PUMPKIN™
REAL-TIME SOFTWARE

RM-IARAVR
Справочное руководство

750 Naples Street • San Francisco, CA 94112 • (415) 584-6360 • http://www.pumpkininc.com
*• **Перевод:** Андрей Шлеенков • http://andromega.narod.ru • mailto:andromega@narod.ru •*

Справочное руководство Salvo для компилятора IAR AVR C

Введение

Данное руководство предназначено для пользователей Salvo, использующих микроконтроллеры AVR® и MegaAVR™[1] компании Atmel (http://www.atmel.com/) с компилятором IAR AVR компании IAR (http://www.iar.com/).

Связанные документы

При создании приложений Salvo с компилятором IAR AVR, вместе с данным руководством должны использоваться следующие документы Salvo:

Руководство пользователя Salvo (Salvo User Manual)
Приложение AN-29 (Application Note AN-29)

Примеры проектов

Примеры проектов Salvo для использования с компилятором IAR AVR могут быть найдены в следующих директориях каждого дистрибутива Salvo для Atmel AVR и MegaAVR:

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

Свойства

Таблица 1 иллюстрирует основные особенности реализации Salvo для компилятора IAR AVR.

основное	
доступные дистрибутивы	Salvo Lite, LE & Pro for Atmel AVR and MegaAVR
поддерживаемые устройства	семейства AVR и MegaAVR
заголовочные файлы	`portiaravr.h`
другие специфические для процессора файлы	`portiaravr.c`
имена поддиректорий проекта	SYSAC
salvocfg.h	
специфические для процессора заголовочные файлы нужны?	да
автоопределение компилятора?	да[2]
библиотеки	
поддиректории `\salvo\lib`	`iaravr-v2` (для компиляторов v2.x) `iaravr-v3` (для компиляторов v3.x)
переключение контекста	
метод	на основе функций `OSDispatch()` & `OSCtxSw()`
`_OSLabel()` требуется?	нет
объем автоматических переменных и параметров функций в задачах	общий объем не должен превышать 254 8-битовых байт
память и регистры	
поддержка внутренней и внешней памяти RAM?	да / да (см. раздел о внешней памяти)
использование регистров	соответствует интерфейсу с языком ассемблера
прерывания	
управляются через	`I` бит
статус прерывания сохраняется в критических секциях?	да
используемый метод	ключевое слово `__monitor`
степень вложенности	не ограничена
альтернативные методы возможны?	не рекомендованы
отладка	
отладка в исходных кодах с библиотеками Salvo Pro?	да
компилятор	
поддержка упакованных битовых полей?	нет
`printf()` / `%p` поддерживается?	да / да
`va_arg()` поддерживается?	да

Таблица 1: Особенности реализации Salvo для компилятора IAR AVR

Библиотеки

Номенклатура

Имена библиотек Salvo для компилятора IAR AVR следуют соглашениям, показанным на примере имени одной из библиотек на Рисунке 1.

Пример имени библиотеки: **`sfiaravr1s-a.r90`**

символы	значение	возможные варианты
`s`	библиотека Salvo	
`f`	тип	**`f`**: freeware **`l`**: standard
`iaravr`	Компилятор Си IAR AVR	
`1`	обобщенный контроллер (см. таблицу)	**`0:`** `-v0` **`1:`** `-v1` **`2:`** `-v2` **`3:`** `-v3` **`4:`** `-v4` **`5:`** `-v5` **`6:`** `-v6`
`s`	модель памяти	**`l:`** `large` **`s:`** `small` **`t:`** `tiny`
`-`	опция	**`-`**: нет опций **`i`**: с отладочной информацией
`a`	конфигурация	**`a`**: многозадачность с задержками и событиями **`d`**: многозадачность с задержками **`e`**: многозадачность с событиями **`m`**: многозадачность только **`t`**: многозадачность с задержками, событиями, с таймаутом

Рисунок 1: Номенклатура библиотек Salvo для компилятора Си IAR AVR

Тип

Дистрибутив Salvo Lite содержит *свободные* (*freeware*) библиотеки. Все остальные дистрибутивы Salvo содержат *стандартные* (*standard*) библиотеки. Дополнительную информацию о типах библиотек см. в главе *Библиотеки* (*Libraries*) документа *Руководство пользователя Salvo* (*Salvo User Manual*).

Целевой процессор

Каждая библиотека создана с обобщенными опциями и предназначена для одного или нескольких процессоров на основании опции компилятора v предназначенной для выбранного процессора. Таблица 2 перечисляет корректные библиотеки для каждого поддерживаемого процессора AVR или MegaAVR.

код процессора	процессор(ы)
0	AT90S2313, AT90S2323, AT90S2333, AT90S2343, AT90S4433, ATtiny13, ATtiny22, ATtiny26, ATtiny2313, обобщенный –v0
1	AT90S4414, AT90S4434, AT90S8515, AT90S8534, AT90S8535, ATmega8, ATmega48, ATmega8515, ATmega8535, обобщенный –v1
2	обобщенный –v2
3	ATmega16, ATmega32, ATmega103, ATmega128, ATmega161, ATmega162, ATmega163, ATmega168, ATmega169, ATmega323, обобщенный –v3
4	обобщенный –v4
5	обобщенный –v5
6	обобщенный –v6

Таблица 2: Процессоры для библиотек Salvo для компилятора IAR AVR

Замечание: Таблица 2 не является исчерпывающей – если вы не уверены, какой целевой код использовать с библиотекой Salvo, обратитесь к разделу *Варианты процессора* в *Справочном руководстве* компилятора Си IAR AVR.

Дополнительно, вы можете рассмотреть опции командной строки, которая передается компилятору при использовании Embedded Workbench для определения того, какой вариант процессора ассоциируется с вашим целевым AVR.

При построении вашего приложения с библиотекой Salvo, вы можете или определить целевой процессор (например `--cpu=8515`) или обобщенный процессор (например `-v1`) для компилятора Си AVR IAR.[3] В любом случае, обобщенная опция, связанная с вашим целевым процессором должна соответствовать целевому коду библиотеки Salvo.

Модель памяти

Компилятор Си IAR AVR поддерживает модели памяти `tiny`, `small` и `large`. При компоновке с библиотеками, модель памяти, применяемая ко всем исходным файлам, должна соответствовать той, которая использовалась в библиотеке – несоответствие вызовет ошибку компоновки с соответствующим сообщением. При построении с исходными кодами, одна и та же модель памяти должна быть применена ко всем исходным файлам.

Замечание: В отличие от опции конфигурации библиотеки, определенной в файле `salvocfg.h` для компоновки с библиотекой, ничто не определено для выбранной модели памяти. Поэтому нужно уделить особое внимание параметрам настройки модели памяти, используемым при построении приложения. Модель памяти обычно определяется на основании всего проекта в Embedded Workbench IDE.

Опция

Пользователи Salvo Pro могут выбирать между двумя наборами библиотек – стандартные библиотеки, и стандартные библиотеки, включающие отладочную информацию исходного уровня.[4] Последняя создана компилятором IAR AVR с опцией командной строки `--debug`. Это добавляет в библиотеки отладочную информацию исходного уровня, делая их идеальными для отладки в исходном уровне и пошаговой отладки в отладчике C-SPY. Чтобы использовать эти библиотеки, просто выберите ту, которая включает отладочную информацию (например, `sliaravrltia.a90`) вместо не содержащей ее (например, `sliaravrlt-a.a90`) в вашем проекте Embedded Workbench.

Конфигурация

Для различных дистрибутивов Salvo обеспечиваются различные конфигурации библиотек, позволяющие пользователю минимизировать код ядра Salvo. См. главу *Библиотеки Руководства пользователя Salvo* для получения дополнительной информации о конфигурациях библиотеки.

Установки компиляции

Библиотеки Salvo для компилятора IAR AVR построены, используя параметры настройки по умолчанию, описанные в главе *Библиотеки Руководства пользователя Salvo*. Специфические для целевого процессора параметры и их замены перечислены в Таблице 3.

ограничения компиляции	
макс. число задач	3
макс. число событий	5
макс. число флагов событий	1
макс. число очередей сообщений	1
специфические для процессора установки	
размер задержки	8 бит
сторожевой таймер	очищается в OSSched( )
счетчик системного времени	доступен, 32 бита

Таблица 3: Установки и замены для библиотек Salvo для компилятора IAR AVR

Замечание: Ограничения компиляции библиотек Salvo для задач, событий и т.п. могут быть изменены в меньшую сторону (все дистрибутивы Salvo) или в большую сторону (все дистрибутивы Salvo кроме Salvo Lite) относительно значений по умолчанию. См. главу *Библиотеки* документа *Руководство пользователя Salvo*.

Доступные библиотеки

Всего доступно 330 библиотек Salvo для компилятора IAR AVR – 165 для AVR C v2.x и 165 для AVR C v3.x. Каждый тип дистрибутива Salvo для Atmel AVR и MegaAVR включает также библиотеки Salvo из младших версий дистрибутивов.

Специфичные для процессора исходные файлы Salvo

Для построения с исходными кодами Salvo Pro необходим исходный файл `portiaravr.c`.

Примеры salvocfg.h

Ниже приводятся примеры `salvocfg.h` файлов конфигурации проекта для различных дистрибутивов Salvo для Atmel AVR и MegaAVR.

Замечание: При изменении заданных по умолчанию числа задач, событий и т.п. при построении с библиотеками Salvo, в файле проекта `salvocfg.h` *должны быть определены* `OSTASKS` и `OSEVENTS` соответственно. При отсутствии определений, будут использованы значения по умолчанию (см. Таблицу 3).

Компиляция с библиотеками Salvo Lite

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

Листинг 1: Пример salvocfg.h для компиляции с библиотеками, используя sfiaravr1s-a.r90

Компиляция с библиотеками Salvo LE & Pro

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

Листинг 2: Пример salvocfg.h для компиляции с библиотеками, используя sliaravr3t-e.r90

Компиляция с библиотеками Salvo Pro и отладка с исходным кодом

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

Листинг 3: Пример salvocfg.h для компиляции с библиотеками используя sliaravr1sit.r90

Компиляция с исходным кодом Salvo Pro

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

Листинг 4: Пример salvocfg.h для компиляции с исходным кодом

Эффективность

Использование памяти

учебные примеры[5]	всего ROM[6]	всего RAM[7]
tu1lite	316	7
tu2lite	448	17
tu3lite	498	19
tu4lite	968	24
tu5lite	1442	34
tu6lite	1546	36
tu6pro[8]	1416	32

Таблица 4: Требования памяти ROM и RAM для приложения Salvo, построенного компилятором IAR AVR

Специальные вопросы

Размер стека

Компилятор IAR AVR использует два раздельных стека – один для адресов возврата (аппаратный стек RSTACK, управляемый SP) и один для передачи параметров и хранения локальных данных (программный стек CSTACK, управляемый Y).

По сравнению с не Salvo не многозадачным приложением со схожей структурой вызовов, соответствующее приложение Salvo потребует дополнительно 6-и байтов (т.е. двух адресов возврата и двух регистров) в аппаратном стеке.[9]

Аппаратный и программный стеки располагаются в одном месте. Однако аппаратный стек растет вниз, а программный стек растет вверх.

Внешняя память RAM

Глобальные объекты Salvo[10] могут быть размещены как во внутренней, так и во внешней памяти данных RAM. Размещение объектов данных контролируется опциями компоновщика – дополнительную информацию см. в документации IAR.

Модели памяти и глобальные объекты Salvo

Атрибуты памяти компилятора *по умолчанию* и *типы указателей по умолчанию* автоматически применяются к глобальным объектам Salvo, основанным на выбранной модели памяти. Поэтому опции конфигурации Salvo OSLOC_XYZ не должны использоваться в Salvo Pro при построении с исходным кодом.

Хранение данных

Таблица 5 иллюстрирует эффект воздействия выбранной модели памяти на глобальные объекты Salvo.

принятая модель памяти	максимальный размер объекта	размер указателя	диапазон адресов
tiny	127 байт	1 байт	0x0-0xFF
small	32 Кбайт	2 байта	0x0-0xFFFF
large	32 Кбайт	3 байта	0x0-0xFFFFFF

Таблица 5: Эффект воздействия модели памяти на глобальные объекты Salvo

Модель памяти `tiny` полезна при желании минимизировать объем RAM, используемой Salvo, и требования к коду ROM Salvo. С моделью памяти `tiny` максимальное число задач и событий строго ограничено.[11]

Модель памяти `small` требуется, когда размер глобальных объектов Salvo превышает 127 байт.

Модель памяти `large`, вероятно, не понадобится для большинства приложений, но включена для полноты. При использовании, объекты Salvo не могут пересекать границу 64 КБ.

Замечание: Приложение пользователя должно быть построено с той же самой моделью памяти, что и используемая библиотекой Salvo. Однако атрибуты памяти данных пользователя *не обязаны быть теми же самыми* что и у глобальных объектов Salvo.

Например, можно было бы построить приложение с моделью памяти `small` и с атрибутом `__tiny`, примененным к некоторым данным пользователя, чтобы максимизировать скорость и минимизировать ROM, ассоциированную с этими пользовательскими данными.

Указатели функций

Переключатель контекста Salvo использует инструкцию AVR `IJMP` для осуществления косвенных вызовов функций. Так как `IJMP` поддерживает размер адреса только `PC(15..0)`, все задачи Salvo должны быть с атрибутом `__nearfunc` (по умолчанию в компиляторе)[12], и поэтому должны быть расположены в пределах первых 128 КБ программы. Атрибут памяти функции `__farfunc` не поддерживается для использования с задачами Salvo.

Оптимизация

Salvo совместим с оптимизацией кода IAR AVR на всех уровнях, при компиляции и с исходными кодами, и с библиотеками.

Глобальные регистровые переменные

Регистры `R4-R15` доступны для пользователя как *глобальные регистровые переменные* с опцией компилятора `--lock_regs`.

[1] Микроконтроллеры tinyAVR не поддерживаются из-за отсутствия памяти RAM.
[2] Выполняется автоматически при помощи символов __GNUC__ и __AVR__, определяемых компилятором.
[3] Как в Embedded Workbench, так и в командной строке.
[4] Библиотеки Salvo Lite и LE не содержат совместимой с C-SPY отладочной информации, потому что это требует включения файлов исходных кодов.
[5] Salvo v3.2.4-dev2 с IAR AVR C v3.10A. Все учебные проекты построены, используя `tiny` модель.
[6] В байтах памяти `CODE` с максимальной оптимизацией по размеру (`-z9`), и с запрещенной оптимизацией перекрестных вызовов. Не включает диапазон заполнения. Другие компиляторы могут сообщать об использовании памяти в словах (1 слово = 2 байта).
[7] В байтах. Не включает ни 32 байта для `CSTACK`, ни 32 байта (16 уровней) для `RSTACK`. Кроме того, не включает `absolutes`.
[8] Salvo Pro проект немного отличается от Salvo Lite проекта из-за конфигурации – см. учебный `salvocfg.h`.
[9] Приложение Salvo Pro может уменьшить это на 2 байта (адрес возврата) встраиванием кода `OSSched()` в исходный код способом inline.
[10] Например, блоки управления задачами, указатели очередей, счетчики и т.д.
[11] Тем более, что стек также расположен в нижней RAM.
[12] То есть два байта, используемые для указателей функции.