

[Home \(/s/\)](#)[Knowledge \(/s/communityknowledge/\)](#)[Log in / Register](#)

Rate This Article :

Bootloader Linker Scripts for PIC32 Devices

🕒 Nov 3, 2020 · Knowledge

Article Number

000012179

Title

Bootloader Linker Scripts for PIC32 Devices

Article URL

<https://microchipsupport.force.com/s/article/Bootloader-Linker-Scripts-for-PIC32-Devices>
(<https://microchipsupport.force.com/s/article/Bootloader-Linker-Scripts-for-PIC32-Devices>)

Question

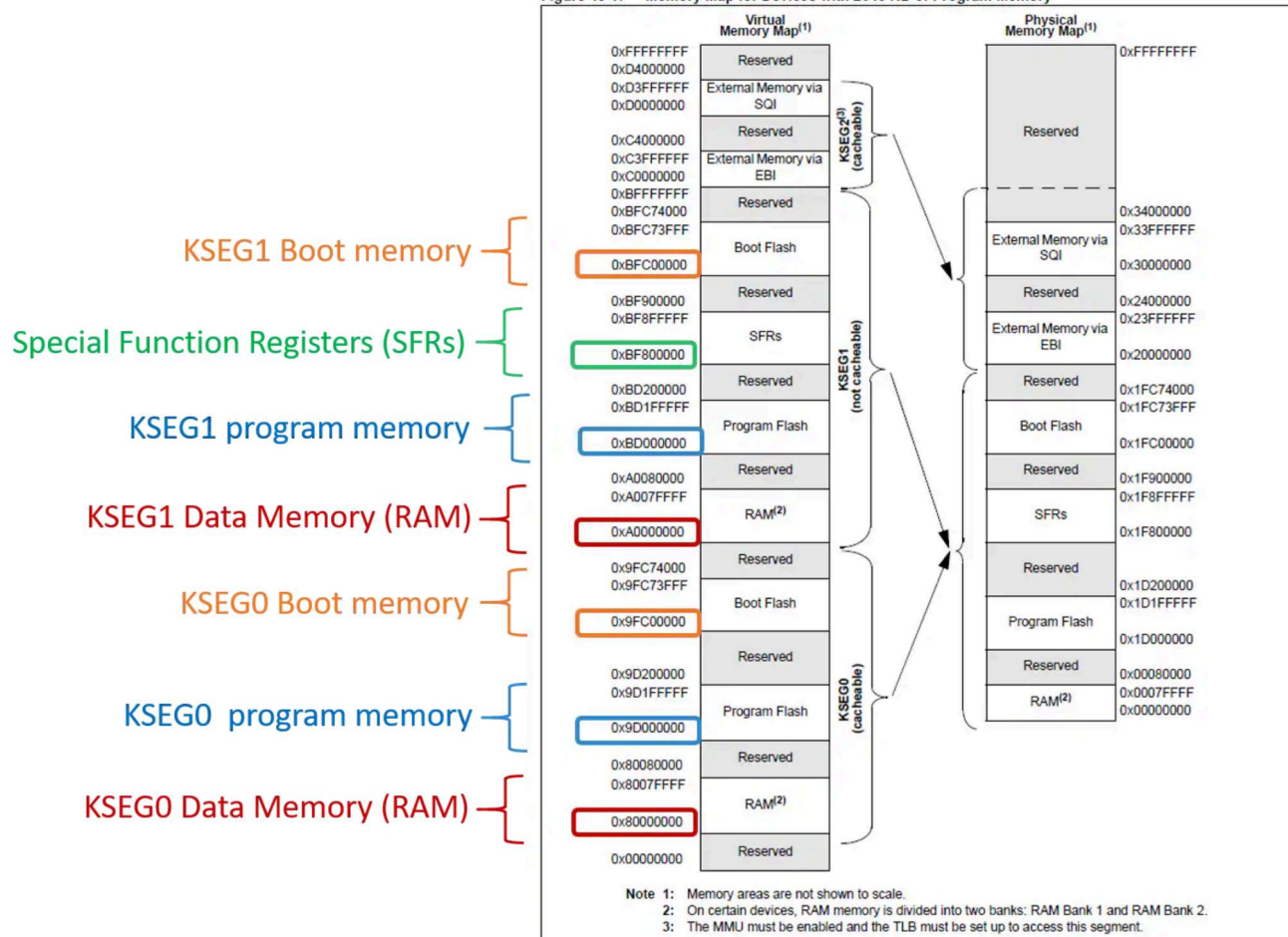
What are the differences between an application linker script and a bootloader linker script for PIC32 devices?

Answer

Depending on whether a program is a bootloader or an application, the linker script that is generated by Harmony will put the program into different areas of memory.

The datasheet for any PIC32 device will show a Memory Map describing the addresses of the various sections of memory in that device, such as the one below:

Figure 48-1: Memory Map for Devices with 2048 KB of Program Memory



It is the job of the linker script to make sure that each part of the program is put into the correct region of memory. When a program is built, the memory usage that the linker script has implemented for that project can be seen by looking at the MAP file, which is usually located in the following directory:

ProjectName/firmware/<ProjectName.X>/dist/default/ProjectName.production.MAP

For a standalone application, the linker script and the memory usage is explained as follows:

Memory usage for standalone app

For a bootloader, the difference is that the bootloader program is usually put into a different location in memory to a standalone application:

Memory usage for bootloader

When an application is to be used along with a bootloader, usually a custom linker script should be used (Harmony can generate this) and it must be carefully examined to make sure that the program and reset address are set up correctly, as shown in the example below:

Linker script for app to be used with a bootloader

```

*****
PROVIDE( vector_exception = 0x00000000 );
PROVIDE( _ebase_address = 0x9d001000 );
*****
/* Memory Address Equates
 * _RESET_ADDR          -- Reset Vector
 * _REV_EXCPT_ADDR      -- Boot exception Vector
 * _DBG_EXCPT_ADDR      -- In-circuit Debugging Exception Vector
 * _SIMPLE_TLB_REFILL_EXCPT_ADDR -- Simple TLB-Refill Exception Vector
 * _CACHE_ERR_EXCPT_ADDR -- Cache-error Exception Vector
 * _GEN_EXCPT_ADDR      -- General Exception Vector
 */
*****
_RESET_ADDR          = 0x9d000000;
_SIMPLE_TLB_REFILL_EXCPT_ADDR = _ebase_address + 0;
_CACHE_ERR_EXCPT_ADDR = _ebase_address + 0x100;
_GEN_EXCPT_ADDR      = _ebase_address + 0x180;
*****
/* Memory Regions
 *
 * Memory regions without attributes cannot be used for orphaned sections.
 * Only sections specifically assigned to these regions can be allocated
 * into these regions.
 *
 * The Debug exception vector is located at 0x9FC00480.
 * The config <address> sections are used to locate the config words at
 * their absolute addresses.
 */
*****
MEMORY
{
  kseg0_boot_mem : ORIGIN = 0x9d000000, LENGTH = 0x400
  kseg0_program_mem (x1) : ORIGIN = 0x9d000000 + 0x400, LENGTH = 0x200000 - 0x400
  kseg0_data_mem (x1) : ORIGIN = 0x9d000000, LENGTH = 0x400
  kseg1_data_mem (x1) : ORIGIN = 0x9d000000, LENGTH = 0x400
  kseg2_ehi_data_mem : ORIGIN = 0x00000000, LENGTH = 0x4000000
  kseg2_sqi_data_mem : ORIGIN = 0x00000000, LENGTH = 0x4000000
  kseg3_ehi_data_mem : ORIGIN = 0x00000000, LENGTH = 0x4000000
}

```

ebase address is now located in program flash along with the application program

The reset address is now at the start of program memory, because the bootloader will have already run first (from boot memory), followed by the application

Linker script will start filling the memory from this address with the application program – note that its in the program flash region of memory

Note that program memory is defined as the address of of "Program flash" in the memory map, because the "program" that is going there is the application

Memory usage for app with bootloader

Microchip PIC32 Memory-Usage Report

kseg0 Program-Memory Usage	section	address	length [bytes]	(dec)	Description
	.text.vfprintf	0x9d000480	0x20c8	8392	
	.rodata	0x9d002548	0x1758	5976	Read-only const
	.text.TCPIP_HTTP_Proces	0x9d003ca0	0x158c	5516	
	.text.TCPIP_ICMPV6_Proc	0x9d00522c	0x158c	5516	
	.text.TcpHandleSeg	0x9d0067b8	0x8dc	3468	
	.text.vfprintf_cdnpsu	0x9d007544	0xc8c	3212	
	.text.TCPIP_HTTP_PostEx	0x9d0081d0	0xb90	2960	
	.dinit	0x9d008460	0xb30	2864	
	.rodata	0x9d008890	0xa88	2792	Read-only const
	.text.Transform	0x9d00a378	0xa8c	2700	
	.text.DRV_USBHS_DEVICE	0x9d00aa04	0xa40	2624	
	.text.TcpSend	0x9d00b844	0x974	2420	
	.text.TCPIP_STACK_Task	0x9d00c1b8	0x8e4	2276	
	.text.TCPIP_UDP_Task	0x9d00ca9c	0x8cc	2252	
	.text.Command_DNS_Serv	0x9d00d368	0x8c4	2244	
	.text.TCPIP_TCP_Task	0x9d00dc2c	0x804	2052	
	.text.TCPIP_DHCP_Task	0x9d00e430	0x7c0	1984	
	.text.DRV_ETHMAC_PIC32M	0x9d00ebf0	0x7a8	1960	
	.text.SYS_FS_MEDIA_MANA	0x9d00f398	0x6e0	1760	
	.text.TCPIP_IPV6_Packet	0x9d00fa78	0x588	1416	
	.text.TCPIP_IPV6_Packet	0x9d00fa78	0x10	16	
	.text.DRV_ETHMAC_ILMMAC	0x9d010010	0xf0	240	
	.cache_err_excpt	0x9d010100	0x10	16	
	.text.TCPIP_HTTP_Print	0x9d010110	0x70	112	
	app_excpt	0x9d010180	0x10	16	General-Exception
	.text.EnetPoolFreeDpt	0x9d010190	0x70	112	
	.text._on_bootstrap	0x9d04e4a8	0x8	8	
	.rodata	0x9d04e4b0	0x8	8	Read-only const
	._MMIO_MEDIA_DATA_MMIO_MED	0x9d000000	0x100000	65536	
	Total kseg0_program_mem used :		0x5e030	385072	18.4% of 0x1ffb80
kseg0 Boot-Memory Usage	section	address	length [bytes]	(dec)	Description
	Total kseg0_boot_mem used :		0	0	
kseg1 Boot-Memory Usage	section	address	length [bytes]	(dec)	Description
	.reset	0x9d000000	0x1b4	436	Reset handler
	Total kseg1_boot_mem used :		0x1b4	436	37.8% of 0x400
	Total Program Memory used :		0x5e1e4	385508	18.4% of 0x200000
kseg0 Data-Memory Usage	section	address	length [bytes]	(dec)	Description
	.sdata	0x80000000	0x5c	92	Small init data
	.sbss	0x8000000c	0x2ac	684	Small uninit data
	.bss	0x80000010	0x23c0	9152	Uninitialized data
	.bss	0x80000ef60	0x8c8	2248	Uninitialized data

URL Name

Bootloader-Linker-Scripts-for-PIC32-Devices



(<https://www.microchip.com>)

Legal (<https://www.microchip.com/legal>) | Privacy Policy

(<https://www.microchip.com/en-us/about/legal-information/privacy-policy>) |

Cookies (<https://www.microchip.com/en-us/about/legal-information/microchip-cookie-statement>) |

©Copyright 1998-2024 Microchip Technology Inc. All rights reserved.
(<https://www.microchip.com>)