MicroConverter™ Technical Note - uC002
Writing an ADuC812 Application in C

INTRODUCTION

This Application Note describes the use of the Keil toolsuite (Keil C51 uVision compiler and Dscope C-Source Simulator) in creating a 'C' application for the ADuC812.

A 2K code limited version of the Keil compiler is included as part of the ADuC812 QuickStart™ Development System from Analog Device, more information on the ADuC812 QuickStart™ Development System and the Keil compiler is available on the Analog Devices MicroConverter web site at www.analog.com/microconverter.

The Keil software development kit has 2 main components namely :

uVision : A windows front end to the 8051 C-Compiler.

dScope : A windows front end to the Keil C-Simulator.

Further information on the range of Keil software deveopment products can be found at www.keil.com.

This application note describes the configuration and simulation of an ADuC812 application written in C and developed in Keil tool's environment. This note assumes you have already installed the Keil toolsuite on your PC and that you have also access to two additional ADuC812 specific header files (ADuC812.inc and ADuC812.h) which are included in the QuickStart™ Development System or available on the Analog Devices web site.

This C-code development example is covered in the following steps :

- Compiling the C-Source Code
    - Configuring uVision-51 for the ADuC812
    - Creating a project
    - Compiling your code

- Simulating the C-Source Code
    - Configuring dScope
    - Customizing the GUI
    - Code Execution

- Conclusions

One Technology Way, P.O. Box 9106, Norwood. MA 02062-9106, U.S.A.
Tel: 617/329-4700 Fax: 617/326-8703

CONFIGURING uVISION-51

Before creating a C-file you first need to configure the uVision-51 tool for the ADuC812. When uVision-51 is launched, the main window appears on the screen as shown in figure 1 below. This is the main work window for your compilation session. To configure the compile session to generate an ADuC812 compatible Intel hex output file you should work through the following steps.

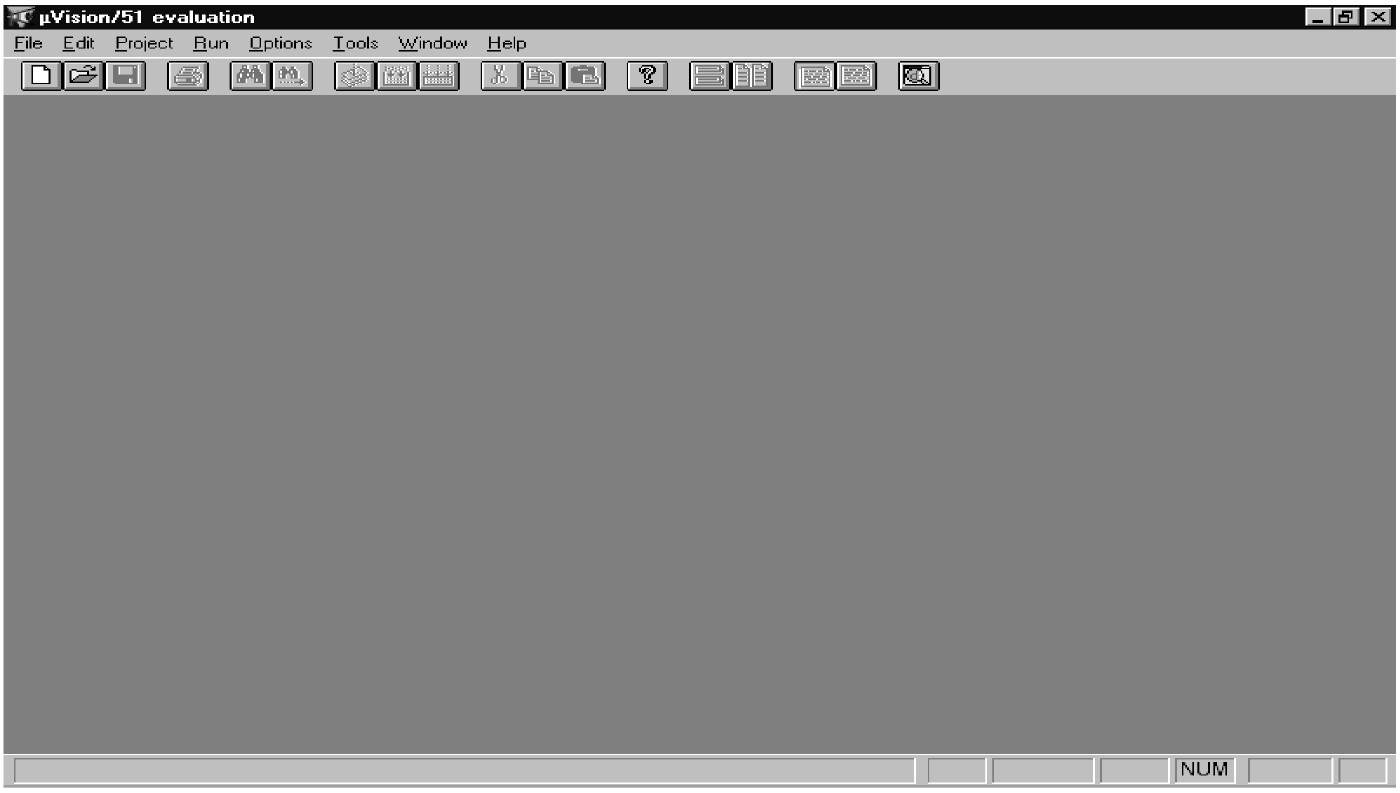
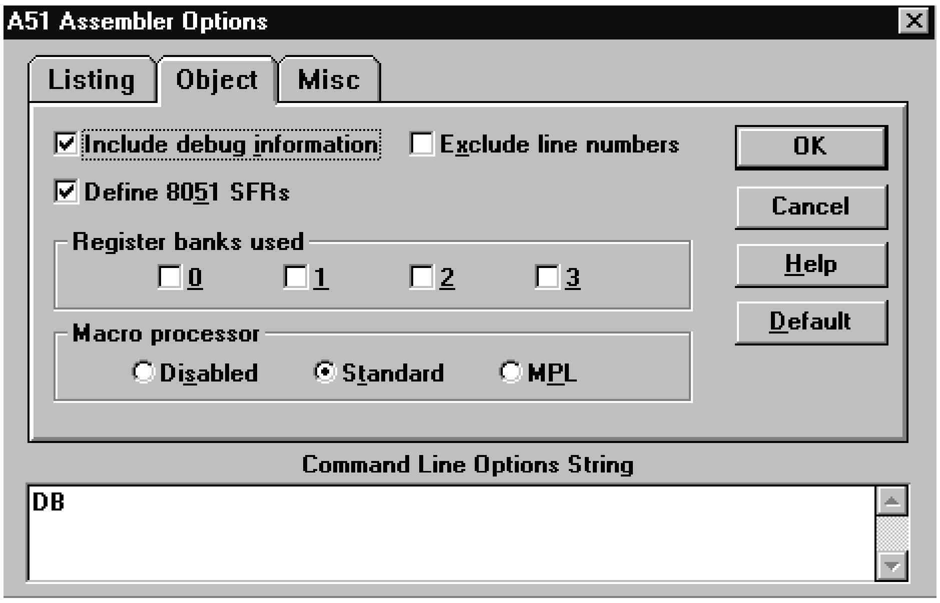


Figure 1. uVision-51 Main window

STEP 1 :
In the main window, choose the Options menu and select the A51 Assembler. From the *'Object'* tab select the options exactly as shown in figure 2.


A51 Assembler Options

Listing | Object | Misc

- [✓] Include debug information
- [ ] Exclude line numbers
- [✓] Define 8051 SFRs

Register banks used: [ ] 0 [ ] 1 [ ] 2 [ ] 3

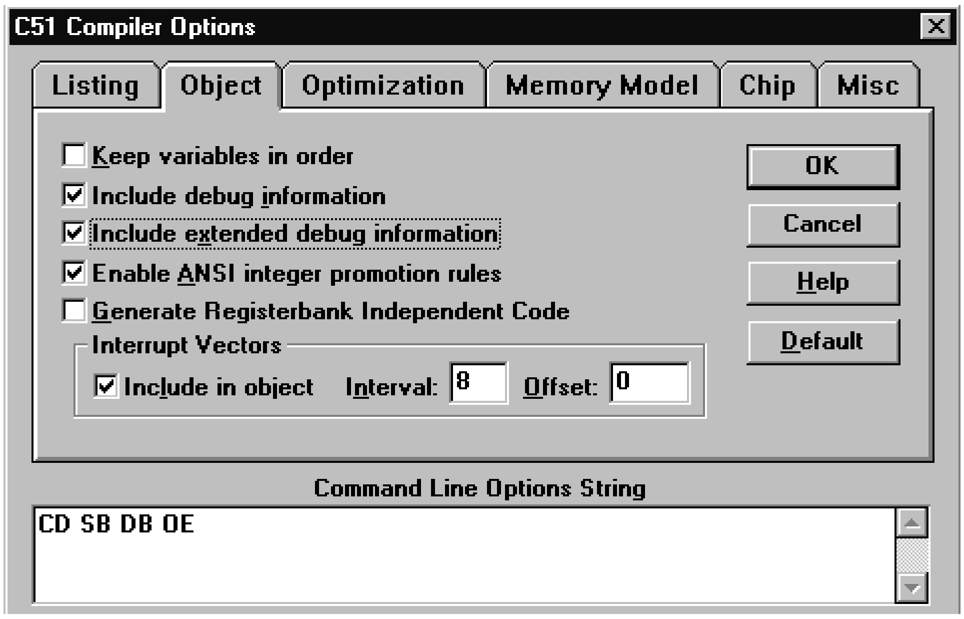
Macro processor: ○ Disabled ● Standard ○ MPL

Command Line Options String

```
DB
```

Figure 2. A51 Assembler Options

STEP 2 :
Again from the Options, select C51 compiler and in the *'Object'* tab select the options exactly as shown in figure 3 below.


C51 Compiler Options

Listing | Object | Optimization | Memory Model | Chip | Misc

- [ ] Keep variables in order
- [✓] Include debug information
- [✓] Include extended debug information
- [✓] Enable ANSI integer promotion rules
- [ ] Generate Registerbank Independent Code

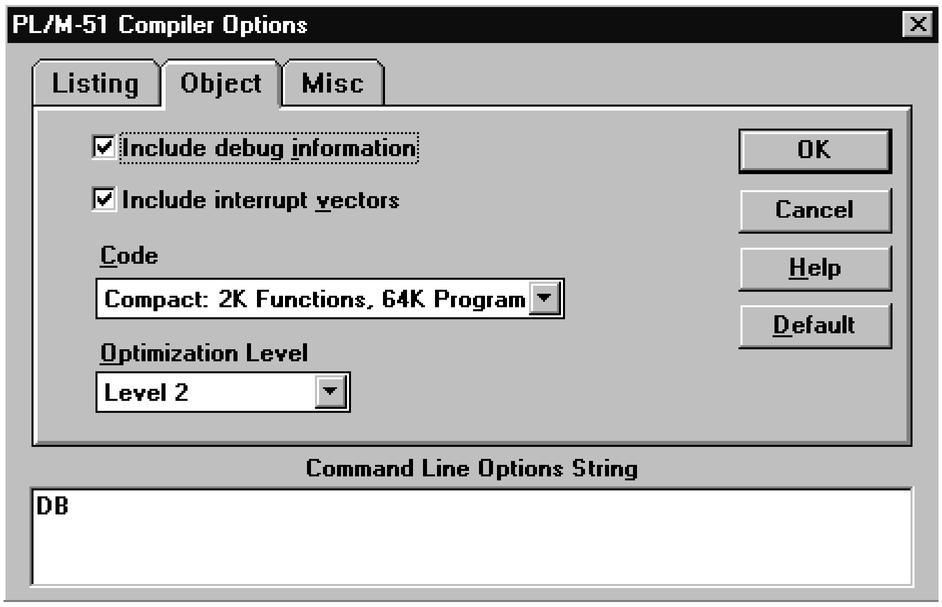
Interrupt Vectors: [✓] Include in object Interval: 8 Offset: 0

Command Line Options String

```
CD SB DB OE
```

Figure 3. C51 Compiler Options

STEP 3 :
From the Options menu, select PL/M-51 Compiler .From the *'Object '* tab select the options exactly as shown in figure 4 below.


PL/M-51 Compiler Options

Listing | Object | Misc

☑ Include debug information
☑ Include interrupt vectors

Code: Compact: 2K Functions, 64K Program

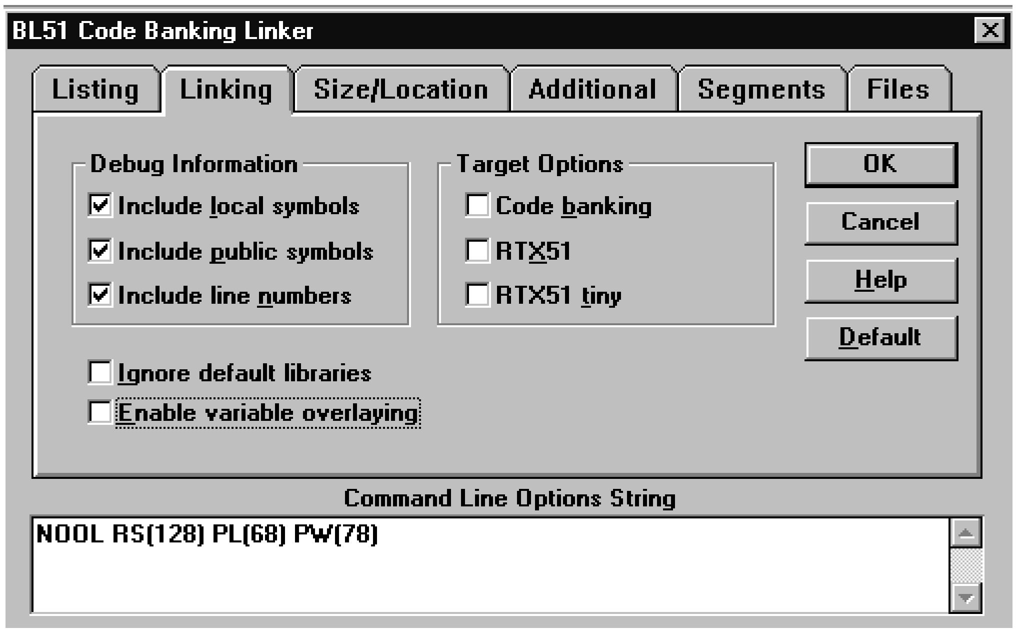
Optimization Level: Level 2

OK | Cancel | Help | Default

Command Line Options String
```
DB
```

Figure 4. PL/M-51 Compiler Options

STEP 4 :
From the Options menu, select BL 51 Code banking linker.From the *'Linking'* tab select the options exactly as shown in figure 5 below.


BL51 Code Banking Linker

Listing | Linking | Size/Location | Additional | Segments | Files

Debug Information
☑ Include local symbols
☑ Include public symbols
☑ Include line numbers

Target Options
☐ Code banking
☐ RTX51
☐ RTX51 tiny

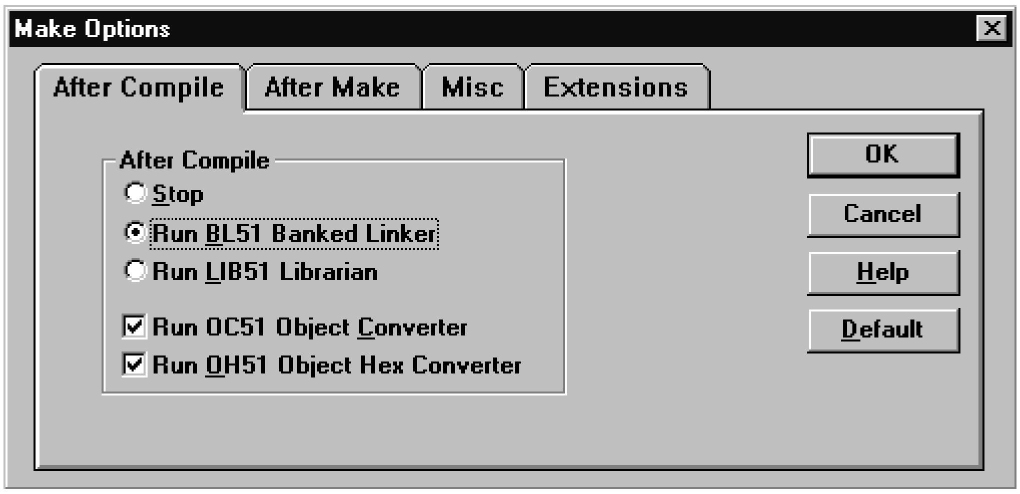
☐ Ignore default libraries
☐ Enable variable overlaying

OK | Cancel | Help | Default

Command Line Options String
```
NOOL RS(128) PL(68) PW(78)
```

Figure 5. BL51 Code banking linker

STEP 5 :
From the Options menu, select Make. From 'Compile ' tab select the options as shown in figure 6 below.



After Compile
- ( ) Stop
- (•) Run BL51 Banked Linker
- ( ) Run LIB51 Librarian
- [✓] Run OC51 Object Converter
- [✓] Run OH51 Object Hex Converter

Figure 6. Make...

STEP 6 :
At this point, to finish the configuration of your uVision-51 you just have to add two libraries in the Keil directory path. Those libraries are specific to the ADuC812 and basically include memory map information detailing the additional SFR's used in the ADuC812.

These additional files are :

ADuC812.inc
ADuC812.h

and are available as part of the QuickStart^{TM} Development System or from the Analog Devices Web site at www.analog.com/microconverter.

The ADuC812.inc file contains all the Processors Declarations while the ADuC812.h file is the header file specific to the ADuC812.

Copy the ADuC812.inc in the Asm directory (the path is *c:\ckeil\asm*).
Copy the ADuC812.h in the Bin directory (the path is *c:\ckeil\bin*).

CREATING A PROJECT

Before writing any C-code, a project associated with our code needs to be created. This can be accomplished by creating a new folder in the Ckeil directory.

Create a folder named Pres in this path : c:\ckeil\examples\pres\.

You can now create a new project in uVision-51. From the main window, choose the Edit menu and select New project. A new window appears as shown in figure 7 below.

Select the folder that you've created previously (pres) and on the left side of the window type the name of your new project, eg. pres.prj and press OK.

Note: After you press OK, a new window appears,this can be ignored, press Cancel to remove.

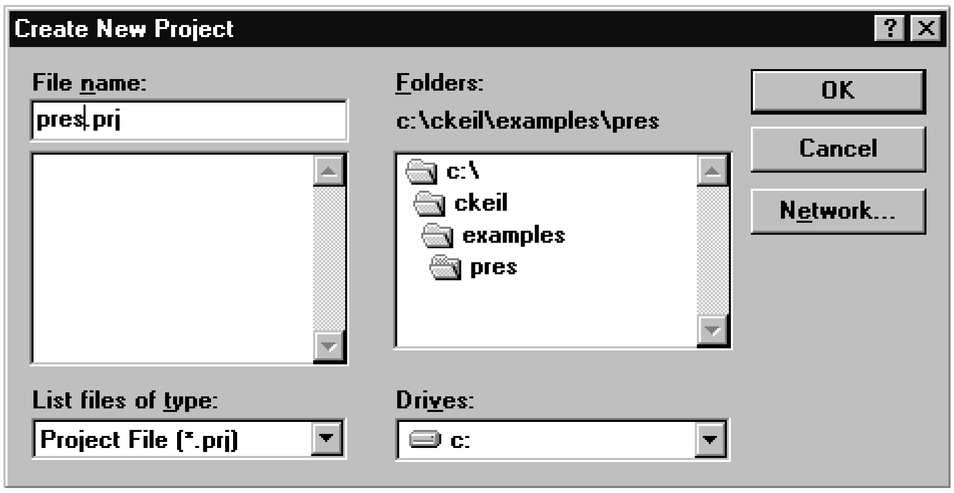


Figure 7. Create New project window

CREATION OF THE C-FILE

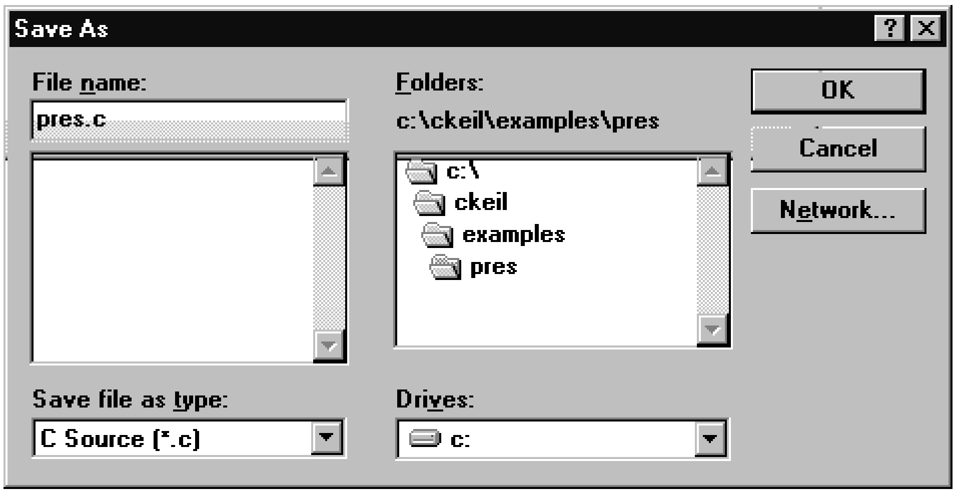
Now it's time to write the C-file. In the main window, choose the File menu and select New. A new window named <untitled 1> appears on the screen as shown below on figure 8.

Figure 8. Create New project window

Type the following short C source code in this new window.

```
/* pres.C - this routine uses the external  INT0 pin to trigger conversions on the ADC and DACs */
#pragma DEBUG OBJECTEXTEND CODE        /* pragma lines can contain state C51       */
#include <stdio.h>                          /* prototype declarations for I/O functions */
#include <ADuC812.h>                        /* ADuC812 header file            */


void interrupt_0 () interrupt 0
{
        ADCCON1 = 0x7C ;
        ADCCON3 = 0x00 ;
        ADCCON2 = 0x10 ;      // single conversion
        printf ( "ADC Channel 0 is converting\n" );
        while ( EA == 1 )
        {
        }

}

void wait (void)
{                // wait function
 ;                  // only to delay for LED flashes
}


void main (void)
{
     int a ;


        TCON ^= 0x01 ;        // setup for the interrupt 0 routine
        IE ^= 0x80 ;
        IE ^= 0x01 ;

        SCON = 0x52;          // setup for the serial port
        TMOD |= 0x20;
        TH1 = 0xFD;
        TR1 = 1;
        TI = 1;

        DAC0H = 0x00 ;
        DAC0L = 0x00 ;
        DACCON = 0xA9 ;
        while (1)
        {
                EA = 1 ;
                printf (" DAC0 updated with : %bX \n ",DAC0L) ;
                for (a = 0; a < 10000; a++)    // Delay for 10000 Counts
                        {
                        wait ();      // call wait function
                        }
                printf ("\n" ) ;
                DAC0L++;

        }
}
```

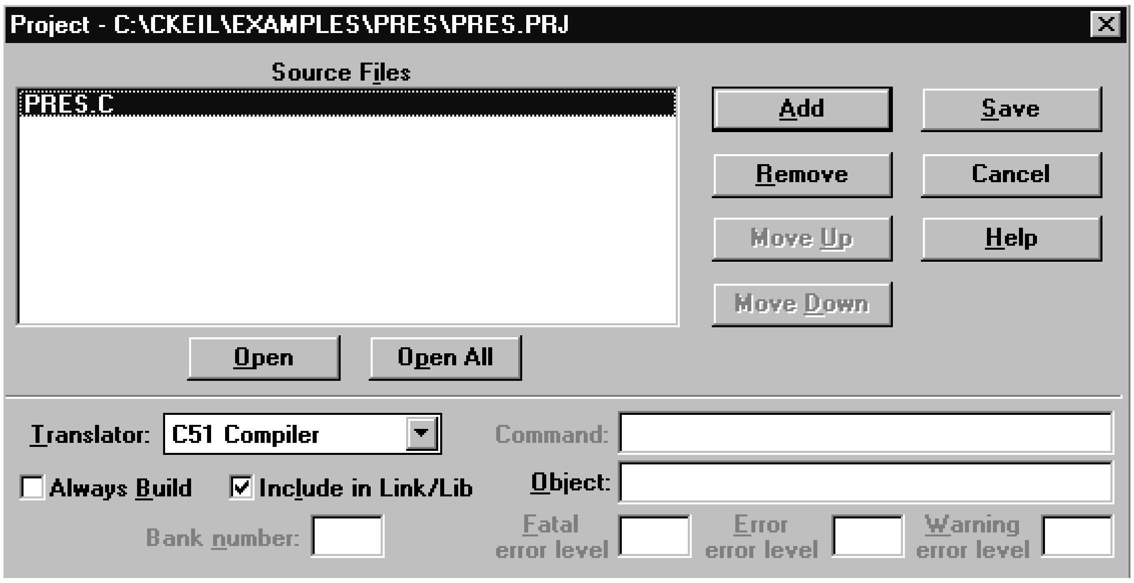
Once you've typed all the code, in the main window choose the file option and select Save. A new window appears as shown in figure 9.
Save your new file (pres.c) under the pres folder you've created previously.
Note: the extension .c is created automatically by uvision-51.



File name: pres.c
Folders: c:\ckeil\examples\pres
c:\
ckeil
examples
pres
Save file as type: C Source (*.c)
Drives: c:

Figure 9. Save as window

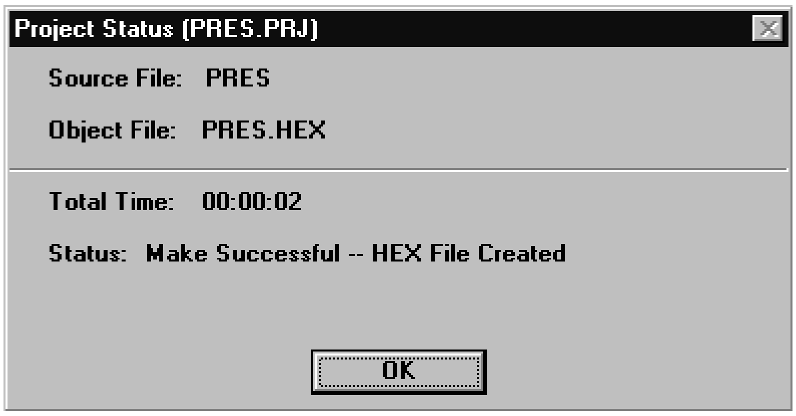
Now before compiling the C-file, we need to include it in our project. From the main window, choose the Project option and select Edit Project. A new window appears as shown in figure 10 below. Click on the Add button, a new window appears. Select the pres.c file which is in our pres folder and click on Add. The result is the window like figure 10 below.



Project - C:\CKEIL\EXAMPLES\PRES\PRES.PRJ
Source Files: PRES.C
Translator: C51 Compiler
☐ Always Build
☑ Include in Link/Lib

Figure 10. Edit project window

The last step in the use of uVision-51 is the compilation step. In the main window, choose the menu Project and select the option Make:update project. The C51 compiler will compile the pres project. If everything goes well the window shown in figure 11 should appear on the screen otherwise it means that either there are mistakes in the c-file or that the configuration of the uVision is incorrect. At this time, all the files that you need (pres.hex, pres.obj, pres.lst etc...) have been created in your pres folder.


Project Status (PRES.PRJ)

Source File: PRES

Object File: PRES.HEX

Total Time: 00:00:02

Status: Make Successful -- HEX File Created

Figure 11. Project status window

SIMULATING THE C-SOURCE CODE

To create the simulation, we need to use the second tool from Keil, Dscope (figure 12). Before simulating the code that has just been compiled, we need to configure Dscope for the ADuC812 by working through the steps in the following pages.

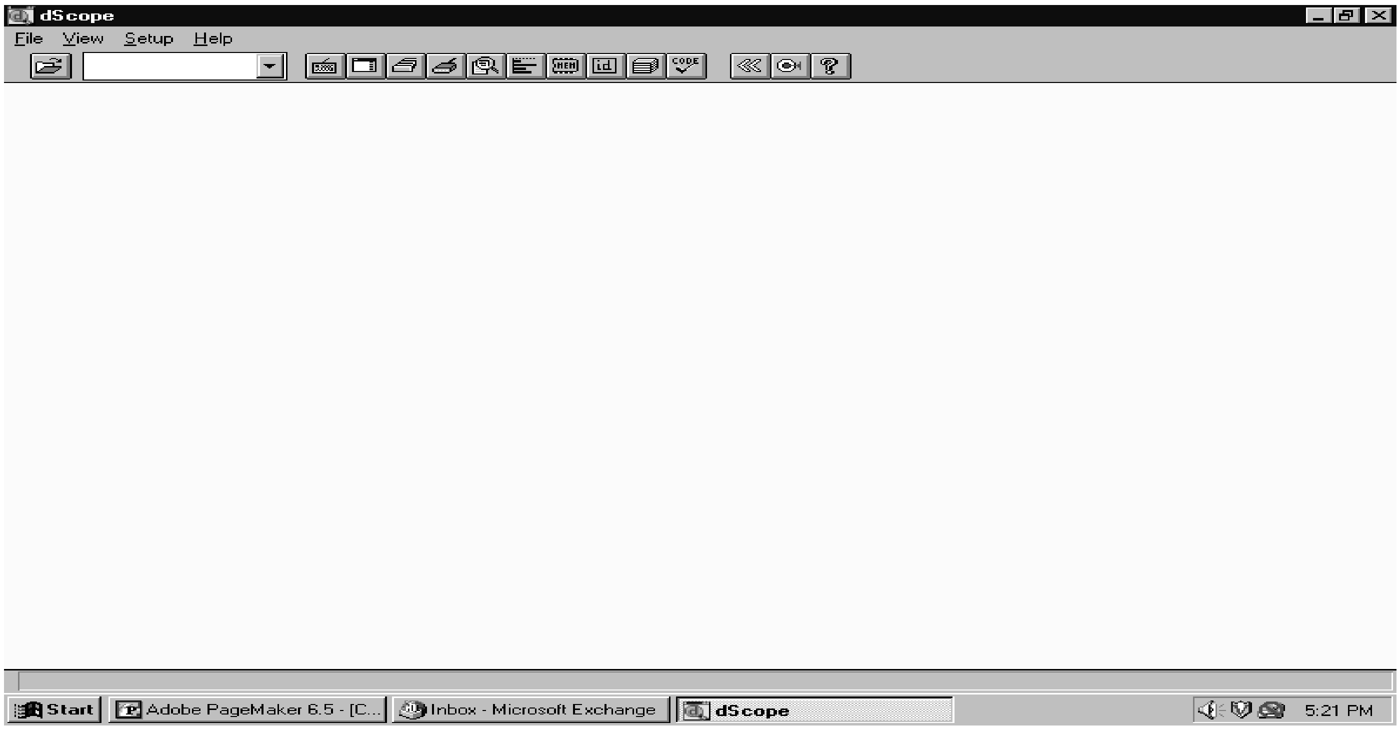


Figure 12. Dscope main window

CONFIGURING DSCOPE

When you launch Dscope, it's possible that the window that you see is not exactly the same that in figure 12. Some small windows can appear inside the main window. To hide those windows use the button bar (figure 13) which is at the top of the main window. By clicking on each button, you'll make a related window appear or disappear.

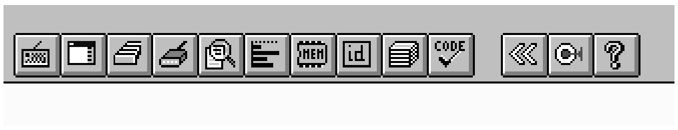


Figure 13. Button bar

The first thing you have to do is to select the processor DLL. In the main window, on the top left corner, there is a scrolling bar. As shown in figure 14 select the 8052 DLL (the core of the ADuC812 is an 8052).

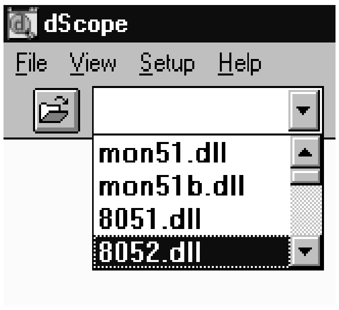
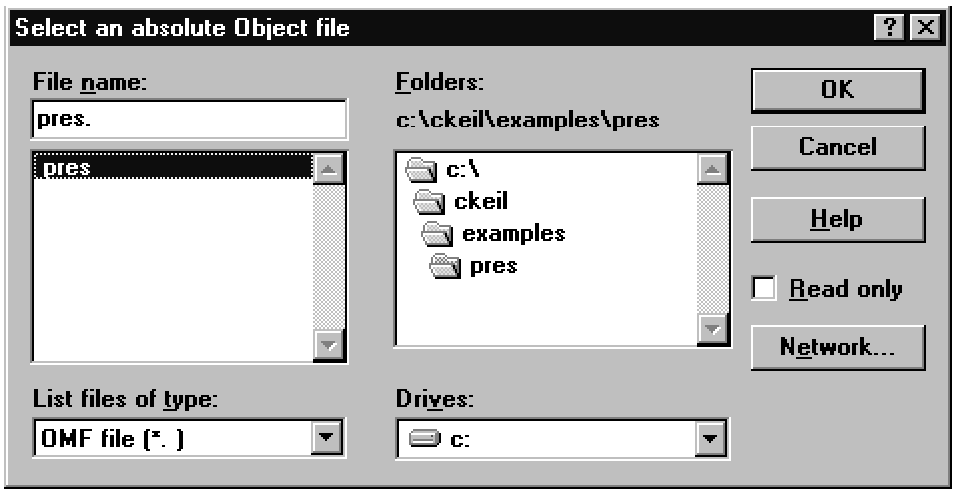


Figure 14. Selection of the DLL

After you've selected the DLL you must declare your object file which has been compiled previously under the C51 compiler. In the main window, choose the menu File and select Load object file. A new window appears as shown in figure 15. Select the file you've created previously (ie pres).



Select an absolute Object file

File name: pres.

pres

Folders: c:\ckeil\examples\pres

c:\
ckeil
examples
pres

List files of type: OMF file (*.)

Drives: c:

Figure 15. Selection of the Absolute object file

CREATION OF THE SIMULATION ENVIRONMENT

The following steps will allow you to create a generic simulation environment that you may want to customize to you own requirements. From the top tool bar, click on the buttons that are shown on figure 16 below.

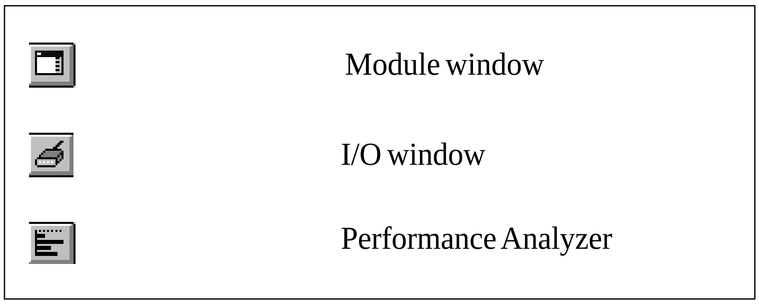
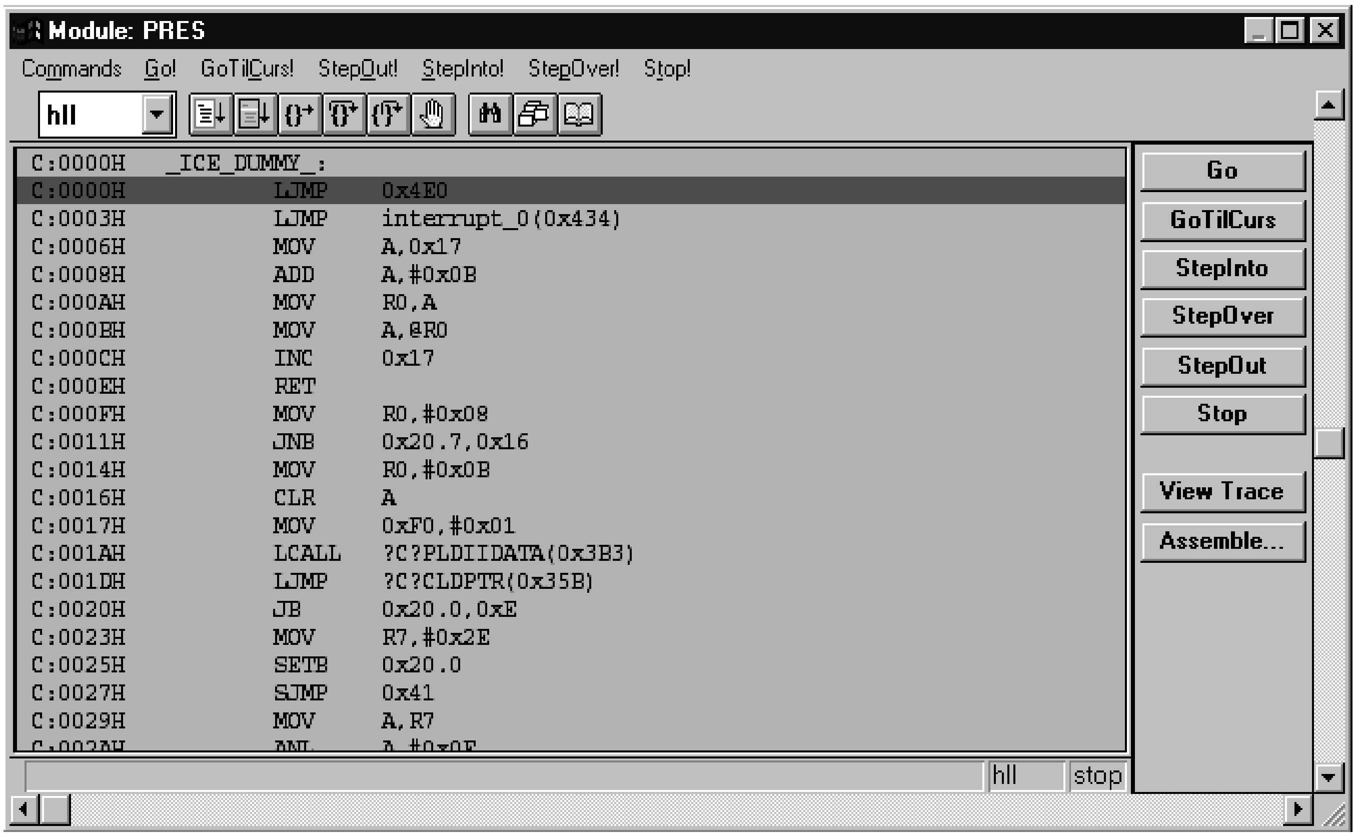


Figure 16. The windows

The windows shown in figure 17, 18 and 19 appear on the screen. Arrange them as required.



```
C:0000H   _ICE_DUMMY_:
C:0000H           LJMP    0x4E0
C:0003H           LJMP    interrupt_0(0x434)
C:0006H           MOV     A,0x17
C:0008H           ADD     A,#0x0B
C:000AH           MOV     R0,A
C:000BH           MOV     A,@R0
C:000CH           INC     0x17
C:000EH           RET
C:000FH           MOV     R0,#0x08
C:0011H           JNB     0x20.7,0x16
C:0014H           MOV     R0,#0x0B
C:0016H           CLR     A
C:0017H           MOV     0xF0,#0x01
C:001AH           LCALL   ?C?PLDIIDATA(0x3B3)
C:001DH           LJMP    ?C?CLDPTR(0x35B)
C:0020H           JB      0x20.0,0xE
C:0023H           MOV     R7,#0x2E
C:0025H           SETB    0x20.0
C:0027H           SJMP    0x41
C:0029H           MOV     A,R7
C:002AH           ANL     A,#0x0F
```

Figure 17. Module window

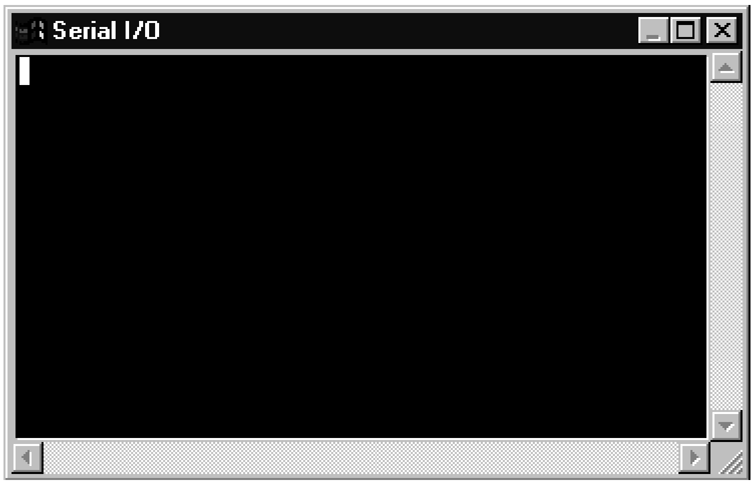
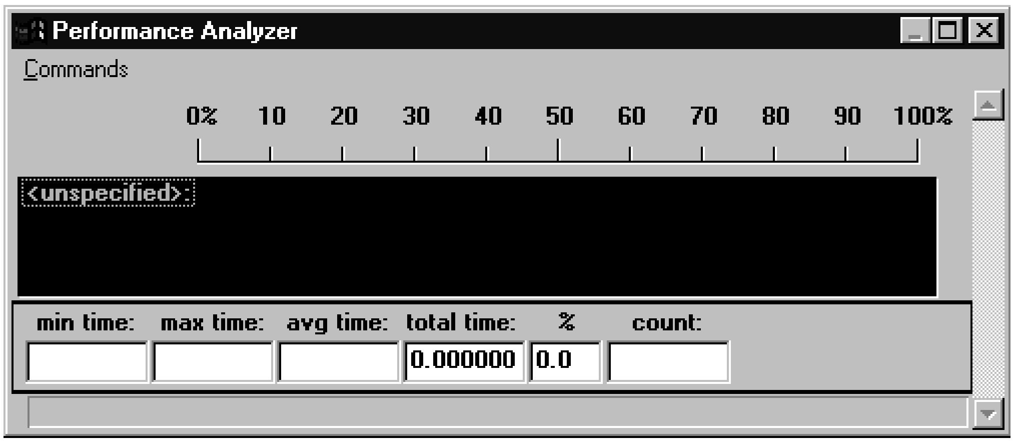


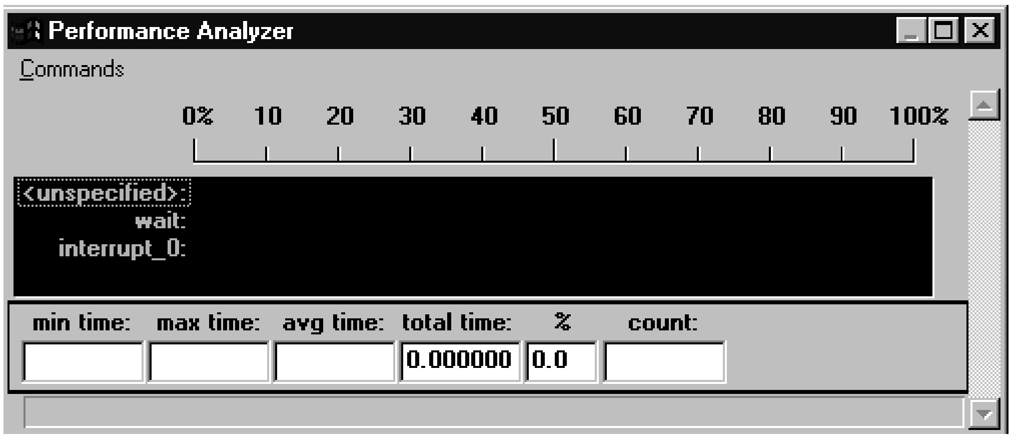
Figure 18. Serial I/O window



```
<unspecified>:
min time:   max time:   avg time:   total time:   %     count:
                                    0.000000      0.0
```

Figure 19. Performance Analyzer window

The next step is the setup of the Performance Analyzer. From the main window, choose the Setup menu and select Setup performance analyzer. In the new window, on the exp: line, type wait and then click on define range. Then on the exp: line, type interrupt_0 and click on define range. Once complete, you can close the setup window by clicking on the Close button. You'll see that on the Performance analyzer window, things have changed (see figure 20). This analyzer gives you the percentage of time that is spend on each function of the code during its execution. The line unspecified is the main function. All you've done is to specify to the Performance Analyzer the functions you're using in your C-file. The use of the Performance Analyzer will be more obvious during the execution of the code later in the application.

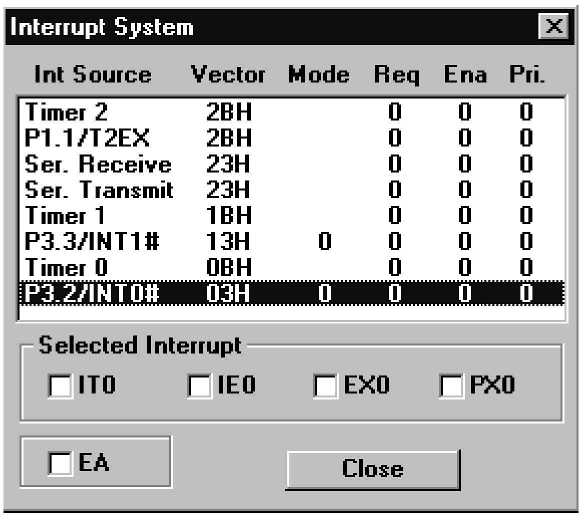


```
<unspecified>:
         wait:
  interrupt_0:
```

min time:	max time:	avg time:	total time:	%	count:
			0.000000	0.0	

Figure 20. Performance Analyzer after setup

The last window we need for the simulation is the window that controls interrupt generation. In our C-Source code example entered previously, an active INT0 will halt DAC execution and trigger an ADC channel#0 conversion. To configure interrupts, choose the menu peripheral in the main window and then select interrupt. A new window appears. Click on the last line of this window (P3.2/INT0#). The result should be like in figure 21 below.



Int Source	Vector	Mode	Req	Ena	Pri.
Timer 2	2BH		0	0	0
P1.1/T2EX	2BH		0	0	0
Ser. Receive	23H		0	0	0
Ser. Transmit	23H		0	0	0
Timer 1	1BH		0	0	0
P3.3/INT1#	13H	0	0	0	0
Timer 0	0BH		0	0	0
P3.2/INT0#	03H	0	0	0	0

Selected Interrupt: ☐ IT0 ☐ IE0 ☐ EX0 ☐ PX0

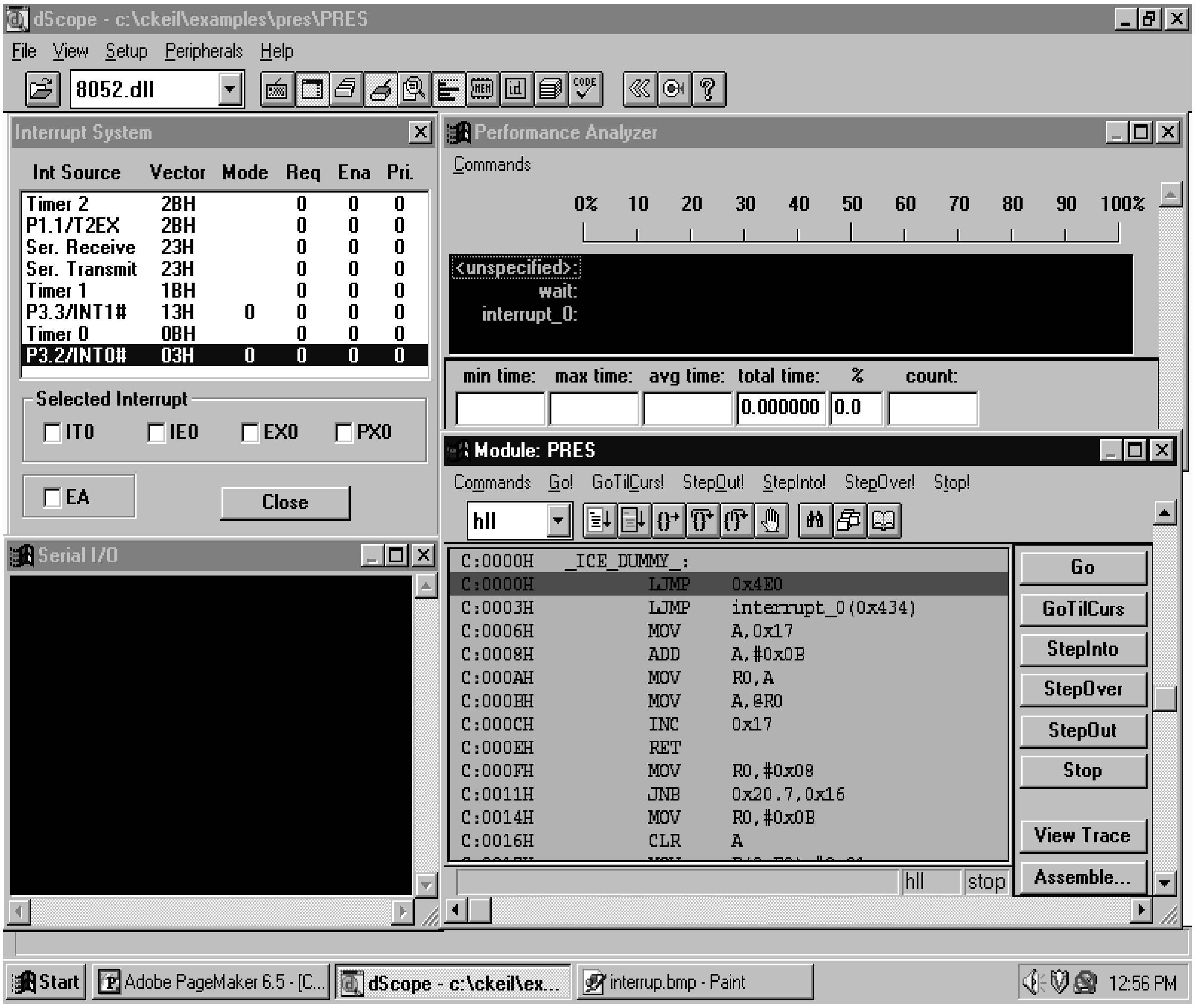
☐ EA

Figure 21. Interrupt window

Rearrange the four windows (module. serial I/O, Performance Analyzer, Interrupt system) as shown overleaf.

CODE EXECUTION

At this time the Dscope main window should look like figure 22.



Int Source	Vector	Mode	Req	Ena	Pri.
Timer 2	2BH		0	0	0
P1.1/T2EX	2BH		0	0	0
Ser. Receive	23H		0	0	0
Ser. Transmit	23H		0	0	0
Timer 1	1BH		0	0	0
P3.3/INT1#	13H	0	0	0	0
Timer 0	0BH		0	0	0
P3.2/INT0#	03H	0	0	0	0

Selected Interrupt: ☐ IT0 ☐ IE0 ☐ EX0 ☐ PX0

☐ EA

0% 10 20 30 40 50 60 70 80 90 100%

```
<unspecified>:
         wait:
  interrupt_0:
```

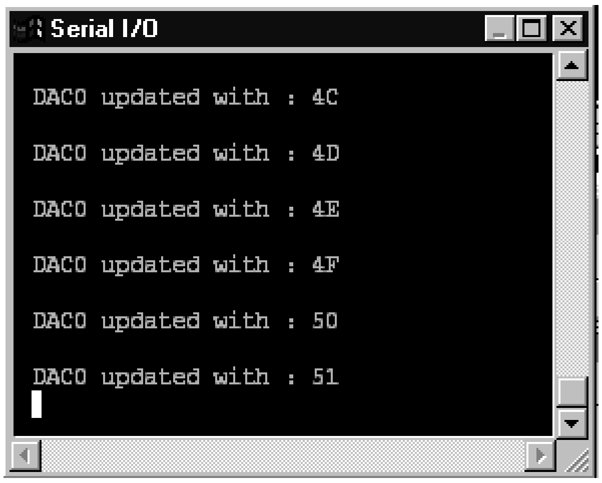
min time:	max time:	avg time:	total time:	%	count:
			0.000000	0.0	

```
C:0000H   _ICE_DUMMY_:
C:0000H            LJMP     0x4E0
C:0003H            LJMP     interrupt_0(0x434)
C:0006H            MOV      A,0x17
C:0008H            ADD      A,#0x0B
C:000AH            MOV      R0,A
C:000BH            MOV      A,@R0
C:000CH            INC      0x17
C:000EH            RET
C:000FH            MOV      R0,#0x08
C:0011H            JNB      0x20.7,0x16
C:0014H            MOV      R0,#0x0B
C:0016H            CLR      A
```

Figure 22. New main Dscope window

The pres.c C-Source routine exercises the ADC conversion, the DAC conversion and the INT0 interrupt system. In the main function DAC0 is updated constantly with a linear ramp. If you generate a simulated INT0 interrupt, the code executed is the code contained in the interrupt_0 function ; a conversion is triggered on ADC channel 0 and you're in a wait state until you release the interrupt to return in the main code.

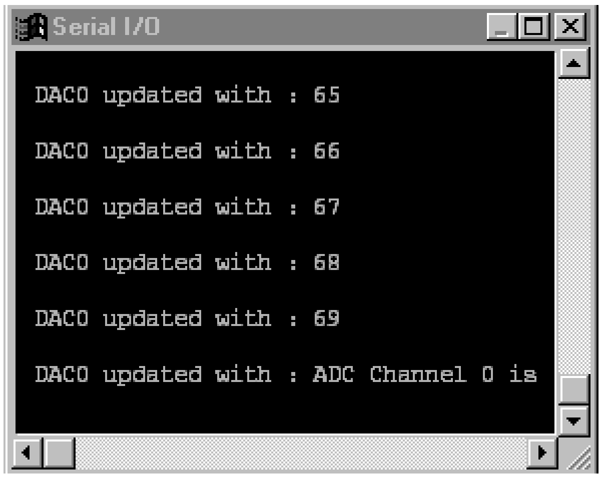
To start the simulation, click on the GO button in the module window. The serial I/O window should be updated as shown in figure 23 below.



```
DAC0 updated with : 4C

DAC0 updated with : 4D

DAC0 updated with : 4E

DAC0 updated with : 4F

DAC0 updated with : 50

DAC0 updated with : 51
```

Figure 23. Serial window updated after GO

To generate an INT0 interrupt, click in the Interrupt system window on IE0. Once you've done that, the serial window is updated (figure 24) and the code is waiting for you to release the interrupt. To release the interrupt, in the Interrupt system window, click on EA. The code will go back in the main function and as a result, the serial window looks again like in figure 23.

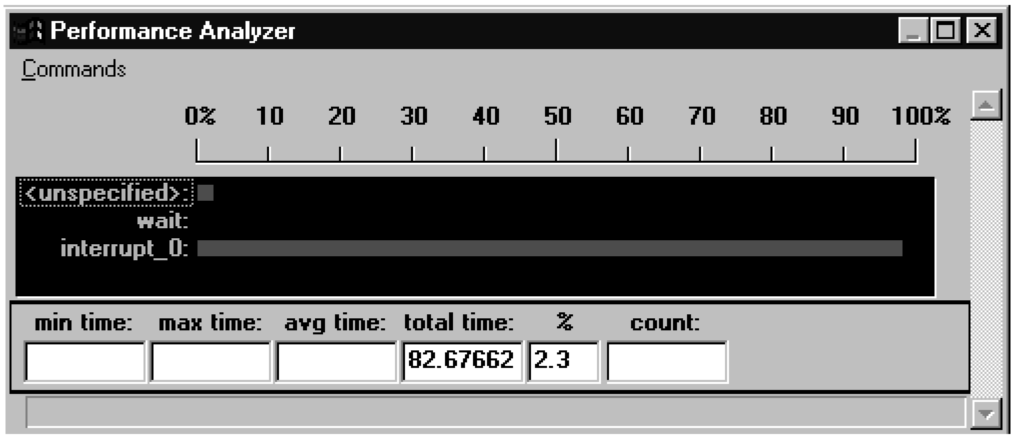


```
DAC0 updated with : 65

DAC0 updated with : 66

DAC0 updated with : 67

DAC0 updated with : 68

DAC0 updated with : 69

DAC0 updated with : ADC Channel 0 is
```

Figure 24. Serial window updated after INTO

The Performance Analyzer allows the programmer to estimate the run time efficeny of code. Figures 25 and 26 describe code execution scenarios that could be analysed during this simulation.

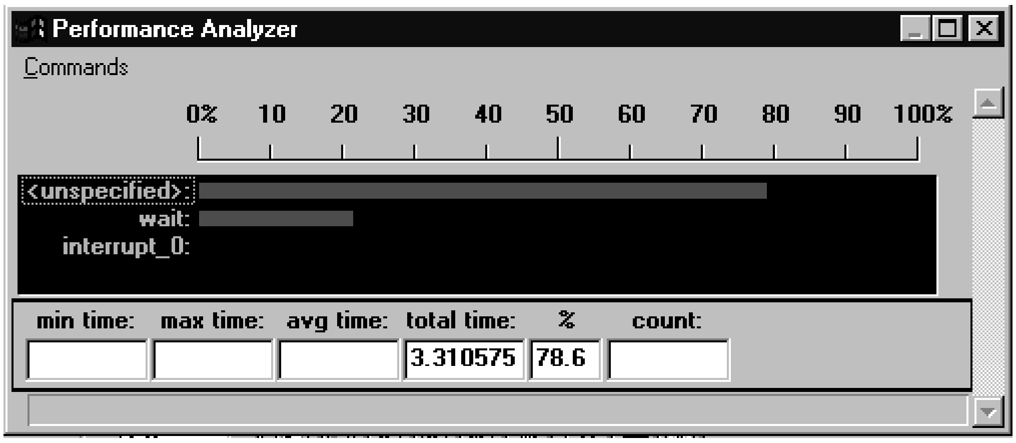
In the first scenario (#1) shown in figure 25 below, the analyzer shows code execution during an active interrupt period. The conversion on ADC channel 0 has been generated and the code is waiting for the user to release the interrupt.



min time:	max time:	avg time:	total time:	%	count:
			82.67662	2.3	

Figure 25.Scenario #1: Performance Analyzer - Interrupt Activated

In this second scenario (#2) shown in figure 26 below, the analyzer shows code execution during the main function with calls to the wait function. The DAC0 is continuously updated and there is obviously no activity on the interrupt_0 bar.



min time:	max time:	avg time:	total time:	%	count:
			3.310575	78.6	

Figure 26. Scenario #2: Performance Analyzer - Main Loop Active

CONCLUSION

This application note is a starting point in the use of the Keil C51 Compiler and Keil Dscope simulator. The HEX file generated by the compiler can be downloaded directly on the ADuC812 QuickStart^{TM} evaluation board using the on-chip serial downloader.

There are many aspects in the Keil toolsuite; this application note illustrated some of them. These tools are a powerful means to develop your ADuC812 applications in high level source code.