

CONTENTS:

1.0	Installation	pg 2
2.0	The Metalink Assembler	pg 3
3.0	The ADuC Windows Serial Downloader (WSD)	pg 4
4.0	The ADuC Serial Port DeBugger	pg 7
5.0	The ADuC Single Pin Emulator	pg 11
6.0	The ADuC Simulator	pg 19
7.0	The ADuC WASP	pg 24
8.0	The ADuC834 Temperature Sensor Demonstration	pg 27
9.0	ADuC834 C-Library	pg 28
10.0	Installed Documentation and Code Directory	pg 29



The ADuC834 QuickStart Development System

Technical Support :

North America :

Grayson.king@analog.com

Europe and ROW :

Darragh.Maxwell@analog.com

Japan :

Toru.Fukushima@analog.com**Marketing Support :**Grainne.Murphy@analog.com

1.0 INSTALLATION

Installing from CD:

- Insert the MicroConverter® QuickStart™ Development System CD ROM into you CD ROM drive and double-click on the file "setup.exe".
- Follow the on screen instructions to install the software on your PC.

Notes:

- Although you can install the software onto any hard drive and into any directory you wish, for the purposes of simplicity the rest of this document will assume that you've installed at the default location of C:\ADuC.
- If you already have a previous 8XX QuickStart Development System tool-suite installed on your machine, this version may also be installed by default at C:\ADuC. The ADuC834 Software tools installation is fully compatible with any previous 8XX QuickStart software installation and will not affect the functionality of the previous tool-suite.


2.0 THE METALINK ASSEMBLER

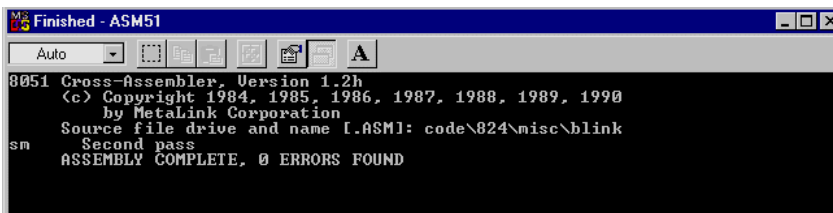
The Metalink 8051 Cross Assembler takes an assembly language source file created with a text editor and translates it into two files, a listing file output (.lst) and a machine language object file in Intel Hex standard format (.hex).

The listing file output (.lst) displays the results of the assembler translation, including any syntax or other errors present in the original source code.

The machine language object file is the second file generated during the assembler process. It is formatted as standard Intel Hex. This file can be used to program the part in-circuit using the serial downloader.

2.1 Using the Metalink Assembler

1. In the ADuC directory, double-click on the  ASM51).
2. In the DOS window that comes up, type the relative path of the assembly file you wish to assemble. For example, to assemble the example file C:\ADuC\Code\834\misc\blink.asm, simply type "Code\834\misc\blink.asm" as shown below.



```
Finished - ASM51
Auto
8051 Cross-Assembler, Version 1.2h
(c) Copyright 1984, 1985, 1986, 1987, 1988, 1989, 1990
by Metalink Corporation
Source file drive and name [ASM1: code\824\misc\blink
sm
Second pass
ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

The assembler will display the text "ASSEMBLY COMPLETE, 0 ERRORS FOUND" indicating that it has successfully assembled the file and has created the hex and list files (e.g. blink.hex & blink.lst) along side the assembly input file (eg. blink.asm). If the assembler indicates assembly errors, you should view the list file (eg. blink.lst) to examine the errors. To view the list file, open it with notepad or any standard text editor.

Note: If the assembler returns an error message indicating a failure to read drive A, or a fatal error opening a file on the A drive, then it is most likely failing to find the MOD52 or MOD834 file referenced by the assembly file. Make sure both MOD52 and MOD834 (plus any other "include" files referenced in your assembly code) are located in the ADuC root directory (where you clicked the ASM51 icon).

The ASM51.exe program can be copied/moved to another directory to prevent typing in the long path name each time. Make sure that the MOD52 or MOD834 file is also moved with the ASM51.exe program.

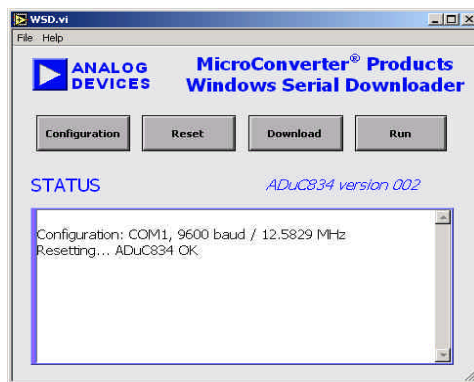
For additional details on the use of the MetaLink ASM51 assembler, please refer to the ASM51 users manual at C:\ADuC\ASM51\ASM51.pdf.

3.0 THE ADuC WINDOWS SERIAL DOWNLOADER (WSD):

The Windows Serial Downloader (WSD) is a windows software program that allows a user to serially download Intel standard Hex files as created by the ASM51 assembler to the MicroConverter while in circuit. The Intel standard hex file is downloaded into the on-chip program FLASH memory via any of the serial ports (COM1->COM4) on a standard PC. The WSD also incorporates the serial download protocols for downloading to Flash/EE data memory, setting of security bits and various RUN options.

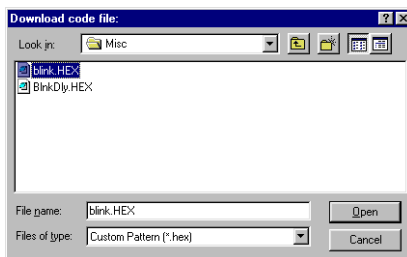
3.1 Opening the Windows Serial Downloader

1. Verify the following link connection on the ADuC834 Evaluation board.
Slide LK3 into the ON position: to enable debug/serial download mode on power-on or Reset
For details on link connection options, refer to the 'ADuC834 Evaluation Board Reference Guide' at C:\ADuC\Documentation\ADuC834\EvalDocs\834EvalGuide_C.pdf.
2. Power up the evaluation board using the power supply provided or a 9V battery (not both). Connect the evaluation board to your PC's COM1 serial port using the provided RS-232 serial port cable.
3. Press the reset button on the evaluation board. From the START menu choose Programs → ADuC → WSD. This loads the Windows Serial Downloader. The WSD executable is located at C:\ADuC\WSD\WSD.exe. The part is automatically reset. 'ADuC834 version 002' should appear above the top right corner of the Status Box as seen below.

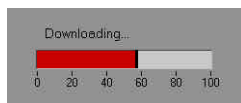


3.2 Downloading using the WSD

4. Click the Download Button. Select the file at C:\ADuC\Code\834\misc\blink.hex. Click on 'Open' to download the file.



While the file is downloading the following window will appear indicating how much of the file has been downloaded.



Once the file has been successfully downloaded the Downloading window will disappear and the Status Box will be updated with the message 'Downloading the File C:\ADuC\WSD\blink.hex...OK'

3.3 Running the Downloaded File

Running using the WSD

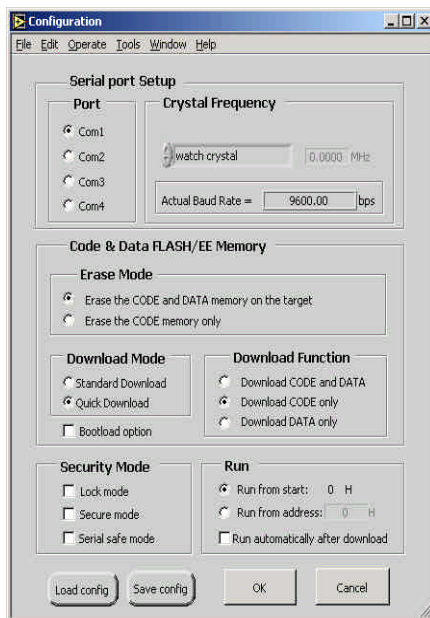
5. Click on the Run Button. The Status Box is updated with the message. "Run Program ... OK". This starts running the program from address 0000h, as can be seen by a flashing LED. To perform additional downloads, press the reset button on the Eval board (with LK3 in the ON position) and continue from step 4.

Manual Run Option

6. Press reset on the Eval board. Download as in step 4. Remove the PSEN pulldown link (LK3) on the Eval board. Press reset on the Eval board. The program starts running automatically after reset as can be seen by a flashing LED.

3.4 Additional Download/RUN Options

The MicroConverter also incorporates a serial download protocol that allows various Download/RUN options (see uC004 at C:\ADuC\Documentation\TechNotes). These options can be easily selected in the Configuration window (click on the Configuration Button) as shown below. As you can see various Erase, Download, Security, and RUN options exist here.



Run Automatically after Download

7. Slide link LK3 to the ON position. Press reset on the Eval board. Click on the Configuration button. Click on the box for 'Run automatically after download' as shown in the configuration window above. Click on OK. Download as in step 4. The program starts running automatically after download as can be seen by a flashing LED.

Run from Address other than 0000h

8. The program can be run from an address other than 0000h. To do this make sure that LK3 is in the ON position. Press reset on the Eval board. Click on the configure menu and deselect the Run automatically after download option. Download as in step 4. Choose File → Run from Address. Enter address 57 in the window as shown below and click on OK. Click on the Run Button. The Status Box is updated with the line 'Run Program from Address 000057h...OK'

NOTE: Use of the Serial Port:

The WSD only uses the UART (RS232) serial port when

- Resetting the device
- Downloading to the device
- Sending the Run command to the device

Therefore the WSD does not have to be closed before launching the Debugger/WASP/Hyperterminal or any other program, which uses the UART (RS232) serial port.

However if another program which uses the UART (RS232) serial is open then the WSD will not be able to communicate with the MicroConverter until the UART is released by disconnecting/closing the other application.

4.0 THE ADuC SERIAL PORT DEBUGGER :

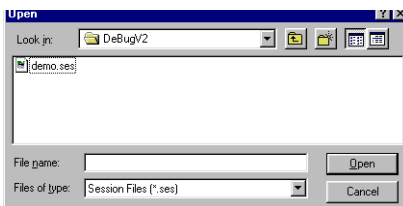
The Debugger is a Windows application that allows the user to debug code execution **on silicon** using the MicroConverter UART (RS232) serial port. The debugger provides access to all on-chip peripherals during a typical debug session as well as single-step and breakpoint code execution control.

4.1 Opening the ADuC DeBugger

1. Verify the following link connections on your ADuC834 evaluation board:
Slide LK3 into the ON position to enable debug/serial download mode on power-on or Reset
Slide LK4 into the ON position to externally short AIN1 to AIN2
Slide LK5 into the ON position to bias AIN2 to 2.5V common mode voltage
Slide LK7 to position A to connect REFIN- to AGND
Slide LK8 to position A to connect REFIN+ to 2.5V external voltage reference
Slide LK9 to position A to configure interface to external data RAM
For details on link connection options, refer to the ‘ADuC834 Evaluation Board Reference Guide’ at C:\ADuC\Documentation\ADuC834\EvalDocs\834EvalGuide_C.pdf.
2. Power up the evaluation board using the provided power supply or a 9V battery (not both). Connect the evaluation board to your PC’s COM1 serial port using the provided RS-232 cable. Press the RESET button on the evaluation board.
3. From the START menu on your PC choose Programs→ ADuC→ DeBugV2 (C:\ADuC\DeBugV2\aduc.exe). The launches the debugger. Click “OK” to get past the opening dialogue box,

4.2 Creating/Opening a Session File

4. If the automatic wizard launches then press cancel to avoid using the wizard during this tutorial session. Select FILE → Open from the pull down menu bar and select ‘demo.ses’ as shown below.




Notes:

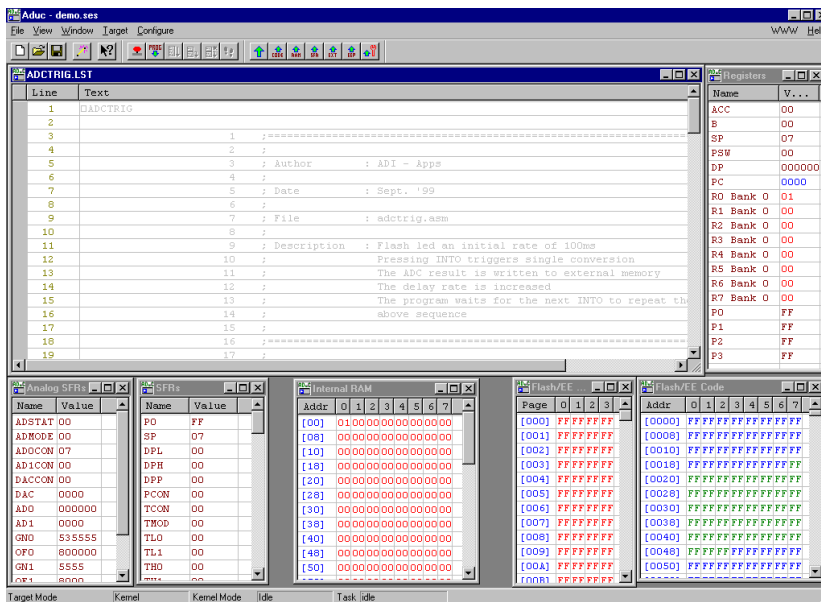
- Instead of choosing ‘Cancel’ you can load the session manually by clicking on ‘Next’ in the ‘Session Wizard’. If the session wizard does not automatically load up, click on the ‘Wizard’ button in the ‘Session’ window or choose ‘Run Wizard Session’ under the ‘Configure’ menu. Type the path of the file you wish to debug in the ‘Files’ menu or using the ‘Browse’ button to locate the file. For this tutorial choose the file C:\ADuC\DeBugV2\adctrig.lst. For all the remaining session wizard steps, simply click on the ‘Next’ button, to accept the default settings. Click the “Finish” button at the final session wizard screen.

- The **adctrig.asm** program toggles the LED at an initial rate of 100ms. Pressing the INTO button triggers a single conversion. The ADC result is written to external data memory, and the delay is increased. The program starts blinking the LED again at the slower rate. Pressing the INTO button again causes another ADC conversion.
- Some information will flash on the screen and the bar at the bottom of the screen will change:

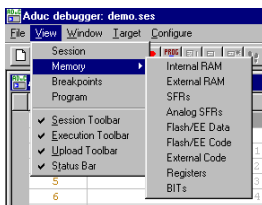
From	Target Mode:	Unknown	Kernel Mode:	Unknown	Task:	Unknown
to	Target Mode:	Kernel	Kernel Mode:	Idle	Task:	idle

If you *don't* see the above information, then click the *reset target* button () on the toolbar.


- At this stage, you should see multiple windows within the ADuC application as shown below: The program listing window (labeled “adctrig.lst”) on the top half of the screen. The Analog SFRs, General SFRs, internal Flash/EE Data memory and internal Flash/EE code memory, are visible in the bottom half of the screen.



- To view additional windows, from the “View” pull-down menu, choose “Memory” → followed by the required window. The selected window will pop-up on the front screen.

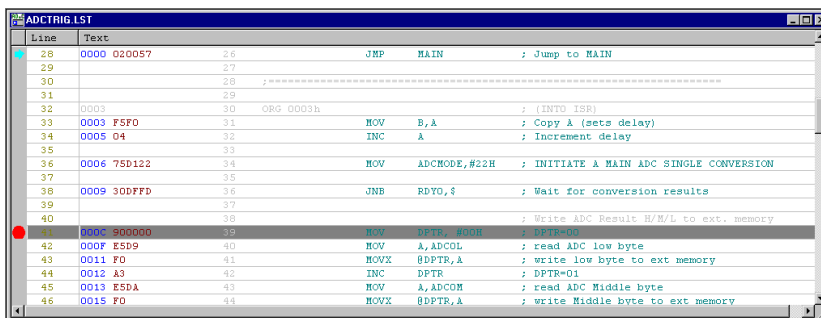



4.3 Downloading to the Chip

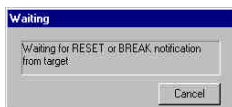
- Click the *download program* button () on the toolbar, or choose “Download Program” from the “Target” pull-down menu. This downloads the program to the chip. Note how the program op-codes now appear in the Flash/EE Code window.

4.4 DeBugging Code

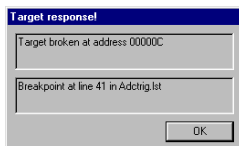
- In the program window (“adctrig.lst”), double click on the line (line 41, address 000Ch – MOV DPTR, #00H) as shown below. This sets a breakpoint at address 000Ch in the External Interrupt 0 ISR. Notice that the breakpoint is indicated by a large red dot to the left of the line.




- Press the *run program* button () on the toolbar, or choose “Run downloaded Program” from the “Target” pull-down menu. This runs the program from the beginning. You will see the red LED on the board flashing and the following dialog box on the screen to confirm that the program is running.



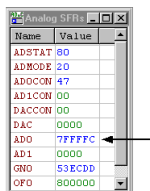
- Press INT0 button to trigger an external interrupt. A breakpoint was set in this Interrupt Service Routine. Once the INT0 button is pressed the program hits this breakpoint, code execution is halted and you will see the following message appear indicating that the chip has hit the breakpoint.



Press “OK”, and the debugger will upload information from the chip.

Note: If you wish to change what is automatically uploaded from the chip, press the *setup uploads* button () on the toolbar, and select what you want to be automatically uploaded.

- In the “Analog SFRs” window, note that ADC0 representing the 3 byte (24-bit) primary ADC conversion result (SFRs ADC0H/M/L) has been updated to reflect a conversion on an externally shorted input.




Note:

- ADC Code 800000h represents zero differential input in bipolar mode.
- To view SFR values in binary or decimal, simply right-click inside the SFRs window.
- Data values colored blue or green, in the SFR window (or any other memory window) indicate, that the data was just uploaded and hence the values accurately represent the values currently on the chip. If the data is blue then the data has changed since the last upload. If the data is green then the data has not changed since the last upload.

Data values colored red, in the SFR window (or any other memory window) indicate, that the data was not just uploaded. Hence the values may have changed since it was last uploaded.



To upload (or change) the value in a particular SFR you can double click on the particular SFR and then READ or WRITE to this SFR by clicking on the READ or WRITE buttons. In the case of a WRITE the value to be written to the SFR should firstly be written into the 'New Value' box.

- Click the *resume* button () on the toolbar, or choose "Resume" from the "Target" pull-down menu. Click "OK" at dialogue box shown below.





Notice that the red LED now continues to toggle at a slower rate. Press INT0 again to trigger another conversion, note the ADC conversion result in ADC0 update as you acknowledge the breakpoint.

Note: You can turn off the above dialogue boxes in "System settings" in the "Configure" pull-down menu.

- You can resume again () , or step through the code using the *single-step* button () on the toolbar, or set additional breakpoints by double-clicking on other lines of code. To clear a breakpoint, simply double-click on the line again.

4.5 Saving/Closing a Session File

- To save your session settings, click the *save session* button () on the toolbar, or choose "Save" from the "File" pull-down menu, and select the desired session file name and location.
- To close the session, choose "Close" from the "File" pull-down menu.
- To open a pre-existing session file, click the *open session* button () on the toolbar, or choose "Open" from the "File" pull-down menu, and select the file you wish to open.

For more information on the use of the Debugger, use the "Help" options (in the top right corner) within the application.

5.0 THE ADuC SINGLE PIN EMULATOR :

The ADuC single pin emulation system provided as part of this ADuC834 QuickStart Development System is called the ‘Advanced Circuit Emulator’ or ACE. Unlike the serial port, assembly code debugger presented in the last chapter, the ACE system facilitates ADuC834 code debug using a non-intrusive single pin path provided by LK2 on the ADuC834 board and also supports **full C-Source debug**.

The single pin emulation system is driven from an identical GUI as the debugger described in Chapter 4, although there are additional features supported via the various pull-down tool-bar menus. The following are the main additional features provided by the Single Pin emulation system :

- True, in-circuit emulation, this effectively frees up the ADuC834 serial port.
- Full C and assembly source debug, including breakpoint and single step capability.
- Simple connection between emulator and target, emulation cable connects to LK2 on board.
- On screen oscilloscope displays analog waveforms, facilitates graphing ADC results.

The complete user manual for the ACE system is contained in the ACE Debug folder at C:\ADuC\ACE

This chapter describes the use of the ACE and walks the reader through a typical ADuC834 C-Source debug session.

5.1 Connecting the ACE System

1. Verify the following link connections on your ADuC834 evaluation board:
Slide LK3 into the ON position to enable debug/serial download mode on power-on or Reset
Slide LK4 into the ON position to externally short AIN1 to AIN2
Slide LK5 into the ON position to bias AIN2 to 2.5V common mode voltage
Slide LK7 to position A to connect REFIN- to AGND
Slide LK8 to position A to connect REFIN+ to 2.5V external voltage reference
Slide LK9 to position A to configure interface to external data RAM
For details on link connection options, refer to the ‘ADuC834 Evaluation Board Reference Guide’ at C:\ADuC\Documentation\ADuC834\EvalDocs\834EvalGuide_C.pdf.
2. Connect the serial port cable from your PC comm. port directly into the 9-way mating socket on the ACE emulator pod. (This tutorial chapter assumes the serial port is connected to COM1 on your PC.)
3. Connect the emulation cable to the ACE emulator. Note that this black emulation cable is polarized, and that the red dot on the cable connector should align with the red mark on the socket on the front of the emulator. Connect the emulation cable to LK2 on the ADuC834 evaluation board. LK2 is a 2 pin header containing EA and digital ground (GND) with a polarized friction lock shroud as shown in below. The EA line is used as the single wire emulation path to the ADuC834 target during the debug session.



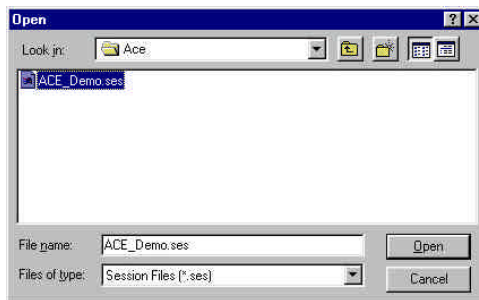
4. Plug in the power supply provided in this kit and connect it to the power socket on the ACE. The LED on the front of the ACE should briefly flash red then orange.
5. Power cycle or press reset on the ADuC834 evaluation board using the 9V battery provided

5.2 Launching the ACE Emulator Software

6. From the START menu on your PC choose Programs → ADuC → ACE Emulator (C:\ADuC\ACE\ace.exe). This launches the Emulator. Click “OK” to get past the opening dialogue box.

5.3 Opening the Session File

7. If the automatic wizard launches then press cancel to avoid using the wizard during this tutorial session. Select FILE → Open from the pull down menu bar and select ‘ACE_demo.ses’ as shown below.



Notes:

- Instead of choosing ‘Cancel’ you can load the session manually by clicking on ‘Next’ in the ‘Session Wizard’. If the session wizard does not automatically load up, click on the ‘Wizard’ button in the ‘Session’ window or choose ‘Run Wizard Session’ under the ‘Configure’ menu. Type the path of the file you wish to debug in the ‘Files’ menu or using the ‘Browse’ button to locate the file. For this tutorial choose the file C:\ADuC\ACE\demo834 For all the remaining session wizard steps, simply click on the ‘Next’ button, to accept the default settings. Click the “Finish” button at the final session wizard screen.
- The C-Source file selected via the browse option does not have any suffix such as .lst or .obj.
- The **demo834.c** program continuously toggles the LED being driven from P3.4 on the ADuC834 Application Board. Each time the LED toggles the DAC word is incremented and the DAC output updated. Pressing the external INTO button on the board triggers a single conversion on the primary ADC channel. In either mode of operation the DAC and ADC Data bytes are written at 9600 Baud to the ADuC834 UART. After each interrupt the rate at which the led toggles is decreased.
- The demo834 C code example used in this tutorial can be modified and recompiled at any time using the evaluation copy of the Keil compiler included on the QuickStart CD. A tech-note describing the compilation process (uC002) is included in the documentation folder of the CD.


- Some information will flash on the screen and the bar at the bottom of the screen will change:

From

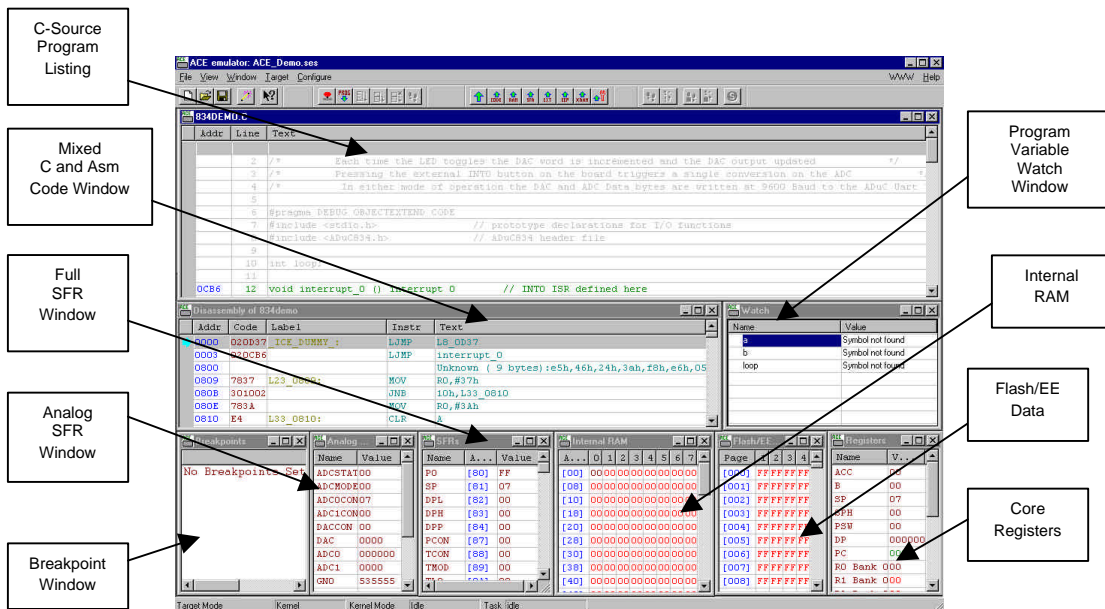


to

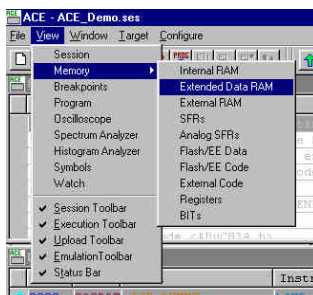


If you *don't* see the above information, then click the *reset target* button () on the toolbar.


- At this stage, you should see multiple windows within the ADuC application as shown below:



- To view additional windows, from the “View” pull-down menu, choose “Memory” → followed by the required window. The selected window will pop-up on the front screen.

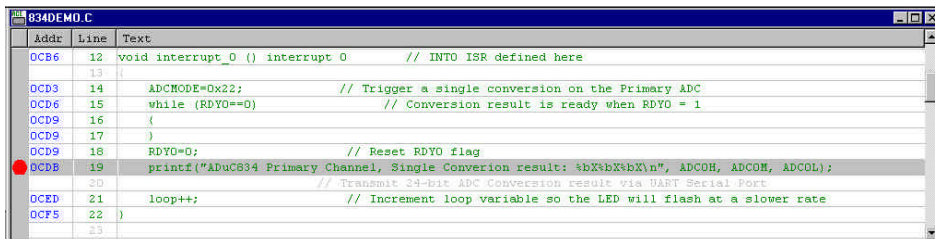


5.4 Downloading to the Chip


- Click the *download program* button () on the toolbar, or choose “Download Program” from the “Target” pull-down menu. This downloads the program to the chip.

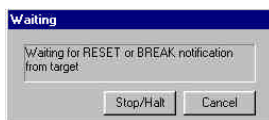
5.5 DeBugging C-Source Code

- In the program window (“demo834.c”), double click on the line (line 19, address 0CDBh – printf) as shown below. This sets a breakpoint at the loop++ instruction, in the External Interrupt 0 ISR. Notice that the breakpoint is indicated by a large red dot to the left of the line.



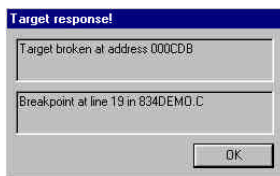
Addr	Line	Text
0CB6	12	void interrupt_0 () interrupt 0 // INTO ISR defined here
0CB7	13	{
0CD3	14	ADCMODE=0x22; // Trigger a single conversion on the Primary ADC
0CD6	15	while (RDY0==0) // Conversion result is ready when RDY0 = 1
0CD9	16	{
0CD9	17	}
0CD9	18	RDY0=0; // Reset RDY0 flag
0CDB	19	printf("ADuC834 Primary Channel, Single Conversion result: %bX%bX%bX\n", ADC0H, ADC0M, ADC0L);
0CE0	20	// Transmit 24-bit ADC Conversion result via UART Serial Port
0CED	21	loop++; // Increment loop variable so the LED will flash at a slower rate
0CF5	22	}
	23	


- Press the *run program* button () on the toolbar, or choose “Run downloaded Program” from the “Target” pull-down menu. This runs the program from the beginning. You will see the red LED on the board flashing and the following dialog box on the screen to confirm that the program is running.



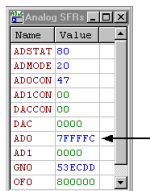
The following points should be noted :

- As shown in the ‘waiting dialogue box, the 834demo.c program can now be halted at any time using the ‘Stop/Halt’ button available on the dialogue box. Alternatively you can click on the ‘Stop/Halt’ button on the top toolbar. This feature is not available using the serial port debugger.
 - The 834demo.c program continuously transmits status/conversion data results via the ADuC834 UART using the ‘printf’ statement during this debug session. The UART serial port can be monitored by connecting to the 9-way D-type serial port connector on the ADuC834 evaluation board. Again, this would not be possible using the serial port debugger where the serial port itself is used as the debug path.
- Press INT0 button on the ADuC834 evaluation to trigger an external interrupt. In (11) above a breakpoint was set in this Interrupt Service Routine, once the INT0 button is pressed the program hits this breakpoint, code execution is halted and you will see the following message appear indicating that the chip has hit the breakpoint. Press “OK”, and the ACE will upload information from the chip



Note: If you wish to change what is automatically uploaded from the chip, press the *setup uploads* button () on the toolbar, and select what you want to be automatically uploaded.

14. In the “Analog SFRs” window, note that ADC0 representing the 3 byte (24-bit) primary ADC conversion result (SFRs ADC0H/M/L) has been updated to reflect a conversion on an externally shorted input.



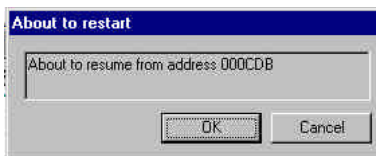
Note:

- ADC Code 800000h represents zero differential input in bipolar mode.
- To view SFR values in binary or decimal, simply right-click inside the SFRs window.
- Data values colored blue or green, in the SFR window (or any other memory window) indicate, that the data was just uploaded and hence the values accurately represent the values currently on the chip. If the data is blue then the data has changed since the last upload. If the data is green then the data has not changed since the last upload.

Data values colored red, in the SFR window (or any other memory window) indicate, that the data was not just uploaded. Hence the values may have changed since it was last uploaded.

To upload (or change) the value in a particular SFR you can double click on the particular SFR and then READ or WRITE to this SFR by clicking on the READ or WRITE buttons. In the case of a WRITE the value to be written to the SFR should firstly be written into the ‘New Value’ box.

15. Click the *resume* button (⏮) on the toolbar, or choose “Resume” from the “Target” pull-down menu. Click “OK” at dialogue box shown below.



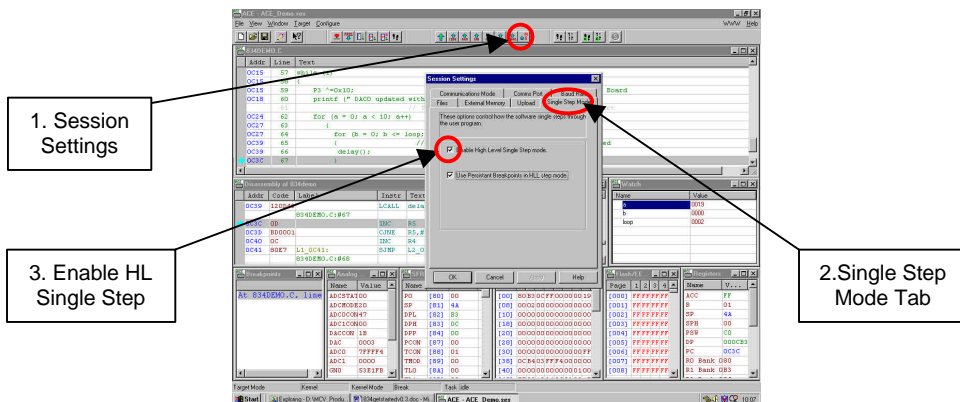
Notice that the red LED now continues to toggle at a slower rate. Press INT0 again to trigger another conversion, note the ADC conversion result in ADC0 update as you acknowledge the breakpoint.

Note: You can turn off the above dialogue boxes in “System settings” in the “Configure” pull-down menu.

16. You can resume again (⏮) or set additional breakpoints by double-clicking on other lines of code. To clear a breakpoint, simply double-click on the line again.
17. If you choose to *single-step* using (⏮) or <F11> or indeed the animate (multiple single steps) options using (⏮) buttons from the top toolbar, then you should note that these single step options will only implement low level assembly single steps in this emulator configuration mode. Single stepping in C and Assembly is now described briefly in the next section.

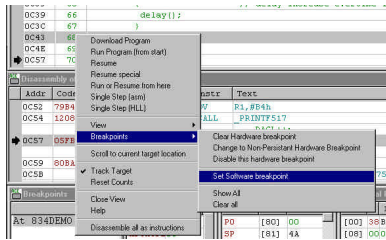
5.6 Single Stepping in C


18. To enable high level single step mode, halt the current program execution by using the ‘Stop/Halt’ (⏏) button available on the dialogue box. Alternatively you can click on the ‘Stop/Halt’ button on the top toolbar.
19. Next, enable high level single stepping by clicking on the *session settings* button (⚙) on the top toolbar. Select the *Single Step Mode* tab from the resultant dialogue box and tick the box shown below to enable *High Level Single Step Mode*.



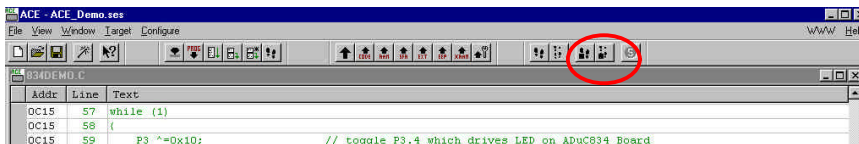
IMPORTANT NOTE :



- Once High-Level Single Step Mode is enabled, a single step breakpoint will automatically be set by the system on each C-Source line. Because invisible breakpoints are set automatically on each C-line, the RUN or RESUME commands will only run or resume code execution to the start of the next C instruction.
 - To run code in High Level Single Step mode the code animation option can be used as described below. Animation mode facilitates code execution by executing C instructions as continuous sequential multiple single step instructions.
 - Software breakpoints can be set in animation mode to automatically halt code execution at a specific point in target execution.
20. First, click the *download program* button (⬇) on the toolbar, or choose “Download Program” from the “Target” pull-down menu. This downloads the program to the chip. The program needs to be re-downloaded at this point because the system has reconfigured the code for high level single step mode.
 21. In the 834Demo.c program listing window, scroll down to line 68 and right click on this line. Select *Breakpoints* → *Set Software breakpoint*. Notice that a software breakpoint is indicated by a large blue dot to the left of the line.

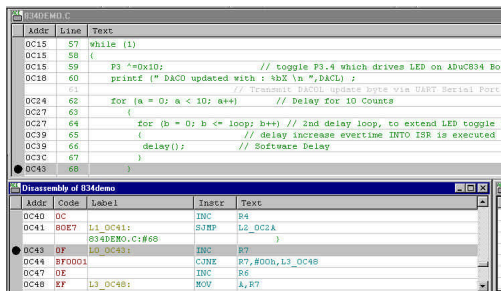



22. Click on the C animate () button from the top toolbar. The 834demo.c program starts executing line by line and halts when it reaches the software breakpoint set at line 68.

- (Note that both the C-animate and C-Step icons reside to the right hand side of the assembly animate and single step icons on the top tool-bar. The C single step and animate controls are easily identified by the small 'c' character embedded in the icons.

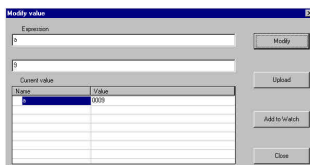






23. Notice that when the program stops at this C instruction (i.e. the end of the while loop), the assembly instructions comprising the C-Source instruction that will now be executed can be viewed directly in the disassembly window.
24. Selecting the Assembly single step or assembly animate ( or ) buttons from the top toolbar will allow you step *INTO* the C instruction which results in this case in an *INC* instruction followed by a *CJNE* instruction.



25. When single stepping *into* a C instruction, you will see the present location of program execution (indicated by blue arrow to the left of the source line) remains unchanged in the 834demo.c window while in the disassembly window, you will see the program execution sequence through the assembly instructions that comprise the C-Source line of code.
26. Selecting the C-Source single step () button will allow you to step out of the current C-source instruction onto the next C-instruction. Try single stepping in C through the core loop (for a=1 to 10) in the 834demo.c program.

27. You should also note also the value of the loop variable 'a' is automatically updated in the watch window as you step into the C-Source instruction.
28. Notice how the watch window updates automatically as the loop variables are incremented. Notice also in the disassembly window how the variables 'a' and 'b' are obviously stored in R5 and R7 respectively, the R5 and R7 registers can also be seen to update automatically in the internal RAM and Registers windows after the C-single steps.
29. If you wish to avoid having to single step in C through the 'for a = 0 to 10' loop. You can simply modify the value of the 'a' variable in the watch window. Right click on the 'a' variable in the watch window and select 'modify selected' from the pull-down menu. Enter a 'new' value of 9 and press modify in the resultant window as shown below. Finally, click 'close'.



30. Now, notice as you C-step through the 'for a = 0 to 10' loop the code steps out of the loop because 'a' is now no longer less than 10 and therefore the code will step onto the printf statement outside the loop.
31. You can now continue to C- single step () or C-animate () through the code. If the interrupt button is pressed on the 834 evaluation board, the code will vector to the INT0 ISR and continue to execute from here.
32. To save your session settings, click the *save session* button () on the toolbar, or choose "Save" from the "File" pull-down menu, and select the desired session file name and location.
33. To close the session, choose "Close" from the "File" pull-down menu.
34. To open a pre-existing session file, click the *open session* button () on the toolbar, or choose "Open" from the "File" pull-down menu, and select the file you wish to open.
35. This tutorial has given a brief introduction to some of the basic single-pin emulator functions. Other features available on the ACE font-end eg. Oscilloscope etc are described in more detail through the on-line help menus, use the "Help" options (in the top right corner) within the application.

6.0 THE ADuC SIMULATOR:

The ADSIM Simulator is a Windows application that fully simulates the functionality of the ADuC812, ADuC816 and the ADuC824, including ADC and DAC peripherals. The simulator provides an easy to use, intuitive interface to the MicroConverter functionality and integrates many standard debug features including multiple breakpoints, single stepping and code execution trace capability.

This tool can be used both as a tutorial guide to the part as well as an efficient way to prove code functionality before moving to a hardware platform. The Simulator accurately simulates the ADC function including conversion timing, auto-calibration and on-chip temperature sensor channel.

Currently the simulator does not support the ADuC834. However because the ADuC834 is so similar to the ADuC824 many of the features of the ADuC834 can be simulated by simulating an ADuC824 instead. The larger memory of the ADuC834 does not matter as the ADuC824 simulator can simulate up to 64kBytes of code. Similarly external XRAM can be used in place of the internal XRAM. The main differences lie around the extended stack pointer, dual data pointers, PWM and ULOAD mode which are not simulated by the ADuC824 simulator. This tutorial chapter will discuss the ADuC824 simulator.

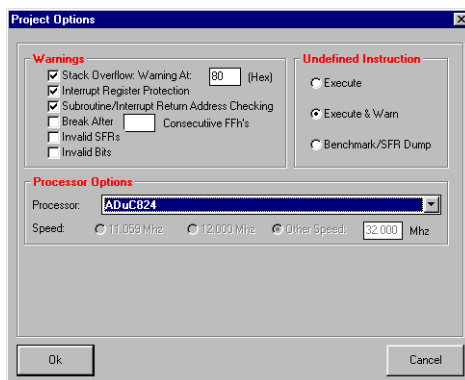
6.1 Opening the ADuC Simulator

1. From the START menu choose Programs→ADuC→ADSim (or double click on the file C:\ADuC\ADSim\ADSim.exe). This loads the Analog Devices Simulator. A blank simulation session should be opened if the simulator was not used before. If the simulator has been used before then some windows may be opened in this simulation. Close any windows that are opened.

6.2 Selecting the required MicroConverter

2. Under the configuration menu choose ‘Project Options’. The required MicroConverter can then be selected in the “Processor Options” box. Choose the ADuC824 to select the ADuC824 MicroConverter as shown below. Click on ‘OK’ to select the ADuC824.

Note: The ADuC824 crystal frequency does not have to be selected as the simulator assumes a 32.768Hz input crystal. Note the different options regarding warnings available to the user. Again if an older version of the simulator was used before then these warnings might be different.

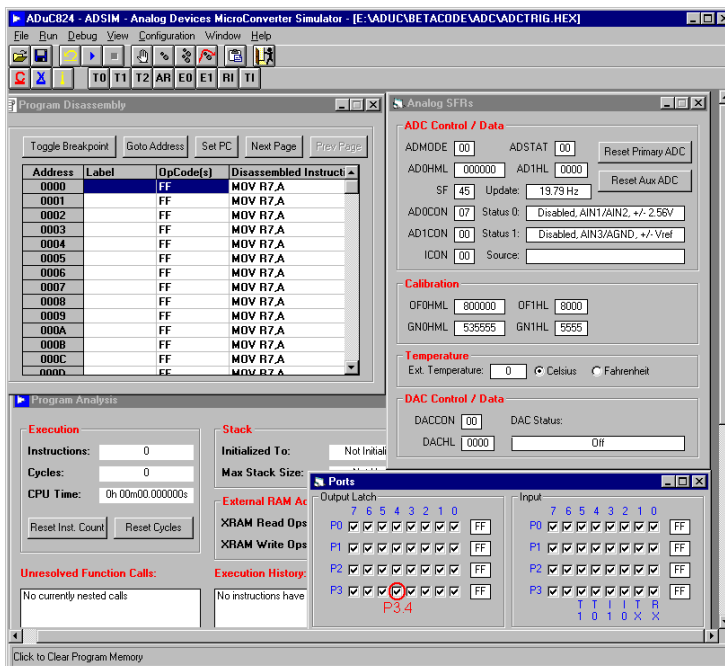



After selecting the ADuC824 device make sure that the top of the simulator window (caption) starts with the string “ADuC824”.

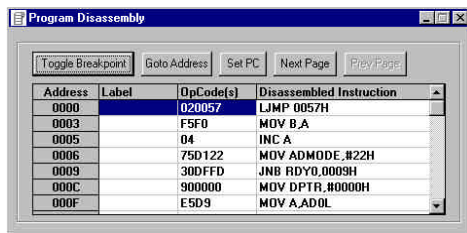
6.3 Starting a Simulation

3. From the View menu select the Program Disassembly option to open the Program Disassembly window as shown below. Since no program has yet being loaded then the program will appear in the erased state (all opcodes are FFh).
4. Open the following windows under the view window.
 - Analog SFRs (under the View→ SFR menu)
 - I/O Ports
 - Program Analysis

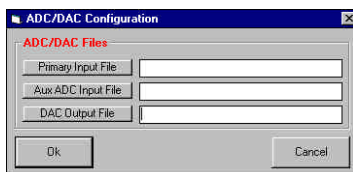
Position the 4 windows (including the Program Disassembly window) as shown below.





5. Under the File Menu select 'Open Intel HEX File' or click on the 'Open Hex File' button . In the 'Open' window that appears open the file adctrig.hex which appears at C:\ADuC\Code\834\ADC\adctrig.hex. The 'Program Disassembly' window is updated with the Opcodes and disassembled instructions of the hex file as shown below. The caption should now include the string '[C:\ADuC\Code\834\ADC\Adctrig.hex]'.

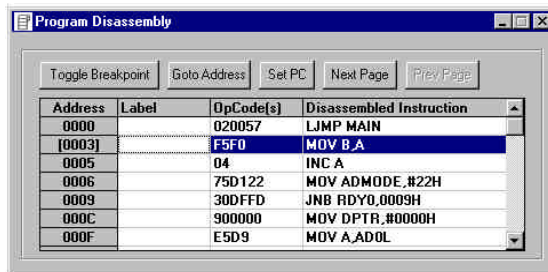




- Select 'Import Map File' under the File menu. In the 'Open' window select the .lst file 'adctrig.lst'. This updates the simulator screen with the function names and/or labels from your program instead of the addresses of these functions/labels. The first line of code LJMP 0057H should change to LJMP MAIN
- Click on the 'ADC/DAC/SPI I/O Files' option under the Configuration menu. The following window should be opened. By clicking on the 'Primary Input File' choose the following input file 'C:\ADuC\ADSim\IOData\24bit.adc'. The other I/O files that exist in this directory can be loaded in a similar manner.

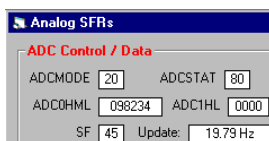


6.4 Running/Simulating a Program

- Press the 'Run' Button . P3.4 (representing the LED on the Eval board) should start toggling. The P3.4 output is simulated in the ports window as marked in the Simulator diagram on the previous page.
Note: P3.4 may toggle very slowly. This is because P3.4 toggles every simulated 100ms. The simulation speed can be changed in the Simulator Configuration window which we will discuss in section 5.5. To see how quickly the time is being simulated keep an eye on the Program Analysis window.
Note: The 'Program Analysis' window can be a very useful tool for debugging software. Notice the instructions and the time being counted in the Program Analysis window. Press the stop button . Notice the Execution History in the Program Analysis window being updated.
- Using the scroll bar on the Program Disassembly window click on the address 0003. The whole line should highlight. Click on the button 'Toggle Breakpoint' to set a breakpoint at this location. The breakpoint has been set correctly if square brackets appear on either side of the address. Once the breakpoint has been set the present instruction is re-highlighted. Press the run button again to resume the program. The program should start running as before.

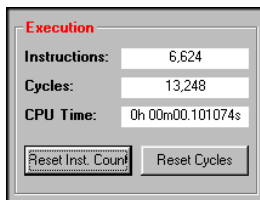



10. Press the  button to simulate an External Interrupt 0. This causes the program to vector to address 0003h. The program will automatically stop at the breakpoint that has been set at this location. Clear the breakpoint at address 0003h by clicking on the address and pressing 'Toggle Breakpoint' again. The square brackets should disappear.
11. Press the Single Step Button  to execute the next instruction. Notice the Program Analysis window being updated. Press the single step button again. Pressing the single step button for a third time executes the MOV ADCMODE, #22h instruction. i.e. initiating a single conversion. In the Program Analysis window press the 'Reset Inst Count' and 'Reset Cycles' buttons to clear the execution box.
12. Set another Breakpoint at 000Ch (after the ADC result is ready) by clicking on the address 000Ch and then clicking the Toggle Breakpoint button. Notice that the ADC0HML register(s) in the Analog SFRs window contains the data 000000h. Press the Run button again. Wait for the program to stop at the breakpoint at 000Ch.
13. Notice that the ADC0HML register is updated with the value 098234 as shown below. This value is taken from the primary ADC input file '24bit.adc'.



14. Notice that the Execution box in the Program Analysis window has now being updated as shown below. The time taken to arrive at the breakpoint is shown in the CPU Time box. This means that the RDY0 flag was not set until 0.101s after the ADC conversion was initiated. During this time the JNB RDY0,0009h instruction was carried out 6,624 times. This CPU Time corresponds to an update frequency of 19.79Hz. This update frequency is also calculated in the Analog SFRs window beside the SF register as shown above.

Note: Because the ADuC834 uses ADC chopping a single conversion is actually two conversions. Hence, as shown below the conversion actually takes two conversions to execute. In continuous conversion mode only the very first ADC conversion takes the double conversion time. All subsequent conversions will take only a single conversion time. This is all simulated correctly on the simulator.



15. Press the run button again to resume the program. P3.4 (LED) should start flashing again. Press the  button again to simulate another external interrupt 0. The program will stop at the breakpoint at address 000Ch. The next value in the primary ADC input file will now appear in the ADC0HML box.

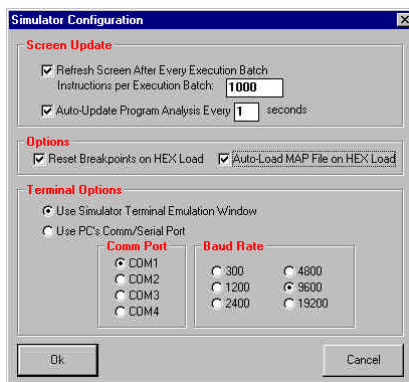


6.5 Simulator Configuration Menu



16. Different simulator settings can be chosen in the Simulator Configuration window under the Configuration menu. You may have noticed that the simulation was very slow. This is due to the fact that only 1 instruction is executed per batch. 100 to 1000 instructions per batch is recommended for quicker simulations.

An option to 'Auto-Load MAP file on a Hex Load' exists. This option imports the labels and function names from the .lst file automatically after loading the Hex file. If the .lst file does not exist in the same directory as the .hex file then the map file will have to be imported manually.

The following options are recommended. Once changed these settings will be saved for all future simulations.



6.6 Saving/Closing/Opening a Simulation

17. To save a simulation click on the 'Save Current Simulation' option under the File menu or click on the Save icon  in the Toolbar.
18. To close the simulation, choose 'Exit' from the File menu or click on the exit program icon  in the toolbar.
19. To open a pre-existing simulation, click on the 'Open Simulation' option under the File menu.

7.0 THE ADuC WASP

The Windows Analysis Software Program (WASP) is a general application for all the MicroConverter products to analyze the noise performance of the MicroConverter. The WASP recognizes which MicroConverter PC is communicating with and automatically downloads the appropriate hex file onto the MicroConverter device. In this document we will talk only about the ADuC834 WASP.

After downloading the appropriate hex file the WASP launches the acquisition and analysis program. This allows the user to configure, control and analyze the noise performance of the Primary and Auxiliary ADCs on the ADuC834.

1. Verify the following link connections on your ADuC834 evaluation board:
Slide LK3 into the ON position to enable debug/serial download mode on power-on or Reset
Slide LK5 into the ON position to bias AIN2 to 2.5V common mode voltage
Slide LK7 to position A to connect REFIN- to AGND
Slide LK8 to position A to connect REFIN+ to 2.5V external voltage reference
For details on link connection options, refer to the 'ADuC834 Evaluation Board Reference Guide' at C:\ADuC\Documentation\ADuC834\EvalDocs\834EvalGuide_C.pdf.
2. Power up your eval board using the provided power supply or a 9V battery (not both). Connect the eval board to your PC's COM1 serial port using the provided RS-232 cable. Press the reset button on the evaluation board.
3. From the START menu choose Programs → ADuC → WASP. This loads the Windows Analysis Software Program executable file WASP.exe at C:\ADuC\WASP\WASP.exe.
4. Click the Download Button. 'ADuC834' should appear and the program start to download. A task bar indicates the download progression. A message appears to tell you when the file is downloaded. The program will autoatically run after this download.



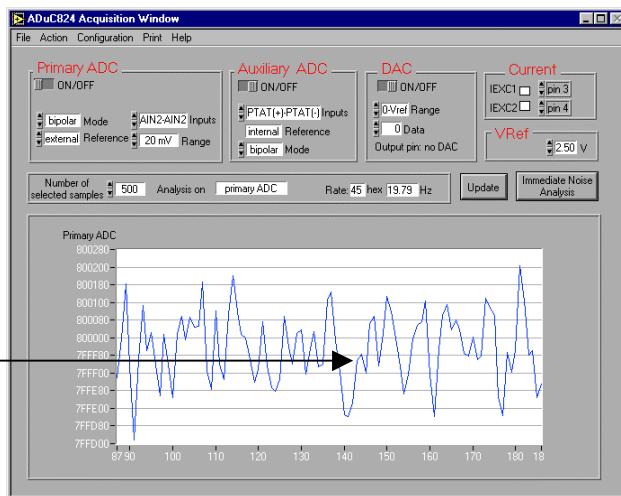
Note: The 'Next' option allows to avoid the 'Download' part. The WASP assumes that the hex file is already on the chip. Don't press reset with LK3 in the ON position. Instead slide LK3 to the OFF position, press reset and choose "ADuC834" in the box before clicking "Next". This will directly launch the Acquisition window corresponding to the ADuC834.

5. By default the ADuC834 WASP enables the Primary ADC configured as below (i.e. Primary ADC converting in bipolar mode using an external reference on the 20mV supply with internally shorted inputs Ain2 → Ain2. The Auxiliary ADC, DAC and Current sources are all disabled)

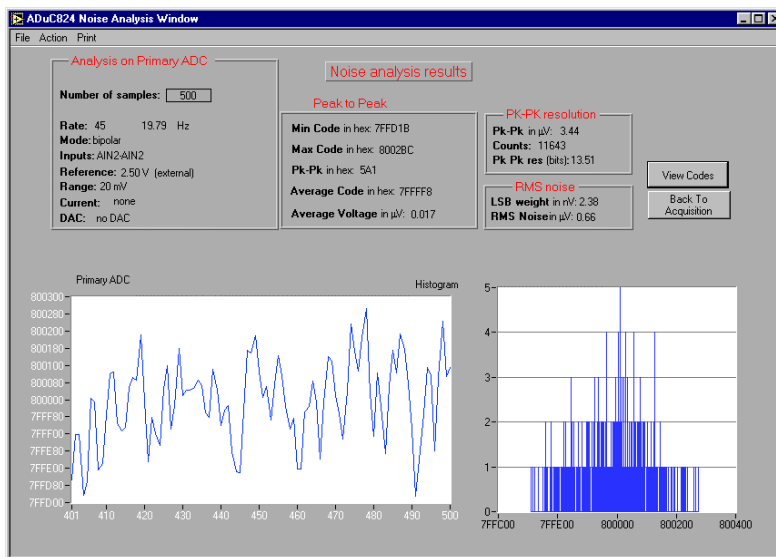
Press the 'Update' button to send this default configuration to the ADuC834 device and begin the conversions. The screen changes to configure for a single Primary ADC acquisition sequence. The results of conversion are displayed in real time.

Because the channel has an internal short then we expect ADC conversions close to 80000h. The WASP does 500 ADC conversions by default and compares the different ADC conversion results.

Primary ADC
measuring noise on
the internal short
(between
AIN2 – AIN2)



- Press 'Immediate Analysis' or wait until you acquire 500 samples to analyze the results. This plots the number of times each conversion result occurs against the conversion result itself.



Notes on Noise Analysis Window:

- The code returned should be that of Zero-Scale bipolar (800000Hex).
- By default the screen is set up to display 500 points before it begins scrolling, this display span may be changed via the 'number displayed' field.
- The noise analysis screen is as shown above. The peak-to-peak resolution (in μV and in bits) and the RMS noise (in μV) are shown on the upper right hand side panels.

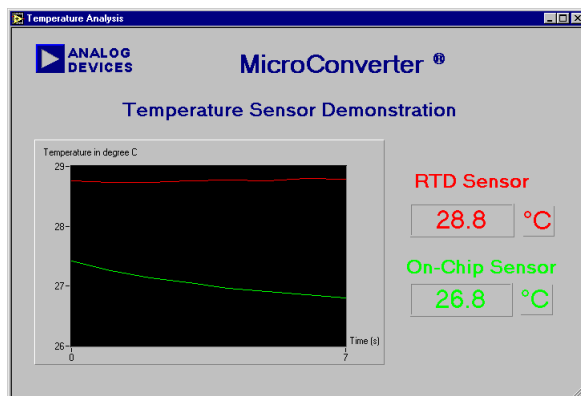
Notes on WASP:

- Select the 'Print' pull-down menu in the Acquisition window or select the Print Window option under the File menu in the Noise Analysis Window to print a hardcopy of the appropriate screen.
- To view the raw data click on the '*View Codes*' button in the Noise Analysis window. The codes can be viewed in hex or decimal formats.
- Click on the *Back to Acquisition* button in the Noise Analysis window to return to the front acquisition panel.
- The ADuC834 DAC and ADuC834 on-chip current sources can also be controlled directly from the Acquisition front panel.

8.0 THE ADuC834 TEMPERATURE SENSOR DEMO

The ADuC834 Temperature Sensor Demonstration (which is also compatible with the ADuC816 and the ADuC824) allows the user see one of the many applications for the ADuC834 in use. The ADuC834 simultaneously takes a temperature measurement from both the external RTD circuit (primary ADC) and from the on-chip temperature sensor (auxiliary ADC) and sends the temperatures to the PC via the serial port. Refer to the ADuC834 Evaluation Board Reference Guide for more details.

1. Verify the following link connections on your ADuC834 evaluation board:
Slide LK3 into the ON position to enable debug/serial download mode on power-on or Reset
Slide LK4 into the OFF position to disable the short between AIN1 and AIN2
Slide LK5 into the OFF position to disable the common mode voltage at AIN2
Slide LK7 to position B to connect REFIN- to negative reference voltage
Slide LK8 to position B to connect REFIN+ to positive reference voltage
Slide LK10 to position A to enable current from pin 3
Slide LK11 into the ON position to connect AIN1 to positive differential RTD voltage drop
Slide LK12 into the ON position to connect AIN2 to negative differential RTD voltage drop
For details on link connection options, refer to the 'ADuC834 Evaluation Board Reference Guide' at C:\ADuC\Documentation\ADuC834\EvalDocs\834EvalGuide_C.pdf.
2. Double click on the executable file C:\ADuC\Demos\TempSensor\TempSensor.exe. This loads the following window. The temperature sensor hex code is downloaded to the chip and the part is run automatically after download.



Note: A calibration feature exists for the RTD temperature sensor. By pressing the INT0 button the ADuC834 assumes the temperature to be 25°C and calibrates the temperature as such.
For improved performance remove the 0.1µF capacitors at REFIN+ and REFIN- for this demo.

9.0 ADuC834 C-LIBRARY

Because of the large program memory of the ADuC834 many user of the ADuC834 will be developing in C. To help them these users, a library of useful functions is provided as part of the QuickStart development system.

Functions are the basic building blocks of the C programming language. Use of functions, allow various tasks to be performed without the need to have a detailed knowledge of how this is achieved. The 834KEIL.lib is a library of functions, currently targeted at the Keil environment, allowing easier C programming of the ADuC834. Most of the functions have a hardware independent interface, which means that there is no need to know which register to program or how to program it.

The ADuC834 C-Library is installed at C:\ADuC_beta834\C-Lib\834. Example code using the C-Libraries are also located in this folder. A technical note to describe how to use the C-Library is located at C:\ADuC\Documentation\TechNotes\uC008.

A list of the functions provided in the C-library are:

PERIPHERAL	FUNCTION	SHORT DESCRIPTION
ADC	AdcCfg()	Configure ADCs
	AdcGo()	Start conversion(s)
	AdcRdy()	Get ADCs status
	AdcRd()	Read conversion result
	AdcGet()	Get ADC conversion
DAC	DacCfg()	Configure DAC
	DacOut()	Output a value on DAC
PLL	PlLDly()	Software delay
	PlIFst()	Select interrupt speed
	PlIWcd()	Write CD bits
	PlIRcd()	Read CD bits
TIC	TicGo()	Set and start timer
	TicHr()	Reads HOUR
	TicMin()	Reads MIN
	TicSec()	Reads SEC
	TicHth()	Reads HTHSEC
	TicVal()	Set and start interval counter
UART	UrtCfg ()	Configure UART and baud rate
	UrtBsy ()	Checks UART busy status
	Putchar	Sends character via UART
	_getkey	Gets character received by UART

Ordinary simple C source code is also provided at C:\ADuC\C-Code\834.

10.0 INSTALLED DOCUMENTATION AND CODE DIRECTORY

10.1 Installed Documentation

The installation of the ADuC834 QuickStart Development System installed documentation for both the ADuC834 and the ADuC812 in the Documentation directory located at C:\ADuC\Documentation. Three directories exist in the Documentation directory, \834, \812 and \TechNotes.

Both the \812 and \834 directories follow a similar directory structure with the directory structure for the ADuC834 discussed below. All Technical Notes for any of the MicroConverter products appear in the \TechNotes directory. Check our website for the latest tech notes.

```
C:\ADuC\Documentation\ADuC834\  
  DataSheets\  
    ADuC834_0.pdf      ADuC834 Preliminary DataSheet  
    errata834_D0.pdf   ADuC834 Errata Sheet  
    834qref0.pdf       ADuC834 Quick Reference Guide  
  EvalDocs\  
    834EvalGuide_C.pdf Reference Guide for the ADuC834 Eval Board  
    834pcb_C.pdf       Schematic and layout for the ADuC834 Eval Board  
  Other\  
    834FAQs.pdf        Frequently Asked Questions (FAQs) for the ADuC834  
    834getstartedv1.0  ADuC834 QuickStart Dev Sys Tutorial Guide
```

Any user of the QuickStart Development system should consult all these documents before proceeding to 'explore' the ADuC834.

10.2 Installed Code Directory

The installation of the ADuC834 QuickStart Development System installed a code directory for all released MicroConverter products located at C:\ADuC\Code. The \834 directory contains example code for the ADuC834 and is should be used with the ADuC834 evaluation board. Directories contained in this folder are:

```
C:\ADuC\Code\834\  
  ADC          - code examples for the ADC  
  BrdTest      - code to check if the evaluation board is working correctly  
  DAC          - code example for the DAC  
  DualDPTR     - code example to show the Dual DPTR in action  
  ExtendedSP   - code examples to show the 11bit SP in action  
  FlashEE      - code example for using the Flash/EE Data Memory  
  I2C          - code examples for I2C master and slave operation  
  Misc         - Miscellaneous MicroConverter code examples  
  PDown        - code example for ADuC834 Power Down and wakeup operation  
  PLLcon       - code example for ADuC834 PLL operation  
  PSMon        - code example for using the Power Supply Monitor  
  SPI          - code examples for SPI master and slave operation  
  TEMPSEN      - code example for the ADuC834 on-chip Temp Sensor  
  TIC          - code example for ADuC834 Timer Interval Counter  
  Timer2       - code examples for Timer 2 operation  
  UART         - code examples for the UART serial port  
  WDTimer      - code example for watchdog timer operation
```