Quick Start Tutorial

Using the TASKING* Software Development Tools with the Intel® 8x930 Family Evaluation Board

This explains how to use the TASKING Microsoft* Windows*-based software development tools with the Intel 8x930 based evaluation board. This document is divided into the following sections:

About this Tutorial

This tutorial focuses on using EDE™, TASKING's integrated Embedded Development Environment. It explains how to create and build a new project using a source code example that is included in this tutorial. It also shows how to configure the tools; assemble and link the code; and invoke the debugger.

TASKING Software Development Tools Tutorial

Before you begin this tutorial, you need the TASKING software development tools. To get it, do one of the following:

- Locate the evaluation version of the TASKING software development tools that shipped with the USB 8x930 peripheral development kit
- Download the evaluation software from the TASKING web site (**www.tasking.com**)
- Purchase a complete software package from TASKING

Starting the Tutorial

Install the TASKING software development tools by completing the following steps.

1. Insert your CD in the CD-ROM player.
2. Choose the Run option:
   - In Windows 95/NT, press the Start button and choose Run
   - In Windows 3.x, open the Program Manager or File Manager File menu and choose Run
3. Type **D:SETUP** at the Command Prompt. Where "D:" represents the drive of the CD-ROM player
4. Press OK.
5. Follow the on-screen prompts.

After you install the TASKING development tools, a new program group appears on your desktop (shown in Figure 1).

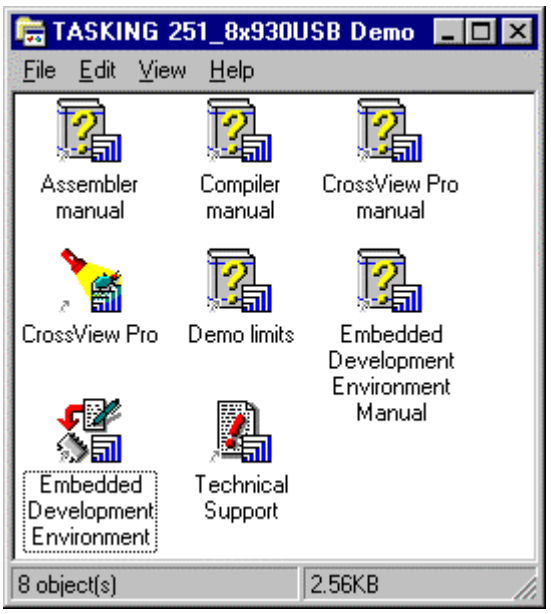

Figure 1 EDE Icons

The evaluation version of the compiler and assembler support a limited number of symbols and operands and the linker allows up to 3K of code size. CrossView Pro also has some restrictions including the About box popping up every 5 minutes. Please refer to the Demo Limits help file for more details.

6. One of the icons within the TASKING program group is for the integrated Embedded Development Environment. Double-click on the EDE icon.

The window shown in Figure 2 appears. From this window you can create projects, edit files, configure the tools, compile, assemble, link, and invoke the debugger.

> ***NOTE*** *When using Windows 95 you can create a shortcut on your desktop by dragging the EDE icon to the desktop using the right mouse button!*

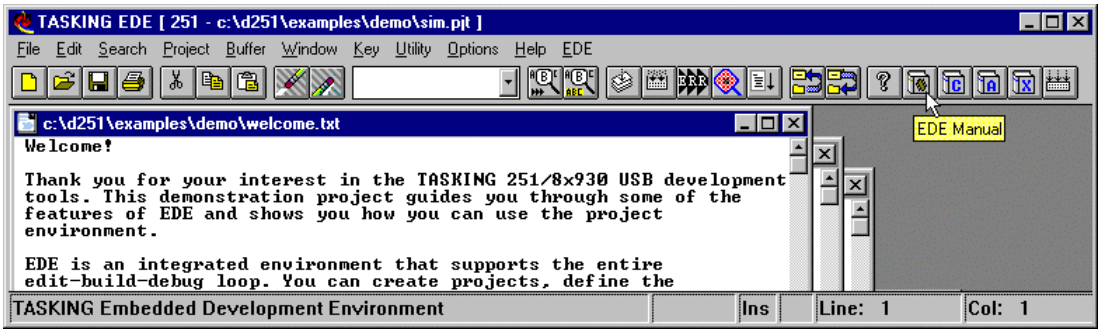

Figure 2 TASKING's EDE

NOTE *EDE is based on the popular Codewright™ for Windows editor from Premia®. The TASKING 251/USB evaluation software contains a demo version of Codewright, with an expiration date. The full 251/USB product incorporates the TASKING OEM version of Codewright. You can also use EDE with you existing Codewright configuration. See EDE manual for details.*

7. Several example project files of C applications are delivered with the product. See the TASKING Quick Start document for details. This tutorial, however, explains how to create your own assembly project for the 8x930 USB Family Board.

Creating a New Project

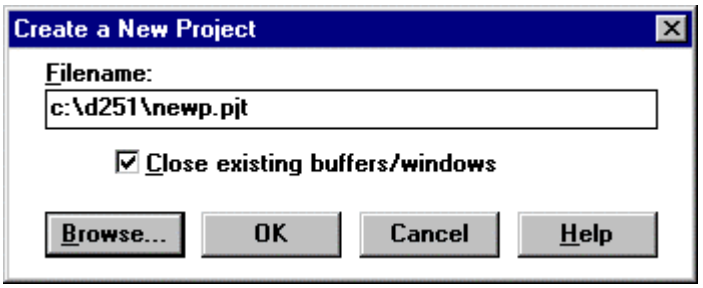

Figure 3 Create a New Project Window

1. Open the Project menu and select the New… menu item. The dialog shown in Figure 3 appears.

2. Either enter the name of the project you are creating or press the Browse… button and use the dialog shown in Figure 4.

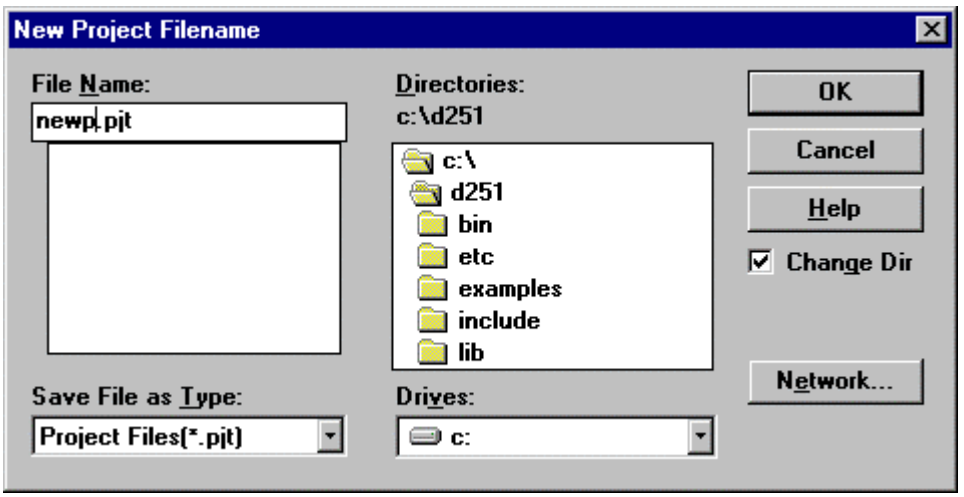

Figure 4 Browse for New Project Filename

3. For this tutorial, choose the name `newp.pjt` and press OK to close the dialog. When you close this dialog, you again see the dialog in Figure 3.

4. Press OK again to close the dialog from Figure 3.

5. When the project file has been created, the EDE project manager asks you to specify the source files of your project, using the dialog of Figure 5. Since we do not have any source files yet, we skip this dialog by pressing Cancel. As soon as we have created a source file, we will use this dialog to the source file to our project.

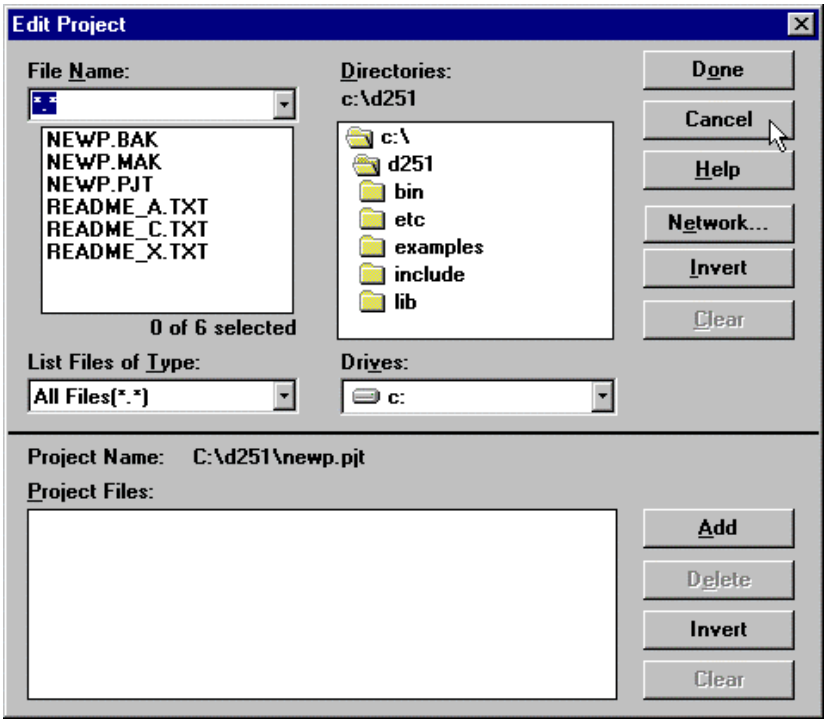

Figure 5 Edit Project

Opening a New Source Code File

1. Open the EDE File menu and choose New. The dialog shown in Figure 6 appears:

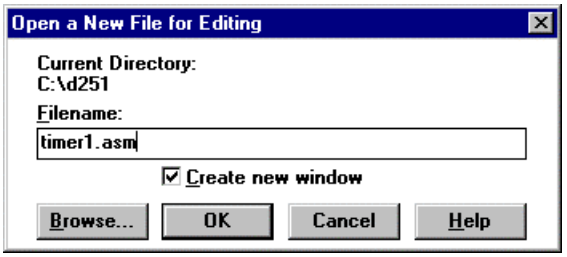

Figure 6 Open New File

2. Enter the name of the file you want to edit and press OK. This tutorial uses the name: `timer1.asm`

3. An editor window appears in the EDE window. Use the editor to type in the example program provided in Figure 7 or use a program of your own. The example program uses a number of `MOV` instructions to port 1 of the 8x930 device. The program sets up timer 1 to cause an interrupt after it overflows. This program is used in the rest of the tutorial.

```
;Use assembler predefined register names, no need to include:
;$include(\d251\include\reg930ax.inc)
   PUBLIC __START              ; Debugger "Reset application" label
   HCSEG  AT 00:4000H          ; Huge Code Segment; anywhere in 16M
   JMP    MAIN
   ORG    00:401BH
   JMP    TIMER_ISR
   ORG    00:4100H
MAIN:
__START:
   MOV    TMOD,#00010000B      ; timer 1 mode 1
   MOV    P1,#00000000B
   MOV    P1,#00000011B
   MOV    P1,#00000111B
   MOV    P1,#00001111B
   SETB   IE.3                 ; ENABLE timer1 interrupt
   SETB   TR1                  ; start timer 1
STAY:
   JMP    STAY                 ; stay here until timer1 overflow
TIMER_ISR:
   CLR    IE.3                 ; DISABLE timer1 interrupt
   MOV    P1,#00010000B
   MOV    P1,#00011000B
   MOV    P1,#00011100B
   MOV    P1,#00011110B
   MOV    P1,#00011111B
STAY1:
   JMP    STAY1                ; stay here forever
   SETB   IE.3
   RETI
END
```

Figure 7 Example Program

4. After you finish typing the code, open the editor's File menu and select Save.

Building the Project

Now that you have a project (`newp.pjt`) and a file (`timer1.asm`), build your project as follows:

1. Open the EDE Project menu and select the Edit… menu item. The dialog shown in Figure 8 appears.

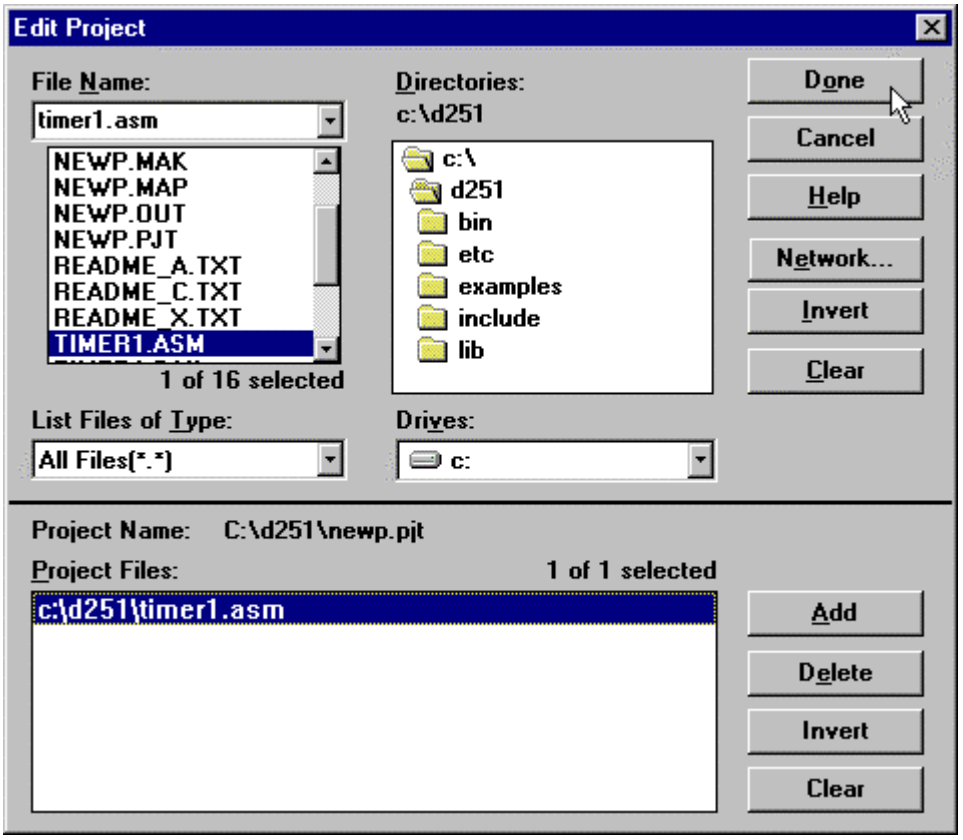

Figure 8 Adding a File

2. Select the file `timer1.asm` and choose Add (or double click the file). Press Done to close the dialog from Figure 8.

Configuring the Processor, Assembler, Linker, and Debugger

1. Configure the Processor by completing the following items:
   - Open the EDE menu and choose Processor Options...
   - Configure the options in the tabs as shown in Figure 9 and 10 and press OK. Check if the DIPswitch MODE0 of your 8x930 Family Evaluation Board reflects your choice of the Processor options: source or binary mode. Please note that the factory

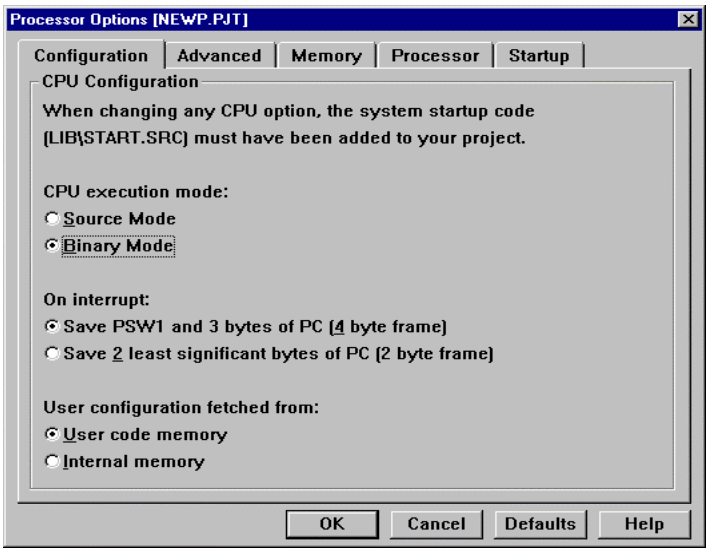

Figure 9 Processor Options - I

     setting of most boards assumes binary mode. So you either have to rebuild the application for binary mode (select Binary Mode and click OK) or set the board to run in source mode (DIPswitch MOD0 must be set to ‘on’ for Source Mode, ‘off’ for Binary Mode).
   - In the Memory tab of the Processor Options (see Figure 10) you specify how the CPU is configured with regards to internal ROM/RAM as well as the external memory interface (RD# and PSEN# functions). For the 8x930 Family Evaluation Board, the USB controller is a ROM-less part and has 1K of internal RAM. The external memory interface is configured for using 18 address lines (A0-A17), so the processor addresses 256KB as one linear space from 0-3FFFFH. In the Linker Options you can now specify the external ROM and RAM areas within the 256KB range.

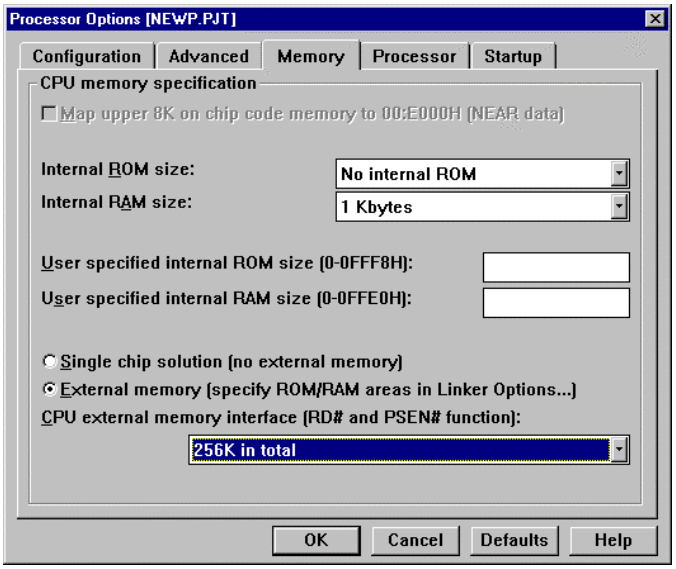

Figure 10 Processor Options - II

2. Configure the Assembler. For this tutorial you do not have to change the default assembler settings. You can always view the current assembler settings by doing the following:
   - Open the EDE menu and choose Assembler Options, Project Options…
   - Use Register Bank 0 (see the Object tab)
   - Use the assembler's predefined set of special function register names (see the Misc tab)

3. Configure the Linker and Locator by completing the following:

- Open the EDE menu and choose Linker Options... There is also a Memory tab inside the Linker Options dialog. This tab is very important, since here you specify where your external ROM and RAM areas are, which memory areas must be reserved and more. Figure 11 shows the settings for the 8x930 USB evaluation

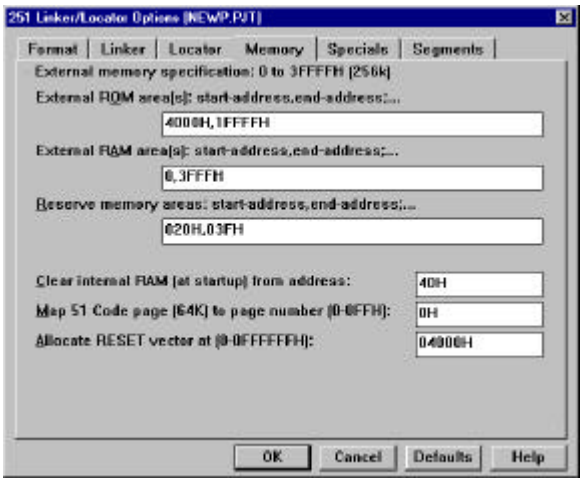

Figure 11 Linker/Locator Configuration

board.

The 8x930 USB board has 128KB of RAM for downloading application code and data. The upper 128 KB of memory is not used, except for the 32K EPROM where RISM resides. RISM expects the application and it's interrupt vectors to be located at 4000H. In this example we have specified RAM in the lower 16KB and the rest of the memory (112K) to be used for downloading user code and constant data.

The other fields of the tab contain values needed when building a C application for the 8x930 Family Evaluation Board. Since this tutorial is an assembly application with absolute segments (and not using any system startup code), you can omit these entries. However, we recommend configuring all the options as shown in Figure 11.

- Specify the output format; IEEE-695 format is used by CrossView Pro (see Format tab, not listed on this page)
- You don't have to link any C library for this assembly application (see Linker tab, not listed on this page)
- Use the standard locator control file (see Locator tab, not listed)
- Configure the options in the tabs as shown in Figure 11 and press OK.

4. Configure the Debugger by completing the following:
   - Open the EDE menu and choose CrossView Pro Options...
   - Select the ROM monitor debugger version of CrossView Pro
   - Select the COM port that is connected to the evaluation board.
   - Select the correct baud rate. In Figure 12, the baud rate 19200 is selected
     - **External serial port**: To use an external serial port, choose a baud rate of 19200.
     - **Internal serial port**: To use an internal serial port, choose a baud rate of 9600.

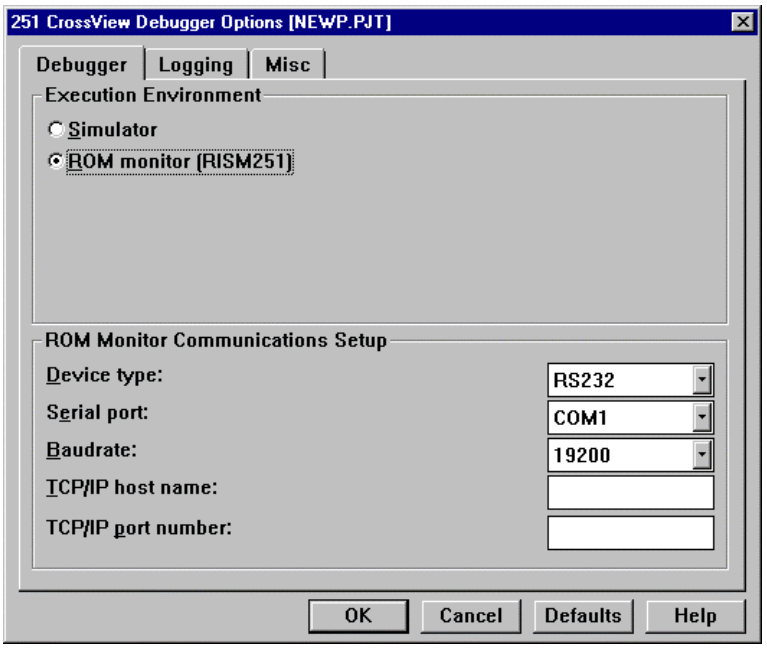

Figure 12 CrossView Pro Debugger Configuration

   - After you set the COM port and baudrate, choose OK.

Assembling the Code and Creating an Executable File

Once you have configured the tools, you can run the assembler and linker.

1. Press the Make (Build) button in the desktop toolbar of EDE (shown in Figure 13). EDE assembles, links and locates the application. To do so EDE has generated a makefile (`<project>.mak`) for this project called `newp.mak`.

Figure 13 Make Button

2. The result of the building process is captured and appears in the Output Window. Open the Window menu and choose Output.

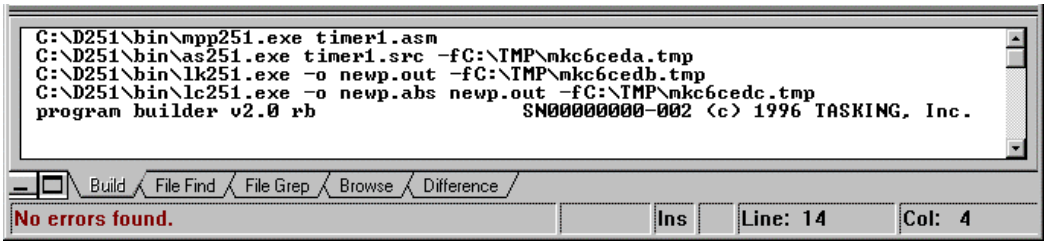

Figure 14 Output Window

3. If the program specified (`timer1.asm` or your own program) has any errors, a list of errors also appears in the Output Window. Press the ‘Goto Next Error’ button and correct the error(s) in the source code. Then, click on the ‘Make’ button again.

Figure 15 Fixing Errors

Downloading the Code to the 8x930 Evaluation Board

You are now ready to run the debugger to download code to the evaluation board.

1. Press the Debugger (CrossView Pro) button in the desktop toolbar of EDE (shown in Figure 16).

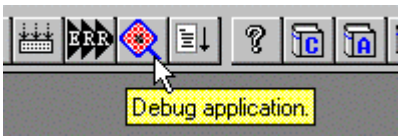

Figure 16 CrossView Pro

2. Press the OK button in the About CrossView Demo dialog. In the demo version of the product this about box will pop up every 5 minutes.
3. After a few seconds, due to the reset sequence of the board, loading of symbols and downloading the code to the board, the desktop window shown in Figure 17 appears:

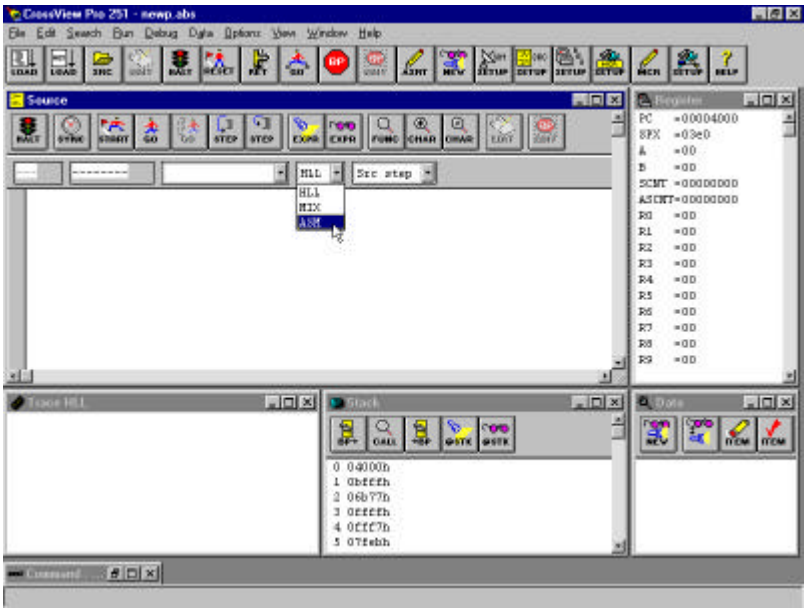

Figure 17 CrossView Pro Desktop

4. Change the view of the source window to Assembly level by clicking on ASM in the View popup list (see Figure 17). This opens the assembly source window with the Program Counter at 00:4000H.

5. Press the ‘Reset’ button in the main desktop toolbar. This resets the board and assigns the reset value of the application to the Program Counter, at 00:4100H

6. For subsequent debugging sessions, you can specify that the board must always be ‘Reset’ when loading an application. Press the ‘Load’ button in the main desktop toolbar, activate the ‘Target reset’ check box and press the ‘Setup’ button’ (see Figure

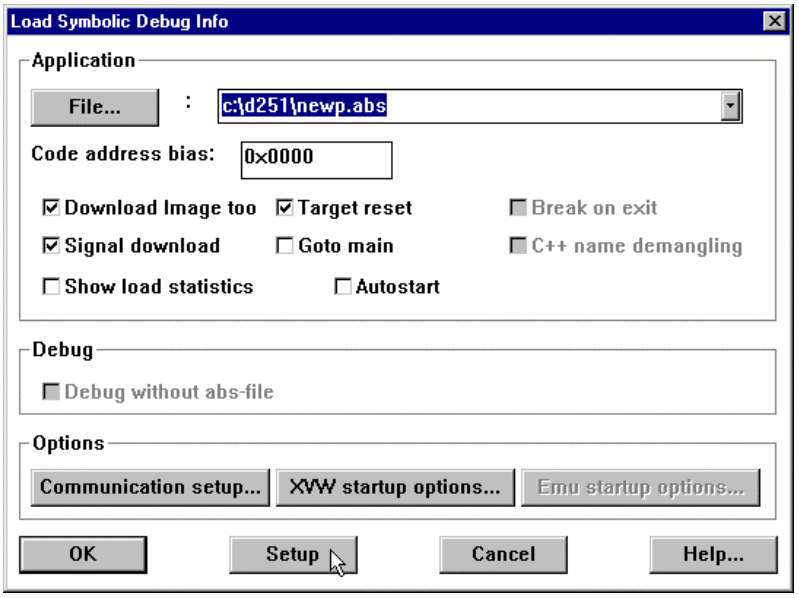

Figure 18 Load Application

18):

7. Set a breakpoint inside the Interrupt Service Routine, at 00:4117H (00:4114H when running in Source Mode), by clicking at the green icon (in the column before the address) which turns the icon red.

8. Single step a few times and inspect the result on the LED’s. Then click on the ‘Go’ button and watch the breakpoint being hit. You should now see the window of Figure 19.

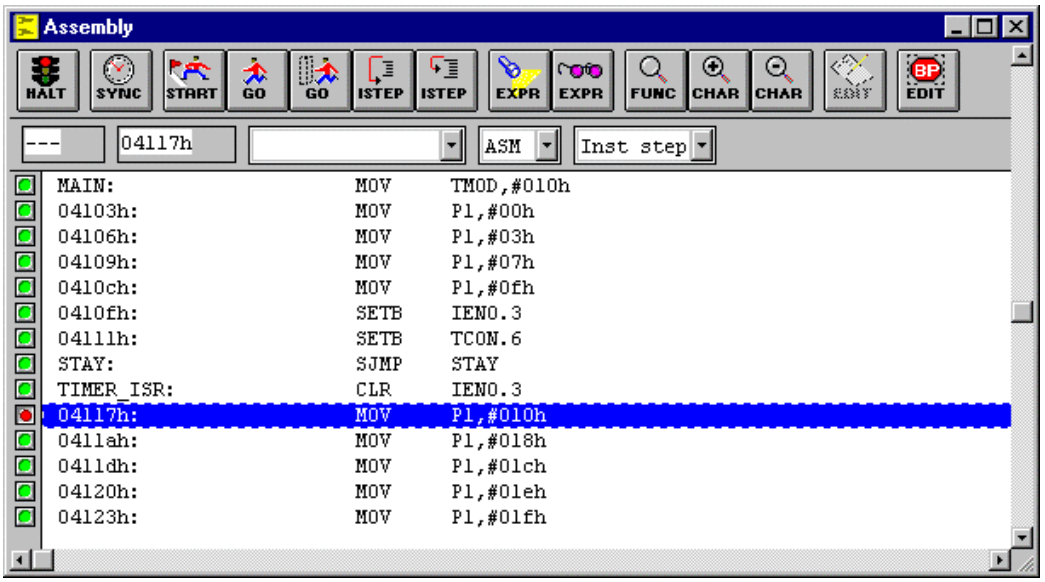

Figure 19 Assembly Source Window

You can now use the debugger window to step through code, set breakpoints, and issue the `Go` command to start program execution. You can also examine special function registers, memory locations, and register values etc.

For additional information on TASKING software development tools, visit TASKING's web site at:

`http://www.tasking.com`