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

Building a Salvo Application with IAR's MSP430 C Compiler and Embedded Workbench IDE

Introduction

This Application Note explains how to use IAR's (http://www.iar.com/) MSP430 C compiler and Embedded Workbench IDE to create a multitasking Salvo application for Texas Instruments' (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 IAR Embedded Workbench for MSP430.

Note IAR Embedded Workbench underwent substantial changes between v1 (EW 2.3, with the last version of the compiler being v1.25B) and v2 (EW 3.2, with v2.x compilers, e.g. v2.10A). The procedures and illustrations in this document are from IAR Embedded Workbench for MSP430 v1 and the associated IAR MSP430 C compiler v1.25B. Where substantive differences exist, they will be noted as such.

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 IAR Embedded Workbench for the MSP430. If necessary, you should also upgrade to the latest TI MSP430 simulator, available from the TI website 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 IAR's MSP430 C compiler:

Salvo User Manual
Salvo Compiler Reference Manual RM-IAR430

Creating and Configuring a New Project

Create a new Embedded Workbench project under File → New → Project → OK. Select MSP430 as the Target CPU Family, navigate to your working directory (in this case we've chosen `c:\temp`) and create a project named `myex1.prj`:

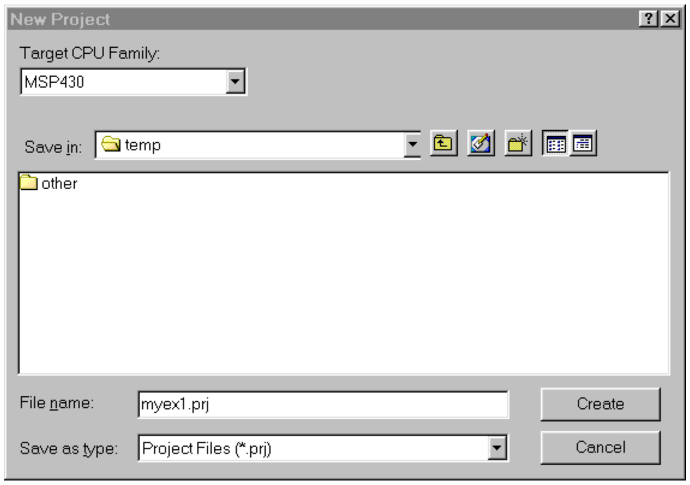


Figure 1: Creating the New Project

Click Create to continue. Choose File → Save to save the project.

Note In Embedded Workbench for MSP430 v2, first you create a *workspace*, and then you create one or more projects within that workspace.

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

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

For each group, choose Project → New Group, add in the Group Name and select OK.

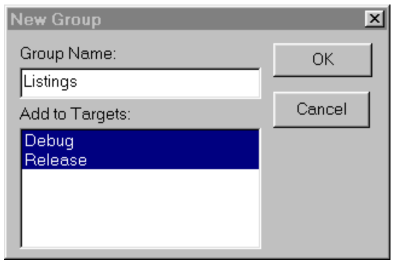


Figure 2: Creating a Group

When finished, your project window should look like this:

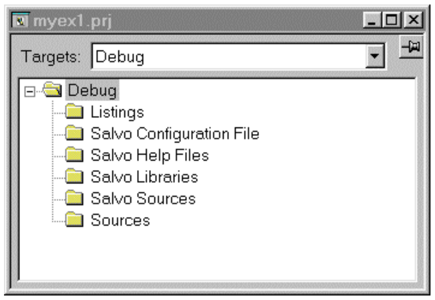
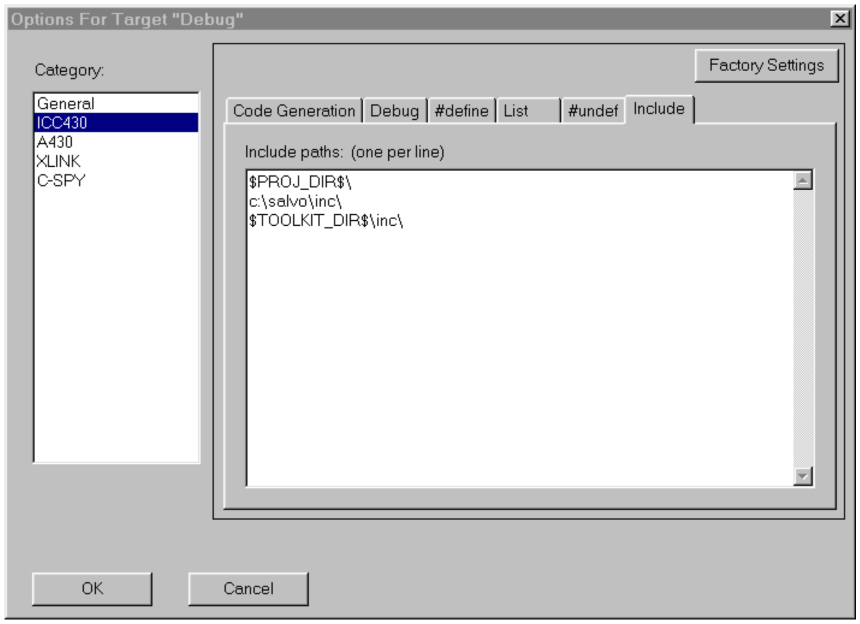


Figure 3: Project Window with Groups

Now let's setup the project's options for Salvo's pathnames, etc. for your particular MSP430 microcontroller. Select Project → Options → ICC430 → #define and define any symbols you may need for your project.[1] Select Project → Options → ICC430 → Include and add the include paths `$PROJ_DIR$\` and `c:\salvo\inc\`:

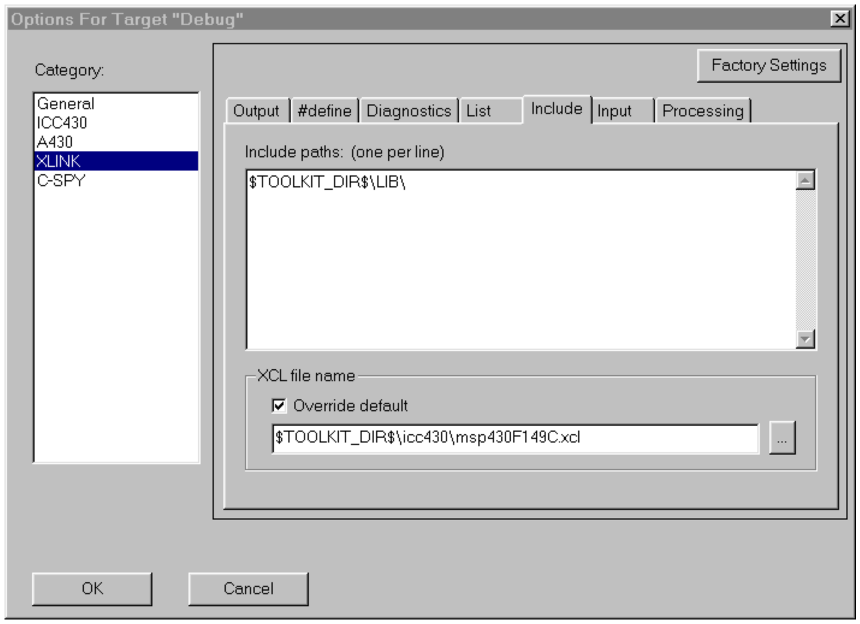


Include paths: (one per line)

```
$PROJ_DIR$\
c:\salvo\inc\
$TOOLKIT_DIR$\inc\
```

Figure 4: ICC430 Settings – Project Include Paths

Next, select XLINK → List → Generate Linker listing. This will create a useful `.map` file with the application's ROM and RAM requirements, etc. Under XLINK → Include, select XCL file name → Override default and select the `.xcl` linker filename[2] that matches your target processor.[3]



Include paths: (one per line)

```
$TOOLKIT_DIR$\LIB\
```

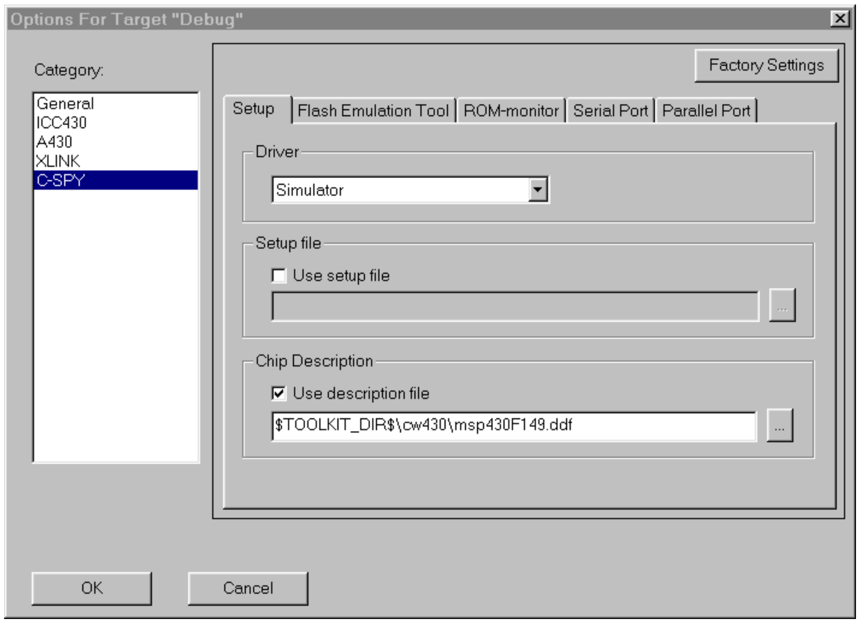
XCL file name

☑ Override default

```
$TOOLKIT_DIR$\icc430\msp430F149C.xcl
```

Figure 5: XLINK Settings – Project XCL File Name

Lastly, under C-SPY → Setup, select the Driver (Flash Emulation Tool, ROM Monitor or Simulator) and select Chip Description → Use description file and select the appropriate description file for your MSP430:



Driver: Simulator

Setup file
☐ Use setup file

Chip Description
☑ Use description file
$TOOLKIT_DIR$\cw430\msp430F149.ddf

Figure 6: C-SPY Settings – Project Chip Description File

Select OK to finish configuring your project.

Adding your Source File(s) to the Project

Now it's time to add files to your project. Choose Project → Files, C/C++ Source Files (*.c,*.cpp,*.cc) under Files of type, select Sources under Add to Group, navigate to your project's directory, select your `main.c` and Add. Your Project Files window should look like this:

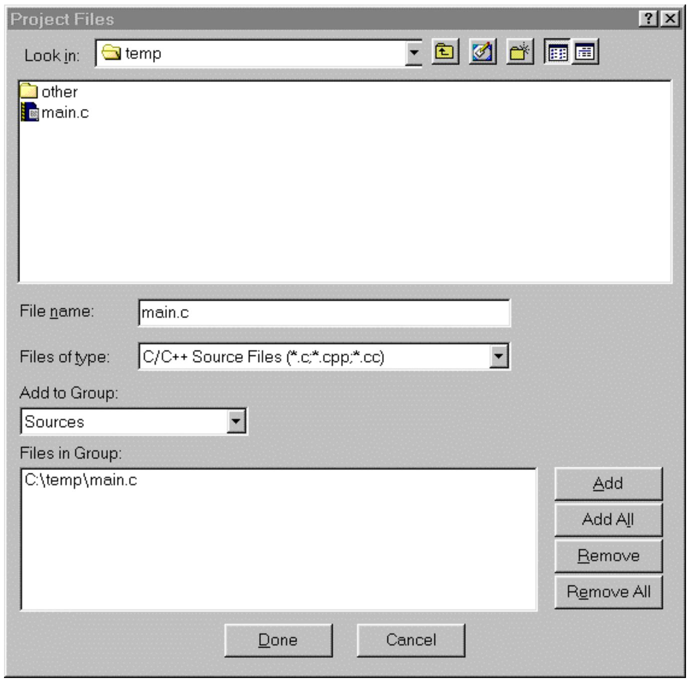


Figure 7: Project Files Window

When finished, select Done, and your project window should look like this:

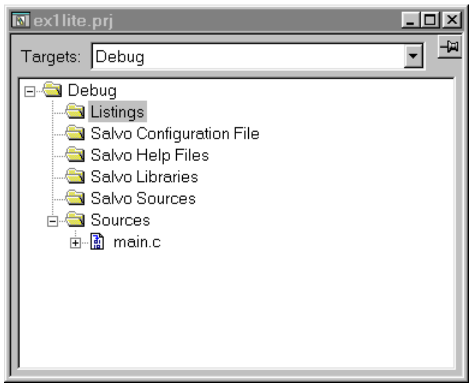


Figure 8: Project Window with Project-Specific Source Files

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 is `sfiar430-a.r43`.[4] Select Project → Files, Library/Object Files (*.r*) under Files of type, Salvo Libraries under Add to Group, navigate to the `\salvo\lib\iar430-v1` directory, select `sfiar430-a.r43` and Add:

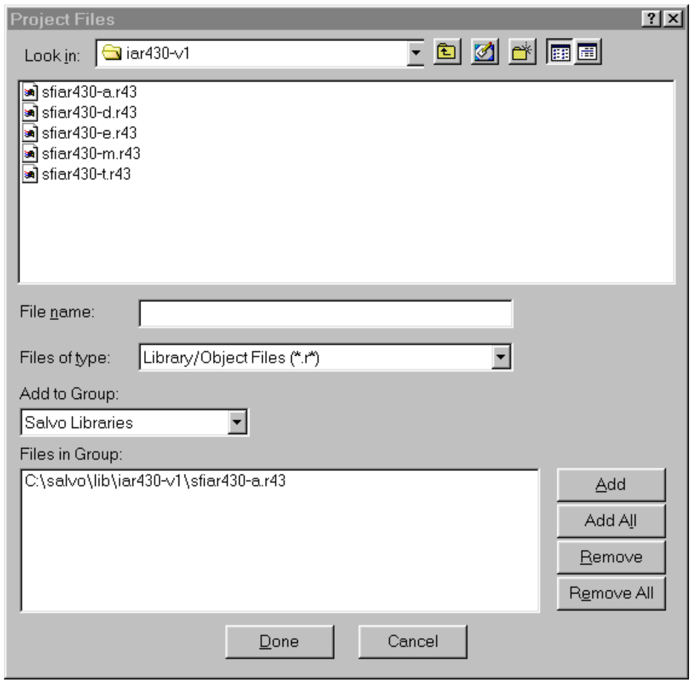


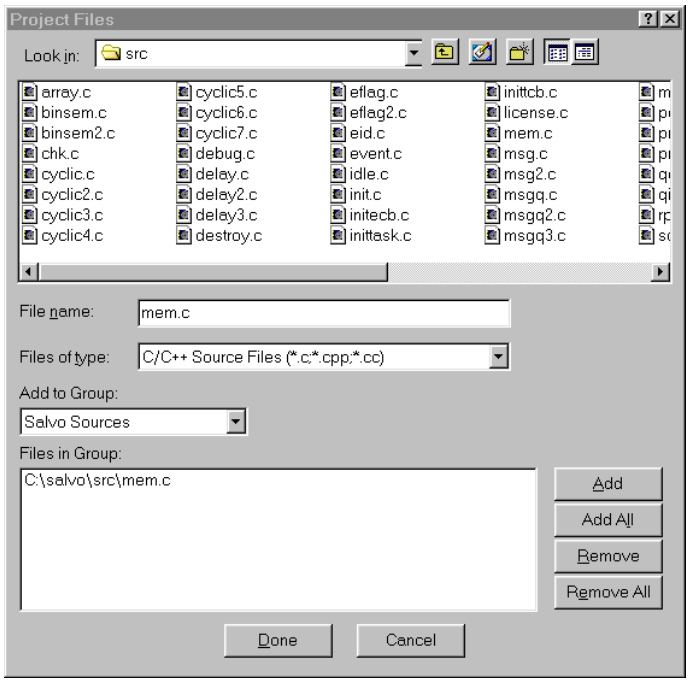
Figure 9: Adding the Library to the Project

Note Salvo libraries for IAR's v1.x MSP430 C compilers are located in `\salvo\lib\iar430-v1`. Libraries for v2.x compilers are located in `\salvo\lib\iar430-v2`. The libraries are not interchangeable.

Select Done 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-IAR430*.

Adding Salvo's mem.c

Salvo library builds also require Salvo's `mem.c` source file as part of each project. Choose Project → Files, C/C++ Source Files (*.c,*.cpp,*.cc) under Files of type, select Salvo Sources under Add to Group, navigate to `\salvo\src`, select `mem.c` and Add. Your Project Files window should look like this:



File name: mem.c
Files of type: C/C++ Source Files (*.c;*.cpp;*.cc)
Add to Group: Salvo Sources
Files in Group: C:\salvo\src\mem.c

Figure 10: Project Files Window

When finished, select Done, and your project window should look like this:

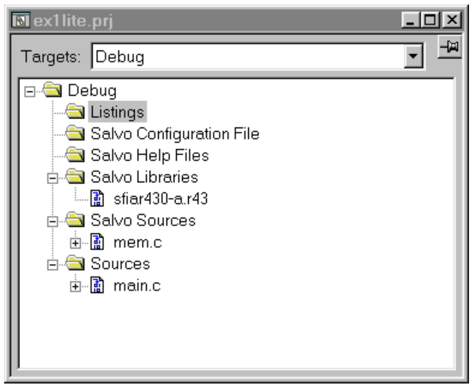


Figure 11: Project Window with Library, mem.c and User Source File(s)

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

Select Project → Files, All Files (*.*) under Files of type, Salvo Configuration File under Add to Group, navigate to your project's directory, select `salvocfg.h` and Add:

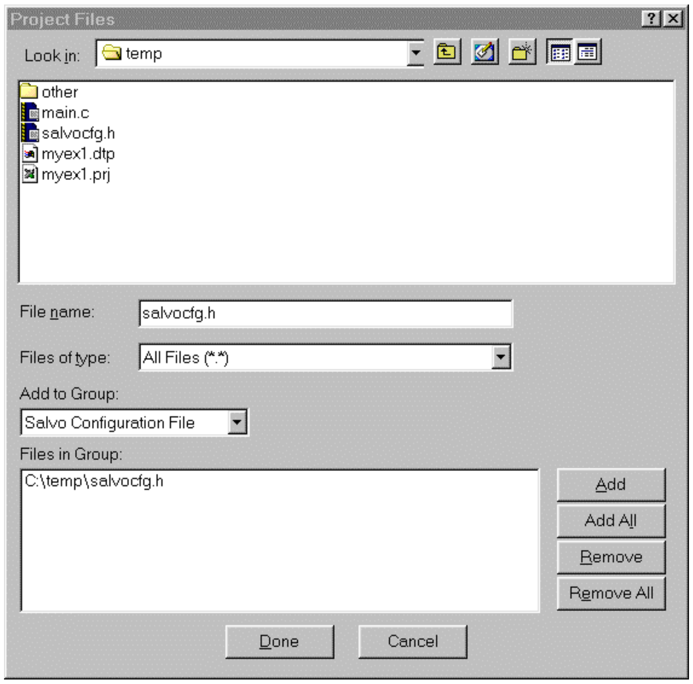


Figure 12: Adding the Configuration File to the Project

Your project window should now look like this:

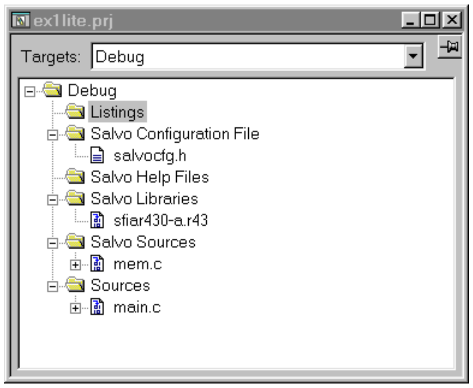


Figure 13: Project Window for a Library Build

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 editing, etc.

Proceed to *Building the Project*, below.

Adding Salvo Source Files

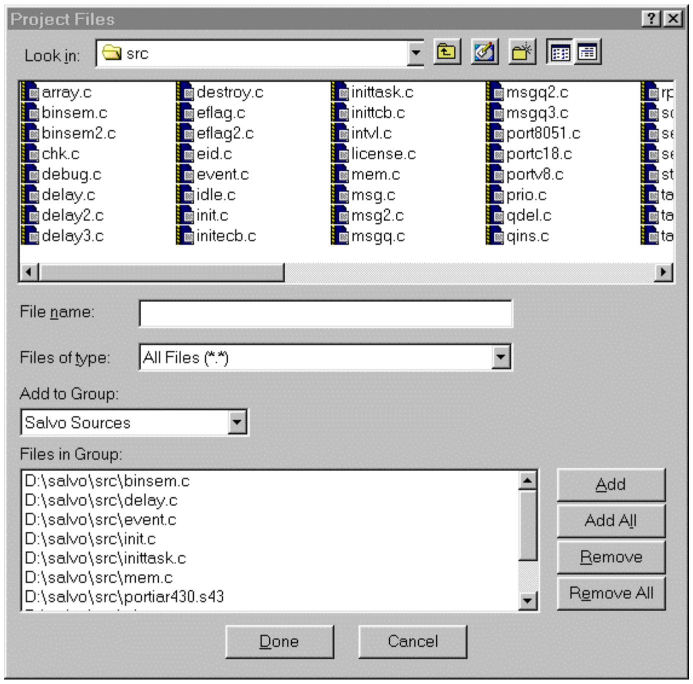
If you have a Salvo distribution that contains source files, you can do a *source code build* instead of a library build. The application in `\salvo\ex\ex1\main.c` contains calls to the following Salvo user services:

```
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                 portiar430.s43
event.c                 qins.c
idle.c                  sched.c
init.c                  timer.c
inittask.c
```

To add these files to your project, select Project → Files, All Files (*.*) under Files of type, Salvo Sources under Add to Group:, navigate to the `\salvo\src` directory, select[5] the files listed above and Add:



File name:

Files of type: All Files (*.*)

Add to Group: Salvo Sources

Files in Group:
D:\salvo\src\binsem.c
D:\salvo\src\delay.c
D:\salvo\src\event.c
D:\salvo\src\init.c
D:\salvo\src\inittask.c
D:\salvo\src\mem.c
D:\salvo\src\portiar430.s43

Figure 14: Adding Salvo Source Files to the Project

Select Done when finished. Your project window should now look like this:

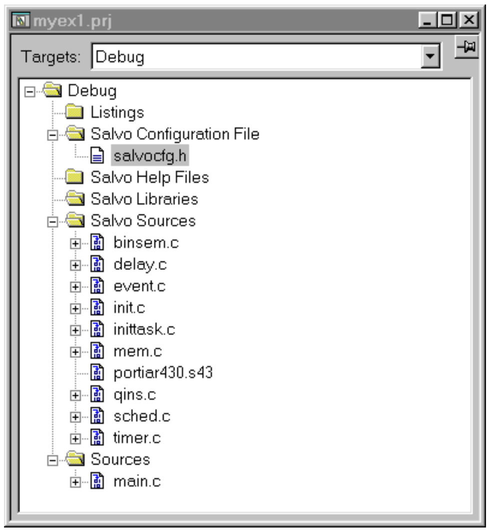


Figure 15: Project Window for a Source Code Build

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

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 or Project → Build All. The build results can be seen in the map file located in the project's `Debug\List` subdirectory:^{6}

```
################################################################################
#                                                                              #
#      IAR Universal Linker V4.53G/WIN                                         #
#                                                                              #
#           Link time     =  17/Apr/2002  10:33:44                             #
#           Target CPU    =  MSP430                                            #
#           List file     =  C:\temp\Debug\List\myex1.map                      #
#           Output file 1 =  C:\temp\Debug\Exe\myex1.d43                       #
#                            Format: debug                                     #
#                            UBROF version 6.0.0                               #
#                            Using library modules for C-SPY (-rt)             #
#           Command line  =  C:\temp\Debug\Obj\binsem.r43                      #
#                            C:\temp\Debug\Obj\delay.r43                       #
#                            C:\temp\Debug\Obj\event.r43                       #
#                            C:\temp\Debug\Obj\init.r43                        #
#                            C:\temp\Debug\Obj\inittask.r43                    #
#                            C:\temp\Debug\Obj\mem.r43                         #
#                            C:\temp\Debug\Obj\portiar430.r43                  #
#                            C:\temp\Debug\Obj\qins.r43                        #
#                            C:\temp\Debug\Obj\sched.r43                       #
#                            C:\temp\Debug\Obj\timer.r43                       #
#                            C:\temp\Debug\Obj\main.r43 -o                     #
#                            C:\temp\Debug\Exe\myex1.d43 -rt -l                #
#                            C:\temp\Debug\List\myex1.map -xms                 #
#                            -IC:\IAR\EW23\430\LIB\ -f                         #
#                            C:\IAR\EW23\430\icc430\msp430F123C.xcl            #
#                            (-cMSP430 -Z(DATA)UDATA0,IDATA0,ECSTR=0200-02FF   #
#                            -Z(DATA)CSTACK#0200-0300 -Z(CODE)INFO=1000-10FF   #
#                            -Z(CODE)CODE,CONST,CSTR,CDATA0,CCSTR=E000-FFDF    #
#                            -Z(CODE)INTVEC=FFE0-FFFF                          #
#                            -e_small_write=_formatted_write                   #
#                            -e_medium_read=_formatted_read cl430.r43)         #
#                                                                              #
#                        Copyright 1987-2001 IAR Systems. All rights reserved. #
################################################################################

[SNIP]
                ****************************************
                *                                      *
                *      SEGMENTS IN ADDRESS ORDER       *
                *                                      *
                ****************************************


SEGMENT              SPACE    START ADDRESS    END ADDRESS      SIZE  TYPE  ALIGN
=======              =====    =============    ===========      ====  ====  =====
UDATA0                                 0200 - 022B                2C   rel    1
ECSTR                                     022C                         rel    1
IDATA0                                    022C                         rel    1
CSTACK                                    0300                         rel    1
INFO                                      1000                         dse    0
CODE                                   E000 - E551               552   rel    1
CSTR                                      E552                         dse    0
CDATA0                                    E552                         rel    1
CCSTR                                     E552                         rel    1
CONST                                     E552                         dse    0
INTVEC                                 FFE0 - FFFF                20   com    1

                ****************************************
                *                                      *
                *        END OF CROSS REFERENCE        *
                *                                      *
                ****************************************

 1 394 bytes of CODE memory
    44 bytes of DATA memory

Errors: none
Warnings: none
```

Listing 3: Source Code Build Results (Abbreviated)

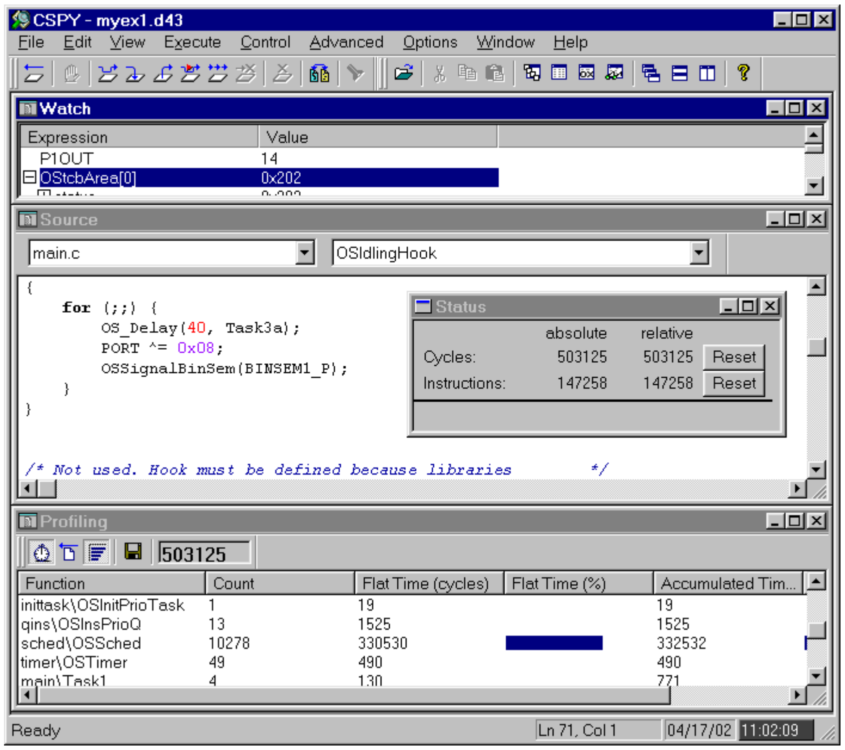
Note The Embedded Workbench for MSP430 projects supplied in the Salvo for TI's MSP430 distributions contain additional help files in each project's Salvo Help Files group.

Tip If you configure Embedded Workbench to display the memory utilization for individual source files and the complete application you won't have to look in the map file. Select Options → Settings → Make Control → Message Filtering Level and choose All.

Testing the Application

You can test and debug this application using the C-SPY debugger and either the simulator or the Flash Emulation Tool. To launch C-SPY, choose Project → Debugger.

You can use all of C-SPY's supported features when debugging and testing Salvo applications. This includes breakpoints, profiling, intelligent watch window, cycle counting, etc.



Expression	Value
P1OUT	14
OStcbArea[0]	0x202

main.c — OSIdlingHook

```
{
    for (;;) {
        OS_Delay(40, Task3a);
        PORT ^= 0x08;
        OSSignalBinSem(BINSEM1_P);
    }
}

/* Not used. Hook must be defined because libraries         */
```

	absolute	relative
Cycles:	503125	503125
Instructions:	147258	147258

503125

Function	Count	Flat Time (cycles)	Flat Time (%)	Accumulated Tim...
inittask\OSInitPrioTask	1	19		19
qins\OSInsPrioQ	13	1525		1525
sched\OSSched	10278	330530	▬	332532
timer\OSTimer	49	490		490
main\Task1	4	130		771

Ready   Ln 71, Col 1   04/17/02 11:02:09

Figure 16: Testing a Salvo Application in C-SPY

Note C-SPY supports debugging at the source code level. 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. Regardless of how you build your Salvo application, you can always step through your own C and assembly code in C-SPY.

Migrating to Embedded Workbench for MSP430 v2

Existing Salvo applications built as projects (`*.pjt`) under IAR's Embedded Workbench for MSP430 v1 can be migrated to v2 using the following steps.

- In Embedded Workbench for MSP430 v2, choose File → New → Workspace to create a new workspace file (`*.eww`).
- Choose File → Insert Project into the Workspace…, select Files of type: Old Project Files (*.prj), navigate to the old project and select Open, then OK.
- Under Project → Options, select the device (e.g. MSP430F149) under General → Target → Device. Set the desired optimizations under ICC430 → Code → Optimizations. Under XLINK → Include, ensure that the XCL file name is not overridden and/or a valid filename is used.
- Remove the existing Salvo library from the project, and replace it with a same-named one from `\salvo\lib\iar430-v2`.

When finished, the new project window will look like this:

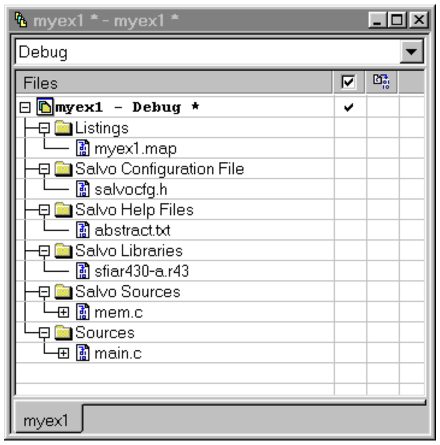
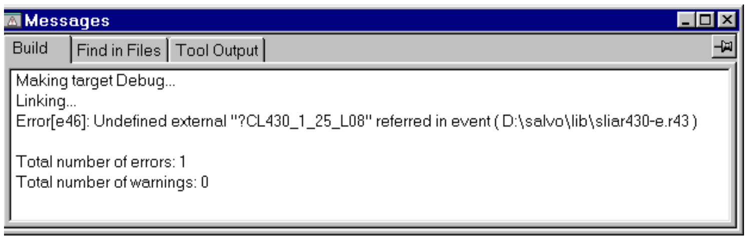


Figure 17: Project Window for a Library Build in Embedded Workbench for MSP430 v2

Troubleshooting

Linker Error: Undefined External (version number)

If you are doing a library build and your version of the IAR MSP430 C Compiler is different from that used to build the Salvo libraries, the linker will issue an error like this one:[7]



Messages

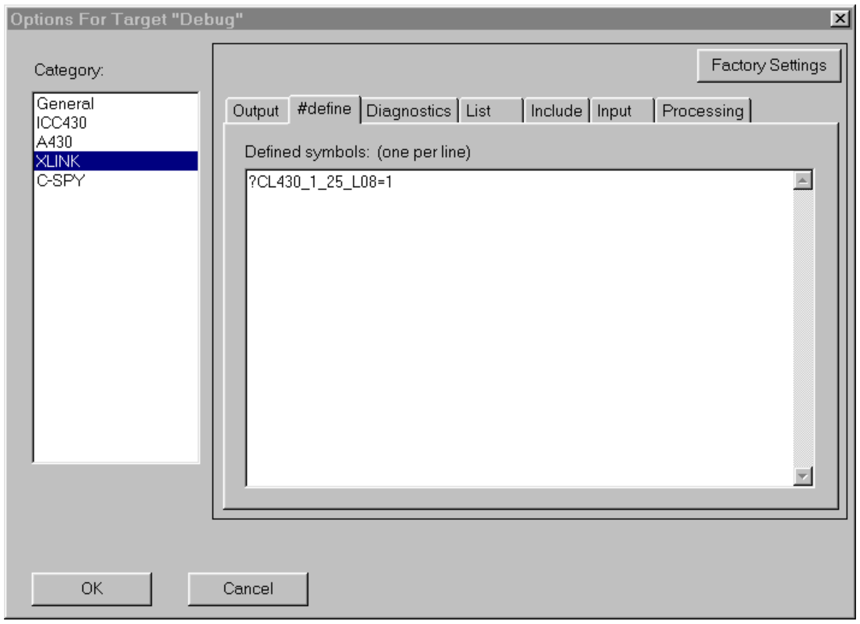
Build | Find in Files | Tool Output

```
Making target Debug...
Linking...
Error[e46]: Undefined external "?CL430_1_25_L08" referred in event ( D:\salvo\lib\sliar430-e.r43 )

Total number of errors: 1
Total number of warnings: 0
```

Figure 18: Linker Error due to Version Mismatch

This type of error occurs because the Salvo library – in this case, `sliar430-e.r43` – references an external symbol (here, `?CL430_1_25_L08`) which is not defined by the version of the compiler you are using. The library references this symbol because it was built with a different version of the IAR MSP430 C compiler. The solution is simply to define this symbol at link time by using Project → Options → XLINK → #define and then assigning a value of 1 to the symbol:



Options For Target "Debug"

Category: General, ICC430, A430, XLINK, C-SPY

Defined symbols: (one per line)

```
?CL430_1_25_L08=1
```

Figure 19: Setting the Link-time Version Number External Symbol

Once this symbol is defined, you'll be able to build your application successfully.

Note This solution should work as long as the major version number of the IAR MSP430 C Compiler you're using matches that used to generate the Salvo libraries. E.g. v1.23A can be used with Salvo libraries built with v1.26A.[8]

Version mismatches like this will occur whenever Salvo users and the Salvo for TI's MSP430 distribution are at different versions of the IAR MSP430 C compiler. While this solution is unlikely to cause any problems, we *strongly recommend that Salvo users keep their IAR MSP430 C compiler up-to-date* to avoid any potential difficulties.

This type of linker error will not happen with source code builds, e.g. when using Salvo Pro to build an application using the Salvo source files as project nodes instead of linking to a Salvo library.

Application Crashes After Changing Processor Type

Remember to `#include` the appropriate header file for your MSP430 variant (see *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.

Example Projects

Example projects for IAR's MSP430 C compiler can be found in the `\salvo\tut\tu1-6\sysq` directories. The include path for each of these projects includes `\salvo\tut\tu1\sysq`, and each project defines the `SYSQ` symbol.

Complete projects using Salvo freeware libraries are contained in the project files `\salvo\tut\tu1-6\sysq\tu1-6lite.*`. 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\sysq\tu1-6le.*`. 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\sysq\tu1-6pro.*`. These projects also define the `MAKE_WITH_SOURCE` symbol.

Note Tutorial projects are provided for IAR Embedded Workbench for MSP430 v1 (`*.prj` files) IAR Embedded Workbench for MSP430 v2 (`*.ewp` & `*.eww` files).

---

[1] This Salvo project supports a wide variety of targets and compilers. For use with IAR's MSP430 compiler, it requires the `SYSQ` 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.

2 `.xcl` filenames ending in '`C`' appear to be for C-language projects. Those ending in '`A`' appear to be for assembly-language projects.

3 We recommend using the Embedded Workbench's argument variables like $PROJ_DIR$ and $TOOLKIT_DIR$ whenever possible.

4 This Salvo Lite library contains all of Salvo's basic functionality. The corresponding Salvo LE and Pro libraries are sliar430-a.r43 and sliar430ia.r43, 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 example was generated using the IAR MSP430 C Compiler v1.26A, with Salvo LE for TI's MSP430 v3.0.3, which was built using v1.25A. Hence the `_1_25_` (for v1.25) in the undefined external symbol.

8 In this example, the major version number is 1.