**Описание выходных секций.**

Команда скрипта компоновщика SECTIONS состоит из набора команд «sections-command», каждая из которых может быть представлена описанием выходной секции. Рассмотрим все выходные секции для этой команды.

* Каждая выходная секция выровнена по 4-ёх байтовой границе.
* Описание выходной секции определяет включаемые в неё входные секции.
* Промежутки между входными секциями (NB: могущие появляться в зависимости от требований к выравниванию входных секций) заполняются паттерном 0xFF.

**.data**

Это секция инициализированных данных. Данная секция направляется в регион RAM.

Исполняемая программа знает, что секция расположена в RAM, но загрузчик помещает начальные значения во FLASH (inidata). Таким образом, одна из задач исполняемого стартового кода – это скопировать начальные значения из FLASH в RAM.

Для инициализации секции стартовым кодом скрипт компоновщика определяет три символа (NB: каждый символ помещается в символьную таблицу, для них определяются адреса, но реально память под них не резервируется, и в них ничего нельзя записать. Единственная применимая операция к символу, объявленному в скрипте компоновщика – это операция получения адреса):

|  |  |
| --- | --- |
| Символ | Назначение |
| \_sidata | LMA-адрес (адрес загрузки секции .data: \_sidata = LOADADDR(.data);) |
| \_sdata | Адрес начала секции .data |
| \_edata | Адрес конца секции .data |

В выходную секцию .data включаются следующие входные секции:

\*(.data\_begin .data\_begin.\*) /\* NB: тестовая секция \*/

\*(.data .data.\*)

\*(.data\_end .data\_end.\*) /\* NB: тестовая секция \*/

**.data\_CCMRAM.**

Это вторая секция инициализированных данных для памяти core-coupled memory (память, сопряженная с процессором). Соответственно, направляется в регион CCMRAM. Для инициализации нескольких секций данных используется другая техника – массив инициализации регионов данных, которая будет описана далее. Такая техника позволяет определить всего два символа скрипта для всего массива инициализации, а не определять для каждого региона свою тройку (\_sidata, \_sdata, \_edata) (NB: это ничего не экономит, просто создаётся меньше символов в символьной таблице). В выходную секцию .data\_CCMRAM включаются следующие входные секции:

\*(.data.CCMRAM .data.CCMRAM.\*)

**.bss.**

Основная секция неинициализированных данных. Данная секция направляется в регион RAM и не является загружаемой. Стартовый код должен обнулить память для этой секции. Соответственно, чтобы получить доступ к ней, в скрипте компоновщика определяются символы:

|  |  |
| --- | --- |
| Символ | Назначение |
| \_sbss | Адрес начала секции .bss |
| \_ebss | Адрес конца секции .bss |

В выходную секцию .bss включаются следующие входные секции:

\*(.bss\_begin .bss\_begin.\*) /\* NB: тестовая секция \*/

\*(.bss .bss.\*)

\*(COMMON) /\* символы, не попавшие ни в одну секцию \*/

\*(.bss\_end .bss\_end.\*) /\* NB: тестовая секция \*/

**.bss\_CCMRAM.**

Вторая секция неинициализированных данных для памяти core-coupled memory (память, сопряженная с процессором). Направляется в регион CCMRAM. Для очистки памяти, соответствующей этой секции также используется массив очистки регионов данных. При этом необходимо объявить всего два символа в скрипте компоновщика.

**.inits.**

Секция массивов инициализации регионов памяти, направляется в регион FLASH. Есть два вида массивов для каждого RAM региона (NB: RAM, CCMRAM и пр.), один для данных (NB: имеет размерность [3]), второй для bss (NB: имеет размерность [2]). Стартовый код выполняет проход по каждому массиву и выполняет соответствующую инициализацию регионов. Сколько регионов, столько же и будет массивов каждого вида. В этой секции массивы размещаются последовательно, один за другим: сперва массивы инициализации данных (для каждого региона), затем массивы очистки (для каждого региона).

Массив инициализации данных включает следующие элементы:

* LMA (адрес загрузки data-секции)
* Адрес начала data-секции.
* Адрес конца data-секции.

Массив очистки bss включает следующие элементы:

* Адрес начала bss-секции.
* Адрес конца bss-секции.

Обязательным требованием к этой секции является выравнивание по границе слова (4 байта), так как код инициализации работает только по словам.

Для каждого вида массива определяется пара символов в символьной таблице, чтобы стартовый код смог получить к ним доступ.

Для массива инициализации данных

|  |  |
| --- | --- |
| Символ | Назначение |
| \_\_data\_regions\_array\_start | Адрес размещения массивов инициализации данных |
| \_\_data\_regions\_array\_end | Значение счётчика положения после размещения массивов инициализации данных (совпадает с адресом размещения массивов очистки bss-секций) |

Для массива очистки данных

|  |  |
| --- | --- |
| Символ | Назначение |
| \_\_bss\_regions\_array\_start | Адрес размещения массивов очистки bss-секций |
| \_\_bss\_regions\_array\_end | Значение счётчика положения после размещения массивов bss-секций |

В данную выходную секцию включаются также старые секции инициализации, предназначенные для содержания «голого» (naked) кода, с прологом/эпилогом, добавленным файлами crti.o/crtn.o при компоновке с файлами стартового кода. Автономный код запуска в настоящее время не запускает их, предпочитая использовать массивы инициализации выше.

KEEP(\*(.init))

KEEP(\*(.fini))

Далее включаются секции с массивами указателей на функции. Эти секции относятся к стартовому и завершающему коду конструкторов и деструкторов C/C++, который вызывается перед/после main(). Секции .preinit\_array и .init\_array связаны с инициализацией C/C++ объектов, а секции .fini\_array связаны с финализацией кода (tear down). Стартовые и конечные символы определяют начало и конец кодовых секций, связанными с этими операциями и могут использоваться из других частей код поддержки среды исполнения.

.preinit\_array и .init\_array секции содержат массивы указателей на функции, которые будут вызваны при инициализации. .fini\_array – это массив указателей на функции, которые будут вызываться при финализации. Начальный и конечный символы используются для перемещения по этим спискам.

Таким образом, определяются следующие символы и включаются следующие секции.

Секции с указателями на функции инициализации объектов C/C++.

|  |  |
| --- | --- |
| Символ | Назначение |
| \_\_preinit\_array\_start | Адрес массива указателей на функции предварительной инициализации |
| \_\_preinit\_array\_end | Значение счётчика положения после размещения массива |

/\* Используется для запуска SystemInit() \*/

KEEP(\*(.preinit\_array\_sysinit .preinit\_array\_sysinit.\*))

/\* Используется для других инициализаций платформы \*/

KEEP(\*(.preinit\_array\_platform .preinit\_array\_platform.\*))

/\* Инициализация приложения. \*/

KEEP(\*(.preinit\_array .preinit\_array.\*))

Секции с указателями на конструкторы.

|  |  |
| --- | --- |
| Символ | Назначение |
| \_\_init\_array\_start | Адрес массива указателей на статические конструкторы |
| \_\_init\_array\_end | Значение счётчика положения после размещения массива |

KEEP(\*(SORT(.init\_array.\*)))

KEEP(\*(.init\_array))

Секции с указателями на деструкторы.

|  |  |
| --- | --- |
| Символ | Назначение |
| \_\_fini\_array\_start | Адрес массива указателей на статические деструкторы |
| \_\_fini\_array\_end | Значение счётчика положения после размещения массива |

KEEP(\*(SORT(.fini\_array.\*)))

KEEP(\*(.fini\_array))

NB: Таким образом, в секции .init нет никакого исполняемого кода! Здесь лежат только секции с массивами указателей, и определяются символы границ этих массивов, которые может использовать стартовый/финализирующий код.

**.text.**

Программный код сохраняется в эту секцию, которая направляется в регион FLASH. Наряду с кодом, сюда также включаются константные данные, таблицы виртуальных функций C++ и некоторые дополнительные секции.

\*(.text .text.\*)

\*(.rodata .rodata.\* .constdata .constdata.\*)

\*(vtable)

KEEP(\*(.eh\_frame\*))

Секция eh\_frame содержит размотку исключения и информацию о языке исходного кода. Каждая запись в этой секции представлена одним CFI (call frame information). Дополнительную информацию можно получить здесь:

<http://refspecs.linuxfoundation.org/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/ehframechpt.html>

Секции-затычки генерируются компоновщиком, чтобы склеить вместе ARM и Thumb код. .glue\_7 используется для вызова ARM кодом Thumb кода. .glue\_7t используется для вызовам Thumb кодом ARM кода. Очевидно, всегда генерируется компоновщиком для некоторых архитектур, так что лучше оставить их здесь.

\*(.glue\_7)

\*(.glue\_7t)

**.ARM.extab и .ARM.exidx.**

Так называемые «магические» секции ARM. Они относятся к размотке. Вы можете найти больше информации в документе «ELF для ARM архитектуры» (aka IHI0044E). Они вам не нужны, если вас не волнует размотка (размотка полезна для исключений C++ и для отладки). Подробнее здесь:

<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0044f/index.html>