Chapter12
The eCos Toolset

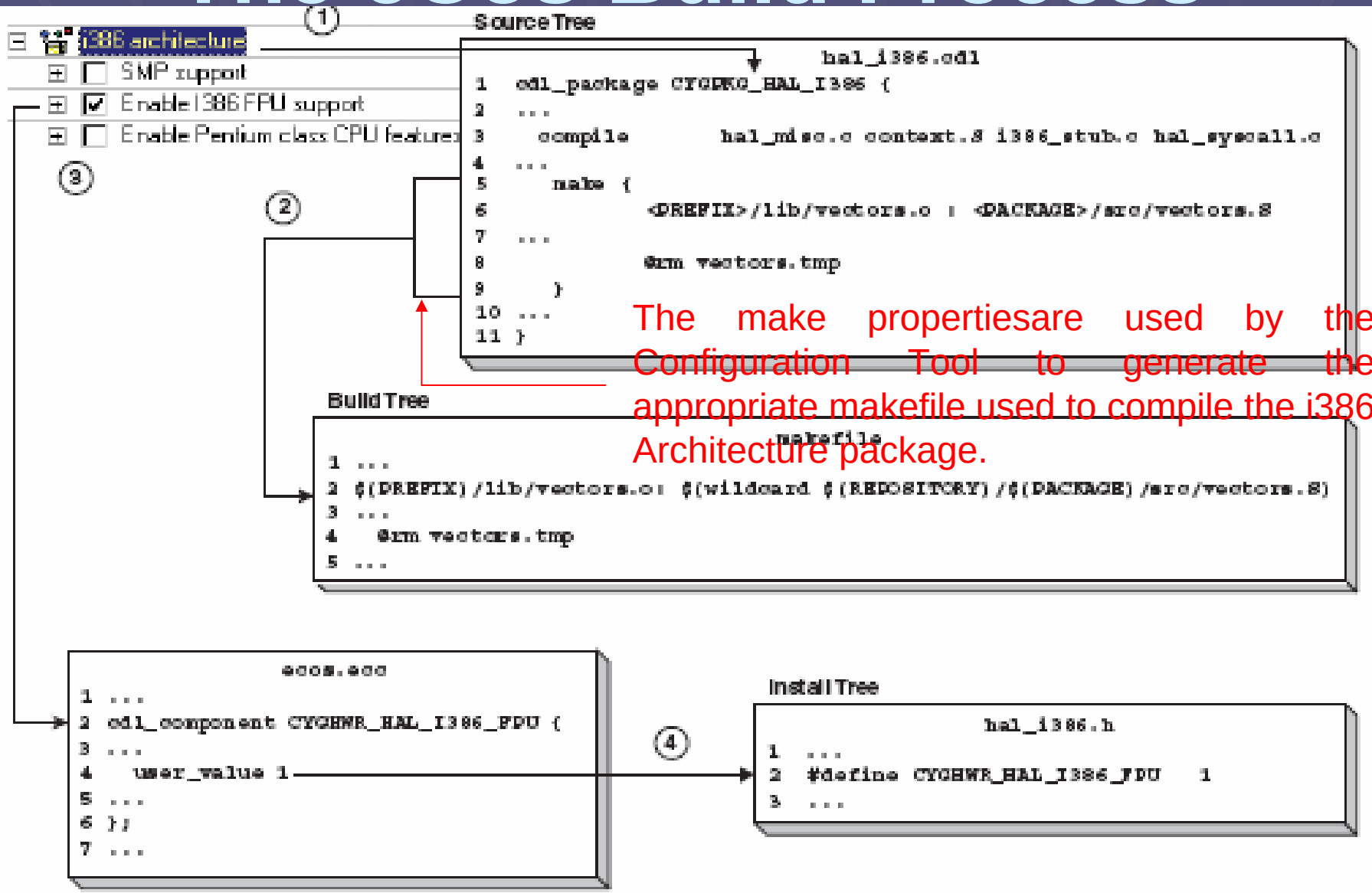
指導老師:張軒彬 老師
學生:杜建志

The eCos Build Process

- The eCos build process involves three separate trees: source, build,and install.

- The source tree is the source code repository, which is located under the packages directory.
- The build tree is generated by the configuration tools and contains intermediate files, such as makefiles and object files.
- The install tree is the location of the eCos main library file and the exported header files, which are used when the application is built.

The eCos Build Process



The make propertiesare used by the Configuration Tool to generate the appropriate makefile used to compile the i386 Architecture package.

Figure 12.1 Configuration Tool file generation diagram.

Build RedBoot

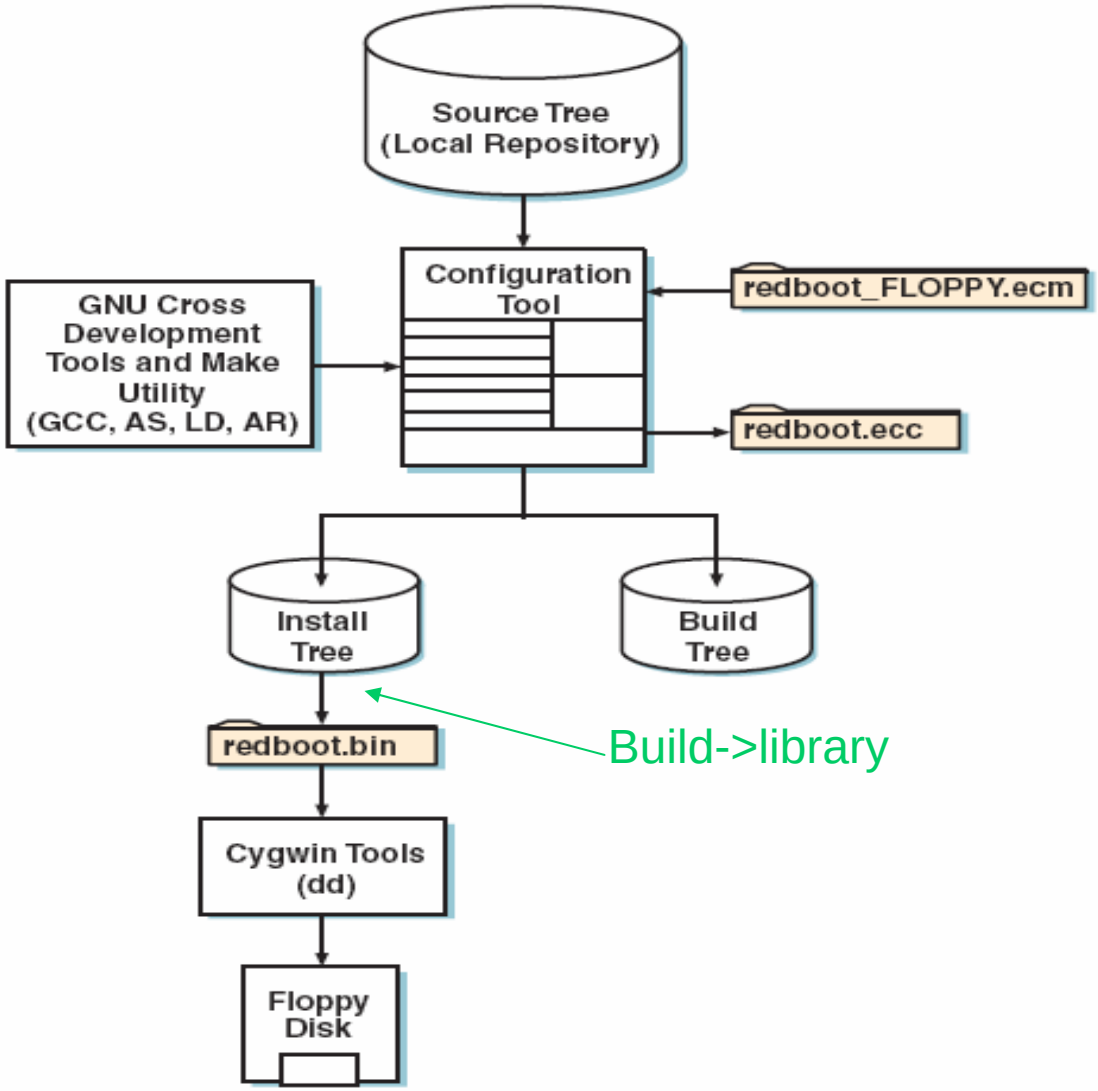
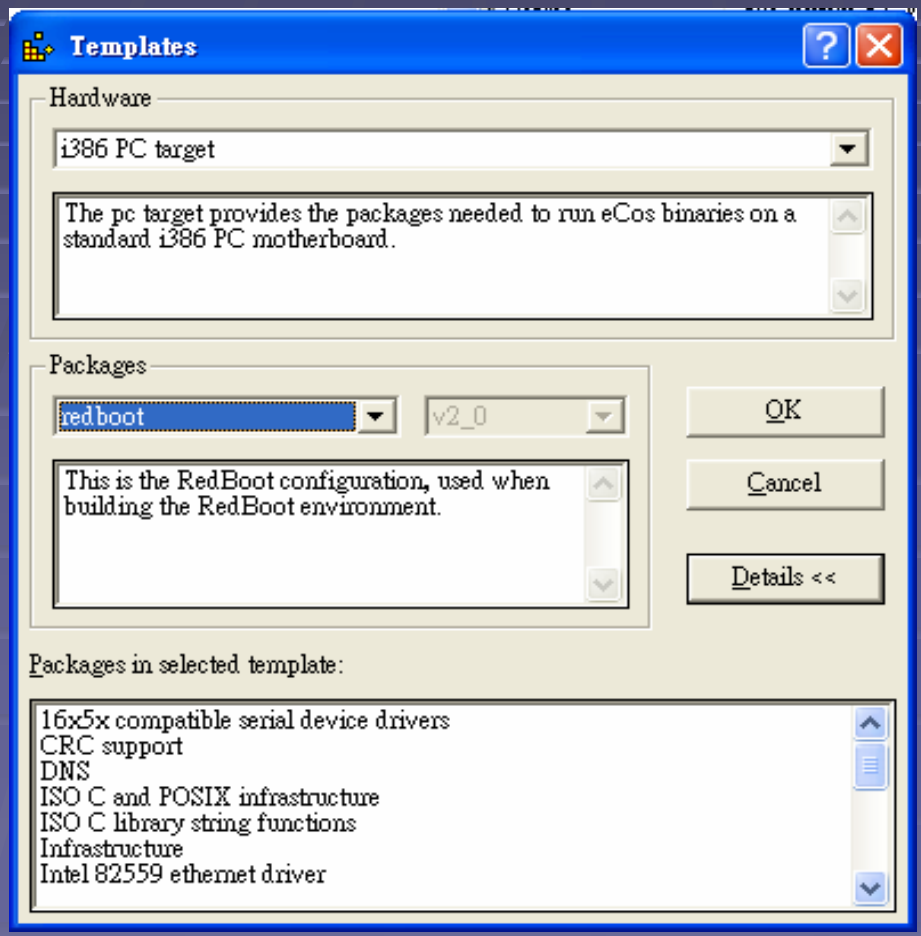


Figure 12.5 RedBoot build and install procedure flow diagram.

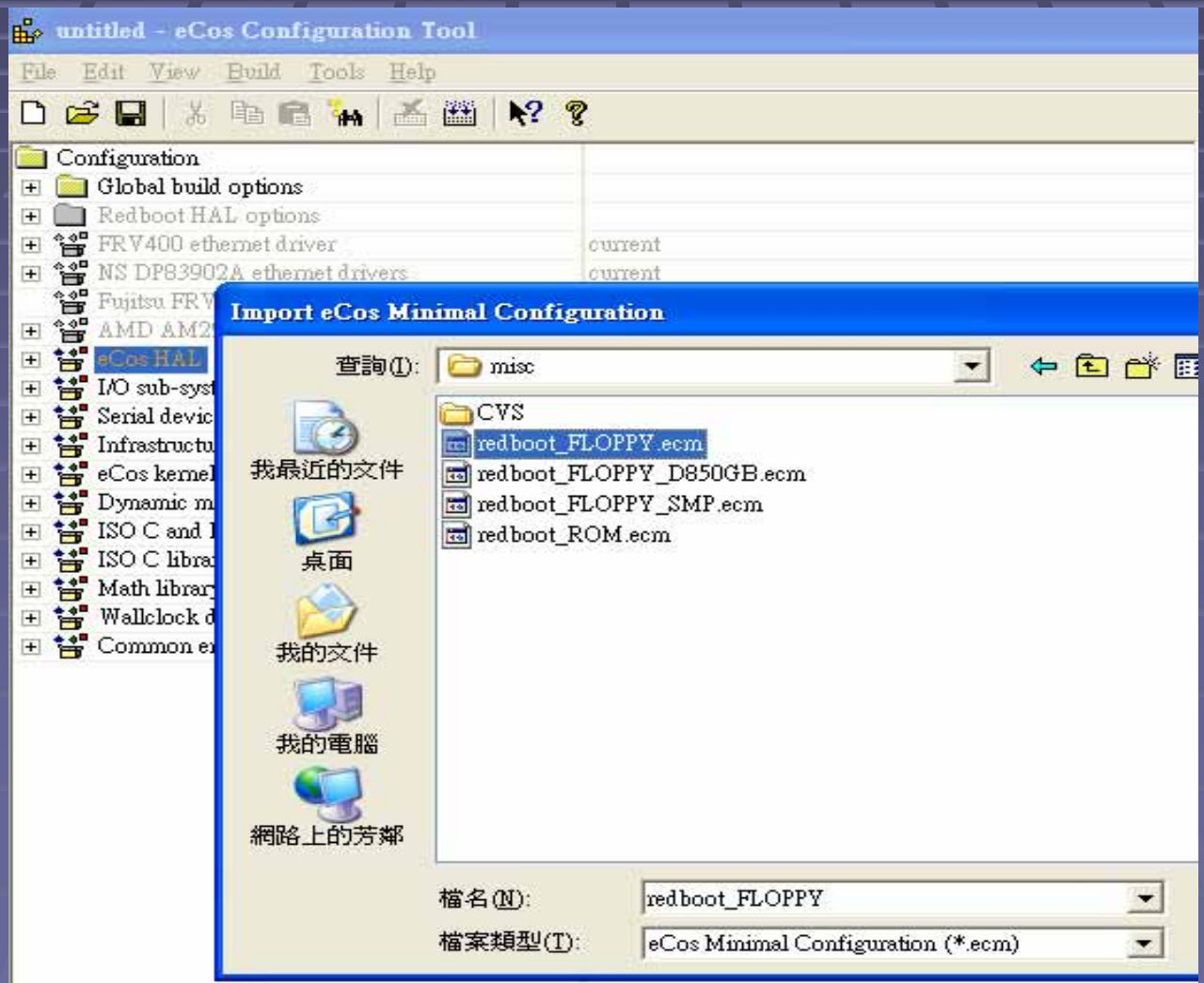
Build RedBoot

選擇template



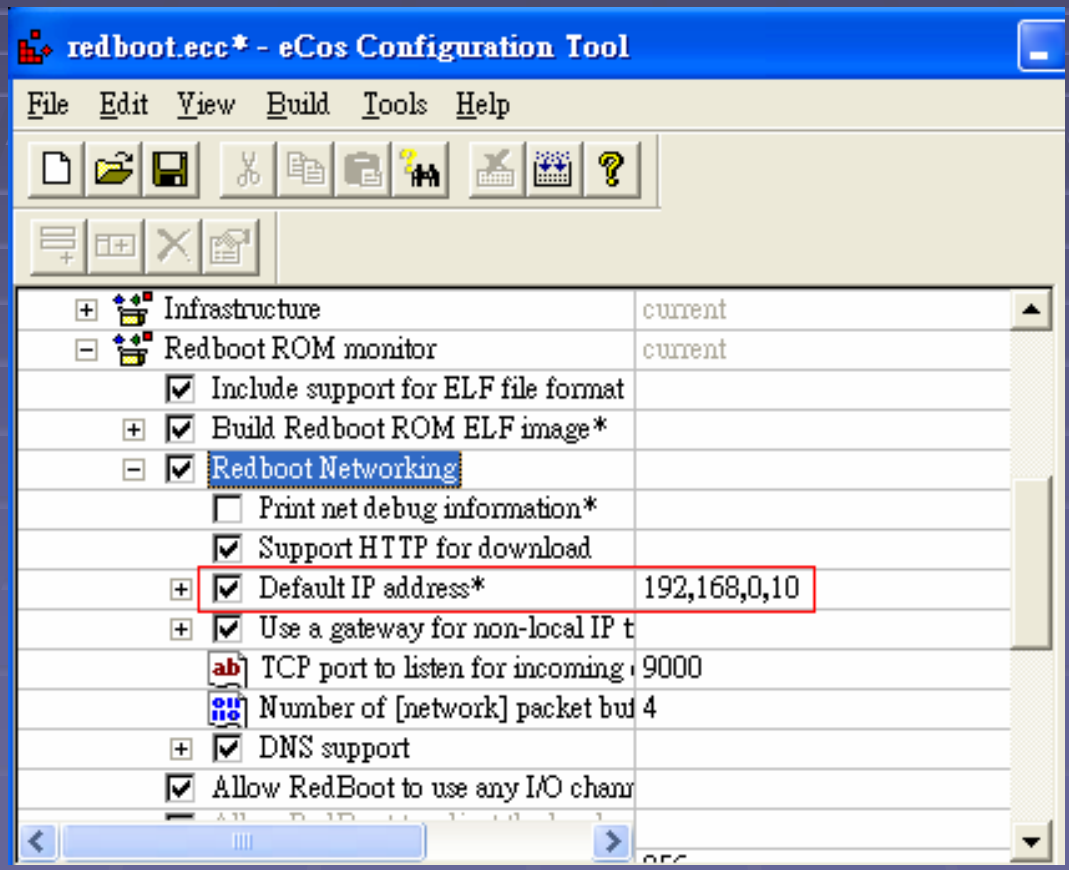
Build RedBoot

直接import minimal configuration



Build RedBoot

設定IP



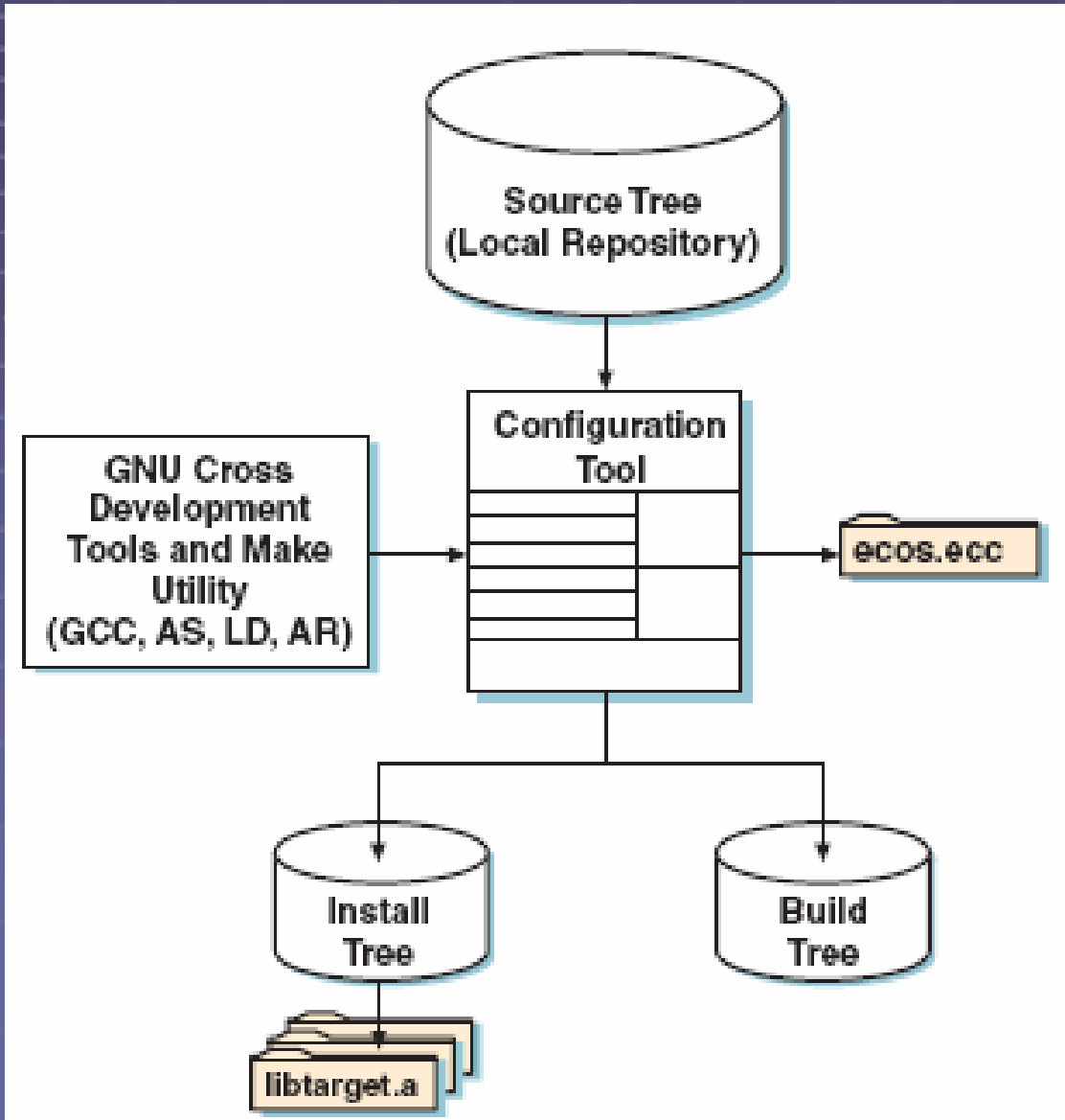
Build RedBoot

- 存檔之後會產生
- The redboot_build subdirectory contains the build tree and is used by the Configuration Tool to store the different files used and created in the build process, such as makefiles and object files.
- The redboot_install subdirectory contains the install tree and includes the final output binary images as well as the header files used for the build process.

Build RedBoot

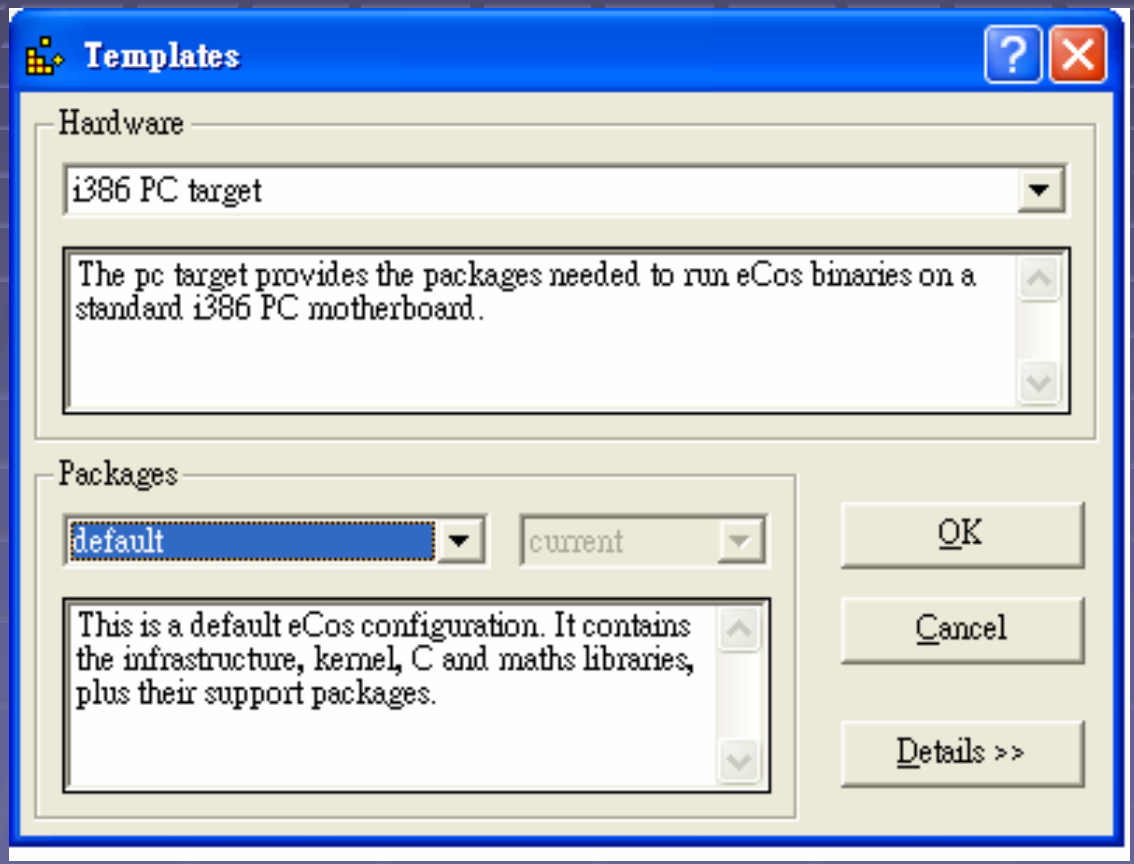
- $ mount -f -b //./a: /dev/fd0
- $ cd d:/workdir/redboot
- $ dd conv=sync if=redboot_install/bin/redboot.bin of=/dev/fd0
- RedBoot floopy disk 完成

Building eCos



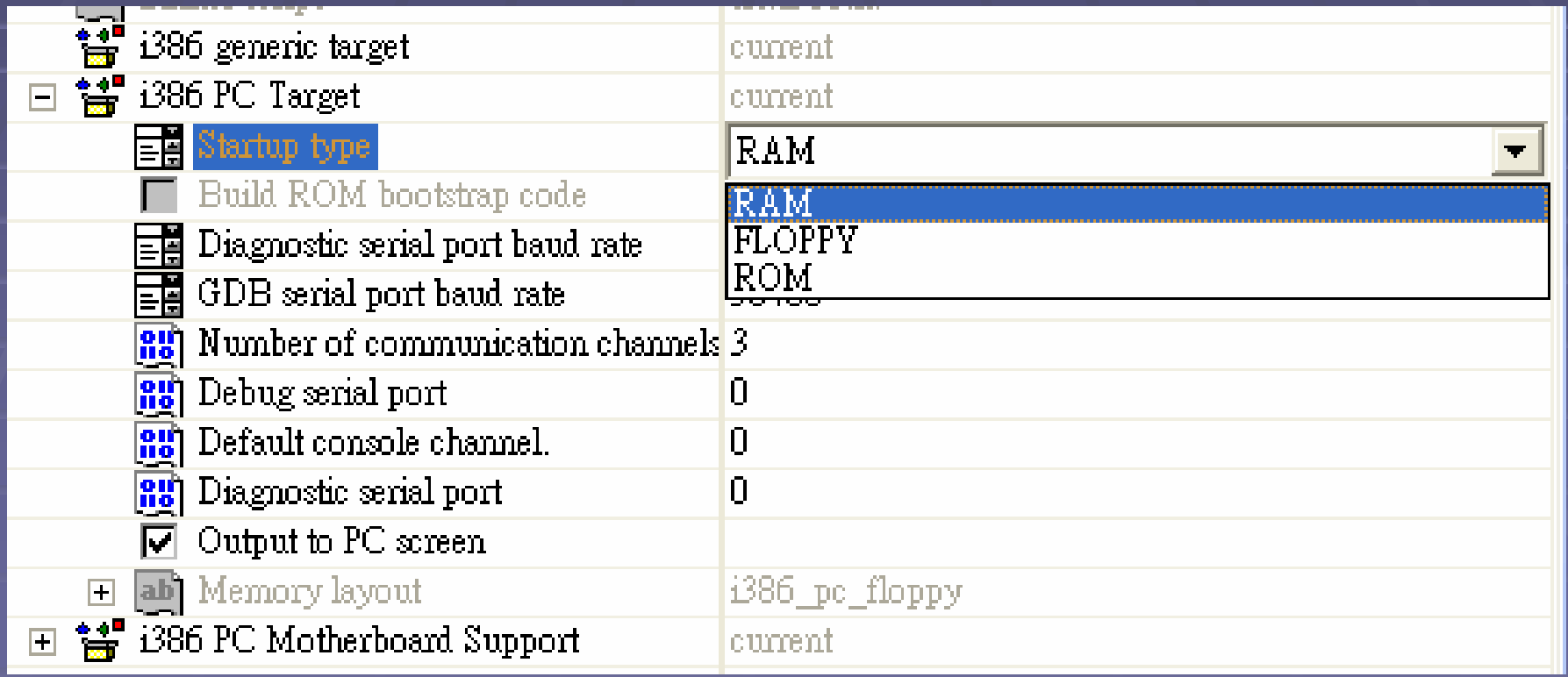
Building eCos

選擇硬體



Building eCos

We want to set the Startup Type to RAM, so that we can load our application into RAM for running and debugging.



Building the Application

There are two threads, A and B,that run in this application. Thread A uses a semaphore to signal Thread B. Both threads are passed a value that they print out when they run.

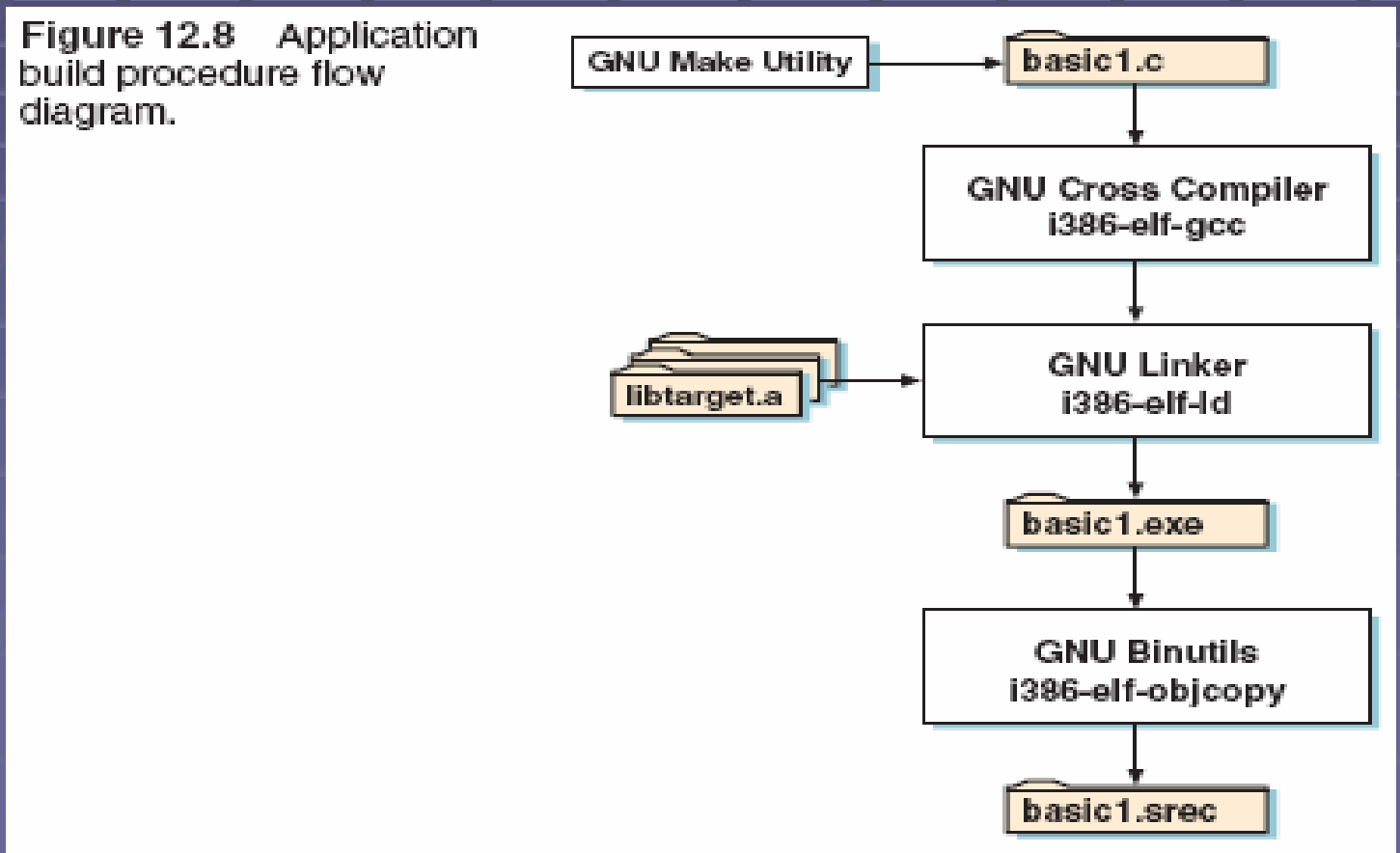


Figure 12.8 Application build procedure flow diagram.

Building the Application

- The files we copy are *basic1.c* and *makefile*.
- **$ make**
  - • **basic1.o—**the compiler output object file.
  - • **basic1.map—**the map file shows the memory layout for the image.
  - • **basic1.exe—**the ELF format executable file.
- **$ i386-elf-objcopy –O srec basic1.exe basic1.srec**
  - we change the ELF file into an S-record format to load with RedBoot.

Building the Application
mackfile

```
1 ## eCos library installation directory
2 PKG_INSTALL_DIR = /cygdrive/d/workdir/ecos/ecos_install
3
4 ## This sets the compiler to i386 PC.
5 XCC = i386-elf-gcc
6
7 ## Build flags.
8 CFLAGS = -g -Wall -I$(PKG_INSTALL_DIR)/include \
9              -ffunction-sections -fdata-sections
10 LDFLAGS = -nostartfiles -L$(PKG_INSTALL_DIR)/lib \
11                -Wl,--gc-sections -Wl,--Map -Wl,basic1.map
12 LIBS = -Ttarget.ld -nostdlib
13 LD = $(XCC)
14
```

指定XCC爲compiler

給compiler的參數

給linker的參數

the linker is built into the compiler executable, i386-elf-gcc, so we use the XCC variable name to invoke the linker.

Building the Application
mackfile

- 15 ## Build rules.
- 16 all: basic1 ← This is the main target to build.
- 17
- 18 basic1.o: basic1.c ← 用basic1.c作basic1
- 19 $(XCC) -c -o $*.o $(CFLAGS) $<
- 20
- 21 basic1: basic1.o ← Build basic1
- 22 $(LD) $(LDFLAGS) -o $@ $@.o $(LIBS)
- **Code Listing 12.3** Example application makefile.

Building the Application

- RedBoot> load -v -m yMODEM
- Target Wait……

Building the Application

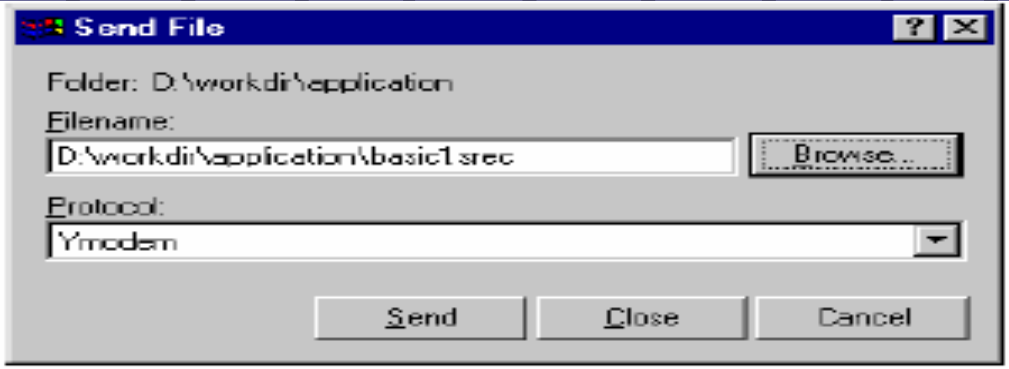


Figure 12.9 HyperTerminal Send File dialog box.

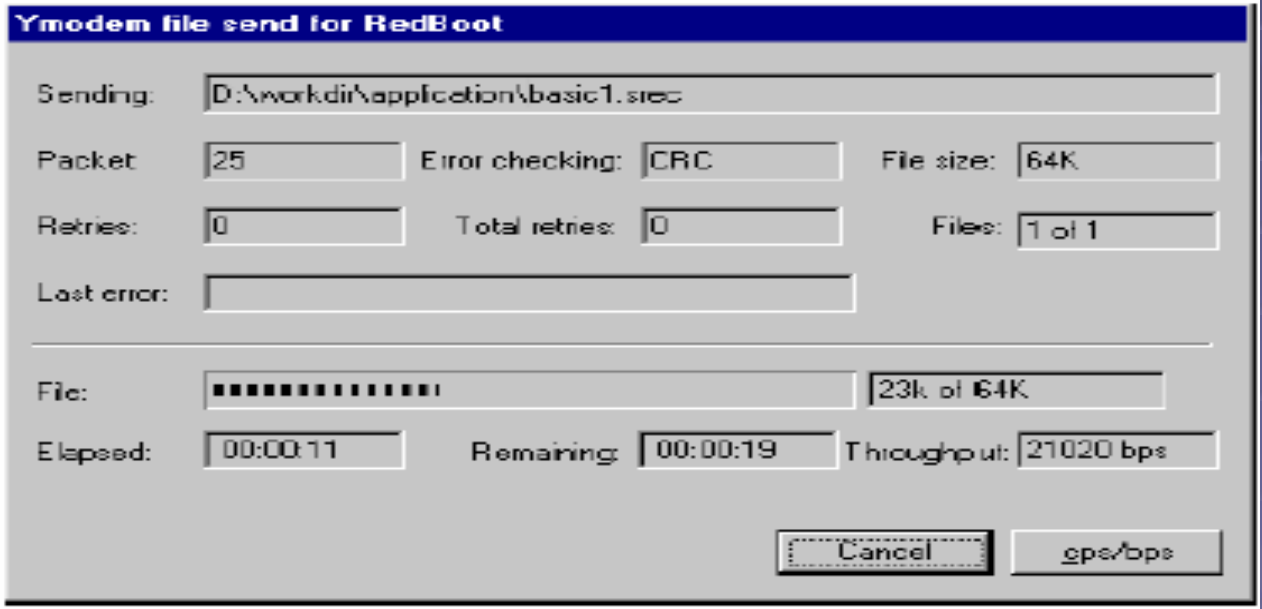


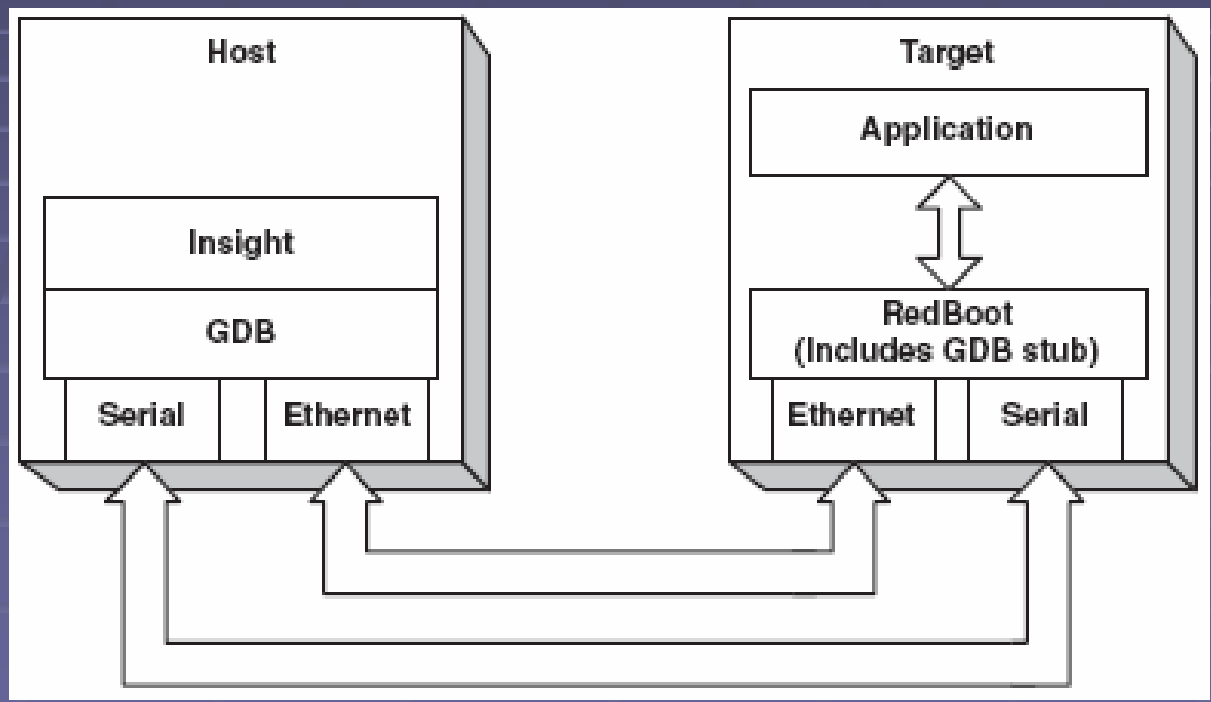
Figure 12.10 HyperTerminal Transfer Status dialog box.

Building the Application

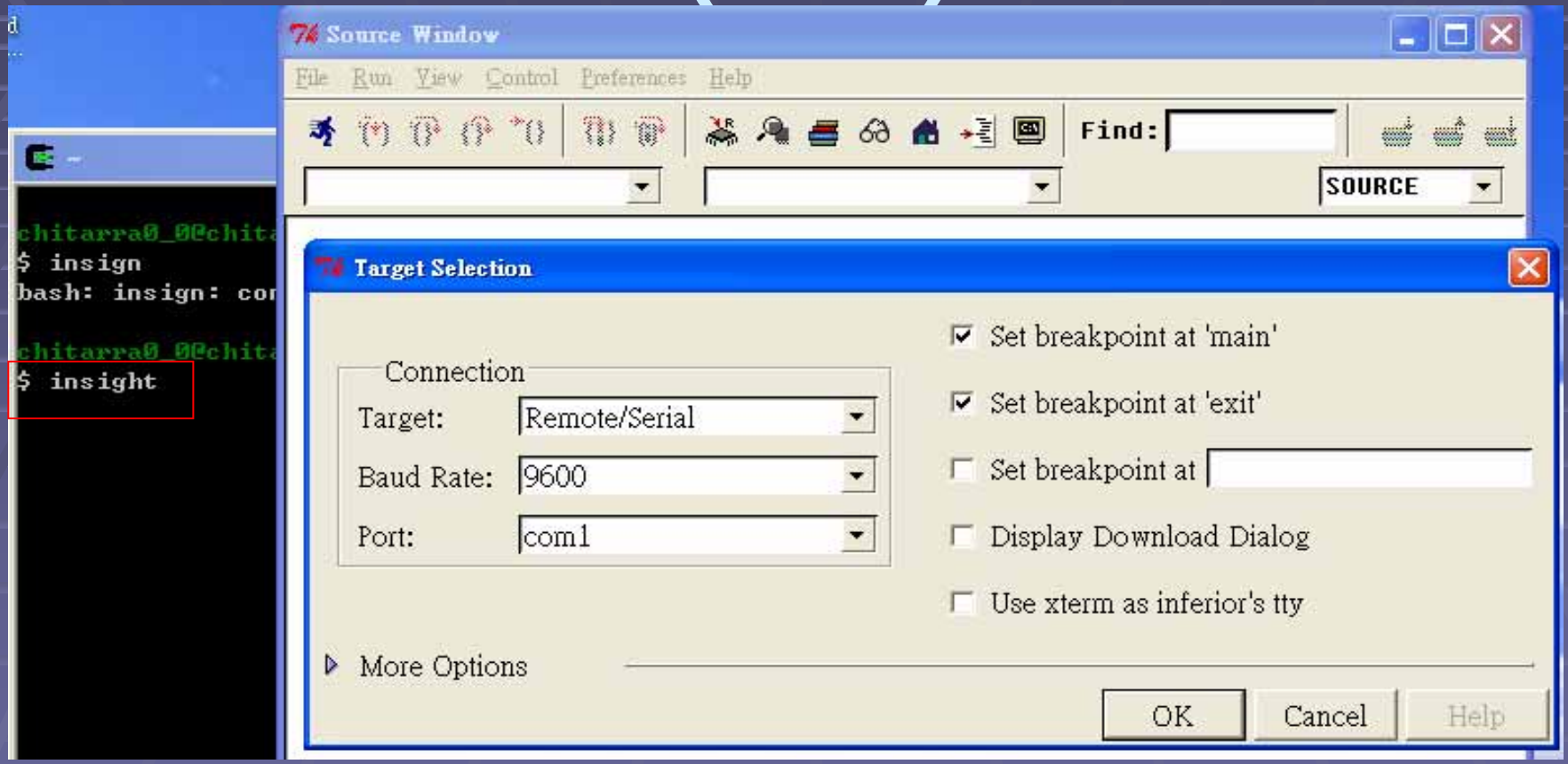
- RedBoot> go

- 1 Hello eCos World!!!
- 2
- 3 Thread A, count: 1 message: 75
- 4 Thread A, count: 2 message: 75
- 5 Thread B, message: 68
- 6 Thread A, count: 3 message: 75
- 7 Thread B, message: 68
- 8 Thread A, count: 4 message: 68
- **Code Listing 12.7** Basic1 example application output.

Building the Application (GDB)



Building the Application (GDB)



傳完後，To start the application, select *Control –> Continue*. The output from the basic1 program is displayed in the console window.

Building the Application

Table 12.2 GDB CLI Commands for Serial and Ethernet Port Debugging

Serial	Ethernet
STEP 1: Change to the application directory. `(gdb) cd d:/workdir/application`	**STEP 1:** Change to the application directory. `(gdb) cd d:/workdir/application`
STEP 2: Set the serial port baud rate. `(gdb) set remotebaud 38400`	**STEP 2:** Connect to the target via the Ethernet port.[a] `(gdb) target remote 192.168.0.10:9000`
STEP 3: Connect to the target via the serial port. `(gdb) target remote com1`	**STEP 3:** Load the application `(gdb) load basic1.exe`
STEP 4: Load the application. `(gdb) load basic1.exe`	**STEP 4:** Run the application. `(gdb) continue`
STEP 5: Run the application. `(gdb) continue`	

END

THANKS