



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

RM-RA430 Reference Manual

Salvo Compiler Reference Manual – CrossWorks for MSP430



Salvo™

The RTOS that runs in tiny places.™

Introduction

This manual is intended for Salvo users who are targeting TI's MSP430 ultra-low-power single-chip microcontroller with Rowley Associates' (<http://www.rowley.co.uk/>) CrossWorks for MSP430 C compiler.

Related Documents

The following Salvo documents should be used in conjunction with this manual when building Salvo applications with Rowley Associates' CrossWorks for MSP430 C compiler:

- *Salvo User Manual*

Example Projects

Example Salvo projects for use with Rowley Associates' CrossWorks for MSP430 C compiler and the CrossStudio IDE can be found in the:

```
\Pumpkin\Salvo\Example\MSP430\MSP430x1xx  
\Pumpkin\Salvo\Example\MSP430\MSP430x4xx
```

directories of every Salvo for TI's MSP430 distribution.

Tip These example projects can be easily modified for any MSP430 device.

Features

Table 1 illustrates important features of Salvo's port to Rowley Associates' CrossWorks for MSP430 C compiler.

General	
Abbreviated as	RA430
Available distributions	Salvo Lite, LE & Pro for TI's MSP430
Supported targets	all MSP430 and MSP430X devices
Header file(s)	salvoportra430.h
Other target-specific file(s)	salvoportra430.asm
salvocfg.h	
Compiler auto-detected?	yes ¹
Include target-specific header file in salvocfg.h?	yes, use #include <msp430.h>
Libraries	
Located in	Lib\RA430-v1 (for v1.x compilers) Lib\RA430-v2 (for v2.x compilers)
Context Switching	
Method	function-based via OSDispatch() & OSCtxSw()
Labels required?	no
Size of auto variables and function parameters in tasks	total size must not exceed 255 8-bit bytes
Interrupts	
Interrupt latency in context switcher	0 cycles
Interrupts in critical sections controlled via	user hooks
Default behavior in critical sections	see example user hooks
Debugging	
Source-level debugging with Pro library builds?	yes
Compiler	
Bitfield packing support?	no
printf() / %p support?	yes / yes
va_arg() support?	yes

Table 1: Features of Salvo port to Rowley Associates' CrossWorks for MSP430 C compiler

Libraries

Nomenclature

The Salvo libraries for Rowley Associates' CrossWorks for MSP430 C compiler follow the naming convention shown in Figure 1.

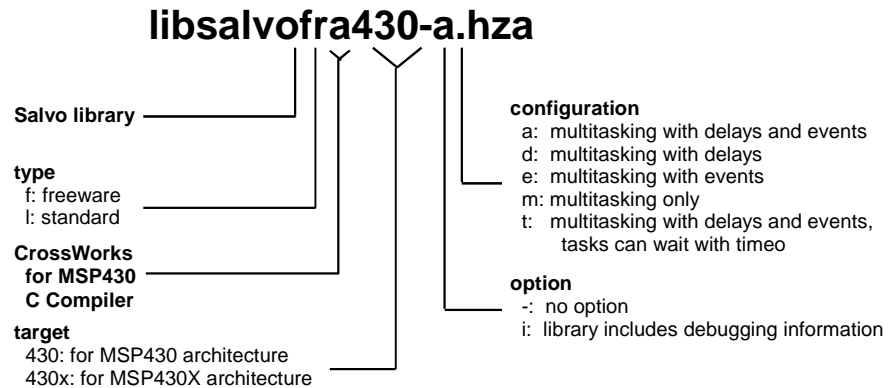


Figure 1: Salvo library nomenclature – Rowley Associates' CrossWorks for MSP430 C compiler

Note Each successive version of Rowley Associates' CrossWorks for MSP430 C compiler *uses different library formats*. Therefore independent sets of Salvo libraries are available for each compiler Table 1, above.

Type

Salvo Lite distributions contain *freeware* libraries. All other Salvo distributions contain *standard* libraries. See the *Libraries* chapter of the *Salvo User Manual* for more information on library types.

Target

When targeting the MSP430X (e.g. MSP430FG4619) and wishing to use MSP430X extensions (i.e. 20-bit address space, etc.), 430x Salvo libraries must be selected. 430 Salvo libraries should be used on the MSP430 (e.g. MSP430F1611).

Option

Salvo Pro users can select between two sets of libraries – standard libraries, and standard libraries incorporating source-level debugging information. The latter have been built with Rowley Associates' CrossWorks for MSP430 C compiler's `-g` command-line option. This adds source-level debugging information to the libraries, making them ideal for source-level debugging and stepping in the CrossStudio debugger. To use these libraries, simply select one that includes the debugging information (e.g. `libsalvolra430it.hza`) instead of one without (e.g. `libsalvolra430-t.hza`) in your CrossStudio project.

Configuration

Different library configurations are provided for different Salvo distributions and to enable the user to minimize the Salvo kernel's footprint. See the *Libraries* chapter of the *Salvo User Manual* for more information on library configurations.

Build Settings

Salvo's libraries for Rowley Associates' CrossWorks for MSP430 C compiler are built using the default settings outlined in the *Libraries* chapter of the *Salvo User Manual*. Target-specific settings and overrides are listed in Table 2.

Target-specific Settings	
Delay sizes	8 bits
Idling hook	dummy, can be overridden
Interrupt hook	disables then restores GIE bit, can be overridden
Watchdog hook	clears WDT without other changes, can be overridden
System tick counter	available, 32 bits
Task priorities	enabled

Table 2: Build settings and overrides for Salvo libraries for Rowley Associates' CrossWorks for MSP430 C compiler

Note Salvo Lite libraries have precompiled limits for the number of supported tasks, events, etc. Salvo LE and Pro libraries have no such limits. See the *Libraries* chapter of the *Salvo User Manual* for more information.

Available Libraries

Salvo Lite for TI's MSP430 contains a single freeware library. Salvo LE for TI's MSP430 adds standard libraries in all configurations. Salvo Pro for TI's MSP430 adds standard libraries in all configurations with debugging information included. Each Salvo for TI's MSP430 distribution contains the Salvo libraries of the lesser distributions beneath it.

Target-Specific Salvo Source Files

The source file `salvoportra430.asm` is needed for Salvo Pro source-code builds. It automatically detects whether the target is part of the MSP430 or MSP430X architecture.

salvocfg.h Examples

Below are examples of `salvocfg.h` project configuration files for various different Salvo distributions and the MSP430F1612.

Salvo Lite Library Build

```
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE        OSF
#define OSLIBRARY_CONFIG      OST
#define OSTASKS                2
#define OSEVENTS              4
#define OSEVENT_FLAGS         0
#define OSMESSAGE_QUEUES      1
```

Listing 1: Example `salvocfg.h` for library build using `libsalvofra430-t.hza`

Salvo LE & Pro Library Build

```
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE        OSL
#define OSLIBRARY_CONFIG      OST
#define OSTASKS                7
#define OSEVENTS              13
#define OSEVENT_FLAGS         3
#define OSMESSAGE_QUEUES      2
```

Listing 2: Example `salvocfg.h` for library build using `libsalvolra430-t.hza` or `libsalvolra430it.hza`

Salvo Pro Source-Code Build

```
#define OSENABLE_IDLING_HOOK    TRUE
#define OSENABLE_SEMAPHORES    TRUE
#define OSTASKS                9
#define OSEVENTS              17
#define OSEVENT_FLAGS         2
#define OSMESSAGE_QUEUES      4
```

Listing 3: Example `salvocfg.h` for source-code build

Performance

Interrupt Latencies

Since Salvo's context switcher for Rowley Associates' CrossWorks for MSP430 C compiler does not need to control interrupts, Salvo applications can easily be created with zero total interrupt latency for interrupts of interest.

In a properly-configured application, only those interrupts that call Salvo services will experience interrupt latency from Salvo's operations. Users must ensure that these interrupt sources are disabled (and re-enabled) via the user interrupt hooks.

Disabling and re-enabling interrupts globally in the user interrupt hooks (i.e., the default user interrupt hook behavior) is of course permitted, but will result in non-zero interrupt latencies for all interrupt sources, even those that do not call Salvo services. See the target-specific source files of this distribution for examples.

Memory Usage

Example Application ²	Program Memory Usage ³	Data Memory Usage ⁴
\\MSP430x1xx\\...\\tut5lite	1788	61
\\MSP430x1xx\\...\\tut5le	1756	61
\\MSP430x1xx\\...\\tut5pro	1964	61

Table 3: ROM and RAM requirements for Salvo applications built with Rowley Associates' CrossWorks for MSP430 C compiler

User Hooks

Overriding Default Hooks

In library builds, users can define new hook functions in their projects and the linker will choose the user function(s) over the default function(s) contained in the Salvo library.

In source-code builds, users can remove the default hook file(s) from the project and substitute their own hook functions.

Idling

The default idling hook in `salvohook_idle.c` is a dummy function, as shown below.

```
void OSIdlingHook ( void )
{
    ;
}
```

Listing 4: Default Salvo idling hook for Rowley Associates' CrossWorks for MSP430 C compiler

Users can replace it (e.g. with a directive to put the FM430 to sleep) by building their own version with their application.

Interrupt

The default interrupt hooks in `salvohook_interrupt_RA430_GIE.c` are shown below.⁵

```
static unsigned s;

void OSDisableHook(void)
{
    s = __disable_interrupt();
}

void OSEnableHook(void)
{
    __set_interrupt(s);
}
```

Listing 5: Default Salvo interrupt hooks for Rowley Associates' CrossWorks for MSP430 C compiler

These functions clear the `GIE` bit (i.e. disable global interrupts) across Salvo's critical section, and restore the `GIE` bit to its pre-critical-section value thereafter. Interrupts are *not* re-enabled inside of ISRs with these functions, thereby avoiding unnecessary or unwanted interrupt nesting. Therefore these default interrupt hooks are suitable for *all* applications.

For greater runtime performance, users can replace these functions with targeted interrupt-control functions by building their own version with their application. For example, if the Salvo service `OSTimer()` is the *only* Salvo service called from the foreground (i.e. interrupt) level, *and* it's called (only) via the `TimerA0` ISR, then the following functions to disable and re-enable `TimerA0`'s interrupt enable bit are all that is required, e.g.:


```
void OSDisableHook ( void )
{
    TACCTL0 &= ~CCIE;
}

void OSEnableHook ( void )
{
    TACCTL0 |= CCIE;
}
```

Listing 6: Example of user interrupt hooks where only TimerA0 ISR calls Salvo services

With the hooks shown in Listing 6, only the TimerA0 interrupt is disabled during Salvo's critical sections. All other interrupts sources (and the GIE bit) are untouched. This allows other interrupts that do not call Salvo services to proceed with *zero interrupt latency* due to Salvo.

Warning Not disabling all source of interrupts that call Salvo services during critical sections will cause the Salvo application to fail.

Watchdog

The default watchdog hook in `salvohook_wdt_RA430_CLRWDT.c` is shown below.⁶

```
void OSClrWDTHook ( void )
{
    WDTCTL = (WDTCTL & 0x00FF) | WDTPW | WDTCNTCL;
}
```

Listing 7: Default Salvo watchdog hook for Rowley Associates' CrossWorks for MSP430 C compiler

Users can replace it (e.g. with a dummy function – this would stop Salvo from clearing the watchdog timer and allow the user to clear it elsewhere) by building their own version with their application.

Compiler Issues

Incompatible Optimizations

Register Allocation

The Register Allocation property should be set to None in Salvo projects. Otherwise you may encounter problems in tasks that contain multiple switch statements and/or other complex structures.

IDE Issues

Salvo Libraries

In version 1.4 of the CrossWorks for MSP430 IDE, Salvo libraries are not automatically identified as libraries, and hence their contents are missed by the linker. To rectify this problem,

- Select the library in the Project Explorer
- Right-click to get the Properties window
- Select the Common configuration
- Set the file type of the Salvo library to be Library

¹ This is done automatically through the `__CROSSWORKS_MSP430` symbol defined by the compiler.

² Salvo 4.1.0-rc0 with v1.4 compiler.

³ In bytes. Salvo code only.

⁴ In bytes. Salvo objects only.

⁵ This hook is valid for all MSP430 and MSP430X targets.

⁶ This hook is valid for MSP430 and MSP430X targets because the watchdog control register is the same for all targets.