HP-41CY Technical Information

Introduction

The HP-41CY is a modified HP-41 calculator (halfnut model) made by the German company W&W Software Products GmbH. It was introduced in Spring 1987 and sold for DM 1499.-. The main features of this calculator are:

- 64K additional memory
- Turbo switch which doubles the speed
- 4K operating system providing 36 new functions

RAM Usage

The HP-41 has 64K of system memory which is divided into 16 4K blocks (pages). The lower 8 pages are used by the operating system, the upper 8 pages are used by plug-in modules. The additional RAM provided by the HP-41CY is divided into two banks A and B. Only one bank is usually active and is mapped into the address space of the upper 8 pages. Note that plug-in modules will shadow the corresponding RAM page. The HP-41CY operating system resides in page 8 which is write protected by the hardware. We will see later why there are two different versions of the operating system.

Block	Addresses	Primary Bank	Secondary Bank	Bank A (CY)	Bank B (CY)
F	F000-FFFF	Port 4: upper page			
E	E000-EFFF	Port 4: lower page			
D	D000-DFFF	Port 3: upper page			
C	C000-CFFF	Port 3: lower page			
B	B000-BFFF	Port 2: upper page			
A	A000-AFFF	Port 2: lower page			
9	9000-9FFF	Port 1: upper page			
8	8000-8FFF	Port 1: lower page		OS/A	OS/B
7	7000-7FFF	HP-IL ROM			
6	6F00-6FFF	Printer ROM	IR printer		
5	5000-5FFF	Timer ROM	CX		
4	4000-4FFF	Take-over ROM			
3	3000-3FFF	Unused / CX			
2	2000-2FFF	System ROM 2			
1	1000-1FFF	System ROM 1			
0	0000-0FFF	System ROM 0			

Bank Switching

The HP-41CY uses the instructions ENROM1, ENROM2 and ENROM3 to map RAM pages into the system address space. Note that these instructions only work when executed in RAM. The following table shows the effect of those instructions.

ENROM1	Reset all pages to bank A
ENROM2	Enable odd pages of bank B
ENROM3	Enable even pages of bank B

This looks rather asymmetric: bank A can only be activated as a whole whereas for bank B it's possible to activate even and odd pages individually. This raises an interesting question when re-initializing a corrupted HP-41CY, as discussed later. Furthermore, this asymmetry is also the reason why there are two versions of the operating system.

The following table shows the different configurations that are possible when combining the bank switching instructions.

Page	[1]		[2]		[3] := PG01		[4] := PG10	
	ENROM1		ENROM2 ENROM3		ENROM1 ENROM2		ENROM1 ENROM3	
	Bank A	Bank B	Bank A	Bank B	Bank A	Bank B	Bank A	Bank B
F								
E								
D								
C								
B								
A								
9								
8	OS/A	OS/B	OS/A	OS/B	OS/A	OS/B	OS/A	OS/B

The HP-41CY command PG<> switches configurations [1] and [3] to [2] and configurations [2] and [4] to [1] respectively. PG01 enables configuration [3] and PG10 enables configuration [4]. You can easily verify this by plugging in a ROM module, copying it to RAM, exercising the different PGx commands and monitoring the available modules using CAT 2.

Now let's think about how the PG<> command can be implemented. If the currently active configuration is [2] or [4] a simple ENROM1 does the trick. However, when the active configuration is [1] or [3] we need to execute ENROM2 and ENROM3. This is the one and only reason why there are two different versions of the HP-41CY operating system. Amazing, isn't it?

The exact differences (module name, PG<> command, ROM revision and checksum) are listed on the next page.

Operating System Differences

Address	-RAMBOX 64a		-RAMBOX 64b		Comment
…					
804E	0C1	a	0C2	b	*Module name*
804F	034	4	034	4	
8050	036	6	036	6	
8051	020		020		
8052	018	X	018	X	
8053	00F	O	00F	O	
8054	002	B	002	B	
8055	00D	M	00D	M	
8056	001	A	001	A	
8057	012	R	012	R	
8058	02D	-	02D	-	
…					
8DB8	0BE	>	0BE	>	*PG<>*
8DB9	03C	<	03C	<	
8DBA	007	G	007	G	
8DBB	010	P	010	P	
8DBC	180	ENROM2	04E	C=0 ALL	
8DBD	140	ENROM3	100	ENROM1	
8DBE	3E0	RTN	3E0	RTN	
…					
8FFB	004	D	004	D	*ROM revision*
8FFC	035	5	034	4	
8FFD	017	W	017	W	
8FFE	017	W	017	W	
8FFF	22A		0BA		*Page Checksum*

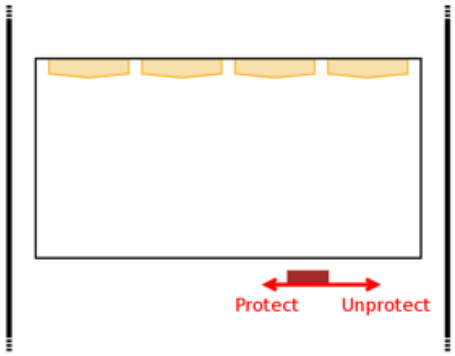
Internal Backup Battery

The internally RAM chips of the HP-41CY are buffered by a small +3V CR3025 lithium cell. This additionally +3V supply voltage is only needed when the four batteries of the HP-41CY are removed or when they dropped down to low voltage levels. Because the operating system itself is also stored in RAM, this hardware architecture results in a long-term problem: When the +6V battery voltage of an HP-41CY falls down for a long time, or when the HP-41CY is stored without batteries, the internal +3V lithium cell buffers the RAM contents. However, when the +3V lithium cell itself drops down, the complete RAM contents will be cleared and unfortunately, the operating system will be lost, too. The typical lifetime of the CR3025 lithium cell is maybe 10 to 20 years, depending on the main battery state of the HP-41CY. Therefore, there is a high risk today that your HP-41CY looses the operating system when you remove the batteries.

In his article „Replacing the internal +3v lithium cell“ Christoph Klug describes a procedure to replace the internal backup battery without loosing the operating system.

OS Write Protection DIP Switch

Using instruction WROM you can write to the RAM of your HP-41CY. This even works when the target page is shadowed by a module plugged into the corresponding port. However, you cannot write to page 8 where the operating system is stored. This is prevented by the HP-41CY hardware. Luckily, there is a switch hidden in the battery compartment to disable the write protection.



Besides this hardware-based write protection there is also a copy protection built into the HP-41CY operating system. If you try to access the page containing the operating system itself (i.e., copy, save or overwrite it), you'll get a NO ACCESS error. However, it's easy to work around this protection as it simply checks if there is a W (017) at address FFD. So you can simply change the value at this address to break it.

Bank Switching Issues

To understand why bank switching can cause crashes you have to know that after a ENROMx instruction is executed the next instruction will be fetched from the target page. Let's assume the active bank is B and we run the PG<> command:

Address	-RAMBOX 64a		-RAMBOX 64b		Comment
8DB8	0BE	>	0BE	>	*PG<>*
8DB9	03C	<	03C	<	
8DBA	007	G	007	G	
8DBB	010	P	010	P	
8DBC	180	ENROM2	04E	C=0 ALL	
8DBD	140	ENROM3	100	ENROM1	
8DBE	3E0	RTN	3E0	RTN	

After executing the ENROM1 instruction in bank B, the RTN instruction at address 8DBE is fetched from bank A. Now it's obvious why the two incarnations of the operating system must perfectly fit together. If page 8 in bank A is corrupted for any reason there is a good chance that the instruction at address 8DBE is not RTN and the calculator goes crazy. However, there is even a bigger problem: if the polling vectors - located at addresses FF4 to FFA in each page - are corrupted in the HP-41CY RAM, the calculator might crash as soon as you turn it on!

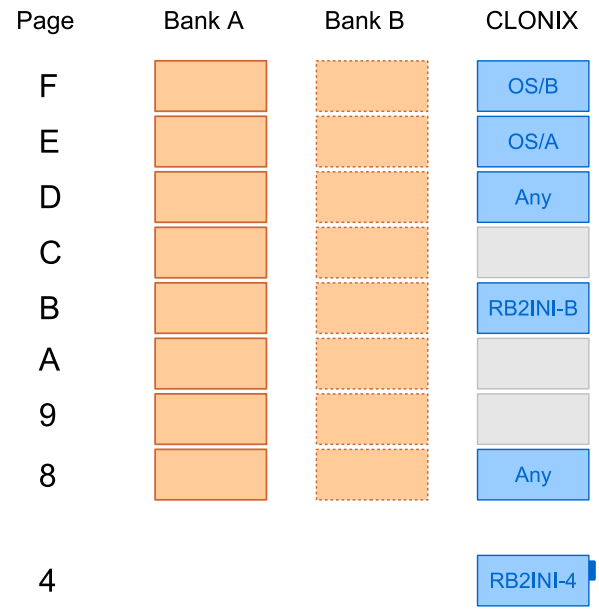
This is the reason why the HP-41CY asks you on MEMORY LOST if the RAM pages should be cleared. At the price of data loss it might bring your calculator back into a working state. The worst case is a corrupted operating system. Usually, you cannot recover your HP-41CY from such an incident. But don't despair, read on

Rising From The Ashes

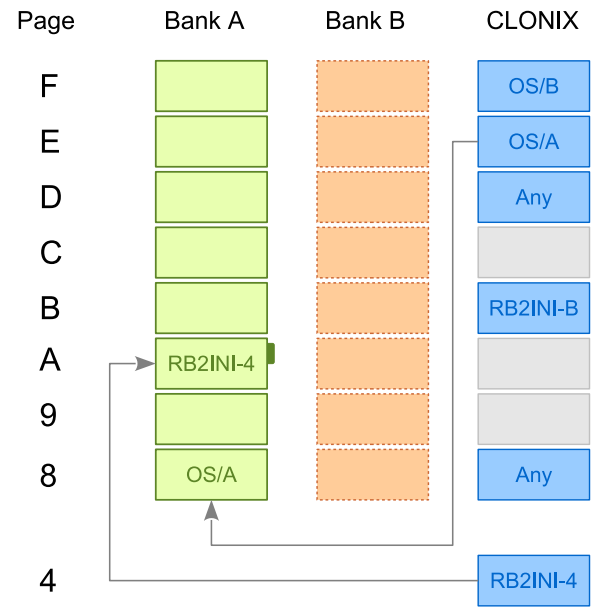
As today the serious HP-41 user owns little gems like the CLONIX module or the PIL-Box, you might think that restoring a corrupted HP-41CY shouldn't be too difficult. Unfortunately, this impression is deceptive. There are three problems to be solved:

1. Find a general procedure to restore the operating system in both banks and to clear the other pages in RAM.
2. Find a way to run the restore procedure that is immune against corrupted polling vectors.
3. Find a safe method to switch to bank A.

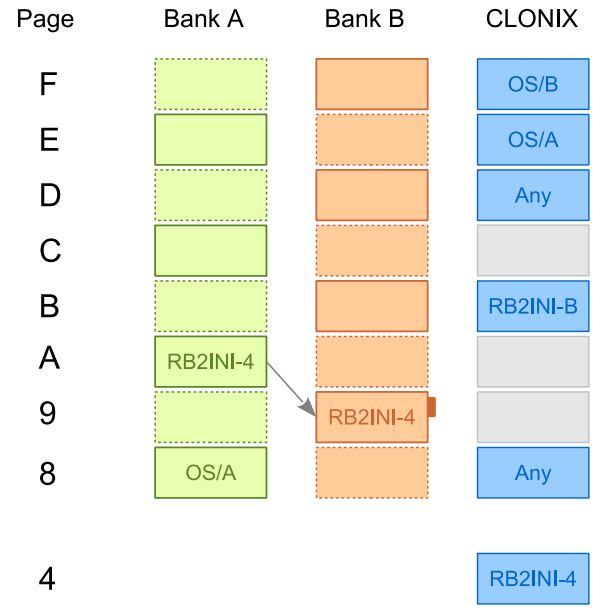
The first problem was not too difficult. My basic idea was to create a CLONIX module containing the restore program as well as the operating system images. Because the ENROMx instructions only have an effect when executed in RAM, I decided that my initialization module replicates itself from ROM (CLONIX) into RAM. Furthermore, I assumed that bank A is active, just to avoid to solve two problems (1. and 3.) at the same time. I came up with the procedure outlined below. It uses the „interleaved" RAM configurations to safely switch from bank A to B (see step 3). RB2INI-4 is the main initialization program written in MCODE, the assembly language of the HP-41. RB2INI-B is just an auxiliary module that helps to solve the third problem. To get the general idea you can ignore it.



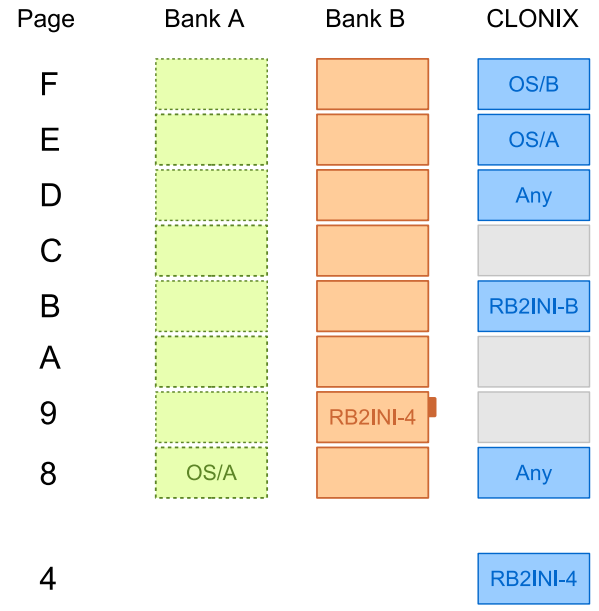
- Assumption: bank A is active
- RB2INI-4 takes over control



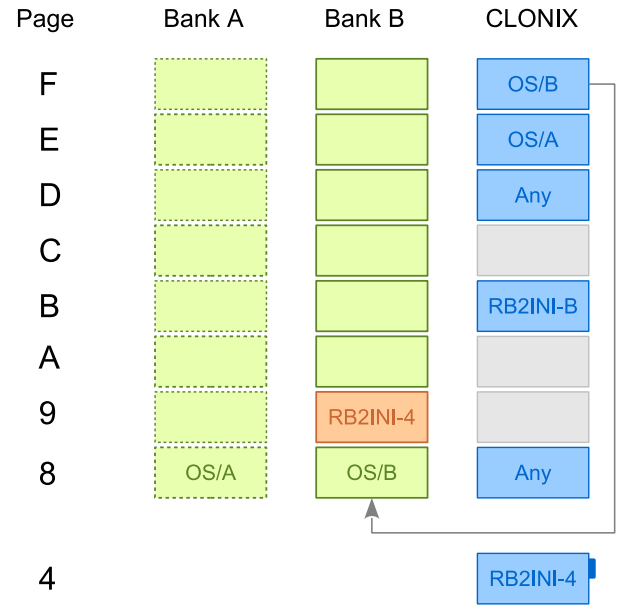
- Copy OS/A from CLONIX to bank A
- Copy RB2INI-4 to page A in bank A
- Clear pages 9 and B to F in bank A
- Transfer control to RB2INI-4 in bank A



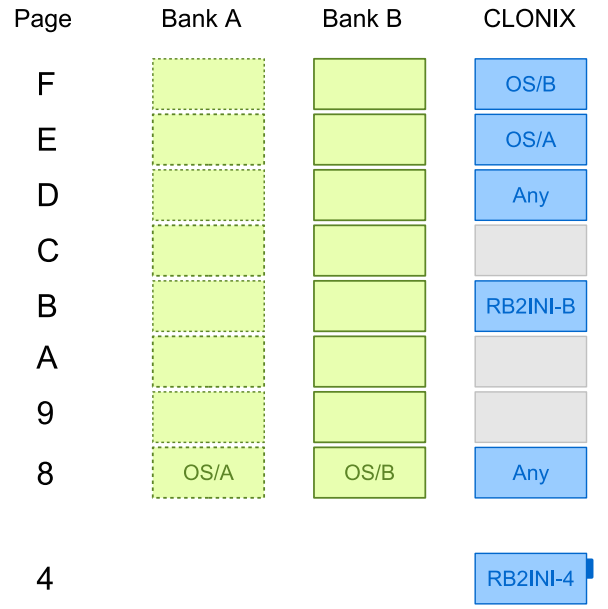
- Activate odd pages of bank B
- Copy RB2INI-4 to page 9 in bank B
- Transfer control to RB2INI-4 in bank B



- Clear page A in bank A
- Activate even pages of bank B



- Copy OS/B from CLONIX to bank B
- Clear pages A to F in bank B
- Transfer control to RB2INI-4 in ROM



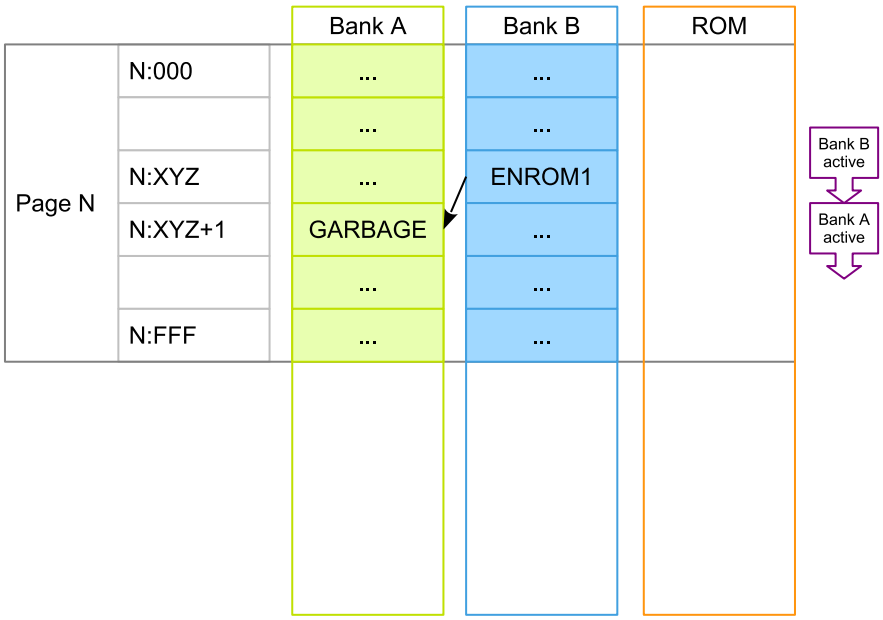
- Clear page 9 in bank B
- Power off

When solving the second problem I initially took the wrong way. I implemented the initialization procedure as a I/O service handler (polling vector pFF8), where I also cleared all the (potentially corrupted) polling vectors in RAM. However, this entry point is just called too late and things may go wrong before. Luckily, I was enlightened at some point. There is already a mechanism in the HP-41 operating system, the so-called take-over ROM's in page 4. When the CPU starts executing the operating system program, one of the very first things it does is to jump to the first address in page 4. If there is no module addressed to this page, the CPU simply continues executing the operating system program. If there is a module addressed to page 4, the CPU will begin executing the program in this module, starting from address 4000. This feature is also used by HP's diagnostic ROM's and seems to be the best choice for the 41CY initialization procedure. Fortunately, it's also supported by the CLONIX programming software.

The third problem really caused me headaches and sleepless nights. As described before and illustrated in the following picture I just didn't see any option for a safe switch from bank B to A when assuming corrupted RAM. Due to the asymmetric bank switching instructions there is simply no way to access and clear RAM in bank A when bank B is active. I was almost ready to give up when I had a sudden inspiration. Because a ROM image shadows the corresponding RAM page I just had to move the bank switching instruction to the end of a page in RAM and put my own ROM into the subsequent page. It's so easy if you know the trick! This is exactly the purpose of the RB2INI-B part. It just jumps back to the RB2INI-4 code and therefore allows for a controlled switch to bank A.

So after a long and interesting journey I only had to put together those three pieces of the puzzle to implement my HP-41CY re-initialization module. Yahoo!

Scenario 1 : Uncontrolled switch from bank B to A (crash!)



Scenario 2 : Controlled switch from bank B to A