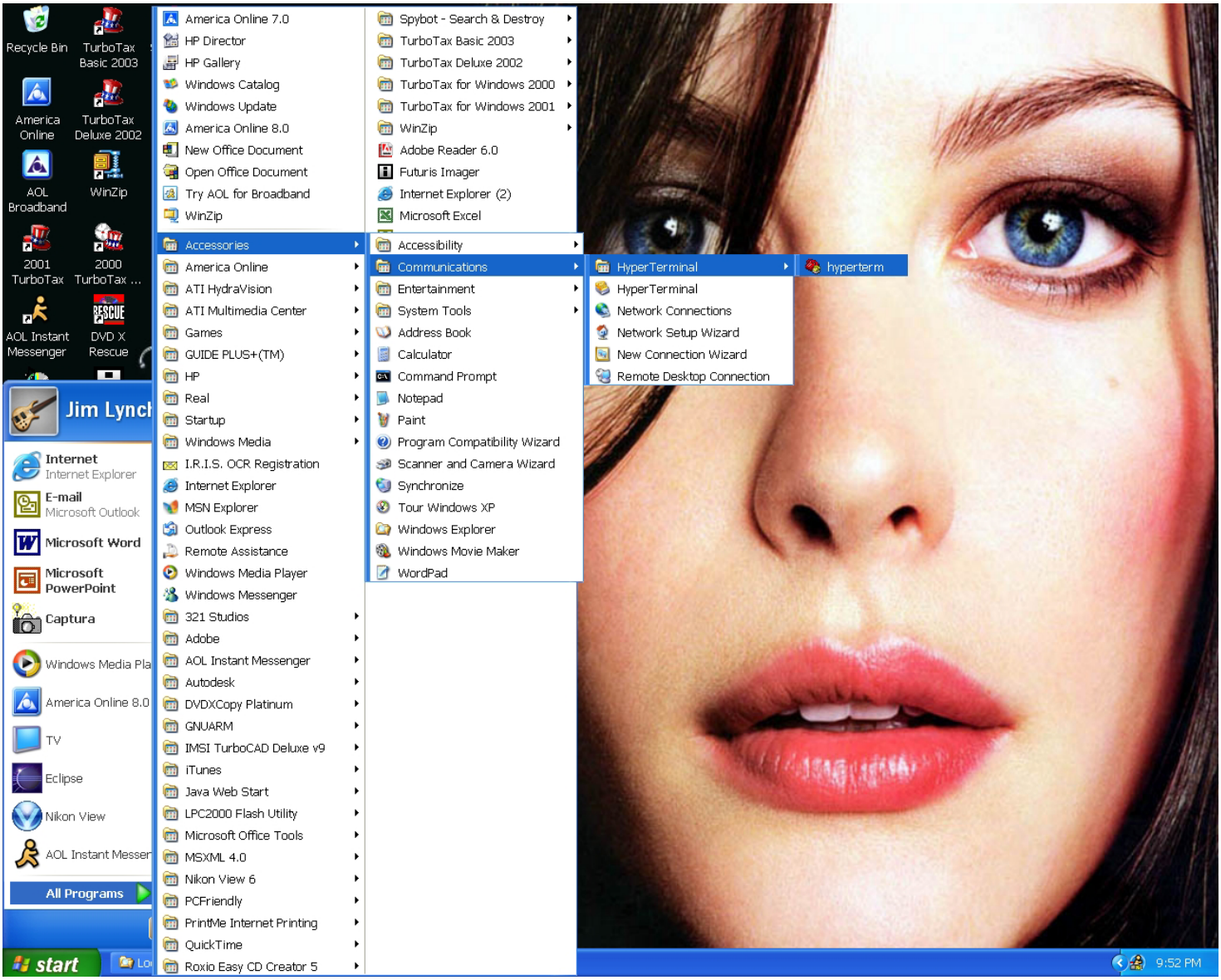
11. Testing the Sample Application

The first thing to do is to **REMOVE THE J11 BOOT JUMPER**!



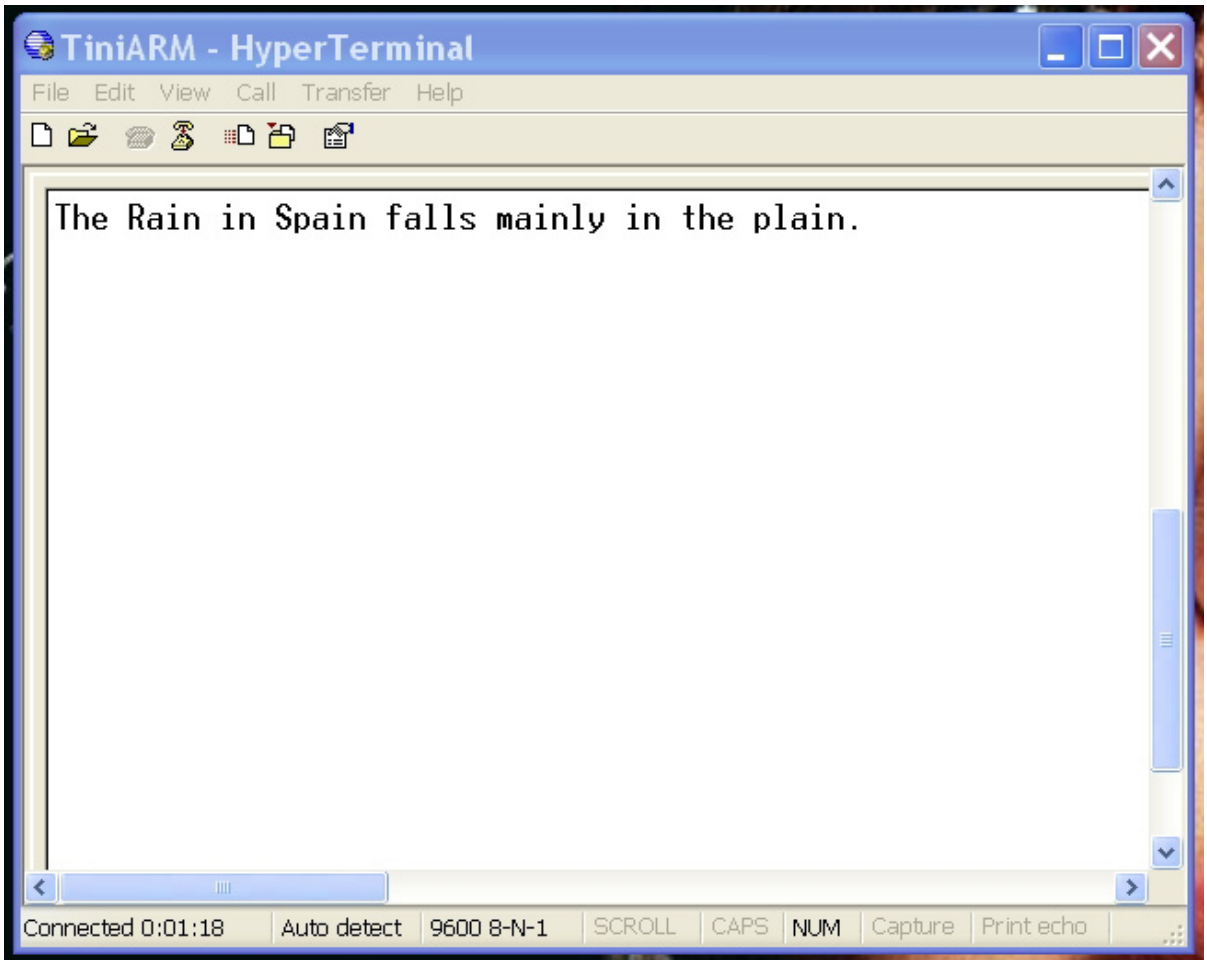
Now hit the **RESET** button again, just to be sure.

Go to the Windows XP desktop and use the **START** menu to run the accessory program **HyperTerm**.



Start the HyperTerminal and set it up to operate over COM1 with the settings of 9600 baud, no parity, 8 data bits etc.

If the TiniARM application is running properly, anything you type on HyperTerm will be echoed back from the TiniARM, as shown below.



Congratulations, you have successfully installed an Eclipse IDE for cross-development of a ARM processor and actually compiled, linked and downloaded an executable to the TiniARM board.

Pat yourself on the back!

About the Author

Jim Lynch lives in Grand Island, New York and is a Project Manager for Control Techniques, a subsidiary of Emerson Electric. He develops embedded software for the company's industrial drives (high power motor controllers) which are sold all over the world.

Mr. Lynch has previously worked for Mennen Medical, Calspan Corporation and the Boeing Company. He has a BSEE from Ohio University and a MSEE from State University of New York at Buffalo.

Jim is a single Father and has two children who now live in Florida and Nevada. He has two brothers, one is a Viet Nam veteran in Hollywood, Florida and the other is the Bishop of St. Petersburg, also in Florida.

Jim plays the guitar and is collecting woodworking machines for future projects that will integrate woodworking and embedded computers.

Lynch can be reached via e-mail at: lynchzilla@aol.com