PUMPKIN™
REAL-TIME SOFTWARE

RM-KCARM
Справочное Руководство

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

Справочное Руководство Salvo для Компилятора Keil CARM

Введение

Данное руководство предназначено для пользователей Salvo, использующих микроконтроллеры ARM7TDMI MCU с компилятором Си Keil's CARM (http://www.keil.com/).

Связанные Документы

При создании приложений Salvo при помощи компилятора Си Keil's CARM совместно с данным руководством должны быть использованы следующие документы Salvo:

Руководство Пользователя Salvo (Salvo User Manual)
Примечание AN-31 (Application Note AN-31)

Примеры Проектов

Примеры проектов Salvo для компилятора Си Keil's CARM и Keil's µVision3 IDE могут быть найдены в следующих директориях каждого дистрибутива Salvo для семейства ARM:

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

Свойства

Таблица 1 иллюстрирует основные особенности реализации Salvo для компилятора Си Keil's CARM.

основное	
доступные дистрибутивы	Salvo Lite, LE & Pro для семейства ARM7TDMI
поддерживаемые устройства	все ARM7TDMI производные
заголовочные файлы	`portkcarm.h`
другие специфические для процессора файлы	`portkcarm.s`
имена поддиректорий проекта	SYSAG
salvocfg.h	
автоопределение компилятора?	да[1]
библиотеки	
поддиректория `\salvo\lib`	`kcarm`
поддержка смешанного режима работы CPU	да
переключение контекста	
метод	на основе функций `OSDispatch()` & `OSCtxSw()`
`_OSLabel()` требуется?	нет
объем автоматических переменных и параметров функций в задачах	общий объем не должен превышать 65,532 8-битовых байт
прерывания	
управляются через	`OSDisableInts()`, `OSEnableInts()`, `OSRestoreInts()`, `OSSaveInts()`
статус прерывания сохраняется в критических секциях?	да, с помощью соответствующих функций пользователя
используемый метод	см. пользовательские функции
отладка	
отладка в исходных кодах?	только при создании с исходными кодами
Компилятор	
поддержка упакованных битовых полей?	нет
`printf()` / `%p` поддерживается?	да / да
`va_arg()` поддерживается?	да

Таблица 1: Особенности Реализации Salvo для компилятора Keil's CARM

Библиотеки

Номенклатура

Имена библиотек Salvo для компилятора Си Keil's CARM следуют соглашениям, показанным на примере имени одной из библиотек на Рисунке 1.

Пример имени библиотеки: **`sfkcarm4lt-a.lib`**

символы	значение	возможные варианты
`s`	Salvo	
`f`	Тип	**`f`**: freeware **`l`**: standard
`kcarm`	Keil CARM Си компилятор	
`4`	ARM архитектура	пример: 4: v4T (ARM7TDMI-S)
`l`	порядок битов	**`b`**: big endian **`l`**: little endian
`t`	набор инструкций	**`a`**: ARM режим **`t`**: Thumb режим
`-`	смешанный режим CPU	**`-`**: не разрешен **`i`**: библиотека содержит средства для смешанного режима CPU
`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 с таймаутом

Рисунок 1: Номенклатура Библиотек Salvo для Компилятора Keil's CARM

Тип

Дистрибутив Salvo Lite содержит *freeware (свободные)* библиотеки. Все остальные дистрибутивы Salvo содержат *standard (стандартные)* библиотеки. См. главу *Библиотеки* в документе *Руководство Пользователя Salvo* для получения дополнительной информации о типах библиотек.

Архитектура ARM

Библиотеки могут быть использованы со всеми микроконтроллерами, основанными на ядрах ARM7, ARM7TDMI(-S), ARM720 и т.п.

Замечание: Библиотеки Salvo могут также быть совместимы с новыми архитектурами ARM, которые обеспечат совместимость по двоичному коду с архитектурой ARM v4T.

Порядок Байтов - Endianness

Показывает порядок байтов (endianness) в библиотеках (big-endian или little-endian),что означает следование старшим или младшим байтом вперед в многобайтных объектах.

Набор Инструкций

Поддерживается два типа библиотек – один тип для ARM инструкций и один тип для набора Thumb инструкций.[2]

Замечание: В отличие от опций конфигурирования определеннных в файле `salvocfg.h` для построения с библиотеками, ни один из набора команд не определен. Поэтому особое внимание нужно уделить инструкциям настройки при редактировании связей с библиотеками Salvo, которые не включают смешанный режим CPU (см. ниже). Набор команд обычно определяется на основании проекта µVision3 IDE.

Смешанный Режим CPU

Компилятор Keil's CARM поддерживает смешанный режим CPU при помощи директивы `INTERWORK`. Библиотеки со смешанным режимом CPU могут быть скомпонованы с приложениями, скомпилированными в ARM или Thumb режиме.

Замечание: Приложение, созданное в смешанном режиме библиотек CPU, будет занимать больше кода, чем сформированное в чистом ARM или Thumb режиме.

Конфигурация

Различные конфигурации библиотек дистрибутивов Salvo позволяют минимизировать объем, занимаемый ядром Salvo. См. главу *Библиотеки* документа *Руководство Пользователя Salvo* для получения более подробной информации.

Установки Построения

Библиотеки Salvo для компилятора Си Keil's CARM созданы при помощи установок по умолчанию, описанных в главе *Библиотеки* документа *Руководство Пользователя Salvo*. Специфические для процессоров установки и ограничения перечислены в Таблице 2.

ограничения библиотек Salvo Lite	
макс. число задач	3
макс. число событий	5
макс. число флагов событий[3]	1
макс. число очередей сообщений[4]	1
аппаратно зависимые установки для всех библиотек Salvo	
размер задержки	8 бит
холостой ход	разрешен
биты разрешения прерываний в критических секциях	управляются пользовательскими функциями
счетчик системного времени	доступен, 32 бита
приоритеты задач	разрешены
сторожевой таймер	управляется пользовательскими функциями

Таблица 2: Ограничения, Установки и Пределы для Библиотек Salvo для Компилятора Си Keil's CARM

Замечание: При создании приложений ARM7TDMI с библиотеками Salvo, количество задач, событий, флагов событий и очередей сообщений ограничено только библиотеками Salvo Lite. В других библиотеках Salvo подобные ограничения отсутствуют. См. главу *Библиотеки* в документе *Руководство Пользователя Salvo* для получения более подробной информации.

Доступные Библиотеки

Существует 40 библиотек Salvo для компилятора Keil's CARM. 20 из них для набора инструкций ARM, и 20 для набора инструкций Thumb. Каждый дистрибутив Salvo для ARM содержит библиотеки Salvo младших версий дистрибутивов.

Примеры salvocfg.h

Ниже приводятся примеры файлов конфигурации проектов `salvocfg.h` для различных дистрибутивов ARM для однокристальных микроконтроллеров Philips LPC2129.

Построение с Библиотеками Salvo Lite

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

Листинг 1: Пример salvocfg.h для Построения с Библиотеками sfkcarm4lt-a.lib

Построение с Библиотеками Salvo LE & Pro

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

Листинг 2: Пример salvocfg.h для Построения с Библиотеками slkcarm4la-a.lib

Построение с Исходными Кодами Salvo Pro

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

Листинг 3: Пример salvocfg.h для Построения с Исходными Кодами

Эффективность

Использование Памяти

учебные примеры[5]	общее ROM[6]	общее RAM[7]
tu1lite	660	42
tu2lite	868	42
tu3lite	948	42
tu4lite	1572	66
tu5lite	2272	95
tu6lite	2416	97
tu6pro[8]	2272	93

Таблица 3: Требования Salvo к памяти ROM и RAM для Приложения Salvo, Созданного Компилятором Си Keil's CARM

Специальные Вопросы

Аппаратная Независимость

Код Salvo для ARM является *чистым* кодом ARM7TDMI. Он не делает никаких предположений или ссылок на зависимую от производителя периферию, находящуюся в каком-либо контроллере ARM7TDMI.

Замечание: Из-за аппаратной независимости необходимо включать в проект аппаратно зависимые заголовочные файлы (т.е. `#include <LPC21xx.h>`) в каждый исходный или конфигурационный файл Salvo. Процессорно-зависимые заголовочные файлы необходимы в пользовательских файлах, осуществляющих доступ к встроенной периферии.

Автоматические Переменные в Задачах Salvo

Из-за деталей исполнения, полный размер всех авто / локальных переменных в каждой задаче Salvo ограничен 65,532 байтами. Маловероятно, что это вызовет какие-либо проблемы в реальных приложениях. Нет никаких ограничений на число и размер авто / локальных переменных в любых других функциях приложения Salvo.

Режимы ARM и Thumb CPU

Построение с Библиотеками

Приложения, которые выполняются исключительно в режимах ARM или Thumb, должны использовать только Salvo библиотеки ARM или Thumb для сокращения размера объектного кода. Если приложение смешивает эти два режима, вместо этого должна использоваться библиотека Salvo со смешанным режимом центрального процессора (библиотека *смешанного режима* Salvo). Во всех случаях, компоновщик выдаст предупреждающие сообщения, если соответствующая функция Salvo (режима ARM или Thumb)[9] не может быть найдена.

Замечание: Макетные версии определяемых пользователем функций управления Salvo (см. Пользовательские Функции управления, ниже) включены в каждую библиотеку Salvo. Чтобы избежать предупреждений во время компоновки и/или ошибок при построении библиотечного проекта используйте библиотеки смешанного режима Salvo. Директива `INTERWORK` должна быть применена к исходным модулям пользователя, которые содержат функции управления.

Построение с Исходным Кодом

Компилятор Си Keil's CARM имеет директивы командной строки для компиляции в режимах ARM (`ARM`) или Thumb (`THUMB`, по умолчанию), а также позволяет смешанный режим CPU (`INTERWORK`). Эти директивы также доступны в µVision3 IDE как опции проекта. В построении с исходными кодами Salvo эти директивы должны применяться одинаково ко всем исходным файлам проекта Salvo.

Переключатель контекста Salvo (`portkcarm.s`) написан на ассемблере и может конфигурироваться через два определенных символа – `MAKE_FOR_ARM` и `MAKE_FOR_INTERWORK`. Ненулевое значение `MAKE_FOR_ARM` вызывает режим кода ARM. Любое другое значение или его отсутствие приводят к режиму Thumb. Точно так же ненулевое значение `MAKE_FOR_INTERWORK` производит модуль, который поддерживает смешанный режим CPU, и может вызываться из режима ARM и Thumb, независимо от установки `MAKE_FOR_ARM`. Любое другое значение или его отсутствие производят неопределенный модуль.

символ	значение	эффект
`MAKE_FOR_ARM`	0	Thumb режим
`MAKE_FOR_ARM`	> 0	ARM режим
`MAKE_FOR_INTERWORK`	0	смешанный режим CPU запрещен
`MAKE_FOR_INTERWORK`	> 0	смешанный режим CPU разрешен

Таблица 4: Значения Символов и Эффект при Ассемблировании portkcarm.s

Следующие символы определены директивой ассемблера `SET`:

```
AA.EXE ... SET(MAKE_FOR_ARM=0, MAKE_FOR_INTERWORK=1)
```

Эти символы могут быть установлены из µVision3 IDE.

Пользовательские Функции Управления

Прерывания в Критических Секциях

Имеется полный контроль над прерываниями в приложении Salvo. Дистрибуции Salvo для ARM требуют, чтобы следующие четыре функции пользователя были определены для управления прерываниями в критических секциях Salvo:

```
OSDisableInts()
OSEnableInts()
OSRestoreInts()
OSSaveInts()
```

Дополнительная информация об этих функциях пользователя может быть найдена в *Руководстве Пользователя Salvo*.

Пример иллюстрирует использование этих функций. Предполагается, что Salvo `OSTimer()` вызывает Philips LPC2129 ARM7TDMI's Timer 0 каждые 10ms. Бит разрешения прерывания Timer 0 процессора LPC2129 расположен в `VICIntEnable[4]`[10]. Никакие другие сервисы Salvo не вызываются из переднего плана / уровня прерывания. Простая схема управления прерыванием, которая гарантирует надлежащую операцию Salvo и вновь разрешает прерывание от Timer 0, показана ниже:

```
void OSDisableInts(void)
{
  VICIntEnClr |= 0x00000010;
}

void OSEnableInts(void)
{
  VICIntEnable |= 0x00000010;
}

void OSSaveInts(void)
{
  ;
}

void OSRestoreInts(void)
{
  VICIntEnable |= 0x00000010;
}
```

Эта группа функций гарантирует, что на входе (`OSSaveInts()`, `OSDisableInts()`) в критические секции Salvo, прерывания Timer 0 запрещены, а на выходе (`OSRestoreInts()`) они вновь разрешены. Вы можете вызвать `OSEnableInts()` в начале приложения Salvo, чтобы разрешить прерывания от Timer 0.

Вышеописанные функции особенно просты для микроконтроллера LPC2129 из-за его способности независимо устанавливать и сбрасывать биты разрешения индивидуальных прерываний. Схема может быть легко расширена на многие источники прерывания, если дополнительные сервисы Salvo (например. `OSSignalBinSem()`) вызываются из других ISR.

В этом примере, нет никакой потребности сохранять и позднее восстанавливать состояния битов разрешения прерывания из-за способности LPC2129 индивидуально устанавливать и очищать эти биты. Другие биты разрешения прерываний, которые не связаны с сервисами Salvo, не будут затронуты функциями, описанными выше.

Замечание: Если сервисы Salvo вызываются из фонового режима, запрещение прерываний в течение критических секций Salvo не требуется.

Сторожевой Таймер

Сторожевой таймер процессора (если существует) может быть автоматически очищен по каждому требованию диспетчера Salvo через соответствующее определение функции пользователя `OSClrWDT()`.

Построение с Исходным Кодом

Пользователи Salvo Pro, желающие минимизировать размер кода, могут строить приложения и собственные библиотеки, переопределяя следующие макроопределения в `salvocfg.h` проекта, таким образом, избегая вызовов пользовательских функций, упомянутых выше. Сокращение размера кода будет минимально и, возможно, не совместимо между различными ARM7TDMI микроконтроллерами.

```
OSDi()
OSEi()
OSEnterCritical()
OSLeaveCritical()
OSResumeCriticalSection()
OSSuspendCriticaslSection()
```

---

[1] Выполняется автоматически при помощи символов `_KEIL_` и `_CA_`, определяемых компилятором.
[2] Режим Thumb является CARM режимом по умолчанию.
[3] Каждый флаг события располагается в своем собственном блоке управления флагом в RAM.
[4] Каждая очередь сообщений располагается в своем собственном блоке управления очередью в RAM.
[5] Salvo v3.3.0-dev8 с DK-ARM v1.4b. CARM в Thumb режиме, установка Optimization Level 7, с акцентом на лучший размер кода.
[6] В байтах, как определено в `Program Size: code-cc`. Включает CARM стартовый файл `Startup.s` (более 250 байт).
[7] В байтах, `DATA`, как определено в `Memory Map` в файле проекта `.map` file. Это общий объем RAM Salvo и требований приложения. Не включает автоматически размещаемую компилятором RAM (1168 bytes) для стека. Замечание: `Program Size: data-dd` включает размер стека.
[8] Построение с Salvo Pro несколько отличается от Salvo Lite при конфигурировании – см. учебные `salvocfg.h`.
[9] Функции Thumb-режима имеют суффиксы `?T` добавляемые к их именам, а функции ARM-режима имеют суффиксы `?A` добавляемые к именам.
[10] Биты устанавливаются записью 1 в `VICIntEnable`, и очищаются записью 1 в `VICIntEnClr`. Запись 0 не будет иметь эффекта.