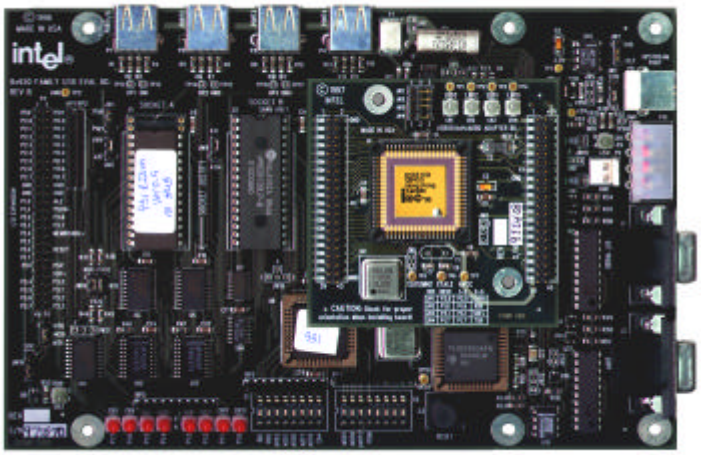
Keil Quick Start Tutorial

Version 1.0

Using the Keil Software Development Tools
with the Intel 8x930 Rev B Family Evaluation Board

& the 8x931Ax and 8x931Hx Adapter Board



Written by Robert Boys

Assisted by Shelley Potter

Keil Software Inc.

Dallas, Texas

October 20, 1997

This tutorial will instruct you on the use of the Keil Microsoft Windows based software development tools with the Intel 8x931Hx or 8x931Ax USB microcontroller. The 8x930xx based evaluation board is used with the 8x931Hx Universal Serial Bus Adapter Board (USB931AxADBD) installed. The 8x 930xx board used is the Rev B board with 4 USB ports. The Intel part number is USB93XEVALBD. The earlier version had 3 USB ports and cannot be used with the 8x 931xx daughterboard.

The 8x 931xx daughter board supports both the 8x931Hx HUB device and the 8x 931Ax. It comes with the Hx device installed. The Intel Evaluation board can then be used as a hub device and will support one upstream port and 4 downstream ports. If your peripheral will have other USB devices plugged into it, you will need a Hub device such as the Intel 8x930Hx or 8x931Hx.

The 8x931Ax and 8x931Hx contain a MCS®8051 core and programs can be written with the Keil C51 C Compiler tool set. The 8x930Ax and 8x930Hx contain a MCS®251 core and use the C251 Tool Set. A companion Quick Start supporting the 8x930xx devices and is available from www.keil.com/~market.

For more information you may visit these Web sites:

http://developer.intel.com/design/usb	http://www.usb.org
http://www.keil.com/usb	http://www.keil.com/~market

Keil is providing USB information of interest to developers. This includes:

- sample enumeration code.
- instructions on how to compile code using the Keil compiler and burn it into an Eprom that will run on the Intel USB board stand-alone.
- C code sample programs to demonstrate other functions of dScope such as the Performance Analyzer
- information on interfacing the Keil tool chain to a customer hardware board.

INDEX: This document is divided into the following sections:

1) This document is continually being updated. For the latest version check www.keil.com/appnotes or Email rboys@keil.com or phone (800) 348-8051. Check the Keil Marketing Web site www.keil.com/~market for any late developments.
2) Every effort has been made to ensure that the information contained here actually works. Please report any errors or omissions to the author at the addresses listed above.

INTRODUCTION

The Intel 8x931 USB microcontroller uses the MCS®8051 core. The Keil C51 compiler comes with an additional header to accommodate the 8x931 USB peripherals. This file is *Reg931.h* and is located in the c:\c51\inc directory. If you are using the evaluation copy of the Keil software, the directory is c:\c51eval\inc. All facilities of the 931 microcontroller are accessible. The rest of this tutorial will assume you are using the evaluation copy that came with your Intel package.

The Keil tool chain consists of the following executables located in the c:\c51eval\bin directory:

µVision	uvw51e.exe
C Compiler	c51.exe
Assembler	a51.exe
Linker	bL51.exe
dScope	dsw51.exe

µVision IDE

µVision is a Windows based front end for the C Compiler and Assembler. It was developed in the USA as was the printed manual set. Compiler, Assembler and Linker options are set with simple mouse clicks. µVision runs on Windows 3.1, 95 and NT. The Compiler, Assembler and Linker are DOS executables. They can be accessed with your favorite batch files if you prefer. This provides maximum flexibility. This Integrated Development Environment (IDE) has been expressly designed with the user in mind. A full function editor is included. All IDE functions are intuitive via pull down menus with prompted selections. An extensive Help utility is included. External executables can be run from within µVision. This includes emulator software.

C51 C Compiler for the 8051, 8x931Hx and 8x931Ax [USB]

The C51 ANSI compiler along with the A51 assembler is designed specifically for the Intel MCS®8051 microcontroller family, including the 8x931 USB. The C51 is 100% compatible with existing 8051 programs. Extensions provide access to all 8051 hardware components. Sample USB/931 code is available: www.keil.com/usb. C51 supports code banking. The compiler can be run in either DOS mode or called from the Windows based front end µVision.

A51 Macro Assembler

This Macro Assembler is included with each Compiler package or is available separately. All utilities needed to complete your project are included for all members of the 8051 family. This Assembler is DOS based or can be run from µVision which is included with every Assembler and Compiler package.

BL51 Linker - Banked Linker

The BL51 Linker/Locator is used to join relocatable object modules and library files together. The object files are created with the C51 compiler or the A51 Assembler using the Keil library files. The linking process results in absolute object modules. The BL51 linker is DOS based and can be run from DOS or through Windows using µVision. A MAP file is produced giving details of the memory structure. The object file may be specified to contain debugging information as required by simulators, debuggers and emulators. The BL51 linker can overlay important internal RAM for variables to greatly increase the speed of processing. The BL51 linker supports banking.

dScope Simulator

dScope is a software simulator. Your code can be debugged either in software on your PC or in your target hardware. When operated in conjunction with the Keil monitor installed in your target hardware, dScope becomes tScope. You have the ability to run/halt, set breakpoints, examine/change memory, view the stack, view/set peripheral information, and apply virtual external signals. dScope has a Performance Analysis feature to ensure your code runs efficiently. dScope has a disassembler/assembler allowing you to make changes in your code without recompiling.

Evaluation Version of the Keil Tool Set

The evaluation version of the Keil tool set is restricted to a 2K code size and the code must be located at 0x4000. Useful object code is produced. Other than these restrictions, the tool set works exactly as the full version does. This allows you to fully evaluate the features and power of Keil products. The full version has no restrictions and is fully ANSI compliant.

ABOUT THIS TUTORIAL

This tutorial consists of several sessions illustrating the features of the Keil Tool Set. The C code program shown in Figure 6 will be compiled and executed using various features of the Keil C51 Tool Set. The focus is on using Keil's µVision integrated development environment. It explains how to create and build a new project using a source code example provided in this tutorial.

The following steps below show you how to configure the tools, compile and link the code, and invoke the debugger, dScope. Before you begin this tutorial, you need the Keil software development tools. To get them do one of the following:

- Locate the evaluation version of the Keil software development tools that was included with the USB 8x931 peripheral development kit. This contains the example code for this tutorial.
- Download the latest version of the software from the Keil web site (www.keil.com).
- Purchase a complete package from Keil Software.

STARTING THE TUTORIAL

Installing the Keil Software

Install the Keil Software development tools by completing the following steps:

1) Insert your installation diskette number 1 in the **A:** drive.
2) In the Program Manager select File/Run… (for Windows 95 select Start/Run…)
3) Type A:\Setup at the Command Line Prompt.
4) Press OK.
5) Follow the on-screen prompts.

After you install the Keil development tools to Windows 3.11, a Keil program group or icons appear on your desktop. (the Windows 95 version is shown in Figure 1). Recall dScope is the Keil Simulator and µVision is the Keil Windows based user interface (or IDE = Integrated Development Environment).

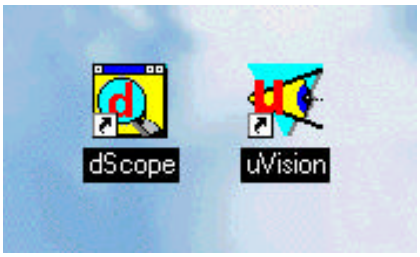


FIGURE 1

For Windows 95 you may have to install the two icons manually as follows. If the two icons are visible in a small window, drag them onto the desktop. If they are not visible, follow these instructions:

1) right mouse click while in the Desktop
2) select New/Shortcut
3) the executable file you want to enter as the Command Line is *c:\c51eval\bin\uvw51e.exe*
4) the suggested name is µVision.
5) repeat this process for *c:\c51eval\bin\dsw51.exe* and name it dScope.

µVision creates the file uvw51e.ini in your c:\windows directory the first time it is started. dScope creates dsw51.ini. Settings are saved in these files for your next session. If you re-install the Keil Tools, you may want to erase these files to obtain a clean install.

Starting the Software:

Double-click on the µVision icon to start the user interface. The compiler, assembler, linker and dScope will be called from within µVision in this tutorial. After you invoke µVision, the window shown in Figure 2 appears. From this window, you can create projects, edit files, configure the tool, assemble, link, and invoke the debugger.

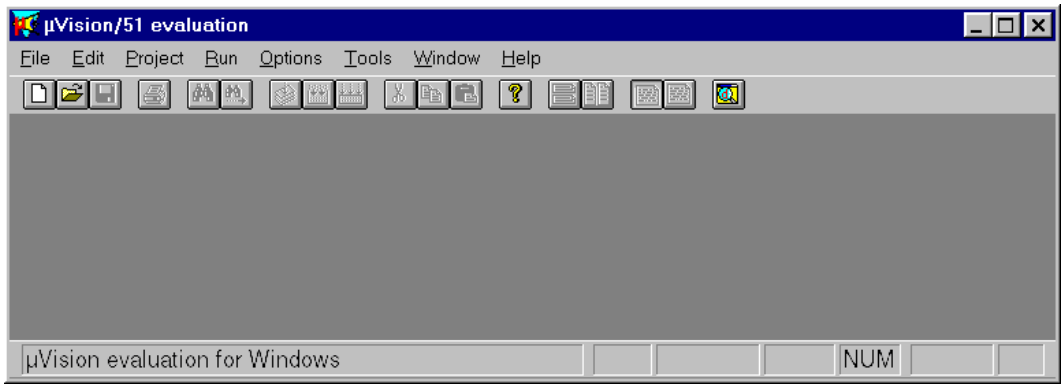


FIGURE 2

CREATING A NEW PROJECT

1. Open the Project menu and choose New Project. The window shown in Figure 3a appears.
2. Enter the name of the project you are creating. For this tutorial, enter the name *usb.prj* and press OK. *usb.prj* will be entered under File name. Figure 3b will then open up.

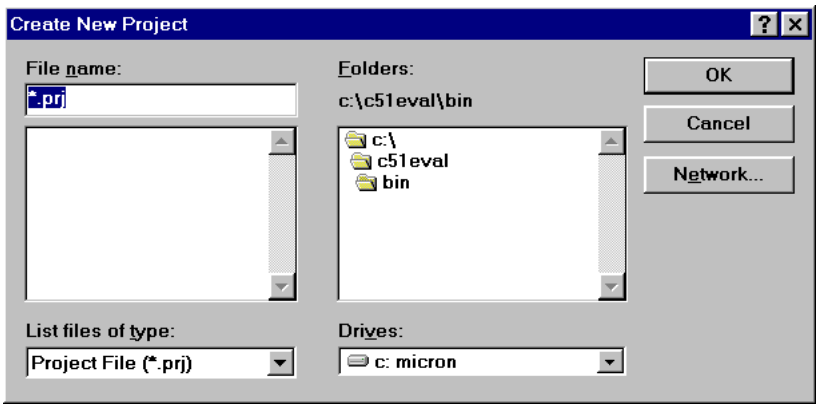


FIGURE 3a

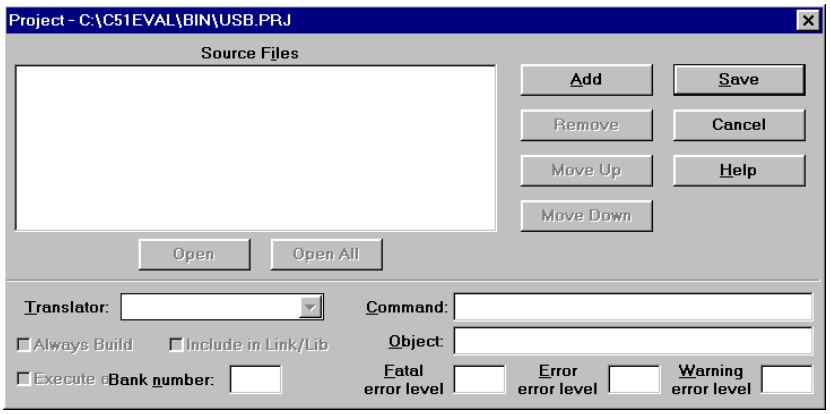


FIGURE 3b

This window is used to add various files to your project. These include ASCII files, C and assembly source, and macros. The list is quite extensive and is found in the Translator window if any files are present in the project. Note this window is blanked out at this time. This window is accessible at any time by selecting Project/Edit Project and you may easily edit your file list. Note that any assembler files must be last in this list. If they are not, changes made to them may not be reflected in the final object file.

When creating a new project in this manner no source files are yet available. Therefore, select *save* to close this window. Your project *usb.prj* will be active. Click on Project and confirm *usb.prj* is visible at the bottom of this pull down menu.

CREATING A NEW SOURCE CODE FILE

This tutorial uses a simple C program to illustrate features of the Keil Tool Set. It is shown in Figure 6. The pathname for this file (*example.c*) is *c:\c51eval\examples\tutorial\example.c*. Type this program in if you do not have this directory: This tutorial needs *example.c* in c:\c51eval\bin.

1) Do this step if you do not have the source. Open the File menu and choose *New* to go to the µVision integrated editor. Use the editor to type in *example.c* in Figure 6. It should look like Figure 5. When you have entered the file, do a *save as* to the c:\c51eval\bin directory as in Figure 4. The filename should be *example.c* for this tutorial. This will keep things easy to follow.
2) If you have the source code as a file, choose *Open* and get *example.c* in the usual fashion. Your window should be similar to Figure 5. Note the color syntax allowing increased readability of your

source code. This sample program uses a number of simple C source lines to demonstrate the Keil tool set. This program is used in the remainder of the tutorial.

3) Open the File menu and choose *Save as*. The window similar to Figure 4 appears. Save this file in the indicated directory. This tutorial uses the directory: *c:\c51eval\bin* and the filename *example.c*. Normally your project and source code would be in a directory of your choosing.

4) At this point you have created a project called *usb.prj* and a C source file called *example.c*.

The next step is to build your project. This includes compiling, assembling, linking and locating and creating the hex file. The hex file would be programmed into an EPROM and is not used here.

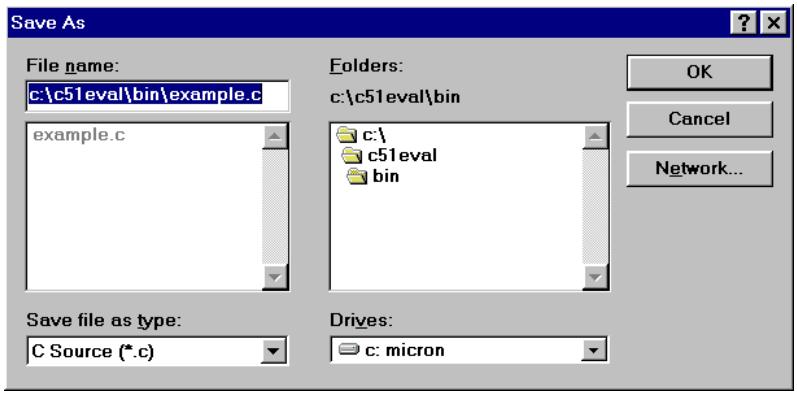


Figure 4

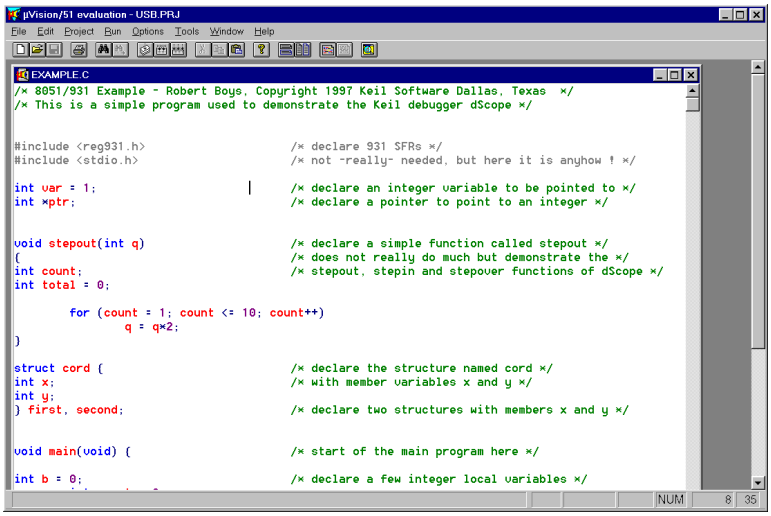


Figure 5

```
/* 8051/931 example.c - Robert Boys, Copyright 1997 Keil Software Dallas, Texas  */
/* This is a simple program used to demonstrate the Keil debugger dScope */

#include <reg931.h>                     /* declare 931 SFRs */
#include <stdio.h>                      /* not -really- needed, but here it is anyhow ! */

int var = 1;                            /* declare an integer variable to be pointed to */
int *ptr;                               /* declare a pointer to point to an integer */

void stepout(int q)                     /* declare a simple function called stepout */
{                                       /* does not really do much but demonstrate the */
        int count;                      /* stepout, stepin and stepover functions* of dScope/
        int total = 0;

        for (count = 1; count <= 10; count++)
                q = q*2;
}
```

```
struct cord {                           /* declare the structure named cord */
int x;                                  /* with member variables x and y */
int y;
} first, second;                        /* declare two structures with members x and y */

void main(void) {                       /* start of the main program here */

        int b = 0;                      /* declare a few integer local variables */
        int count = 0;
        long bigcount = 0;

        ptr = &var;                     /* declare ptr to point to the variable var */
        *ptr = 3;
        first.x = 5;                    /* and initialize them */
        first.y = 5;
        second.x = 2;
        second.y = 2;

        while(1)        {               /* start an endless loop here */
                P1 = 0;
                first.x++;              /* increment all the pointers */
                second.y++;             /* just for something to do */
                first.y++;
                second.x++;

                count++;                * increment this thing while we are here  */
                stepout(b);             /* pay a visit to the function for a time  */
                bigcount=bigcount+2;    /* add two to this long variable */

                *ptr=*ptr*3;            /* multiply var by three using its pointer */
                P1=0xff;
                }
}
```

Figure 6. Example Program

BUILDING THE PROJECT

1) Open the Project menu and choose *Edit Project*. The window shown in Figure 7 appears. This is where you add various files to your project. Note there are no files in the project yet.

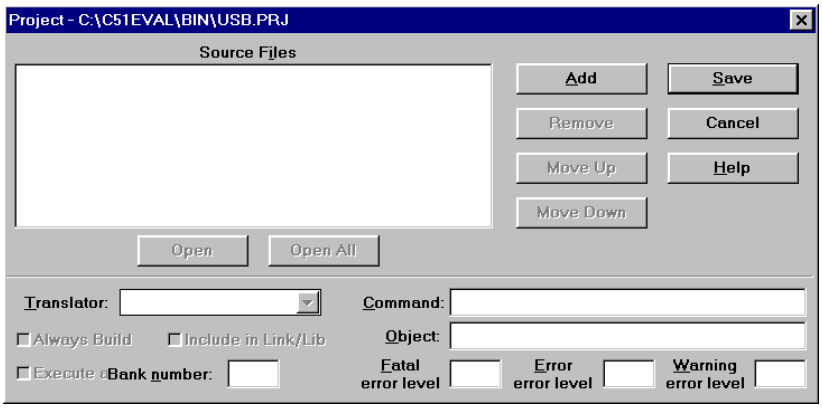


FIGURE 7

2) Choose *Add*. The window shown in Figure 8 appears. This is where you add the files to your project.

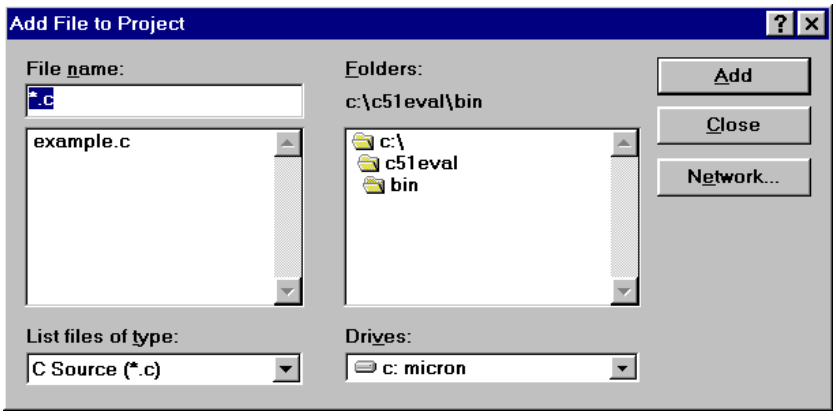


FIGURE 8

3) Select the file: *example.c* and press Enter.
4) Choose *Add* then C*lose*. *Example.c* will be listed in the Source Files window.
5) Make sure *Include in Link/Lib* is checked.
6) Choose *Save*.

CONFIGURING THE MAKE UTILITY, ASSEMBLER, AND LINKER

Keil Software has a Make utility that can compile and link C and/or assembly coded files. It also has other configuration and default setting features. Before using the Make utility, configure the make options. Make the changes as indicated, and leave all other options set to their default values.

In this tutorial, most of the default values are used. µVision allows you to set various options with mouse clicks and these settings are saved in your project file. Examine some of the other tabs in each window to see the flexibility of µVision. Do not make any changes other than those explained here.

1) **Configure the Make utility.**

- Open the Options menu and choose *Make*.
- Configure the options in the window as shown in Figure 9 and press *OK*.

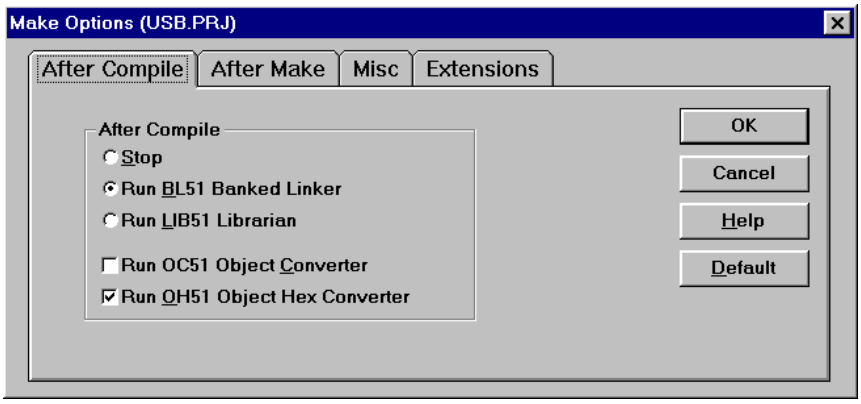


FIGURE 9

2) **Configure the Compiler.**

- Open the Options menu and choose *C51 Compiler*. Click on the *Object* tab.
- Note the DOS command line options are visible at the bottom of the window. You will see what actions are taken with your selections.
- Configure the options in the window under Object as shown in Figure 10 and press *OK*.
  It is important to turn on the Include debug information. This will allow us to see the source code in the debug window in the simulator and the monitor.
- The assembler is not used in this tutorial. Assembler options are set in a similar way as the other components of the tool chain.

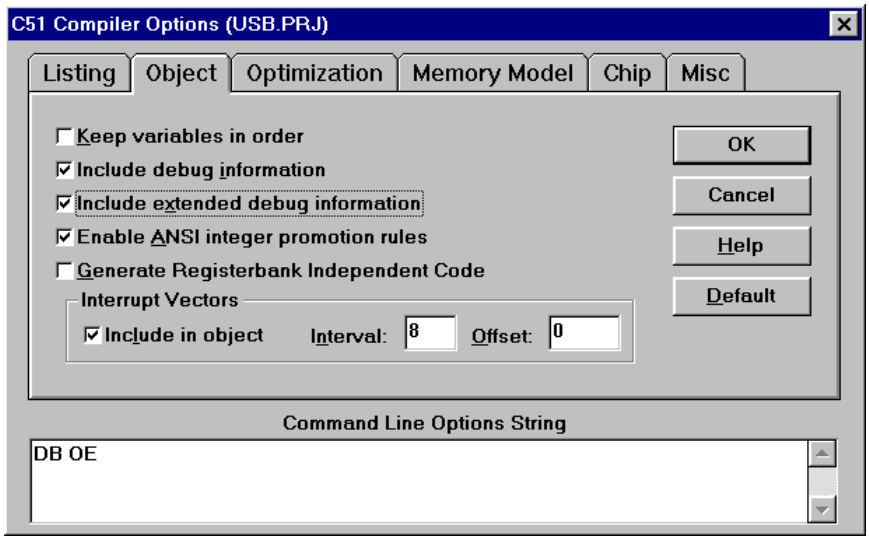


FIGURE 10

3) Configure the Linker.

- Open the Options menu and choose *BL51 Code Banking Linker* and click the *Linking* tab.
- Ensure the default conditions as shown are properly set. Note this is where the Banking, Real-time Operating System and Variable Overlaying are controlled. See Figure 11.

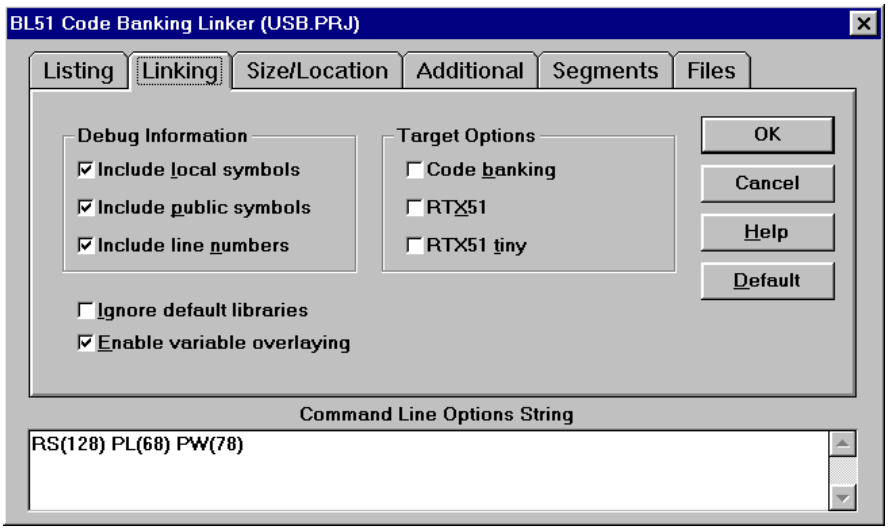


FIGURE 11

COMPILING THE CODE AND CREATING AN EXECUTABLE FILE

With the tool configured, you are ready to run the compiler and linker using the Make utility.

1. Click on the *"Build All"* icon (it has three arrows pointing downwards) or open Options and select Make.

- If the program specified (*example.c*) has any errors, they will appear on the screen. Use the editor to correct the error(s) in the source code and save the file. You can double-click on the error of interest and µVision will take you to the offending line in the source code. You can edit this line and rebuild the project by repeating this section: beginning at step 1.
- If there are no errors, the code is assembled and linked with the executable code ready to be downloaded to the board. The Project Status window will state "Make Successful - HEX File Created" if everything is working properly. Continue to the next section.

The following files in the directory \bin are associated with this project:

- example.c original source file - needed for debugging purposes.
- example.bak a backup file produced by µVision.
- example.lst a listing file of the source example.
- example.obj a relocatable object file. Needs to be linked.
- usb.prj the Project File. Note that the output code assumes the name of the project file.
- usb.m51 map file.
- usb.hex Intel Hex File Created by the Object to Hex Converter oh51.exe.
- usb absolute object file with debugging information (if so set in the compiler option) This file is the input for the Keil simulator dScope and emulators. This file is created by the linker.

DOWNLOADING THE CODE TO THE 8X931 EVALUATION BOARD

You must make sure the Intel board is configured and loaded according to the Adapter Board User's Guide. The procedure is to remove the existing 8x930Ax microcontroller U3, install a new PLD U8, EPROM U1, and the Adapter board itself. The appropriate jumpers must be set according to the User's Guide.

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

Supply regulated 5 volts to P9 on the board. Please carefully apply the correct polarity. With the serial cable not connected and the power applied to the board; the LEDs will indicate a self test and the even numbers of P1 plus p1.7 will remain illuminated at the conclusion. Pressing the reset button will repeat the self test. The green LED CR1 always remains lit as long as reset is not asserted. The four green LEDs on the adapter board will also blink in sequence. Connect the serial cable from a COM port on your PC to the UART port on the Intel USB board.

With the serial cable hooked up; all the LEDs remain illuminated as the board is held in the reset condition by the serial port. This assumes the serial port has not been initialized by dScope in a previous session.

Click on the debugger icon on the µVision toolbar. This icon is a "d" seen through a magnifying glass and is yellow and blue in colour. This is the same icon installed for dScope in your desktop area. If you leave the mouse pointer on an icon for a few seconds, the word "debug" will appear. A window similar to that shown in Figure 12 is displayed. You may want to move and resize the windows. The window marked "*Module: <none>*" is the Debug window. The Command window will be needed to enter commands.

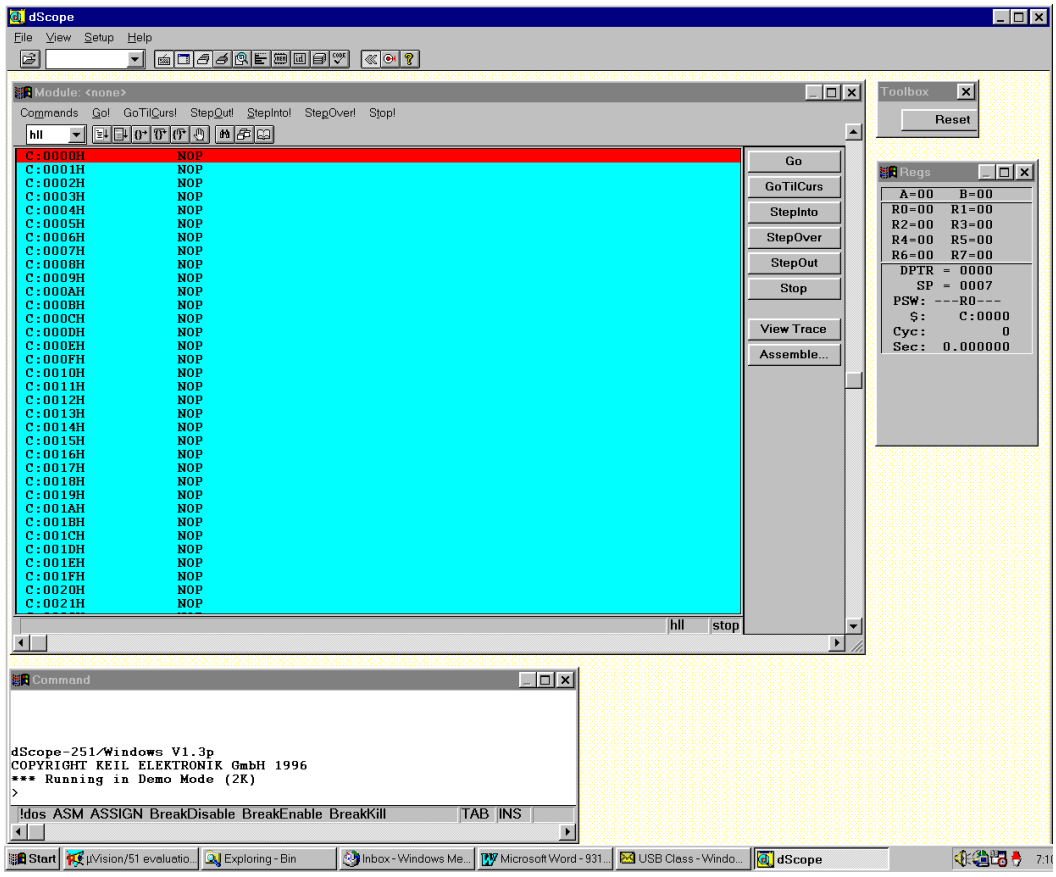


FIGURE 12

INTERFACE SETUP

dScope is a software debugger and simulator. tScope is the debugger connected to a hardware system. Programs are run on the target hardware rather than in the virtual microcontroller in your PC. The Keil debugger tScope will communicate with the Intel RISM monitor contained in the EPROM on the USB board. The COM port will be dependent on your configuration and the speed is 9600 baud.

The upper left hand corner of Figure 12 shows that dScope is active in this case. This will change to tScope if the appropriate dll (i.e. rism51.dll) is chosen. If you select *8051.dll*, you will get dScope. Your simulation will not be run on the Intel USB board but in your PC as already stated.

1) To establish communication with the evaluation board, choose the correct driver using the pulldown list on the left portion of the debugger toolbar. In this example, rism51.dll is chosen.
2) *rism51.dll*: When the serial link is connected and the *rism51.dll* is selected for the first time, the leds will blink. The reset button in the tScope Toolbox window will also cause the lights to blink. If you get this far, things are working properly. tScope should be displayed in the upper left hand corner.
3) This will happen only if communication is established. If the wrong COM port or speed is selected, a series of communication I/O timeouts will be indicated in the Command window. Should this happen; follow the instructions under Communications Setup.

COMMUNICATIONS SETUP

You need to follow these instructions if proper communication was not established. After a series of attempts dScope will error out as indicated in the command window and a "Target System not found " window will open. You can now change the COM port and baud rate settings.

Select the appropriate COM port for your system. The speed must be 9600 for the external UART and the Serial Break must not be enabled for this exercise. * Make sure your serial cable is plugged into the UART!

After you set the COM port and baud rate to 9600, click on *Try Again.*

LOADING THE EXECUTABLE FILE TO THE DEBUGGER ENVIRONMENT

1) Open the File menu and choose *Load Object file.*
2) In the Window provided, select the file: *usb*. The file is downloaded to the board and the debugger. This file has no extension. Press OK.
3) To see the Command window if not already visible, select View/Command Window. To see the source code, select the Debug window in the same fashion if necessary. *Module: example* will appear at the top of debug screen.

You should see the source in the Debug window as in Figure 13 depending on how you have set your windows. You can single step using the step-into icon. You are now ready to use the dScope window (really tScope) to step through code, set breakpoints, and issue the Go command to start program execution. You can examine special function registers, memory locations, and register values, etc.

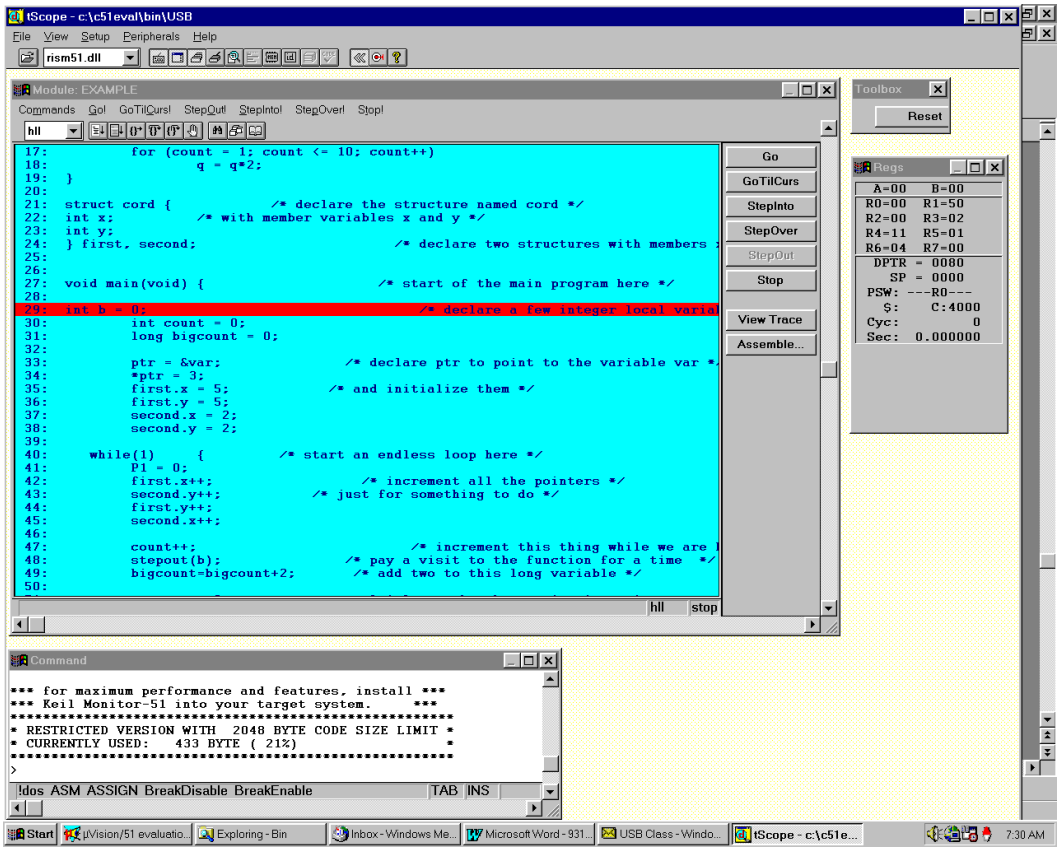


FIGURE 13

SAMPLE SESSIONS

High Level and Assembly Language Listing

1) Note the screen displays High Level language source code information. This is because the Debug window is set to view HLL as indicated in the box just beneath "Commands"
2) Open this box and change it to Mixed and the screen will display both HLL and Assembly listings. Also under commands/view/mixed.
3) Change it back to HLL.

Single Stepping

1) dScope uses "*Step Into*" to single step one instruction at a time. "*Step Into*" is also used to enter a function in the same fashion.
2) "*Step Over*" means to skip over a function that you are not interested in.
3) "*Step Out*" is used to exit a function you are currently in. "*Step Out*" is very useful if you find yourself in a function you are not interested in and need to return quickly to your intended function.
4) If dScope gets stuck - do a reset using the RESET box on the Toolbox or click on the button next to the Help "?", reload the file and/or the DLL. You can also try the 8051.dll to isolate any problems from the evaluation board. You can also use the hardware RESET button on the board.
5) Put a breakpoint on Line 49 and press Go. Click on StepInto until you enter the function Stepout. Note that you must click on StepInto 10 times to exit the loop. Press Go to go back to Line 49.
6) Repeat the process using StepOver and you will not enter the function although it will be executed. This is useful to skip function calls you are not interested in debugging.

7) Note that the StepOut button is grayed out and not available in tScope. StepOut is available in the simulator dScope and provides a quick escape from a function by executing the next *return* instruction.

Breakpoints

1) Reset using the RESET box on the Toolbox or click on the button with the red pointer at top of screen. The lights on the board should change.
2) Click on line *41 P1 = 0* and a colored bar appears marking this position.
3) You could click on *GoTilCurs!* to reach this point or you could double-click and a breakpoint is set. Set a breakpoint here with the double click. The [BR0] indicates the first breakpoint.
4) Click on GO and the program will run and stop at the breakpoint. - 6 LEDs will be on.
5) Click on *StepInto* and the instruction will be executed extinguishing the LEDs.
6) Double click on the breakpoint to remove it.
7) Click on Line *52 P1 = 0xff* and press *GoTilCurs!*.
8) Click on *StepInto* to execute the instruction and 6 of the LEDs will light up.
9) Click on RESET.

In Line Assembler

1) Open the Commands option and click on Inline Assembler or if Show Dialogbar is active, click on the Assemble button in the Debug window.
2) The Inline Assembler windows opens as in Figure 14.
3) Note that you could enter mnemonics in the window titled *enter MCS-51 instruction:* Do not do this at this time. Just note the ability to make small changes in the program without recompiling the program.
4) Click on Close.

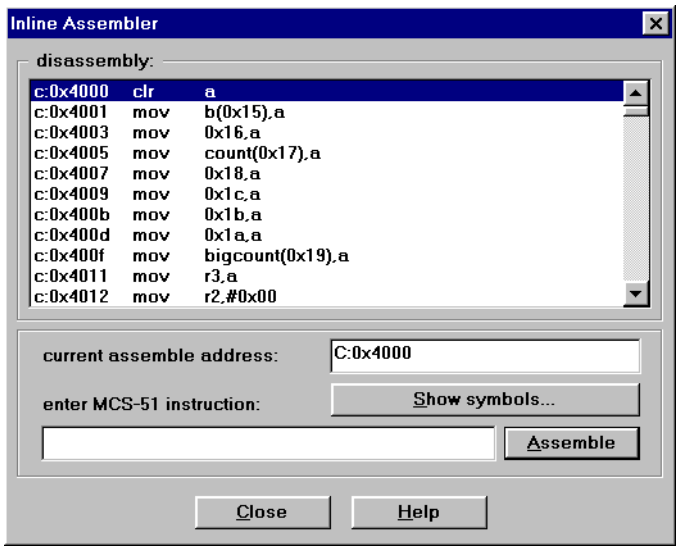


Figure 14

Memory Window

1) The Memory Window shows as the default the data space starting at D:0x0000. This is the area where data variables are kept in this example.
2) Open the View menu and select Memory Window. Position this window off the Debug window.
3) Note that as you step through the program that the contents of the memory changes as the variable values are adjusted.

4) You can change the memory area with the DISPLAY command in the Command window. Enter d c:0x00 and the code area will be displayed.

5) Change back to the data area with d d:0x00

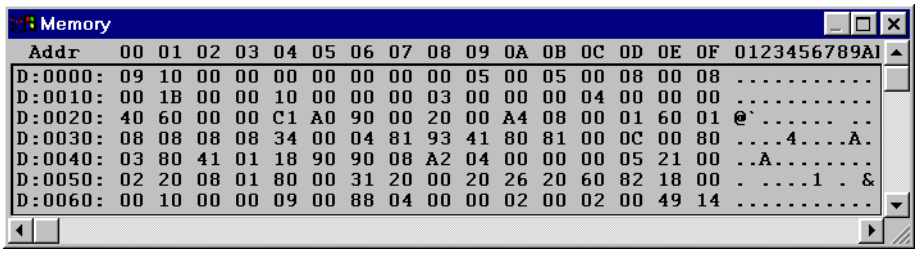


Figure 15

Watch Window

1) The Watch Window displays memory contents as specified by their name. Structures can also be displayed.

2) Open the Setup menu and select *Watchpoints*. In the Expr.: window, enter the following symbols one at a time followed by clicking on *Define watch*.
bigcount, first, second and *var*. Note that if you also select *0xnn* the values will be displayed in decimal rather than hexadecimal notation. Try this for one or two of the variables.

3) These variables will now be displayed in the Watchpoint window. Click on *Close*.

4) Click on *StepOver* and the variable values will changes as appropriate.

5) Note: when are using dScope (i.e. the software debugger) and if in the *Update Memory Window* and *Update Watch Window* (in the Setup menu) are activated - the variable values will change as the program is running. This feature is not possible with tScope running on a hardware board.

6) Open the Peripherals menu and select Configuration. Activate *enable serial break* and click on *Apply* then *Close*.

7) Do a RESET and start over as before. Press GO and then STOP and note that the values change in both the Watch and Memory windows when the processor is stopped.

USB ENUMERATION

Introduction

USB devices are hot-pluggable. When a device is connected to a PC, the peripheral is detected and certain communication occurs between it and the host. The appropriate drivers will be selected and the peripheral will be installed on the host (the PC). This is the enumeration process and this can be demonstrated with the Intel Evaluation board.

Host - Windows 95 B

A host is also a Hub. The fastest way to get a host is to get a PC motherboard with one or two USB ports installed and running Windows 95 B. The B version of Windows 95 is sold at present only (October 1997) to OEMs of PCs. It cannot be purchased separately.

It can be identified by checking the version of Windows by right clicking on "My Computer" and selecting properties. The general tab will show the Windows version. If it lists the system as *4.00.950 B* then USB can be supported. A floppy disk should come with the computer labeled “USB Supplement for Windows 95”. This is called “OSR 2.1”. This disk must be installed on your system and the USB driver must be listed in the Device Manager under System Properties.

Demonstrating the Enumeration Process

1) Power up the 931 board without the serial cable connected. RISM need not be running.
2) Your USB equipped PC must have OSR 2.1 installed as mentioned above.
3) Connect the USB cable between the Intel board and the PC. The mouse arrow will turn into a hourglass, then a window indicating detection will appear.
4) If no valid USB device was detected If the hourglass returns to the mouse pointer; this means no valid USB device was detected.
5) This indicates that Windows has detected the peripheral (the Intel board). It is trying to locate a driver using the identifiers passed to it by the Intel board.
6) Windows will not be able to find a driver since none are loaded. The window in Figure 16 will indicate that Windows could not find the appropriate drivers and is giving you the opportunity of providing them.

Figure 16

CONCLUSION

What else is there ?

Plenty…

- Open the Command option in the Debug window and notice you can select High Level Language, assembler or both (mixed) to be displayed and the trace. Note the trace only works for software simulation - tScope is unable to determine the actual bus values in your hardware.
- Open View and note the many windows you can open such as Watchpoints, Register, Performance Analysis and Code Coverage.
- Open Peripherals and you can see windows illustrating values of I/O Ports, Timers and counters.
- Open the Setup window and you can set many options such as the Performance Analyzer, Register sets, breakpoints and Watchpoints.

- On line Help is also available.
- An Application Note giving instructions on how to replace the RISM Eprom in the Intel evaluation board with your own code produced with Keil software. This note is for the 8x930Ax but can be adapted for the 8x931. Get application Note 112 from www.keil.com

What is next ?

Plenty…

- More sessions using C code will be prepared. More dScope features will be explored such as the Performance Analyzer, Watchpoints and Memory windows.
- Check www.keil.com/~market for further details of more USB data.

For additional information on Keil software development tools, visit the Keil web site at : www.Keil.com. Check the Keil Marketing Group Web page at www.keil.com/~market.

Any problems or questions ? Call us. We are here to help you with your USB project.

Robert Boys
Dallas, Texas
October 15, 1997
800-248-8051
rboys@keil.com

Intel Support from Keil Software Inc. Dallas, Texas

Keil Software develops, manufactures, and distributes embedded software development tools for the 8051, 251, and USB microcontroller families. Tools include C compilers, Assemblers, Real-time Executives, Debuggers and Simulators, Integrated Environments, and Evaluation Boards. Keil has a training facility in Dallas for 8051 and USB products. Keil provides distribution, product development, and technical support from its office in Dallas, Texas.

Keil's web site [www.keil.com] provides the latest information about our development tools, demo programs, software updates, application notes, example programs, and links to other sources of information.

C51 C Compiler for the entire 8051 family and the Intel 8x931Hx/Ax

The C51 is a full ANSI C compiler with assembler. µVision is the Windows User Interface that drives the optimizing C51. The Compiler, Assembler and Linker options are point and click. µVision and the manuals are written in the USA and extensive on-line help is included. Free technical support from Dallas via a 1-800 line is included for one year.

C251 C Compiler for the 251 and the 8x930 [USB]

The C251 ANSI compiler with the A251 assembler are designed specifically for the Intel and Temic 251 microcontroller family, including the 8x930 USB. C251 is 100% compatible with existing 8051 programs. Extensions provide access to all 251 hardware components. Sample USB/251 code is available: www.keil.com/usb. C251 supports code banking, source and binary modes. Keil has a free tutorial available for the new Intel 4 port USB evaluation module and the C251.

A51, A251: Macro Assemblers for all Keil supported microcontrollers

These three Macro Assemblers are included with their respective Compiler package or are available separately. All utilities needed to complete your project are included for all members of the assembler's family. These Assemblers are DOS based or can be run from µVision which is included with every Assembler and Compiler package.

µVision Integrated Development Environment - Windows based User Interface.

µVision is a USA developed Windows-based front end for all Keil Compilers and Assemblers. It includes an editor, project manager, and make facility. Compiler, assembler, and linker options are set by pointing and clicking on prompted selections. Program Manager conveniently accesses your files, and 3rd party executables and also calls the Keil Simulator dScope easing the transition from utility to utility. Works with Windows 3.11, 95 and NT.

dScope-Debugger and Simulator for the 8051, 251 and 930. Windows or DOS based.

dScope is a source-level debugger that lets you debug programs created by Keil compilers. dScope simulates your program either in stand-alone mode or in your target using the monitor. External hardware, signals, and interrupts can be simulated. Viewable windows include View Program from I/O, Trace, Stack, Watch and CPU registers plus more.

TR51/FR51/FR251 Full-Function RTOS for the entire 8051, 251, 8x930 and 8x931 families

The RTX51 and RTX251 are a multitasking Real-time Operating Systems for the entire 8051, 251, 8x931 and 8x930 families. This powerful tool lets you manage multiple tasks on a single CPU. The RTX51 Full includes CAN libraries. The RTX51 Tiny is a subset of the RTX51 Full. Functions include interrupt and memory management, clock, and semaphores. Royalty-free.

CAN Library

The RTX51 Full RTOS supports CAN controllers with the included libraries. The CAN libraries are sold with the RTOS. The CAN interface is becoming popular for automotive and industrial markets. 11 and 29 bit identifiers are supported. Keil 8051 C compilers interface with the RTOS and CAN libraries. Keil supports all 8051 CAN microcontrollers including the Intel 526 and 527.

Upgrades

Free upgrades for Keil compilers are available from the Web to registered users. Special upgrade prices are available for selected Franklin and Archimedes compilers. Call Keil for details on how to save money using this special program.

Keil Software
16990 Dallas Parkway, Suite 120
Dallas, Texas 75248-1903

phone (800) 348-8051
phone (972) 735-8052
fax (972) 735-8055
Email sales@keil.com
Web www.keil.com