PUMPKIN™
REAL-TIME SOFTWARE

AN-20
Application Note

750 Naples Street • San Francisco, CA 94112 • (415) 584-6360 • http://www.pumpkininc.com

Building a Salvo Application with ImageCraft's ICC430 Development Tools

Introduction

This Application Note explains how to use ImageCraft's (http://www.imagecraft.com/) ICC430 Development Tools to create a multitasking Salvo application for TI's (http://www.ti.com/) MSP430 ultra-low-power microcontrollers.

We will show you how to build the Salvo application contained in `\salvo\ex\ex1\main.c` for an MSP430F149 using ICC430 v6.02. For more information on how to write a Salvo application, please see the *Salvo User Manual*.

Before You Begin

If you have not already done so, install the ImageCraft ICC430 Embedded Tools. Familiarize yourself with the ICC430 IDE. You will also need an MSP430 Flash Emulation Tool (FET) for debugging. More information is available at http://www.ti.com/sc/msp430.

Related Documents

The following Salvo documents should be used in conjunction with this manual when building Salvo applications with ImageCraft's ICC430 Development Tools:

Salvo User Manual
Salvo Compiler Reference Manual RM-ICC430

created by Andrew E. Kalman on Oct 5, 2002 updated on Jul 23, 2003

Creating and Configuring a New Project

Create a new ICC430 project under Project → New. Navigate to your working directory (in this case we've chosen `c:\temp`) and create a project named `myex1.prj`:

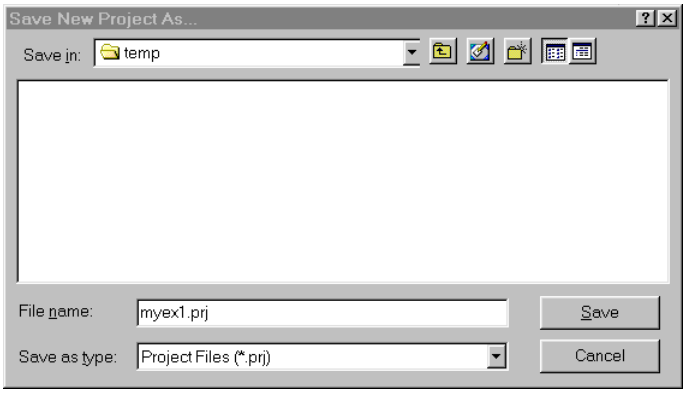

Figure 1: Creating the New Project

Click Save to continue. The ICC430 IDE will automatically save the project whenever you close it.

In order to manage your project effectively, we recommend that you create a set of folders for your project. They are:

Listings
Salvo Configuration File
Salvo Help Files
Salvo Libraries
Salvo Sources
Sources

For each folder,[1] choose Add Folder… by right-clicking in the Project window, enter the desired name under Folder Name and click OK.

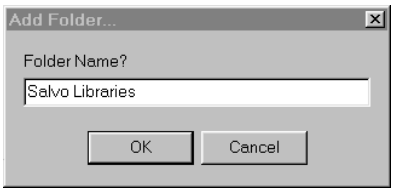

Figure 2: Creating a Group

When finished, your Project Manager window should look like this:

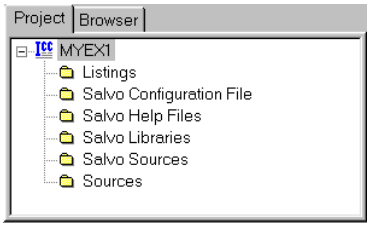

Figure 3: Project Manager Window with Folders

Now let's setup the project's options for Salvo's pathnames, etc. Open the Compiler Options window by selecting Project → Options… → Paths. Add the project's own include path and `\salvo\inc\`,[2] separated by semicolons:

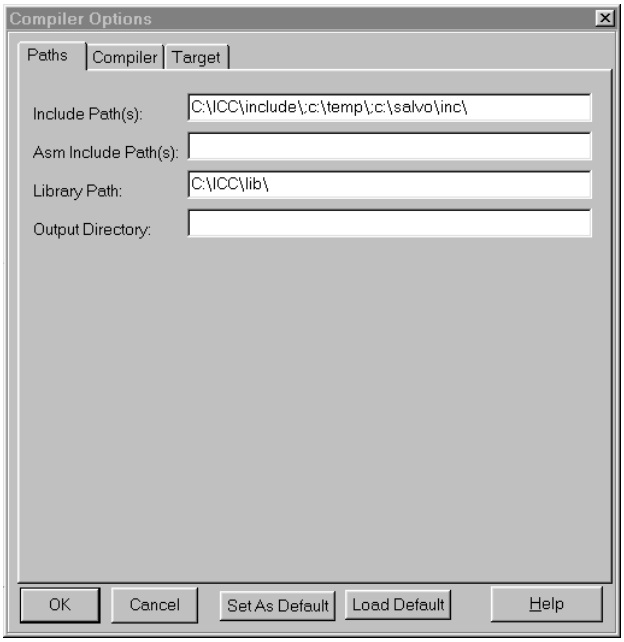

Figure 4: ICC430 Settings – Project Include Paths

Next, define any symbols[3] you may need for your project in the Compiler Options window by selecting Compiler and entering the symbols under Macro Define(s):

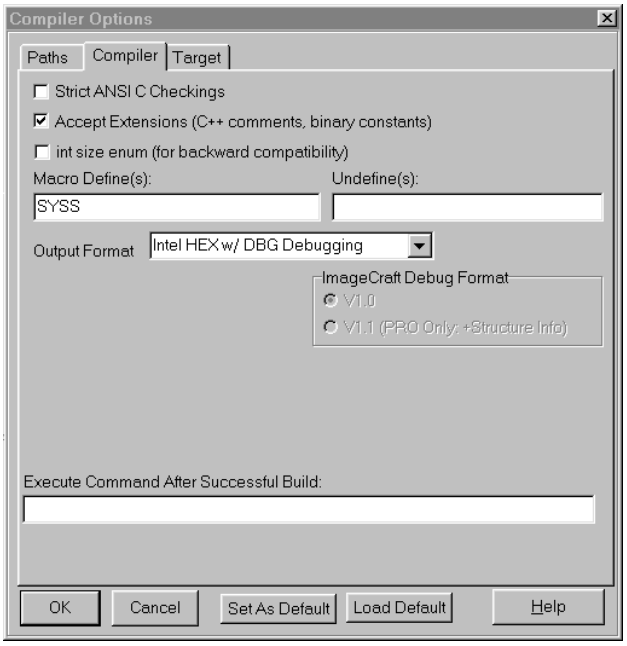

Figure 5: ICC430 Options – Project Compiler Settings

Lastly, in the Compiler Options window under Target, select the appropriate Device Configuration:

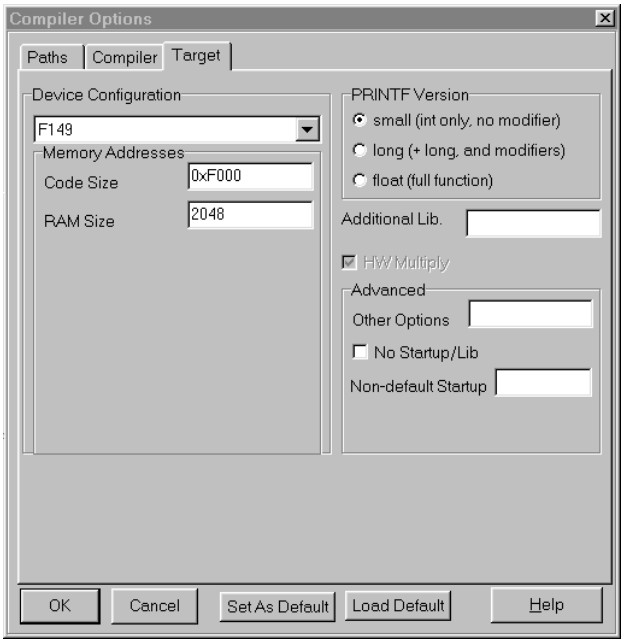

Figure 6: ICC430 Options – Project Target Settings

Click OK to finish setting your project's options.

Adding your Source File(s) to the Project

Now it's time to add files to your project. In the Project Manager window, select the Sources folder, right-click to choose Add Files…, choose Files of type: Source Files (*.c, *.s, *.h), navigate to your project's directory, select your `main.c` and click Open. Your Add Files… window should look like this:

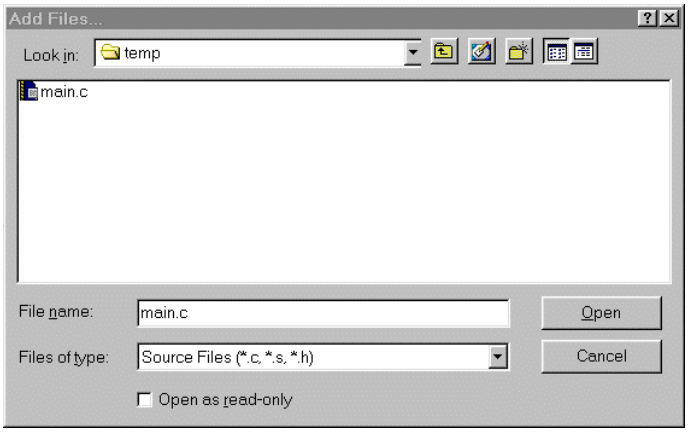

Figure 7: Add Files … Window

When finished, your Project Manager window should look like this:

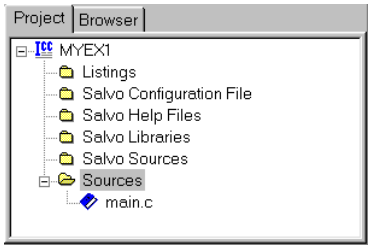

Figure 8: Project Manager Window with Project-Specific Source Files

Adding Salvo-specific Files to the Project

Now it's time to add the Salvo files your project needs. Salvo applications can be built by linking to precompiled Salvo libraries, or with the Salvo source code files as nodes in your project.

Adding a Library

For a *library build*, a fully-featured Salvo freeware library for the MSP430 for use with ICC430 is `libsficc430-a.a`.[4] Select Project → Options… → Target, and under Additional Lib. enter `sficc430-a`:

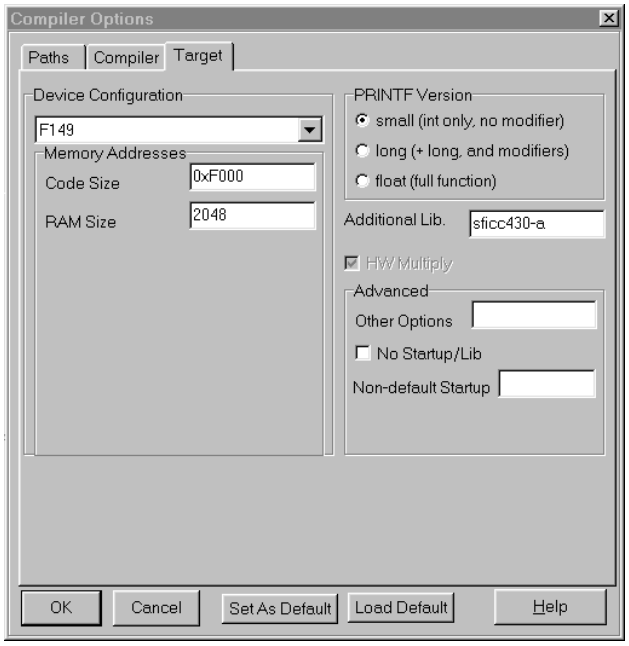

Figure 9: Adding the Library to the Project

Click OK when you are finished. You can find more information on Salvo libraries in the *Salvo User Manual* and in the *Salvo Compiler Reference Manual RM-ICC430*.

Adding Salvo's mem.c

Salvo library builds also require Salvo's `mem.c` source file as part of each project. In the Project Manager window, select the Salvo Sources folder, right-click to choose Add Files…, choose Files of type: Source Files (*.c, *.s, *.h), navigate to `\salvo\src`, select `mem.c` and click Open. Your Add Files… window should look like this:

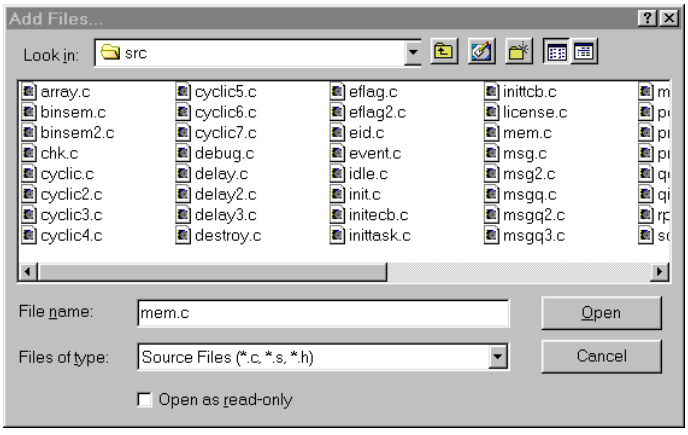

Figure 10: Add Files … Window

The salvocfg.h Header File

You will also need a `salvocfg.h` file for this project. To use the library selected in Figure 9, your `salvocfg.h` should contain only:

```
#define OSUSE_LIBRARY          TRUE
#define OSLIBRARY_TYPE         OSF
#define OSLIBRARY_CONFIG       OSA
```

Listing 1: salvocfg.h for a Library Build

Create this file and save it in your project directory, e.g. `c:\temp\salvocfg.h`. For convenience, add it to your project's Salvo Configuration File folder:

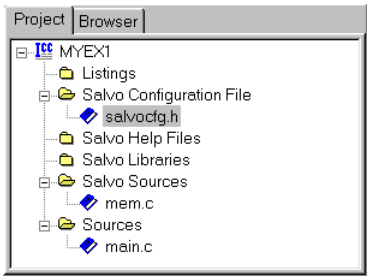

Figure 11: Project Manager Window for Library Build

Proceed to *Building the Project*, below.

Adding Salvo Source Files

If you have a Salvo distribution that contains source files, you can do a *source code build* instead of a library build. The application in `\salvo\ex\ex1\main.c` contains calls to the following Salvo user services:

```
OS_Delay()              OSInit()
OS_WaitBinSem()         OSSignalBinSem()
OSCreateBinSem()        OSSched()
OSCreateTask()          OSTimer()
OSEi()
```

You must add the Salvo source files that contain these user services, as well as those that contain internal Salvo services, to your project. The *Reference* chapter of the *Salvo User Manual* lists the source file for each user service. Internal services are in other Salvo source files. For this project, the complete list is:

```
binsem.c                mem.c
delay.c                 porticc430.s
event.c                 qins.c
idle.c                  sched.c
init.c                  timer.c
inittask.c
```

In the Project Manager window, select the Salvo Sources folder, right-click to choose Add Files…, choose Files of type: Source Files (*.c, *.s, *.h), navigate to the `\salvo\src` directory and select[5] the `*.c` files listed above. Your Add Files… window should look like this:

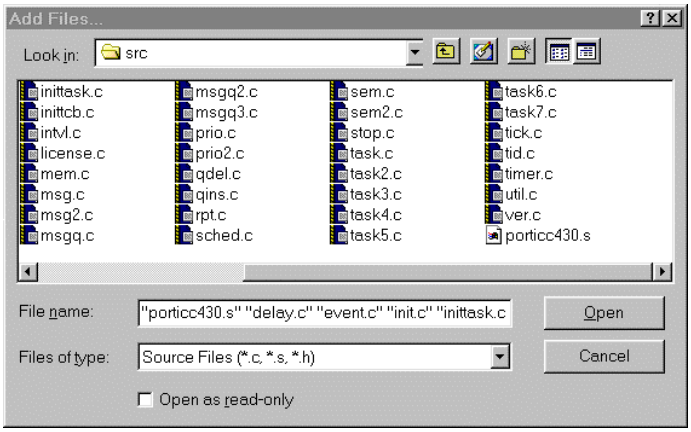

Figure 12: Adding Salvo Source Files to the Project

Click Open when finished.

The salvocfg.h Header File

You will also need a `salvocfg.h` file for this project. Configuration files for source code builds are quite different from those for library builds (see Listing 1, above). For a source code build, the `salvocfg.h` for this project contains only:

```
#define OSBYTES_OF_DELAYS            1
#define OSENABLE_IDLING_HOOK         TRUE
#define OSENABLE_BINARY_SEMAPHORES   TRUE
#define OSEVENTS                     1
#define OSTASKS                      3
```

Listing 2: salvocfg.h for a Source Code Build

Create this file and save it in your project directory, e.g. `c:\temp\salvocfg.h`. For convenience, add it to your project's Salvo Configuration File folder:

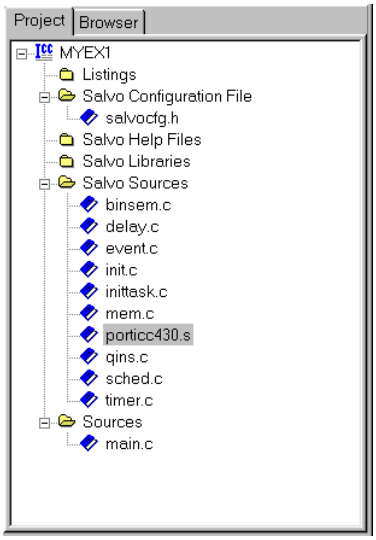

Figure 13: Project Manager Window for a Source-Code Build

Tip The advantage of placing the various project files in the groups shown above is that you can quickly navigate to them and open them for viewing, editing, etc.

Building the Project

For a successful compile, your project must also include a header file (e.g. `#include <msp430x14x.h>`) for the particular chip you are using. Normally, this is included in each of your source files (e.g. `main.c`), or in a header file that's included in each of your source files (e.g. `main.h`).

With everything in place, you can now build the project using Project → Make Project or Project → Rebuild All. The IDE's status window will reflect the ICC430 command lines:

```
C:\ICC\BIN\imakew -f myex1.mak
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\binsem.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\delay.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\event.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\init.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\inittask.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\mem.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult   C:\salvo\src\porticc430.s
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\qins.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\sched.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\salvo\src\timer.c
    icc430 -c -IC:\ICC\include\ -Ic:\temp\ –Ic:\salvo\inc\ -e
-DSYSS  -l -g -Wf-hwmult  C:\temp\main.c
    icc430 -o myex1 -LC:\ICC\lib\ -g -blit:0x1000.0xFFDF
-bdata:0x200.0x0A00 -dram_end:0x0A00 -fintelhex @myex1.lk
-lsficc430-a
Done.
```

Listing 3: Build Results for A Successful Source-Code Build

The map (`*.mp`) file located in the project's directory contains address, symbol and other useful information:[6]

```
Area                                Addr   Size   Decimal Bytes (Attributes)
--------------------------------   ----   ----   ------- ----- ------------
                             text   1000   0798 =   1944. bytes (rel,con,rom)

      Addr  Global Symbol
     -----  --------------------------------
      1000  __text_start
      1000  __start
      104E  _exit
      1050  _Task1
      1070  _Task2
      109A  _Task3
      10C2  _OSIdlingHook
      10C4  _main
      114A  _Timer_A
      117C  _OSDispatch
      1192  _OSCtxSw
      11BC  _OSTimer
      11EC  _OSSched
      1324  _OSInsPrioQ
      142C  _OSCreateTask
      14D2  _OSInitPrioTask
      1512  _OSInit
      1548  _OSWaitEvent
      15D4  _OSDelay
      167C  _OSCreateBinSem
      169E  _OSWaitBinSem
      16E0  _OSSignalBinSem
      176C  asgnblk
      1796  __MSP430Setup
      1798  __text_end

Area                                Addr   Size   Decimal Bytes (Attributes)
--------------------------------   ----   ----   ------- ----- ------------
                              bss   0200   0030 =     48. bytes (rel,con,ram)

      Addr  Global Symbol
     -----  --------------------------------
      0200  __bss_start
      0200  _OSframeP
      0202  _OSglStat
      0204  _OStimerTicks
      0208  _OSdelayQP
      020A  _OSsigQoutP
      020C  _OSsigQinP
      020E  _OSecbArea
      0214  _OSeligQP
      0216  _OStcbArea
      022E  _OScTcbP
      0230  __bss_end

Area                                Addr   Size   Decimal Bytes (Attributes)
--------------------------------   ----   ----   ------- ----- ------------
                           vector   0000   10000 =  65536. bytes (abs,ovr,rom)

Files Linked      [ module(s) ]

C:\ICC\lib\crt430.o [ crt430.s ]
mem.o     [ mem.c ]
main.o    [ main.c ]
<library> [ porticc430.s, timer.s, sched.s, qins.s, inittask.s, init.s, event.s,
delay.s, binsem.s, asgnblk.s, setup.s ]


User Global Definitions

ram_end = 0xa00

User Base Address Definitions

lit = 0x1000
data = 0x2000
```

Listing 4: Map File for a Source-Code Build

Note The ICC430 projects supplied in the Salvo for TI's MSP430 distributions contain additional help files in each project's Salvo Help Files group.

Using the Browser

By selecting the Intel HEX w/DBG Debugging output format (see Figure 5), ICC430 will build debug information for your project that can be used by the IDE's Browser:

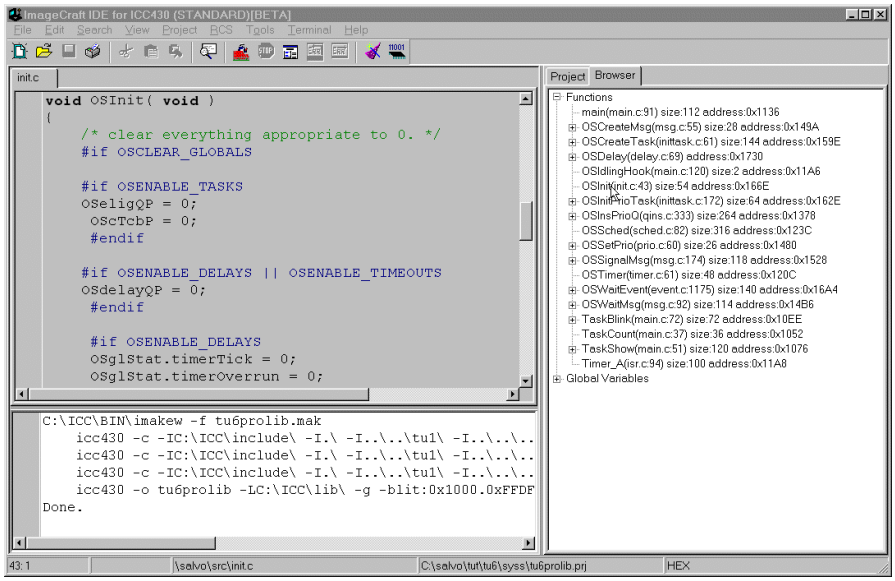

Figure 14: Browsing at the Source-Code Level

By double-clicking in the Browser on the function of interest, the source code that contains the function will be displayed in the Editors window. This works with any source code (project- or Salvo-specific), and also with the `i`-option Salvo libraries that include debugging information.

Testing the Application

You can test and debug this application using the Flash Emulation Tool. The NoICE 430 Remote Debugger (http://www.noicedebugger.com) launches when you choose Tools → NoICE430 Debugger. The NoICE430 debugger supports breakpoints, watch windows, mixed source/disassembly display, etc.

Tip When using NoICE430, be sure to select Files of type: ImageCraft DBG files when loading the application into your target. This will ensure that debug info is available to the debugger.

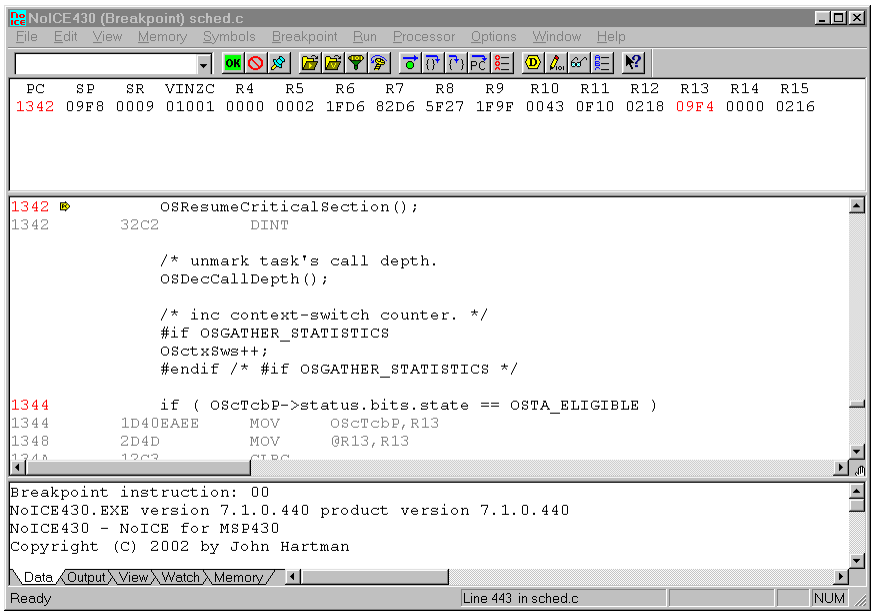

Figure 15: Testing a Salvo Application in NoICE430

Tip When debugging with NoICE430, the project's map (`*.mp`) file and listing (`*.lst`) files are very useful because they list the addresses of functions and variables in ROM and RAM. This information can be used in the monitor program to set breakpoints, display memory, better understand trace results, etc.

Note ICC430 can create generate debugging info via the `-g` command-line option. Only applications built from the Salvo source code or a Salvo Pro library enable you to step through Salvo services (e.g. `OSCreateBinSem()`) at the source code level when using an external debugger. Regardless of how you build your Salvo application, you can always step through your own C and assembly code with ICC430's output.

Alternatively, you can download your project to the FET using Tools → Flash Downloader, browsing for the hex file and then clicking on Program FLASH. This will enable you to program your target system and run your application, but no ICE-like debugging facilities are provided.

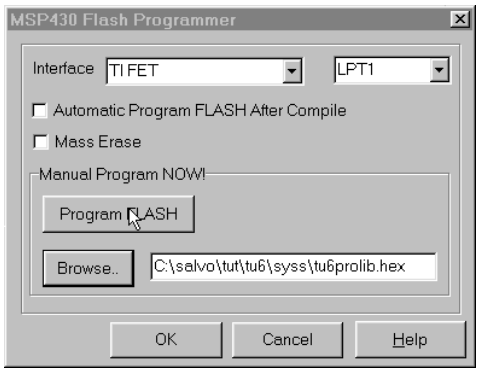

Figure 16: Programming via Flash Programmer

Troubleshooting

Cannot find and/or read include file(s)

If you fail to add `\salvo\inc` to the project's include paths (see Figure 4) the compiler will generate an error like this one:

```
!E D:\salvo\src\event.c(30): Could not find
include file "salvo.h"
```

Figure 17: Compiler Error due to Missing \salvo\inc Include Path

By adding `\salvo\inc` to the project's include path, you enable the compiler to find the main Salvo header file `salvo.h`, as well as other included Salvo header files.

If you fail to add the project's own directory to the project's include paths (see Figure 4) the compiler will generate an error like this one:

```
!E c:/salvo/inc/salvo.h(292):
D:\salvo\src\delay.c(27): Could not find
include file "salvocfg.h"
```

Figure 18: Compiler Error due to Missing Project Include Path

By adding the project's own directory to the project's include path, you enable the compiler to find the project-specific header file `salvocfg.h`.

Unable to Communicate with FET

Be sure to set[7] the mode of your computer's parallel (LPT) port to ECP (instead of bi-directional, SPP or EPP), or else NOICE430 and the FLASH Downloader are likely to fail.

Application Crashes After Changing Processor Type

Remember to `#include` the appropriate header file for your MSP430 variant (see to *Building the Project*, above). While the common SFR locations are consistent across the entire MSP430 family, the interrupt vectors are not. Therefore mainline code may work correctly, but the application will crash if interrupt vectors are not in the right locations.

Cannot Resolve Location of Salvo Source Files

The Salvo Pro libraries with embedded debug information (`i`-option) reference the salvo source files in their default location, `\salvo\src`. If you have placed these files in an alternate location and you want to use debugging information, you can edit the library files and change the pathnames that reference Salvo source files. An automated method (e.g. a perl script) is recommended.

Example Projects

Example projects for the ICC430 Development Tools are found in the `\salvo\tut\tu1-6\syss` directories. The include path for each of these projects includes `\salvo\tut\tu1\syss`, and each project defines the `SYSS` symbol.

Complete projects using Salvo freeware libraries are contained in the project files `\salvo\tut\tu1-6\syss\tu1-6lite.prj`. These projects also define the `MAKE_WITH_FREE_LIB` symbol.

Complete projects using Salvo standard libraries are contained in the project files `\salvo\tut\tu1-6\syss\tu1-6le.prj`. These projects also define the `MAKE_WITH_STD_LIB` symbol.

Complete projects using Salvo standard libraries with embedded debugging information are contained in the project files `\salvo\tut\tu1-6\syss\tu1-6prolib.prj`. These projects also define the `MAKE_WITH_STD_LIB` symbol.

Complete projects using Salvo source code are contained in the project files `\salvo\tut\tu1-6\syss\tu1-6pro.prj`. These projects also define the `MAKE_WITH_SOURCE` symbol.

---

1 Since folders cannot be deleted, you should rename the default `Files`, `Headers` and `Documents` folders to `Listings`, `Salvo Configuration File` and `Salvo Help Files`, respectively.

2 ICC430 also supports pathnames relative to the project's home directory. Using relative pathnames is recommended, as it makes a project much more portable.

3 This Salvo project supports a wide variety of targets and compilers. For use with ICC430 Development Tools, it requires the `SYST` defined symbol, as well as the symbols `MAKE_WITH_FREE_LIB` or `MAKE_WITH_STD_LIB` for library builds. When you write your own projects, you may not require any symbols.

4 This Salvo Lite library contains all of Salvo's basic functionality. The corresponding Salvo LE and Pro libraries are `libslicc430-a.a` and `libslicc430ia.a`, respectively.

5 You can Ctrl-select multiple files at once.

6 We recommend that you add the project's map file to your project's `Listings` group.

7 This is usually done in the computer's BIOS.