Section reference Ссылка на раздел

• Сводка разделов и блоков.
• Описание разделов и блоков.
Для получения дополнительной информации см. Модули и разделы, стр. 96.

Сводка разделов и блоков

В этой таблице перечислены разделы и блоки ELF, которые используются инструментами сборки IAR:

Section	Description
.bss	Holds zero-initialized static and global variables. Содержит инициализированные нулем статические и глобальные переменные.
CSTACK	Holds the stack used by C or C++ programs. Содержит стек, используемый программами C или C ++.
.data	Holds static and global initialized variables. Содержит статические и глобальные инициализированные переменные.
.data_init	Holds initial values for .data sections when the linker directive initialize is used. Содержит начальные значения для секций .data при использовании директивы компоновщика initialize.
.exc.text	Holds exception-related code. Содержит код, связанный с исключениями.
HEAP	Holds the heap used for dynamically allocated data. Содержит кучу, используемую для динамически выделяемых данных.
__iar_tls$$DATA	Holds initial values for TLS variables. Содержит начальные значения для переменных TLS.
.iar.dynexit	Holds the atexit table. Содержит таблицу atexit.
.iar.locale_table	Holds the locale table for the selected locales. Содержит таблицу языковых стандартов для выбранных языковых стандартов.
.init_array	Holds a table of dynamic initialization functions. Содержит таблицу функций динамической инициализации.
.intvec	Holds the reset vector table Содержит таблицу векторов сброса

IRQ_STACK	Holds the stack for interrupt requests, IRQ, and exceptions. Содержит стек для запросов прерывания, IRQ и исключений.
.noinit	Holds __no_init static and global variables. Содержит статические и глобальные переменные __no_init.
.preinit_array	Holds a table of dynamic initialization functions. Содержит таблицу функций динамической инициализации.
.prepreinit_array	Holds a table of dynamic initialization functions. Содержит таблицу функций динамической инициализации.
.rodata	Holds constant data. Содержит постоянные данные (константы).
.text	Holds the program code. Содержит программный код.
.textrw	Holds __ramfunc declared program code. Содержит объявленный программный код.
.textrw_init	Holds initializers for the .textrw declared section. Содержит инициализаторы для объявленной секции .textrw.
Veneer$$CMSE	Holds secure gateway veneers. Для надежных межсетевых покрытий.

В дополнение к разделам ELF, используемым в вашем приложении, инструменты используют ряд других разделов ELF для различных целей:
• Разделы, начинающиеся с .debug, обычно содержат отладочную информацию в формате DWARF.
• Разделы, начинающиеся с .iar.debug, содержат дополнительную отладочную информацию в формате IAR.
• Раздел .comment содержит инструменты и командные строки, используемые для создания файла.
• Разделы, начинающиеся с .rel или .rela, содержат информацию о перемещении ELF.
• Раздел .symtab содержит таблицу символов для файла.
• Раздел .strtab содержит имена символов в таблице символов.
• Раздел .shstrtab содержит названия разделов.

Описание разделов и блоков

В этом разделе содержится справочная информация о каждом разделе, где:
•Описание описывает, какой тип содержания содержит раздел и, если необходимо, как этот раздел обрабатывается компоновщиком.
•Размещение памяти описывает ограничения размещения памяти.

Для получения информации о том, как выделить разделы в памяти путем изменения файла конфигурации компоновщика, см. Размещение кода и данных - файл конфигурации компоновщика, стр. 99.

.bss
Содержит инициализированные нулем статические и глобальные переменные.
Этот раздел можно разместить в любом месте памяти.

CSTACK
Блок, содержащий внутренний стек данных.
Размещение в памяти Этот блок можно разместить в любом месте памяти.
См. Также Настройка стековой памяти, стр. 118.

.data
Содержит статические и глобальные инициализированные переменные. В объектных файлах сюда входят начальные значения. Когда используется инициализация директивы компоновщика, соответствующий ***.data_init***
Для каждого раздела .data создается раздел, содержащий возможно сжатые начальные значения.
Этот раздел можно разместить в любом месте памяти.

.data_init
Содержит возможно сжатые начальные значения для секций ***.data***. Этот раздел создается компоновщиком, если используется директива компоновщика инициализации.
Этот раздел можно разместить в любом месте памяти.

.exc.text
Содержит код, который выполняется только тогда, когда ваше приложение обрабатывает исключение.
В той же памяти, что и ***.text.***
См. Также Обработка исключений, стр. 206.

HEAP
Содержит кучу, используемую для динамически выделяемых данных в памяти, другими словами, данные, выделенные malloc и free, а в C ++ **-** ***new*** и ***delete***.
Этот раздел можно разместить в любом месте памяти.
См. Также Настройка кучи памяти, стр. 118.

__iar_tls$$DATA
Содержит начальные значения для переменных TLS. Этот раздел создается компоновщиком, если используется параметр компоновщика ***--threaded_lib***.
В дополнение к обычным способам доступа к началу, концу и размеру этого блока (см. Операторы выделенного раздела, стр. 200), вы также можете использовать оператор ***__iar_tls $$ DATA $$ Align*** для доступа к выравниванию области локальной памяти потока в исходном коде C / C ++.

Этот раздел можно разместить в любом месте памяти.
См. Также Управление многопоточной средой, стр. 165

.iar.dynexit
Содержит таблицу вызовов, которые необходимо сделать при выходе.
Этот раздел можно разместить в любом месте памяти.
См. Также Установка лимита atexit, стр. 118.

.iar.locale_table
Содержит таблицу языковых стандартов для выбранных языковых стандартов.
См. Также Locale, стр. 164.

.init_array
Содержит указатели на подпрограммы, вызываемые для инициализации одного или нескольких объектов C ++ со статической продолжительностью хранения.
Этот раздел можно разместить в любом месте памяти.

.intvec
Содержит таблицу векторов сброса и векторы исключений, которые содержат инструкции перехода для ***cstartup***, процедуры обслуживания прерывания и т. д. Размещение этого раздела зависит от устройства. См. Руководство производителя оборудования.

IRQ_STACK
Содержит стек, который используется при обслуживании исключений IRQ. Другие стеки могут быть добавлены по мере необходимости для обслуживания других типов исключений: FIQ, SVC, ABT и UND. Файл ***cstartup.s*** необходимо изменить, чтобы инициализировать используемые указатели стека исключений.
Примечание: этот раздел не используется при компиляции для Cortex-M.
Этот раздел можно разместить в любом месте памяти.
См. Также Стек исключений, стр. 216.

.noinit
Содержит статические и глобальные переменные ***__no_init***.
Этот раздел можно разместить в любом месте памяти.

.preinit_array
Подобно .init_array, но используется библиотекой для выполнения некоторых инициализаций C ++ раньше других.
Этот раздел можно разместить в любом месте памяти.
См. Также .init_array, стр. 534.

.prepreinit_array
Описание Подобно ***.init_array***, но используется, когда статическая инициализация C переписывается как динамическая инициализация. Выполняется перед динамической инициализацией C ++.

Этот раздел можно разместить в любом месте памяти.
См. Также .init_array, стр. 534.

.rodata

Описание Содержит постоянные данные. Сюда могут входить постоянные переменные, строковые и агрегатные литералы и т. д.
Этот раздел можно разместить в любом месте памяти.

.text

Содержит программный код, включая код для инициализации системы.
Этот раздел можно разместить в любом месте памяти.

.textrw

Содержит объявленный программный код ***__ramfunc***.
Этот раздел можно разместить в любом месте памяти.
См. Также ***__ramfunc***, стр. 393.

.textrw_init

Содержит инициализаторы для объявленных разделов ***.textrw***.
Этот раздел можно разместить в любом месте памяти.
См. Также ***__ramfunc***, стр. 393.

Veneer$$CMSE

В этом разделе содержатся защитные оболочки шлюза, автоматически создаваемые компоновщиком для каждой функции входа, как определено расширенным ключевым словом ***__cmse_nonsecure_entry***.
Этот раздел должен быть помещен в область памяти ***NSC*** (non-secure callable незащищенный вызов). Области ***NSC*** могут быть запрограммированы с использованием ***SAU*** (security attribution unit блок атрибуции безопасности) или ***IDAU*** (implementation-defined attribute unit блок атрибута, определяемый реализацией). Для получения информации о том, как программировать ***SAU*** или ***IDAU***, смотрите документацию к вашему ядру Armv8-M.
См. Также ***Arm TrustZone®***, стр. 232, ***--cmse***, стр. 282, ***__cmse_nonsecure_entry***, стр. 387, и ***--import_cmse_lib_out***, стр. 344