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

RM-ICCAVR
Reference Manual

Salvo Compiler Reference Manual – ImageCraft ICCAVR

Salvo™
The RTOS that runs in tiny places.™

Introduction

This manual is intended for Salvo users who are targeting Atmel (http://www.atmel.com/) AVR® and MegaAVR™ microcontrollers[1] with ImageCraft's (http://www.imagecraft.com/) ICCAVR C compiler.

Related Documents

The following Salvo documents should be used in conjunction with this manual when building Salvo applications with ImageCraft's ICCAVR C compiler:

Salvo User Manual
Application Note AN-24

Example Projects

Example Salvo projects for use with ImageCraft's ICCAVR C compiler and the ImageCraft IDE can be found in the:

```
\salvo\ex\ex1\sysv
\salvo\tut\tu1\sysv
\salvo\tut\tu2\sysv
\salvo\tut\tu3\sysv
\salvo\tut\tu4\sysv
\salvo\tut\tu5\sysv
\salvo\tut\tu6\sysv
```

directories of every Salvo for Atmel AVR and MegaAVR distribution.

Features

Table 1 illustrates important features of Salvo's port to ImageCraft's ICCAVR C compiler.

general	
available distributions	Salvo Lite, LE & Pro for Atmel AVR and MegaAVR
additional distributions	Salvo tiny & SE for Atmel AVR and MegaAVR & ICCAVR
supported targets	AVR and MegaAVR family
header file(s)	`porticcavr.h`
other target-specific file(s)	`porticcavr.s`, `porticcatmega.s`
project subdirectory name(s)	SYSV
salvocfg.h	
compiler auto-detected?	yes[2]
libraries	
`\salvo\lib` subdirectory	`iccavr`
context switching	
method	function- and label-based via `OSDispatch()` & `OSCtxSw(label)`
`_OSLabel()` required?	yes
size of auto variables and function parameters in tasks	total size must not exceed 254 8-bit bytes
memory & registers	
internal and external RAM supported?	yes, via `-bsalvoram:0xstart.0xend`
`R20..R23` used?	no
interrupts	
controlled via	`I` bit
interrupt status preserved in critical sections?	yes
method used	saved on stack via `#pragma monitor`
nesting limit	unlimited
alternate methods possible?	yes[3]
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 ImageCraft's ICCAVR C Compiler

Libraries

Nomenclature

The Salvo libraries for ImageCraft's ICCAVR C compiler follow the naming convention shown in Figure 1.

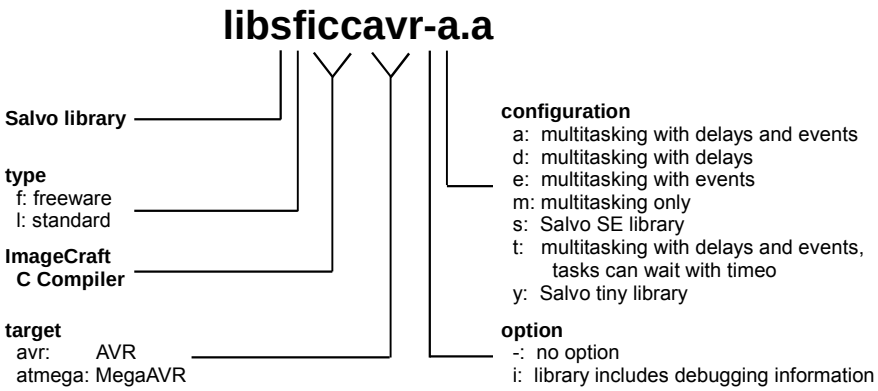

Figure 1: Salvo Library Nomenclature – ImageCraft's ICCAVR C Compiler

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

Each library is intended for one or more specific processors. Table 2 lists the correct library for each AVR and MegaAVR processor.

target code	processor(s)
`avr:`	AT90S2313, AT90S2323, AT90S2333, AT90S2343, AT90S4433, AT90S4434, AT90S4444, AT90S8515(A), AT90S8534, AT90S8535, ATmega8(L), ATtiny26
`atmega:`	AT94K05, AT94K10, AT94K40, ATmega103, ATmega128(L), ATmega16, ATmega161(L), ATmega162, ATmega163(L), ATmega169, ATmega32, ATmega323(L)

Table 2: Processors for Salvo Libraries – ImageCraft's ICCAVR C Compiler

Note The target code for an unlisted processor can generally be deduced by whether the processor is a member of the standard AVR or MegaAVR family.

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 ImageCraft's ICCAVR 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 ICCAVR debugger. To use these libraries, simply select one that includes the debugging information (e.g. `libsliccavrit.a`) instead of one without (e.g. `libsliccavr-t.a`) in your ICCAVR 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 ImageCraft's ICCAVR 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 3.

compiled limits	
max. number of tasks	3
max. number of events	5
max. number of event flags	1
max. number of message queues	1
target-specific settings	
delay sizes	8 bits
watchdog timer	cleared in `OSSched()`.
system tick counter	available, 32 bits

Table 3: Build Settings and Overrides for Salvo Libraries for ImageCraft's ICCAVR 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

There are a total of 34 Salvo libraries for ImageCraft's ICCAVR C compiler. Each Salvo for Atmel AVR and MegaAVR distribution contains the Salvo libraries of the lesser distributions beneath it.

Target-Specific Salvo Source Files

One of two different source files, `porticcavr.s` or `porticcatmega.s`, is required for Salvo Pro source code builds. Use the one appropriate for your target as per the target code nomenclature shown in Table 2.

Note `porticcavr.s` will work on every AVR target that has 8KB or less of program memory. `porticcatmega.s` is required for MegaAVR targets can address more than 8KB of program memory.

salvocfg.h Examples

Below are examples of `salvocfg.h` project configuration files for different Salvo for Atmel AVR and MegaAVR distributions targeting the AT90S8515.

Note When overriding the default number of tasks, events, etc. in a Salvo library build, `OSTASKS` and `OSEVENTS` (respectively) *must also be defined* in the project's `salvocfg.h`. If left undefined, the default values (see Table 3) will be used.

Salvo Lite Library Build

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

Listing 1: Example salvocfg.h for Library Build Using libsficcavr-a.a

Salvo tiny Library Build

```
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE          OSL
#define OSLIBRARY_CONFIG        OSY
```

Listing 2: Example salvocfg.h for Library Build Using libsliccavr-y.a

Salvo SE Library Build

```
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE          OSL
#define OSLIBRARY_CONFIG        OSS
```

Listing 3: Example salvocfg.h for Library Build Using libsliccavr-s.a

Salvo LE & Pro Library Build

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

Listing 4: Example salvocfg.h for Library Build Using libsliccavr-a.a or libsliccavria.a

Salvo Pro Source-Code Build

```
#define OSENABLE_IDLING_HOOK    TRUE
#define OSENABLE_SEMAPHORES     TRUE
#define OSEVENTS                1
#define OSTASKS                 3
```

Listing 5: Example salvocfg.h for Source-Code Build

Performance

Memory Usage

tutorial memory usage[4]	total ROM[5]	total RAM[6]
`tu1lite`	239 / 232	22
`tu2lite`	325 / 317	22
`tu3lite`	358 / 349	24
`tu4lite`	695 / 654	33
`tu5lite`	1058 / 961	45
`tu6lite`	1144 / 1038[7]	47[8]
`tu6pro`[9]	1025 / 947[10]	43[11]

Table 4: ROM and RAM requirements for Salvo Applications built with ImageCraft's ICCAVR C Compiler

Special Considerations

Stack Issues

ImageCraft's ICCAVR C compiler uses two separate stacks – one for return addresses (the hardware stack, which uses `SP`) and one for parameter passing and local storage (the software stack, which uses `Y`).

Compared to a non-Salvo, non-multitasking application with similar call trees, the corresponding Salvo application will require an additional 4 bytes (i.e. two return addresses) in the hardware stack.[12]

The size of the hardware stack can be set in the ICCAVR IDE via Project → Options → Target → Advanced → Return Stack Size or on the `iccavr` linker command line, e.g.:

```
iccavr … -dhwstk_size:20 …
```

Applications using nested interrupts, floating points or `longs` will require a hardware stack larger than the default size – see ICCAVR Help for more information.

External SRAM

Salvo's global objects[13] can be placed in internal or external RAM. In ImageCraft's ICCAVR IDE, the placement of objects (e.g. variables) in the `data` *program area* is controlled via Project →

Options → Target → Device Configuration (Internal SRAM), etc. On the `iccavr` linker command line, placement of these objects is specified via –`bdata:start.end`, e.g.:

```
iccavr … -bdata:0x260.0xffff …
```

specifies that the `data` program area start at `0x260` (the end of internal SRAM) and extend to `0xFFFF` (the 64K boundary).

Salvo's global objects can be placed – as a group – anywhere in RAM (internal or external) by specifying the start and end addresses of the `salvoram` program area. This applies to source-code and library builds. For example, to place all of Salvo's global objects in a 256-byte block of external RAM just beneath the 32K boundary, use

```
iccavr … -bsalvoram:0x7F00.0x7FFF …
```

when linking your application.[14]

Note If you do not use the `-b` linker command-line argument, the `salvoram` program area will be located immediately after the `bss` program area in the `data` program area. Therefore it is only required if you wish to locate Salvo's global objects separately from your program's variables, etc. You can override the order of the program areas by using the `.area` assembler directive.

Data Segments

The `RAMPD` register is normally used to access the entire data space on processors with more than 64K bytes data space. There are no provisions for accessing Salvo's global objects outside of the current data segment of 64K bytes.

Code Compressor

Salvo is compatible with ImageCraft's ICCAVR Code Compressor[15] in both library- and source-code builds.

Indirect Function Calls

In order for Code Compressor to work properly, all indirectly called functions must be called through `xicall`. The context-switching method employed by Salvo with ImageCraft's ICCAVR C compiler uses `xicall` for all of its indirect function calls.

Registers R20..R23

ICCAVR can be instructed to not use registers `R20..R23`. In practice, this has little effect on the Salvo code – it may result in a small speedup and smaller ROM size.

The Salvo libraries are built *without* using `R20..R23` so that control of these registers is left to the programmer.

Salvo Pro users can control the use of these registers in a source-code build.

Library Locations

ImageCraft's ICCAVR C compiler expects libraries to be in `\icc\lib`. Therefore the Salvo installer places its libraries for ICCAVR in both `\salvo\lib\iccavr` and `\icc\lib`.

---

1 tinyAVR devices are not supported because of their lack of RAM.

2 This is done automatically through the `__IMAGECRAFT__` and `_AVR` symbols defined by the compiler.

3 Since saving and restoring of the `I` bit is intimately associated with the compiler's `#pragma monitor`, alternate methods are generally not recommended.

4 Salvo v3.2.0 with ICCAVR v6.28.

5 In words. Second number reflects ROM size with Code Compressor enabled. Includes interrupt vectors and `func_lit` table for functions called indirectly via `xicall`. `R20..R23` are not used.

6 In bytes. Does not include RAM reserved for the return address (hardware) stack or the parameter passing and local storage (software) stack.

7 Includes 2 bytes from the `idata` section.

8 Includes 2 bytes from the `data` section.

9 Salvo Pro build differs slightly from Salvo Lite build due to configuration – see tutorial's `salvocfg.h`.

10 Includes 2 bytes from the `idata` section.

11 Includes 2 bytes from the `data` section.

12 Salvo Pro application can reduce this by 2 bytes (one return address) by inlining `OSSched()`.

13 E.g. task control blocks, queue pointers, counters, etc.

14 Failure to allocate enough RAM for the `salvoram` program area will result in an `area 'salvoram' not large enough` linker error.

15 Code Compressor is included in ICCAVR Professional.