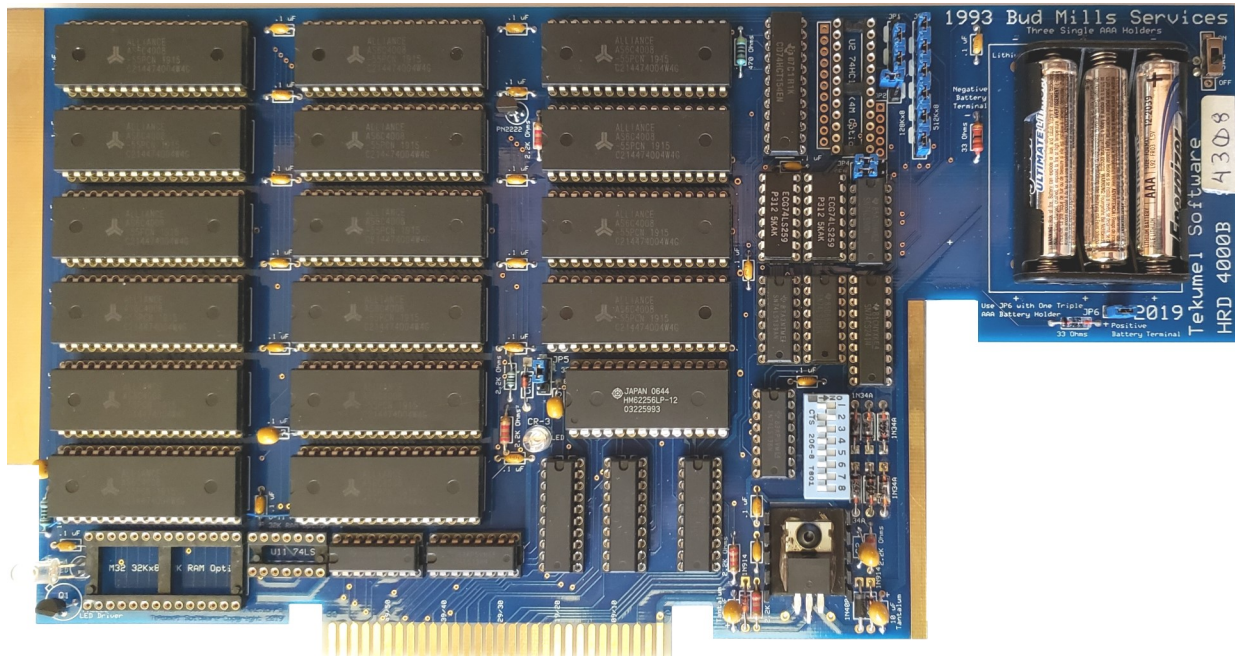


# Software Developer's Guide to the HRD4000B



Written by DCWarren  
Distributed by Tekumel Software  
Version 1.0  
Aug, 2020

Diligent research has been done to validate the accuracy of information in this guide. The author and Tekumel Software do not assume liability for any errors or inaccuracies that may still be contained herein. We do invite constructive comments for corrections or improvements from the TI-99 user community.

Version 1.0 and all subsequent revisions of this guide may be reproduced solely for non-commercial usage. Selling or otherwise charging for the reproductions and/or distribution of this package, in part or in totality, and regardless of distribution method, is prohibited. This guide and all associated files and documentation are not salable items.

T. Tesch, J. Fetzner, and D. Warren reserve the right to make changes to this guide at any time to assure accuracy and add content in order to provide the best possible product for the community.

Horizon Ramdisk wiki and source:

<https://github.com/horizonramdisk/Horizon-Ramdisk-ti994a/wiki>

## Table of Contents

1.0 Introduction.....	5
2.0 Short History of the Horizon HRD.....	5
3.0 Definition of Terms .....	6
3.1 Memory .....	6
3.2 Bank vs. Page vs RACKs vs Layers.....	7
3.3 RAMBO .....	7
3.4 Phoenix .....	7
4.0 DSR Implementation.....	8
5.0 CRU Implementation .....	8
5.1 CRU Quick Facts .....	8
5.2 CRU Bits 1–13 Memory Access .....	9
5.3 CRU Bit 14—DSR Memory Page Selection .....	9
5.4 CRU Bit 15—RAMBO Selection .....	9
6.0 Operational Modes-Physical Configurations .....	9
7.0 OpMode 0 – Standard Configuration .....	12
7.1 CRU Assignments – CRU bits 1-13.....	12
7.2 Conditions for Determining Number of RACKs.....	12
7.2.1 First Condition – Board not fully populated.....	12
7.2.2 Second Condition – Board fully populated with 128Kx8 SRAMs .....	13
7.2.3 Third Condition – Board fully populated with 512Kx8 SRAMs.....	13
7.2.4 Option to combine the Second and Third Conditions for simplification.....	13
7.2.5 Example.....	13
7.3 CRU Assignments – CRU bit 14 .....	14
8.0 OpMode 1 – Dual RAMDisk Configuration .....	14
8.1 Physical Partitioning of Memory in OpMode 1 .....	18
8.1.1 Conditions for Determining the Number of RACKs.....	18
8.1.2 Example.....	18
8.2 CRU Assignments – CRU bit 14 .....	20

9.0 OpMode 2 – Phoenix Configuration .....	20
9.1 Physical Partitioning of Memory in OpMode 2 .....	21
9.2 Some Observations of OpMode2.....	21
9.3 Epilogue for OpMode 2 .....	21
10.0 Hey, isn't there a fourth mode you're not talking about? .....	22
11.0 RAMBO .....	22
11.1 CRU Assignments .....	23
11.2 Some Observations and Considerations .....	23
APPENDIX A – CRU Address DIP Switch.....	25
APPENDIX B – RAMBO Operation with ROS 8.xx .....	27
APPENDIX C – RACK to RAMBO Cart Space Address Calculation .....	32
APPENDIX D – OpMode Correlation to DSR Memory and SRAM .....	34

Figure 1 – HRD4000B configured for 512Kx8 SRAMs.....	10
Figure 2 – OpMode 0 – Standard Configuration.....	10
Figure 3 – OpMode 1 – Dual RAMDisk Configuration .....	11
Figure 4 – OpMode 2 –Phoenix Configuration .....	11
Figure 5 - DSR Memory Mapping of M32 for OpModes 0 and 1 .....	15
Figure 6 – OpModes 0/1 CRU assignments for HRD4000B with 128Kx8 SRAMs .....	16
Figure 7 – OpModes 0/1 CRU assignments for HRD4000B with 512Kx8 SRAMs .....	17
Figure 8 - OpMode 1 Bank D CRU Bit NOT USED .....	20
Figure 9 - Phoenix CRU Address Line 11 .....	22
Figure 10 - RAMBO RACK Correlation to Cartridge Space.....	24
Figure 11 - HRD4000B CRU Base Address Assignments.....	26
Figure 12 - OpMode Correlation to DSR Memory and SRAM .....	35

## 1.0 Introduction

The Horizon RAMDisk (HRD) series cards for the TI-99/4A have both been a favorite of users and long lived in terms of aftermarket production and capability upgrades. The latest iteration of the HRD series is the HRD4000B and, like the HRD4000, hosts seventeen battery backed Static RAMs (SRAMs), the Phoenix modification, the RAM Block Operator (RAMBO) modification and the optional 32k memory expansion as integral circuitry on a single board. Unlike the HRD4000 and its predecessors, it provides access to two swappable (6k/6k) SRAM Device Service Routine (DSR) memory pages in each card mode for DSR software routines.

The purpose of this document is to provide you, the software developer, with knowledge of how the HRD4000B functions/behaves from a software perspective; to understand and utilize all of its capability to create upgrades or new functionalities. After all, the HRD4000B at its most basic level is a card containing battery-backed SRAM under control of a CRU interface with a battery-backed programmable DSR! The card's original and enduring purpose is to function as a solid-state floppy disk drive. However, it is quite feasible to change the purpose of the card entirely by simply re-writing the DSR.

Even though this document is focused on software control of the HRD, it will be necessary to present at least a top level explanation of some of the circuitry and architecture, especially in reference to Phoenix and RAMBO operations. Those explanations will be sprinkled throughout the document as needed. It is also assumed that you, as the software developer and devoted hobbyist, have knowledge of the TI-99/4A standards and assembly language. But, before diving into the details of the card, a short history is in order as well as some definitions of terms for our discussions.

## 2.0 Short History of the Horizon HRD

There are many existing references to the Horizon HRD series of cards but one concise history reference comes from the Swedish user group, "PROGRMBITEN" 04/1992:

### "HISTORY

Horizon began selling in Feb 1986. The original version had 64 kbit (8 kbytes) static RAM chips which must be mounted in two layers on top of each other to get the maximum size of 192 kbytes (24 chips).

Elektronik-Service in Germany started to sell Horizon boards with 392 kbytes during 1987. They used 32 kbytes chips and a small daughter board to handle the coding. One known serial number is 1216. See photo in PB 99-4.

HRD+ came in Feb 1988 with 32 kbytes chips which gave the possibility to Horizon with 384 kbytes (12 chips without any need to stack RAM chips on top of each other. Highest serial number 1900.

Horizon 2000 came in Sept 1988 and it does not need to stack the control chips (U2). Serial number 2000-2200.

Horizon 3000 came in Feb 1989 and it can be used with 32, 128 or 512 kbytes chips. Only 32 kbytes chips <were> used in the beginning for economical reasons. 128 kbytes RAM chips <were> not used until Apr 1990 when Horizon was made with 1536 kbytes (12 chips). Change<s> #0 - #2 <were> also included. Serial number 3400- .

Horizon 3000B has also change #3 included. Serial number 3400- .

Horizon 4000 came in Oct 1992 with RAMBO included as standard. It now uses 512 kbytes static RAM chips so you can have up to 9 Mbytes."

Written by Jan Alexandersson

Jan references four "changes" made to the HRD as it evolved; changes #0 - #3(rev 1-91). There is also a change #4 that added a diode. These were generally improvements to memory stability or for card reset/recovery. The details of these changes are left to the reader to research if interested; suffice it to say that they are accounted for, along with other stability improvements, on the HRD4000B.

### 3.0 Definition of Terms

Before continuing with the HRD4000B description, it is important to define some terms that will be used for the remainder of this document. These terms are defined within the framework of the HRD as a RAMDisk, its original purpose. This provides the most commonality across documentation even though the application(s) you may have in mind are not RAMDisk applications. In spite of trying, these terms may still not be totally consistent with other documentation for the Horizon HRDs so the reader should be aware when referencing other source material.

#### 3.1 Memory

There are three, functionally distinct blocks of memory on the HRD4000B. They are; memory for the RAMDisk function, memory for the DSR function and an optional 32K expansion memory. To avoid confusion throughout this document, the RAMDisk SRAM memory will be referred to as "SRAM." DSR SRAM memory will be referred to as "DSR memory." The optional 32K SRAM memory will be referred to as "expansion memory."

## 3.2 Bank vs. Page vs RACKs vs Layers

**Banks, pages, RACKs and layers** are terms used to refer to different aspects of the SRAMs.

A **bank** is a physical SRAM memory chip. It is possible to have up to 32 banks (memory chips) on the HRD4000B (M0 to M31) by stacking a second set of 16 chips on top of the first, board-level-layer set of 16 chips (see definition of layers below.) The memory chips can be 128Kx8 or 512Kx8 SRAMs.

This leads to the concept of pages. A **page** is a 2K memory space *within a bank and on a 2K address boundary*. So, a 128Kx8 memory chip has 64 pages (128K/2K). Likewise, a 512Kx8 chip has 256 pages.

RACKs come from the concept of RAMtraCKs. **RACKs** are simply a 0-to-n designation of all the 2K pages on the board within all of the SRAM banks, in a linear contiguous order. Example: for a card with sixteen 512Kx8 SRAMs; 16 banks x 256 pages (per bank) = 4096 RACKs; 0 to 4095. All RACKs present in the upper 2K (>5800 to >5FFF) addressing space of the 8K DSR space and are selected by the CRU interface.

**Layers** refer to physical layers of SRAM on the HRD board. SRAMs can be “stacked” on top of each other to add additional memory to the HRD. The 1<sup>st</sup> layer (layer 1) consists of SRAMs on the board level of the HRD. The 2<sup>nd</sup> layer (layer 2) consists of SRAMs stacked on top of the first layer of SRAMs. The HRD4000B supports two layers but, *its most common configuration is one layer of 16 512Kx8 SRAMs for a total of 8Mbytes*.

## 3.3 RAMBO

RAMBO is short for RAM Block Operator. It's functionality that allows 8K blocks of the SRAMs to be paged into the TI-99/4A console cartridge space, 0x6000 to 0x7FFF, for use as program or data space. This was once a separate add-on board that was later incorporated onto the HRD main board.

## 3.4 Phoenix

The Phoenix functionality allows the HRD4000B to be logically divided into two RAMDisk cards, both addressed with just a single CRU base address. It was originally designed for use with the Geneve 9640 with one RAMDisk designated as the BOOT drive and the other as a normal RAMDisk. This was also a separate modification that is now incorporated onto the HRD card.

NOTE: Phoenix, as stated, is functionality originally designed for use for the Geneve 9640. HOWEVER, two of the CRU address DIP switches are designated as “Phoenix CRU addresses.” Mixing references of Phoenix as a function and Phoenix as CRU addresses can lead to some confusion. Those DIP switches can



be/are used for some non-Phoenix functionality so, in that case, they will be referred to as “Phoenix DIP switches” to differentiate them from Phoenix functionality (ref Appenix A).

### 4.0 DSR Implementation

Device Service Routine memory is implemented on the production HRD4000B with a 32Kx8 SRAM. It presents in DSR memory space, 0x4000 to 0x57FF, when the HRD is enabled. Two 6K DSR memory pages are available on the card for each of its possible modes of operation. More information can be found in each of the Operational Mode and RAMBO sections that follow.

**NOTE:** It is possible for the early prototype HRD4000B cards to have only an 8Kx8 DSR memory chip! This is an option on cards 4300 to 4349 selected with an additional jumper (JP5) that is not on the production boards (4350-> ).

Something of note, it is quite acceptable to reserve RACKs to contain additional code for a DSR or other applications that you may have in mind. This is currently done in the RAMDisk Operating Software (ROS) that is provided for the HRD. So the possibilities are to use one or two DSR pages AND reserve RACKs for additional program space!

### 5.0 CRU Implementation

The Horizon HRD is controlled by the CRU interface. *<There are no memory mapped ports on the HRD>* Some quick facts to get us started:

#### 5.1 CRU Quick Facts

- All Horizon HRDs have a 16 bit CRU space even though all the bits may not be used. The exception is the first HRD which used 8 bits.
- All 16 CRU bits are utilized on the HRD4000B!
- All CRU bits are LATCHED!! Once set, they stay set until you change them or there is a system hard reset. This implies that they need to be initialized to a known state at some point, usually at power up or with a power up routine.
- The CRU bits CANNOT be read back. If needed, the software will have to keep track of which bits are set or reset.
- Bit 0 will ALWAYS be used to turn the card on/off.
- Bit 15 has more or less become the DE FACTO bit for RAMBO.
- Bit 14 is reserved by the HRD4000B to page DSR memory. It was unused by the HRD4000.
- This leaves Bits 1-13 to control memory; the pages, banks and SRAM layers.

## 5.2 CRU Bits 1–13 Memory Access

Access by software to all of available RACKs on any Horizon HRD4000B is done with the CRU bits 1-13. How many of those bits are needed depends on how much and what type of memory is installed. Figures 6 and 7 detail the CRU assignments for the HRD4000B and you can see that the lower set of CRU bits, starting at 1, control access to pages within a bank, a following set of four CRU bits control which bank on a layer is accessed and a final bit controls which layer of SRAM (1 or 2) is selected.

Sounds complicated? Not from an application perspective! Most applications really don't need to know how many pages are in a bank or how many banks there are or how many layers of SRAM exist. Your application ONLY needs to know how many RACKs there are and their CRU address space(s)! (*Unless for some reason you really do need to know those details such as a software diagnostic routine to identify bad SRAM*). So, your application will first and foremost need to determine the number of RACKs that are on a particular card and make that available to the rest of your routines. The logic for how that is achieved will vary depending on the operational mode of the card and will be examined in each of operational mode sections that follow. After that it's just a matter of selecting the RACKs that you want for your application.

## 5.3 CRU Bit 14—DSR Memory Page Selection

The CRU bit 14 toggles between two 6K DSR memory pages. The default for this bit is zero upon powerup.

## 5.4 CRU Bit 15—RAMBO Selection

The CRU bit 15 enables RAMBO mode (see section 11 RAMBO). The default for this bit is zero (not enabled) upon powerup.

## 6.0 Operational Modes-Physical Configurations

The HRD4000B is versatile and can be configured to perform in different ways depending on your requirements. This is accomplished by setting jumpers on the HRD circuit card. Once the HRD4000B jumpers JP1 and JP3 are configured for the type of SRAM installed on the card (figure 1), JP2 and JP4 can next be configured to provide one of three distinct *operational modes* for the HRD4000B. Any software developed will need to “know” how to operate with at least one of these modes and, you may need to guide the user on how to set these modes to match your application requirements.

- Operational Mode 0 (OpMode 0) is the standard, single card RAMDisk configuration (figure 2). *This is the predominant configuration for an HRD and recommended for most applications and use!*
- OpMode 1 splits the card logically into two RAMDisks, each with its own CRU base address (figure 3).

- OpMode 2 is the Phoenix configuration which splits the card into two logical RAMDisks both accessed through a single Phoenix CRU base address (figure 4).

Each of these OpModes are explained in separate sections.

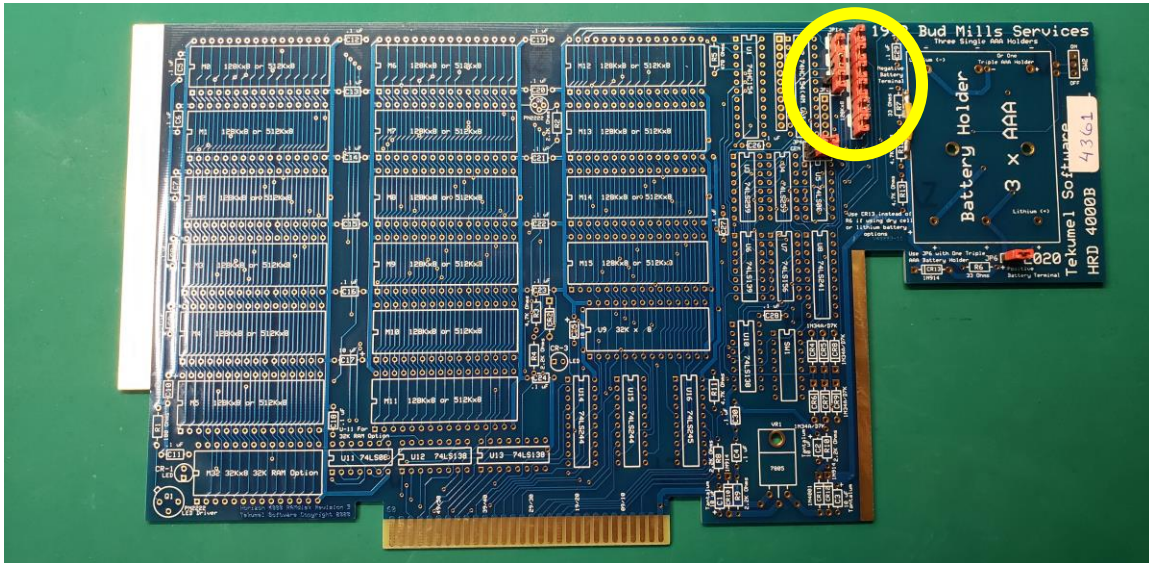


Figure 1 – HRD4000B configured for 512Kx8 SRAMs

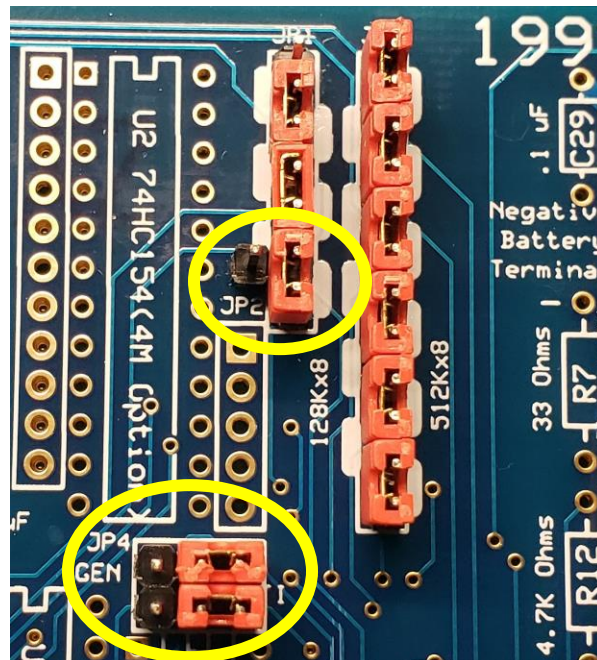


Figure 2 – OpMode 0 – Standard Configuration



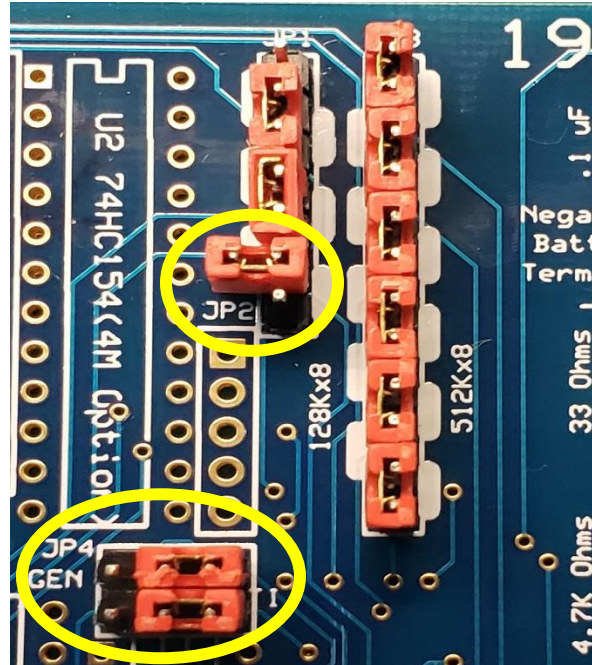


Figure 3 – OpMode 1 – Dual RAMDisk Configuration

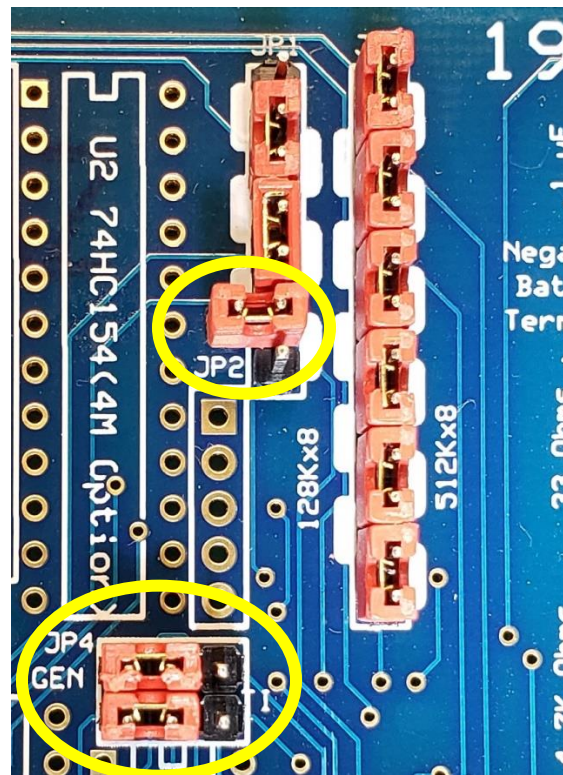


Figure 4 – OpMode 2 –Phoenix Configuration

## 7.0 OpMode 0 – Standard Configuration

Operational mode 0 (figure 2) is the standard HRD RAMDisk configuration as it was originally designed to operate *and is the most common configuration you can expect to work with*. It has a DSR space, 0x4000 to 0x5FFF, with the DSR memory showing 6k from 0x4000 to 0x57FF and a 2K RACK showing from 0x5800 to 0x5FFF. The RACKs are addressed by the CRU starting at CRU bit 1 and use successive bits up to CRU bit 13 to form RACK numbers addressing the first to the final RACK (number of bits used depends on choice of SRAM chips and how many chips are installed.) The card is enabled/disabled in standard fashion with CRU bit 0 by using a DSRLNK or other custom routine. On the HRD4000B, CRU bit 14 is assigned to swap between two 6K DSR memory pages within the DSR address space. The highest bit, CRU 15, enables/disables RAMBO mode (discussed later in its own separate section).

**NOTE:** The user should only use a single CRU **non-Phoenix** base address DIP switch setting for this mode (see appendix A for more information).

### 7.1 CRU Assignments – CRU bits 1-13

The CRU bits 1-13 are reserved for mapping SRAM memory as RACKs. Some of the CRU bit assignments for SRAM mapping are physically different depending on how JP1 and JP3 are configured; for either 128Kx8 SRAMs or 512Kx8 SRAMs. The CRU maps for the two configurations are shown in figures 6 and 7. Those bits address RACKs in the 0x5800 to 0x5FFF DSR space for your applications/data. But, before your applications use the RACKs memory, they must first “know” how many RACKs are on the HRD board.

### 7.2 Conditions for Determining Number of RACKs

There are three conditions that must be considered in the design of your algorithm to determine the number of RACKs on an HRD4000B board in OpMode 0. The first condition is that the board is not fully populated with two layers of SRAM. The second is that the board is fully populated with two layers of 128x8k SRAM. The third is that the board is fully populated with two layers of 512kx8 SRAMs. Your algorithm must be able to work with each condition in order to determine number of RACKs available. *In all these cases, it is assumed that the user has not mixed SRAM chip types and has populated SRAMs on the board without skipping a SRAM space!*

#### 7.2.1 First Condition – Board not fully populated

One approach for determining number of RACKs for the first condition is to check for SRAM in RACK space starting with RACK 0, continuing to subsequent RACKs in a linear fashion, and stopping at the first RACK space that does not contain SRAM. That count gives you the total number of RACKs on the board.

## 7.2.2 Second Condition – Board fully populated with 128Kx8 SRAMs

In this condition, when your count reaches 2048 RACKs (RACK 2047) and you haven't detected a RACK space with no SRAM, the next RACK counted will "flip" back around to RACK 0 and start counting up again to RACK 2047. When you encounter this you will know that the board is fully populated with 128Kx8 SRAMs.

This happens because CRU bits 0-11 roll over to zero as the count addresses CRU bit 12. CRU bits 12 and 13 are not used with 128Kx8 SRAMs and are physically disconnected on the board by jumpers JP1 and JP3. So, in this case, all the SRAM memory selection "sees" is that the CRU bits 1-11 have started counting again starting at RACK 0.

## 7.2.3 Third Condition – Board fully populated with 512Kx8 SRAMs

For this condition, when you reach the state where the CRU bits 1-13 are all set and you've counted 8192 RACKs, you're done.

## 7.2.4 Option to combine the Second and Third Conditions for simplification

*If it is not important that the DSR memory swaps during the determination of the number of RACKs, you can use a single strategy for both conditions two and three.* To determine number of RACKs, your algorithm can step through the CRU bits 1-14 and look for the first RACK with no SRAM or find the first "roll over" back to RACK 0. If you choose this simplifying strategy, it may be wise to reset the CRU bit 14 when done if it was set. *Be careful not to spill over to the CRU bit 15 or you will enable the RAMBO with possible unintended consequences!*

## 7.2.5 Example

Let's look at an example for a board populated with 128Kx8 SRAM. Figure 6 shows that the CRU bits 1 to 11 are used to address SRAM RACKs. Starting at RACK 0 and checking successive RACKs for SRAM:

```
15-----8 7-----0
00000000 00000001 Start at RACK 0 and look for SRAM (bit 0 turns card on!)
00000000 00000011 RACK 1
00000000 00000101 RACK 2
```

If you encounter your first RACK with no SRAM RACK space then all possible RACKs have been counted for OpMode 0 and you're done.

```
00001111 11111101 RACK 2046
00001111 11111111 RACK 2047 - You've reached the end for all 2048 RACKs if
each space showed SRAM.
00010000 00000001 RACK 0 - Continuing to the next count, the CRU addressing
rolls back to RACK 0! CRU bits 1 – 11 are zero again. If this
```

happens then the card is fully populated with 128Kx8 SRAMs, both layers.

The same logic applies to a card populated with 512Kx8 SRAM using the CRU bits 1 to 13. The only difference is that the roll back occurs on a fully populated card when you first reach CRU bit 14 (see figure 7) *which will also swap the DSR memory page so be aware.*

This illustrates just one possible approach to determining number of RACKs in OpMode 0. There are no doubt other approaches that can be used but, understanding how the card behaves when addressed is really the key and purpose of this example.

### 7.3 CRU Assignments – CRU bit 14

Simple, the CRU bit 14 swaps between two 6K DSR memory pages in the 0x4000 to 0x57FF DSR memory space. CRU 14 default is zero on power up (see figure 5 and Appendix D for additional mapping information).

## 8.0 OpMode 1 – Dual RAMDisk Configuration

In OpMode 1, the HRD4000B is logically “split” into two separate cards, each with its own CRU base address, SRAM and DSR memory. The jumper configuration for OpMode 1 is pictured in figure 3. To DSRLNK routines it “appears” as though there are two cards in the Peripheral Expansion Box (PEB)! This operability is a fortunate consequence of the addition of the Phoenix functionality (whether intended or not is unknown to the author) but does come with a couple of complications. The first is that **two CRU base addresses** on the CRU selection DIP switches must be selected; one “non-Phoenix” DIP switch AND one “Phoenix” DIP switch which limits address choices somewhat (see Appendix A for “CRU Address DIP switch selections”).

The second is that there is a “gap” in the memory selection CRU bit map which complicates RACK determination and memory management (see figure 8).

This OpMode, as with OpMode 0, has a DSR space, 0x4000 to 0x5FFF, with the DSR memory showing 6k from 0x4000 to 0x57FF and a 2K RACK showing from 0x5800 to 0x5FFF. The card is enabled and disabled in standard fashion with the CRU bit 0 by using a DSRLNK or other custom routine. CRU bit 14 is assigned to swap between two 6K DSR memory pages within the DSR space of EACH logical card. The highest bit, CRU 15, still enables and disables RAMBO mode.

DSR space address	>4000	<b>Space 0</b> OpMode 1 Phoenix DIP address selected and DSR space bit 14=0	M32 physical address	>0000
	>57FF			
	>5800	RACK if RAMBO=0 else		
	>5FFF	"hidden" 2K if RAMBO=1		
	>4000	<b>Space 1</b> OpMode 1 Phoenix DIP address selected and DSR space bit 14=1		
	>57FF			
	>5800	RACK if RAMBO=0 else		
	>5FFF	"hidden" 2K if RAMBO=1		
	>4000	<b>Space 2</b> OpModes 0,1 non-Phoenix DIP address selected and DSR space bit 14=0		
	>57FF			
	>5800	RACK if RAMBO=0 else		
	>5FFF	"hidden" 2K if RAMBO=1		
	>4000	<b>Space 3</b> OpModes 0,1 non-Phoenix DIP address selected and DSR space bit 14=1		
	>57FF			
	>5800	RACK if RAMBO=0 else		
	>5FFF	"hidden" 2K if RAMBO=1		>7FFF

Figure 5 - DSR Memory Mapping of M32 for OpModes 0 and 1

Again, the first task that software needs to accomplish is to determine the number of RACKs on each logical card and make those values available to other routines so, understanding how memory is partitioned in this OpMode is important.



HRD4000B w/128Kx8 chips		
CRU bit	Bit Function	Notes
0	DSR	Turns card DSR on/off
1	A12	(AB3) These 6 bits form the upper memory address of RAMDisk SRAM
2	A11	(AB4) for M0 - M31.
3	A14	(AB2) <b>Bit Function</b> column is SRAM pin, (ABx) is schematic designation.
4	A13	(AB1) "
5	A15	(AB6) "
6	A16	(AB5) "
7	A	RAMDisk SRAM chip select-These shared four bits form sixteen SRAM chip
8	B	selects by U1 and U2; the 74HC154 4-Line to 16-Line Demultiplexers.
9	C	U1 selects M0-M15, U2 selects M16-M31-toggled by CRU bit 11
10	D	"
11	SRAM Layer	Layer select: <b>0</b> =for bottom layer M0-M15, <b>1</b> =for top layer M16-M31 by toggling *G of U1 and U2, the 74HC154s. <b>Together: CRU bits 1 - 11 address all possible 2K RAMtraCKs (RACKs)!</b>
12	-	Not connected
13	-	Not connected
14	DSR space	Swaps DSR space: <b>0</b> for space 0 or space 2, <b>1</b> for space 1 or space 3 This CRU bit toggles A13 of U9, the 32Kx8 DSR memory IC.  <b>NOTE:</b> Using a Phoenix CRU DIP switch selection sets A14 of U9 to <b>0</b> and with this CRU bit allows access to space 0 or space 1 of U9.  Using a non-Phoenix DIP switch selection sets A14 of U9 to <b>1</b> and with this CRU bit allows access to space 2 or space 3 of U9.  By splitting the card with JP2 and using both a non-Phoenix and a Phoenix CRU DIP switch selection, all 32K becomes available; 16K for the non-Phoenix half and 16K for the Phoenix half.
15	RAMBO & DSR upper 2K	RAMBO select: <b>0</b> for non-RAMBO, <b>1</b> for RAMBO. Also "unhides" upper 2k of U9 8K DSR space by swapping out the RACK that is in that space! <b>NOTE:</b> CRU bits 1,2 remain latched but are not applied to RAMBO window.

Figure 6 – OpModes 0/1 CRU assignments for HRD4000B with 128Kx8 SRAMs

HRD4000B w/512Kx8 chips		
CRU bit	Bit Function	Notes
0	DSR	Turns card DSR on/off
1	A12	(AB3) These 8 bits form the upper memory address of RAMDisk SRAM
2	A11	(AB4) for M0 - M31.
3	A14	(AB2) <b>Bit Function</b> column is SRAM pin, (ABx) is schematic designation.
4	A13	(AB1) "
5	A18	(AB7) "
6	A15	(AB6) "
7	A16	(AB5) "
8	A17	(AB8) "
9	A	RAMDisk SRAM chip select-These shared four bits form sixteen SRAM chip
10	B	selects by U1 and U2; the 74HC154 4-Line to 16-Line Demultiplexers.
11	C	U1 selects M0-M15, U2 selects M16-M31-toggled by CRU bit 13
12	D	"
13	SRAM Layer	Layer select: <b>0</b> =for bottom layer M0-M15, <b>1</b> =for top layer M16-M31 by toggling *G of U1 and U2, the 74HC154s. <b>Together: CRU bits 1 - 13 address all possible 2K RAMtraCKs (RACKs)!</b>
14	DSR space	Swaps DSR space: <b>0</b> for space 0 or space 2, <b>1</b> for space 1 or space 3 This CRU bit toggles A13 of U9, the 32Kx8 DSR memory IC.  <b>NOTE:</b> Using a Phoenix CRU DIP switch selection sets A14 of U9 to <b>0</b> and with this CRU bit allows access to space 0 or space 1 of U9.  Using a non-Phoenix DIP switch selection sets A14 of U9 to <b>1</b> and with this CRU bit allows access to space 2 or space 3 of U9.  By splitting the card with JP2 and using both a non-Phoenix and a Phoenix CRU DIP switch selection, all 32K becomes available; 16K for the non-Phoenix half and 16K for the Phoenix half.
15	RAMBO & DSR upper 2K	RAMBO select: <b>0</b> for non-RAMBO, <b>1</b> for RAMBO. Also "unhides" upper 2k of U9 8K DSR space by swapping out the RACK that is in that space! <b>NOTE:</b> CRU bits 1,2 remain latched but are not applied to RAMBO window.

Figure 7 – OpModes 0/1 CRU assignments for HRD4000B with 512Kx8 SRAMs

### 8.1 Physical Partitioning of Memory in OpMode 1

In Operational Mode 1 the most significant **Bank** selection CRU bit is *disconnected* from CRU operations and that bank selection bit is, instead, determined by the CRU DIP switch address. Specifically, the CRU bit 10 is disconnected on a board with 128Kx8 SRAM and the CRU bit 12 is disconnected on a board with 512Kx8 SRAM (see figure 8). A bit written to either of these CRU bits in OpMode 1 is “ignored.” Instead, the Phoenix CRU DIP switch address will set the most significant bank selection bit to zero and the non-Phoenix CRU DIP switch address will set it to one (via an SR latch).

So, when software turns the HRD 4000B card on with a Phoenix address, SRAMs M0-M7 & M16-M24 are available as well as DSR spaces 0 & 1. For the non-Phoenix address, SRAMs M8-M15 & M25-M31 and DSR spaces 2 & 3 are available. Again, which of the two DSR spaces selected on either logical card is determined by the state of CRU bit 14 (figure 5 and Appendix D).

A couple of observations based on that arrangement. SRAM layer one and layer two are NOT in a contiguous memory space. That has software implications. And, just because the SRAM memory space is split in two it doesn't necessarily mean that there are equal numbers of RACKs available on both logical cards. That is a function of how many SRAMs are installed. Example; a board with twelve 512Kx8 SRAMs will show 2048 RACKs on one split and 1024 RACKs on the other.

#### 8.1.1 Conditions for Determining the Number of RACKs

The conditions for determining the number of RACKs in OpMode 1 are similar to those of OpMode 0 with two exceptions. The first is that there is one less bit in the **Pages/Banks** CRU selection bits because the most significant **Bank** selection CRU bit is ignored. Secondly, the layers must be checked separately unlike the example in OpMode 0. This means that your algorithm may need to repeat the RACKs search twice, once for each layer or use some other method to account for the possibility of two layers of SRAM.

So, condition one is a *layer* that is not fully populated. Condition two is a *layer* fully populated with 128Kx8 SRAM. And, condition three is a *layer* fully populated with 512Kx8 SRAM (reference the “Conditions for Determining Number of RACKs” in the OpMode 0 section).

#### 8.1.2 Example

Let's look at an example for a board populated with 128Kx8 SRAM. Figure 6 shows that the CRU bits 1 to 11 are used to address SRAM RACKs, however, CRU bit 10 is now ignored (figure 8).

Starting at RACK 0 and checking successive RACKs for SRAM:

```
15-----8 7-----0
00000x00 00000001 Start at RACK 0 and look for SRAM (bit 0 turns card on!)
00000x00 00000011 RACK 1
00000x00 00000101 RACK 2
```

.  
If you encounter your first RACK with no SRAM RACK space then all possible RACKs on this layer have been counted and you're done.

```
.
00000x11 11111101 RACK 510
00000x11 11111111 RACK 511 - You've reached the end for all 512 RACKs if
each space showed SRAM.
```

bit 10

00000x00 00000001 RACK 0 - Continuing to the next count ↓, the CRU addressing rolls back to RACK 0! CRU bits 1 – 9 are zero again. If this happens then the first card layer is fully populated with 128Kx8 SRAMs.

*The same logic applies to a card populated with 512Kx8 SRAM using the CRU bits 1 to 13. The only difference is that the roll back occurs on a fully populated first layer when you first reach CRU bit 12 (see figure 8).*

Now, the process repeats after setting the second layer bit (CRU bit 11 in this example).

```
15-----8 7-----0
00001x00 00000001 Start at RACK 512 and look for SRAM (bit 0 turns card on!)
00001x00 00000011 RACK 513
00001x00 00000101 RACK 514
```

.  
If you encounter your first RACK with no SRAM RACK space then all possible RACKs on this layer have been counted and you're done.

```
.
00001x11 11111101 RACK 1022
00001x11 11111111 RACK 1023 - You've reached the end for all 1024 RACKs if
each space showed SRAM.
```

bit 10

00001x00 00000001 RACK 0 - Continuing to the next count ↓, the CRU addressing rolls back to RACK 512! CRU bits 1 – 9 are zero again. If this happens then the second card layer is fully populated with 128Kx8 SRAMs.

*The same logic applies to a card populated with 512Kx8 SRAM using the CRU bits 1 to 13 when second layer CRU bit 13 set. The only difference is that the roll back*

occurs on a fully populated second layer when you first reach CRU bit 12 (see figure 8).

## 8.2 CRU Assignments – CRU bit 14

The CRU bit 14 swaps between two DSR memory pages in the 0x4000 to 0x57FF DSR memory space for EACH of the CRU DIP switch base addresses. CRU 14 default is zero on power up. Specifically the two DSR spaces available in this OpMode are spaces 2 or 3 for the non-Phoenix CRU DIP switch address and spaces 0 or 1 for the Phoenix CRU DIP switch address as detailed in figure 5 (also see Appendix A for “CRU Address DIP switch selections” for setup details and Appendix D for OpMode mapping). *Which DSR spaces are used is academic, the application software really doesn't care what space in the 32Kx8 DSR memory is being addressed, just that they're unique and don't overlap.*

HRD4000B w/128Kx8 chips			
	CRU	Bit	Notes
	bit	Function	
Banks	7	A	Ramdisk SRAM chip select-These shared four bits form sixteen SRAM chip
	8	B	selects by U1 and U2; the 74HC154 4-Line to 16-Line Demultiplexers.
	9	C	U1 selects M0-M15, U2 selects M16-M31-toggled by CRU bit 11
	10	D	NOT USED IN OpMode 1
	11	SRAM Layer	Layer select: 0=for bottom layer M0-M15, 1=for top layer M16-M31 by toggling *G of U1 and U2, the 74HC154s. Together: CRU bits 1 - 11 address all possible 2K RAMtraCKs (RACKs)!

HRD4000B w/512Kx8 chips			
	CRU	Bit	Notes
	bit	Function	
Banks	9	A	Ramdisk SRAM chip select-These shared four bits form sixteen SRAM chip
	10	B	selects by U1 and U2; the 74HC154 4-Line to 16-Line Demultiplexers.
	11	C	U1 selects M0-M15, U2 selects M16-M31-toggled by CRU bit 13
	12	D	NOT USED IN OpMode 1
	13	SRAM Layer	Layer select: 0=for bottom layer M0-M15, 1=for top layer M16-M31 by toggling *G of U1 and U2, the 74HC154s. Together: CRU bits 1 - 13 address all possible 2K RAMtraCKs (RACKs)!

Figure 8 - OpMode 1 Bank D CRU Bit NOT USED

## 9.0 OpMode 2 – Phoenix Configuration

In OpMode 2, the HRD4000B is logically “split” into two separate cards, each with its own SRAM and DSR memory. **However, both logical cards share a single Phoenix CRU DIP switch address!** This configuration was originally designed to work with the Geneve 9640 to allow for both a BOOT drive and a RAMDisk. The jumper configuration for OpMode 2 is pictured in figure 4.

This OpMode, as with OpModes 0 and 1, has a DSR space, 0x4000 to 0x5FFF, with the DSR memory showing 6k from 0x4000 to 0x57FF and a 2K RACK showing from 0x5800 to

0x5FFF. The card is enabled and disabled in standard fashion with CRU bit 0 by using a DSRLNK or other custom routine. CRU bit 14 is assigned to swap between two 6K DSR memory pages within the DSR space for EACH logical card. The highest bit, CRU 15, still enables and disables RAMBO mode.

### 9.1 Physical Partitioning of Memory in OpMode 2

In Operational Mode 2, the most significant **Bank** selection CRU bit is determined by either an 8-bit CRU write operation or a 16-bit CRU write operation. It is the *write operation* that sets the bit, not what is written to the bit! An 8-bit CRU write operation sets the Bank bit to one and a 16-bit write operation sets it to zero. Technically, any CRU write operation within the 0 to 7 bit range will set it to one and any CRU write operation within the 8 to 15 bit range will set it to zero and this is what effectively splits memory on the card into two sections. This occurs because the most significant Bank bit latches (SR latch) a derived bit from the PEB bus address line 11 (A11) during a CRU write operation. Figure 9 illustrates that A11 switches from a zero to a one at the CRU bit 7/8 boundary. And, incidentally, the DSR page is also flipped as A11 goes from one state to the other because A11 is associated with an address line on the DSR memory chip.

So, when any of the lower 8 CRU bits are written, SRAMs M8-M15 & M25-M31 and DSR spaces 2 & 3 SRAMs become available. For CRU writes from bit 8 to 15, M0-M7 & M16-M24 and DSR spaces 0 & 1 are available. Again, which of the two DSR spaces selected on a particular logical card split is determined by the state of CRU bit 14 (see Appendix D).

### 9.2 Some Observations of OpMode2


There is some interesting behavior in this mode! For an 8-bit CRU write you really only have access to 128 RACKs at a time because bit 0 is reserved to turn the card on and off leaving only seven bits for addressing the RACKs. Not to worry, you can use a 16-bit write to select the next set of RACKs depending on which memory selection bits you set (see previous sections for more on memory selection as well as figures 6 and 7). But when you do that, remember, your DSR memory swaps until you return to another 8-bit CRU write operation...then it swaps back! So, if you are using the DSR spaces for your application/data then it becomes especially important to keep track of which space you're in as you swap back and forth between the 8-bit writes and the 16-bit writes. It's a wild and crazy ride!

### 9.3 Epilogue for OpMode 2

Even though the HRD4000B is compatible with the Geneve 9640, this particular mode may be obsolete or no longer used. It did require additional software to make it work; it is left to the reader to research if interested. This mode is most similar to OpMode 1 with the same "gap" in the memory map between SRAM layers. It is without a doubt

the most difficult mode to work with but, if you have a need to split the HRD using only a single CRU DIP switch base address then it is an option.

PHOENIX 8 bit/16 bit CRU Address Logical Card Split <sup>1</sup>														
CRU	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14		
0	1	x <sup>2</sup>	x	x	x	0	0	0	0	0	0	0	>1x00	
1	1	x	x	x	x	0	0	0	0	0	0	1	>1x02	
2	1	x	x	x	x	0	0	0	0	0	1	0	>1x04	
3	1	x	x	x	x	0	0	0	0	0	1	1	>1x06	
4	1	x	x	x	x	0	0	0	0	1	0	0	>1x08	
5	1	x	x	x	x	0	0	0	0	1	0	1	>1x0A	
6	1	x	x	x	x	0	0	0	0	1	1	0	>1x0C	
7	1	x	x	x	x	0	0	0	0	1	1	1	>1x0E	
8	1	x	x	x	x	0	0	0	1	0	0	0	>1x10	
9	1	x	x	x	x	0	0	0	1	0	0	1	>1x12	
10	1	x	x	x	x	0	0	0	1	0	1	0	>1x14	
11	1	x	x	x	x	0	0	0	1	0	1	1	>1x16	
12	1	x	x	x	x	0	0	0	1	1	0	0	>1x18	
13	1	x	x	x	x	0	0	0	1	1	0	1	>1x1A	
14	1	x	x	x	x	0	0	0	1	1	1	0	>1x1C	
15	1	x	x	x	x	0	0	0	1	1	1	1	>1x1E	



NOTES: 1. 8-bit-write addresses BOOT disk, 16-bit-write addresses Ramdisk  
2. A3-A7 make up base CRU address of card; >1000 to >1F00

Figure 9 - Phoenix CRU Address Line 11

## 10.0 Hey, isn't there a fourth mode you're not talking about?

Yes, yes. There is but it's more like an appendix; it's there but not really useful. If you start from the Phoenix configuration (figure 4) then put JP2 back in the vertical position (in line with the other jumpers), you'll be in this configuration. So, what does it do? Simply put, it gives you access to SRAMs like the single card OpMode 0 case and the crazy DSR swapping behavior of OpMode2. Perhaps I'm mistaken; maybe it is useful for some application?

## 11.0 RAMBO

The RAM Block Operator mode, RAMBO, is a configuration that does two things. First, it presents four RACKs of memory in the 8K cartridge address space, 0x6000 to 0x7FFF, even if there is a cartridge plugged into the console. Second, it disables SRAM from showing in the RACK space of the card, 0x5800 to 0x5FFF. Instead, the entire 8k of the DSR memory becomes visible and usable from 0x4000 to 0x5FFF. The top 2K of DSR



memory, which is normally not addressable, is sometimes referred to as the “hidden 2K” of the DSR memory (see figure 5) made visible as a consequence of RAMBO.

### 11.1 CRU Assignments

RAMBO is activated when the CRU bit 15 is set to one. When active, the CRU bits 1 and 2 are no longer routed to memory address selection. Both CRU bits can still be set and reset, but they will not affect which RACKs are presented in the cartridge address space. This is because RAMBO works on 8K SRAM boundaries and not the 2K RACK SRAM boundaries so the lower two CRU bits are “ignored.” However, when you exit RAMBO, the RACK that is selected will “reappear” in the RACK space of the card. In general, it is *recommended* that those two CRU bits be set to zero for RAMBO but it is not a requirement. That leaves RAMBO memory addressing to start at the CRU bit 3. Starting at the CRU bit 3 and advancing to subsequent addresses will page the RAMBO cartridge space SRAM in 8K pages up to the end of all available SRAM memory.

### 11.2 Some Observations and Considerations

This is stating the obvious but; don't run software in the cartridge space that enables RAMBO. RAMBO takes over that address space and your program will no longer be there.

In general, applications only care that there are 8K SRAM pages to work with. They don't necessarily care which RACKs are in that space at any one time. You will have to keep track of memory use, of course, and how many 8K pages the application has to work with. It can be managed knowing how many RACKs are available on the card.

**IF**, you need to know which RACKs are being presented in the 8K cartridge space, it can be calculated. First, restating, there are four RACKs projected into the 8K cartridge space. **The RACKs presented in cartridge space are not presented in RACK counting order.** Example: If RACK 0 is selected and then RAMBO is enabled; 0x6000-0x67FF will present RACK 0, 0x6800-6FFF will present RACK 2, 0x7000-77FF will present RACK 1 and 0x7800-0x7FFF will present RACK 3. **They aren't in order!**

Another consideration, if you had RACK 1 selected and then activated RAMBO what would be presented in the 8K cartridge space would be the same; RACK 0, RACK2, RACK 1, RACK 3 in that order. Ditto for RACK 2 or RACK 3 being selected before activating RAMBO. Remember, RAMBO ignores the CRU bits 1 and 2 so any RACKs selected within an 8K memory boundary will appear in cartridge space in the above relative sequence. Figure 10 attempts to illustrate the point. This also means that if you are mixing the use of 2K RACKs and 8K RAMBO pages, you need to be careful to stay on 8K SRAM boundaries with RAMBO or you may overwrite a 2K RACK unintentionally. If you have a need to correlate the RACK currently selected with its address in cartridge space while in RAMBO mode, see Appendix C for example code that could be used.



Finally, RAMBO works as designed in both OpMode 0 and OpMode 1. The author has not experimented with RAMBO while in OpMode2, Phoenix, not believing that there is value in doing so.

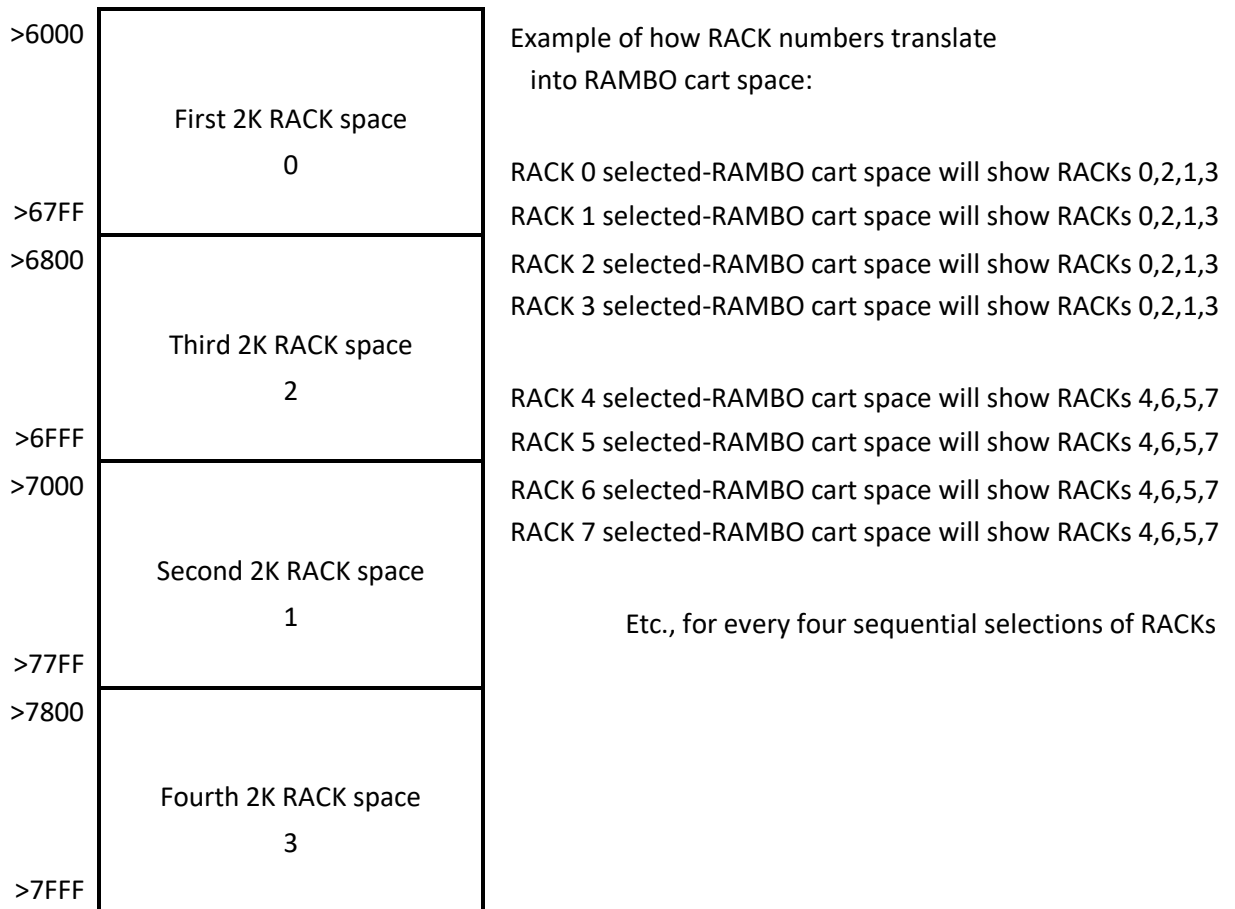
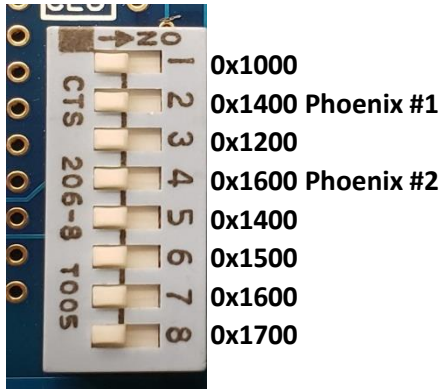


Figure 10 - RAMBO RACK Correlation to Cartridge Space

## APPENDIX A – CRU Address DIP Switch



**Figure 11 - HRD4000B CRU Base Address Assignments**

## Setting CRU Base Address

For OpMode 0 single card operation, select only ONE CRU DIP switch from 1,3,5,6,7,8 to the ON position (do not use DIP switch 2 or 4). All other DIP switches should be in the OFF position.

For OpMode 1, dual logical card operation, TWO CRU DIP switches need to be selected. First select either DIP switch 2 (Phoenix #1) OR DIP switch 4 (Phoenix #2). Then select either; 1,3,5,6,7,8 for the second DIP switch. **READ CAUTION FIRST** and read “CRU Assignment Considerations” below.

For OpMode 2 Geneve 9640 Phoenix operation, set only one of the Phoenix DIP switches (2 or 4) to the ON position. All others should be in the off position.

**>>> CAUTION! <<<**

For all CRU DIP switch selections observe the following! If the configuration calls for using a non-Phoenix DIP switch, then only choose **ONLY ONE** DIP switch from 1,3,5,6,7,8 at any time. If your configuration calls for using (*or also using*) a Phoenix DIP switch, then only choose either 2 **OR** 4. Not observing this caution will create an address conflict for the HORIZON.

**This is important!** If DIP switch 2 is used then DIP switch 5 **CANNOT** be used as a second selection. If DIP switch 4 is used then DIP switch 7 **CANNOT** be used as a second selection. DIP switch 2 is physically connected to DIP switch 5 and DIP switch 4 is physically connected to DIP switch 7 on the HORIZON HRD4000B board. Not observing this caution will create an address conflict.

**CRU Assignment Consideration:** The DIP Switch selection of the CRU base address for the HORIZON card is dependent on other cards you have installed in your PEB. Selecting a CRU address that is being used by another card will cause a conflict with unpredictable results.

## APPENDIX B – RAMBO Operation with ROS 8.xx

Reprinted from, "HORIZON RAMDISK User Operating Manual" 3 May 2020. Explanation of how ROS software ver 8.14F and later interfaces with RAMBO functionality. NOTE that the CRU assignments in this section are for an earlier version of Horizon HRD, not the HRD4000B.

### 3.5.B0 BANK OPERATOR for RAMBO/HORIZON cards

This section of the manual will try to explain the memory management sub-program called >B0 and how to interface with it using assembly code. This low-level sub-program will only work on Horizon cards with the RAMBO hardware installed. If you need more help on using this sub-program, see section 5.6 of this manual.

The original Horizon hardware only allowed the Horizon memory to be accessed in 2K blocks at the DSR space from >5800 thru >5FFF. The rest of the DSR space from >4000 thru >57FF was reserved for the Horizon ROS. This limited access to the memory on the Horizon card was fine for uses like a ramdisk, but to write a program which really executed from and used the Horizon memory was almost impossible. There needed to be a new way of accessing the Horizon memory, but without losing the wonderful and powerful ramdisk features and functions.

With this in mind, OPA/BMS decided to design a simple upgrade kit for all Horizon cards which would allow the upgraded Horizon to work the same in the original Horizon mode; but when accessed thru a new mode it would switch to a whole new paging system which would be better designed for writing large programs, etc. The ROS would also have to be upgraded to support the new mode by adding a super memory-management sub-program to ease in the writing of large programs as well as allowing both ramdisk files and program space to share the same memory.

Finally, with the RAMBO hardware and the SERIES 8 software, it is now possible to write large programs which used the RAMBO program space for DATA storage very easily. The SERIES 8 software added a sub-program called >B0 to handle the complex memory management needed to make sure all programs run perfectly without destroying any files or other programs which may be sharing the memory on-board the Horizon card.

All the values for the sub-program >B0 are transferred thru CPU PAD addresses, which will be explained later on. Access to the sub-program >B0 is similar to the normal DSRLNK type >8 which is for DSR devices like "DSK1.", and "RS232."; except the DSR only needs a 2-byte VDP PAB of >01B0, the first byte being the length and the second byte being the sub-program name >B0. The following section is divided into five different parts, the first two explaining the CRU interface bits and the last three explaining the three different ways to access the sub-program >B0.

#### **HORIZON/RAMBO CRU INTERFACE ACCESS:**

The original Horizon interface has a CRU address range from >1000 thru >1700. The 8- or 16- bit CRU bits relate to the pages, banks, and SRAM/DSR selections. See the "<Software Developers Guide to the Horizon HRD4000B>", prior revision hardware construction guides, and previous ROS operating manual for more details.

***The following information was retained from the 1990 version of the user operating manual. It has not been fully updated to reflect 512K\*8 chip usage or more than one HCT chip for dual layer operation. Additional review and investigation is necessary to encompass both the HRD4000B and earlier hardware revisions. Use as a loose reference only.***

## HORIZON MODE CRU INTERFACE:

### **CRU BITS | Definition of the CRU interface bits (15=MSBit / 0=LSBit)**

CRU BITS	Definition of the CRU interface bits (15=MSBit / 0=LSBit)
0	= 6K >4000 OFF/ON (0=ROS OFF, 1=ROS ON) Used by ROS and DSRLNK.
1 thru 7	= 2K >5800 Page number 0 thru 127, (Max. of 256K for 8K*8 cards).
8 thru 9	= Select one of four banks of 256K of memory, (Max. of 1024K for 32K*8 cards).
10 thru 11	= Select one of four banks of 1024K of memory, (Max. of 4096K for 128K*8 cards).
12 thru 14	= Undefined, may be used on a future series of Horizon cards.
14	= DSR Space (only w/32K dsr chip in U9) <i>&lt;note by author: specific for HRD4000B&gt;</i>
15	= Reset to 0: used to change to RAMBO mode, on original Horizons it was undefined.

## RAMBO MODE CRU INTERFACE DIRECT ACCESS:

It is highly recommended that all RAMBO memory access is performed via the DSR routines.

A Horizon card with RAMBO installed can work the same as an original Horizon as detailed above or when bit 15 of the CRU interface is set to 1, it can be accessed as follows:

### **CRU BITS Definition of the CRU interface bits (15=MSBit / 0=LSBit)**

CRU BITS	Definition of the CRU interface bits (15=MSBit / 0=LSBit)
0	= 8K >4000 OFF/ON (0=ROS OFF, 1=ROS ON) Used by ROS and DSRLNK.
1 thru 2	= Not used when in RAMBO mode, recommend both be reset to 0.
3 thru 7	= 8K >6000 Page number 0 thru 31, (Max. of 256K for 8K*8 cards).
8 thru 9	= Select one of four banks of 256K of memory, (Max. of 1024K for 32K*8 cards).
10 thru 11	= Select one of four banks of 1024K of memory, (Max. of 4096K for 128K*8 cards).
12 thru 14	= Undefined, may be used on a future series of Horizon cards.
14	= DSR Space (only w/32K dsr chip in U9) <i>&lt;note by author: specific for HRD4000B&gt;</i>
15	= Set to 1: used to change to original Horizon mode.

## **RAMBO DSR ACCESS:**

ACCESS MODE #1 FOR SUB-PROGRAM >B0 (Gets Max number of free 8K pages):

- >834A IN Must be cleared before access to get true number of free pages.  
OUT Max. number of free 8K pages for all RAMBOs in the TI system.
- >834C IN >6000 = gets the Max. number of free 8K pages in the system.  
OUT Not changed, will be the same value as IN.
- >8350 IN Not used for input to the sub-program >B0.  
OUT >0000 if no errors, or >FFFF if errors detected.

Accessing >B0 with >834A and >834C cleared will, upon exit, return to the program in >834A the max. number of free 8K pages for all RAMBOs in your expansion box. This access causes all cards to be switched to original Horizon mode, and being so, can't be accessed from within the >6000 thru >7FFF space. The following is an assembly example which is accessed like so: BL @GETMAX.

	REF VMBW,DSRLNK	Normal VMBW and DSRLNK E/A BLWPs.
GETMAX	CLR @>834A	Clear Max. number of pages.
	CLR @>834C	Ask for Max. number of free 8K pages.
	LI R0,VDPPAB	Address of the PAB in VDP memory.
	LI R1,CALLBO	Address of the PAB in CPU memory.
	LI R2,2	Length of the PAB to be written to VDP memory.
	BLWP @VMBW	Write the PAB (>01B0) to VDP RAM.
	MOV R0,@>8356	Make >8356 point to the length byte of the PAB in VDP.
	BLWP @DSRLNK	Execute the sub-program >B0.
	DATA >A	
	MOV @>8350,R0	Check for errors. EQUAL BIT IS SET IF NO ERRORS
	RT	Return to program.
CALLBO	DATA >01B0	PAB data for sub-program >B0.

ACCESS MODE #2 FOR SUB-PROGRAM >B0 (Selects an 8K page and turns on RAMBO):

>834A	IN	Not used for input to the sub-program >B0.
	OUT	CRU address of the RAMBO/HORIZON card that is currently on.
>834C	IN	A page number from 1 to max. number of free 8K pages.
	OUT	The 16 bit CRU value used to select the page and turn RAMBO on.
>8350	IN	Not used for input to the sub-program >B0.
	OUT	>0000 if no errors, or >FFFF if errors detected.

Accessing >B0 with >834C containing the number of the page you want will turn on the correct RAMBO/HORIZON card and then select the page you specified, and upon exit will return the CRU address and CRU value used to select the page in >834A and >834C. This command also overrides any cartridge which may be plugged into the module port, and will also turn off the P-GRAM card if it was on. This function makes sure no other device is responding at the same >6000 space of memory. The following is an assembly example which is accessed like so: BL @SETPAG with the next line containing a line of DATA with the number of the page you want.

	REF VMBW,DSRLNK	Normal VMBW and DSRLNK E/A BLWPs.
SETPAG	MOV *R11+,@>834C	Get number of the page to select.
	LI R0,VDPPAB	Address of the PAB in VDP memory.
	LI R1,CALLBO	Address of the PAB in CPU memory.
	LI R2,2	Length of the PAB to be written to VDP memory.
	BLWP @VMBW	Write the PAB (>01B0) to VDP RAM.
	MOV R0,@>8356	Make >8356 point to the length byte of the PAB in VDP.
	BLWP @DSRLNK	Execute the sub-program >B0.

	DATA >A	
	MOV @>8350,R0	Check for errors. EQUAL BIT IS SET IF NO ERRORS
	RT	Return to program.
CALLBO	DATA >01B0	PAB data for sub-program >B0.

ACCESS MODE #3 FOR SUB-PROGRAM >B0 (Turns off all RAMBO/HORIZON cards):

>834A IN Not used for input to the sub-program >B0.  
OUT Not changed, will be the same value as IN.

>834C IN >FFFF = Tell all cards to switch back to original Horizon mode.  
OUT Not changed, will be the same value as IN.

>8350 IN Not used for input to the sub-program >B0.  
OUT >0000 if no errors, or >FFFF if errors detected.

Accessing >B0 with >834C containing >FFFF will tell all the Horizon cards in your computer system to switch back to original Horizon mode. This call, since it will remove any page at >6000, can't be accessed from the >6000 space. If for some reason a RAMBO program doesn't switch all the cards back to original Horizon mode, the ROS will do a complete reset any time the TI Title Screen appears. The previous assembly examples should be enough to figure out how to use this mode of the sub-program >B0.

V8.14F Copyright 1990 OPA and Bud Mills Services  
V8.32/8.38/8.42c Updates 2020 T. Tesch  
Horizon Ramdisk wiki and source:  
<https://github.com/horizonramdisk/Horizon-Ramdisk-ti994a/wiki>

Horizon 4000B hardware, layout, and design ownership  
Transferred from Bud Mills Services  
to J. Fetzner (Tekumel Software) 2018



## APPENDIX C – RACK to RAMBO Cart Space Address Calculation

```

*
* Subroutine to convert RACK number
*   to Starting Address in cartridge
*   ROM space where 2K RACK segment
*   resides while in RAMBO mode of the
*   Horizon RAMDisk card.
*
*   BL @RTOADR
*
*   ENTER: RACK number in R0
*
*   LEAVE: R0 contains start address in
*   cartridge space for that RACK
*
*   ALTERS: R0
*   DESTROYS: R1
*
*   DWarren 26 Jan 2020
*
*=====
*
* Perform: R0=R0-(INT(R0/4)*4) to
*   determine which 2K block within the
*   8K space we are looking for
*
RTOADR
        ANDI R0,>0003
*
* Isolate and reverse the two LSbits to
*   calculate 2K page position within
*   the 8K space
*
        MOV R0,R1
        SRL R1,1
        SLA R0,1
        XOR R0,R1
        ANDI R1,>0003
*
* Use page position to calculate actual
*   cartridge space address for given
*   RACK
*
        CLR R0
RTOAD1
        DEC R1
        JNC RTOAD2
        AI R0,>800
        JMP RTOAD1
RTOAD2
        AI R0,>6000
*
        RT

```

## APPENDIX D – OpMode Correlation to DSR Memory and SRAM

		Set/Reset by CRU Bit 14	Set/Reset by Pin 8 of U5 SR Latch per OpMode			
		DSR Memory 32Kx8 U9 Pin 26 - A13	DSR Memory 32Kx8 U9 Pin 1 - A14	DSR Memory 32Kx8 U9 2K SPACE #	DSR Memory 32Kx8 U9 Address	SRAM Accessible
<b>OpMode 0</b>	non-PHOENIX CRU Address	0	1	2	0x7000-0x77FF	M0 - M31
	non-PHOENIX CRU Address	1	1	3	0x7800-0x7FFF	M0 - M31
<b>OpMode 1</b>	non-PHOENIX CRU Address	0	1	2	0x7000-0x77FF	M8 - M15, M24 - M31
	non-PHOENIX CRU Address	1	1	3	0x7800-0x7FFF	M8 - M15, M24 - M31
	PHOENIX CRU Address	0	0	0	0x6000-0x67FF	M0 - M7, M16 - M23
	PHOENIX CRU Address	1	0	1	0x6800-0x6FFF	M0 - M7, M16 - M23
<b>OpMode 2</b>	Lower 8-Bit CRU <u>Write</u>	0	1	2	0x7000-0x77FF	M8 - M15, M24 - M31
	Lower 8-Bit CRU <u>Write</u>	1	1	3	0x7800-0x7FFF	M8 - M15, M24 - M31
	Upper 8-Bit CRU <u>Write</u>	0	0	0	0x6000-0x67FF	M0 - M7, M16 - M23
	Upper 8-Bit CRU <u>Write</u>	1	0	1	0x6800-0x6FFF	M0 - M7, M16 - M23

Figure 12 - OpMode Correlation to DSR Memory and SRAM

For HRD4000B Schematic, reference "Construction Guide Horizon HRD4000B 2020"  
Tekumel Software on <https://github.com/horizonramdisk/Horizon-Ramdisk-ti994a/wiki>