

INTRODUCTION

This manual will guide you through the first steps of getting the SE-8051ICD running with the Crossware 8051 Development Suite and the Atmel Flexible In-System Programming system (FLIP).

The Flash Microcontroller Starter Kit consists of:

- the SE-8051ICD board fitted with an Atmel T89C51RD2 flash microcontroller
- a header cable that allows the SE-8051ICD to be used as an in-circuit debugger
- a serial cable to connect a PC to the SE-8051ICD
- a power supply and mains cable
- a 4k byte evaluation version of the Crossware 8051 Development Suite which includes an ANSI C compiler, Assembler, Simulator/Virtual Workshop, Debugger, Code Creation Wizards, Embedded Development Studio IDE and on-line manuals
- Atmel's Flexible In-System Programming system (FLIP)
- this Getting Started guide.

INSTALLING THE SOFTWARE ON YOUR PC

The Crossware 8051 Development Suite and Atmel's FLIP are provided on CD.

To install the 8051 Development Suite run the SETUP.EXE program that is located in the root directory of the CD and follow the on-screen instructions. An icon labeled Crossware Tools Demonstrator will be placed on your Desktop and you can use this to start the Crossware Embedded Development Studio or alternatively you can select Crossware Tools Demonstrator from the Programs menu.

To install FLIP, run the SETUP.EXE program that is located in the flip directory of the CD. You can start FLIP by selecting Atmel from the Program menu followed by FLIP 1.6.0 followed by FLIP 1.6.0. There is no need to install the USB drivers to work with the the 8051 Starter Kit.

Note that FLIP is an Atmel program supplied by Crossware with Atmel's permission.

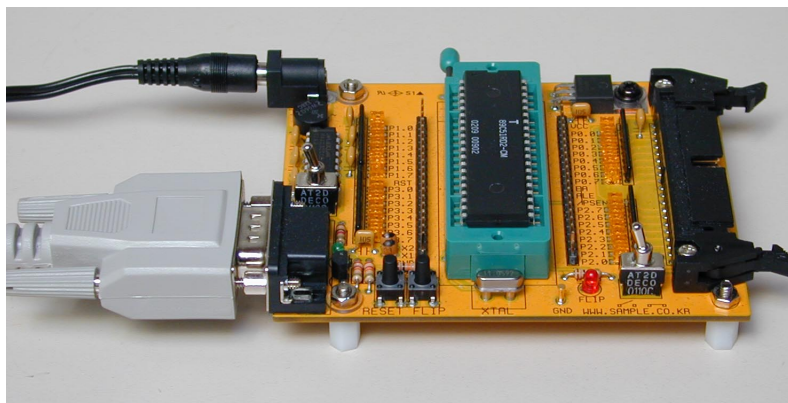
LIMITATIONS OF THE EVALUATION SOFTWARE

Programs produced in IEEE 695 format will be located in any address range that you select and all memory models are supported. Full debug records can be included to support comprehensive source level simulation and on-chip debugging. This is the best format to select for the development process. Only the Crossware evaluation environment which produced the program will be able to interpret the IEEE 695 file.

To download a stand-alone program to a target board it must be produced in a different format such as Intel hex. For these other formats, only small and large memory models can be used and the program code will be relocated to the address range 4000-4FFF hex. The evaluation software cannot therefore be used to produce stand-alone programs for target boards with less than approximately 20k bytes of program memory.

SETTING UP THE SE-8051ICD TARGET BOARD

Connect the serial cable to the 9-pin 'D' connector on the target board. Connect the power supply to the target board. Make sure that the switch next to the 9-pin 'D' connector is switched to DEBUG and that the switch next to the FLIP LED is open.



Connect the mains cable to the power supply.



Connect the other end of the serial cable to a serial port on your PC.

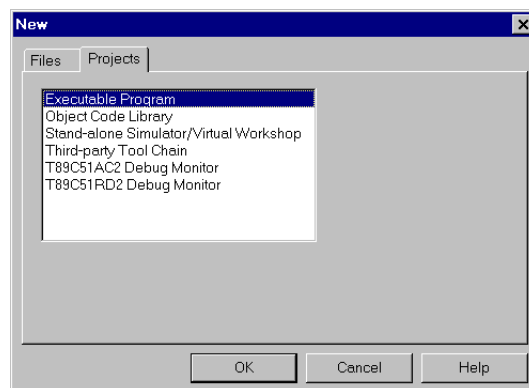
Connect the mains plug to your mains supply and switch on. The green LED will illuminate indicating that the target board is running.

A circuit diagram for the target board is included at the end of this booklet.

CREATING A TEST PROGRAM

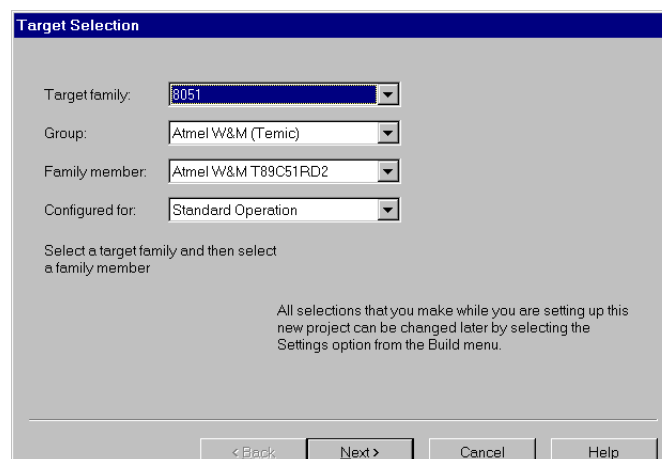
This section take you through the steps required to create a simple test program that will be downloaded to the target board using FLIP. This program will toggle the port pins so that you can observe the yellow LEDs on the board switching on and off.

Start the Embedded Development Studio. Select New from the File menu (**File→New**). Select the **Projects** tab (if it is not already active). Ensure that **Executable Program** is selected as the project type and click on **OK**.



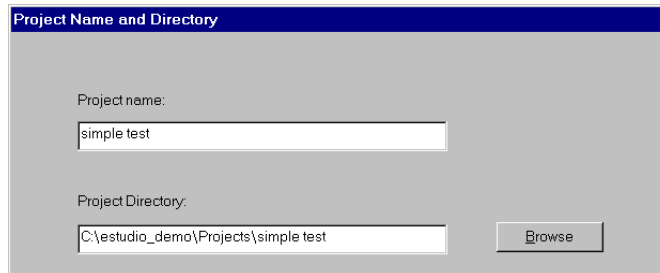
The New Project Wizard will take you through the steps of setting up a project to create an executable program.

Firstly the **Target Selection** page will be displayed:



Select **8051** as the **Target Family**, select **Atmel W&M (Temic)** as the **Group**, then select **Atmel W&M T89C51RD2** as the **Family Member**. Leave the **Configured for** item as **Standard Operation**.

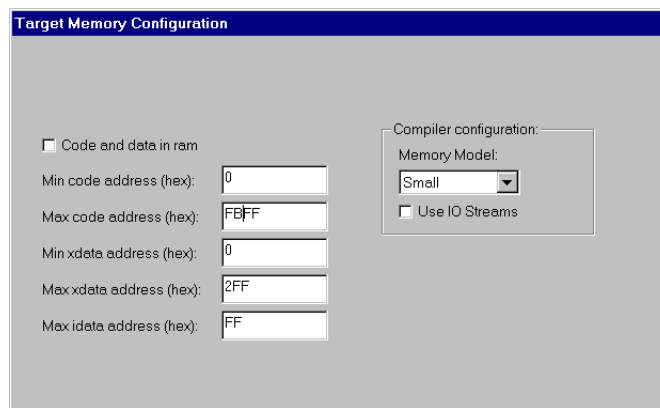
Click on **Next** and the **Project Name and Directory** page will open:



The screenshot shows a dialog box titled "Project Name and Directory". It contains two text input fields. The first field, labeled "Project name:", contains the text "simple test". The second field, labeled "Project Directory:", contains the path "C:\estudio_demo\Projects\simple test". To the right of the second field is a "Browse" button.

Enter a project name into the **Project name** field. In the above example we have entered the project name **simple test**. The **Project Directory** field has automatically been set to C:\estudio_demo\Projects\simple test but you can change this if you wish.

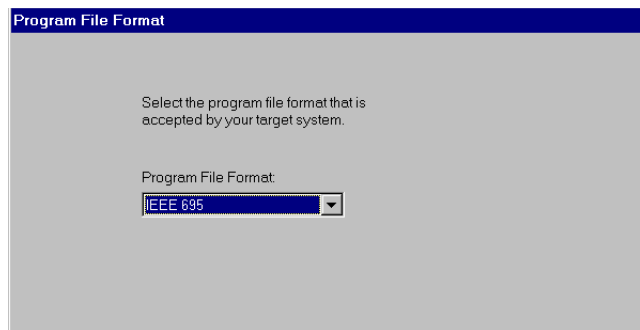
Click on **Next** and the **Target Memory Configuration** page will open:



The screenshot shows a dialog box titled "Target Memory Configuration". It has several sections. On the left, there is a checkbox labeled "Code and data in ram" which is unchecked. Below it are five text input fields for memory addresses in hexadecimal: "Min code address (hex):" with "0", "Max code address (hex):" with "FBFF", "Min xdata address (hex):" with "0", "Max xdata address (hex):" with "2FF", and "Max idata address (hex):" with "FF". On the right, there is a "Compiler configuration:" section containing a "Memory Model:" dropdown menu set to "Small" and an unchecked checkbox labeled "Use IO Streams".

Leave these items unchanged.

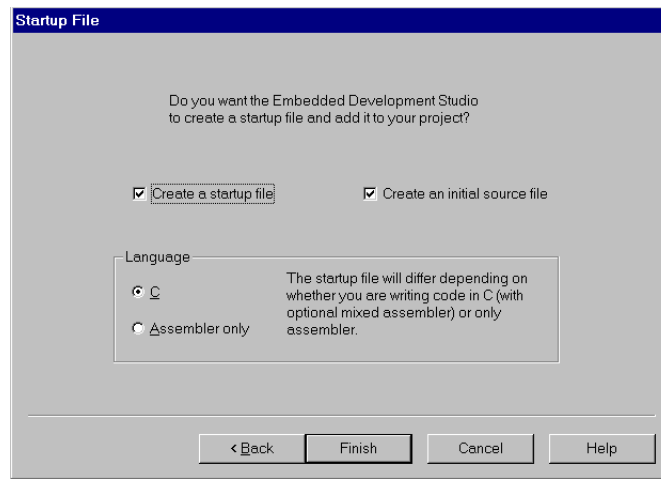
Click on **Next** and the **Additional Libraries** page will open. There are no additional libraries and so click on **Next** and the **Program File Format** page will open:



The screenshot shows a dialog box titled "Program File Format". It contains a text instruction: "Select the program file format that is accepted by your target system." Below this is a dropdown menu labeled "Program File Format" which is currently set to "IEEE 695".

Select **IEEE 695** as the **Program File Format**.

Click on **Next** and the **Startup File** page will open:



Leave these items unchanged.

Click on **Finish** and your project and an initial set of files will be created.

Source file main.c will be opened and displayed ready for editing. Add the line shown in bold (below) to this initial source file:

```
// 8051 Initial C Source File
#include "xstdsys.h"

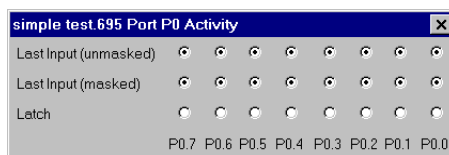
main()
{
    // use the Wizards (see Wizards menu)
    // to configure the microcontroller
    while (1)
    {
        long i;
        for (i = 0; i < 1000; i++);
        _p0 = 0X00;
        for (i = 0; i < 1000; i++);
        _p0 = 0XFF;
    }
}
```

Select **Rebuild All** from the **Build** menu (**Build→Rebuild All**) to compile and link your program. If any errors occur they will be due to typing errors in the lines that you have added. Correct these and rebuild again. Once your program has compiled and linked successfully, you can run it immediately in the simulator.

RUNNING YOUR PROGRAM IN THE SIMULATOR

To run your program in the simulator, select **Step Into** from the **Simulate** menu. (If the Simulate menu is not visible, select **Use Simulation** from the **Debug** menu.) The file startup.asm will be opened and the cursor will be placed in this view on the line after the label **__START**. Select **Step Into** again (or press the F8 key) to step to the next source line. Continue single stepping until you reach the first **for** loop in main.c.

Now select **Port 0** from the **View** menu. The Port 0 activity view will open displaying the activity on the simulating port pins:



Now select **Go** from the **Simulate** menu. Your program will run continuously and you will see the latches for port pins P0.0 to P0.7 toggling in the activity view.

Select **Close Simulator** from the **Simulate** menu to close the simulator.

Next you will run your program on the target board. To download your program to the target board you need to re-link it in

Intel hex. format and use FLIP to perform the download operation.

GENERATING INTEL HEX OUTPUT

To change the output format to Intel Hex, select **Settings** from the **Build** menu and click on the **Linker** tab. Select **Intel hex** in the **Output Format** drop down list. Note the path and file name in the **Output file name** field as you will need to know this to load your program into FLIP. Click on **OK**. Select **Rebuild All** from the **Build** menu to rebuild your program in Intel hex. format.

DOWNLOADING WITH FLIP

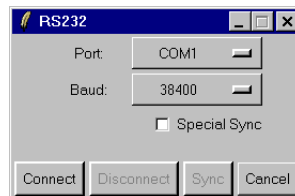
The T89C51RD2 is factory programmed with a boot loader that is located at in the program address range FC00 to FFFF hex. This boot loader will be activated if you hold down the switch labeled FLIP and press and release the RESET switch. (When the FLIP switch is closed the red LED will be illuminated.) When the boot loader is running, FLIP can be used to communicate with the target board via the serial cable and download programs into the flash memory of the T89C51RD2.

Start FLIP by selecting **Atmel**→**FLIP1.6.0**→**FLIP1.6.0** from the **Programs** menu. Now from within FLIP, select **Select** from the **Devices** menu and select **T89C51RD2** from the list of devices.

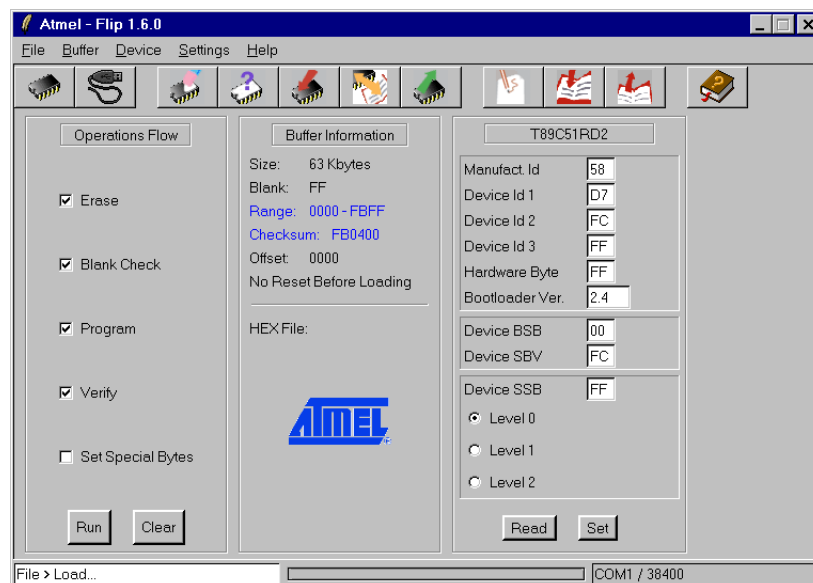


Click on **OK**.

From the **Settings** menu, select **Communications** followed by **RS232**. Select the port to which the serial cable is connected to your PC.



On the target board, hold down the FLIP button (the red LED will light up), press and release the RESET button. Release the FLIP button (the red LED will go out). In FLIP's RS232 window, click on **Connect**. FLIP will interrogate the hardware and display the software copies of the chips hardware registers.



Select **Load Hex** from the **File** menu and use the **Open File** dialog box to select the Intel hex. file that you have just built and

click on **Open**. Your program will be loaded into FLIPs buffer.

Ensure that **Erase, Blank Check, Program** and **Verify** are all checked and click on **Run**. Your program will be written into flash memory on the T89C51RD2. FLIPs status bar will display the progress of this operation.

RUNNING YOUR PROGRAM STAND-ALONE

To run the program that you have just programmed into flash, simply press the RESET button on the target board. You will see the yellow LEDs for port pins P0.0 to P0.7 flashing as your program runs.

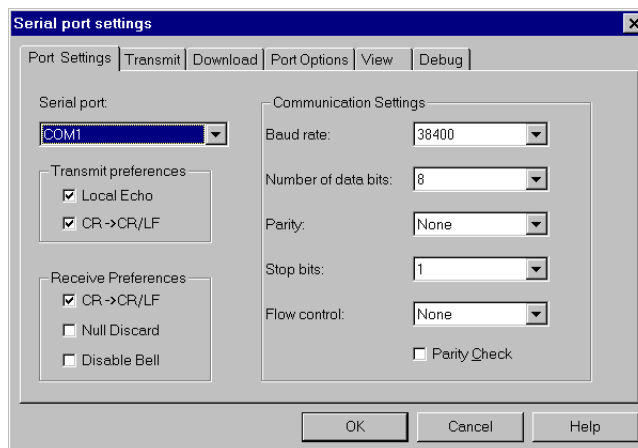
That is all you can see. You cannot see what your program is doing and what register it is setting. You cannot start and stop it except by pressing the RESET button. To observe the operation of your program as it runs on the target board, you need to run it with the debug monitor. The next step then is to build and download the debug monitor.

BUILDING THE DEBUG MONITOR

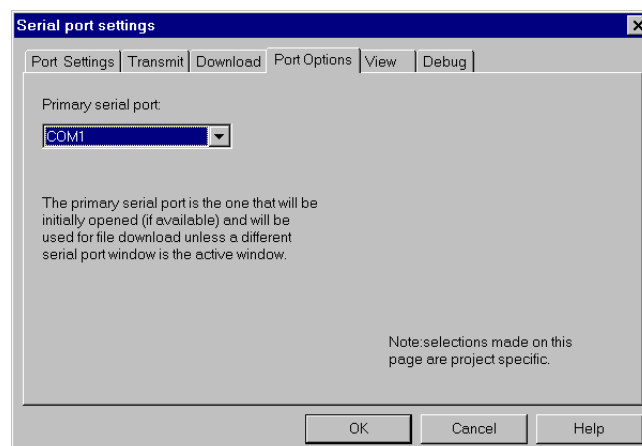
In the Crossware Embedded Development Studio, select **File→New→Projects** and select **T89C51RD2 Debug Monitor** from the list of project types. Click on **OK** and then enter for example **RD2 DM** as the project name. Click on **Finish** to create the new project.

Select **UART** from the **Wizards** menu. You will see that the UART is configured as an 8-bit UART with a baud rate of 38400. Click **Cancel** to leave this unchanged. The Crossware environment needs to be setup to work with this UART configuration so that it will communicate with the debug monitor correctly.

Select **Settings** from the **Comms** menu and in the **Port Settings** tab, select in the **Serial Port** drop down list, the PC serial port that you have used to connect to the target board. Then select 38400 as the baud rate, 8 as the number of data bits, None as parity, 1 as the number of stop bits and None as flow control:

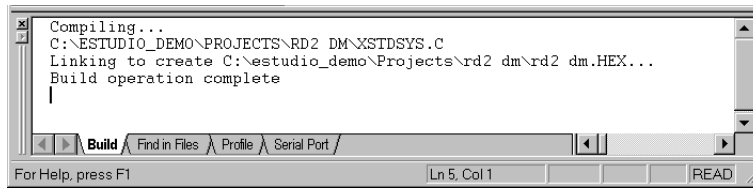


Click on the **Port Options** tab and select as the **Primary Serial Port** the PC serial port that you have used to connect to the target board.



Click on **OK**.

Select **Rebuild All** from the **Build** menu to build the debug monitor. Note the path and file name of the debug monitor displayed after the words **Linking to create** in the Build view.

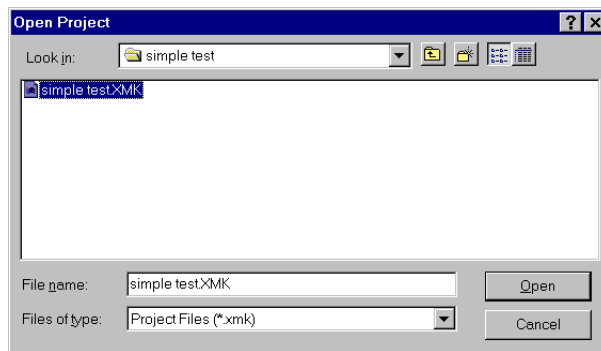


From FLIP, select **Load Hex** from the **File** menu and load the debug monitor into FLIP. On the target board, hold down the FLIP button while you press and release the RESET button and then release the FLIP button. Click on **Run** and the debug monitor will be programmed into the T89C51RD2 in place of your previous program.

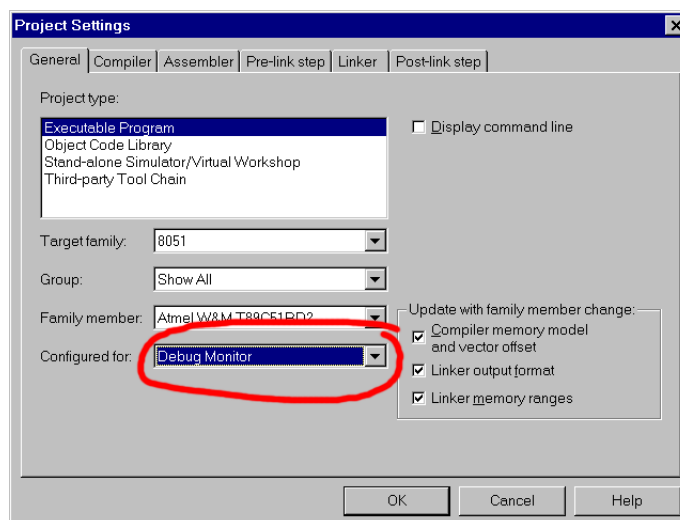
When this is complete, select FLIPs RS232 view and click on **Disconnect** to release the serial port for use by the Embedded Development Studio. Press the RESET button on the target board to start the debug monitor.

RUNNING YOUR PROGRAM IN THE DEBUG MONITOR

From the **File** menu of the Embedded Development Studio, select **Open Project**. Use the **Open Project** dialog box to select the **simple test** project that you created earlier.



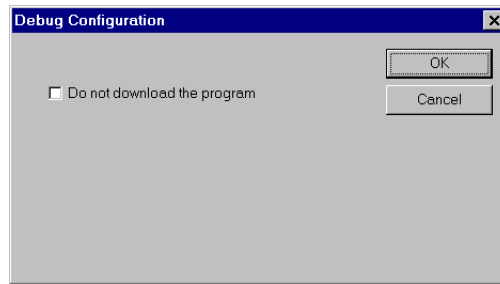
Select **Build**→**Settings** and in the **General** tab change **Standard Operation** to **Debug Monitor** in the **Configured for** drop down list.



Click on **OK**.

Select **Build**→**Rebuild All** to build a version of your program suitable for running with the debug monitor.

Select **Use Remote Debugger** from the **Simulate** menu. The **Simulate** menu will be replaced by a **Debug** menu. Select **Step Into** from the **Debug** menu. The debugger on the PC will communicate with the debug monitor on the target board and after it has successfully identified it, it will display a dialog box:



(If you have forgotten to press RESET to bring the board out of FLIP mode and to start the debug monitor you will get a different message.)

Click on **OK** to start the download process. The status bar will display progress as the debug monitor writes your program into flash memory. When download is complete, startup.asm will be displayed with the cursor positioned just after the `__START` label. As with the simulator, single step up to the first **for** loop in main.c. Then select **Go**. You will see the yellow LED's for pins P0.0 to P0.7 flashing as they were when your program was running stand-alone. However this time they will be flashing more rapidly. This is because the debug monitor has configured the chip to run in X2 mode - twice as fast as before.

Select **Halt Debugger** from the **Debug** menu. Your program will stop and the LEDs will stop flashing. Position the cursor on the line:

```
_p0 = 0X00;
```

and select **Set Breakpoint at Cursor** from the **Debug** menu. Move the cursor to the line:

```
_p0 = 0XFF;
```

and select **Set Breakpoint at Cursor** again.

You will see red vertical bars on the left hand side of these two lines indicating that the breakpoints have been set.

Now select **Go** from the **Debug** menu. Your program will run to one of these breakpoints. Select **Go** again and your program will run to the other breakpoint. Repeatedly select **Go** (or press the F5 key) and you will see the LEDs on the target board turning on or off after each command.

You have now created a test program, run it in the simulator, run it stand-alone on the target board and run it on the target board with the debugger.

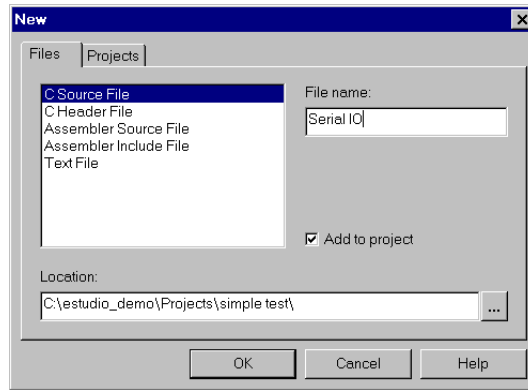
The next step is to add I/O handlers to your program so that it can use the UART for communication purposes.

ADDING UART I/O HANDLERS

C programs require functions `putch()` and `getch()` to handle the I/O for library functions such as `puts()` and `gets()` to send and receive data. For the T89C51RD2, the Crossware UART Wizard will create these for you.

Two sets of handler are created by the Wizard - one which will operate stand-alone and the other which operate with the hooks provided in the debug monitor. (The debug monitor uses the UART to communicate with the debugger and so your program must share it with the debug monitor.)

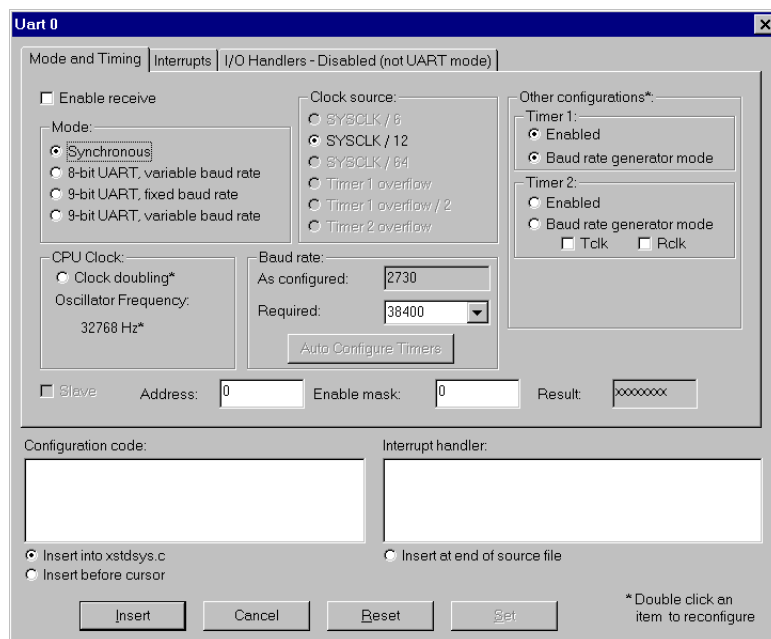
With your **simple test** project loaded into the Embedded Development Studio, select **File→New**. In the **Files** tab select **C Source File** and enter a file name such as **Serial IO** into the **File name** field.



Click on **OK**.

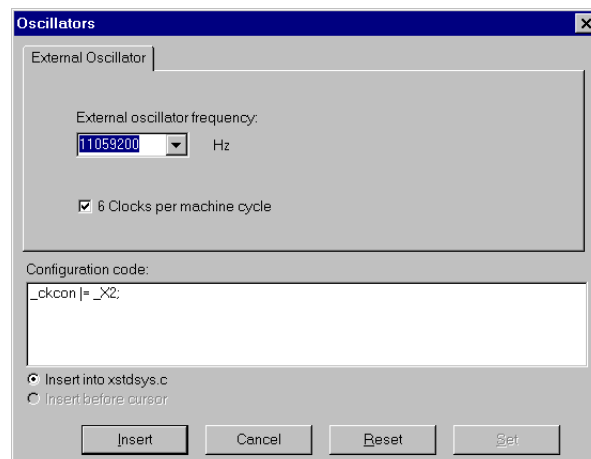
A file with the name Serial IO.c will be created and added to your project. The file will be opened ready for editing.

Place the cursor in this view and select **Uart** from the **Wizards** menu. The Uart Wizard will open.



Observe that the oscillator frequency is 32768 Hz. However, the oscillator on the target board is 11,059,200 Hz. Therefore we first need to set the oscillator frequency.

Double click on the **Clock Doubling** item or on the **32768 Hz** item. The Oscillator Wizard will open. Select **11059200** from the **External Oscillator Frequency** drop down list. Check the **6 Clocks per machine cycle** to enable clock doubling.

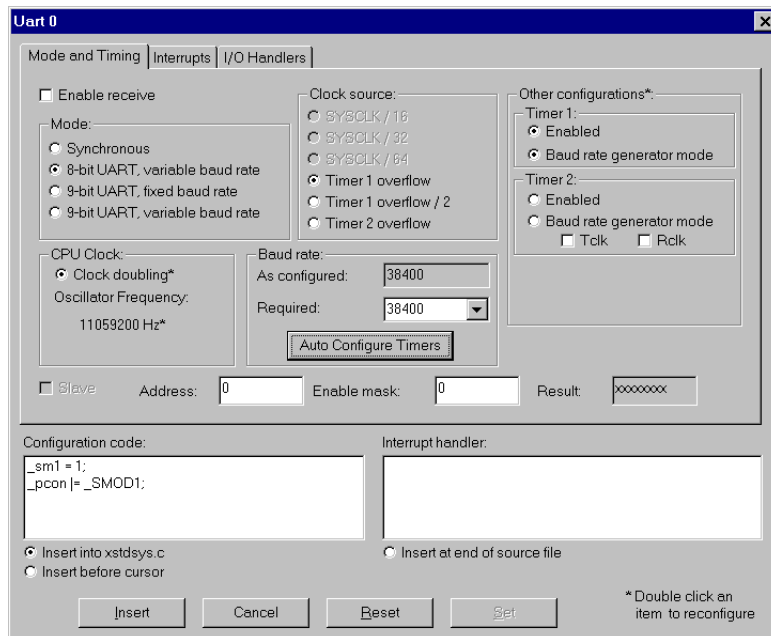


Observe the configuration code for enabling clock doubling.

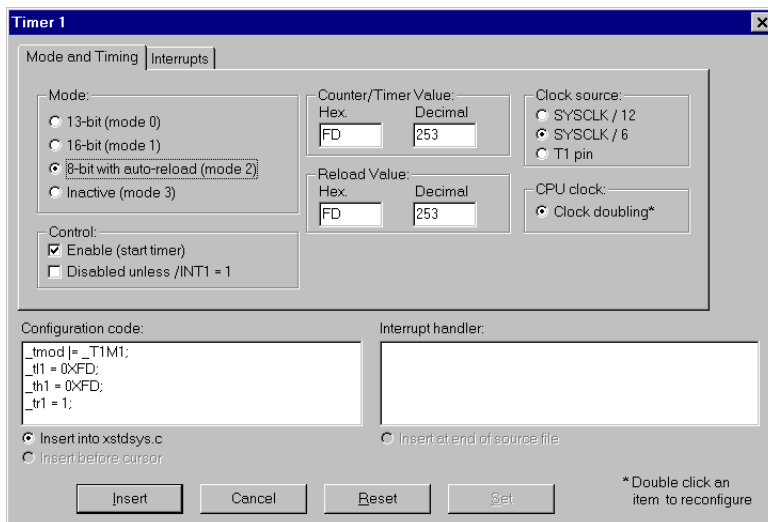
Click in **Insert**. The oscillator configuration code will be written into your **xstdsys.c** project file which was created when you created your project.

You will then be returned to the UART Wizard.

Select **38400** from the **Required** drop down list in the **Baud Rate** group. Click on **8-bit UART, variable baud rate** in the **Mode** group. Click on **Timer 1 overflow** in the **Clock Source** group. Then click on the **Auto Configure Timers** button. Timer 1 will be configured as a baud rate generator with the correct overflow rate to give a baud rate of 38400.

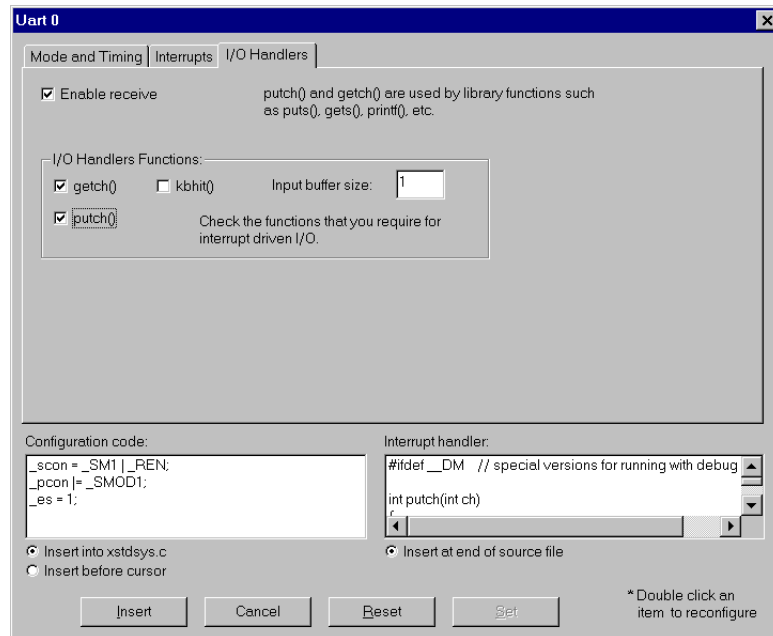


The timer 1 configuration code will have been immediately written into xstdsys.c. You can see this configuration code. Double click on the **Enabled** item in the **Timer 1** group. The Timer 1 Wizard will open:



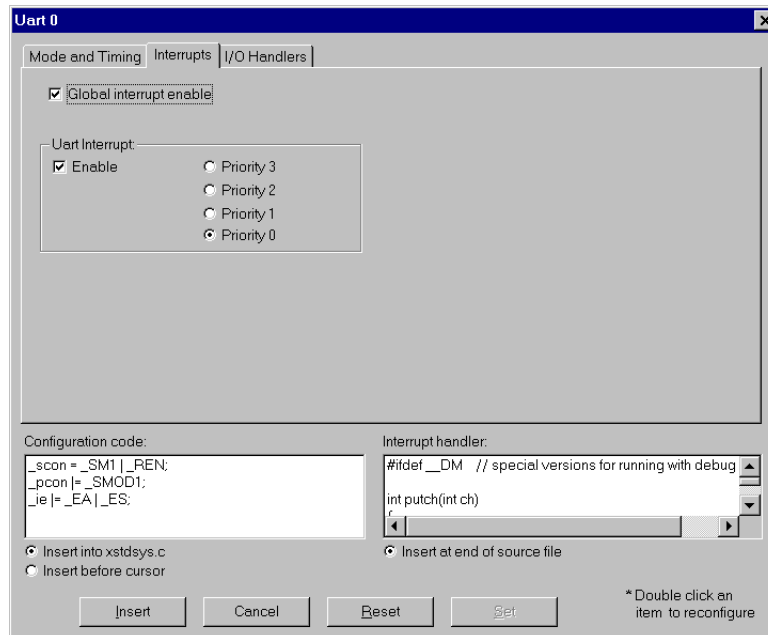
Click on **Cancel** to leave this unchanged and return to the UART Wizard.

In the UART Wizard, click on the **I/O Handlers** tab. Check the **getch()** item in the **I/O Handlers Functions** group and also check the **putch()** item in this same group.



Notice that the **Interrupt handler** list box has been filled.

Now click on the **Interrupts** tab. Observe that the UART interrupt has already been enabled. This was done when you clicked in getch() and putch() since these are interrupt driven. However, it is also necessary at some point to enable global interrupts. So check the **Global interrupt enable** item to do this.



Click on **Insert**. The interrupt handler code will be inserted into **Serial IO.c** and the UART configuration code will be written into **xstdsys.c**.

Now open your source file **main.c** and add the source code shown in bold below:

```
// 8051 Initial C Source File
#include "xstdsys.h"

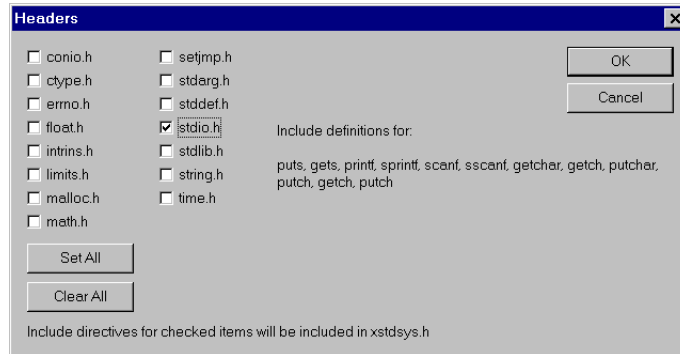
main()
{
    // use the Wizards (see Wizards menu)
    // to configure the microcontroller
    puts("Enter characters now:");
    while (1)
    {
```

```

long i;
int ch;
for (i = 0; i < 1000; i++);
_p0 = 0X00;
ch = getch();
putch(ch);
for (i = 0; i < 1000; i++);
_p0 = 0XFF;
ch = getch();
putch(ch);
}
}

```

Next, select **Headers** from the **Wizards** menu.



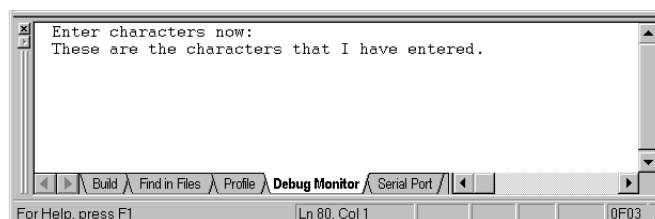
Check the **stdio.h** item and click on **OK**. The line:

```
#include <stdio.h>
```

will be added to header file **xstdsys.h**. File **xstdsys.h** was created when you created your new example project and is automatically included in new C source files created from the **New** menu.

Select **Build->Rebuild All** to rebuild your program.

Your project is still configured to operate with the debug monitor and so once your modified program has been successfully compiled and linked, you can immediately run it in the debugger by selecting **Go** from the **Debug** menu.



A **Debug Monitor** view will open and the words **Enter characters now:** will be displayed. They were written to this view by the target board. Click the mouse on this **Window** and start typing. The characters that you enter are sent to the target board which is waiting for them in the call to `getch()`. They are immediately echoed back to the PC with the `putch()` call. Notice that each time you enter a character, the P0.0 to P0.7 LEDs on the target board switch on or off.

Select **Debug->Close Debugger** to close the debugger.

Next we will run this same program in stand-alone mode. Select **Build->Setting->General** and change the **Configured for** item back to **Standard Operation**.

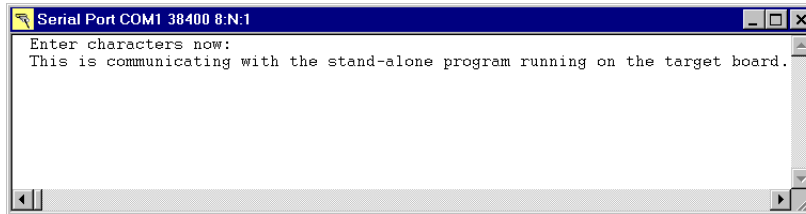
Select **Build->Rebuild All** to rebuild your program. It will be produced in Intel hex format.

Use **FLIP** to write this program into the T89C51RD2 and then disconnect **FLIP** to release the serial port.

In the **Embedded Development Studio**, select **Open Serial Port** from the **Comms** menu.

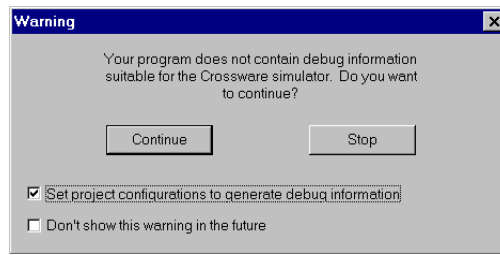
Check the title bar to make sure that the correct COM port has been opened. If you have not disconnected FLIP then the Primary Serial Port will be unavailable and a different COM port will be opened. If this has happened, close the Serial Port window, disconnect FLIP and select **Comms→Serial Port** again to open the correct COM port.

Press the RESET button on the target board. You should see the same thing in the Serial Port window that you saw in the Debug Monitor view. However this time when you enter text into the window, it is echoed back twice - once due to the serial port settings and once by the target board. Right mouse click on the Serial Port window and select **Settings** from the context menu. Click on the **Port Settings** tab. Uncheck **Local Echo** and **CR→CR/LF** in the **Transmit Preferences** group and click on **OK**. Now enter more text and the behaviour will have been corrected.

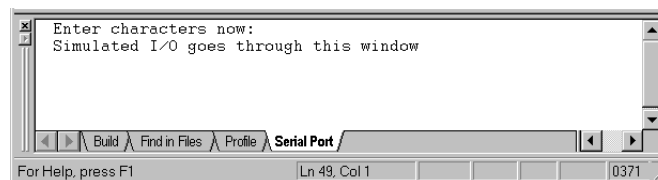


You can also run this same program in the simulator.

Select **Use Simulation** from the **Debug** menu. The Debug menu will be replaced by a **Simulate** menu. Select **Simulate→Go**. A dialog box will be displayed to tell you that your program does not contain and debug information. (The file format is Intel hex which does cannot contain debug records.)



Check the **Set project configuration to generate debug information** item and click on **Stop**. You will be prompted to rebuild your program. When you have done this select **Simulate→Go** again.



Don't forget to change the format back to Intel hex (using the **Build→Settings→Linker** tab) if you want to download your program again to run stand-alone on the target board .

SE-8051CD

T89C51RD2 IN-CIRCUIT DEBUGGER

