PUMPKIN™
REAL-TIME SOFTWARE

RM-MCC32
Reference Manual

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

Salvo Compiler Reference Manual – Microchip MPLAB C32

Salvo™
The RTOS that runs in tiny places.™

created by Andrew E. Kalman on Oct 30, 2007 updated on Nov 20, 2007

Introduction

This manual is intended for Salvo users who are targeting Microchip PIC32 single-chip microcontrollers with Microchip's (http://www.microchip.com/) MPLAB C32 C compiler and MPLAB IDE development suite.

Related Documents

The following Salvo documents should be used in conjunction with this manual when building Salvo applications with Microchip MPLAB C32:

Salvo User Manual

Example Projects

Example Salvo projects for use with Microchip MPLAB C32 can be found in the:

```
\Pumpkin\Salvo\Example\PIC\PIC32
```

directories of every Salvo for PIC32 distribution.

Features

Table 1 illustrates important features of Salvo's port to Microchip MPLAB C32.

General	
Abbreviated as	MCC32
Available distributions	Salvo Lite, LE & Pro for PIC32
Supported targets	all PIC32 devices
Header file(s)	`salvoportmcc32.h`
Other target-specific file(s)	`salvoportmcc32.S`
salvocfg.h	
Compiler auto-detected?	yes[1]
Include target-specific header file in `salvocfg.h`?	recommended
Libraries	
Located in	Lib\MCC32
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 65,535 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 Microchip MPLAB C32 C compiler

Libraries

Nomenclature

The Salvo libraries for Microchip MPLAB C32 follow the naming convention shown in Figure 1.

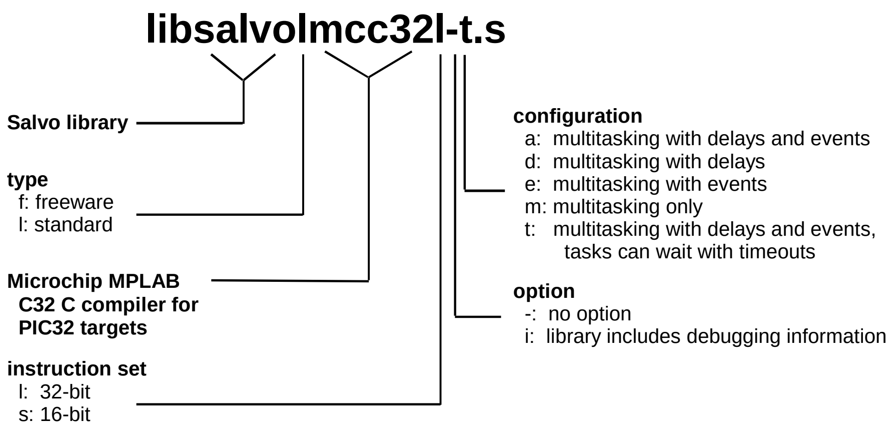
libsalvolmcc32l-t.s
Salvo library
type
f: freeware
l: standard
Microchip MPLAB C32 C compiler for PIC32 targets
instruction set
l: 32-bit
s: 16-bit
configuration
a: multitasking with delays and events
d: multitasking with delays
e: multitasking with events
m: multitasking only
t: multitasking with delays and events, tasks can wait with timeouts
option
-: no option
i: library includes debugging information

Figure 1: Salvo library nomenclature – Microchip MPLAB C32 C compiler

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

Since the CPU instruction set is common to all target architectures based on the PIC32 core, all PIC32 targets use the same Salvo libraries.

Instruction Set

The PIC32's M4K core supports both 32-bit (default) and 16-bit instruction sets. When building Salvo applications, it is imperative that all files (source files and libraries) use the same instruction set.

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 the appropriate command-line options. This adds source-level debugging information to the libraries, making them ideal for source-level debugging and stepping in MPLAB. To use these libraries, simply select one that includes the debugging information (e.g. `libsalvolmcc32lit.a`) instead of one without (e.g. `libsalvolmcc32l-t.a`) in your MPLAB 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 Microchip MPLAB C32 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.

Compiled Limits	
Max. number of tasks	4
Max. number of events	8
Max. number of event flags	1
Max. number of message queues	1
Target-specific Settings	
Delay sizes	8 bits
Idling hook	dummy, can be overridden
Interrupt hook	dummy, can be overridden
Watchdog hook	dummy, can be overridden
System tick counter	available, 32 bits
Task priorities	enabled

Table 2: Build settings and overrides for Salvo libraries for Microchip MPLAB C32 C compiler

Note The compiled limits for tasks, events, etc. in Salvo libraries can be overridden to be less (all Salvo distributions) or more (all Salvo distributions except Salvo Lite) than the library default. See the *Libraries* chapter of the *Salvo User Manual* for more information.

Available Libraries

Salvo Lite for PIC32 contains two freeware libraries in a single configuration. Salvo LE for PIC32 adds standard libraries in multiple configurations. Salvo Pro for PIC32 adds standard libraries in multiple configurations with debugging information included.

Each Salvo for PIC32 distribution contains the Salvo libraries of the lesser distributions beneath it. Additionally, Salvo Pro distributions contain makefiles for all possible library configurations.

Target-Specific Salvo Source Files

salvoportmcc32.S

The source file `salvoportmcc32.S` is required for Salvo Pro source-code builds.

32-bit vs. 16-bit Mode

By default, `salvoportmcc32.S` is assembled for 32-bit mode.

salvocfg.h Examples

Below are examples of `salvocfg.h` project configuration files for various different Salvo distributions for PIC32.

Salvo Lite Library Build

```
#define OSUSE_LIBRARY              TRUE
#define OSLIBRARY_TYPE             OSF
#define OSLIBRARY_CONFIG           OST
#define OSTASKS                    3
#define OSEVENTS                   4
#define OSEVENT_FLAGS              0
#define OSMESSAGE_QUEUES           1
```

Listing 1: Example salvocfg.h for library build using libsalvofmcc32l-t.a

Salvo LE & Pro Library Build

```
#define OSUSE_LIBRARY              TRUE
#define OSLIBRARY_TYPE             OSL
```

```
#define OSLIBRARY_CONFIG          OSA
#define OSTASKS                   7
#define OSEVENTS                  11
#define OSEVENT_FLAGS             0
#define OSMESSAGE_QUEUES          4
```

Listing 2: Example salvocfg.h for library build using libsalvolmcc32l-t.a or libsalvolmcc32lit.a

Salvo Pro Source-Code Build

```
#define OSEVENTS                     9
#define OSEVENT_FLAGS                1
#define OSMESSAGE_QUEUES             2
#define OSTASKS                      17

#define OSENABLE_BINARY_SEMAPHORES   TRUE
#define OSENABLE_IDLING_HOOK         TRUE
#define OSENABLE_TIMEOUTS            TRUE
#define OSBYTES_OF_DELAYS            1
#define OSBYTES_OF_TICKS             4
```

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

Performance

Interrupt Latencies

Since Salvo's context switcher for Microchip MPLAB C32 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

Examples of the total memory usage of actual Salvo-based applications are listed below.

Example Application[2]	Program Memory Usage[3]	Data Memory Usage[4]
`tut5lite`	1609	123
`tut5le`	1572	123
`tut5pro`	1537	119

Table 3: Program and data memory requirements for Salvo applications built with Microchip MPLAB C32 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 Microchip MPLAB C32 C compiler

Users can replace it (e.g. with a directive to put the PIC32 to sleep) by adding their own version to their application.

Interrupt

The default interrupt hooks in `salvohook_interrupt.c` is a dummy function, as shown below.

```
void OSDisableHook(void)
{
  ;
}


void OSEnableHook(void)
{
  ;
}
```

Listing 5: Default Salvo interrupt hooks for Microchip MPLAB C32 C compiler

Users should replace these hooks by adding their own functions to either globally disable and re-enable all interrupt sources, or (preferred) disable and re-enable only those interrupt sources whose ISRs call Salvo services.

Note 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.c` is a dummy function, as shown below:

```
void OSClrWDTHook(void)
{
  ;
}
```

By replacing this hook with one that clears the watchdog timer, the watchdog timer will be cleared each time Salvo's scheduler is called.

1 This is done automatically through the `__PIC32MX__` symbols defined by the compiler.
2 Salvo 4.0.3.
3 In bytes.
4 In bytes. This represents all of Salvo's objects. Does not include RAM allocated to the heap or stack. Salvo applications typically require the same (small) stack size as simple, non-multitasking applications.