Card Labs

Project: RMON Remote Monitor

Title: Rabbit 2000 Memory Map

Doc Number: RMON-SW-100
Revision: 1.02

Classification: Released to Public

DOCUMENT CLASSIFICATION

For more information, please contact:

Card Labs
132 Harvest Park Way NE,
Calgary, Alberta, Canada T3K 4K7

Tel: (403) 567-9328
Fax: (403) 567-9329

info@cardlabs.com
http://www.cardlabs.com

REVISION HISTORY

Rev.	Date	Description	Author
1.00	15 Mar 2002	Original	Scot Kornak
1.01	25 Feb 2003	Released for public distribution.	Scot Kornak
1.02	11 Mar 2003	DATASEG, STACKSEG & XPC physical addresses fixed (thanks to Bill Auerbach). Rabbit recommends 32k reserved for boot loader at 0x80000.	Scot Kornak

Print Date: 11 March 2003

CONTENTS

1 Introduction 3

2 Definitions 3

3 Logical Memory Map 4

3.1 Interrupt Vector Tables 5

4 Physical Memory Map 6

4.1 Physical Memory Map 6

5 Contributions 7

1 Introduction

The RMON Remote Monitor is a command line monitor for the Rabbit 2000 processor. It provides a means to monitor and control the target hardware, examine memory and registers, and upload new firmware. It uses the on-board Flash EPROM and SRAM on Rabbit modules and external serial dataflash. This document is created to determine how memory is used in a Rabbit program to aid in the design of the remote firmware upload utility.

This document describes the default internal memory allocation of the Rabbit Dynamic C compiler, version 7.21.

This document is released to the general public to benefit people learning about the Rabbit processor. Please send your comments and suggestions regarding this document to info@cardlabs.com.

If anyone would like to contribute to this document, please e-mail your written sections. A section on the user block and system ID block in Flash would be very useful. All material that is accepted will be credited to the submitting author. Please do not send information that you do not hold the rights to (e.g. material just cut from other manuals).

2 Definitions

Physical Memory - the memory devices external to the Rabbit processor connected to the address and data bus. The physical address space is 1Mbyte.

Physical Address (PA) - the 20 bit address sent to the external memory devices. The address lines external to Rabbit processor carry the physical address. The logical address is converted by the MMU (Memory Management Unit) to the physical address.

Logical Memory - the 64k memory space that is directly accessed by the Rabbit processor core. Logical memory is the 64k portion of physical memory that is visible to the processor core at any one time. The MMU creates a number of windows (which add up to a total of 64k in size) that allow the processor core to peer into physical memory. Some of these windows can be moved around so that the core can see all of physical memory, a piece at a time.

Logical Address (LA) - the 16 bit address of the Rabbit processor core to the MMU.

Root Memory Segment - the memory starting at LA 0x0000 that holds root code. Root code is always visible to the processor. It is used for BIOS, interrupt routines, and constants. Other code needs to be visible only when executed and can be located in extended memory. When the Rabbit comes out of reset, code execution starts at 0x0000. The root memory segment is always mapped to PA 0x0000. i.e. it is the only portion of logical memory that has the same physical memory in typical use.

Data Segment - the data segment contains C variables and it is mapped to RAM. The data segment also holds the interrupt vector tables.

Stack Segment - the stack size is usually 4k, extending from LA 0xD000 to 0xDFFF. It is always mapped to RAM. The stack starts at the highest address and grows downwards.

Extended Memory Segment - a 8k memory window from LA 0xE000 to 0xFFFF used to execute code not in root memory (i.e. extended code). It is also used as a window to view or modify a section of physical memory.

Root Memory - all logical addresses below 0xE000 are root memory (including the stack and data segments) because they can be directly accessed by the processor core.

Extended Memory - any physical memory that is accessed through the extended memory segment is extended memory. This is all of physical memory that is not accessed through the root, data, or stack memory segments.

3 Logical Memory Map

This memory map shows the default memory organization of a Dynamic C 7.21 program compiled to Flash EPROM.

Logical Address (0 to 64k) (RabbitBIOS.C addr names)		**Comments**
0x0000 (0k) (ROOTCODEORG) 0x5FFF (24k)	**Root Memory** RabbitBios.C and BIOSLIB libraries code All Root Code and Constants	Always maps to PA 0x00000. Code from RabbitBios.C starts at LA 0x0000.
0x6000 (24k) (DATAORG)	**Data Segment** Contains all Root Data (i.e. variables) Watch Code	Typically mapped to PA 0xB2000 in XMEM. The start address of the Data Segment is determined by the lower nybble of the `SEGSIZE` register. e.g.) `SEGSIZE` = 0xX6 gives a start LA of 0x6000. The Data segment maps to the physical address PA = `DATASEG` * 4096 + start LA
0xCE00 (INTVEC_BASE)	Internal Interrupt Vector Table 0x00 - System Periodic Int. 0x10 - ??? 0x20 - RST 10, used for? 0x30 - RST 18, used for? 0x40 - RST 20, used for? 0x50 - RST 28, debug breakpoint 0x70 - RST 38, "Development Vector" 0x80 - Slave Port 0xA0 - Timer A 0xB0 - Timer B 0xC0 - Serial Port A 0xD0 - Serial Port B 0xE0 - Serial Port C 0xF0 - Serial Port D	Located at LA (STACKORG - 0x200) The LA start address of the Internal Interrupt Vector Table is determined by the `IIR` register which sets the upper byte of the LA. The interrupt number provides the lower byte of the address as shown to the left.
0xCF00 (XINTVEC_BASE) 0xCFFF (52k)	Ext. Interrupt Vector Table 0x00 - INT0 0x10 - INT1 0x20 - 0xFF unused in the Rabbit 2000.	Located at LA (STACKORG - 0x100) The LA start address of the External Interrupt Vector Table is determined by the `EIR` register which sets the upper byte of the LA. The end address of the Data segment is determined by the start address of the Stack segment.

0xD000 (52k) (STACKORG)	**Stack Segment** Stack Allocated with xalloc()	Typically mapped to PA 0xA8000 in XMEM. The start address of the Stack Segment is determined by the upper nybble of the `SEGSIZE` register. e.g.) `SEGSIZE` = 0xDX gives a start LA of 0xD000. The Stack segment maps to the physical address PA = `STACKSEG` * 4096 + start LA
	… ^^^^ …stack grows downwards	
0xDFFF (56k)	Stack top	Always ends at 0xDFFF.
0xE000 (56k) (XMEMORYORG)	**Extended Memory Segment** Used to execute XMEM Code and access XMEM Data	Always starts at 0xE000. The Extended Memory Segment maps to either XMEM code or data. The physical address that is set with the `XPC` register. PA = `XPC` register * 4096 + 0xE000.
0xFFFF (64k)		

Figure 1. Rabbit Logical Memory Map

3.1 Interrupt Vector Tables

The Internal and External Interrupt Vector Tables are not just a list of 16 bit vector addresses like the Z80. Each 16 byte entry has code which is executed when the interrupt occurs. Small interrupt routines can fit entirely in the table. The table entry contains a jump to the interrupt routine if more code is required.

4 Physical Memory Map

A Rabbit system usually has both Flash EEPROM and SRAM with the Flash mapped to physical address 0x00000 and the SRAM at a higher address (usually on a 256k boundary). The four `MBxCR` registers make it easy to change this mapping during execution to allow one to run root code out of SRAM and other useful things.

4.1 Physical Memory Map

This memory map shows the typical memory organization with 512k Flash/ROM at 0x00000, 512k SRAM at 0x80000, and the default memory organization of a Dynamic C 7.21 program compiled to Flash EPROM.

Physical Addr (0 to 1024k)		**Comments**
0x00000 (0k)	**Flash EEPROM or ROM**	
	BIOS Code	Code from BIOS.LIB starts at 0x0000. Directly mapped to logical memory space.
	Root Code and Constants	Must be located between 0x00000 and 0x0DFFF. Directly mapped to logical memory space.
	XMEM Code and Constants	The `XPC` register determines where in physical memory the Extended Memory Segment maps to.
	System ID block	
0x7FFFF (512k)		
0x80000 (512k)	**SRAM**	
(the size of the boot loader may vary with the version of Dynamic C) 0x807FFF	Reserved for COLDLOAD.BIN or PILOT.BIN. Dynamic C places a boot loader program in RAM here during the download process to flash.	Although Rabbit says to reserve 32k (see comments in 7.32 STACK.LIB), only 0x8011E286 bytes were used in the DC 7.21 loading process.
0x88000 0xA7FFF	Unused Extended RAM	XMEM allocated using the `xalloc()` function goes here
0xA8000 (672k) (Dynamic C will move this area depending on the amount of memory in the system) 0xA8FFF	Stack Segment Stack Bottom … … ^^^^ …stack grows downwards … Stack top	The `STACKSEG` register determines the PA of the Stack Segment. (`STACKSEG`=0xA8) The `SP` register, the system stack pointer, is an offset from the start of the Stack Segment. The stack starts at a high address and grows downwards.

0xA9000 (676k) 0xB1FFF	Unused	
0xB2000 (712k) (Dynamic C will move this area depending on the amount of memory in the system) 0xB8FFF	<u>Data Segment</u>	The DATASEG register determines the PA of the Data Segment. (DATASEG=0xB2)
0xB9000 (740k) 0xFFFFF (1024k)	Unused	

Figure 2. Rabbit Physical Memory Map

5 Contributions

Thanks to Bill Auerbach of Softools for correcting the formula for the DATASEG, STACKSEG, and XPC physical address start formulas.