NXP Semiconductors
Application Note

Document Number: AN5234
Rev. 2, May 2016

MPC5743R Single Core Initialization

by: Randy Dees and Shanaka Yapa

Contents

1 Introduction

The MPC574xR family of microcontrollers are based on the Power Architecture® cores and the enhanced Time Processing Unit (eTPU). They are primarily intended for Automotive Powertrain applications. They can be used in other types of applications, including motor control. Some of the devices in the MPC574xR family include dual independent cores. The dual-core capability, however, is a single core plus a second core, that for safety applications, includes a delayed lock-step core. From an overall device standpoint, there are three cores instantiated in the MCU. The delayed lock-step core executes the same instructions to support safety critical applications. The delay reduces the simultaneous switching currents of the cores by having the cores execute with a delay.

The MPC5743R is a single core version of the MPC574xR. The single-core device includes the safety core with the delayed lock-step core. All of the current MPC574xR family of devices implement all of the cores, however, on the single core device, the second core must only be used for the initial boot operation of the device and should be disabled in the Boot Header Configuration word. It should not be used by user application code.

This application note describes the proper initialization of the single core (with delayed lock-step core) MPC5743R device.

2 MPC574xR family device differences

A summary of the primary differences between the different devices in the MPC574xR family is shown in the following table. (Only the differences are shown in this table. Features that are the same are not listed, see the MPC5746R Reference Manual for the complete features.)

Table 1. MPC574xR family of devices

Feature		MPC5746R	MPC5745R	MPC5743R
Main core		2[1]	2[1]	1[1]
Locked Step Core		1	1	1
Flash memory	Total	4256 KB + 16 KB[2]	3232 KB + 16 KB[2]	2208 KB + 16 KB[2]
	Code Flash	3984 KB	2960 KB	1936 KB
	BAF	16 KB	16 KB	16 KB
	Data flash	256 KB	256 KB	256 KB
System RAM		256 KB	192 KB	128 KB
Core frequency		200 MHz	150 MHz[3]	200 MHz
LINFlex		4+2MSB support	4+2MSB support	3+2MSB support[4]
DSPI		5+2MSB support	5+2MSB support[5]	4+2MSB support[6]
Package	LQFP144	—	—	•
	LQFP176	—	•	•
	MAPBGA252	•	•	—
	MAPBGA292 Emulation Device	•	—	—

1. Core0 + Locked step core will be supported. Core 1 will be used for Boot Assist Flash (BAF) execution at start-up only.
2. User test flash (UTEST)
3. 252 BGA package supports 200 MHz operation.
4. LINFlex 3 is not supported.
5. DSPI 4 is not supported on the 176 LQFP device.
6. DSPI 4 is not supported. In addition, DSPI M1 is not supported on the 144 LQFP device.

2.1 MPC574xR flash definition

The different devices in the MPC574xR family implement different flash memory sizes.

A summary of the flash size allocation is shown in the following table.

Table 2. Flash usage allocation

Description	MPC5746R	MPC5745R	MPC5743R
One-time Programmable (OTP)[1]	32 KB	32 KB	32 KB
Data Flash (for EEPROM emulation)	256 KB	256 KB	256 KB
Code Flash	3984 KB	2960 KB	1936 KB

1. The 16 KB of the OTP area is reserved for the Boot Assist Flash (BAF) and 16 KB for device specific definitions and Device Configuration Format (DCF) records (User Test/UTEST).

The table below shows the implemented flash blocks for each of the defined devices, as well as the partition number (for read/while write restrictions[1]).

Table 3. Flash memory definition

Start Address	End Address	Allocated Size [KB]	Description	RWW Partition	Block size	MPC5746R 3984 KB + 256 KB + 32 KB	MPC5745R 2960 KB + 256 KB + 32 KB	MPC5743R 1936 KB + 256 KB + 32 KB
Reserved - no overlay								
0x0000_0000	0x003F_FFFF	4096	Reserved flash	—	—			
UTEST NVM Block - no overlay								
0x0040_0000	0x0040_3FFF	16	UTEST NVM Block Space 16KB	Partition 0	16k	•	•	•
0x0040_4000	0x0040_7FFF	16	16KB BAF block0	Partition 0 - Low	16k	•	•	•
0x0040_8000	0x005F_FFFF	2016	Reserved flash	—	—			
Reserved - no overlay								
0x0060_0000	0x0067_FFFF	512	Reserved flash	—	—			
Reserved - no overlay								
0x0068_0000	0x007F_FFFF	1536	Reserved flash	—	—			
Data Flash - no overlay								
0x0080_0000	0x0080_3FFF	16	FMC EEPROM Block0	Partition 2 - Mid	16k	•	•	•
0x0080_4000	0x0080_7FFF	16	FMC EEPROM Block1	Partition 2 - Mid	16k	•	•	•
0x0080_8000	0x0080_BFFF	16	FMC EEPROM Block2	Partition 2 - Mid	16k	•	•	•
0x0080_C000	0x0080_FFFF	16	FMC EEPROM Block3	Partition 2 - Mid	16k	•	•	•
0x0081_0000	0x0081_3FFF	16	FMC EEPROM Block4	Partition 3 - Mid	16k	•	•	•
0x0081_4000	0x0081_7FFF	16	FMC EEPROM Block5	Partition 3 - Mid	16k	•	•	•
0x0081_8000	0x0081_BFFF	16	FMC EEPROM Block6	Partition 3 - Mid	16k	•	•	•
0x0081_C000	0x0081_FFFF	16	FMC EEPROM Block7	Partition 3 - Mid	16k	•	•	•
0x0082_0000	0x0082_FFFF	64	FMC EEPROM Block0	Partition 2 - Mid	64k	•	•	•
0x0083_0000	0x0083_FFFF	64	FMC EEPROM Block1	Partition 3 - Mid	64k	•	•	•

Table continues on the next page...

1. See the device reference manual for more information about the flash restrictions for reading while programming the flash.

Table 3. Flash memory definition (continued)

Start Address	End Address	Allocated Size [KB]	Description	RWW Partition	Block size	MPC5746R 3984 KB + 256 KB + 32 KB	MPC5745R 2960 KB + 256 KB + 32 KB	MPC5743R 1936 KB + 256 KB + 32 KB
0x0084_0000	0x009F_FFFF	1792	Reserved flash	—	—			
Low & Mid Flash Blocks - no overlay								
0x00A0_0000	0x00F9_BFFF	5744	Reserved flash	—	—			
0x00F9_C000	0x00F9_FFFF	16	16KB Code Flash block5	Partition 1 - Low	16k	•	•	•
0x00FA_0000	0x00FA_3FFF	16	16KB Code Flash block1	Partition 0 - Low	16k	•	•	•
0x00FA_4000	0x00FA_7FFF	16	16KB Code Flash block2	Partition 0 - Low	16k	•	•	•
0x00FA_8000	0x00FA_BFFF	16	16KB Code Flash block3	Partition 1 - Low	16k	•	•	•
0x00FA_C000	0x00FA_FFFF	16	16KB Code Flash block4	Partition 1 - Low	16k	•	•	•
0x00FB_0000	0x00FB_7FFF	32	32KB Code Flash block0	Partition 0 - Low	32k	•	•	•
0x00FB_8000	0x00FB_FFFF	32	32KB Code Flash block1	Partition 1 - Low	32k	•	•	•
0x00FC_0000	0x00FC_FFFF	64	64KB Code Flash block0	Partition 0 - Low	64k	•	•	•
0x00FD_0000	0x00FD_FFFF	64	64KB Code Flash block1	Partition 0 - Low	64k	•	•	•
0x00FE_0000	0x00FE_FFFF	64	64KB Code Flash block2	Partition 1 - Low	64k	•	•	•
0x00FF_0000	0x00FF_FFFF	64	64KB Code Flash block3	Partition 1 - Low	64k	•	•	•
Large Flash Blocks - no overlay								
0x0100_0000	0x0103_FFFF	256	256KB Code Flash block0	Partition 6 - 256K	256k	•	•	•
0x0104_0000	0x0107_FFFF	256	256KB Code Flash block1	Partition 6 - 256K	256k	•	•	•
0x0108_0000	0x010B_FFFF	256	256KB Code Flash block2	Partition 6 - 256K	256k	•	•	•
0x010C_0000	0x010F_FFFF	256	256KB Code Flash block3	Partition 7 - 256K	256k	•	•	•
0x0110_0000	0x0113_FFFF	256	256KB Code Flash block4	Partition 7 - 256K	256k	•	•	•
0x0114_0000	0x0117_FFFF	256	256KB Code Flash block5	Partition 7 - 256K	256k	•	•	•
0x0118_0000	0x011B_FFFF	256	256KB Code Flash block6	Partition 6 - 256K	256k	•	•	
0x011C_0000	0x011F_FFFF	256	256KB Code Flash block7	Partition 6 - 256K	256k	•	•	
0x0120_0000	0x0123_FFFF	256	256KB Code Flash block8	Partition 6 - 256K	256k	•	•	
0x0124_0000	0x0127_FFFF	256	256KB Code Flash block9	Partition 6 - 256K	256k	•	•	
0x0128_0000	0x012B_FFFF	256	256KB Code Flash block10	Partition 7 - 256K	256k	•		
0x012C_0000	0x012F_FFFF	256	256KB Code Flash block11	Partition 7 - 256K	256k	•		
0x0130_0000	0x0133_FFFF	256	256KB Code Flash block12	Partition 7 - 256K	256k	•		

Table continues on the next page...

Table 3. Flash memory definition (continued)

Start Address	End Address	Allocated Size [KB]	Description	RWW Partition	Block size	MPC5746R 3984 KB + 256 KB + 32 KB	MPC5745R 2960 KB + 256 KB + 32 KB	MPC5743R 1936 KB + 256 KB + 32 KB
0x0134_0000	0x0137_FFFF	256	256KB Code Flash block13	Partition 7 - 256K	256k	•		
0x0138_0000	0x01FF_FFFF	12800	Reserved flash	—	—			
Reserved - no overlay								
0x0200_0000	0x07FF_FFFF	98304	Reserved flash	—	—			
Mirror Reserved Flash								
0x0800_0000	0x089F_FFFF	10240	Reserved flash	—	—			
Mirror Low & Mid flash blocks								
0x08A0_0000	0x08F9_BFFF	5744	Reserved flash	—	—			
0x08F9_C000	0x08F9_FFFF	16	Mirror 16 KB Code Flash block5	Partition 1 - Low	16k	•	•	•
0x08FA_0000	0x08FA_3FFF	16	Mirror 16 KB Code Flash block1	Partition 0 - Low	16k	•	•	•
0x08FA_4000	0x08FA_7FFF	16	Mirror 16 KB Code Flash block2	Partition 0 - Low	16k	•	•	•
0x08FA_8000	0x08FA_BFFF	16	Mirror 16 KB Code Flash block3	Partition 1 - Low	16k	•	•	•
0x08FA_C000	0x08FA_FFFF	16	Mirror 16 KB Code Flash block4	Partition 1 - Low	16k	•	•	•
0x08FB_0000	0x08FB_7FFF	32	Mirror 32 KB Code Flash block0	Partition 0 - Low	32k	•	•	•
0x08FB_8000	0x08FB_FFFF	32	Mirror 32 KB Code Flash block1	Partition 1 - Low	32k	•	•	•
0x08FC_0000	0x08FC_FFFF	64	Mirror 64 KB Code Flash block0	Partition 0 - Low	64k	•	•	•
0x08FD_0000	0x08FD_FFFF	64	Mirror 64 KB Code Flash block1	Partition 0 - Low	64k	•	•	•
0x08FE_0000	0x08FE_FFFF	64	Mirror 64 KB Code Flash block2	Partition 1 - Low	64k	•	•	•
0x08FF_0000	0x08FF_FFFF	64	Mirror 64 KB Code Flash block3	Partition 1 - Low	64k	•	•	•
Mirror large flash blocks								

Table continues on the next page...

Table 3. Flash memory definition (continued)

Start Address	End Address	Allocated Size [KB]	Description	RWW Partition	Block size	MPC5746R 3984 KB + 256 KB + 32 KB	MPC5745R 2960 KB + 256 KB + 32 KB	MPC5743R 1936 KB + 256 KB + 32 KB
0x0900_0000	0x0903_FFFF	256	Mirror 256 KB Code Flash block0	Partition 6 - 256K	256k	•	•	•
0x0904_0000	0x0907_FFFF	256	Mirror 256 KB Code Flash block1	Partition 6 - 256K	256k	•	•	•
0x0908_0000	0x090B_FFFF	256	Mirror 256 KB Code Flash block2	Partition 6 - 256K	256k	•	•	•
0x090C_0000	0x090F_FFFF	256	Mirror 256 KB Code Flash block3	Partition 7 - 256K	256k	•	•	•
0x0910_0000	0x0913_FFFF	256	Mirror 256 KB Code Flash block4	Partition 7 - 256K	256k	•	•	•
0x0914_0000	0x0917_FFFF	256	Mirror 256 KB Code Flash block5	Partition 7 - 256K	256k	•	•	•
0x0918_0000	0x091B_FFFF	256	Mirror 256 KB Code Flash block6	Partition 6 - 256K	256k	•	•	
0x091C_0000	0x091F_FFFF	256	Mirror 256 KB Code Flash block7	Partition 6 - 256K	256k	•	•	
0x0920_0000	0x0923_FFFF	256	Mirror 256 KB Code Flash block8	Partition 6 - 256K	256k	•	•	
0x09240_000	0x0927_FFFF	256	Mirror 256 KB Code Flash block9	Partition 6 - 256K	256k	•	•	
0x0928_0000	0x092B_FFFF	256	Mirror 256 KB Code Flash block10	Partition 7 - 256K	256k	•		
0x092C_0000	0x092F_FFFF	256	Mirror 256 KB Code Flash block11	Partition 7 - 256K	256k	•		
0x0930_0000	0x0933_FFFF	256	Mirror 256 KB Code Flash block12	Partition 7 - 256K	256k	•		
0x0934_0000	0x0937_FFFF	256	Mirror 256 KB Code Flash block13	Partition 7 - 256K	256k	•		
0x0938_0000	0x0BFF_FFFF	49152	Mirror reserved space	—	—			

2.2 MPC574xR SRAM definition

The different devices in the MPC574xR family implement different SRAM memory sizes. The table below shows the implemented SRAM allocation for each of the defined devices.

- **System RAM** - The System RAM (SRAM) is intended for use for data storage and can be accessed by any core or by Direct Memory Access (DMA).
- **Local Instruction Memory** - A local Instruction SRAM (I-MEM) is available for each core. It allows fast access (one clock) to instructions. Code must be copied from flash into the I-MEM via program control.
- **Local Data Memory** - A local Data SRAM (D-MEM) is available for each core. It is commonly used for the processor stack and for critical variables that are primarily used by a single core. The access time is 1 clock for reads. Any other core also has access to the D-MEM, however, the access time is longer.

The CPU1 (Core 1) local memory should not be accessed by devices that are defined with a single core (MPC5743R).

NOTE

All SRAM must be initialized prior to use to set the Error Correction Code (ECC) for each SRAM location. The SRAM is initialized by writing a 64-bit double word to each location of the SRAM. Once initialized, writes to the SRAMs can be byte (8-bit), half-word (16-bit), word (32-bit), or double-word (64-bit).

Table 4. SRAM memory definition

Start Address	End Address	Allocated Size [KB]	Description	Actual size	MPC5746R 256 KB + 96 KB	MPC5745R 192 KB + 96 KB	MPC5743R 128 KB + 48 KB
System RAM							
0x4000_0000	0x4000_7FFF	32	System Standby RAM	32 KB	•	•	•
0x4000_8000	0x4001_FFFF	96	System RAM	96 KB	•	•	•
0x4002_0000	0x4002_FFFF	64	System RAM	64 KB	•	•	
0x4003_0000	0x4003_FFFF	64	System RAM	64 KB	•		
0x4004_0000	0x4FFF_FFFF	261888	Reserved	—			
Local Memory							
0x5000_0000	0x5000_3FFF	64	I-MEM CPU0	16 KB	•	•	•
0x5000_4000	0x5000_FFFF		Allocated for I-MEM CPU0	[48 KB]			
0x5001_0000	0x507F_FFFF	8128	Reserved I-MEM CPU0	—			
0x5080_0000	0x5080_7FFF	64	D-MEM CPU0	32 KB	•	•	•
0x5080_8000	0x5080_FFFF		Allocated D-MEM CPU0	[32 KB]			
0x5081_0000	0x50FF_FFFF	8128	Reserved D-MEM CPU0	—			
0x5100_0000	0x5100_3FFF	64	I-MEM CPU1	16 KB	•	•	
0x5100_4000	0x5100_FFFF		Allocated for I-MEM CPU1	[48K]			
0x5101_0000	0x517F_FFFF	8128	Reserved I-MEM CPU1	—			
0x5180_0000	0x5180_7FFF	64	D-MEM CPU1	32 KB	•	•	

Table continues on the next page...

Table 4. SRAM memory definition (continued)

0x5180_8000	0x5180_FFFF		Allocated D-MEM CPU1	[32 KB]			
0x5181_0000	0x51FF_FFFF	8128	Reserved D-MEM CPU1	—			

2.3 MPC574xR cores

The MPC574xR family of devices include either 1 or 2 main processing cores, On the two (2) core devices, both cores are e200z425 cores that can operate at up to 200 MHz. In addition, one of the two cores includes an e200z424 core, as a third core[2]. This core runs in a delayed lockstep mode (if enabled). The following table shows the cores that are available for use for the different devices in this family.

Table 5. Cores available in the MPC574xR devices

Device	Number of cores	Core 0	Core C (Lock-step core)	Core 1
MPC5746R	2 (3[1])	e200z425	e200z424	e200z425
MPC5745R	2 (3[1])	e200z425	e200z424	e200z425
MPC5743R	1 (2[1])	e200z425	e200z424	e200z425

1. Counting the lock-step core.

3 Start-up operation

The MPC5743R (single core versions of the MPC574xR family) boots from Core 1. However, this core is not available for use after the boot sequence is completed.

When exiting reset, Core 1 begins executing code from the Boot Assist Flash (BAF). The BAF code searches the internal flash for a valid boot header. The addresses of the boot header search locations are shown in the following table.

Table 6. MPC574xR boot flash addresses

Search order	Block	Address
1	16KB Code Flash block 0	0x00F9_C000
2	16KB Code Flash block 1	0x00FA_0000
3	16KB Code Flash block 2	0x00FA_4000
4	16KB Code Flash block 3	0x00FA_8000
5	16KB Code Flash block 4	0x00FA_C000
6	256KB Code Flash block 0	0x0100_0000
7	256KB Code Flash block 1	0x0104_0000
8	256KB Code Flash block 2	0x0108_0000

Table continues on the next page...

2. This core shares instruction cache, local memory interface, and bus interfaces as core 0. This core cannot be used as a standalone core.

Table 6. MPC574xR boot flash addresses (continued)

Search order	Block	Address
9	256KB Code Flash block 3	0x010C_0000

If a valid boot header is found, the BAF code determines which cores should be enabled by the BAF code and determines the start location of the user code. In the case of the MPC5743, the boot header should only enable core 0. The MPC5746R and the MPC5745R can have either or both of the cores enabled by the BAF.

3.1 Boot header

The boot header locations are read by the BAF to determine if the flash block is a valid boot location. The BAF checks for a value of 0x005A in the first half-word (16-bits) of the flash block. If the reset configuration half-word (RCHW) is valid, additional information is loaded by the BAF from the flash.

Table 7. Boot header structure

Address offset	Contents
0x00	Boot header configuration
0x04	Reserved
0x08	Reserved
0x10	Core 0 Reset vector
0x14	Core 1 Reset vector
0x18	Checker CPU0 Reset vector

The boot header field is used to not only determine whether the flash block is valid for booting, but also for selecting which core will be enabled following the BAF execution.. The configuration is shown in the following tables.

Table 8. Boot header configuration

Address offset 0x0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0				0x0				0x5				0xA			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0				0x0				0x0				Core 1	Core C	Core 0	0b0

Table 9. Boot header bit definitions

Bits	Field Name	Description
0:7	Reserved	These bits should be programmed to 0.
8:15	Boot Identifier	The Boot Identifier field serves two functions: • Indicates which block in flash memory contains the boot program • Indicates if the flash memory is programmed (BOOTID = 0x5A) or invalid
16:27	Reserved	These bits should be programmed to 0.

Table continues on the next page...

Table 9. Boot header bit definitions (continued)

Bits	Field Name	Description
28	CPU1	CPU1 is enabled (will start execution at CPU1 Reset Vector)[1]
29	CPUC	CPU_SC is enabled (will start execution at CPU_SC Reset Vector, must be the same as CPU0 Reset Vector)[2]
30	CPU0	CPU0 is enabled (will start execution at CPU0 Reset Vector)
31	Reserved	This bits should be programmed to 0.

1. For the MPC5743R, this bit must be programmed to a 0b0.
2. CPUC is the lock-step "checker" core. It can be disabled for a slight improvement in operating current.

For the MPC5743R, Core 1 should always be set to 0b0 (disabled). Therefore, for these devices, the recommended setting of the boot header configuration location is 0x005A_0006. This configuration enables Core 0 along with the Core 0 Checker core (CORE C).

After the BAF determines which cores to enable (following BAF execution), the Core reset vector is read for the start location for each core that is enabled for execution. The start location and initial run mode (DRUN) are loaded into the Mode Entry Module Core Control register (MC_ME_CCTL*n*[DRUN] and the core address (reset vector) register (MC_ME_CADDR*n*.

4 Green Hills example linker file

Below is an example of a typical linker file for the MPC5743R and MPC5744R for the Green Hills Multi-compiler.

```
/*---------- MODIFY THIS SECTION FOR THE ACTUAL DEVICE AND PROJECT -----------*/
DEFAULTS {
// Define Boot Header area Size
   BOOTFLASH_SIZE = 0x20 // The first 32 bytes of a flash block are used for the flash
header.
// Define Boot Header area Base Address
   BOOTFLASH_BASE_ADDR = 0x01000000

// Define Core 0 Flash Allocation with Large Flash Block
   FLASH_SIZE = 1536K - BOOTFLASH_SIZE
// Define Core 0 Flash Base Address
   FLASH_BASE_ADDR =  BOOTFLASH_BASE_ADDR + BOOTFLASH_SIZE

// Define Standby RAM Allocation
   C0_STBYRAM_SIZE =  32K
// Define Standby RAM Address
   C0_STBYRAM_BASE_ADDR = 0x40000000

// Define Core 0 SRAM Allocation for MPC5743R
   C0_SRAM_SIZE =  96K
// Define Processor 0 SRAM Base Address
   C0_SRAM_BASE_ADDR = 0x40008000

// Define Core 0 IMEM
   C0_IMEM_SIZE =  16K
// Define Processor 0 IMEM Base Address
   C0_IMEM_BASE_ADDR = 0x50000000

// Define Core 0 DMEM
   C0_DMEM_SIZE =  32K
// Define Processor 0 DMEM Base Address
   C0_DMEM_BASE_ADDR = 0x50800000

// Stack located at end of DMEM
```

```
// Define Stack Size
   STACK_SIZE = 4K

}

/*---------- DO NOT MODIFY ANYTHING BELOW THIS POINT -------*/
MEMORY {
    bootflash : org = BOOTFLASH_BASE_ADDR, len = BOOTFLASH_SIZE
    int_flash : org = FLASH_BASE_ADDR,     len = FLASH_SIZE
    int_stbyram  : org = C0_STBYRAM_BASE_ADDR,   len = C0_STBYRAM_SIZE
    int_sram  : org = C0_SRAM_BASE_ADDR,   len = C0_SRAM_SIZE
    int_imem  : org = C0_IMEM_BASE_ADDR,   len = C0_IMEM_SIZE
    int_dmem  : org = C0_DMEM_BASE_ADDR,   len = C0_DMEM_SIZE-STACK_SIZE
    stack_ram : org = (C0_DMEM_BASE_ADDR+C0_DMEM_SIZE-STACK_SIZE)  len = STACK_SIZE
}

SECTIONS
{
    .bh               : {} >  bootflash
    .init             : {} > int_flash
    .text             : {} > .              /* BookE Code   */
    .vletext          : {} > .              /* VLE Code
*/
    .fixaddr          : {} > .              /* Required for         */
    .fixtype          : {} > .              /* compatibility with   */
    .secinfo          : {} > .              /* GHS provided startup */
    .syscall          : {} > .              /* code                 */

    .rodata  : {*(.rdata) *(.rodata)} > .   /* Read Only Data */

    .ROM.data   ROM(.data)       : {} > .   /* Store Initialised RAM Variables */
    .ROM.sdata  ROM(.sdata)      : {} > .   /* temporarily in Flash            */
    .xptn_vectors ALIGN(0x100)   : {} > .  /* Exception Vector Table (IVPR) - align 256 byte
boundary */
    .isrvectbl    ALIGN(0x1000)   : {} > .   /* ISR Vector Table - must be 4K aligned */

//    .IVOR4_HWvectors ALIGN(0x1000): {} > .   /* IVOR4 HW Vector Table (IVPR) - align 4K
boundary */

    .stbyram          : {} > int_stbyram    /* Standby Ram Data               */
    .data             : {} > int_sram       /* Initialised Data               */
    .bss              : {} > .              /* Uninitialised Data             */
    .sdabase ALIGN (2): {} > .              /* Base location for SDA Area     */
    .sdata            : {} > .              /* Small Initialised Data (Area1)  */
    .sbss             : {} > .              /* Small Uninitialised Data (Area1)*/
    .sdata2           : {} > .              /* Small Initialised Constant Data */
    .sbss2            : {} > .              /* Small Uninitialised Data (Area2)*/

    .heap   ALIGN(16) PAD(1K)         : {} > int_sram   /* Heap Area  */
    .stack  ALIGN(4)  PAD(STACK_SIZE) : {} > stack_ram  /* Stack Area */

/*-------------------------------------------------------------*/
/*  Example of allocating section at absolute address         */
/*                                                            */
/*  .my_section 0x40001000 :{} > int_flash                    */
/*                                                            */
/*  Linker uses "0x40001000" address, rather than "int_flash" */
/*-------------------------------------------------------------*/

/*-------- LABELS USED IN CODE -------------------------------*/
/* Interrupt Handler Parameters */
__IVPR = ADDR(.xptn_vectors);

/* Label for IMEM and DMEM Init */
    __IMEM_START_CPU0 = C0_IMEM_BASE_ADDR;
    __IMEM_END_CPU0   = C0_IMEM_BASE_ADDR + C0_IMEM_SIZE;

    __DMEM_START_CPU0 = C0_DMEM_BASE_ADDR;
    __DMEM_END_CPU0   = C0_DMEM_BASE_ADDR + C0_DMEM_SIZE-STACK_SIZE;
```

Example memory usage

```
/* Labels for Copying Initialised Data from Flash to RAM */
    __DATA_SRAM_ADDR_CPU0  = ADDR(.data);
    __SDATA_SRAM_ADDR_CPU0 = ADDR(.sdata);

    __DATA_SIZE_CPU0   = SIZEOF(.data);
    __SDATA_SIZE_CPU0  = SIZEOF(.sdata);

    __DATA_ROM_ADDR_CPU0  = ADDR(.ROM.data);
    __SDATA_ROM_ADDR_CPU0 = ADDR(.ROM.sdata);

/* Labels for stack */
    __STACK_SIZE = STACK_SIZE;
    __STACK_START_CPU0 = C0_DMEM_BASE_ADDR+C0_DMEM_SIZE-STACK_SIZE;
    __STACK_END_CPU0   = C0_DMEM_BASE_ADDR+C0_DMEM_SIZE;

/* Lables for Standby RAM */
    __STBYRAM_ADDR  = MEMADDR(int_stbyram);
    __STBYRAM_SIZE   = MEMENDADDR(int_stbyram) - MEMADDR(int_stbyram);

/* These special symbols mark the bounds of RAM and ROM memory. */
/* They are used by the MULTI debugger.                         */

    __ghs_ramstart  = MEMADDR(int_sram);
    __ghs_ramend    = MEMENDADDR(int_sram);
    __ghs_romstart  = MEMADDR(int_flash);
    __ghs_romend    = MEMENDADDR(int_flash);

    __ghs_rambootcodestart = 0;           /* zero for ROM image */
    __ghs_rambootcodeend = 0;             /* zero for ROM image */
    __ghs_rombootcodestart = MEMADDR(int_flash);
    __ghs_rombootcodeend = MEMENDADDR(int_flash);

}
```

5 Example memory usage

The following table shows the memory usage of the Green Hills Software (GHS) example linker file.

Table 10. Example Memory Usage

Start Address	End Address	Memory Allocation	Section	Allocated Size [Bytes]	Description	Example usage
Flash						
0x0040_0000	0x0040_3FFF	—	—	16 KB	UTEST - for user configuration	Not used in this GHS[1] example code.
0x0040_4000	0x0040_7FFF	—	—	16 KB	Boot Assist flash (no user code)	Does not require allocation in the GHS project.
0x0080_0000	0x0083_FFFF	—	—	256 KB	EEPROM emulation area	Not used in this GHS example code.

Table continues on the next page...

Table 10. Example Memory Usage (continued)

Start Address	End Address	Memory Allocation	Section	Allocated Size [Bytes]	Description	Example usage
0x00FA_0000	0x00FF_FFFF	—	—	400 KB	Small & Medium Flash Blocks - no overlay	Not used in this GHS example code.
0x0100_0000	0x0100_001F	bootflash	.bh	20 Bytes	Boot header structure	BOOTFLASH_BASE_ADDR and BOOTFLASH_SIZE
0x0100_0020	0x01FF_FFFF	int_flash	.init, .text, .vletext, .fixaddr, .fixtype, .secinfo, .syscall, .rodata, .ROM.data, .ROM.sdata, .xptn_vectors, .isrvectbl		User program code area	FLASH_BASE_ADDR and FLASH_SIZE
0x08F9_C000	0x08FF_FFFF	—	—	6144 KB	Mirror Small & medium flash blocks	Not used in this GHS example code
0x0900_0000	0x09FF_FFFF	—	—	16384 KB	Mirror large flash blocks	Could be used read only data that allows overlay.
SRAM						
0x4000_0000	0x4000_7FFF	int_stbysam	.stbyram	32 KB	Standby SRAM	Not used in this GHS example code.
0x4000_8000	0x4001_FFFF [2]	int_sram	.data, .bss, .sdabase, .sdata, .sbss, .sdata2, .sbss2, .heap,	224 KB/ 160 KB/ 96KB[3]	Volatile memory for variables, data and compiler heap support	
Local Memory (SRAM)						
0x5000_0000	0x5000_3FFF	int_imem	—	16 KB	I-MEM - can hold frequently executed code	Not used in this GHS example code.
0x5080_0000	0x5080_7FFF	int_dmem	—	28 KB	D-MEM - Could be used for frequently accessed global variable	Not used in this GHS example code.
			.stack[4]	4 KB	4KB of D-MEM used for stack space	Used for stack storage.

1. Green Hills Software (compiler)
2. 0x4002_FFFF for the MPC5745R and 0x4003_FFFF for the MPC5746R.
3. 96KB for the MPC5743R, 160KB for the MPC5745R, and 224KB, for the MPC5746R.
4. The stack space is from 0x5080_7000 to 0x5080_7FFF

6 Revision history

Table 11. Revision history

Revision	Description	Date
0	Initial customer release	December 2015
1	corrected data flash size of the MPC5743 and added clarification for BAF and UTEST in MPC574xR family device differences and MPC574xR flash definition.	February 2016
	Corrected data flash used in the MPC5743R. (MPC574xR flash definition)	
	Corrected SRAM Actual size of 96 KB SRAM block (Allocated and Actual are the same). (MPC574xR SRAM definition)	
	Clarified allocated memory space versus actual (added address for the actual end addresses.	
	Corrected end addresses in the Example memory usage table for non-MPC5743R devices.	
	Fixed formatting in several tables, including fixing a footnote location.	
	Corrected comment on Core 0 DMEM in the example linker file.	
	Added revision history section	
	Changed Freescale references to NXP.	
2	Corrected flash block mapping (MPC574xR flash definition). Revision 6 of the MPC5746R Reference Manual included an incorrect flash memory map that was used in this application note. The Reference Manual is currently being corrected.	May 2016

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number AN5234
Revision 2, May 2016