Compilando una aplicación para ARM con Eclipse

Por. Paul Aguayo S.
www.olimex.cl
Basado en el turorial de James P. Lynch

Introducción

Hace un par de años atrás había 5 billones de microcontroladores a lo largo del planeta. Solo 300 millones de éstos se encuentran en computadores de escritorio PC's. El resto de ellos se encuentra en tostadoras, autos, aviones, portones eléctricos, aspiradoras, etc. Aquí es donde está la verdadera acción para ingenieros eléctricos y programadores.

La mayoría de la gente que empieza con microcontroladores se desenvuelve en el mundo de los PIC. Hay muchos sitios en la web que son devotos a este microcontrolador ya que son muy baratos hay; herramientas gratuitas para trabajar con ellos y Microchip ofrece todo el soporte necesario.

Ahora si quieres un microcontrolador en el estado del arte entonces debes jugar con los ARM. Texas Instruments, Philips y Atmel han desarrollado microcontroladores de 32-bits con arquitectura ARM de muy bajo costo. Estos chips integran memoria RAM y FLASH, un rico set de periféricos como serial I/O PWM, I2C, SSI, Timers etc. Además de altas prestaciones y bajo consumo.

Un muy buen ejemplo de este grupo es la familia Philips LPC2000. El LPC2106 posee las siguientes especificaciones y todo incluido en un paquete de 48 pines que cuesta alrededor de $11.80 USD

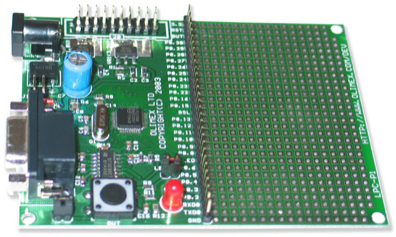
Principales características

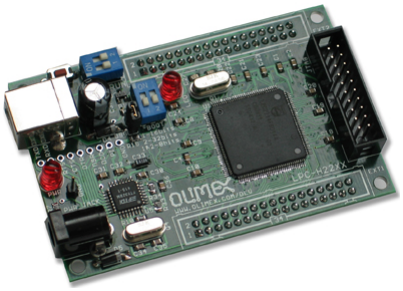
- 16/32-bit ARM7TDMI-S processor.
- 64 kB on-chip Static RAM.
- 128 kB on-chip Flash Program Memory. In-System Programming (ISP) and In-Application Programming (IAP) via on-chip boot-loader software.
- Vectored Interrupt Controller with configurable priorities and vector addresses.
- JTAG interface enables breakpoints and watch points.
- Multiple serial interfaces including two UARTs (16C550), Fast I²C (400 kbits/s) and SPI™.
- Two 32-bit timers (7 capture/compare channels), PWM unit (6 outputs), Real Time Clock and Watchdog.

- Up to thirty-two 5 V tolerant general-purpose I/O pins in a tiny LQFP48 (7 x 7 mm^2) package.
- 60 MHz maximum CPU clock available from programmable on-chip Phase-Locked Loop with settling time of 100 us.
- On-chip crystal oscillator with an operating range of 1 MHz to 30 MHz.
- Two low power modes: Idle and Power-down.
- Processor wake-up from Power-down mode via external interrupt.
- Individual enable/disable of peripheral functions for power optimization.
- Dual power supply:
    - CPU operating voltage range of 1.65 V to 1.95 V (1.8 V +- 8.3 pct.).
    - I/O power supply range of 3.0 V to 3.6 V (3.3 V +- 10 pct.) with 5 V tolerant I/O pads.

Este tutorial ha sido diseñado para estudiantes y hobbistas que cuentan con recursos limitados. Describe en gran detalle como descargar e instalar todos los programas necesarios para poder desarrollar una aplicación con un microcontrolador ARM. Todos los programas utilizados son completamente gratuitos.

Olimex ha ensamblado placas de desarrollo que poseen este microcontrolador tales como la LPC-P2106 y la LPC-H2294





Veamos cual es el software necesario para editar, compilar, linkear y descargar aplicaciones al LPC2106.

El software para desarrollo de sistemas embebidos siempre ha sido considerado como “profesional” y el precio va de acuerdo a esto. Es común para un ingeniero de una compañía que se dedica a hacer desarrollos gastar entre $1000 USD y $5000 USD por un paquete de

desarrollo profesional. El software comercial para la arquitectura ARM es fácil de instalar, está bien documentado y rara vez presenta errores (bugs). De hecho la mayoría de ellos pueden descargar el programa en la RAM o en la FLASH y se pueden poner breakpoints en ambos casos al momento de hacer el debug. Los compiladores profesionales son más eficientes y generan un código compacto y rápido.

El Rowley CrossWorks recomendado por Olimex cuesta del orden de $900 USD, claramente está fuera del rango del estudiante y del experimentador.

Afortunadamente existen alternativas a estas herramientas profesionales como el "GNU toolset". GNU es el movimiento del software libre (open-source). Fue utilizado para crear el sistema operativo Linux. El GNU Toolset incluye compiladores, linkers y utilidades para la mayoría de los microprocesadores incluyendo la arquitectura ARM. El GNU toolset es gratuito.

Uno de los editores de código más utilizado por estos días es Eclipse open-source Integrated Development Enviroment (IDE). Agregando el plug-in CDT (C/C++ Development Toolkit), se puede editar fácilmente código C usando como compilador el GNU. Eclipse es también gratuito.

Philips provee una aplicación Windows que permite transferir un archivo hex creado por el compilador GNU en la memoria flash EPROM interna del LPC2106. Esta herramienta también es gratuita.

Macraigor ha desarrollado una aplicación Windows gratuita llamada OCDremote que permite a Eclipse/GDB (GNU Debugger) acceder a la memoria del microprocesador via el puerto JTAG utilizando un dispositivo llamado **"Wiggler"**. Se obtienen mejores resultados utilizando el debugger open-source **Insight** en vez del debugger que viene incluido con Eclipse; sin embargo sólo se puede utilizar con programas que corren desde la memoria RAM

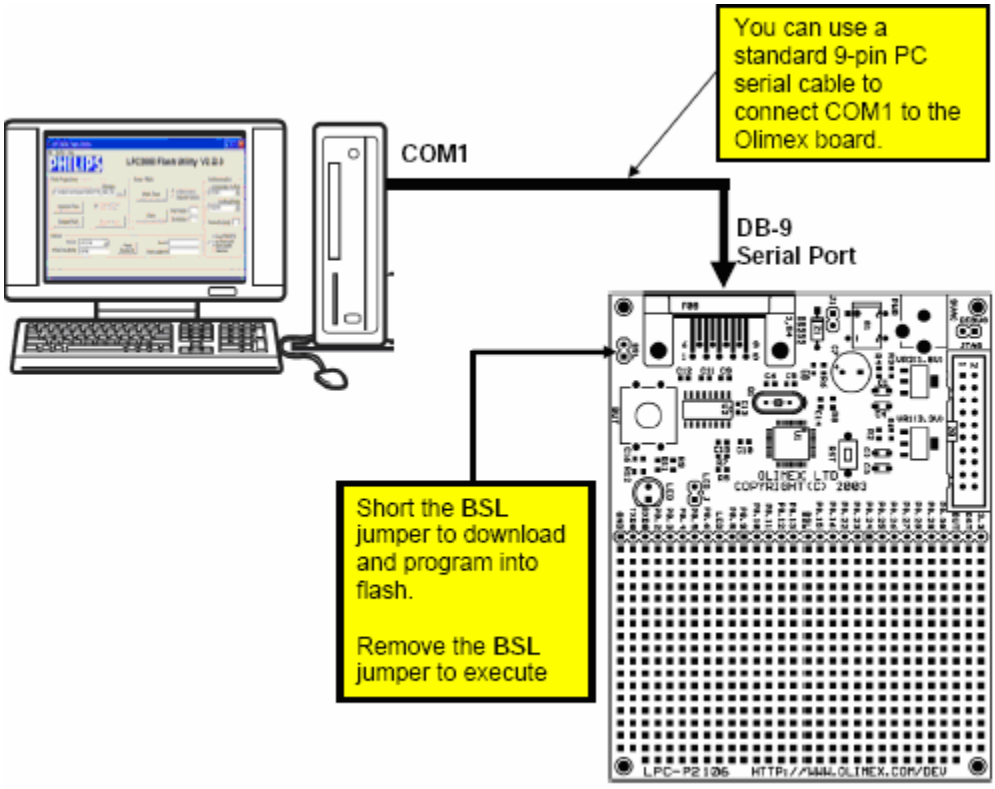
Hasta este punto probablemente estés diciendo, bien! Todas estas herramientas son gratuitas. En honor a la verdad veamos que es lo que realmente nos ofrecen estas herramientas open-source.

- Las herramientas GNU actualmente no generan un código tan eficiente como el de los compiladores profesionales.
- El Insight Debugger no puede poner breakpoints en la memoria FLASH ya que no puede borrar ni programar la FLASH

– El OCDRemote no soporta breakpoints de hardware.

Crearemos entonces un set de herramientas Eclipse/GNU y dividiremos este tutorial en 2 partes:

A. Aplicación Programada en la FLASH

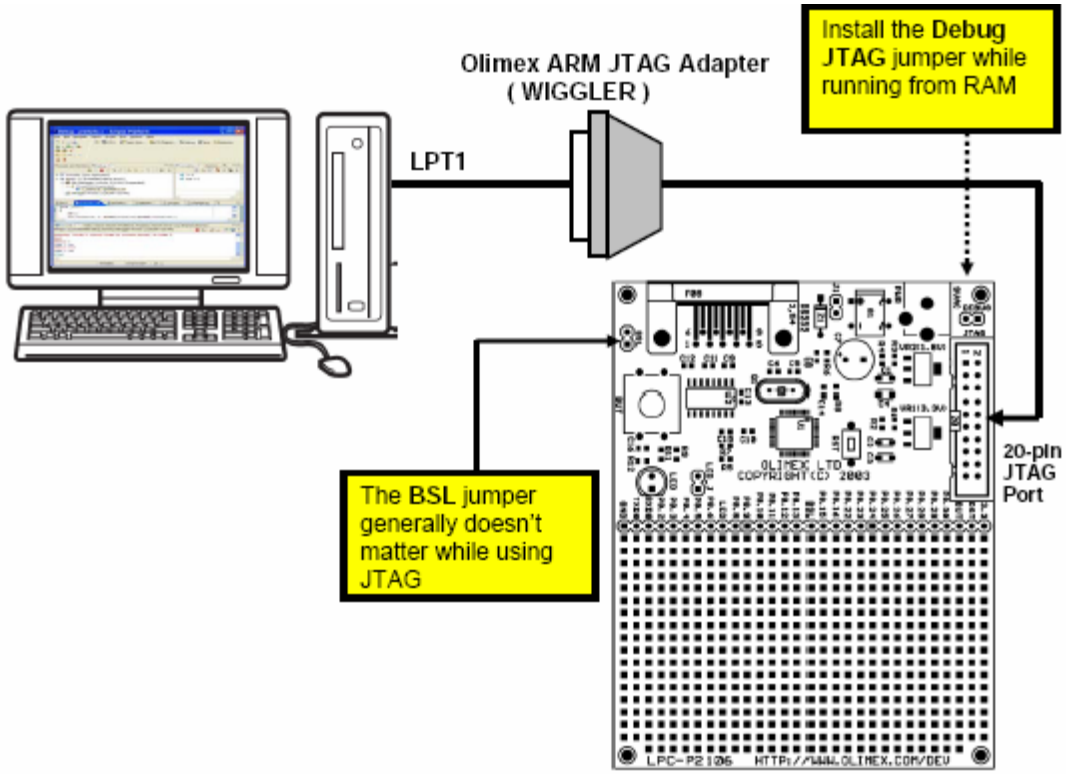


En este modo, Eclipse/GNU ensambla, compila y linkea la aplicación para programarla en la memoria FLASH. A la salida de este proceso tenemos un archivo en formato Intel hex, por ejemplo main.hex

La aplicación Philips In-System Programming (ISP) se corre desde Eclipse y permite descargar el archivo hex creado anteriormente en la memoria FLASH a través de un puerto estándar de comunicación serial (COM1 por ejemplo). El jumper Boot Strap Loader (BSL) debe estar puesto antes de correr la aplicación ISP.Para ejecutar la aplicación hay que remover el jumper BSL y apretar el botón RESET para empezar con la aplicación.

Desafortunadamente, el Insight debugger no puede setear un breakpoint de software (ya que no puede programar la FLASH) y tampoco soporta los breakpoints de hardware, por lo que no hay posibilidad de hacer un debug del programa **en este modo.**

B. Aplicación Programada en la RAM



En este modo el Eclipse/GNU ensambla, compila y linkea la aplicación para cargarla en la memoria RAM. La salida después de este proceso de compilación y link es un archivo del tipo main.out.

El PC es conectado por el puerto paralelo LPT1 al JTAG utilizando el ARM-JTAG de Olimex. El ARM-JATG de Olimex es un clon del Macraigor Wiggler.

Puedes correr el programa OCDRemote como una herramienta externa desde Eclipse. El Insight debugger comunica el Macraigor OCDRemote con el JTAG. Con el Insight debugger, puede conectar el Wiggler y cargar el archivo generado por el GNU main.out en la RAM. En este modo se pueden utilizar breakpoints de software, ver las variables y estructuras, y por supuesto correr la aplicación.

El problema con este modo es que la aplicación debe correr en la memoria RAM del LPC2106, la cual es de 64Kbytes.

El propósito de este tutorial es guiar a los estudiantes y a los hobbistas recopilando la documentación necesaria para tener funcionando un ambiente de desarrollo para la arquitectura ARM, para lo cual haremos

un programa simple que haga encender y apagar una luz. Hay dos variantes para este programa: la versión basada en la FLASH y la versión basada en la memoria RAM. A lo largo del tutorial se sustituirá el debugger que viene con Eclipse por el debugger Insight por que es más fácil de utilizar y más confiable.

Recomendación.

Si no tienes experiencia previa con los microcontroladores ARM no hay mejor libro introductorio que el "**The Insider's Guide to the Philips ARM7-Based Microcontrollers**" por Trevor Martin.

Este libro se puede descargar gratuitamente desde el sitio web de Hitex http://www.hitex.co.uk/arm/lpc2000book/index.html

Instalando los componentes necesarios

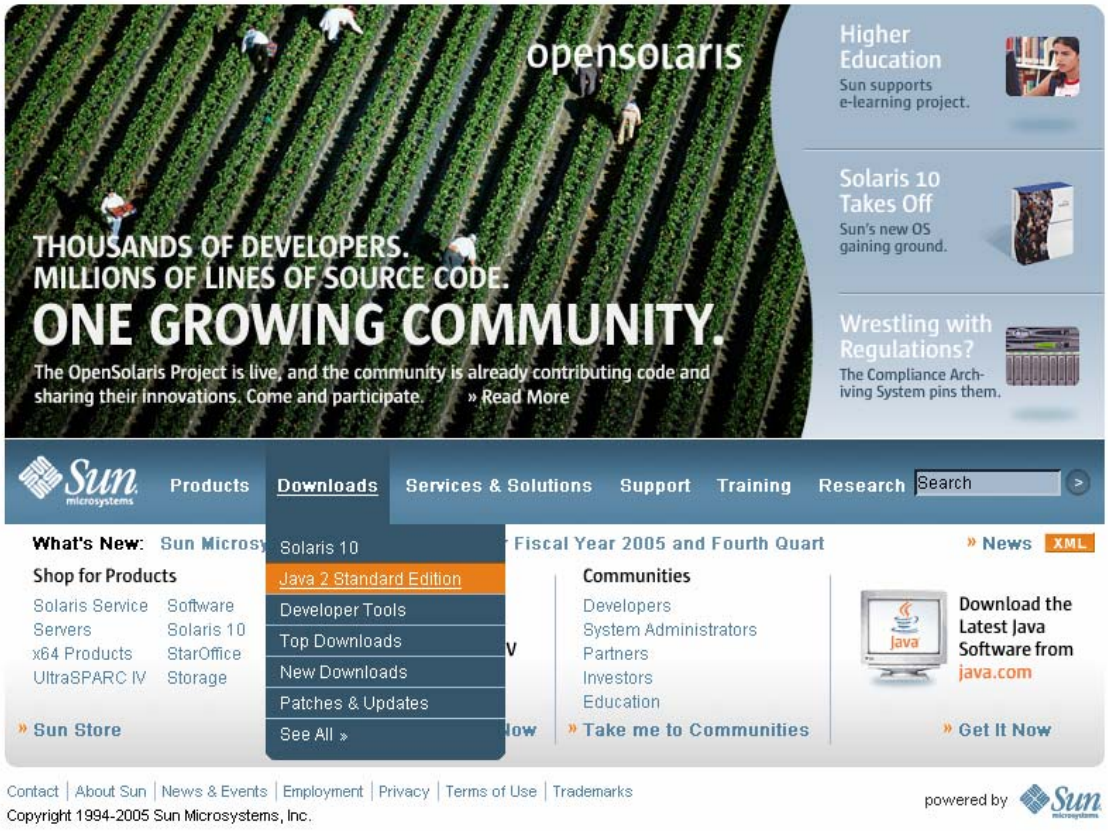
Para preparar el ambiente de desarrollo usando Eclipse es necesario descargar e instalar las siguientes aplicaciones:

1. SUN Java Runtime
2. Eclipse IDE
3. Eclipse CDT Plug-in for C++/C
4. CYGWIN GNU C++/C Compiler and Toolset for Windows
5. GNUARM GNU C++/C Compiler for ARM Targets
6. GNUARM Insight Debugger
7. Philips Flash Programmer for LPC2100 Family CPUs
8. Macraigor OCDremote for JTAG debugging

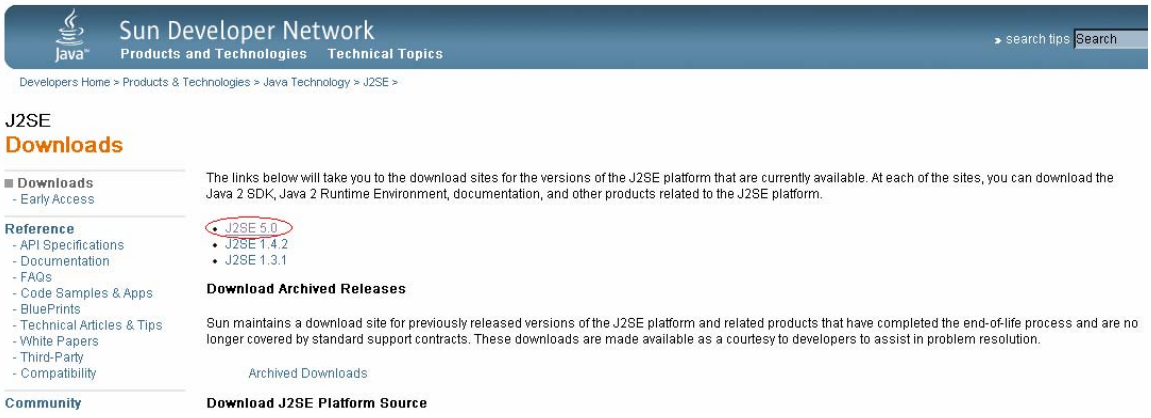
Java Runtime

Eclipse fue escrito completamente en JAVA, luego para que éste funcione correctamente se debe tener instalado JAVA runtime. La mayoría de la gente ya lo tiene instalado en su computador con sistema operativo Windows, pero en caso de que no lo tengas instalado aquí se muestra como hacerlo.

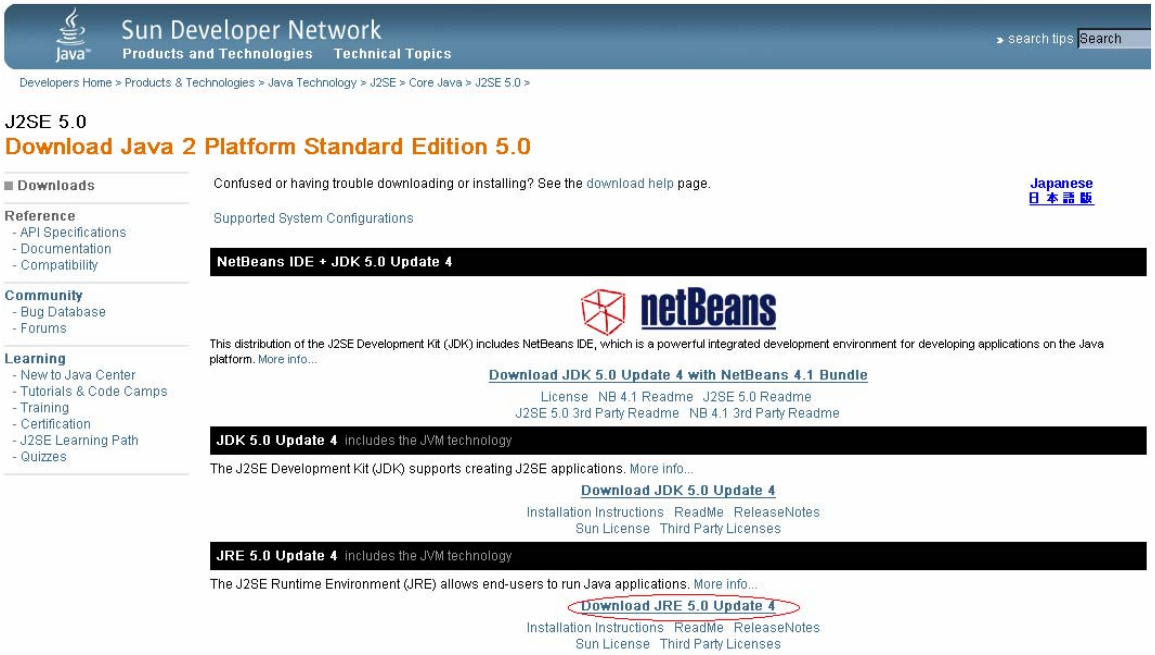
Entra a la página www.sun.com. La siguiente pantalla aparecerá:



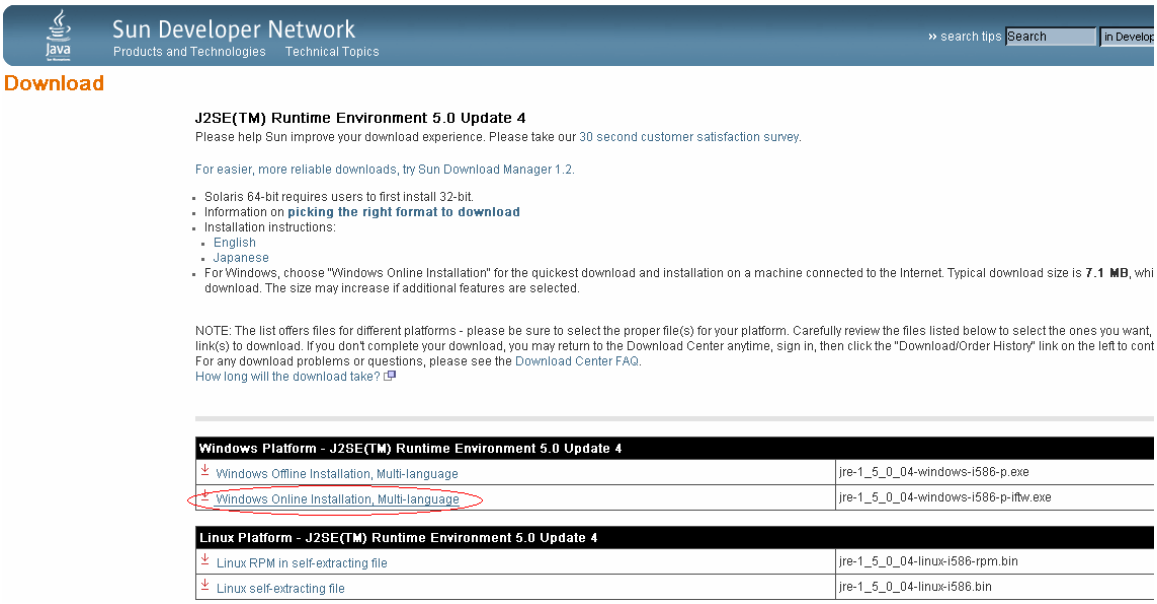
Haz click en “Donwloads –> Java 2 Standard Edition” para continuar. Luego selecciona la última versión.



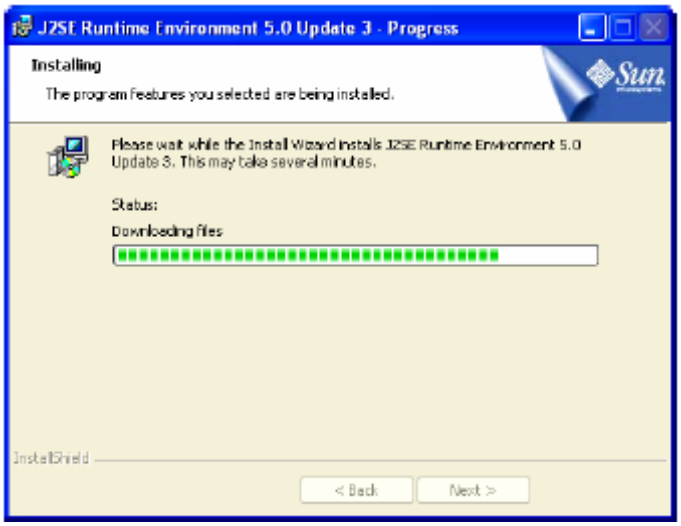
Específicamente necesitamos instalar el Java Runtime Enviroment (JRE), luego hay que hacer clic en “Download JRE 5.0 Update 3.”



Luego hay que aceptar la licencia de SUN para proceder y luego seleccionar la opción de instalación en línea para Windows



Una vez que termina la descarga la instalación comienza en forma automática.



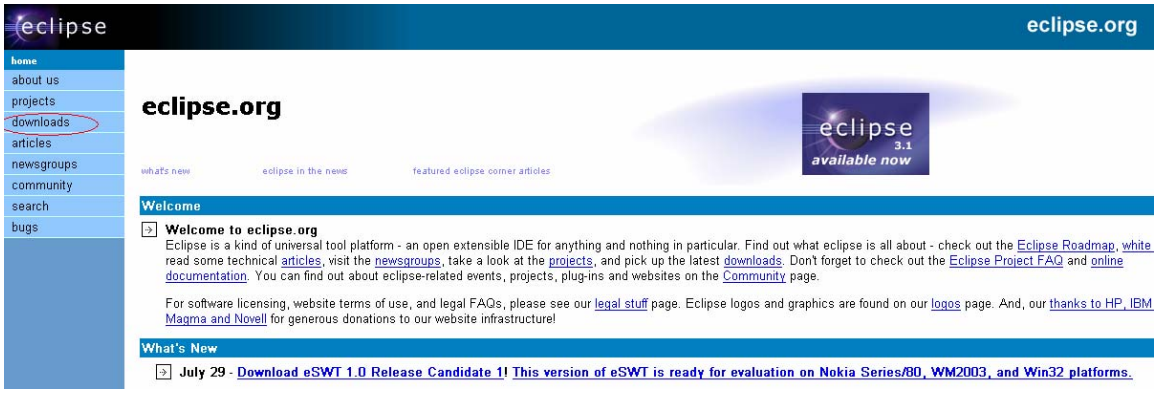
Instalación de Eclipse IDE

Eclipse IDE es una plataforma de desarrollo integrada similar a Visual Studio de Microsoft. Fue desarrollada originalmente por IBM y ha sido donada a la comunidad open source.

Eclipse, por si solo, esta configurado para editar y debuggear programas en JAVA. Instalando el plug-in de CDT se puede usar Eclipse para editar y debuggear programas escritos en C/C++. (lo cual veremos más adelante)

Cuando éste configurado correctamente lo utilizaremos para compilar y debuggear aplicaciones ARM.

Eclipse se puede descargar gratuitamente en el sitio www.eclipse.org

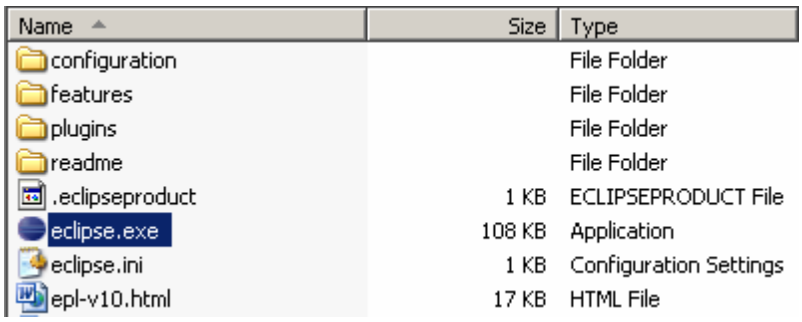


Haz clic en “download” para comenzar la instalación

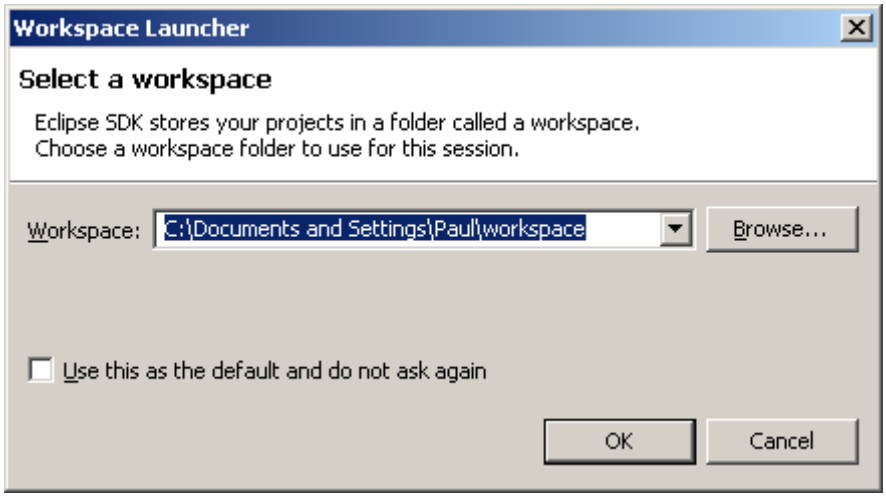
Luego selecciona el mirror que más cercano.

Cuando se trabaja con Eclipse y CDT hay que asegurarse de que la versión del plug-in de CDT que se seleccionó sea compatible con la versión de Eclipse que se va a seleccionar. La información de compatibilidad se encuentra en el sitio web de eclipse.

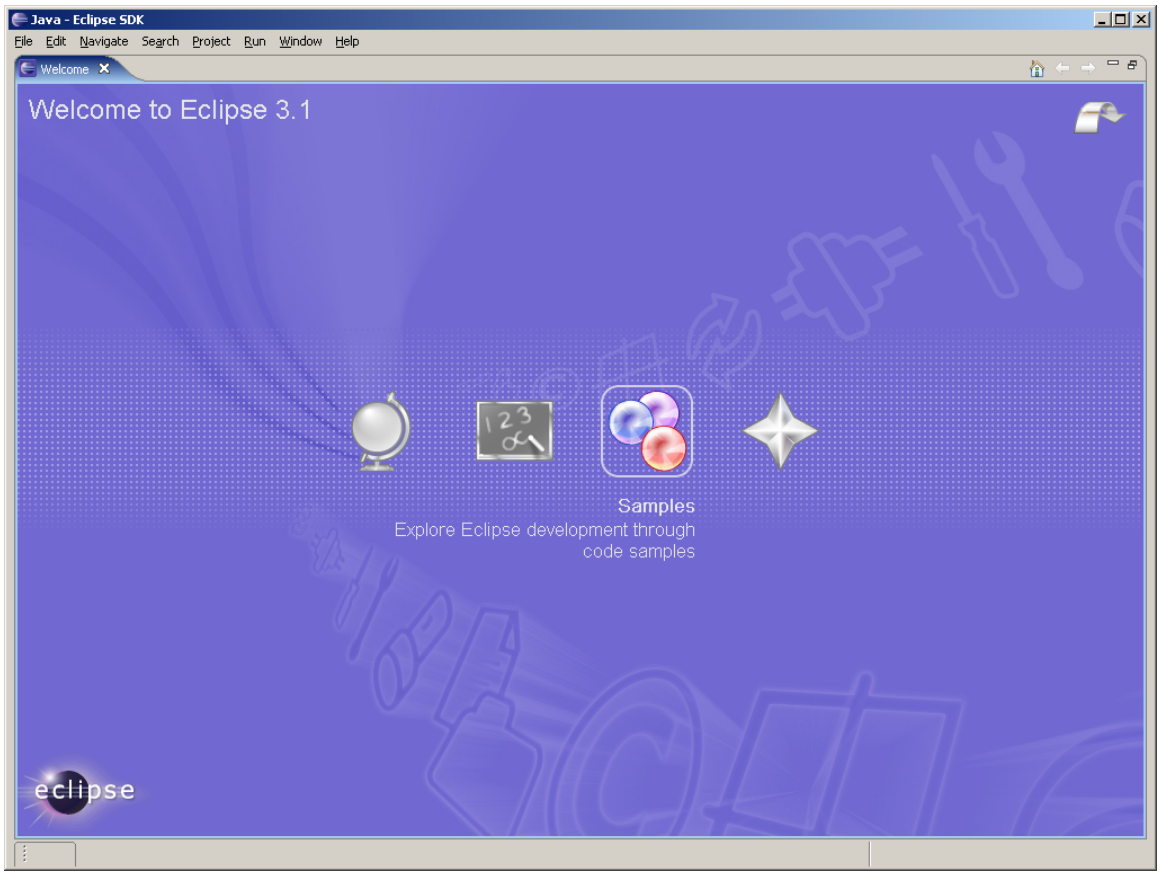
Una vez descomprimido hay que hacer click en el archivo eclipse.exe



Lo primero que aparece es una ventana que pregunta por el espacio de trabajo (Workspace). Puede ser cualquier lugar de tu disco duro.



Después de seleccionar el espacio de trabajo aparecerá la ventana principal de Eclipse. Con esto ya tenemos instalado Eclipse en nuestro computador. Ahora falta integrar CDT para poder trabajar con programas en C con Eclipse.



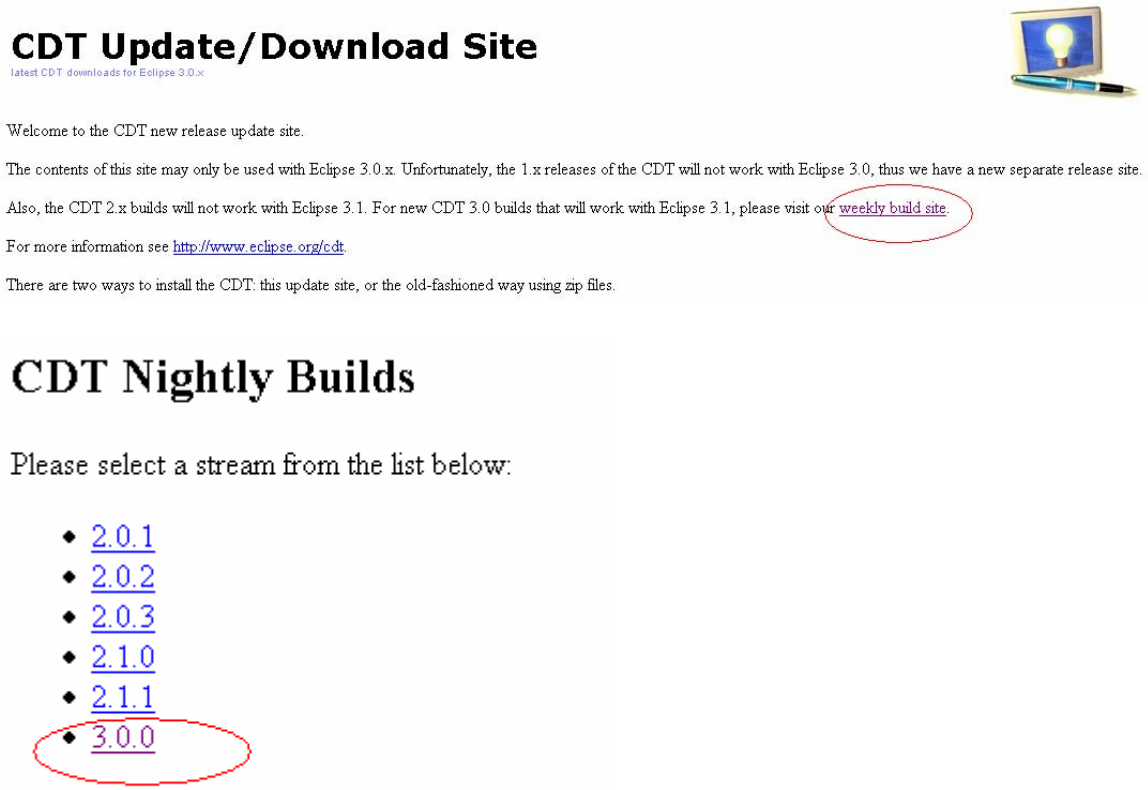
Eclipse CDT

Eclipse esta diseñado para editar y debuggear programas en JAVA. Para que este sea capaz de trabajar con programas en C y C++ es necesario descargar el CDT (C Development Toolkit). El CDT es una colección de archivos que se instalan dentro del directorio de eclipse.

El plug-in CDT debe ser compatible con la versión de Eclipse que acabamos de instalar. Para verificar esto sigue el siguiente link:

http://update.eclipse.org/tools/cdt/releases/new

En mi caso utilizaré la versión de CDT 3.0 ya que es compatible con la versión de Eclipse 3.1



CDT Update/Download Site

latest CDT downloads for Eclipse 3.0.x

Welcome to the CDT new release update site.

The contents of this site may only be used with Eclipse 3.0.x. Unfortunately, the 1.x releases of the CDT will not work with Eclipse 3.0, thus we have a new separate release site.

Also, the CDT 2.x builds will not work with Eclipse 3.1. For new CDT 3.0 builds that will work with Eclipse 3.1, please visit our weekly build site.

For more information see http://www.eclipse.org/cdt.

There are two ways to install the CDT: this update site, or the old-fashioned way using zip files.

CDT Nightly Builds

Please select a stream from the list below:

- 2.0.1
- 2.0.2
- 2.0.3
- 2.1.0
- 2.1.1
- 3.0.0

CDT 3.0.0 Builds

Note that these builds run against Eclipse 3.1 only

Please select a build from the list below:

- **M5** (I200503211118)
- **M6** (I200504140301)
- **M7** (I200506171115)
- **RC1** (I200506291405)
- I200507111111
- **RC2** (I200507121024)
- I200507180855
- I200507251030

CDT Runtime Feature

includes editor, search, builders, launch, debug, gnu toolchain integrations for build/debug, user documentation

AIX/ppc	org.eclipse.cdt-3.0.0-I200507251030-aix.ppc.tar.gz
Linux/x86	org.eclipse.cdt-3.0.0-I200507251030-linux.x86.tar.gz
Linux/x86_64	org.eclipse.cdt-3.0.0-I200507251030-linux.x86_64.tar.gz
Linux/ppc	org.eclipse.cdt-3.0.0-I200507251030-linux.ppc.tar.gz
Linux/ia64	org.eclipse.cdt-3.0.0-I200507251030-linux.ia64.tar.gz
MacOSX/ppc	org.eclipse.cdt-3.0.0-I200507251030-macosx.ppc.tar.gz
QNX/x86	org.eclipse.cdt-3.0.0-I200507251030-qnx.x86.tar.gz
Solaris/sparc	org.eclipse.cdt-3.0.0-I200507251030-solaris.sparc.tar.gz
Win32/x86	org.eclipse.cdt-3.0.0-I200507251030-win32.x86.zip

CDT SDK Feature

superset of runtime that adds source and extension point schemas

AIX/ppc	org.eclipse.cdt.sdk-3.0.0-I200507251030-aix.ppc.tar.gz
Linux/x86	org.eclipse.cdt.sdk-3.0.0-I200507251030-linux.x86.tar.gz
Linux/x86_64	org.eclipse.cdt.sdk-3.0.0-I200507251030-linux.x86_64.tar.gz
Linux/ppc	org.eclipse.cdt.sdk-3.0.0-I200507251030-linux.ppc.tar.gz
Linux/ia64	org.eclipse.cdt.sdk-3.0.0-I200507251030-linux.ia64.tar.gz
MacOSX/ppc	org.eclipse.cdt.sdk-3.0.0-I200507251030-macosx.ppc.tar.gz
QNX/x86	org.eclipse.cdt.sdk-3.0.0-I200507251030-qnx.x86.tar.gz
Solaris/sparc	org.eclipse.cdt.sdk-3.0.0-I200507251030-solaris.sparc.tar.gz
Win32/x86	org.eclipse.cdt.sdk-3.0.0-I200507251030-win32.x86.zip

Descargamos los dos archivos correspondientes a la versión de Windows, el Runtime y el SDK. Descomprimimos los archivos con winzip u otro similar y copiamos las carpetas sobre el directorio eclipse. De esta forma tenemos instalada la versión CDT 3.0

Para verificar que CDT quedó correctamente instalado reiniciamos eclipse y vamos al menú "File -> New -> Project". Si todo está correcto entonces podremos ver que dentro de las opciones aparece C y C++.

CYGWIN GNU Toolset para Windows

El set de herramientas GNU es una implementación de código abierto (open-source) de un compilador universal; puede trabajar con C, C++, AD, FORTRAN y JAVA. Todos estos compiladores pueden ser utilizados en la mayoría de las plataformas (como el microcontrolador ARM de 32 bits).

Desafortunadamente para nosotros que utilizamos computadores basados en la plataforma INTEL/Microsoft, el GNU fue originalmente desarrollado e implementado para el sistema operativo Linux. Pero al rescate viene Cygwin. Esta compañía ha creado un set de librerías dinámicas para Windows que engañan al GNU y lo hace creer que está corriendo en una plataforma Linux.

Si instalamos el compilador GNU usando Cygwin podemos literalmente abrir una ventana de comando DOS y tipear un comando DOS como el siguiente:

```
>arm-elf-gcc  –g  –c    main.c
```

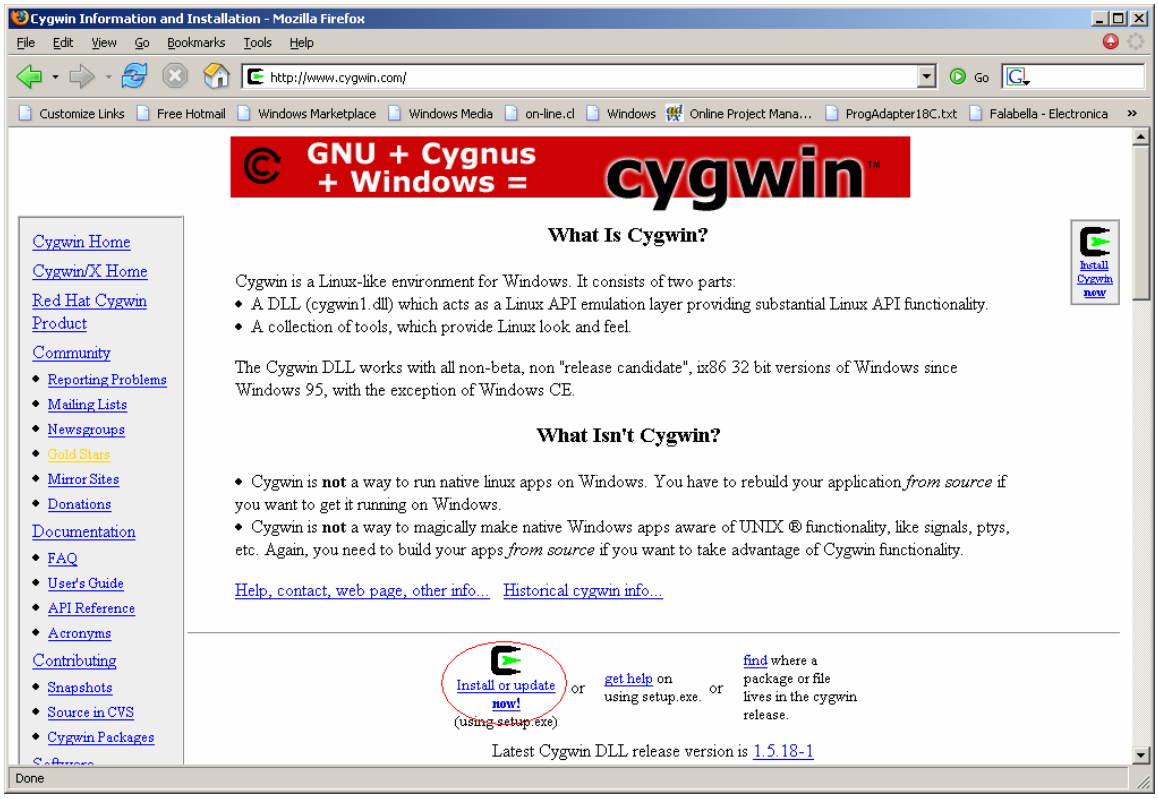
La instrucción anterior compilará el archivo main.c y generará un archivo objeto main.o para la arquitectura ARM. En otras palabras si instalas Cygwin GNU apropiadamente puedes olvidarte de que el compilador GNU esta basado en Linux.

Normalmente, la instalación de Cygwin nos da un completo set de herramientas para trabajar con la arquitectura Intel/Windows.

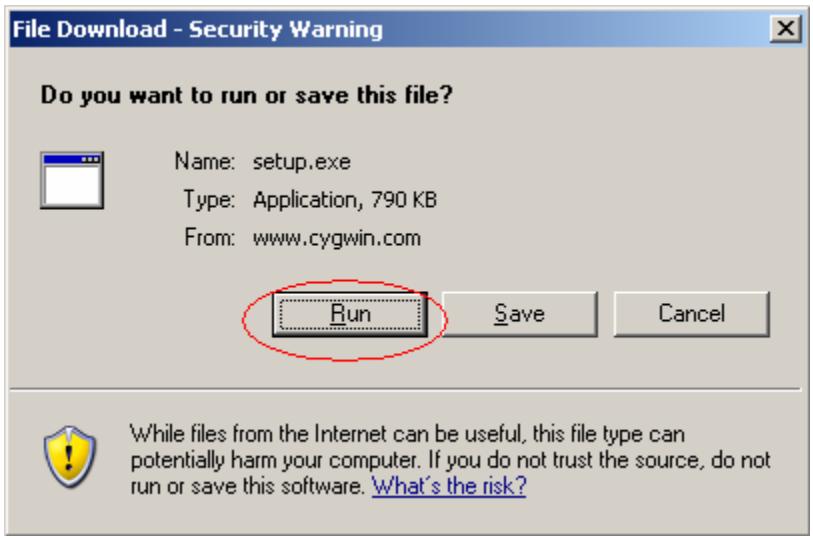
Es importante mencionar que GNUARM no incluye la utilidad **make**, Cygwin incluye esta utilidad. Por este motivo **es necesario** agregar a la PATH de Windows las siguientes carpetas c:/cygwin/bin y c:/programfiles/gnuarm/bin

El sitio donde se puede descargar Cygwin es www.cygwin.com. A continuación veremos como instalar Cygwin.

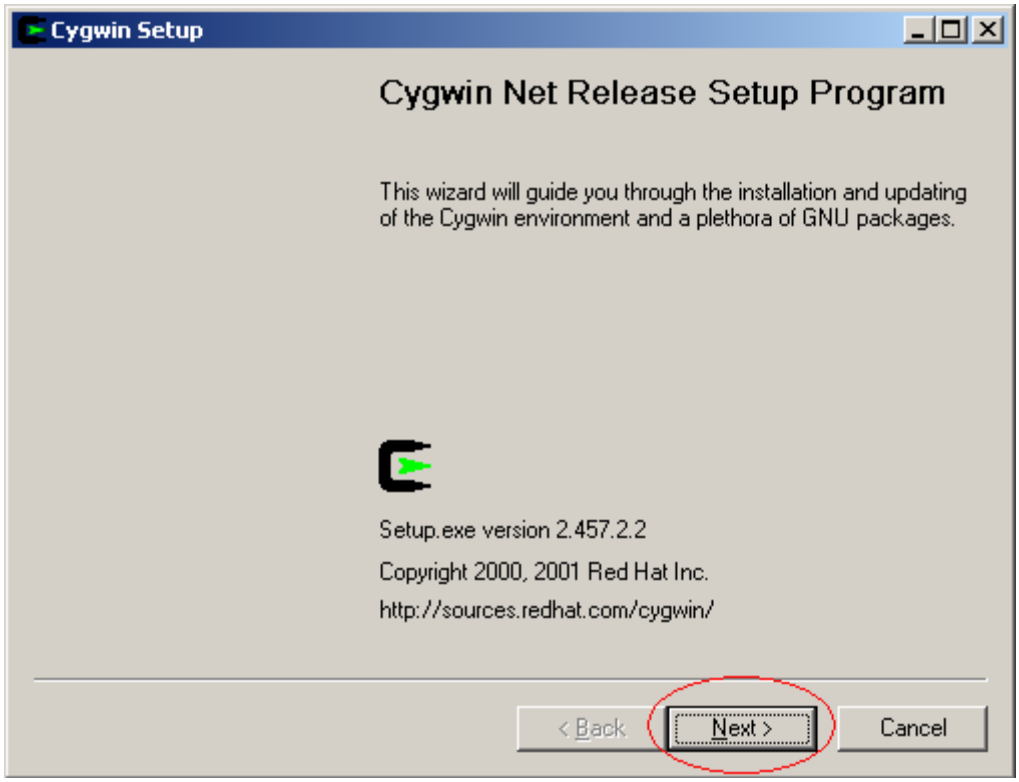
El sitio de Cygwin se ve de la siguiente forma:



Lo primero que haremos es hacer click en el icono de instalación, descargamos el ejecutable setup.exe y lo ejecutamos de forma automática.

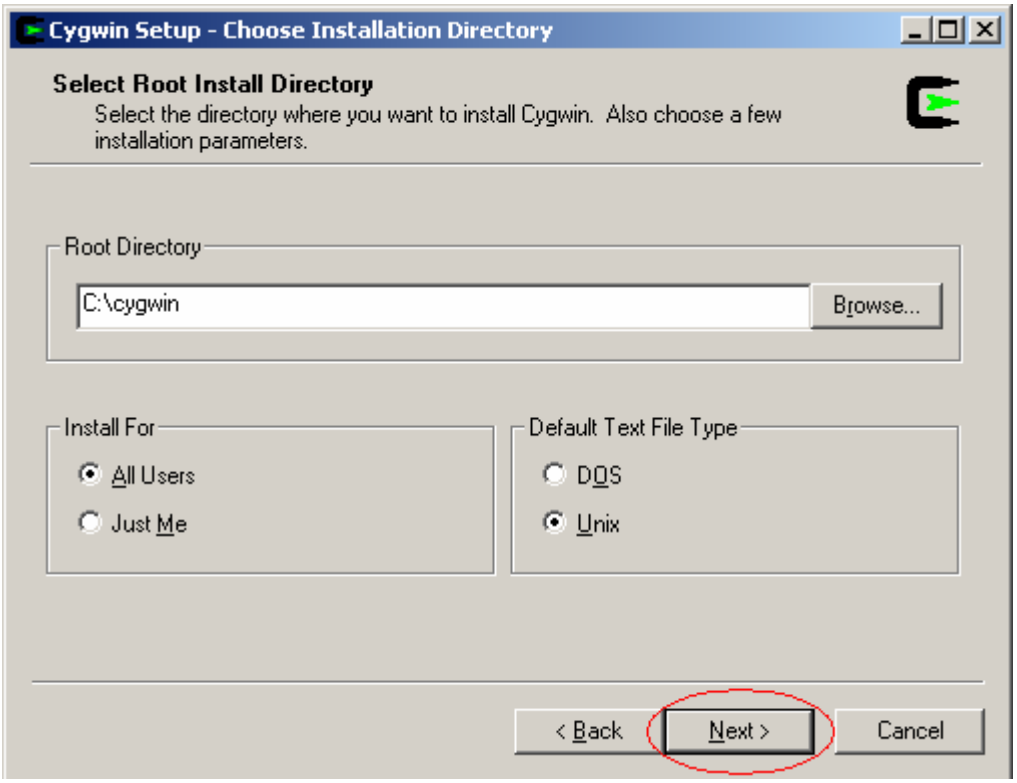


A continuación aparecerá la siguiente pantalla, en la cual hacemos click en continuar (next) para seguir con la instalación.

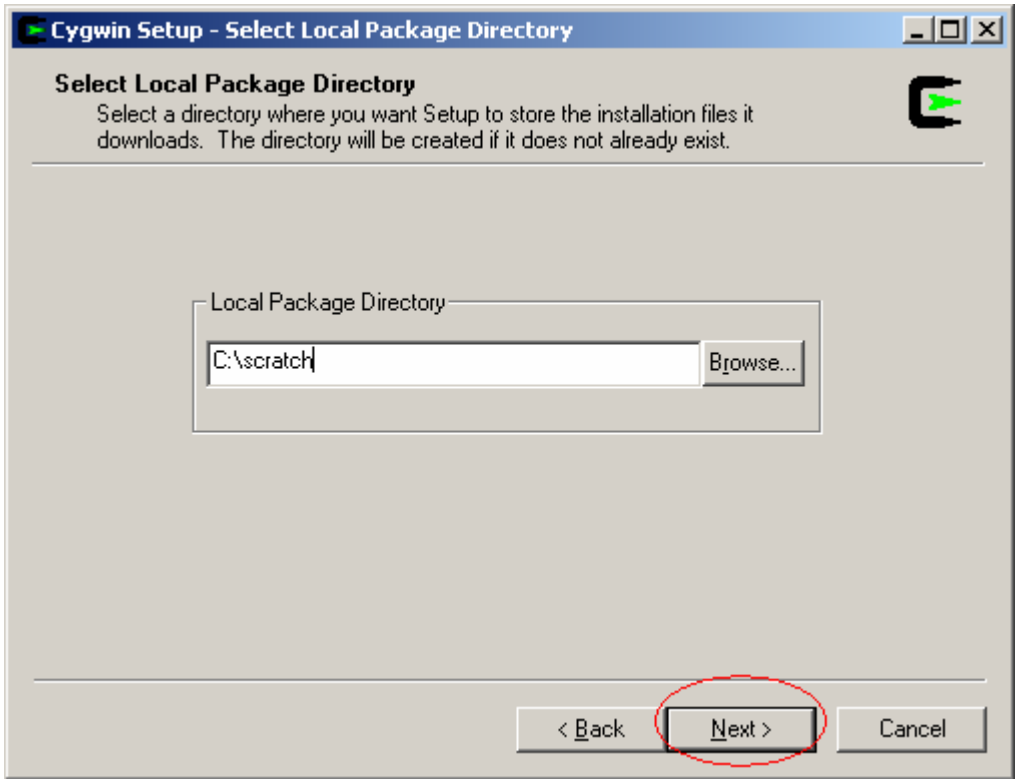


Seleccionamos la opción instalar desde Internet ("Install from Internet") y luego click en continuar.

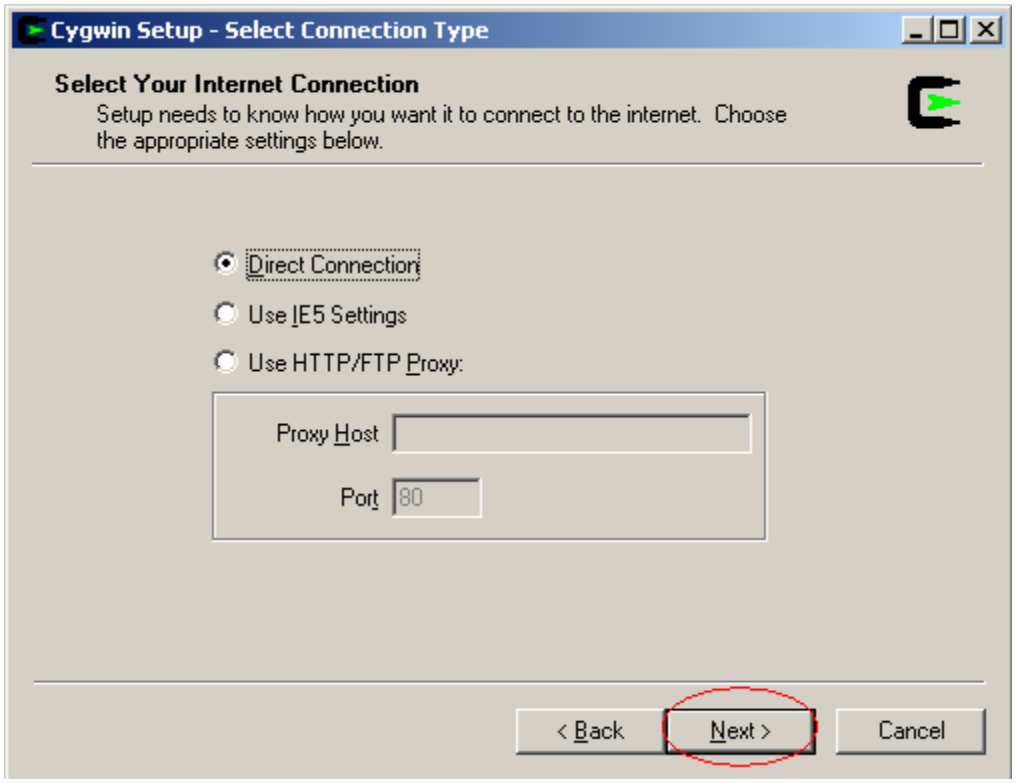
El programa de instalación preguntará el directorio en el cual será instalado Cygwin. Seleccionamos el directorio que deseemos y luego hacemos click en siguiente.



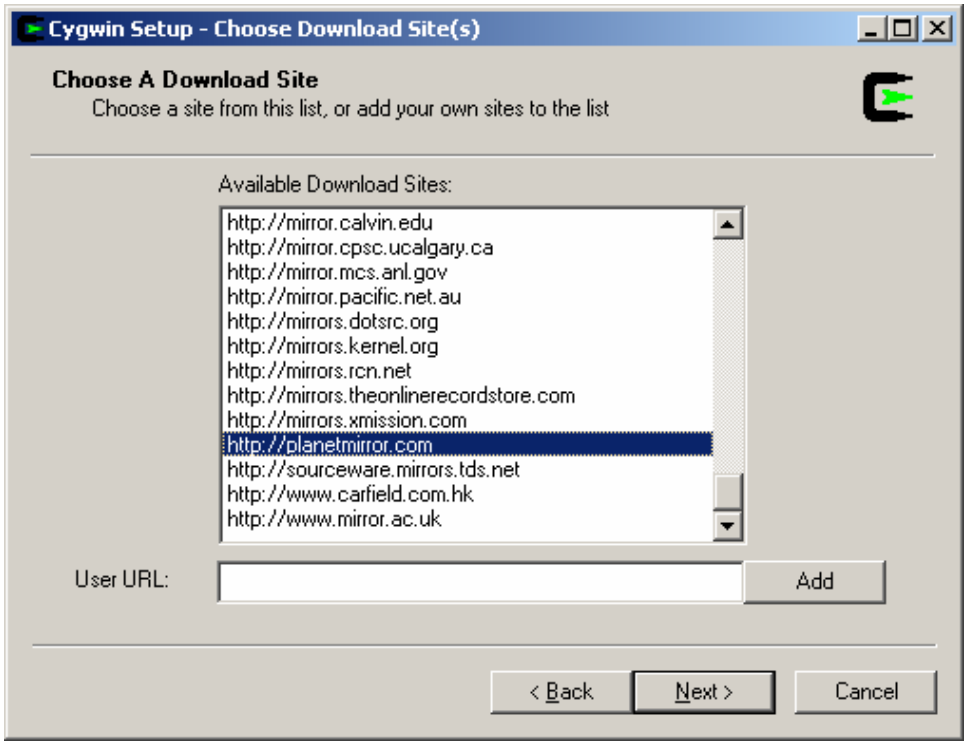
Ahora seleccionamos el directorio donde quedarán los archivos de instalación. En nuestro caso utilizaremos el directorio c:/scratch



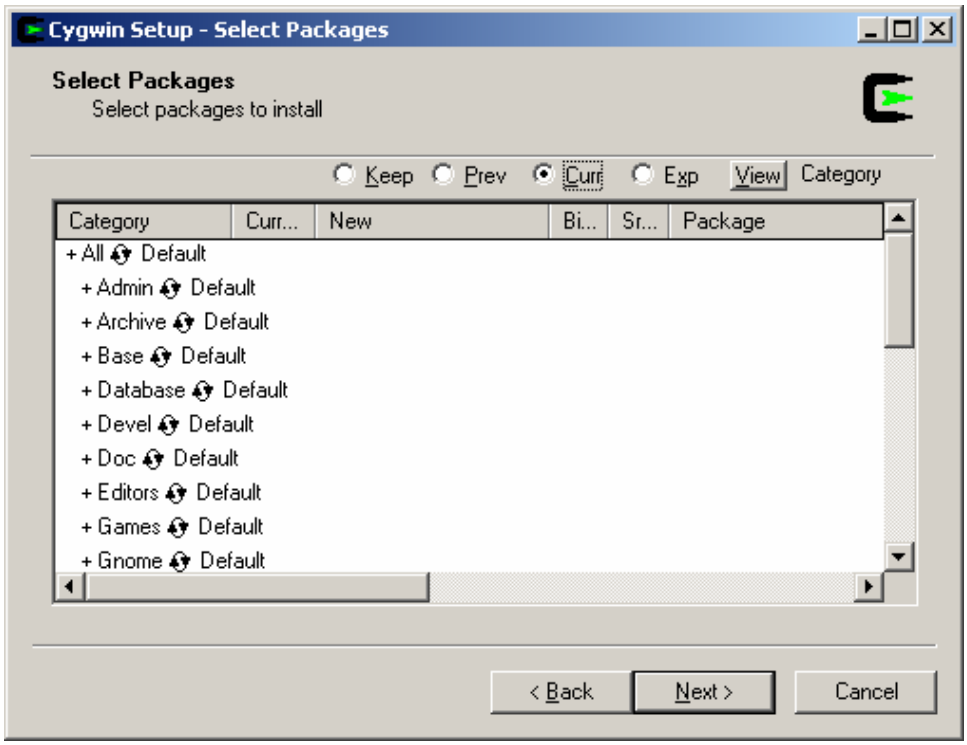
Luego seleccionamos nuestro tipo de conexión. Como no tengo ninguna configuración especial en mi red ocupo “Direct Connection”.



El instalador presentará ahora una lista de mirrors de los cuales se pueden descargar los archivos de Cygwin. Selecciona el mirror más cercano. Como no sé cual es más cercano a mí seleccionaré planetmirror.



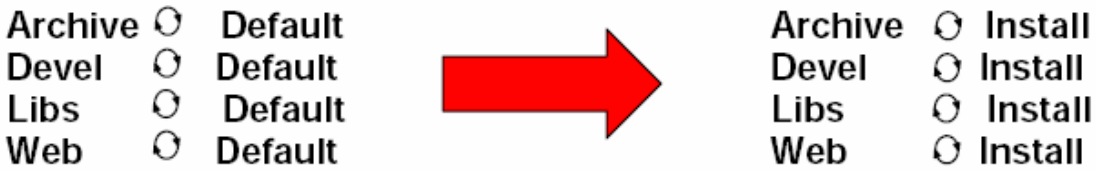
El programa de instalación descargará y mostrará ahora una lista de componentes para instalar. Seleccionaremos los que nos interesan y continuaremos con la instalación de Cygwin.



Si miras los paquetes seleccionados por defecto verás la línea:

Debes hacer click en el círculo con las flechas para que la opción cambie a instalar

Los paquetes que vamos a instalar para que GNU funcione correctamente son los siguientes:



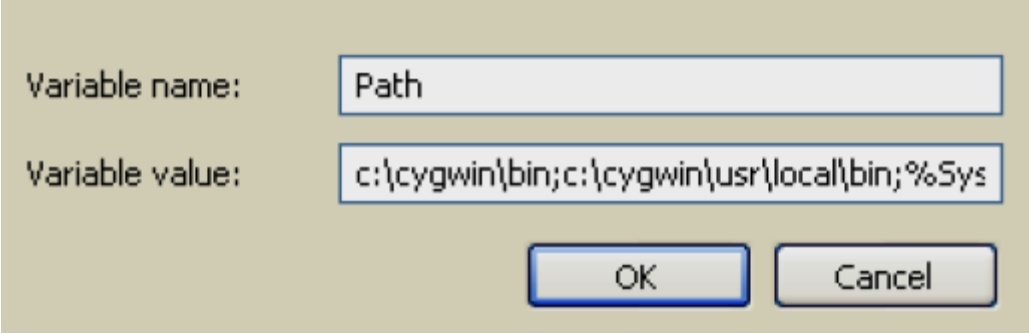
Una vez seleccionados los paquetes proseguiremos con la instalación haciendo click en el botón “Next”. Cygwin comenzará a descargarse.

En este proceso se descargan del orden de 700Mb en el directorio escogido lo cual toma del orden de 30 minutos. Cuando la instalación se complete el programa preguntará si deseas dejar un icono en el

menú de inicio. Selecciona no a ambos. Estos iconos permiten abrir un emulador de BASH para hacer algunas tareas que podríamos hacer bajo Linux, en este caso no es necesario tener esta característica.

El directorio c:\cygwin\bin debe ser agregado a la path de Windows XP. Esto permitirá que Eclipse pueda encontrar la utilidad make.

Para agregar el directorio a la Path es necesario ir al menú “Inicio -> Panel de Control” y luego hacer click en el icono “Sistema”. Haz click en “Avanzado” y luego selecciona “Variables de Entorno”. Selecciona Path y luego presiona el botón editar. En la línea “valor de la variable” agrega el directorio. No olvides poner un punto y coma para separar el directorio de los otros ya existentes



Hemos terminado con la instalación de Cygwin.

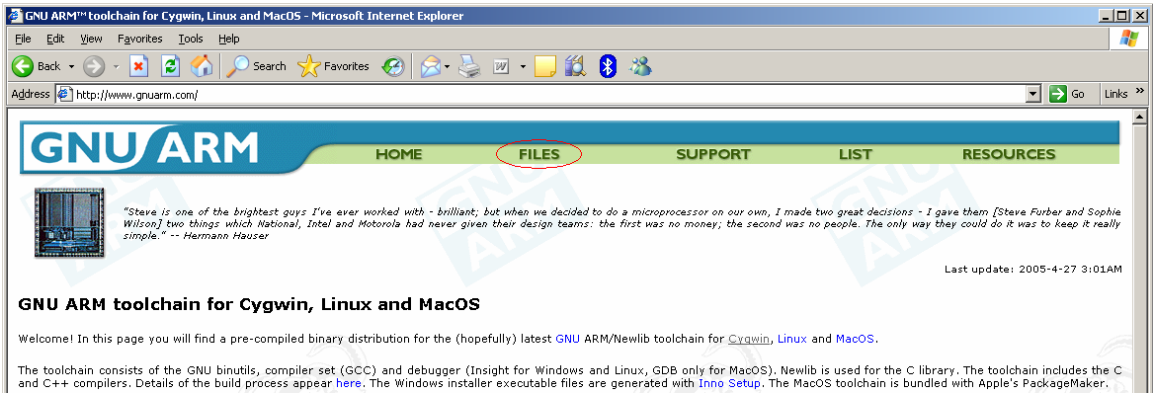
Descargando el GNUARM

En este punto tenemos todas las herramientas GNU necesarias para compilar y linkear el software para computadores Intel/Windows. Es posible utilizar todas ellas para construir un compilador GNU a medida para un procesador de la familia ARM. En el libro de Lewin A.R.W. Edwards “Embedded System Design on a Shoestring” se describe como hacer esto.

Afortunadamente, Pablo Bleyer Kocik y la gente de gnuarm.com desarrollaron un compilador gratuito para procesadores ARM. Sólo hay que descargarlo e instalarlo.

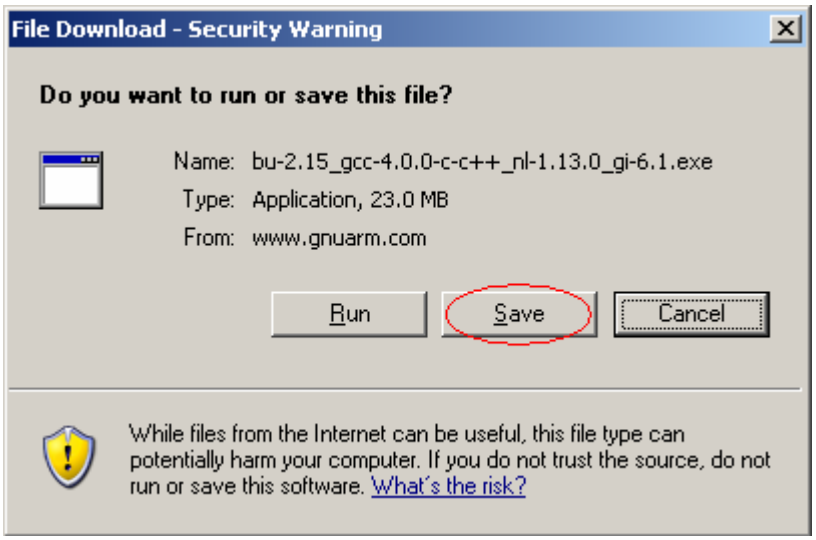
http://www.gnuarm.com

El sitio de GNUARM se ve de la siguiente forma, hacemos click en “Files”

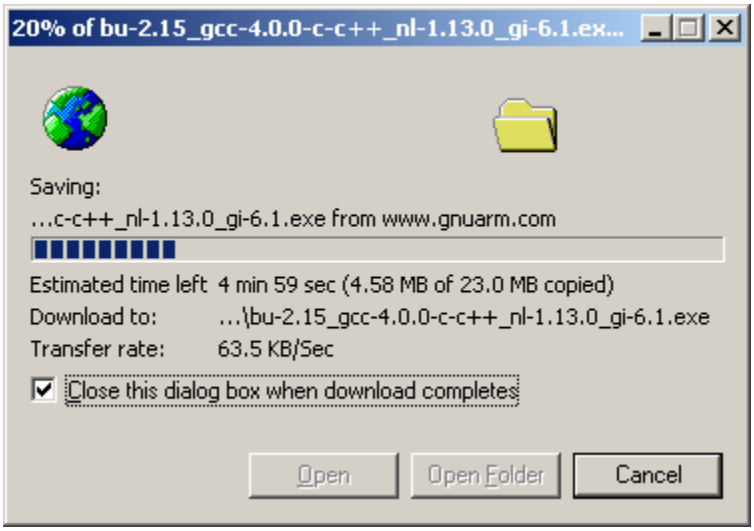


La descarga correcta es GCC-4.0 toolchain

Como en las otras descargas hacemos click y elegimos guardar.

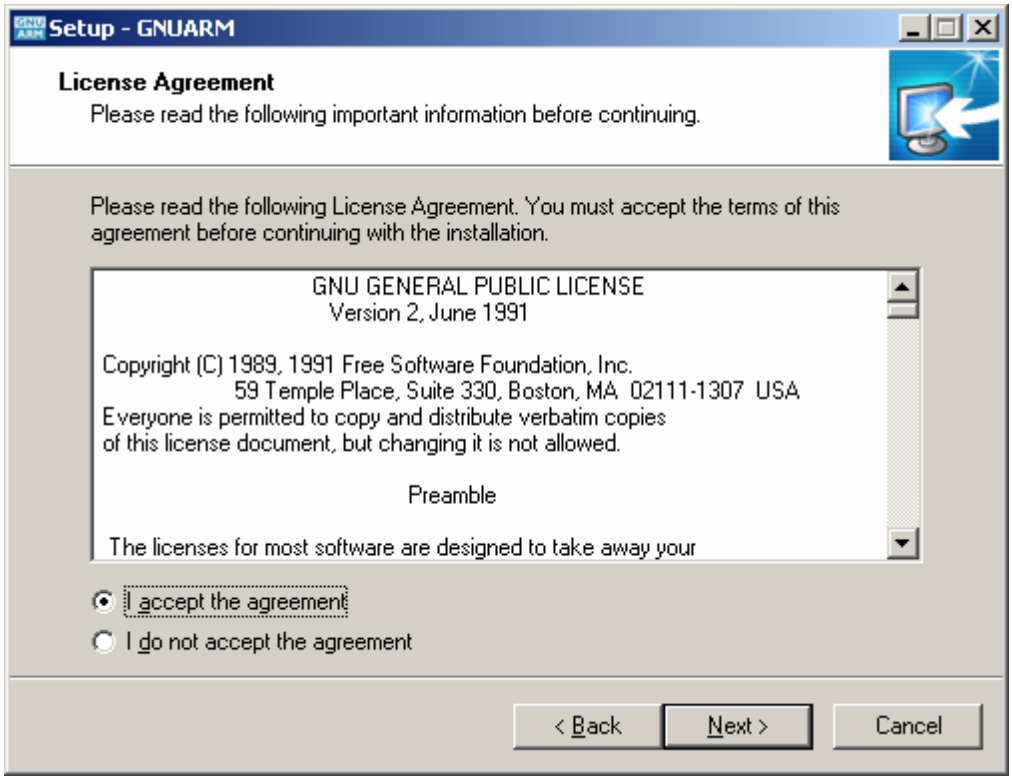


Seleccionamos una vez más el directorio c:\scratch y comenzamos la descarga.

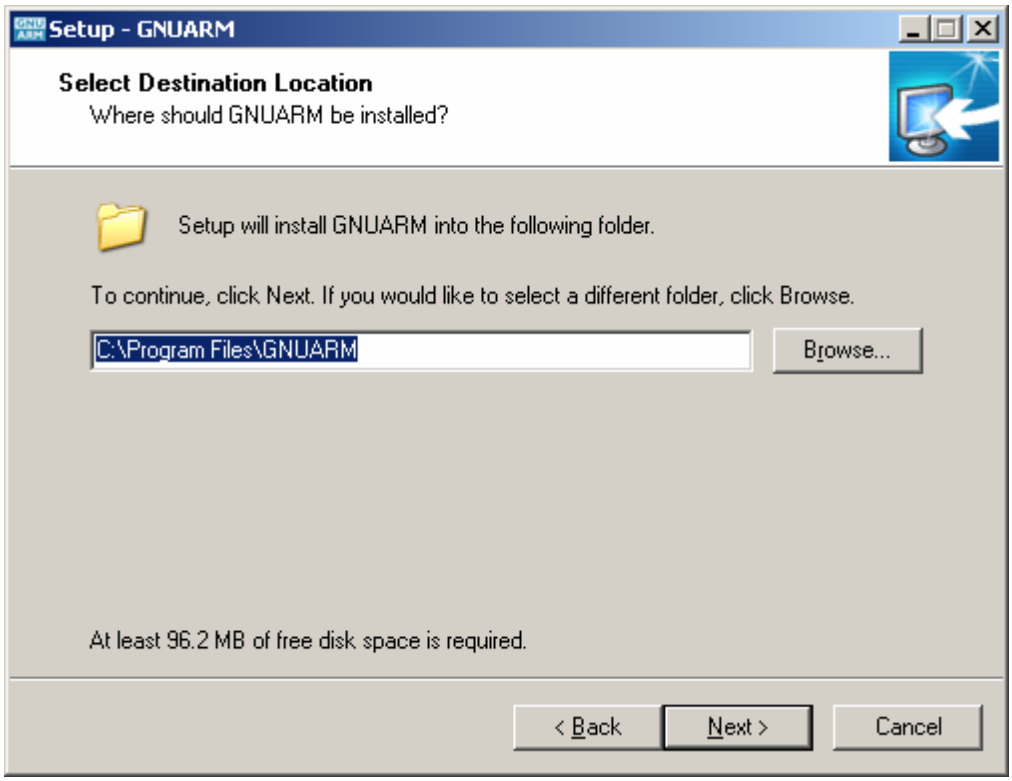


En el directorio de descarga aparecerá el archivo bu-2.15_gcc-4.0.0-c-c++_nl-1.13.0_gi-6.1.exe hacemos doble click sobre él y comenzamos con la instalación.

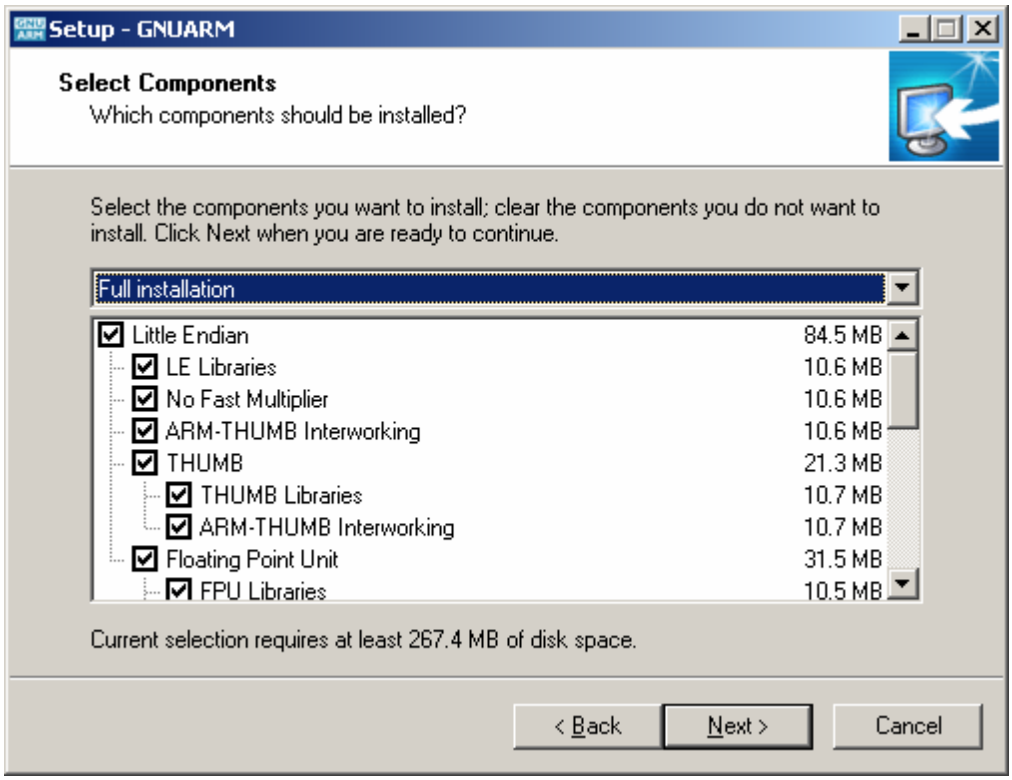
Aceptamos la licencia

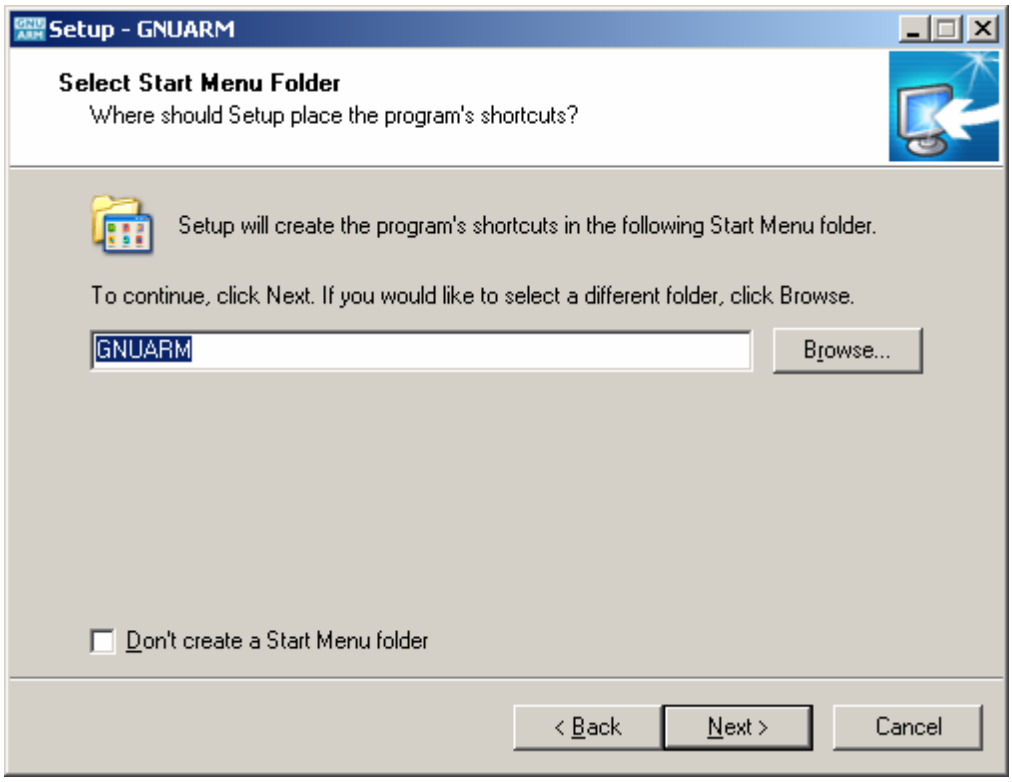


Seleccionamos el directorio de instalación



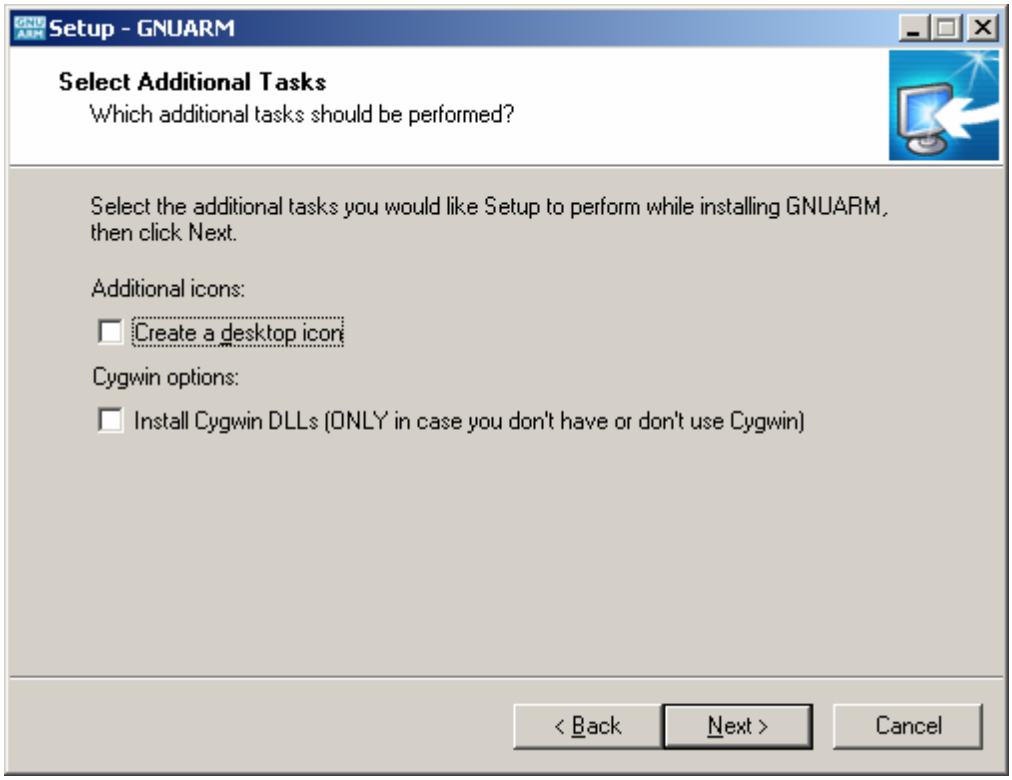
Seleccionamos los paquetes a instalar, tomaremos los paquetes por defecto.



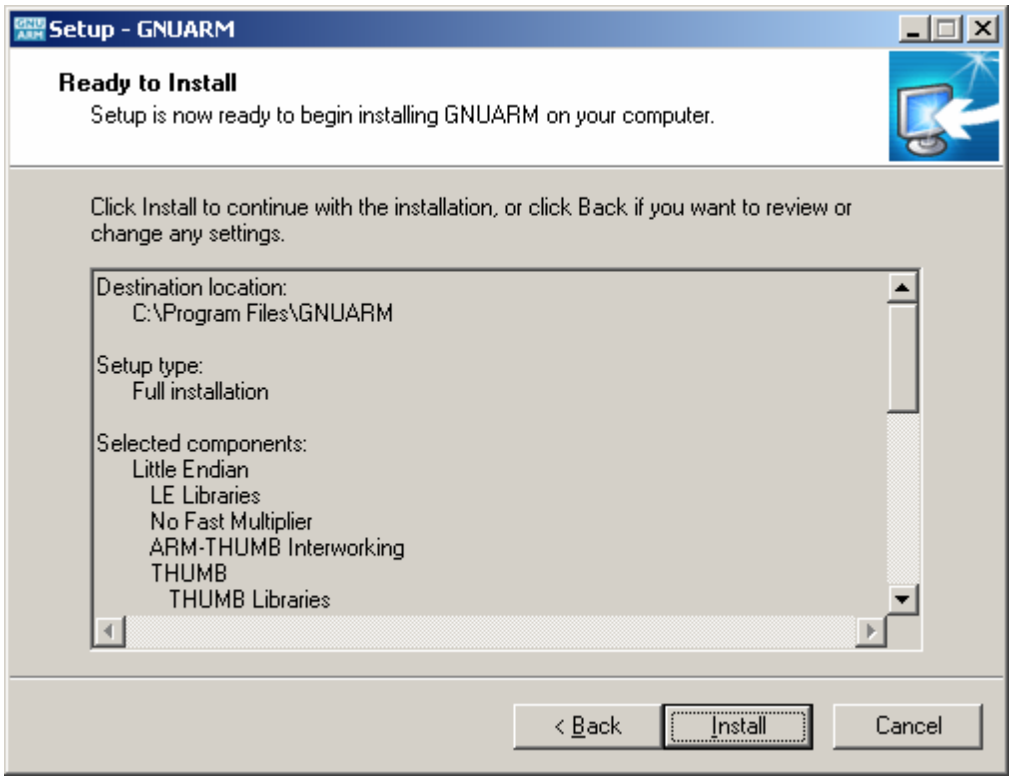


Es importante notar que **NO hay que instalar "Install Cygwin DLLs"**, por que ya los instalamos con anterioridad al instalar el

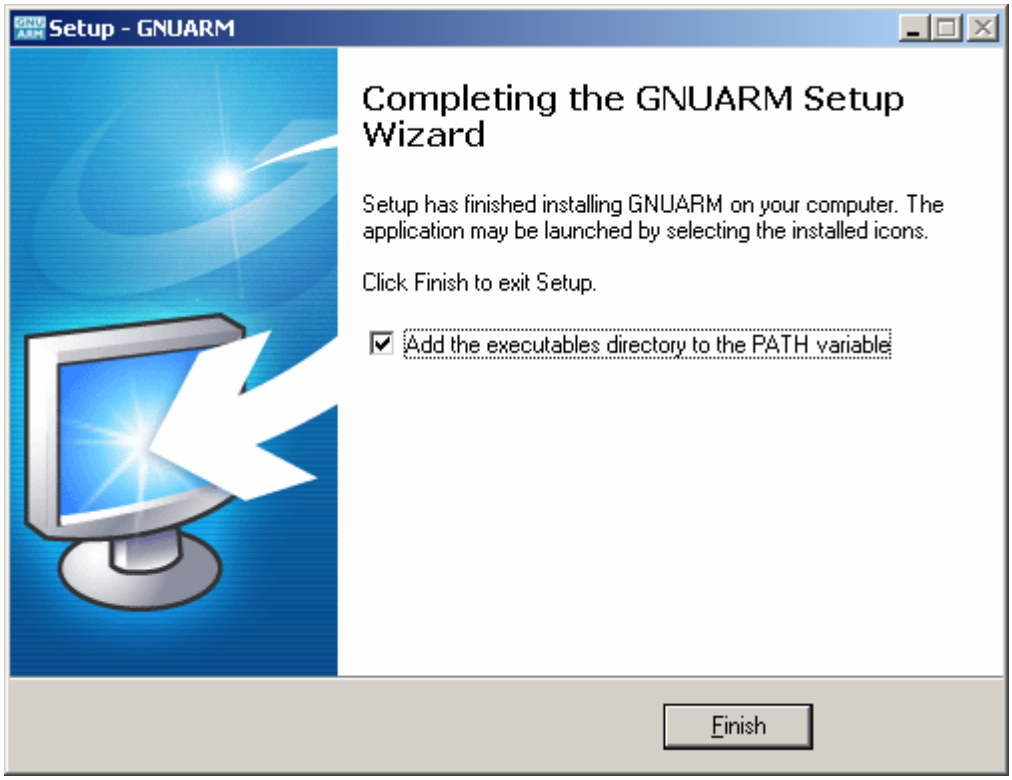
Cygwin. Por otro lado como todas las operaciones que haremos con GNUARM serán desde eclipse no necesitamos “desktop icon”



Ahora comenzamos la instalación.



Una vez terminada la instalación **es necesario agregar las variables en la PATH de Window**s. Esto es crucial.



Con esto completamos la instalación del compilador. Como Eclipse lo llama utilizando el make file no nos preocuparemos más de GNUARM.

Es bueno saber que existe un grupo en Yahoo sobre GNUARM en el cual se publican preguntas y sus respuestas. El sitio de GNUARM también contiene links a toda la documentación sobre ARM que pudieses necesitar.

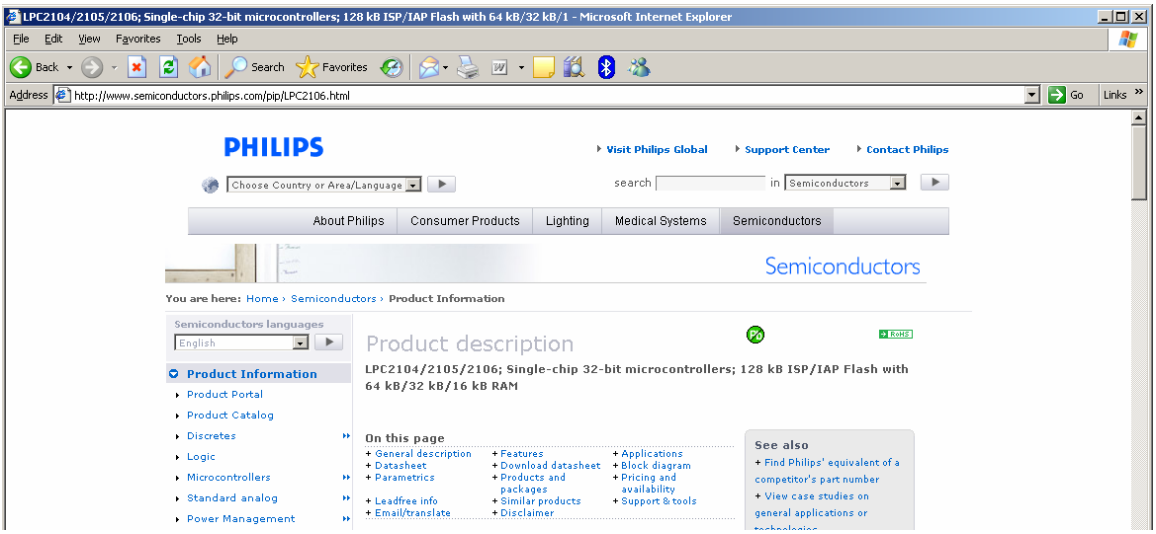
Instalando el Philips LPC2000 Flash utility en Eclipse

El "Philips LPC2000 Flash Utility" permite descargar archivos hex utilizando el puerto COM1 a la placa memoria flash Olimex LPC-P2106 (o RAM).

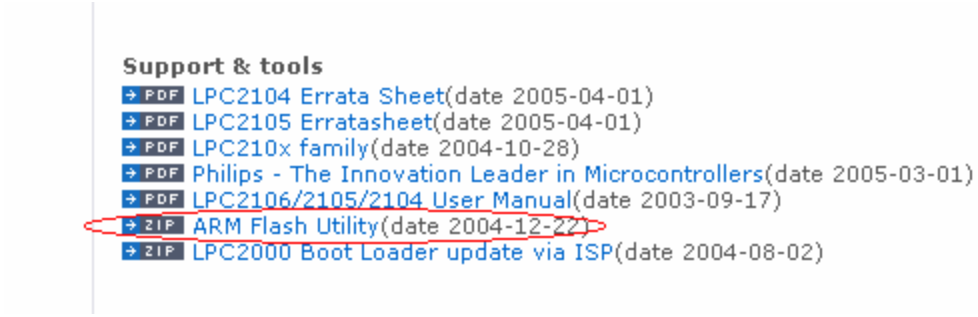
La última versión del programa la podemos descargar desde la página de Philips.

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

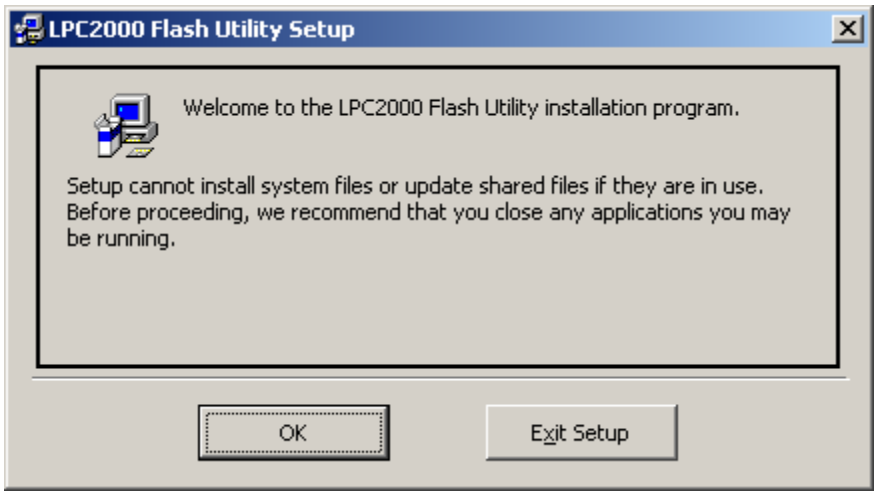
La siguiente ventana aparecerá:



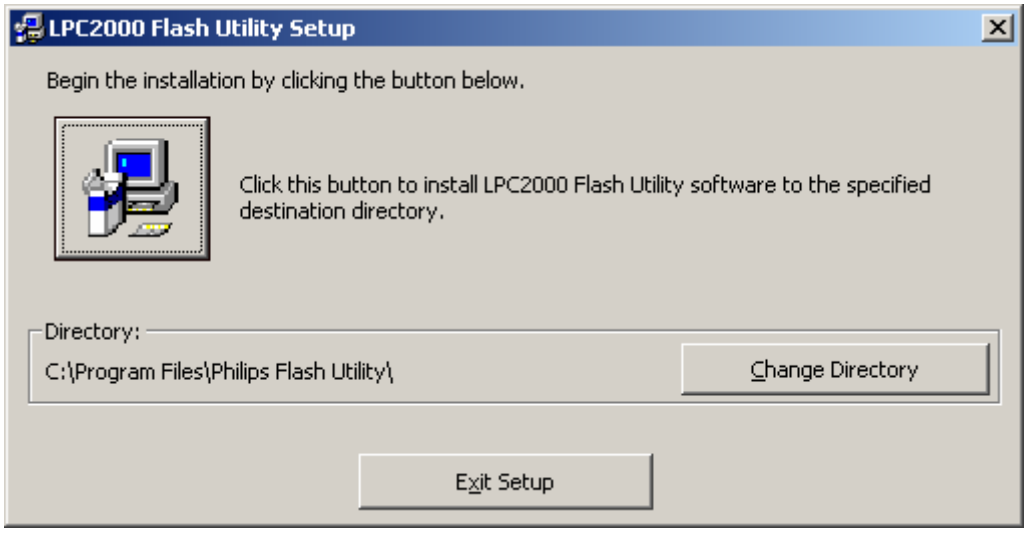
Bajamos y seleccionamos ARM FLASH Utility



Como lo hicimos anteriormente descargamos el archivo en el directorio c:\scratch .Una vez descargado lo descomprimimos y ejecutamos el archivo setup.exe

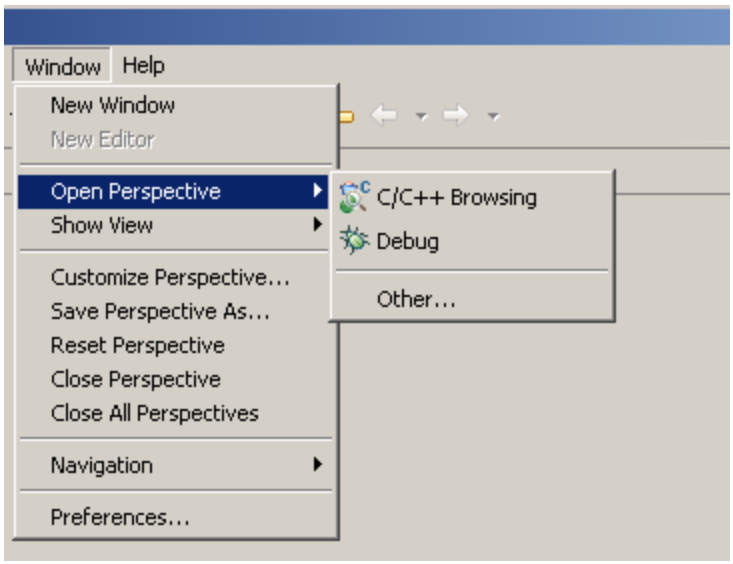


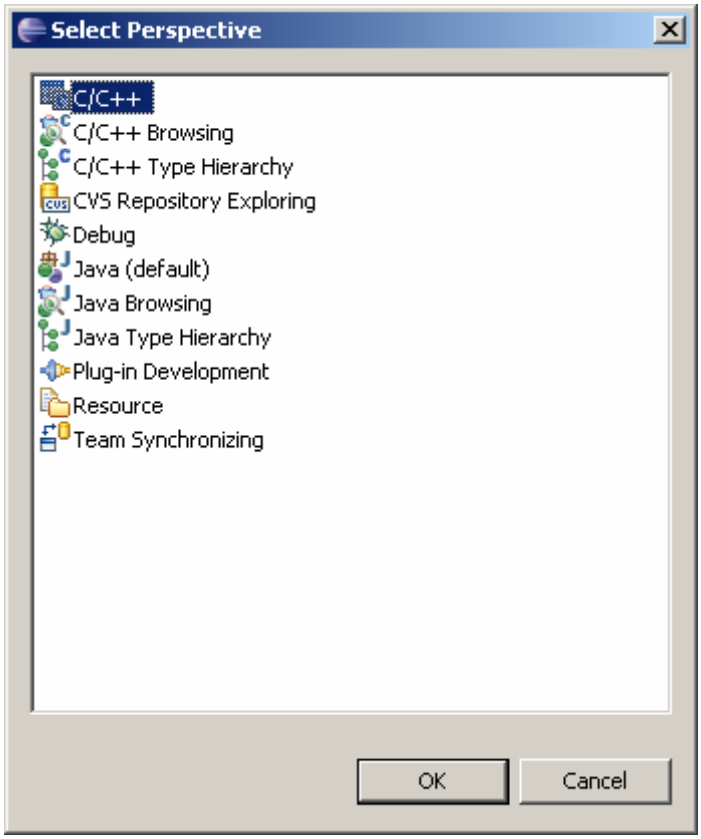
Seleccionamos el directorio por defecto y continuamos con la instalación.



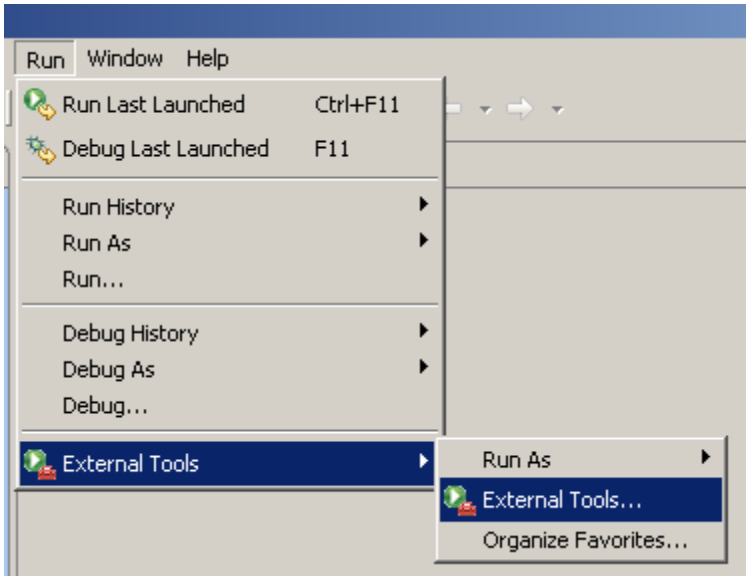
Ahora que el LPC2000 Flash Utility está instalado en nuestro computador, necesitamos configurar eclipse para que pueda llamarlo desde el menú “External tools”.

Ejecutamos eclipse. El layout de la pantalla que aparece se llama “Perspective”. La perspectiva por defecto se llama “Java”. Cambiamos la perspectiva a C/C++ haciendo click en el menú “Window –> Open perspective ->Other…” y luego seleccionando C/C++

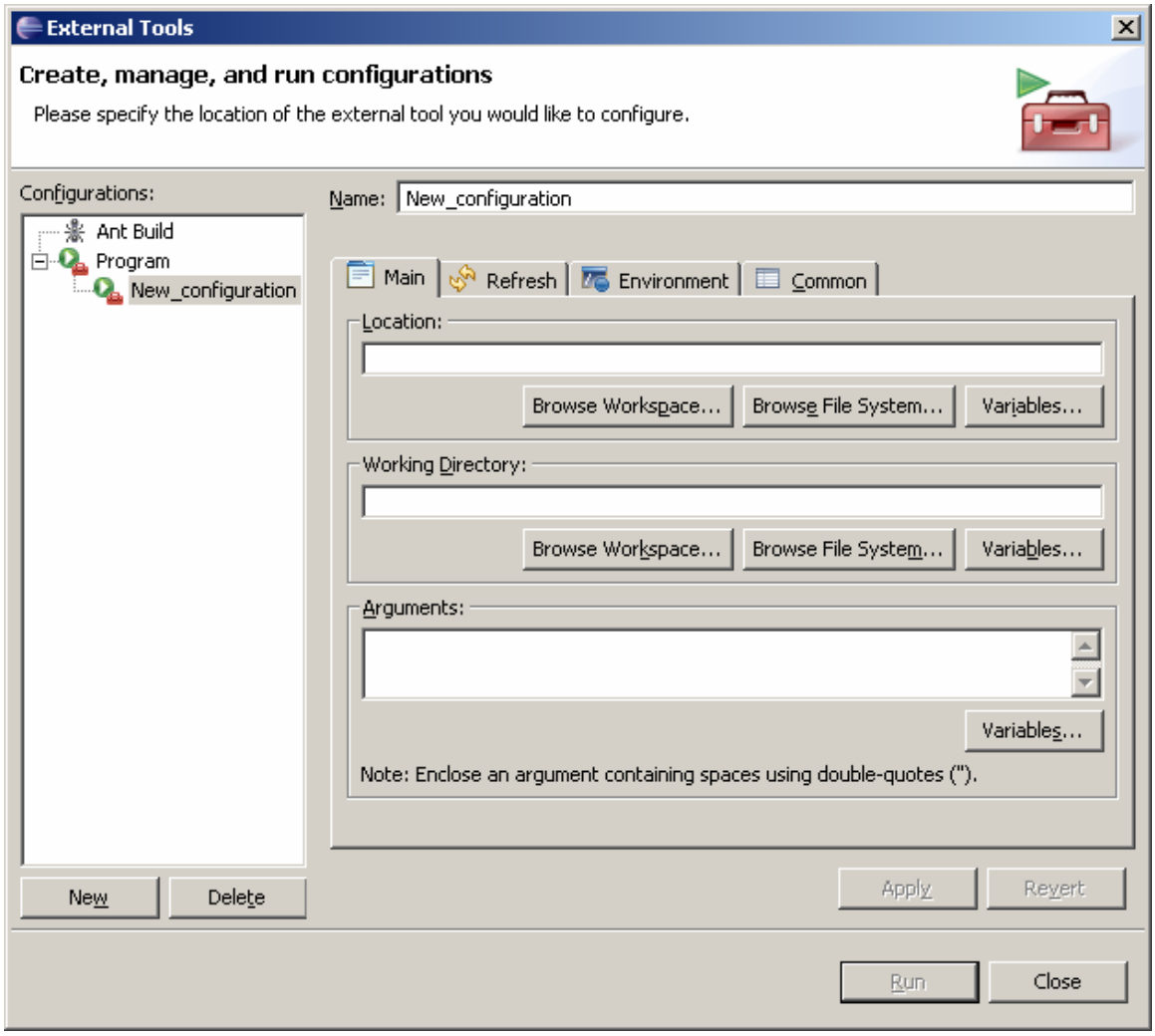




Ahora necesitamos llamar la utilidad de Philips como una herramienta externa, para lo cual vamos al menú "RUN -> External Tools -> External Tools".



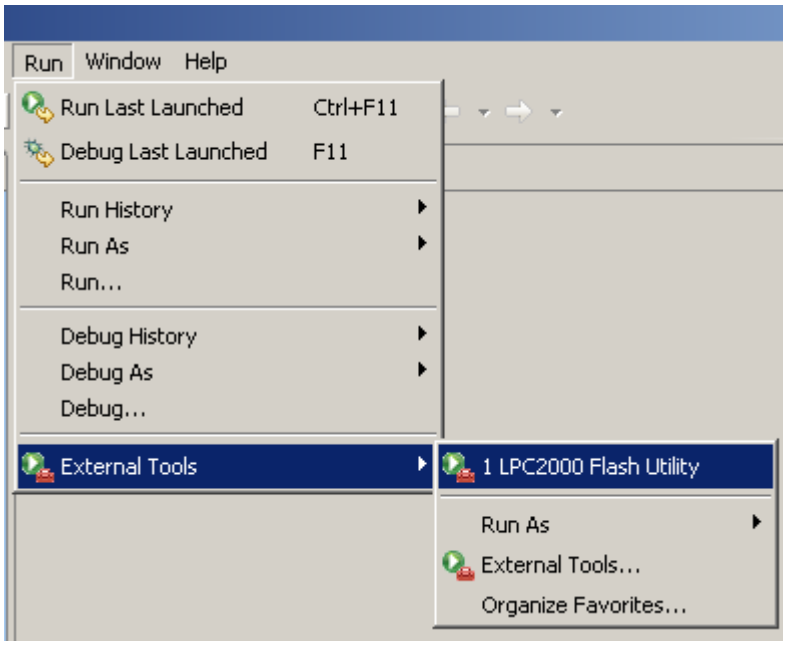
Hacemos click en "Program" y agregamos uno nuevo haciendo click en el botón "New"

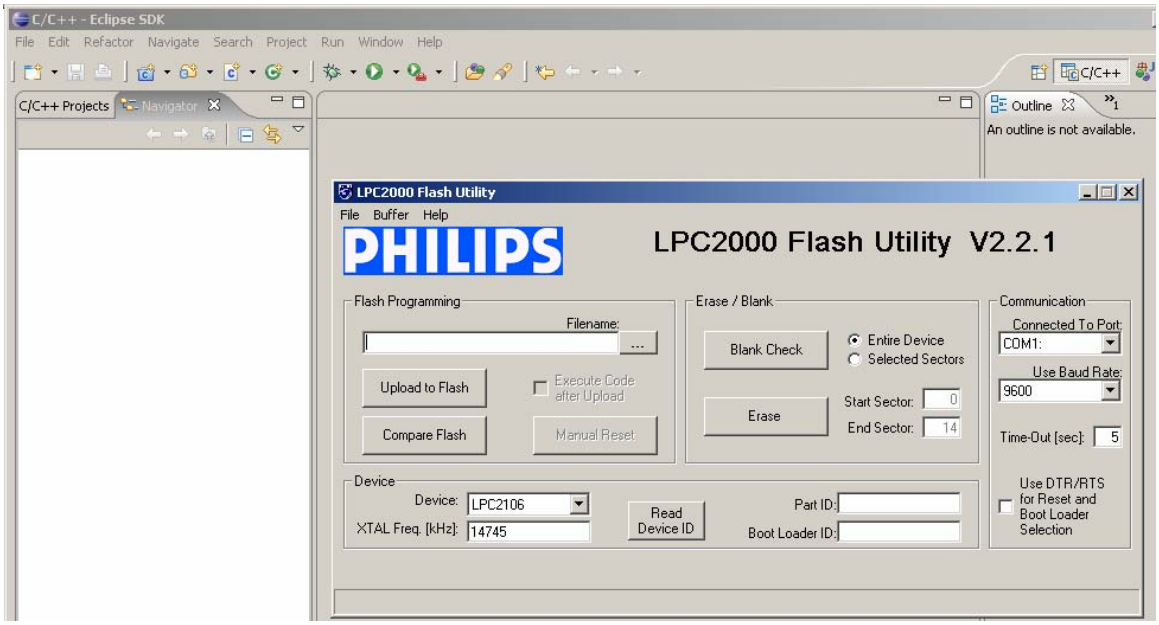


En “Name” reemplazaremos New-configuration con LPC2000 Flash Utility. En “Location” usamos “Browse File System” y vamos a buscar la aplicación de Philips. El archivo se encuentra en C:\Program Files\Philips Flash Utility\LPC210x_ISP.exe si es que utilizamos el directorio de instalación por defecto. Luego hacemos click en “Apply”.

Ahora agregamos LPC2000 a la lista de favoritos en el menú “Run->External Tools-> Organize Favorites”

Verificamos ahora de que LPC2000 funcione correctamente haciendo click en el menú “RUN->External Tools->LPC2000 Flash Utility”





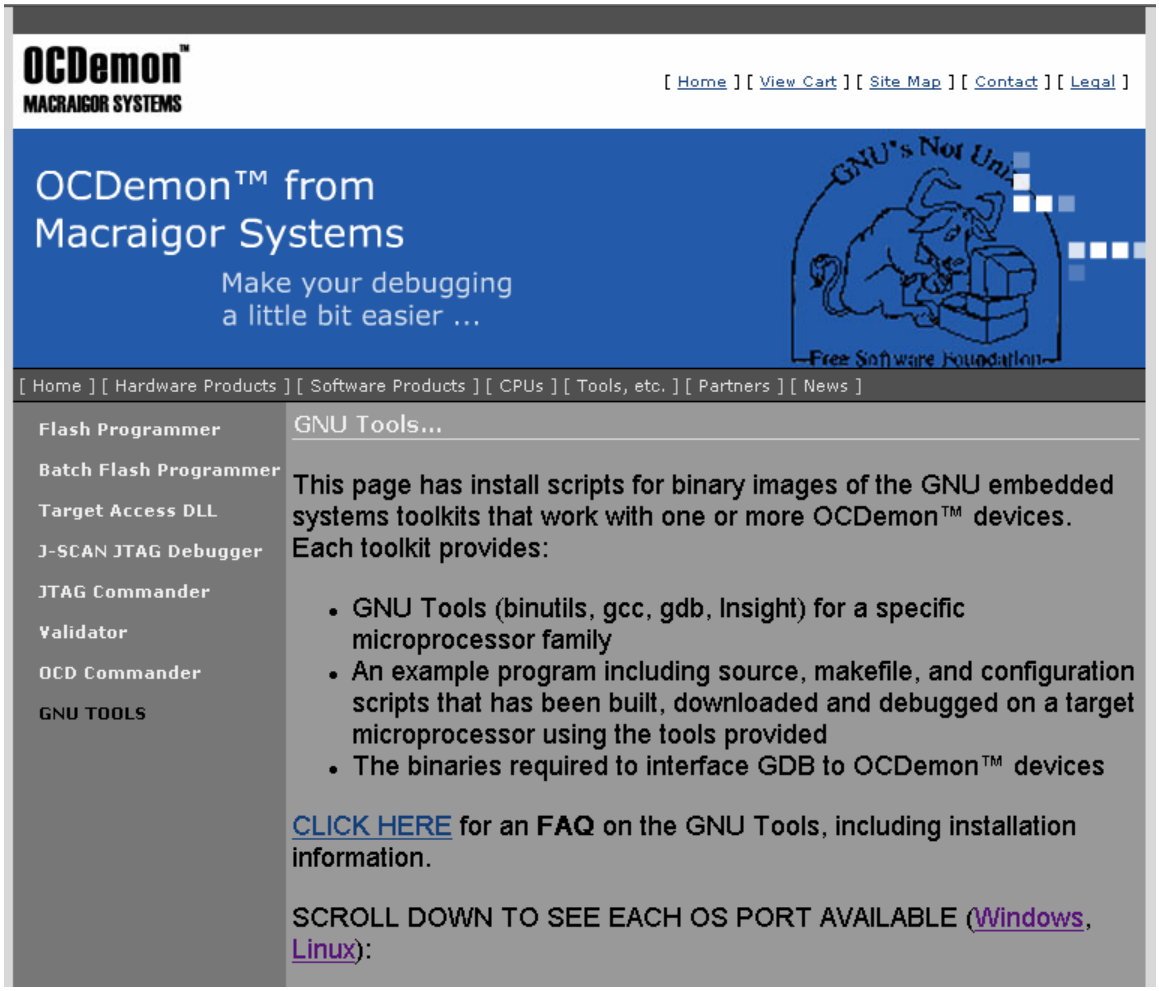
Ahora cancelamos y salimos de Eclipse para seguir con la instalación de OCDRemote.

Instalando el Macraigor OCDremote.

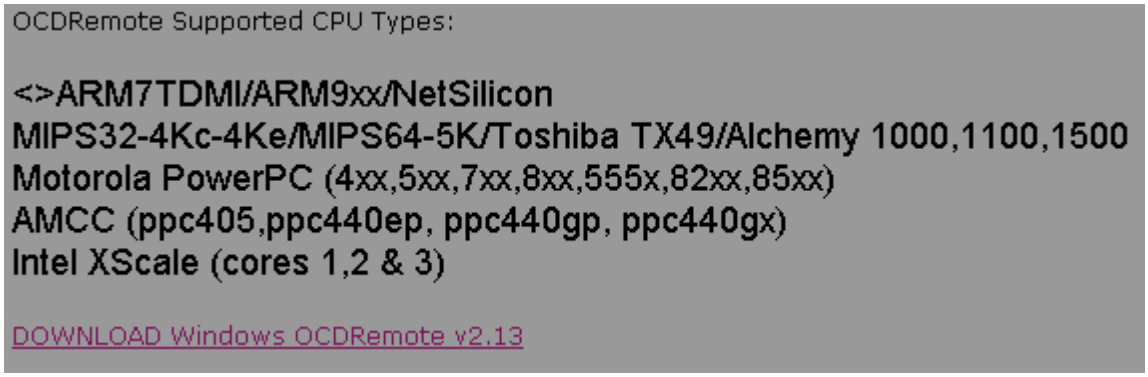
OCDRemote es una utilidad que escucha el puerto TCP/IP y traduce la comunicación GDB a comandos reconocidos por el JATG Wiggler. Esto permite a Eclipse/GDB comunicarse con la tarjeta Olimex LPC-P2106 via Ethernet. Macraigor mantiene esta utilidad disponible en Internet como “freeware”. El OCDRemote puede ser descargado desde:

http://www.macraigor.com/full_gnu.htm

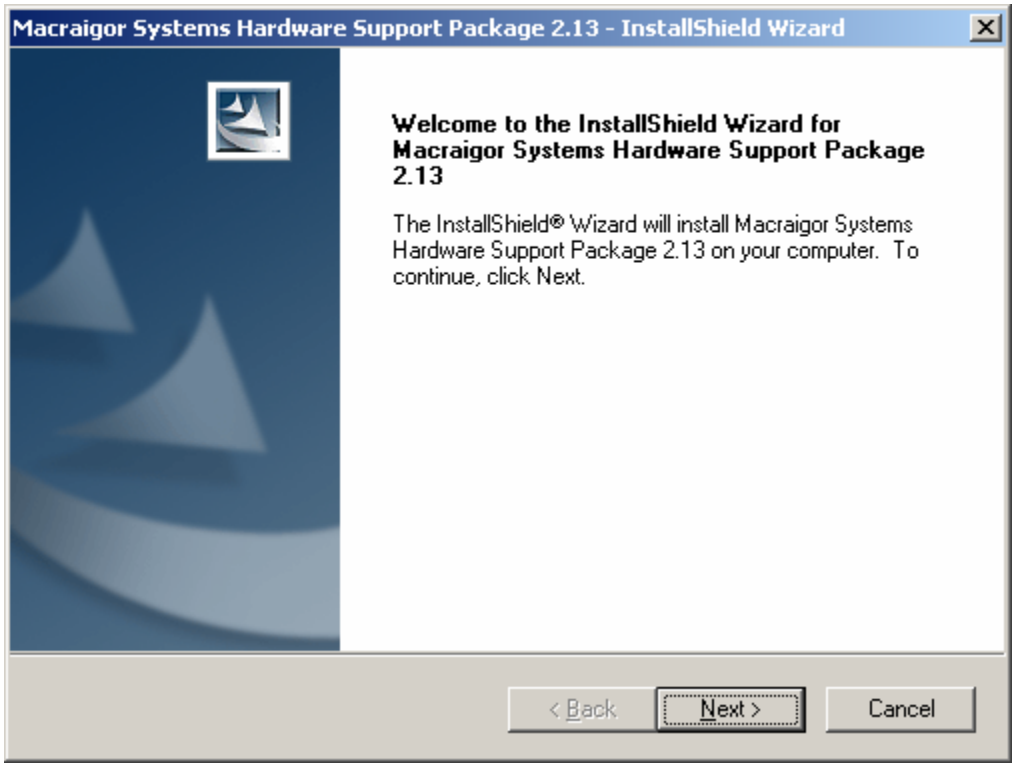
La página se ve de la siguiente forma:



Si bajamos un poco podremos ver el link a “Download Windows OCDRemote v2.13”



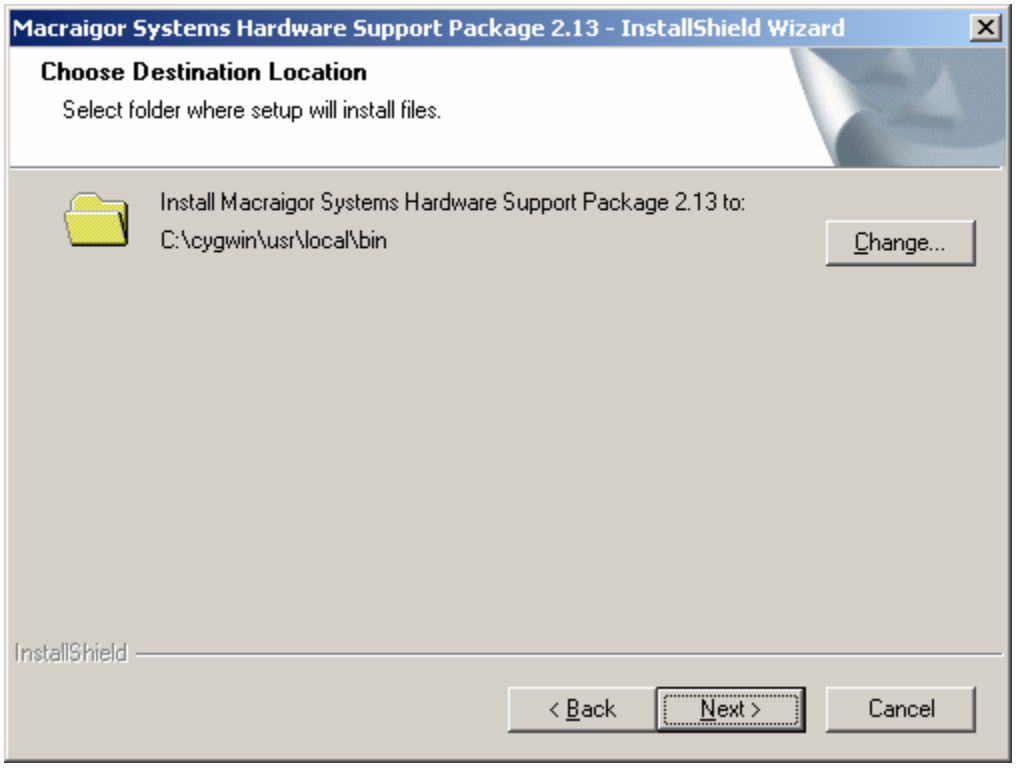
Procedemos a la descarga e instalación



La siguiente pantalla permite escoger donde será instalado el OCDRemote. OCDRemote se instala generalmente en **c:/Cygwin/usr/local/bin**.

Hay que asegurarse de que el directorio que indiquemos se encuentre en la PATH de Windows.

Hacemos click en siguiente “Next” para continuar con la instalación.



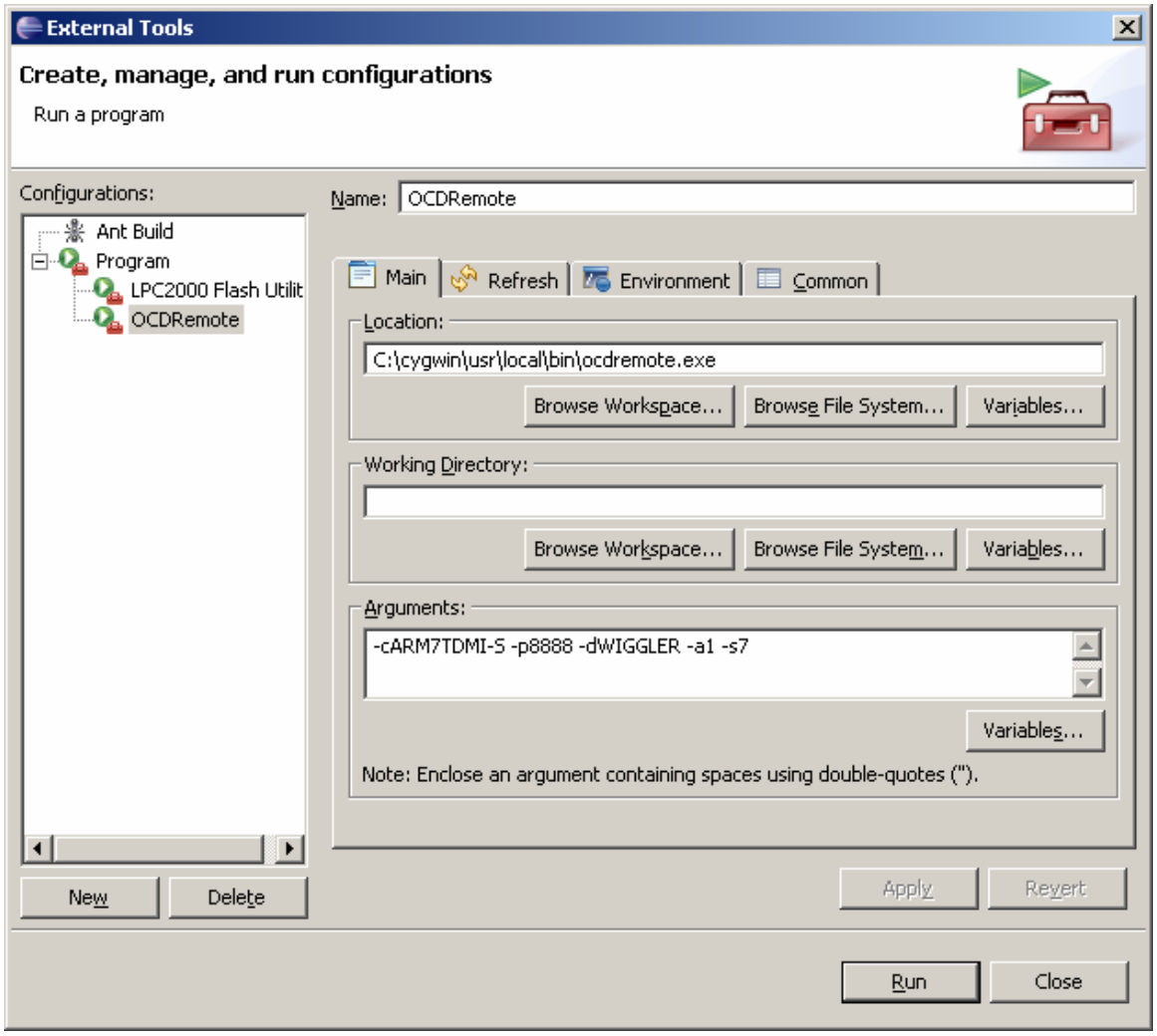
Una vez terminada la instalación el programa nos pedirá reiniciar el equipo para actualizar el registro de Windows. Seleccionamos “YES” y reiniciamos el PC.

Al igual que la aplicación “Philips Flash Utility” queremos que Macraigor OCDremote se ejecute como una aplicación externa desde Eclipse. Luego es necesario realizar la misma configuración que se hizo anteriormente en el menú “RUN->External Tools-External Tools…”

Hacemos clic en “New” y escribimos en el campo “name” OCDRemote Utilizamos también el botón “browse file system” para encontrar la aplicación que debiera estar en c:\Cygwin\usr\local\bin:

Como argumento necesitamos poner lo siguiente:

-cARM7DMI-S	especifica el tipo de CPU al cual se accederá
-p8888	especifica el pseudo puerto TCI/IP que se utlizará
-dWIGGLER	especifica el JTAG que se utlizará
-a1	especifica LPT1 para el Wiggler
-s7	especifica next-to-slowest speed



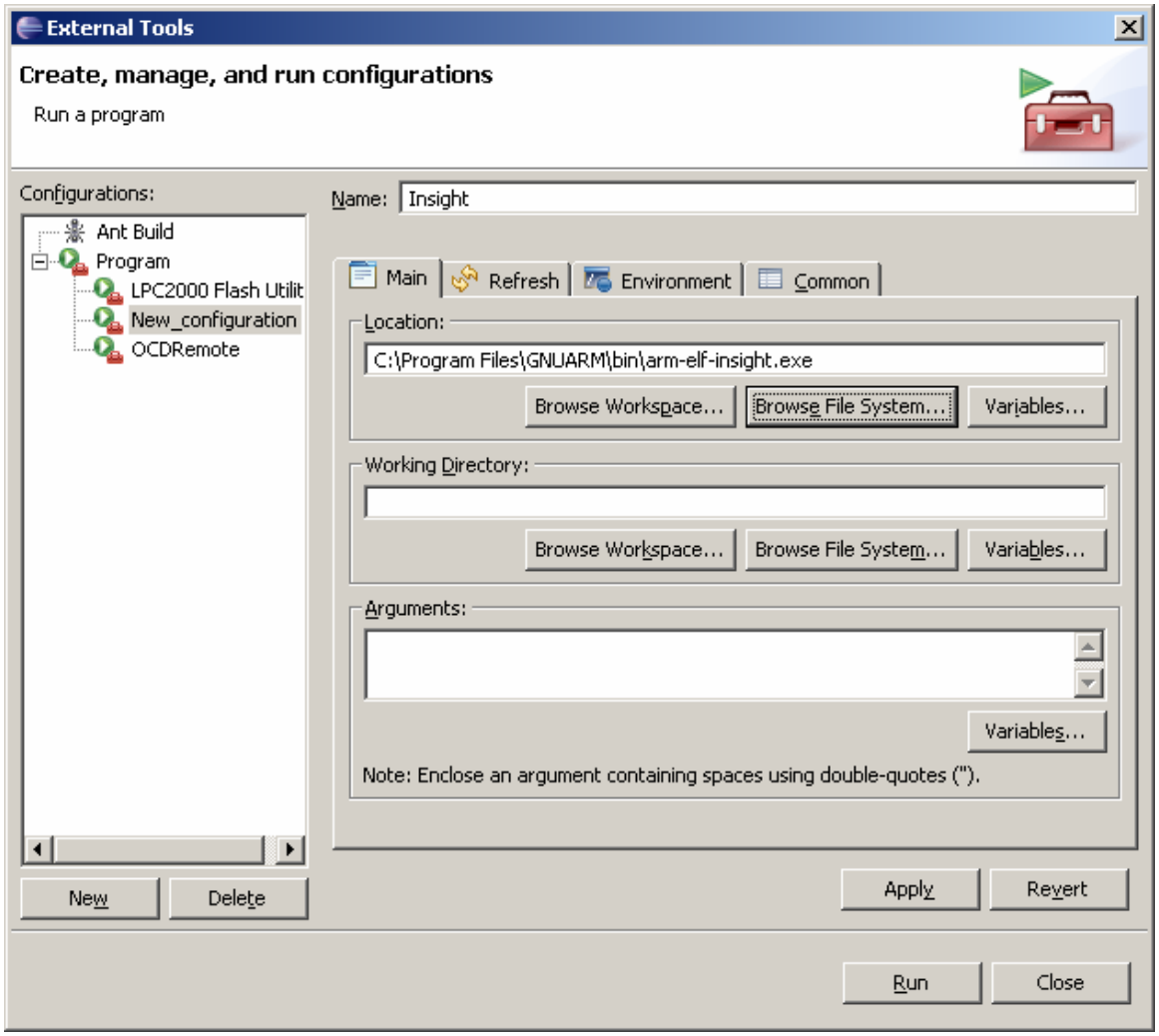
Presionamos el botón “Apply” para no perder los cambios que hicimos. Agregamos también esta herramienta a los favoritos en el menú “Run->External Tools-> Organize Favorites”

Instalando el INSIGHT Graphical Debugger

Eclipse CDT tiene su propio debugger, el cual emplea el protocolo serial GDB. La verdad es que no funciona muy bien. Por esto es que decidimos utilizar un debugger que viene con GNUARM el cual puede trabajar con la interfaz JATG Wiggler.

Instalemos el Insight debugger como una herramienta externa en Eclipse.

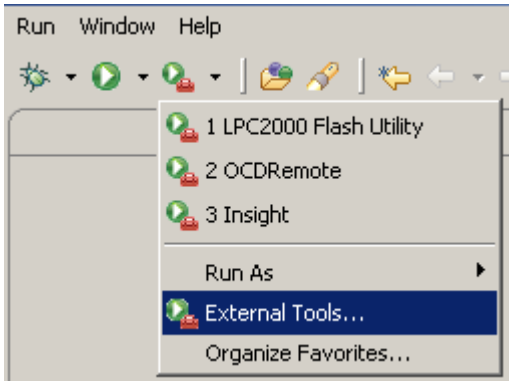
Click en "Run-External Tools – External Tools…" y lo configuramos de la siguiente forma:



Por ahora dejaremos los campos "Working Directory" y "Arguments" vacíos. Estos campos deben contener el nombre del workspace y el

nombre del archivo .out que vamos a utilizar. Procedemos a agregar Insight a nuestra lista de favoritos.

Como una revisión final verificamos que al presionar el botón “Run External Tools” aparezcan todas las aplicaciones que instalamos.

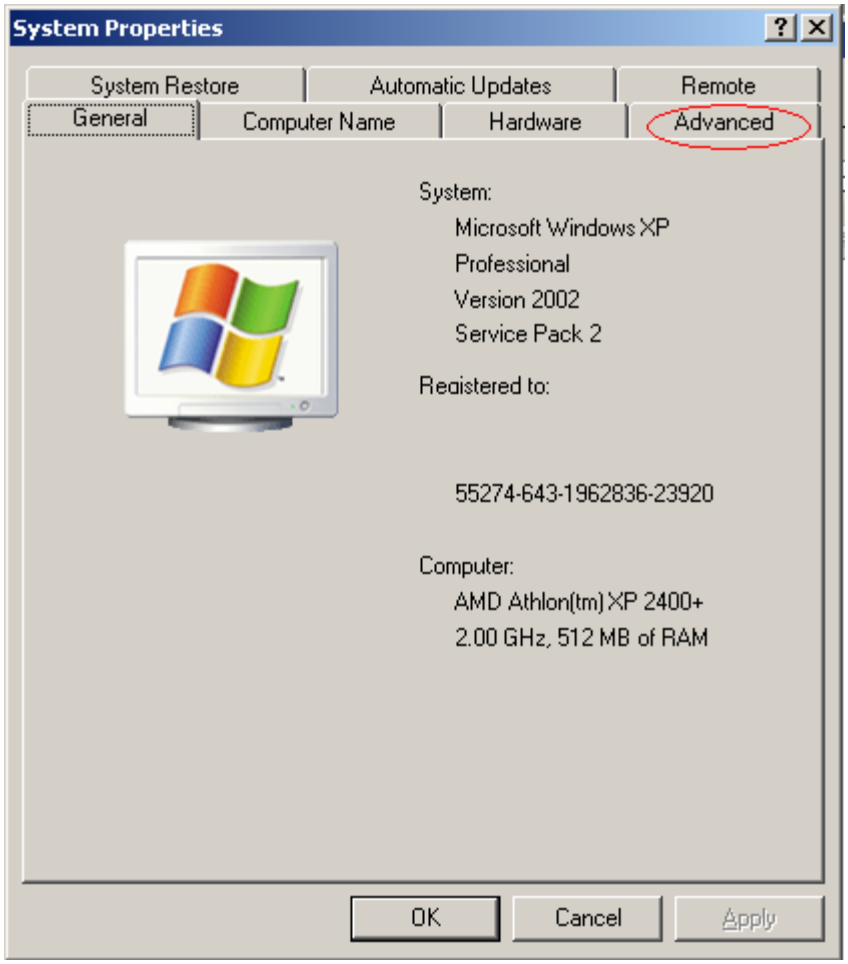


Verificando la PATH de Windows

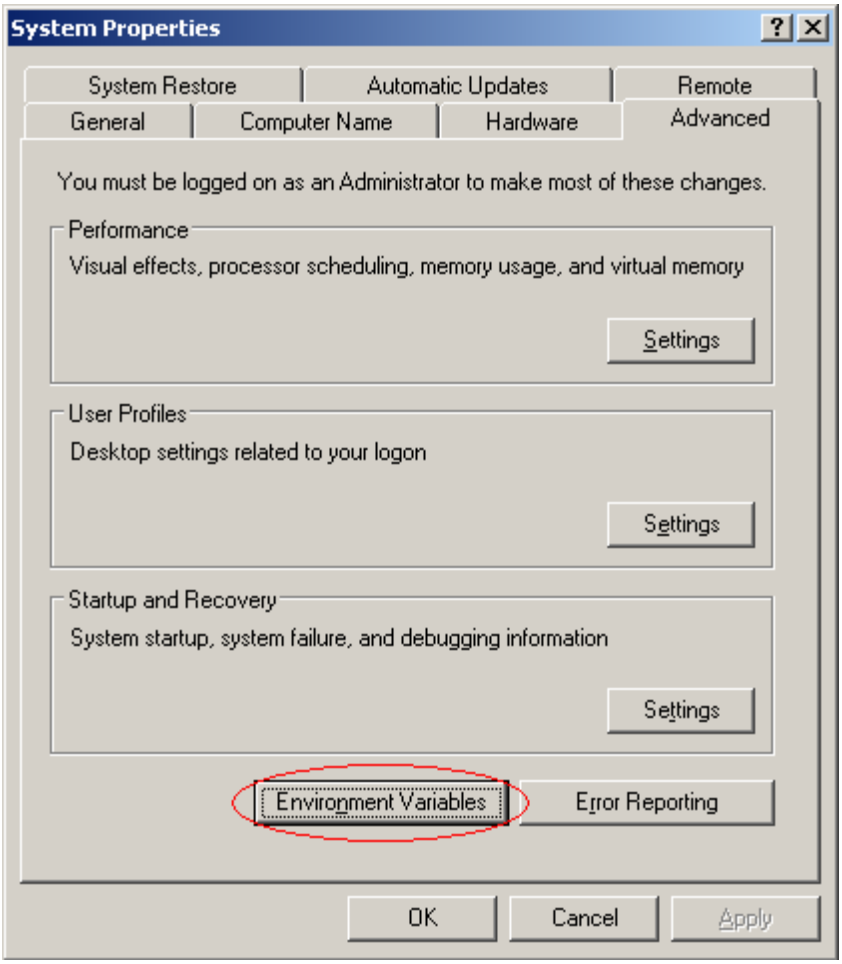
Hay un paso crucial para que todo funcione correctamente. Hay que asegurarse de que la variable de entorno PATH de Windows esteécorrectamente configurada para Cygwin, GNUARM y OCDremote.

c:\Cygwin\bin
c:\program files\gnuarm\bin
c:\Cygwin\usr\local\bin

Para verificar que la variable este correcta hay que ir al menú
“Inicio -> Panel de Control -> Sistema ->Avanzado”



Y luego “Environment Variables” (variables de entorno).

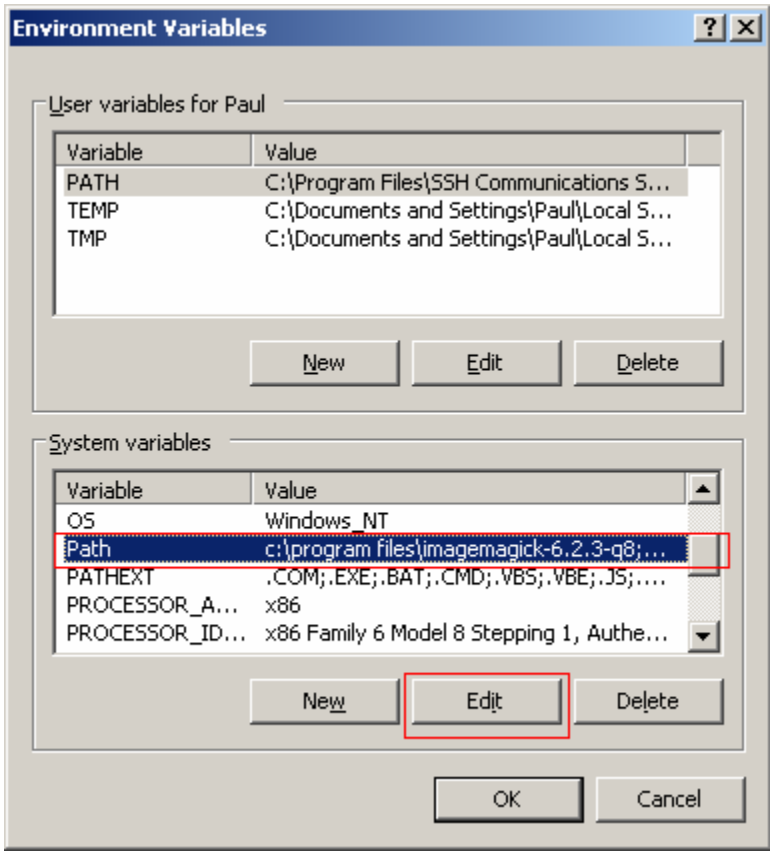


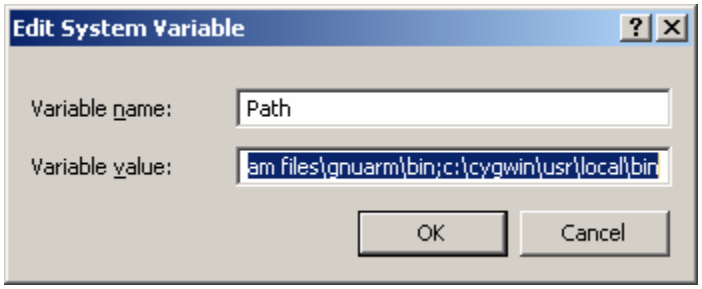
Seleccionamos la variable “Path” y luego hacemos click en “Edit” y verificamos que este la siguiente ruta:

c:\Cygwin\bin;c:\program files\gnuarm\bin;c:\Cygwin\usr\local\bin

El no tener correctamente configurada la path suele ser el principal foco de error al trabajar con Eclipse, por lo que hay que tener especial cuidado en que este correctamente escrito.

Y finalmente hemos terminado con la configuración de Eclipse para que pueda utilizarse para programar microcontroladores ARM (la familia Philips LPC2000 en específico).


Environment Variables
User variables for Paul
Variable
Value
PATH
C:\Program Files\SSH Communications S...
TEMP
C:\Documents and Settings\Paul\Local S...
TMP
C:\Documents and Settings\Paul\Local S...
New
Edit
Delete
System variables
Variable
Value
OS
Windows_NT
Path
c:\program files\imagemagick-6.2.3-q8;...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...
x86
PROCESSOR_ID...
x86 Family 6 Model 8 Stepping 1, Authe...
New
Edit
Delete
OK
Cancel

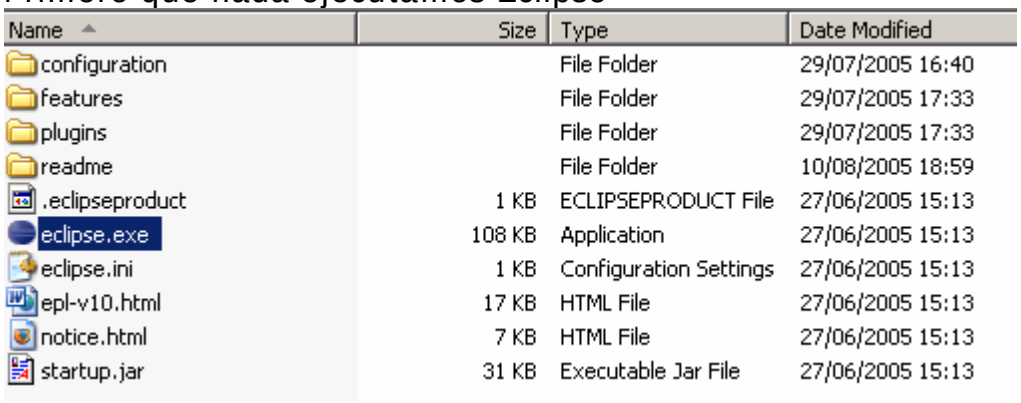

Edit System Variable
Variable name:
Path
Variable value:
am files\gnuarm\bin;c:\cygwin\usr\local\bin
OK
Cancel

Creando un Proyecto con Eclipse.

Hasta aquí tenemos Eclipse funcionando correctamente con lo cual podemos compilar programas escritos en C/C++ para microprocesadores ARM.

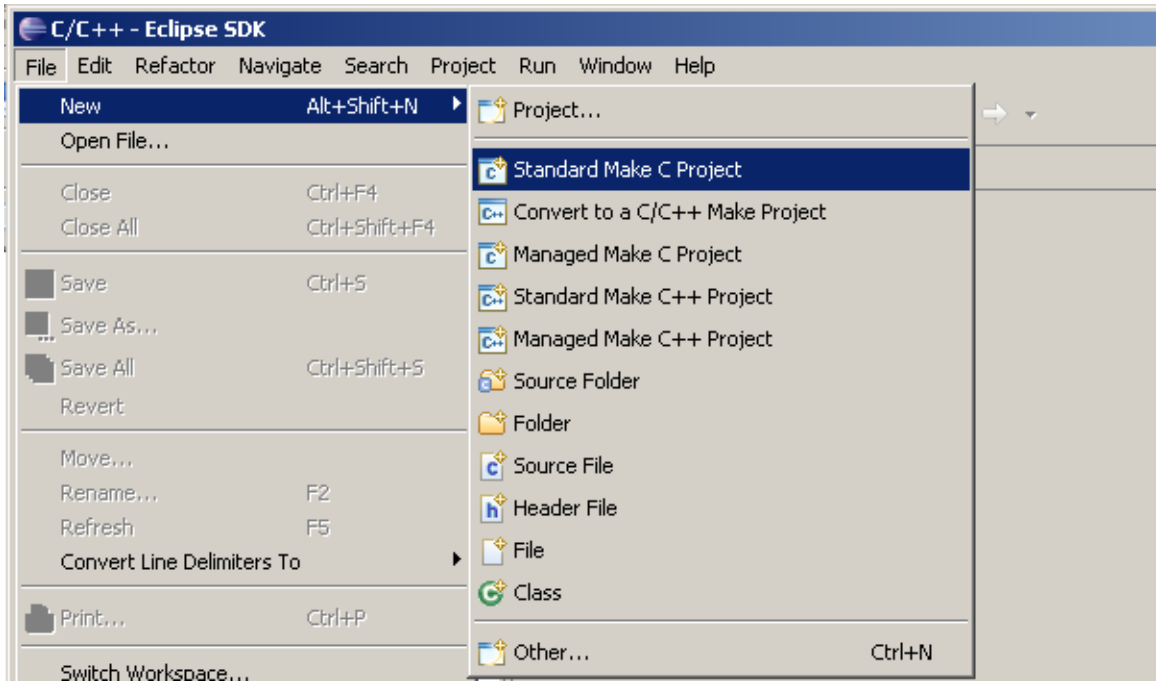
Ahora crearemos un proyecto llamado “demo2106_blink_flash” el cual encenderá el LED que está conectado a el puerto P0.7. Esta demo no utiliza interrupciones y corre utilizando la memoria flash del microcontrolador.

Primero que nada ejecutamos Eclipse



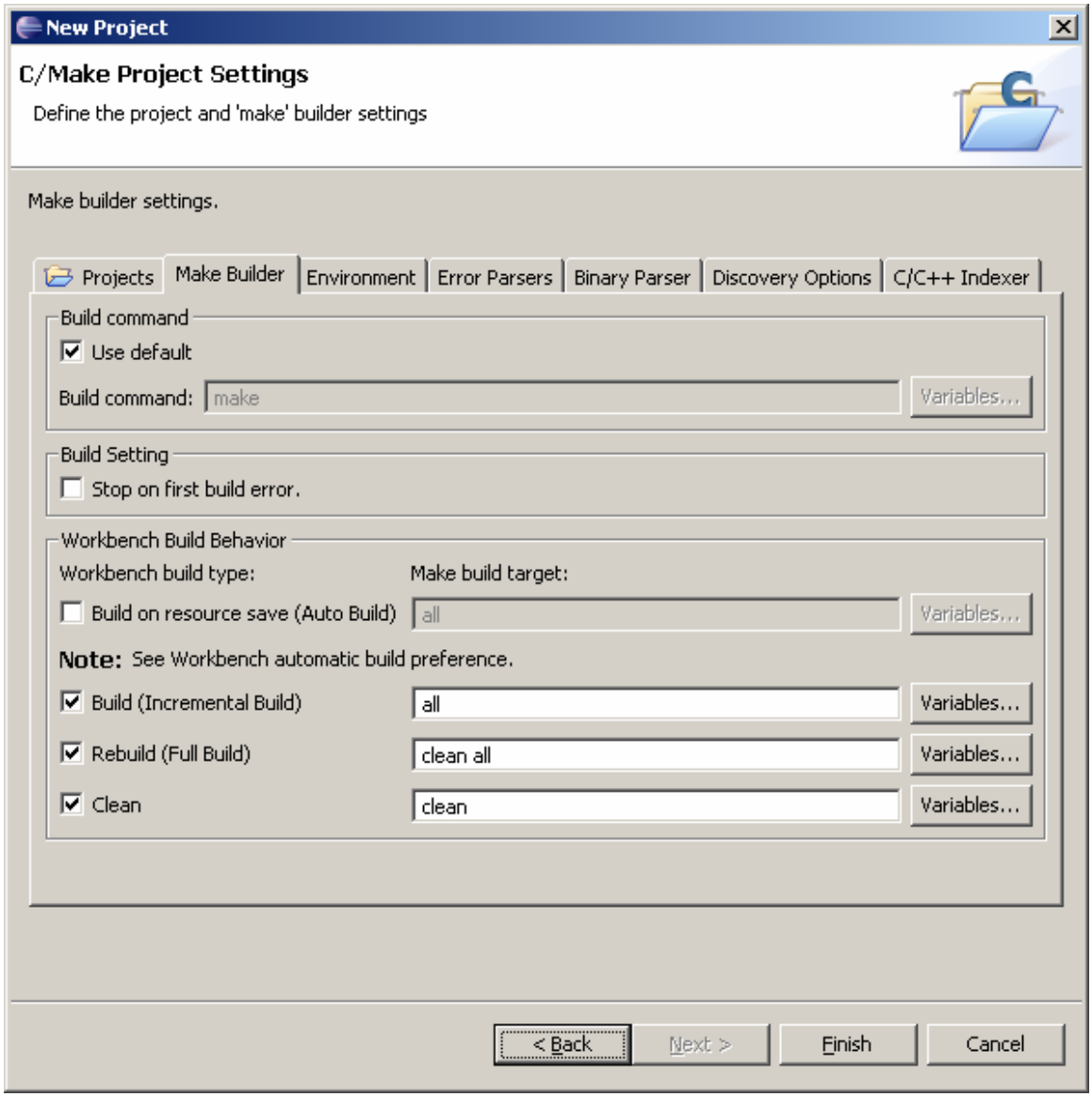
Name	Size	Type	Date Modified
configuration		File Folder	29/07/2005 16:40
features		File Folder	29/07/2005 17:33
plugins		File Folder	29/07/2005 17:33
readme		File Folder	10/08/2005 18:59
.eclipseproduct	1 KB	ECLIPSEPRODUCT File	27/06/2005 15:13
eclipse.exe	108 KB	Application	27/06/2005 15:13
eclipse.ini	1 KB	Configuration Settings	27/06/2005 15:13
epl-v10.html	17 KB	HTML File	27/06/2005 15:13
notice.html	7 KB	HTML File	27/06/2005 15:13
startup.jar	31 KB	Executable Jar File	27/06/2005 15:13

Ahora vamos al menú “File->New->New Project->Standard Make C Project” y luego hacemos click en el botón “Next”



Ahora seleccionamos el nombre del proyecto en “Project Name” (demo2106_blink_flash) como en el cuadro que se muestra mas abajo, luego hacemos click en “Next”.

En la siguiente pantalla seleccionamos el menú “Make Builder” y verificamos que la configuración sea la siguiente:



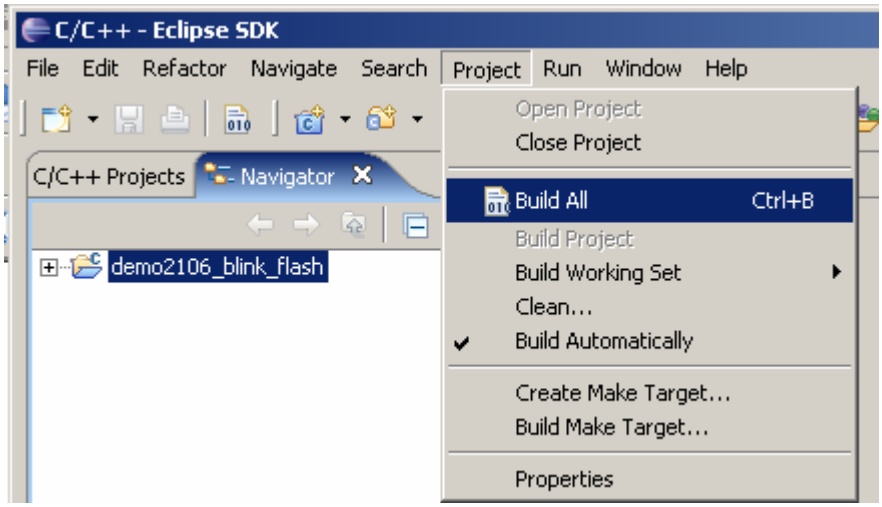
Recordemos que Cygwin, el cual instalamos anteriormente en el tutorial provee el archivo make.exe el cual debe estar en c:\Cygwin\bin.

Este es un buen momento para ver cuales son las diferencias entre “Build All”, “Build Project” y “Clean”

Build All Ejecutará el comando “make clean all” el cual primero limpia (borra) todos los objetos, y archivos de salida y luego re compila todo.

Build Project Ejecutará el comando “make all” Este no limpiará nada y sólo compilara los archivos que no estén actualizados

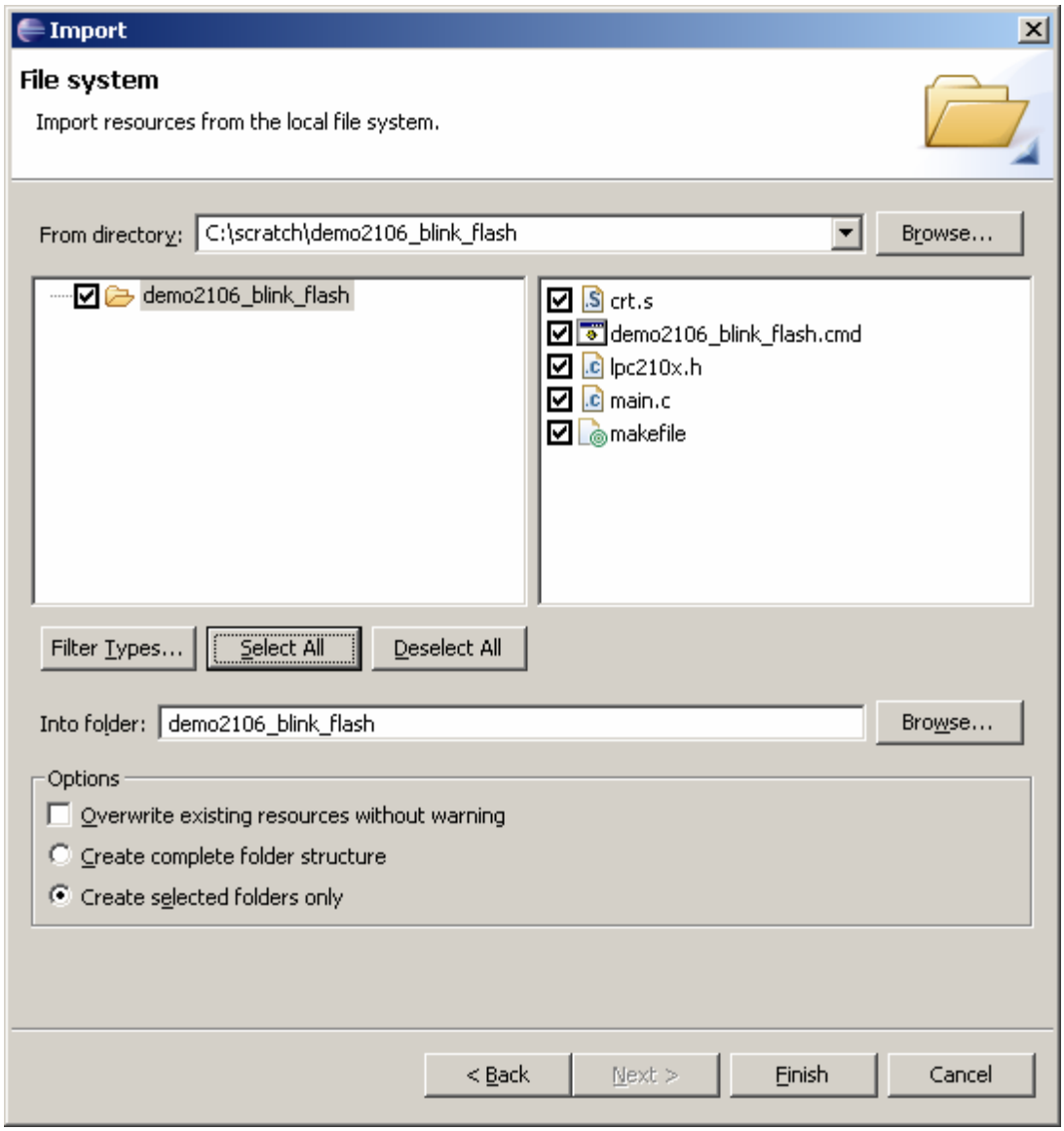
Clean Ejecutará el comando “make clean” el cual limpiará (borra) todos los objetos y archivos de salida



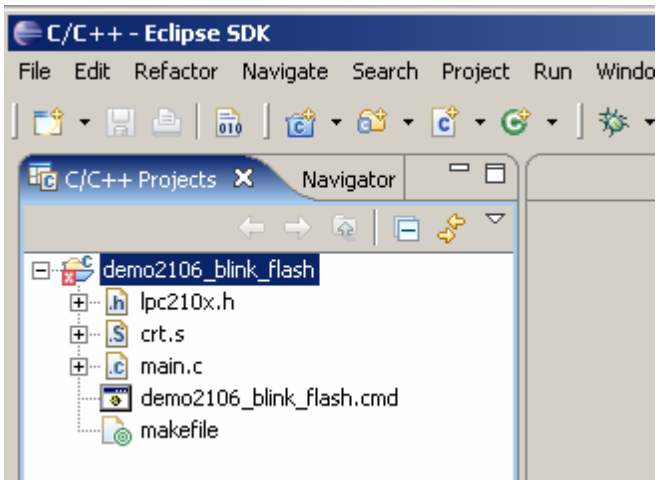
Esto es lo mismo que abrir una ventana DOS y tipear directamente un comando como:

>make clean all

Ahora Eclipse nos muestra en el lado izquierdo los proyecto C/C++. Por ahora, no tenemos ningún archivo. Importemos los archivos fuente al proyecto. Asumiendo que ya tenemos descomprimido el archivo “demo2106_blink_flash.zip” el cual esta asociado a este tutorial, haciendo click en “File->import” y luego en el menú “Import” click en “File System”. Cuando la ventana “Import->File System” aparezca, hacemos click en el botón “Browse” y buscamos el directorio en donde tenemos descomprimidos los archivos del tutorial en nuestro caso C:\scratch\demo2106_blink_flash



Luego seleccionamos todos los archivos y hacemos click en “Finish”. Una vez realizado esto podremos ver los archivos en el navegador de Eclipse



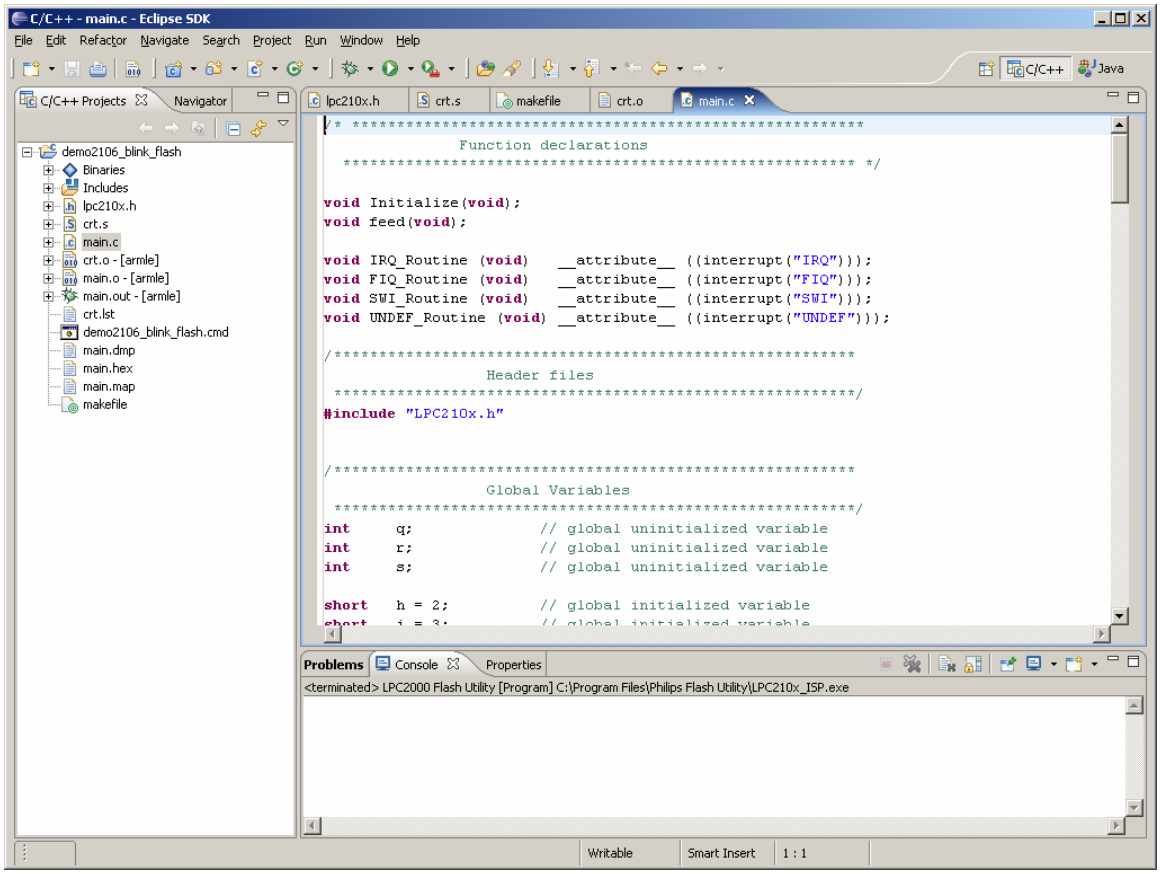
Es una buena idea identificar los archivos que hemos importado:

Descripción de los archivos del Proyecto

Lpc210x.h	Archivo Header Standard para el LPC2106
crt.c	Archivo de inicialización assembler
main.c	Programa principal en C
makefile	GNU make file
demo2106_blink_flash.cmd	GNU Linker script

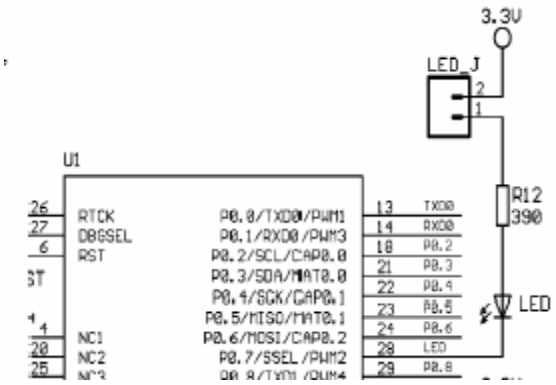
Descripción del programa principal main.c

Veamos como es el programa principal haciendo doble click sobre el archivo main.c al costado izquierdo.



El programa principal comienza con algunas funciones prototipo. Nota que las rutinas de interrupción descritas en crt.s se encuentran en main(). Hemos estado utilizando la sintaxis del compilador GNU C que identifica las rutinas de interrupción y hace que el compilador guarde y restaure los registro, etc, cuando se produce una interrupción.

Lo que haremos con este programa es cambiar el estado del la salida I/O del puerto P0.7 de la tarjeta Olimex LPC-P2106



Con esta configuración del hardware tenemos que:

P0.7 =1 // LED Apagado
P0.7=0 // LED Encendido

El Philips LPC2106 tiene 32 pines de I/O, etiquetados desde el P0.0 al P0.31. La mayoría de estos pines tienen 2 o tres posibles usos. Por ejemplo, el pin P0.7 tiene tres posibles usos: Puerto de entrada y salida digital I/O, SPI slave select y PWM. Normalmente, se selecciona que función se utilizará con el Pin Connect Block. El pin connect Block está compuesto por dos registros de 21 bits, PINSEL0 y PINSEL1. Cada registro Pin Select tiene dos bits para cada pin I/O, permitiendo de esta forma seleccionar al menos 3 funciones para cada pin.

Por ejemplo, el pin P0.7 es controlado los bits 14 y 15 del PINSEL0. La siguiente configuración sería la adecuada para utilizarlo como salida PWM2.

```
PINSEL0 = 0x00008000;
```

Afortunadamente, el PIN Connect Block se setea en 0 por defecto, lo que significa que todos los pines quedan configurados como I/O de propósito general. Debido a esto no es necesario configurar el registro Pin Select en este ejemplo.

Lo que si tenemos que hacer es configurar la dirección del puerto I/O P0.7, esto se hace de la siguiente manera:

```
IODIR |= 0x00000080; //configura el puerto P0.7 como salida
                    // 1 = salida, 0 = entrada
```

Los puertos I/O del ARM son manipulados por el registro IOSET y IOCLR. Nunca se escribe directamente en el puerto, se utiliza el bit correspondiente en el registro IOSET para activar un bit y se utiliza el bit correspondiente en el registro IOCLR para limpiar ese bit. Los lectores podrán preguntarse que pasa si el mismo bit es seteado en ambos registros?, la respuesta es “El primero gana”. La última instrucción que utilice IOSET o IOCLR prevalece.

Volviendo al ejemplo, para apagar el LEB P0.7, podemos escribir:

```
IOSET = 0x00000080;
```

Ahora para encender el LED utilizamos la instrucción:

IOCLR = 0x00000080;

Como se puede ver es relativamente sencillo manipular los puertos de I/O del ARM.

Para hacer que el LED parpadee para siempre hacemos un loop infinito. Utilizamos el valor 5000000 en el contador para esperar medio segundo entre cada cambio de estado del led.

```
while (1) {
for (j = 0; j < 5000000; j++ ); / / wait 500 msec
IOSET = 0x00000080; / / red led off
for (j = 0; j < 5000000; j++ ); / / wait 500 msec
IOCLR = 0x00000080; / / red led on
}
```

Este esquema es ineficiente ya que pierde ciclos del CPU sin hacer nada, pero sirve para ejemplificar el uso del puerto.

La función Initialize(); requiere un poco más de explicación. Hay que configurar el PLL (Paced Lock Loop) y para esto necesitamos un poco de matemáticas.

La placa Olimex LPC-P2106 tiene un cristal de 14.7456Mhz. Si queremos que el LPC2106 corra a 53.2368 Mhz (hay que multiplicar el valor del cristal por 3 en este caso)

De acuerdo con el manual del LPC2106:

M=cclk/Fosc
Donde
M = es el multiplicador del PLL (bits 0-4 del PLLCGF)
clck = 53236800 Hz
Fosc = 14745600 hz

Resolviendo M=53236800/14745600=3.6103515625
Si redondeamos M=4

Nota que M-1 debe ser el valor que se utilice en los bits 0-4 del PLLCFG (asignado a esos 3 bits)

El CCO (Current Controller Oscilator) debe operar en el rango 156Mhz a 320Mhz

De acuerdo con el manual del LPC2106
Fcco = cclk x 2 x P
Donde:
Fcco= es la frecuencia del CCO
Cclk = 53236800 hz
P = es el divisor del PLL (bits 5-6 del PLLCFG)

Resolviendo

Fcco = 23236800 x2 xP
P=2 (valor arbitrario)
Fcco = 53236800 x2 x2
Fcco= 212947200 hz (buena elección de P ya que el valor resultante quedo entre 156 y 320 mhz)

De la tabla 19 (página 48) del manual del LPC2106 P=2, PLLCFG bits 5-6 = 1 (asignar 1 a estos bits)

Finalmente: PLLCFG= 0 01 00011 = 0x23

Nota Final: Para cargar el registro PLLCFG es necesario usar la siguiente secuencia para escribir 0XAA seguido de un 0x55 en el registro PLLFEED. Esto es llevado a cabo por la función feed()

Ahora que tenemos claro de donde vienen los valores tenemos que configurar el PLL para lo cual ocupamos las siguientes líneas:

```
// Setting Multiplier and Divider values
PLLCFG = 0x23;
feed();
```

Luego tenemos que habilitar el módulo Memory Accelerator y configurar la memoria FLASH para que corra a ¼ de la velocidad del reloj. Es por esto que mucha gente prefiere ejecutar los programas desde la memoria RAM que es mucho más rápida

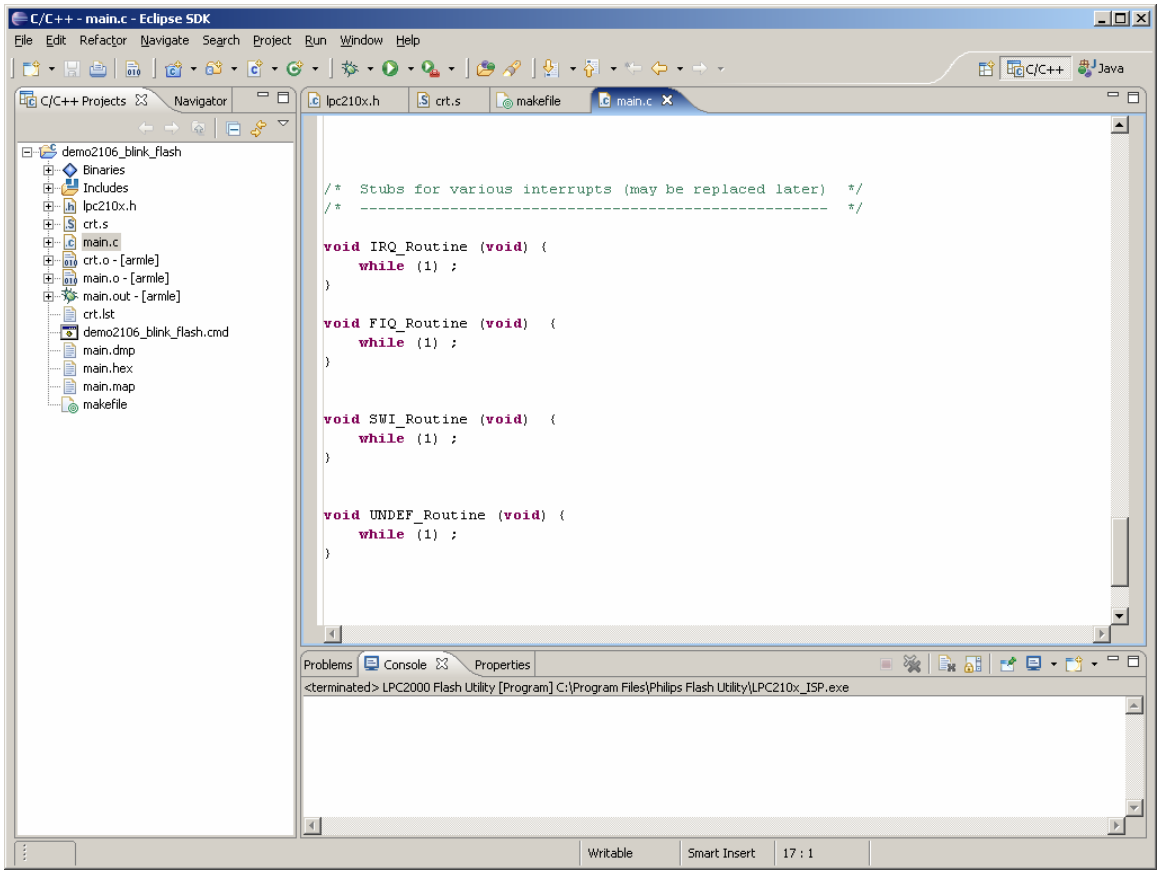
```
// Enabling MAM and setting number of clocks used for Flash memory fetch
// (4 cclks in this case)
MAMCR=0x2;
```

MAMTIM=0x4;

La velocidad del reloj de los periféricos es también 53.2Mhz

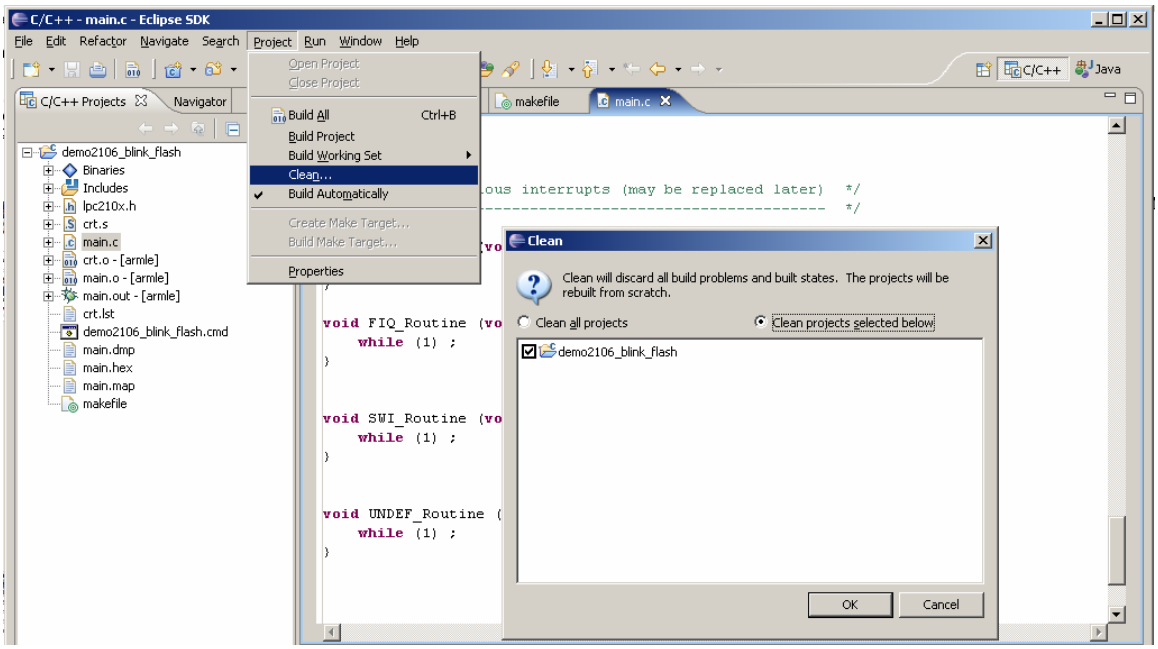
// Setting peripheral Clock (pclk) to System Clock (cclk)
VPBDIV=0x1;

Al final del código de la función main() se pueden ver alguna rutinas de atención de interrupción. Están ahí pero no hacen nada, es sólo para ilustrar como se deben declarar las interrupciones. Este ejemplo no utiliza ningún tipo de interrupciones.

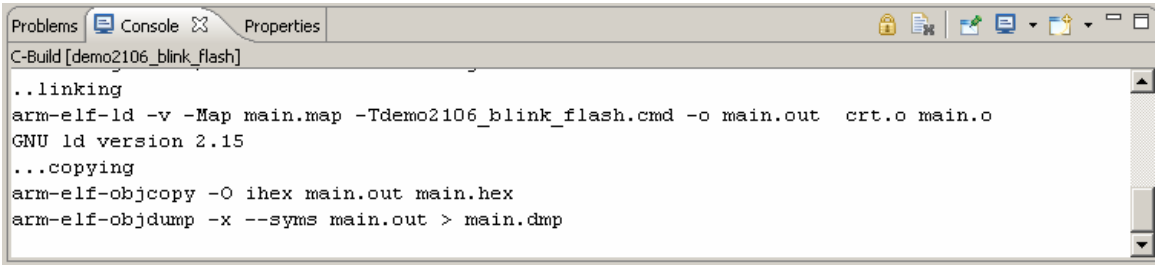


Compilando y Linkeando la aplicación.

OK, ahora es la hora de trabajar con la tarjeta. Primero hay que limpiar el proyecto “Clean”. Para limpiar el proyecto vamos al menú “Project->Clean” y marcamos el proyecto que queremos limpiar.

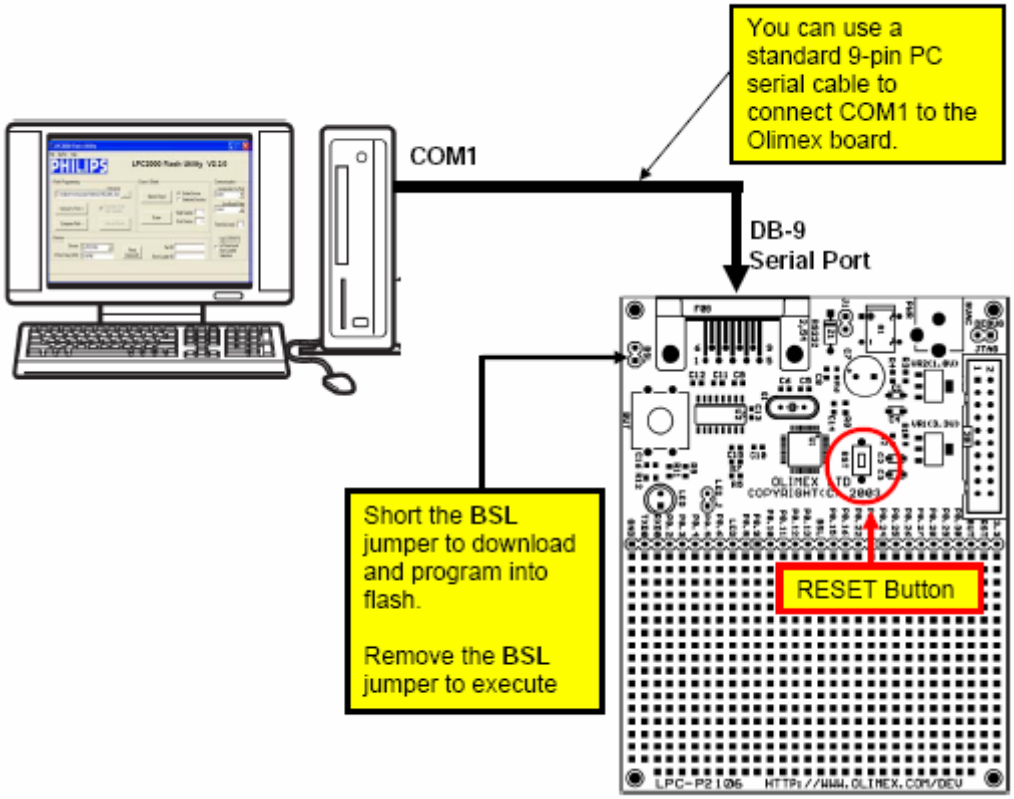


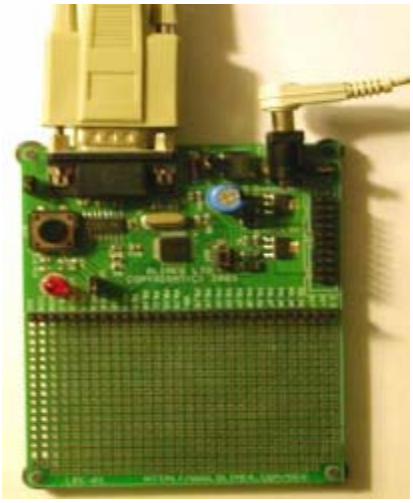
Puedes ver los resultados de la operación de limpieza en la ventana “Console” en la parte inferior de Eclipse.



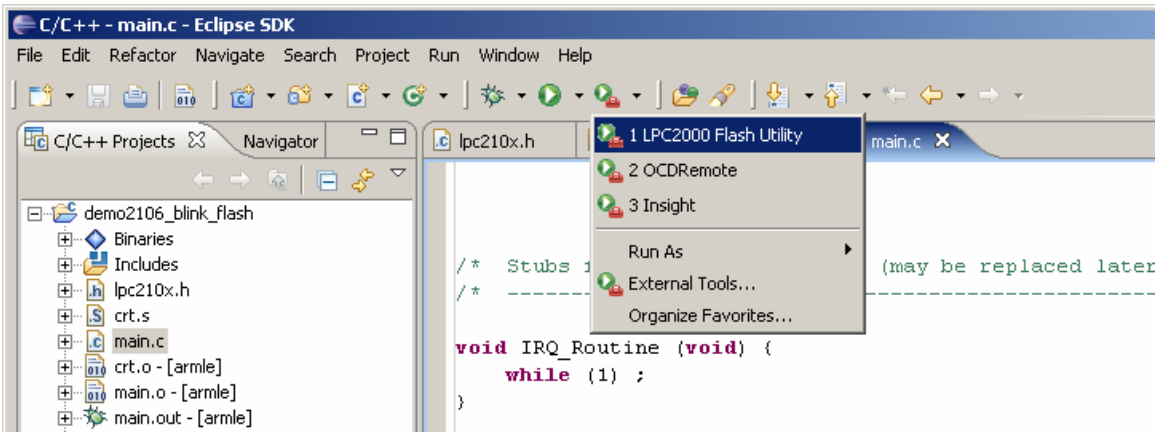
Configurando el Hardware.

Para este tutorial utilizaremos la tarjeta LPC-P2106 Prototype Board. Conectamos un cable serial DB9 desde el puerto del computador COM1 a la tarjeta LPC-P2106. Conectamos la fuente de poder de 9Volts en el conector de PWR. Dejamos cerrado el jumper BSL

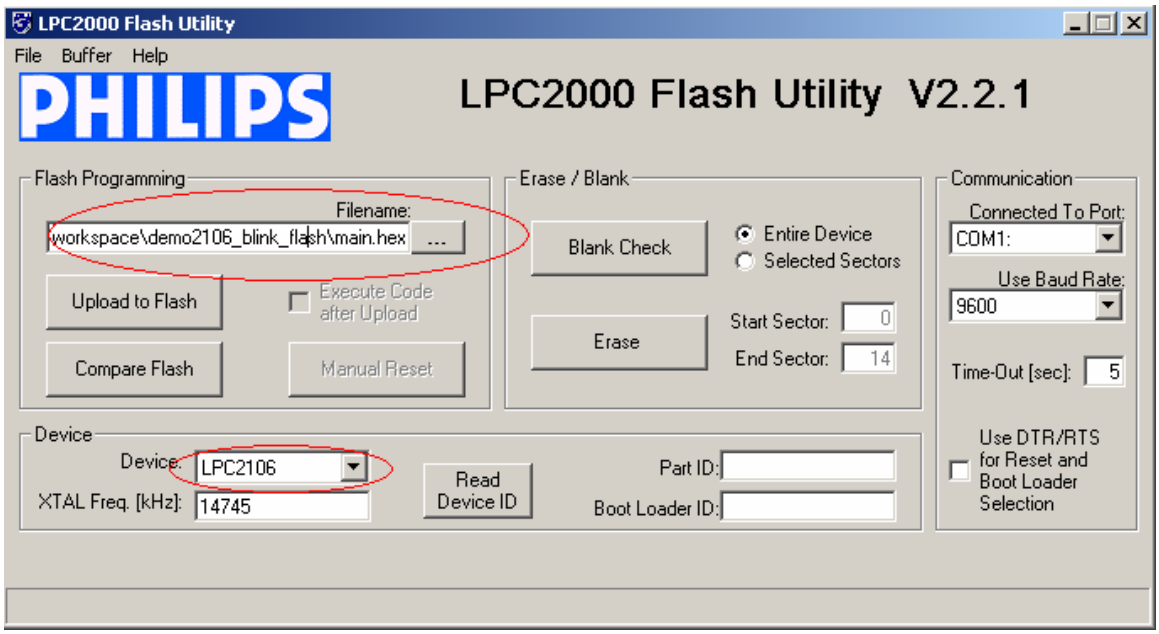




Para traspasar el archivo utilizaremos el LPC200 Flash Utility. La forma rápida de llegar a él es utilizando el botón que agregamos en “External Tools”.



El programa de Philips abrirá una pantalla

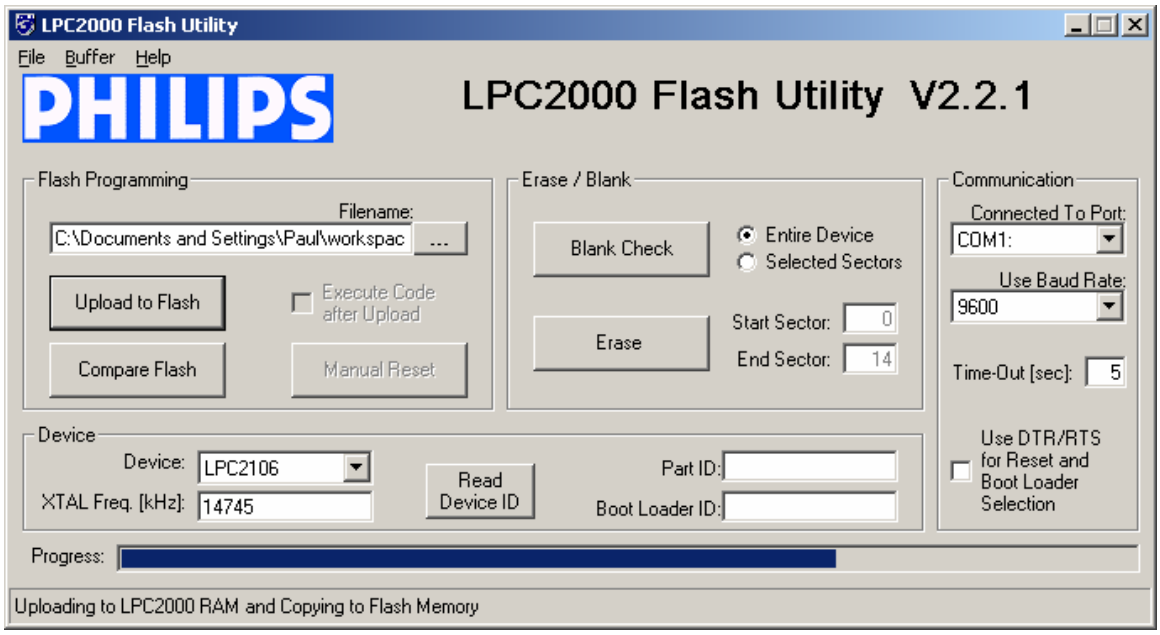


Ahora buscamos el archivo main.hex que debe estar en el workspace que escogimos para el proyecto. Seleccionamos el dispositivo LPC2106.

Ahora hacemos click en “Upload to Flash” para comenzar la descarga de nuestro programa en el microcontrolador.

El programa de Philips pedirá presionar el botón de reset antes de descargar la aplicación

La descarga comenzará ahora. Se puede ver la barra azul de progreso mientras se descarga el programa. Una vez terminado el proceso quitamos el jumper BSL (boot strap loader) y hacemos click en el botón RST.

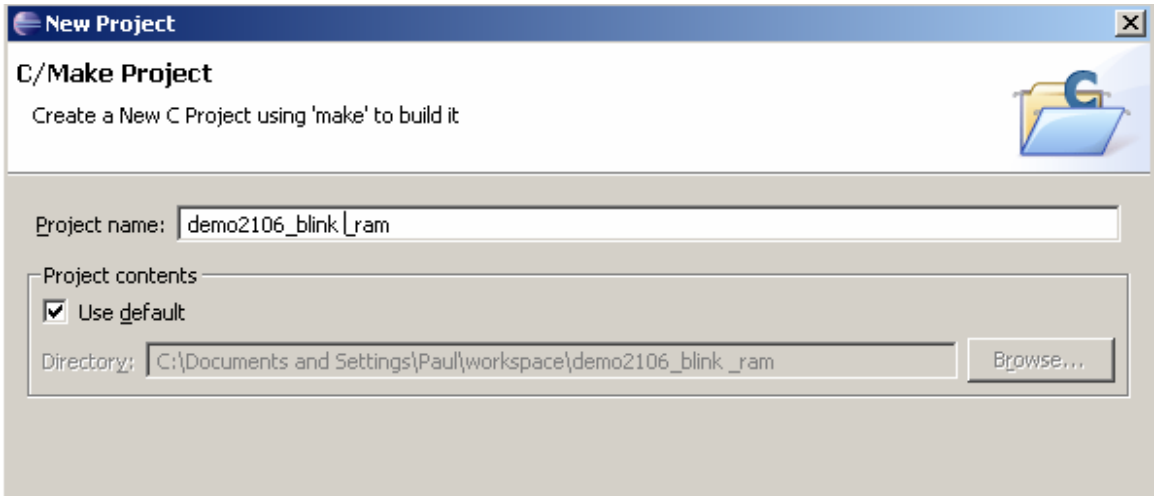


Y por fin tenemos el led funcionando!!

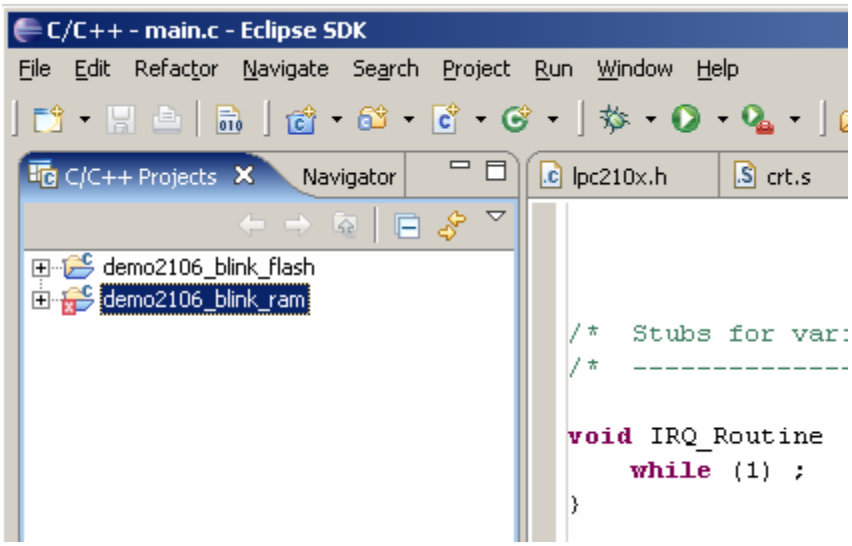
Creando un nuevo proyecto que corra en la memoria RAM

Ahora crearemos un nuevo proyecto que corra en la memoria RAM. Algunas modificaciones menores en el código son requeridas. También veremos como se utiliza la aplicación de Philips para trabajar con la memoria RAM. Luego veremos como utilizar esta aplicación con el debugger Insigth y el JTAG Wiggler.

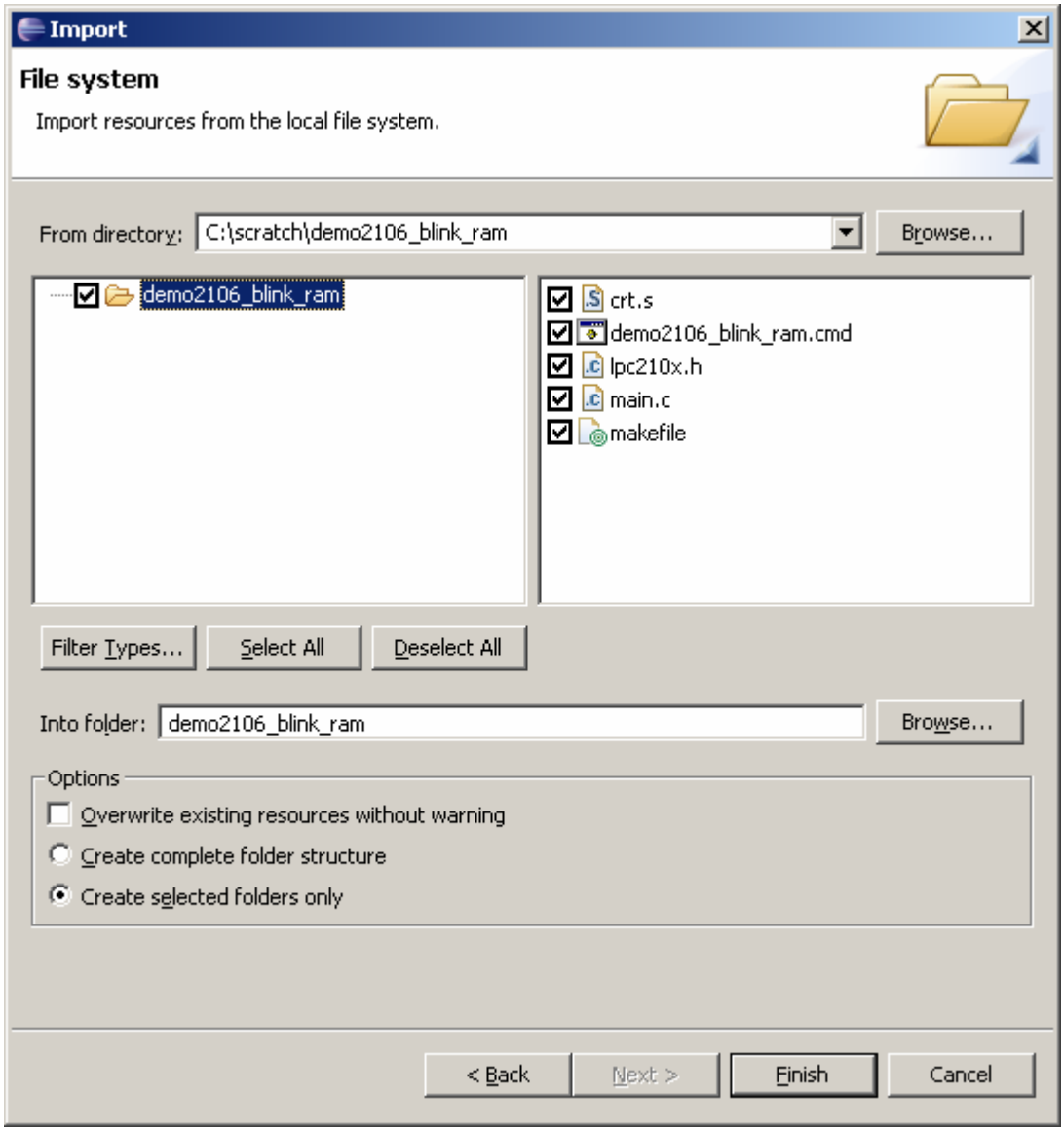
Usando los mismos procedimientos vistos anteriormente crearemos un nuevo proyecto llamado demo2106_blink_ram



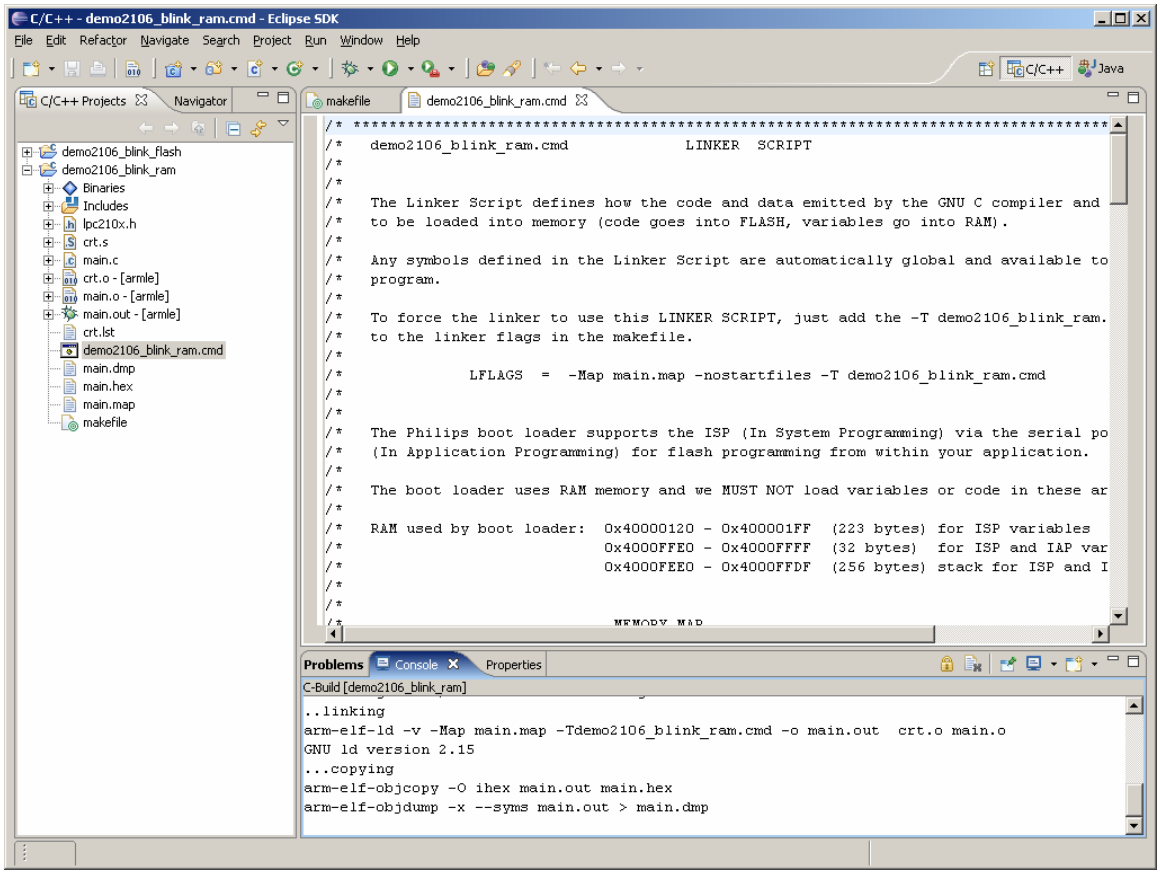
Notemos que ahora existen 2 proyectos, el nuevo no tiene archivos y aparece con una cruz roja en su ícono.



Ahora importamos los archivos de prueba usando el menú "File->Import"



Ahora si hacemos "Clean and Build" veremos nuestro proyecto compilado como resultado.

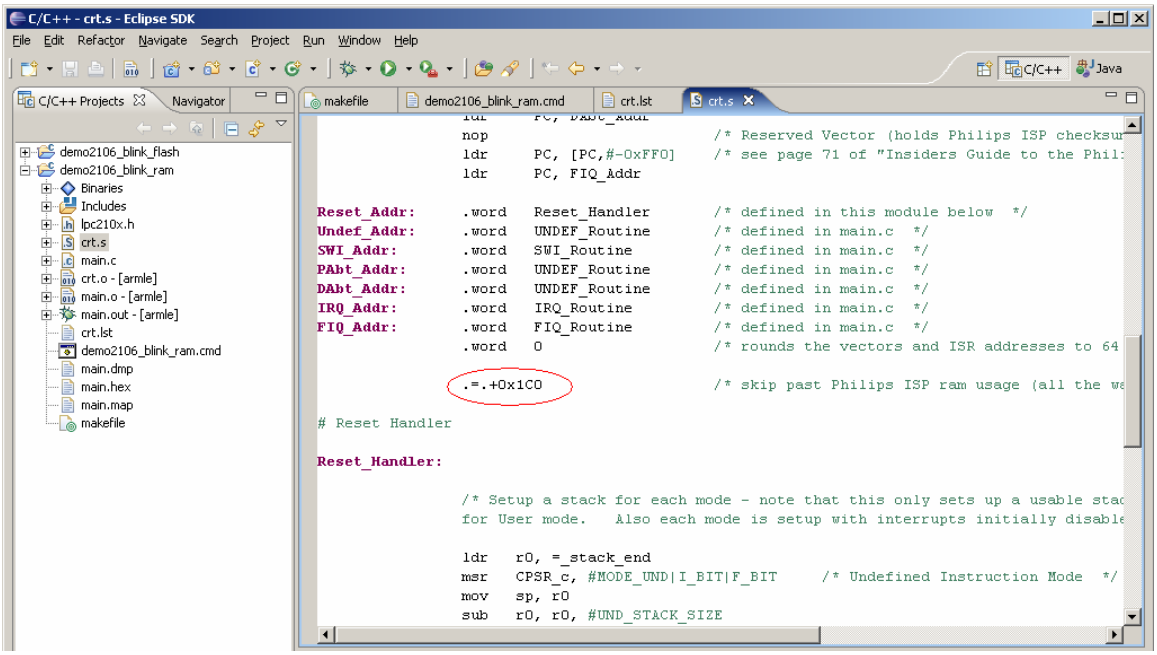


Diferencias entre la versión RAM y la versión Flash.

Archivo crt.s

En el archivo de inicialización en asemmbler usamos un pequeño truco para que el código no se mezcle con alguna otra información existente en el área baja de la RAM.

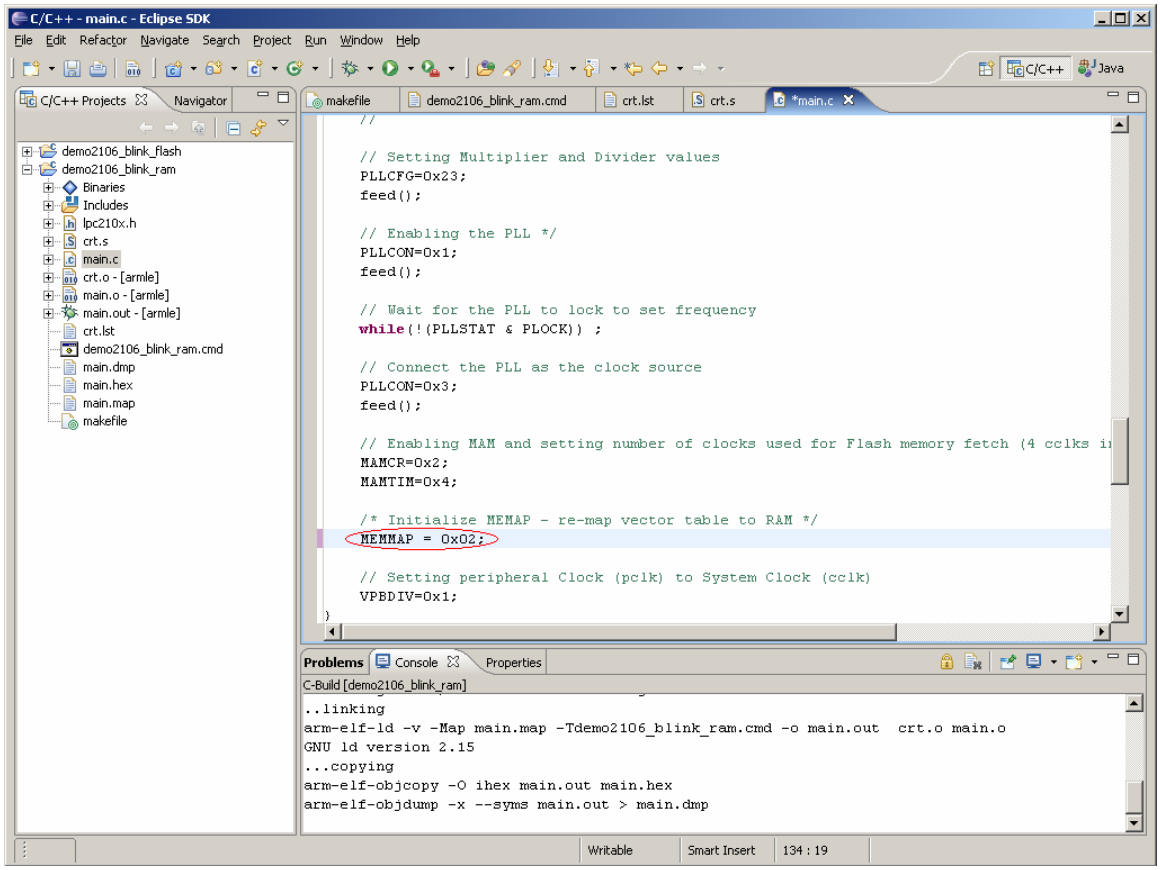
Recordemos que en este proyecto el código y las variables serán cargadas en la memoria RAM partiendo de la dirección 0x40000000. La ubicación del contador es aumentada de la siguiente forma .=.+0x1C0 para mover el Reset_Handler a la dirección 0x40000200. Esto deja un espacio donde la ISP Flash Utility de Philips puede utilizar la memoria RAM. Hay otras formas de hacer esto.



Archivo main.c

Aquí hay sólo una línea adicional. Esta línea re mapea la dirección de el vector de interrupción a la RAM x40000000.

Como no estamos utilizando interrupciones en este ejemplo esto en realidad no era necesario. Pero igual agregamos la línea para tener una referencia de lo que se debiera hacer para que funcione correctamente con interrupciones.

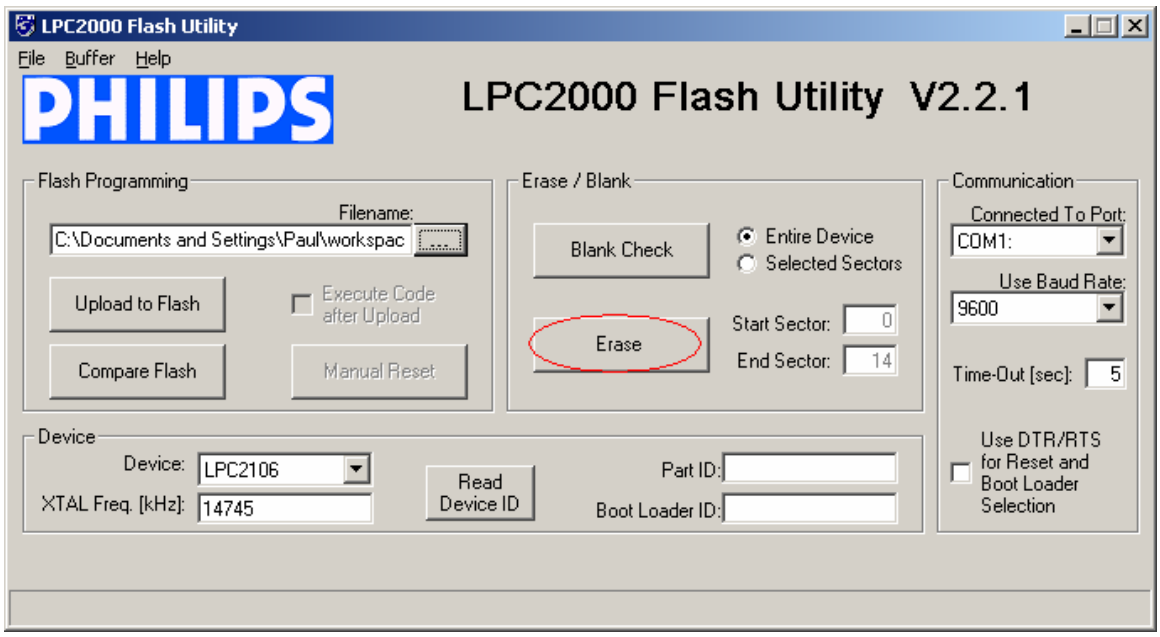


Antes de seguir y ejecutar la aplicación hagamos un pequeño experimento y comentemos la línea MEMMAP = 0x02. El programa aún funcionará correctamente.

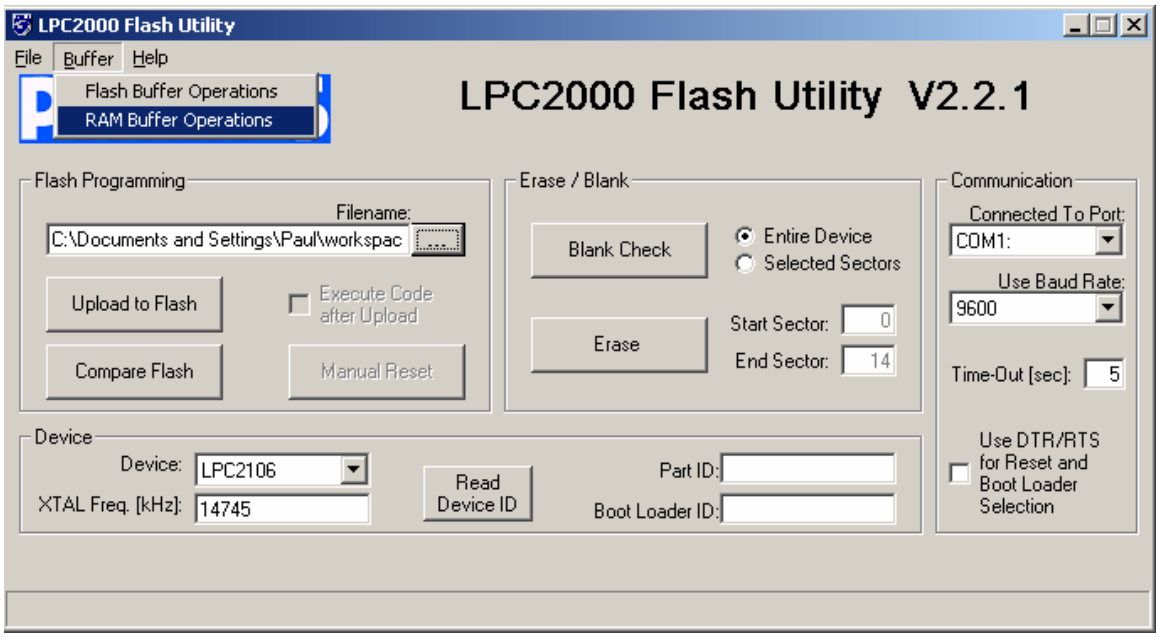
Hay dos razones para esto. La primera es que en este ejemplo no estamos utilizando interrupciones y la segunda es que el ISP Flash Loader de Philips fuerza a la CPU a empezar en la dirección del Reset_Handler, la cual es 0x40000200.

Si en este punto no has limpiado el proyecto, hazlo ahora. Asegúrate de que el jumper BSL esté instalado.

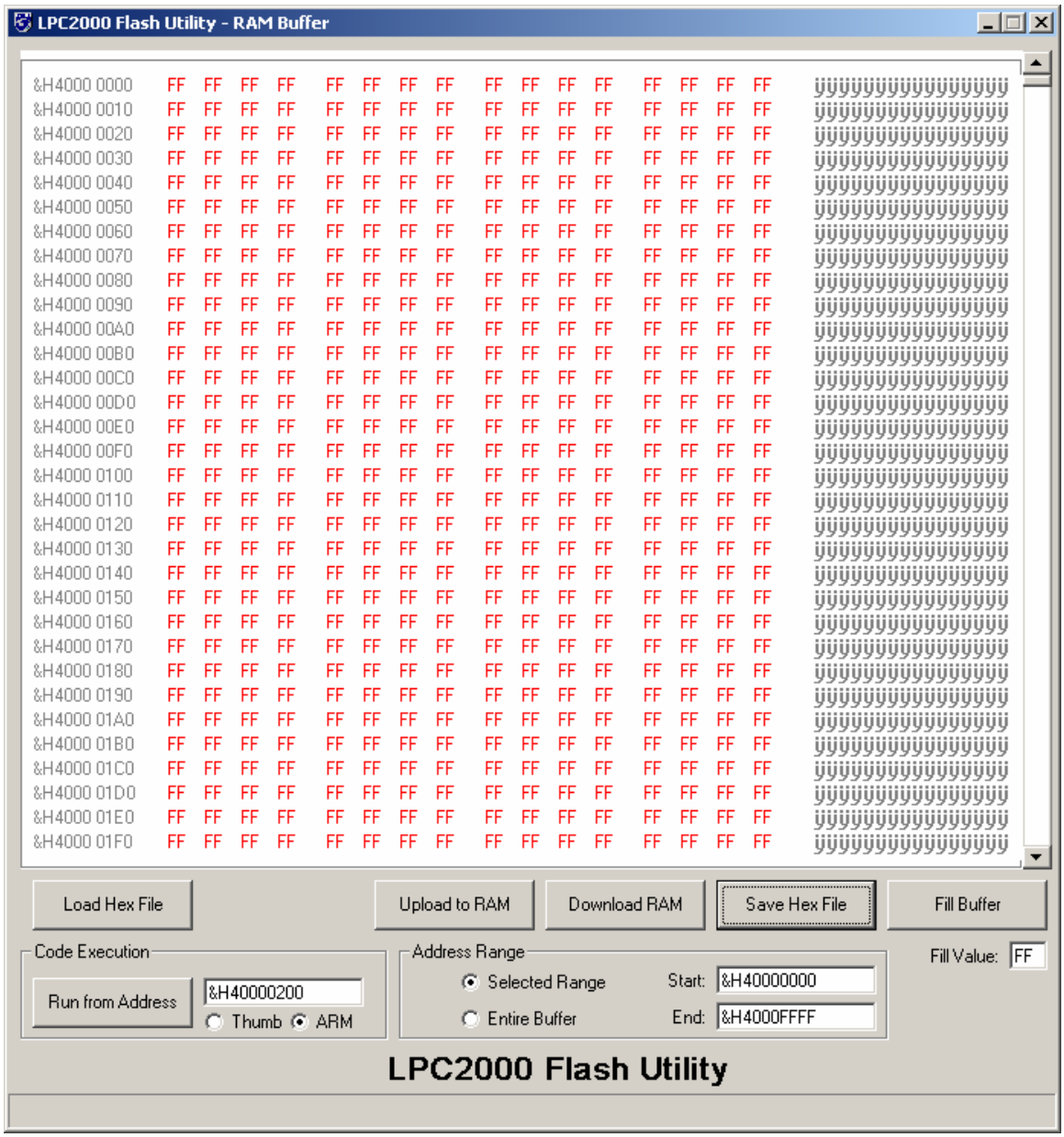
Ahora utilizaremos la aplicación de Philips desde el botón “External Tools”. Hacemos click en “Erase” para limpiar la memoria Flash.



Ahora podemos estar seguros que el programa ya no está en la memoria Flash. Ahora vamos a Buffer y seleccionamos “RAM Buffer Operations”

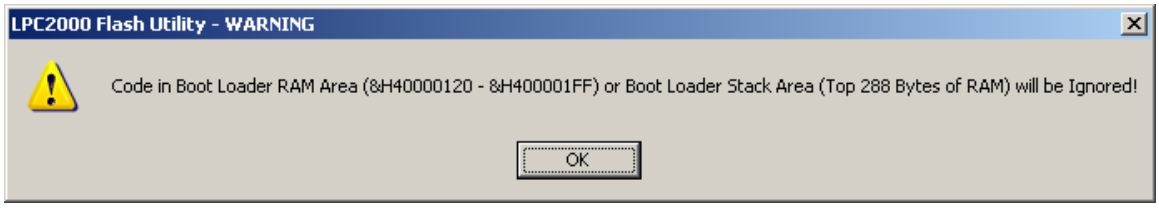


Ahora aparecerá en pantalla la siguiente ventana

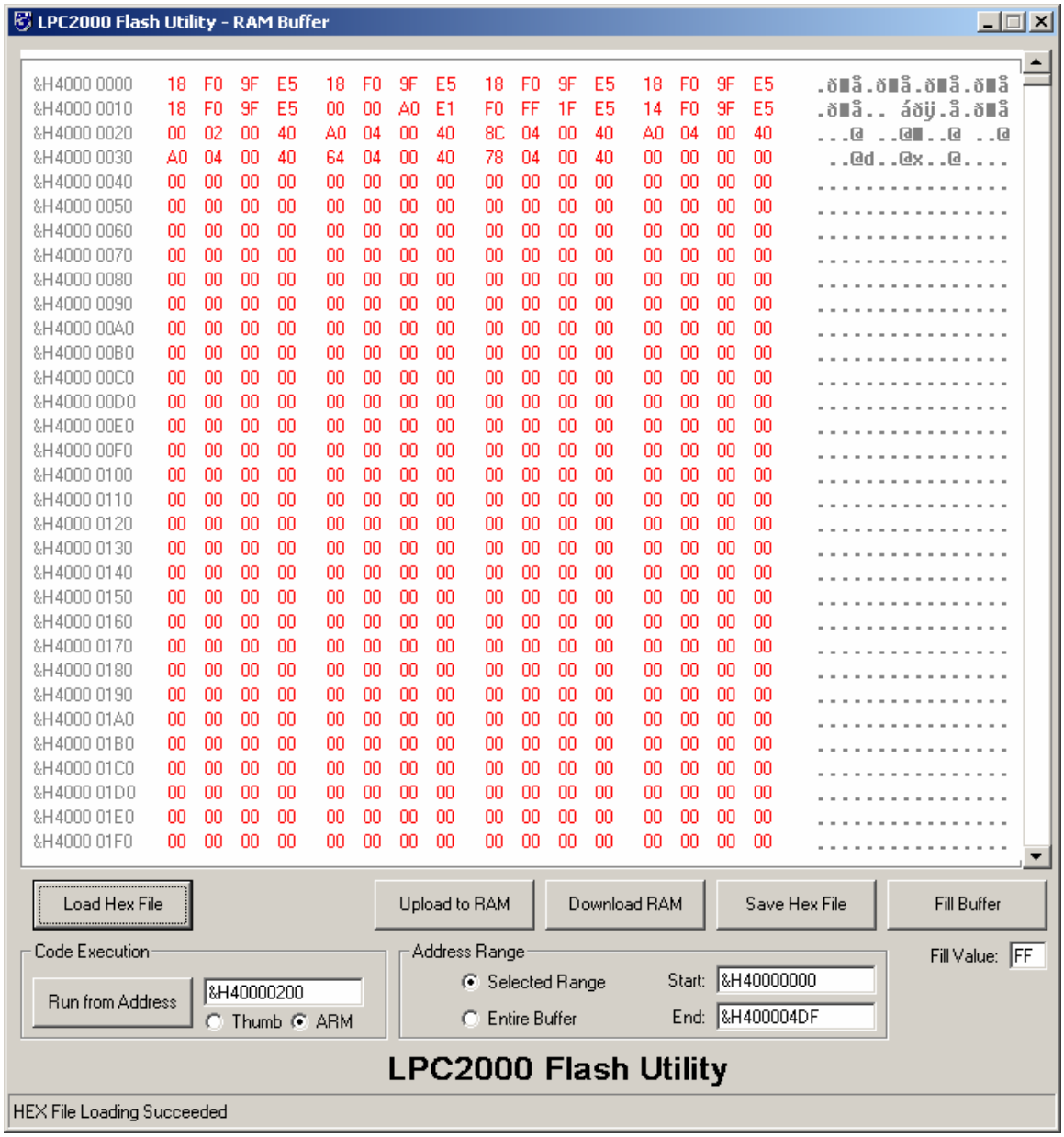


Hacemos click en “Load Hex File”. Con esto estamos diciéndole al programa de Philips que archivo hex programar.

Al cargar el archivo aparecerá el siguiente mensaje de advertencia, el programa aun correrá bien ya que nos preocupamos de no poner datos en esta área de la memoria



Ahora la pantalla se verá de la siguiente forma. Hacemos click en Upload to RAM y la barra de progreso comenzará a moverse. **No** es necesario remover el jumper BSL. Hacemos click en "Run from Address" para ejecutar el programa.

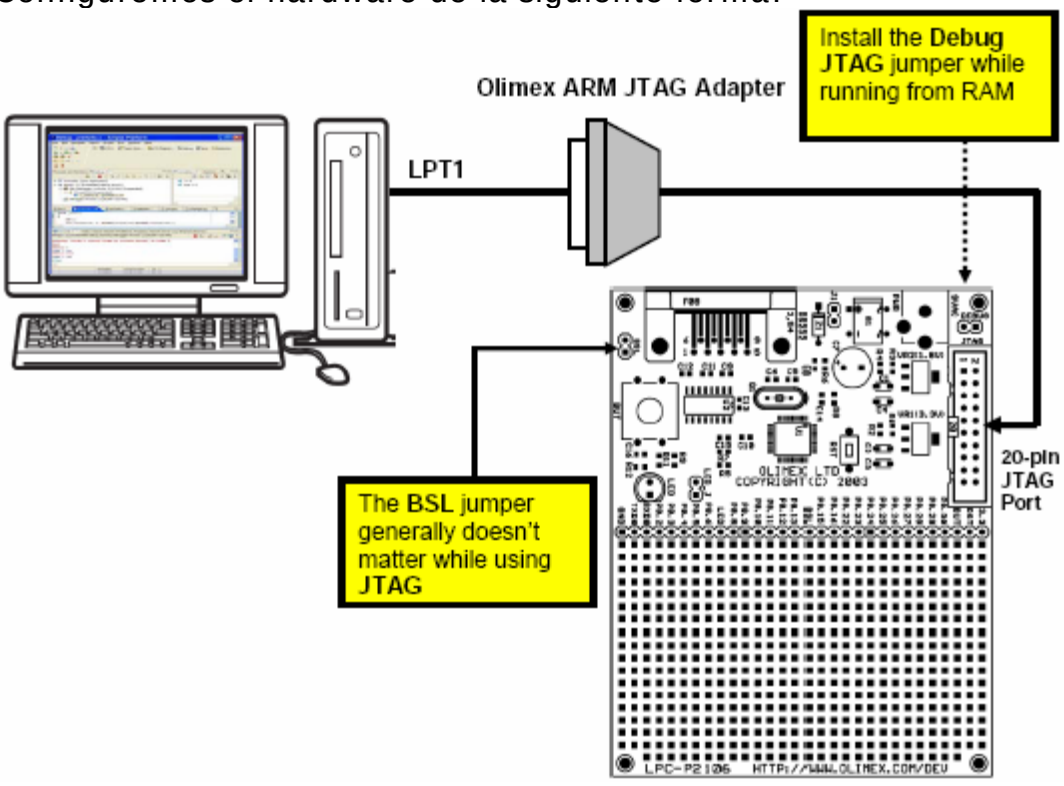


Y nuevamente tenemos la luz parpadeando

Ejecutando el programa en la RAM con el Insight Debugger.

El ejercicio previo trataba sobre como correr una aplicación en la memoria RAM lo cual tiene fines académicos ya que en esencia no tiene valor práctico.

Configuremos el hardware de la siguiente forma:



El JTAG de Olimex es un clone de Macgraigor Wiggler. Este se conecta en la extensión de 20 pines que viene en la placa LPC-P2106. El cable de color rojo debe ir hacia el lado de la alimentación. Haz estas conexiones con la alimentación desconectada.

Cuando se utiliza el JTAG no importa si se utiliza el jumper BSL.

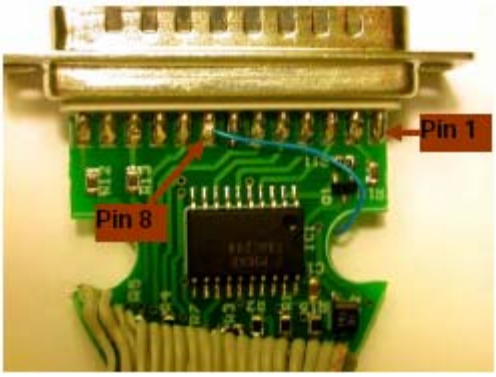
Sobre el Wiggler.

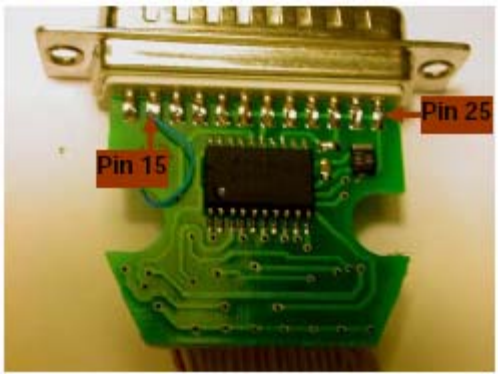
Wiggler es uno de los productos de la compañía canadiense Macraigor. Este se conecta por el puerto paralelo al PC.

Existen muchos esquemáticos en la web para el Wiggler. Obviamente la compañía Macraigor no está muy contenta con los "clones" de su producto, por lo que pusieron un pequeño impedimento para que éstos funcionaran. La última versión de OCDRemote espera un cortocircuito entre los pines 8 y 15 del puerto paralelo.

Olimex ha modificado sus programadores y conectado estos pines haciendo un cortocircuito entre ellos, sin embargo los JTAG que fueron vendidos anteriormente no tienen esta modificación.

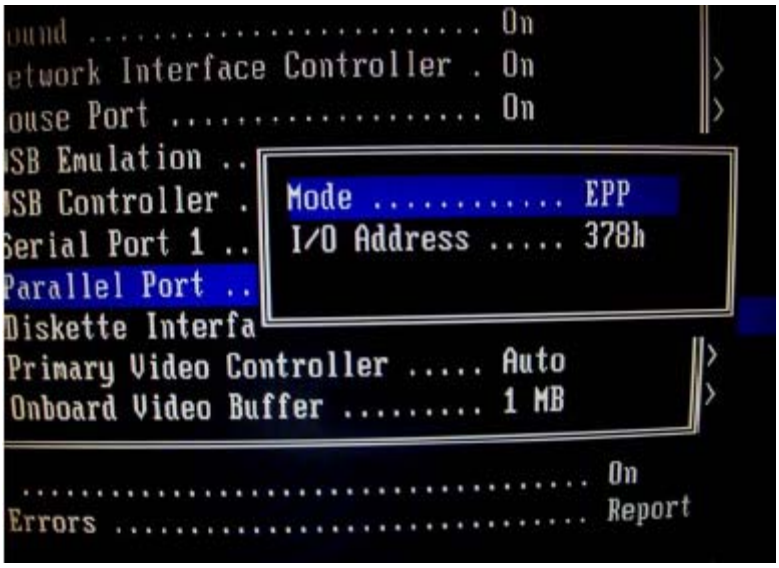
Para verificar si existe el cortocircuito es posible utilizar un tester entre los pines 8 y 15 y medir la resistividad. En caso de no estar cortocircuitados basta con soldar un cable entre estos pines.





Otra cosa que tenemos que tener en cuenta con el Wiggler es que éste no funciona con el puerto paralelo configurado como ECP. Para poder

cambiar este modo tenemos que ingresar a la bios del PC. En la mayoría de los PC's esto se hace presionando la tecla "supr" apenas suena el beep al encender el PC.



El Wiggler no puede hacer operaciones breakpoint en la memoria FLASH. La aplicación OCDremote no puede trabajar con los comandos GDB –z los cuales se refieren a breakpoints. Esta es la razón por la cual las aplicaciones que queramos debuggear están limitadas a correr exclusivamente en la RAM. Obviamente los 64K de memoria que trae el LPC2106 es la aplicación máxima que cae en la RAM.

Preparación final antes de empezar con el Insight Debugger

Antes de comenzar con Insight hay que mencionar que los debuggers en general no se llevan muy bien con la optimización de código, y éste no es diferente. Hemos estado compilando con la opción –O3 la cual hace algunos pasos de optimización.

Sólo para asegurarnos eliminemos la optimización del código cambiando la siguiente línea:

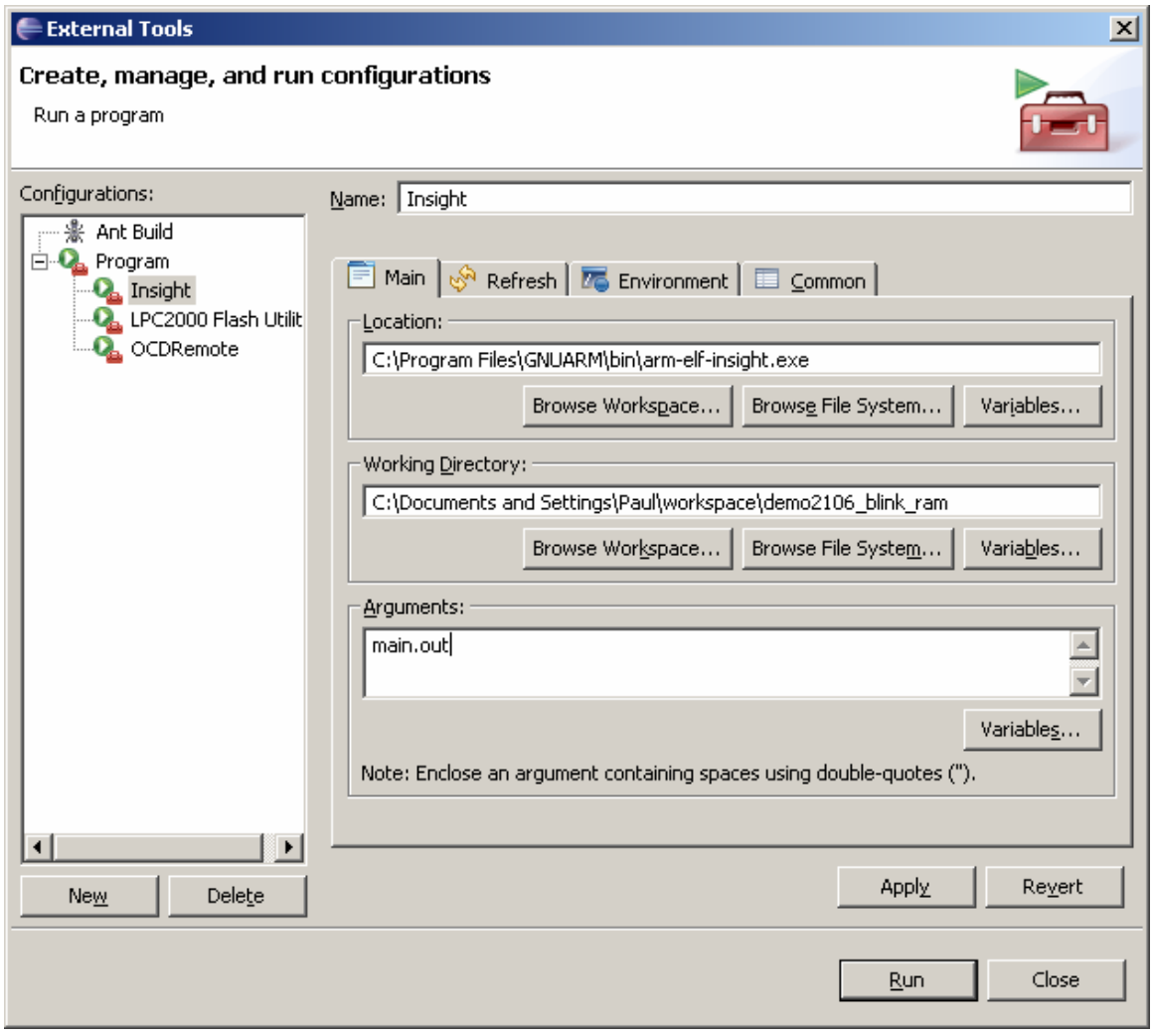
File: **makefile.mak**

```
NAME   = demo2106_blink_ram

CC      = arm-elf-gcc
LD      = arm-elf-ld -v
AR      = arm-elf-ar
AS      = arm-elf-as
CP      = arm-elf-objcopy
OD      = arm-elf-objdump

CFLAGS  = -I./ -c -fno-common -O0 -g
AFLAGS  = -ahls -mapcs-32 -o crt.o
LFLAGS  =  -Map main.map -Tdemo2106_blink_ram.cmd
CPFLAGS = -O ihex
ODFLAGS = -x --syms

all: test
```

Turn off compiler optimization by setting compiler flag to:

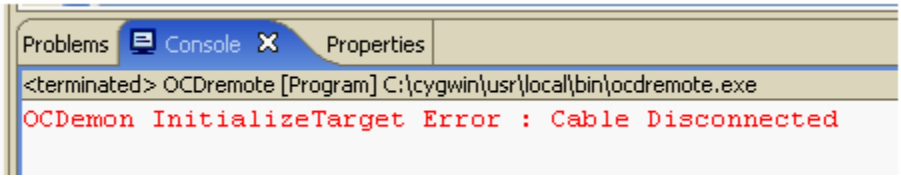
-O0 - no optimization

También debemos configurar insight para este proyecto en específico, para lo cual vamos a External Tools y en "Working Directory" escribimos la carpeta en donde se encuentra el proyecto y en "Arguments" escribimos main.out

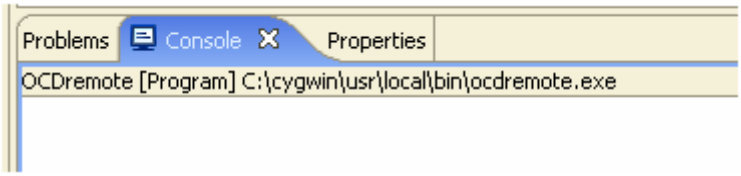


Inicializando el OCDremote

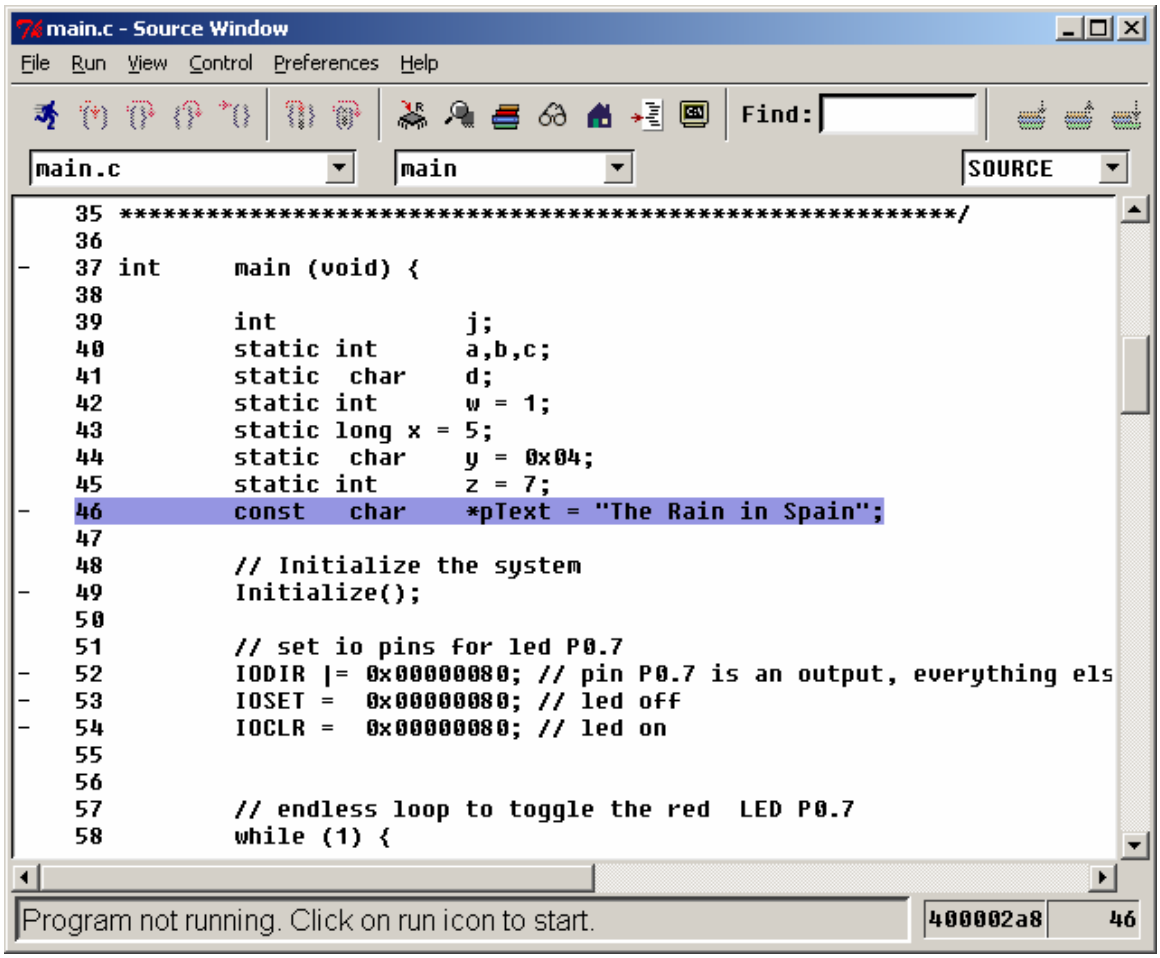
Vamos al botón “External Tools” y seleccionamos OCDremote. Hay veces en que OCD no quiere partir y nos envía este error:



Sólo nos queda inicializarlo nuevamente hasta que funcione:

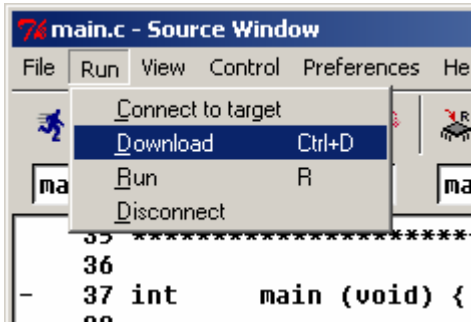


A veces no parte por que tenemos múltiples copias del programa corriendo, las cuales las podemos eliminar utilizando el task manager de Windows (ctrl-alt-del)

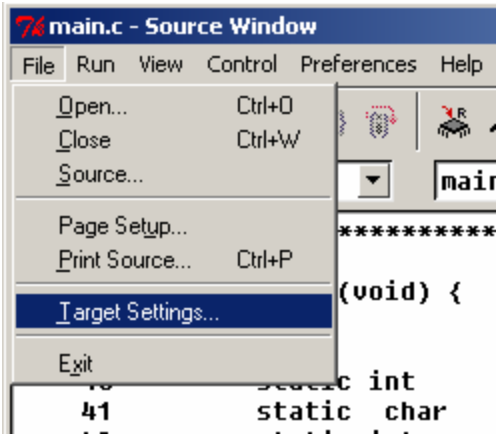


Descargando la aplicación en la RAM

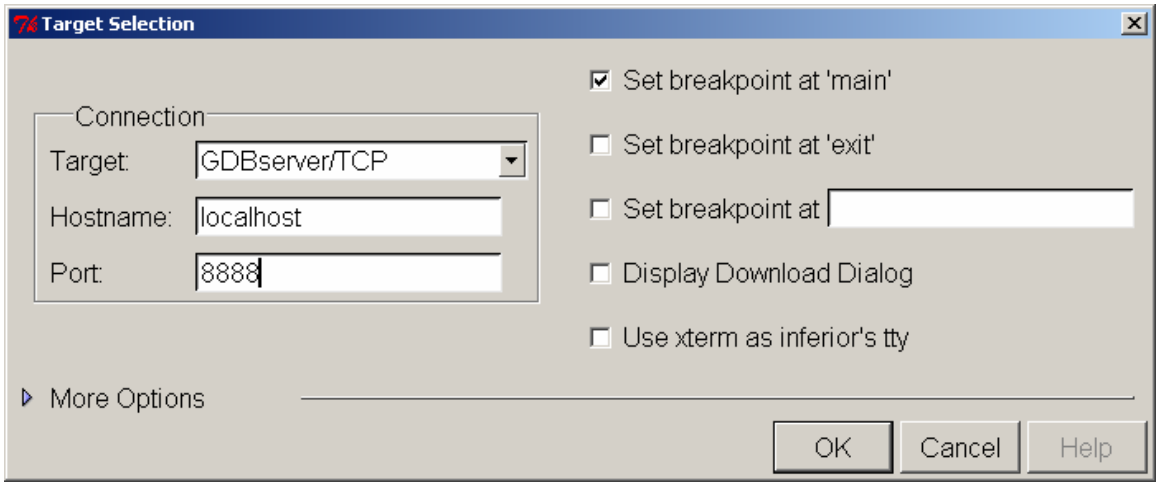
El primer paso es “descargar” la aplicación (main.out). Click en “Run->Download”



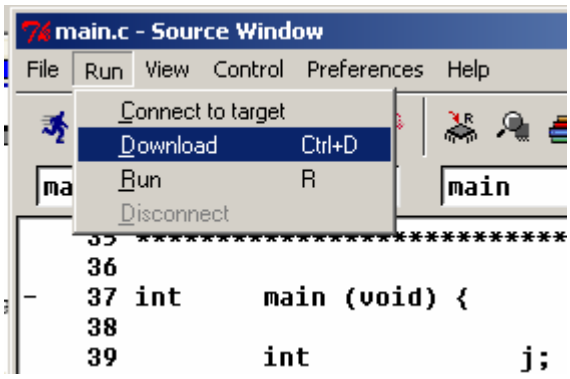
Debemos configurar el modo de comunicación de Insight con el microcontrolador , para esto entramos al menú “File->Target Settings…”



Configuramos la comunicación con el servidor GDB utilizando el puerto TCP como sigue:



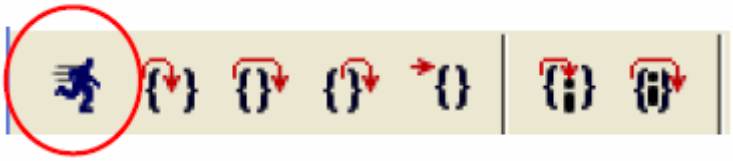
Click en “OK” para continuar. Ahora vamos al menú “Run->Download”



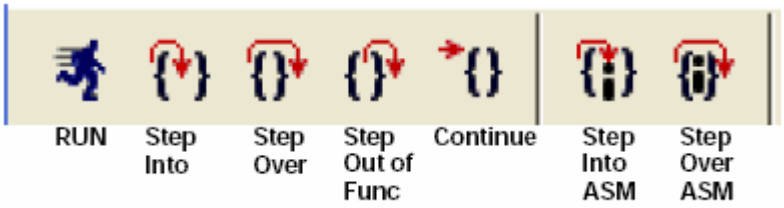
Insight se conectará a través del Wiggler. Esto cargará el programa en la memoria del LPC2106. La barra azul de progreso se moverá lentamente hasta que la descarga este finalizada.

Corriendo la aplicación

Hacemos click en el botón RUN para comenzar con la rutina main().



Insight tiene los siguientes botones asociados al debugg de la aplicación.



RUN (STOP)	Comienza a correr la aplicación Detiene la aplicación
STEP INTO	Ejecuta la siguiente línea de código C. Si la siguiente línea es una función entonces seguirá dentro “INTO” de la función
STEP OVER	Ejecuta la siguiente línea de código C. Si es una función seguirá con la línea después de la función “OVER”
STEP OUT OF FUNCTION	Sale afuera de una Función en C
CONTINUE	Continua con la ejecución del programa hasta el próximo break point.
STEP INTO ASM	Avanza una instrucción ASM, si es una subrutina entonces entrará a ella
STEP OVER ASM	Avanza una instrucción en ASM Si es una rama de una subrutina entonces saltará la subrutina y seguirá con la próxima instrucción

Palabras finales.

Este tutorial ha sido diseñado para estudiantes y hobbistas que cuentan con recursos limitados. Describe en gran detalle como descargar e instalar todos los programas necesarios para poder desarrollar una aplicación con un microcontrolador ARM. Todos los programas utilizados son completamente gratuitos.

Si eres un ingeniero profesional que esta desarrollando una aplicación con un microcontrolador ARM y estas ocupando estas herramientas mi recomendación es que debes considerar otra opción. Los compiladores profesionales como IAR, Rowel y Keil son mucho más eficientes y generalmente tienen una interfaz libre de errores. Estos permiten debuggear en la RAM y en la FLASH con un simple click. También cuentan con un soporte telefónico. Estos paquetes profesionales ahorraran tiempo y dinero en la empresa.

Algunos sitios de interés

Documentación ARM
http://www.arm.com/documentation/

Documentación de la familia LPC2000
http://www.semiconductors.philips.com/pip/LPC2106.html

Documentación GNU.
http://dsl.ee.unsw.edu.au/dsl-cdrom/gnutools/doc/

Eclipse
www.eclipse.org