PUMPKIN™
REAL-TIME SOFTWARE

RM-GCCAVR
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 GNU avr-gcc

Введение

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

Связанные документы

При создании приложений Salvo с компилятором GNU avr-gcc, вместе с данным руководством должны использоваться следующие документы Salvo:

Руководство пользователя Salvo (Salvo User Manual)
Приложение AN-28 (Application Note AN-28)

Application Note AN-28 объясняет, как использовать make-файлы для успешного создания приложение Salvo, используя GNU avr-gcc компилятор и связанные инструменты.

Примеры проектов

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

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

Свойства

Таблица 1 иллюстрирует основные особенности реализации Salvo для компилятора GNU avr-gcc.

основное	
доступные дистрибутивы	Salvo Lite, LE & Pro for Atmel AVR and MegaAVR
поддерживаемые устройства	семейства AVR и MegaAVR
заголовочные файлы	`portgccavr.h`
другие специфические для процессора файлы	`portgccavr.s`
имена поддиректорий проекта	SYSY
salvocfg.h	
специфический для процессора заголовочный файл требуется?	да
автоопределение компилятора?	да[2]
переключение контекста	
метод	на основе функций `OSDispatch()` & `OSCtxSw(label)`
`_OSLabel()` требуется?	да
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	`R0`, `R16`, `R17` используются для временного хранения в процедуре `OSCtxSw()`. `R0`, `R18` используются для временного хранения в процедуре `OSDispatch()`.
прерывания	
управляются через	`I` бит
статус прерывания сохраняется в критических секциях?	да
используемый метод	сохранение в локальных переменных в начале и восстановление при окончании процедуры
степень вложенности	не ограничена
альтернативные методы возможны?	не рекомендуются
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 Си GNU avr-gcc

Библиотеки

Номенклатура

Имена библиотек Salvo для компилятора GNU avr-gcc следуют соглашениям, показанным на примере имени одной из библиотек на Рисунке 1.

Пример имени библиотеки: **`libsfgccavr-a.a`**

символы	значение	возможные варианты
`lib`	библиотека	
`s`	Salvo	
`f`	тип	**`f`**: freeware **`l`**: standard
`gcc`	GCC-AVR	
`avr`	AVR и MegaAVR	
`-`	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 и ожиданиями

Рисунок 1: Номенклатура библиотек Salvo для компилятора Си GNU avr-gcc

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

Один набор библиотек будет работать со всеми микроконтроллерами AVR, имеющими следующие средства:

- Внутренняя SRAM
- Инструкция `ICALL`

Опция

Пользователи Salvo Pro могут выбирать между двумя наборами библиотек – стандартные библиотеки, и стандартные библиотеки, включающие отладочную информацию уровня исходного кода.[3] Последние были построены с опцией командной строки `-g` компилятора GNU avr-gcc. Это добавляет отладочную информацию исходного уровня к библиотекам, делая их идеальными для отладки на исходном уровне и пошаговой отладки в IDE AVRStudio. Чтобы использовать эти библиотеки, просто выберите в вашем проекте ту, которая включает отладочную информацию (например, `libslgccavrit.a` вместо `libslgccavr-t.a`).

Конфигурация

Для различных дистрибутивов Salvo предусмотрены различные конфигурации библиотек, позволяющие пользователю минимизировать код ядра Salvo. Дополнительную информацию о конфигурации библиотек см. в главе *Библиотеки* документа *Руководство пользователя Salvo*.

Установки компиляции

Библиотеки Salvo для компилятора Си GNU avr-gcc построены, используя установки по умолчанию, описанные в главе *Библиотеки* документа *Руководство пользователя Salvo*. Специфические для процессоров установки и их замены перечислены в Таблице 2.

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 2: Установки и замены для библиотек Salvo для компилятора Си GNU avr-gcc

Замечание: Ограничения компиляции библиотек Salvo 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 15 библиотек Salvo для компилятора Си GNU avr-gcc. Каждый тип дистрибутива Salvo для Atmel AVR и MegaAVR включает также библиотеки Salvo из младших версий дистрибутивов.

Специфичные для процессора исходные файлы Salvo

Для компиляции с исходным кодом Salvo Pro требуется исходный файл `portgccavr.S`.

Замечание: Никогда не переименовывайте `portgccavr.S` в `portgccavr.s`, поскольку make-файл рассматривает их как два различных файла. `.S` файл – пользовательский ассемблерный код, тогда как `.s` файл – промежуточный файл, произведенный `avr-gcc`, который может быть удален.

Примеры salvocfg.h

Ниже приводятся примеры файлов конфигурации проекта `salvocfg.h` 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 2).

Компиляция с библиотеками Salvo Lite

```
#include <avr/io.h>
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE          OSF
#define OSLIBRARY_CONFIG        OSA
```

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

Компиляция с библиотеками Salvo LE & Pro

```
#include <avr/io.h>
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE          OSL
#define OSLIBRARY_CONFIG        OSA
```

Листинг 2: Пример salvocfg.h для компиляции с библиотеками, используя libslgccavr-a.a

Компиляция с библиотеками Pro и отладкой в исходном коде

```
#include <avr/io.h>
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE          OSL
#define OSLIBRARY_OPTION        OSI
#define OSLIBRARY_CONFIG        OSA
```

Листинг 3: Пример salvocfg.h для компиляции с библиотеками используя libslgccavria.a

Компиляция с исходным кодом Salvo Pro

```
#include <avr/io.h>
#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[4]	всего ROM[5]	всего RAM[6]
tu1lite	197	24
tu2lite	253	24
tu3lite	280	26
tu4lite	562	35
tu5lite	770	47
tu6lite	848	48
tu6pro[7]	760	44

Таблица 4: Требования памяти ROM и RAM для приложения Salvo, генерируемого компилятором Си GNU avr-gcc

Специальные вопросы

Размер стека

Компилятор Си GNU avr-gcc использует два раздельных стека – один для адресов возврата (аппаратный стек, управляемый SP) и один для передачи параметров и хранения локальных данных (программный стек, управляемый Y).

По сравнению с не Salvo не многозадачным приложением со схожей структурой вызовов, соответствующее приложение Salvo потребует дополнительно 4-х байт (т.е. двух адресов возврата) в аппаратном стеке.[8]

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

Внешняя память SRAM

Глобальные объекты Salvo[9] могут быть помещены во внутреннюю или внешнюю память RAM. Размещение объектов данных контролируется опциями компоновщика, в формате `-Wl, -Tdata=0x800000+start` для размещения секции программы `data` (т.е. для хранения переменных), где 0x800000+start означает hex-адрес со смещением 0x800000. Нет никаких ограничений на предел адреса, вы отвечаете как программист, чтобы не использовать больше памяти RAM, чем вы имеете. Например:

```
avr-gcc ... –Wl, -Tdata=0x800260 ...
```

определяет, что область программы `data` начинается с `0x260` (конец внутренней SRAM). Это может быть добавлено в ваш make-файл с остальными флагами компоновщика (обычно называемыми `LFLAGS`), и фактически ваш make-файл может быть уже настроен для работы с внешней SRAM. Если вы используете это, вы также должны убедиться, что аппаратные средства процессора настроены для использования внешней SRAM. Лучше сделать это как можно раньше, фактически самый лучший путь состоит в том, чтобы сделать файл с именем `xram.S`, и заполнить его чем-либо подобным:

```
;; begin xram.S

#include <avr/io.h>

    .section .initl,"ax",@progbits

    ldi  R16,(1 << SRE) | (1 << SRW)
    out  _SFR_IO_ADDR(MCUCR),R16

;; end xram.S
```

Это работало бы для AT90S8515, разрешая внешнюю XRAM с одним тактом ожидания. Файл должен быть добавлен в ваш проект. Отметьте, что код, выполняемый в этом файле, располагается в самом начале, прежде чем что-нибудь типа стека будет настроено.

Сегменты данных

Регистр RAMPD обычно используется для доступа ко всему пространству данных в процессорах с более чем 64 КБ пространством памяти данных. Не имеется никаких средств для доступа к глобальным объектам Salvo вне текущего сегмента данных размером в 64 КБ.

Оптимизатор

Salvo совместим с оптимизатором кода GNU avr-gcc на всех уровнях при компиляции и с исходным кодом и с библиотеками.

Регистр R1

Библиотека `avr-libc`, которая является неотъемлемой частью avr-gcc, предполагает, что регистр `R1` будет всегда иметь значение `0`. Salvo не изменяет этот регистр.

Расположение библиотек

Инсталлятор Salvo размещает библиотеки для avr-gcc в папке `/salvo/lib/gccavr`. При компоновке с библиотеками Salvo, должен быть определен дополнительный путь к библиотекам в фазе компоновки, например через:

```
avr-gcc ... -L c:/salvo/lib/gccavr ...
```

Это обычно делается автоматически как часть системы make-файла Salvo для использования с avr-gcc и WinAVR.

Благодарности

Colin O'Flynn написал переключатель контекста Salvo в `portgccavr.S`, создал систему make-файла проекта Salvo, и написал большую часть документации, окружающей версию Salvo для компилятора GNU avr-gcc. Colin активен в сообществе AVR и является автором различного материала об AVR, который может быть найден на популярном веб-сайте AVR Freaks (http://www.avrfreaks.net/).

[1] Микроконтроллеры tinyAVR не поддерживаются из-за отсутствия памяти RAM.
[2] Выполняется автоматически при помощи символов `__GNUC__` и `__AVR__`, определяемых компилятором.
[3] Библиотеки Salvo, предоставляемые Salvo Lite и LE не содержат avr-gcc-debugger совместимую отладочную информацию, потому что это требует включения файлов исходного кода.
[4] Salvo v.3.2.0 с WINAVR редакции 20030424.
[5] В словах с опцией оптимизации `-Os`. Заметьте, что avr-size возвращает размер в байтах, а не в словах.
[6] В байтах, это секция `.bss` файла `.elf`.
[7] Компиляция с Salvo Pro несколько отличается от компиляции с Salvo Lite при конфигурировании – см. учебный пример `salvocfg.h`.
[8] Приложение Salvo Pro может уменьшить это на 2 байта (адрес возврата) встраиванием кода `OSSched()` в исходный код приложения способом inline.
[9] Например, блоки управления задачами, указатели очереди, счетчики, и т.д.