TMS470R1x Memory Mapping

Ralf Wagner *Advanced Embedded Control*

ABSTRACT

This document describes how to run a program in RAM, load another application to the RAM base address, and map the whole RAM to address zero without using a software interrupt in the boot memory block. The mapping is necessary, for example, if the RAM application needs to use specific interrupts. The interrupt vector table for the RAM application has then to be mapped to address zero.

Contents

List of Figures

1 Introduction

Sometimes mapping to RAM is necessary. For example, if the RAM application needs to use specific interrupts, the interrupt vector table for the RAM application has to be mapped to address zero.

2 Method

We begin this process while the RAM main application (RAM1 application) is still running. The new application (RAM2 application) or vector table could be a part of the RAM1 application (for example, a special interrupt vector table) or it could be an independent program that is loaded with the aid of the RAM1 application into a specific memory area. The new application has to be copied or linked to the beginning of the RAM block area.

At the end of the RAM has to be an area for a module named trampoline. This module is written in assembler. It switches the RAM area to zero and jumps to the new application.

Note: Execute this step in privileged mode.

The memory map is set up as shown in Figure 1.

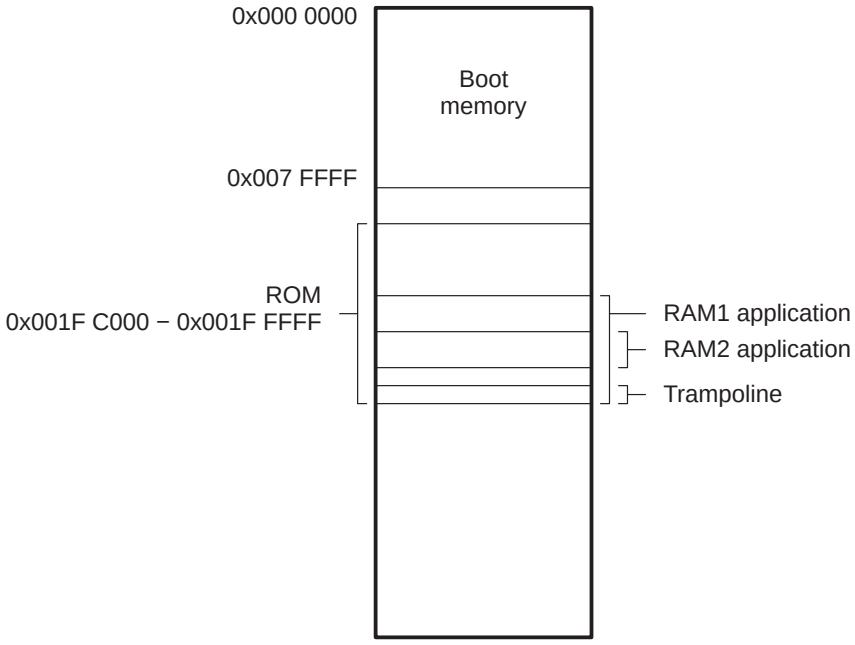

Figure 1. Example of Memory Map Setup

Copy the new application to the beginning of the RAM block. The first RAM address is address zero after remapping. RAM1 application has to handle that with a memcopy or load application from an external source.

Disable all interrupts and map the flash to next address behind the RAM block. No hole can be present between the RAM and the flash.

The trampoline is responsible for the memory mapping. It maps the flash to the new address area and maps the RAM down to zero. The mapping of the block, of which the program executes is very critical. As soon as one of the MFBAHR registers are rewritten, the RAM is mapped to another address and only the two instructions in the pipeline can be executed. These instructions must set up the RAM completely and jump to the defined address.

If there is no legal memory where the program counter points to (next fetch address), the fetch of the next instruction generates a prefetch abort. Therefore, there must be a valid program code at the next fetch addresses. This code will not be executed. That is the reason for mapping the flash bank directly to the end of the RAM behind the last RAM instructions and for placing the trampoline to the absolute end of the RAM. See Figure 2.

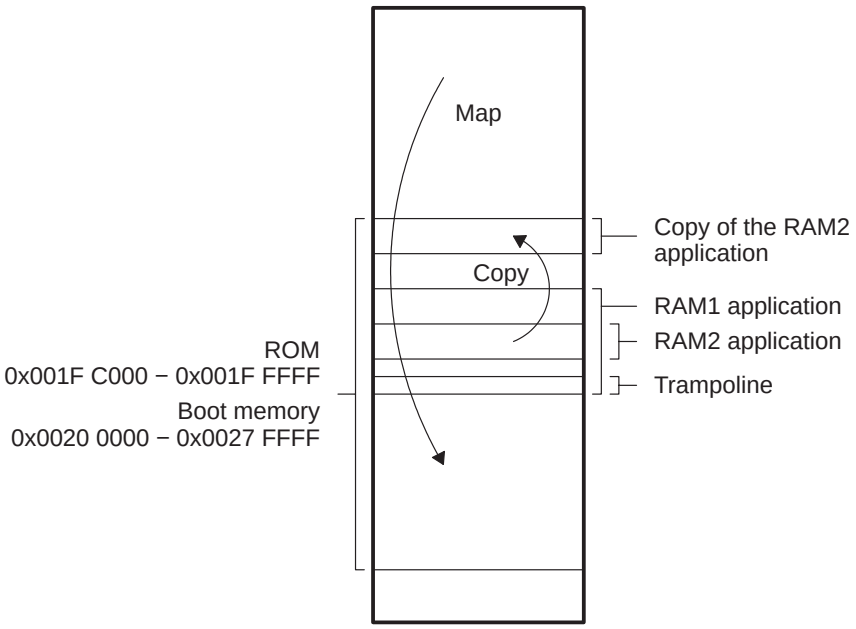

Figure 2. Remapping the Memory

The program code in section 2.1 is an example of the remapping with the trampoline. It is defined as its own section to map it to the absolute end of the RAM. For assembling, the registers R5, R6, and R9 were reserved for the trampoline.

First, the boot memory with the chip select line zero is mapped to address 0x0020 0000. For mapping the RAM to address zero, it is necessary to modify both MFBAxR registers. But if the first register is modified, the RAM is mapped to another address and only the next two instructions in the pipeline are valid. These two instructions must modify the other MFBAxR register and then point to a valid address for the new (target) application code to be mapped to. This address is most likely 0x00000000.

2.1 Code Snippet Setting Up Trampoline

```
        .sect   ".lastRAMblock"   ; 9 instructions, defines -> 36 bytes
        .global _trampoline

MFBAHR0       .word  0xFFFFFE00
_trampoline
       ; map the boot memory from zero behind the end of the RAM block
       LDR R6, MFBAHR0
       MOV    R5, #0x20     ; address 0x00200000
       STR    R5, [R6]      ; store in MFBALR0

       ; switch RAM to zero
       MOV    R5, #0x0            ; map RAM to 0x00000000
       MOV R9, #0x50       ; 16K RAM at address 0x00000000
       STR R9, [R6,#0x14]  ; write 0x0050 to MFBALR2
       STR    R5, [R6,#0x10]      ; write zero to MFBAHR2
       ; jump to address zero
       MOV    PC, #0x0
```

After the trampoline code is executed, the RAM is successfully switched to address and the RAM2 application (copied to the beginning of the RAM) will be executed. Figure 3 shows the results.

NOTE: Reconfigure the stack pointers because the stack area must also be moved.

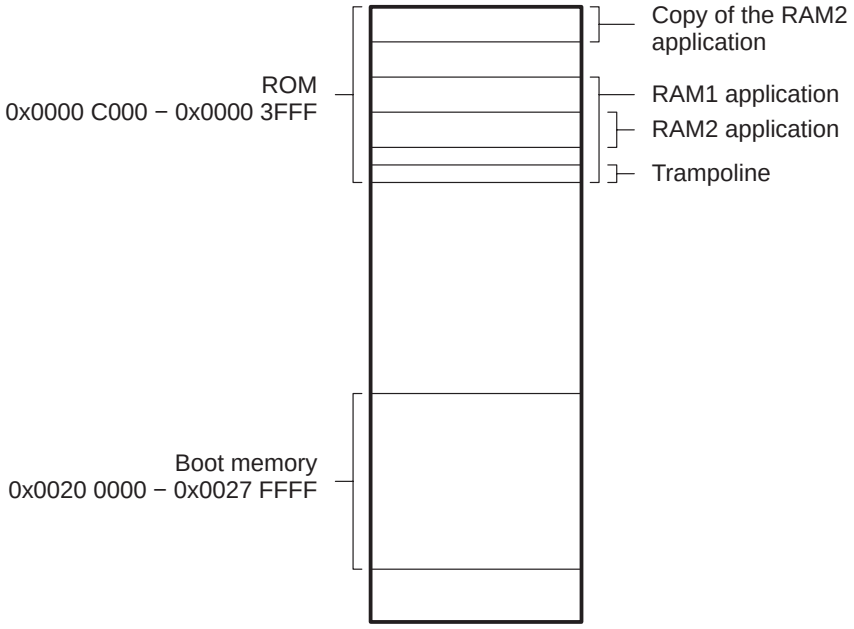

Figure 3. Remapped Memory

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265