NXP Semiconductors Application Notes

MPC5777C MCAL4.3 Dual Core Project Implementation

by: NXP Semiconductors

1. Introduction

This application note introduce the dual core project implementation in MCAL4.3.

Additionally, this sample application is for MPC5777C microcontroller with AUTOSAR MCAL4.3 version RTM1.0.0.

This application note provides an introduction about startup and link file modification to implement dual core project.

This document also provides the sample code in the software package accompanying this document.

The following table shows the abbreviations used throughout the document.

Contents

1.	Intro	oduction	1
2.	MC	AL4.3 Startup and Link file	2
		Dual core memory section	
		Dual core startup file	
3.		e 0 and core 1 reset vector configuration	
		npile and testing	
		Compiling GHS	
		Test result	
5.	Refe	erences	8



Term/Acronym	Definition		
BAM	Boot Assist Module		
IVPR	Interrupt vector prefix register		

Table 1. Acronym and definition

2. MCAL4.3 Startup and Link file

MCAL4.3 startup and link file is only for one core, so the startup and link file needs to be modified for two cores.

2.1. Dual core memory section

Flash_rsvd1 is used to store RCW, flash_memory is used by core 0. You should modify the link file which allocates the flash_memory 1 to store the core 1 startup code and fix the vector reset address as 0x00ED0000. You also need to configure this address in EB tresos.

Note: This linker comm	and file to be used with MPC5777C device only
MEMORY {	
/* 5777C – Flash 8	.0 MB - 0x00800000 - 0x00FFFFFF */
flash_rsvdl	: ORIGIN = 0x00800000, LENGTH = 0x20
/*flash_memory	: ORIGIN = 0x00800020, LENGTH = 0x7CFFE0*/
flash_memory	: ORIGIN = 0x00800020, LENGTH = 0x6CFFE0
flash_memory1	: ORIGIN = 0x00ED0000, LENGTH = 0x100000
flash_vec	: ORIGIN = 0x00FD0000, LENGTH = 0x010000
flash vec corel	: ORIGIN = 0x00FE0000, LENGTH = 0x010000
flash_rsvd2	: ORIGIN = ., LENGTH = 0

Figure 1. Memory for two core

The following figure shows the allocated stack for core 0, the address starts from 0x4003EBC0. The core1 stack address startx from 0x4003FBC0 and the length is 4k.

MPC5777C MCAL4.3 Dual Core Project Implementation, Rev. 0, 12/2020

```
/* 5777C - SRAM 512 KB : 0x40000000 - 0x4007FFFF */
ram_rsvd1 : ORIGIN = 0x20000000, LENGTH = 0
int_sram
                  : ORIGIN = 0x40000000, LENGTH = 0x3EB60
res ram
                   : ORIGIN = 0 \times 4003 \times 260, LENGTH = 0 \times 60
                    : ORIGIN = 0 \times 4003 \text{EBC0}, LENGTH = 0 \times 1000
iram stack
iram_stack_core1
                       : ORIGIN = 0x4003FBC0, LENGTH = 0x1000
int timers
                    : ORIGIN = 0x40040BC0, LENGTH = 0x0340
int results : ORIGIN = 0x40040F00, LENGTH = 0x0100
int sram no cacheable : ORIGIN = 0x40041000, LENGTH = 0x3F000
                     : ORIGIN = ., LENGTH = 0
ram rsvd2
```

Figure 2. Stack for core1

The following figure shows the allocated new section for core 1.

- .text1: section for core 1 startup code
- .isrvectbl_core1: section for core 1 interrupt vector table
- .isrvectbl_core_core1: section for core1 core vector table(IVOR)
- __IV_ADDR_core1: get the flash_vec_core1 address
- ____SP_INIT_core1: get iram_stack_core1 address

.text1	: > flash_memoryl
.isrvectbl .isrvectbl_core	ALIGN(0x10000) : > flash_vec : > .
.isrvectbl_core1 .isrvectbl_core_core1	<pre>ALIGN(0x10000) : > flash_vec_core1 : > .</pre>
IV_ADDR	<pre>= MEMADDR(flash_vec);</pre>
IV_ADDR_core1	<pre>= MEMADDR(flash_vec_corel);</pre>
SP_END_core1 = A	DDR(iram_stack_corel) + SIZEOF(iram_stack_corel); DDR(iram_stack_corel); ZEOF(iram_stack_corel);

Figure 3. Allocated new section for core 1

2.2. Dual core startup file

As shown in the following figure, in order to put the startup code into their own section, use the GHS compiler key words to set the core 1 startup code to .text1 section.

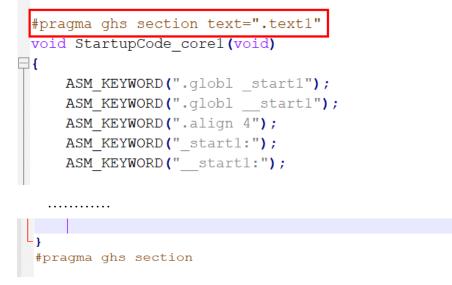


Figure 4. Startup code for core 1

The following figure shows the IVPR register that needs to be initialized to the address of the interrupt vector table and exception vector handler. These handler addresses are provided as configuration parameters in link setting.

```
/* Initialize IVPR register to address of Interrupt Vector Table */
ASM_KEYWORD (" e_lis r5, __IV_ADDR_corel@h ");
ASM_KEYWORD (" e_or2i r5,__IV_ADDR_core1@1 ");
ASM KEYWORD (" mtIVPR r5");
ASM KEYWORD (" se mr r0, r31 ");
ASM_KEYWORD(" e_lis r5, IVOR0_Handler_core1@h ");
ASM_KEYWORD(" e_or2i r5, IVOR0_Handler_corel@l ");
ASM_KEYWORD(" mtspr 400, r5
                                ");
ASM KEYWORD (" e lis r5, IVOR1 Handler core1@h ");
ASM KEYWORD (" e or2i r5, IVOR1 Handler core1@1 ");
ASM KEYWORD (" mtspr 401, r5
                               ");
ASM_KEYWORD(" e_lis r5, IVOR2_Handler_corel@h ");
ASM_KEYWORD(" e_or2i r5, IVOR2_Handler_core1@1 ");
ASM_KEYWORD(" mtspr 402, r5
                                ");
ASM KEYWORD (" e lis r5, IVOR3 Handler corel@h ");
ASM KEYWORD (" e or2i r5, IVOR3 Handler corel@1 ");
ASM_KEYWORD (" mtspr 403, r5
                                ");
ASM KEYWORD(" e lis r5, IVOR4 Handler corel@h ");
ASM_KEYWORD(" e_or2i r5, IVOR4_Handler_core1@1 ");
ASM_KEYWORD(" mtspr 404, r5
                                ");
```

Figure 5. Interrupt vector and exception handler initialization

MPC5777C MCAL4.3 Dual Core Project Implementation, Rev. 0, 12/2020

The machine check and exception for core 1 should be enabled as shown in the following figure.

```
/* define MSR mask to enable Machine Check and Exception / IRQ */
ASM_KEYWORD(" .equ MSR_Mask1, 0x00009000 ");
ASM_KEYWORD(" mfmsr r5 ");
ASM_KEYWORD(" e_lis r6, MSR_Mask1@h ");
ASM_KEYWORD(" e_or2i r6, MSR_Mask1@l ");
ASM_KEYWORD(" se_or r5, r6 ");
ASM_KEYWORD(" mtmsr r5 ");
ASM_KEYWORD(" se_isync");
```

Figure 6. Machine check and exception for core 1

Start-up code should initialize user stack pointer for core 1, as shown in the following figure.

Figure 7. Stack initialization for core 1

3. Core 0 and core 1 reset vector configuration

On the MPC577C device, the BAM occupies 16 KB of memory space, 0xFFFF_C000 to 0xFFFF_FFF. The actual code size of the BAM program is less than 4 KB and starts at 0xFFFF_F000, repeating itself after every 4 KB in the BAM address space. BAM program execution as it reset vector from address 0xFFFF_FFFC.

The BAM exits to user code at 0xFFFF_FF8. The last instruction executed by BAM is a BLR. The link register is pre-loaded with the user application start address. The value of the start address depends on the boot mode:

- Booting from internal or external flash memory: 32-bit word following a valid RCHW holds the start address value
- Serial boot is set according to the serial boot protocol

The following table shows the BAM address map.

Table 2	. BAM	memory	map
---------	-------	--------	-----

Address	Description		
0xFFFF_C000 – 0xFFFF_EFFF	BAM program mirrored		
0xFFFF_F000 – 0xFFFF_FFF	BAM program		
0xFFFF_FFC	MCU reset vector		
0xFFFF_FF8	BAM last executed instruction		

Compile and testing

In order to make the two cores have a fixed vector initial address which is consistent with the EB configuration, the core 0 reset vector address and core 1 reset vector address should be configured with 0xFFFF_FFFC(4294967292) and 0x00ED0000(15532032) in EB tresos.

 McuCore0ResetVectorRegister 		
Name 😸 McuCore0ResetVectorRegister		
Core0 reset vector (0 -> 4294967295)	▶ 4294967292	
Core 0 reset enable.	🖹 🗌 🖉 ▾ Core 0 VLE select. 🖹 🗌	/ -
Figure 8. Configure the re	eset vector for core0 in EB	
 McuCore1ResetVectorRegister 		
McuCore1ResetVectorRegister McuCore1ResetVectorRegister		
	☐ 15532032	

Figure 9. Configure the reset vector for core1 in EB

4. Compile and testing

4.1. Compiling GHS

The test hardware is the MPC5777C EVB. Compiler GHS version is compiler 2019.5.4(it is recommended to update to the latest version), EB tresos version is 25.1 and the MCAL package version is MPC5777C MCAL4.3_1.0.0.

Open the build folder as shown in the following figure.

^			
ame	Date modified	Туре	Size
bin	2020/5/14 10:04	File folder	
cmm	2020/1/21 14:57	File folder	
linkfiles	2020/3/23 11:07	File folder	
make	2020/3/23 11:07	File folder	
launch.bat	2020/3/23 11:08	Windows Batch File	3 KB
make.bat	2020/1/21 14:57	Windows Batch File	5 KB
makefile	2020/1/21 14:57	File	16 KB

Figure 10. Sample code build folder



Figure 11. Launch.bat file

TRESOS_DIR: EB tresos installation path

MAKE_DIR : make.exe file path

GHS_DIR: GHS compiler installation path

PLUGINS_DIR : The MCAL4.3 plugins installation path

TRESOS_WORKSPACE_DIR: EB project generated code path, which is

 $C: vds-mpc5777c-mcal4.3 \\ MPC5777C_MCAL4_3_RTM_1_0_0_Sample_Application \\ Tresos \\ Name and Name and$

Workspace\MPC5777C_4.3_sample_applications\output

To build the sample, execute the following command to run launch.bat: launch.bat

C:\Windows\System32\cmd.exe	_		×
C:\Work_Material\software\vds-mpc5777c-mcal4.3\vds-mpc5777c-mcal4.3\MPC5777C_MCAL4_3_RTM_1_0_0_Sample_Ap	plicat	ion\ec	lips
e\plugins\PlatformIntegration_TS_T2D41M10I0R0\build>launch.bat			

Figure 12. Build command

The object files and linker output file (PlatformIntegration.elf) will be generated in the /bin subdirectory.

Open the cmm folder and run the smp_demo.cmm script using **LAUTERBACH** debugger and debug the sample application code.

4.2. Test result

This sample application was tested on the MPC5777C EVB as showing in the following figure. Core 0 is running the MCAN transmission and core 1 is running the CAN transmission.

References



Figure 13. MPC5777C EVB board

PCAN-View	Transmit V	iew Trace	Window Help		PCAN-View	B PCAN-View File CAN Edit Transmit View Trace Windo ■	
CAN-ID 011h	Type		PCANUSE Pro FD III III III IIII IIIIIIII IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Error Generator Cycle Time 355852.6	Available ECAN hardware Available ECAN Vas Por FD Device FFFFFFFR, Channel 1 Available PCAN-USB Pro FD: Device FFFFFFFR, Channel 2	CAN-ID Type Len Data 031h 8 31 31 31 31 31 31 31 31 31 31 31 31 31 3	PCAN-USB Pro FD: Device FFFFFFFh, Channel 2
02 700h		16	12 13 14 15 16 17 18 19 00 00 00 00 00 00 00 00 00 00 00 00 00	150.3		CAN-ID Type Len Data	
	Type	Length 16	Data Cycle Ti 12 13 14 15 16 17 18 19 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	Count Trigge 3900 Time		600h 8 12 13 14 15 16 13 18 19	Eleck Frequencyi - <u>Nominal Bit rates</u> 40 MHz -> S00 kBit/s -> Filter settings
Connected to h	ardware PCAN-	USB Pro FD,	Channel 2 🍕 Bit rate: 500 kBit/s / 2 MBit	t/s Status: OK	(e) Standard From: (000) (Her) To: 77F (Her) Listen-only mode OK Cancel (Her) Herp	Connected to hardware PCAN-USB Pro FD, Channel	Standard From: 000 (Hex) To: 7FF (Hex) Extended Listen-only mode OK Cancel Ø Help

Figure 14. MCAN sample transmission in core0

5. References

- MPC5777C Reference Manual
- MPC5777C EVB User Guide

MPC5777C MCAL4.3 Dual Core Project Implementation, Rev. 0, 12/2020

Figure 15. CAN transmission in core1

How to Reach Us:

Home Page: nxp.com

Web Support: nxp.com/support Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:https://www.nylicense.nylicense.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. Arm, AMBA, Arm Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. Arm7, Arm9, Arm11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, Mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2020 NXP B.V.



Document Number: AN13063 Rev. 0 12/2020

