



MODULE CREATOR
DIEGO DÍAZ

ZVC REPLICA

PRESENTATION
SYLVAIN CÔTÉ

The new ZVC Replica and the ZEPR0M module revisited

HHC 2024

20240902A

Agenda

- * INTRODUCTION
- * ZEPROM
- * ZVC & ZVC REPLICA
- * THE PROGRAMMER ROM
- * SETUP
- * FLAT 16K
- * BANK SWITCHING
- * FLAT 8K
- * USER CODE FUNCTIONS
- * BANK SWITCHED 16K
- * REFERENCES
- * QUESTIONS

INTRODUCTION



Overview

- * This presentation will cover the following HP-41 products ...
 - * The ZEPRROM module, from Zengrange.
 - * The ZVC module, from Zengrange.
 - * The ZVC Replica module, from Diego Díaz.
 - * The Programmer ROM, from Zengrange.
- * ... and how to configure and program the ZEPRROM module.

Why talking about this old tech ?

- * Because ...

- * the ZVC Replica is now available at a reasonable cost. *(50€ or 80€)*

(The original ZVC is extremely rare and asking prices can go higher than 1K USD.)

- * ZEPR0M modules are available on eBay.

(They often disguised themselves as surveying modules.)

- * of the user code bank switching.

(Provided by the Programmer ROM.)

- * of the persistent switched banks.

(Unique feature of the ZEPR0M module.)

- * of its low energy consumption.

(Sleep: 3 μ A / Run: 2.5 mA.)

Thanks to ...

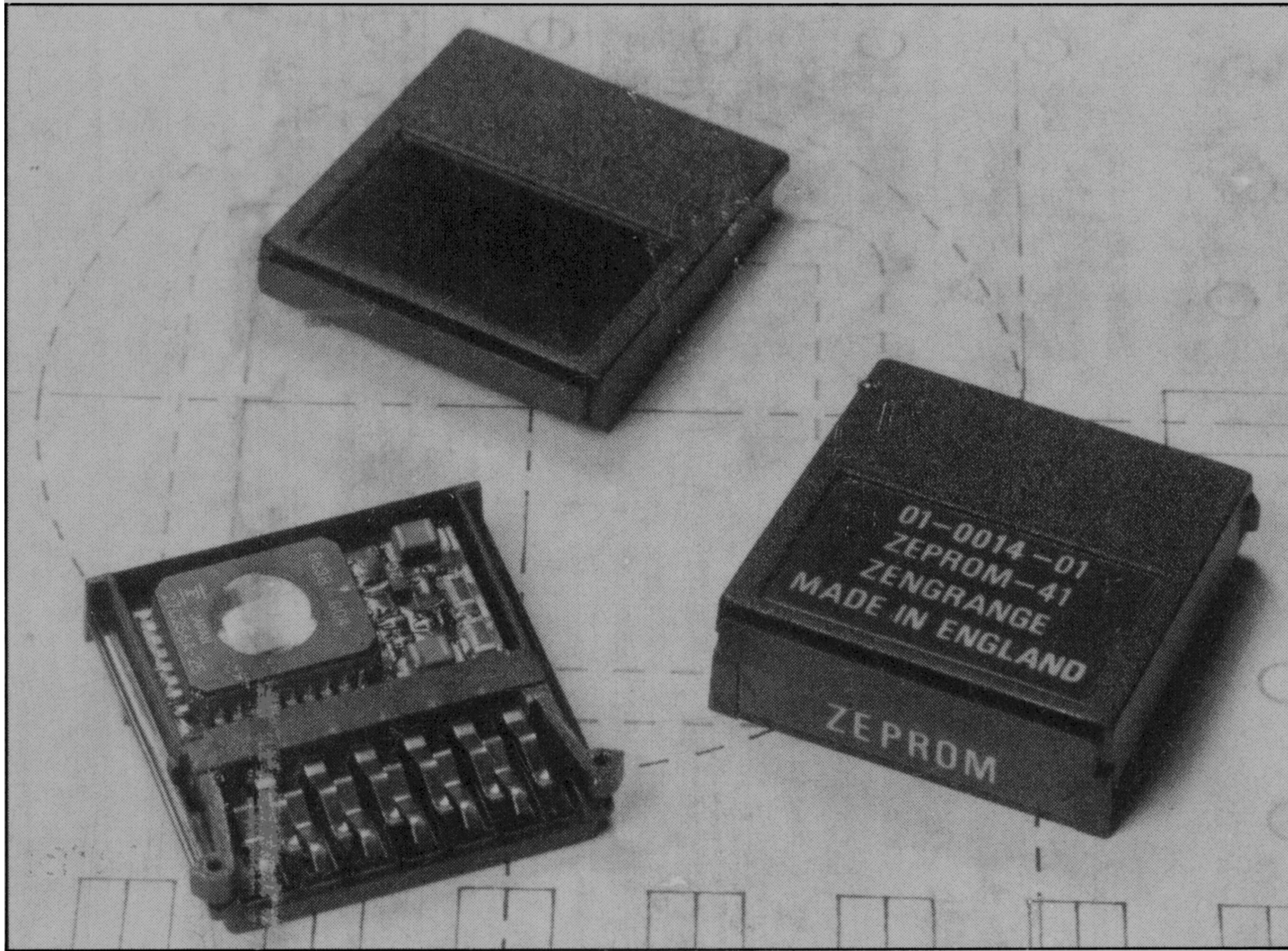
- * Diego Díaz for making the ZVC Replica and for his support.

ZEPROM

**THE ZENGRANGE ERASABLE PROGRAMMABLE READ ONLY MEMORY MODULE
FOR THE HP-41 HANDHELD CALCULATOR**

HHC 2024

HP-41 Module



Overview

- * A ZEPRROM is a 16K 10-bit EPROM in a HP-41 module case.
- * The ZEPRROM module was introduced in 1987 with a SRP of \$135.00 USD.
- * A ZEPRROM can be configured as a →
 - * straight 16K module.
 - * bank-switched 16K module.
 - * bank-switched 12K module.
 - * straight 8K module.

Straight 16K Configuration

Module Port	4K Page	ZEPROM in port 1	ZEPROM in port 2	ZEPROM in port 3	ZEPROM in port 4
4	F			Core 3	Core 3
	E			Core 2	Core 2
3	D			Core 1	Core 1
	C			Core 0	Core 0
2	B	Core 3	Core 3		
	A	Core 2	Core 2		
1	9	Core 1	Core 1		
	8	Core 0	Core 0		

Bank-Switched 16K Configuration

Module Port	4K Page	ZEPROM in port 1		ZEPROM in port 2		ZEPROM in port 3		ZEPROM in port 4	
		Bank 1	Bank 2	Bank 1	Bank 2	Bank 1	Bank 2	Bank 1	Bank 2
4	F							Core 1	Core 3
	E							Core 0	Core 2
3	D					Core 1	Core 3		
	C					Core 0	Core 2		
2	B			Core 1	Core 3				
	A			Core 0	Core 2				
1	9	Core 1	Core 3						
	8	Core 0	Core 2						

Bank-Switched 12K Configuration

Module Port	4K Page	ZEPROM in port 1		ZEPROM in port 2		ZEPROM in port 3		ZEPROM in port 4	
		Bank 1	Bank 2	Bank 1	Bank 2	Bank 1	Bank 2	Bank 1	Bank 2
4	F							Core 1	Core 3
	E							Core 0	Core 2
3	D					Core 1	Core 3		
	C					Core 0	Core 2		
2	B			Core 1	Core 3				
	A			Core 0	Core 2				
1	9	Core 1	Core 3						
	8	Core 0	Core 2						

Core 2 is a clone of Core 0



ZEPROM Programmer's Manual (p.8-9)

Straight 8K Configuration

Module Port	4K Page	ZEPROM in port 1		ZEPROM in port 2		ZEPROM in port 3		ZEPROM in port 4	
		Bank 1	Bank 2	Bank 1	Bank 2	Bank 1	Bank 2	Bank 1	Bank 2
4	F							Core 1	Core 3
	E							Core 0	Core 2
3	D					Core 1	Core 3		
	C					Core 0	Core 2		
2	B			Core 1	Core 3				
	A			Core 0	Core 2				
1	9	Core 1	Core 3						
	8	Core 0	Core 2						

Core 2 is a clone of Core 0



ZEPROM Programmer's Manual (p.7-8)

Core 3 is a near clone of Core 1

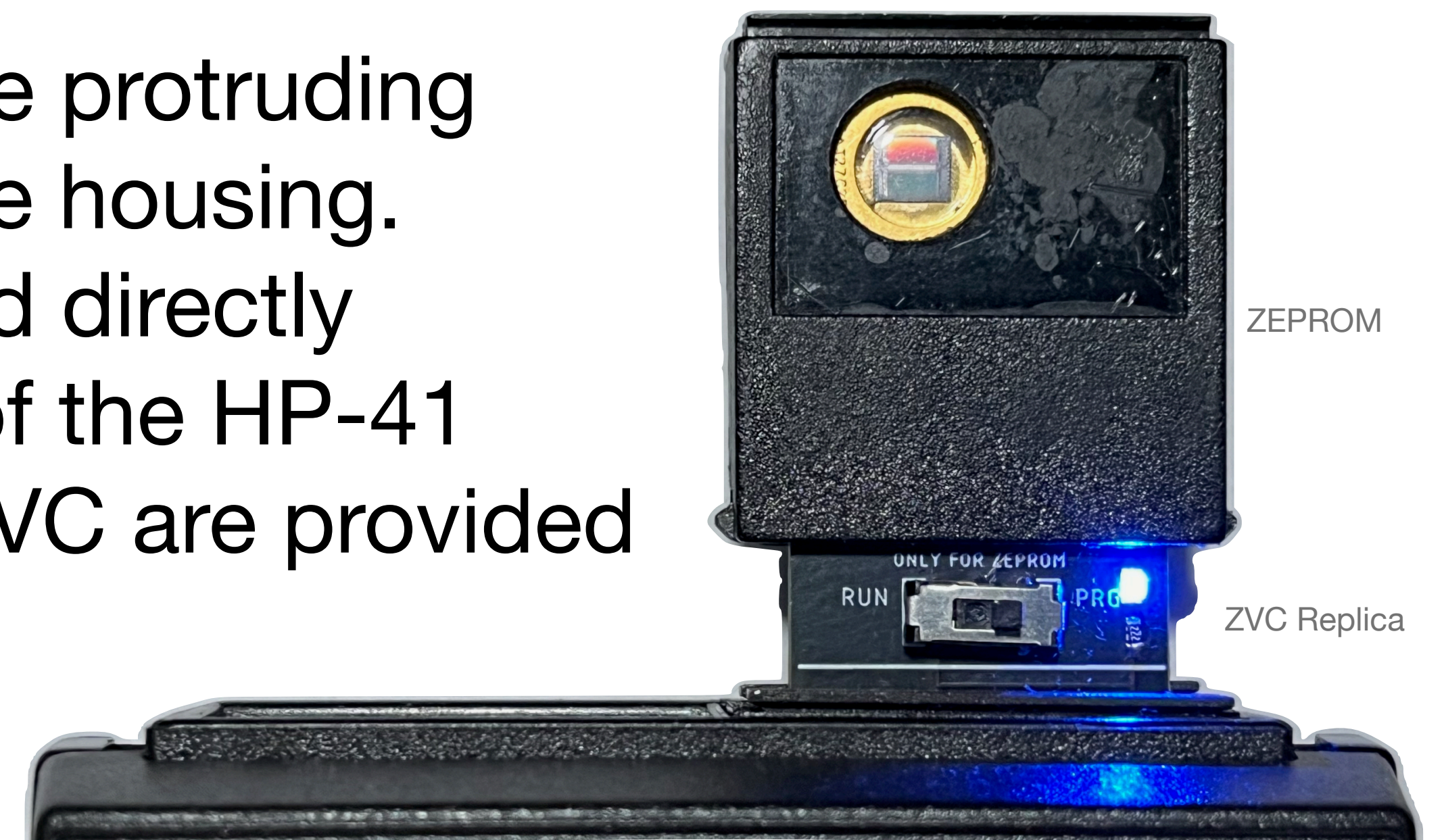
ZVC & ZVC REPLICA

THE ZEPROM VOLTAGE CONVERTER
IS A LOW COST ZEPROM PROGRAMMING MODULE
FOR THE HP-41 HANDHELD CALCULATOR



ZVC Overview

- * The ZEPROM Voltage Converter (ZVC) module has been designed as a companion product to the ZEPROM module and provides the user with an inexpensive, simple-to-use means of burning (programming) ZEPROMs when used with software such as the PROGRAMMER ROM.
- * The ZVC is a single unit, bare board device protruding from the back of a standard HP-41 module housing. This housing allows the ZVC to be plugged directly into one of the four I/O ports at the back of the HP-41 and the contacts at the other end of the ZVC are provided for plugging the ZEPROM module onto.

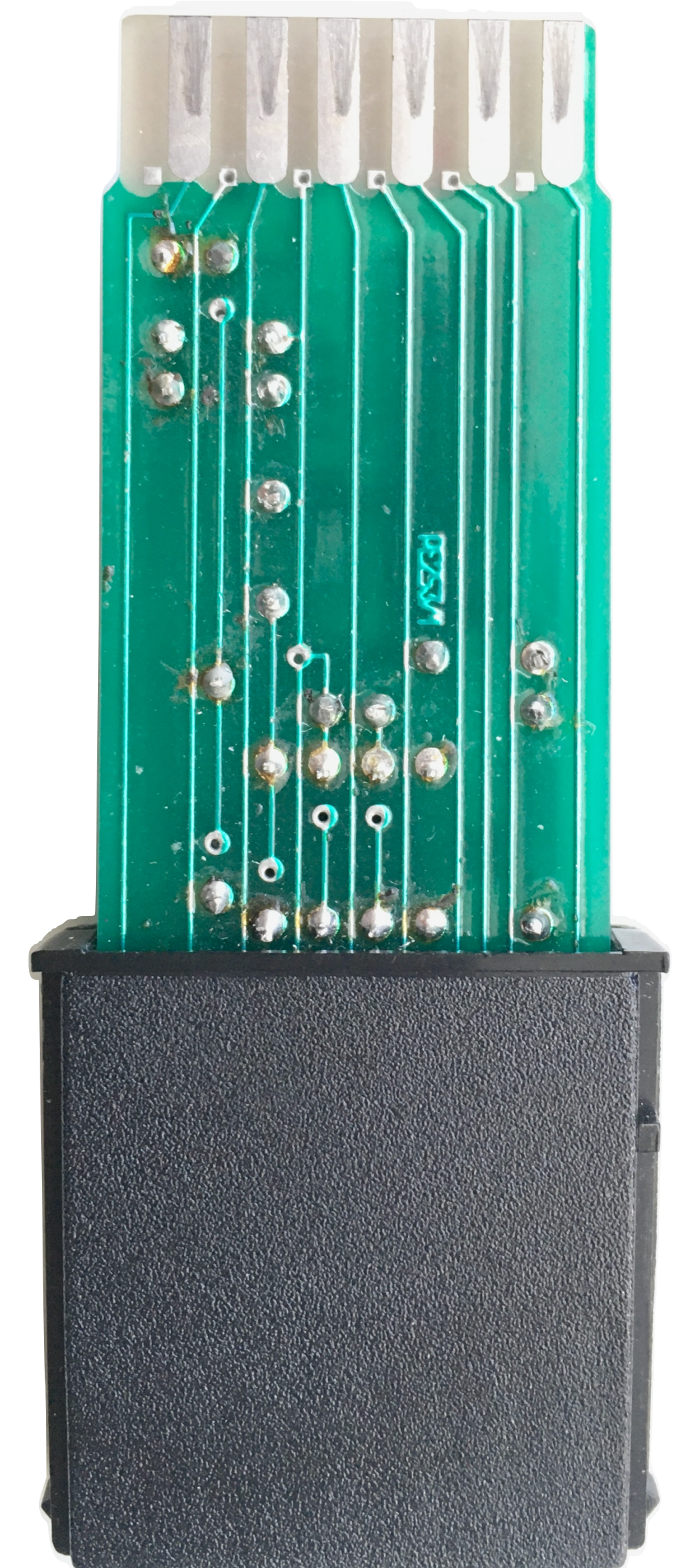
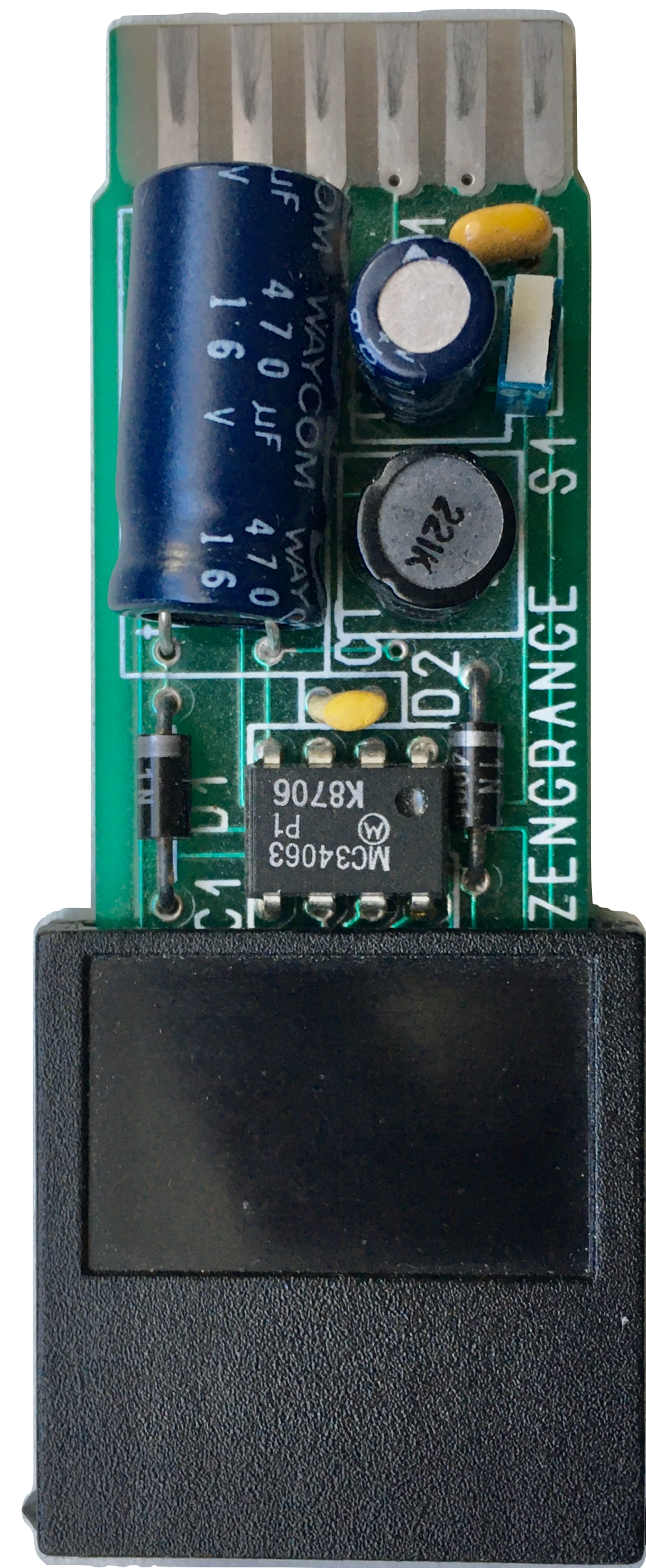


ZVC Module

- * The ZVC module was introduced in 1987 with a SRP of \$59.95 USD.



OFF → switch up
→ run / bypass
→ 6v
ON → switch down
→ programming
→ 12.5v



ZVC Replica Module



- OFF → switch left
→ RUN / bypass
→ 6v
→ LED off
-
- ON → switch right
→ PRG / programming
→ 12.5v
→ LED on

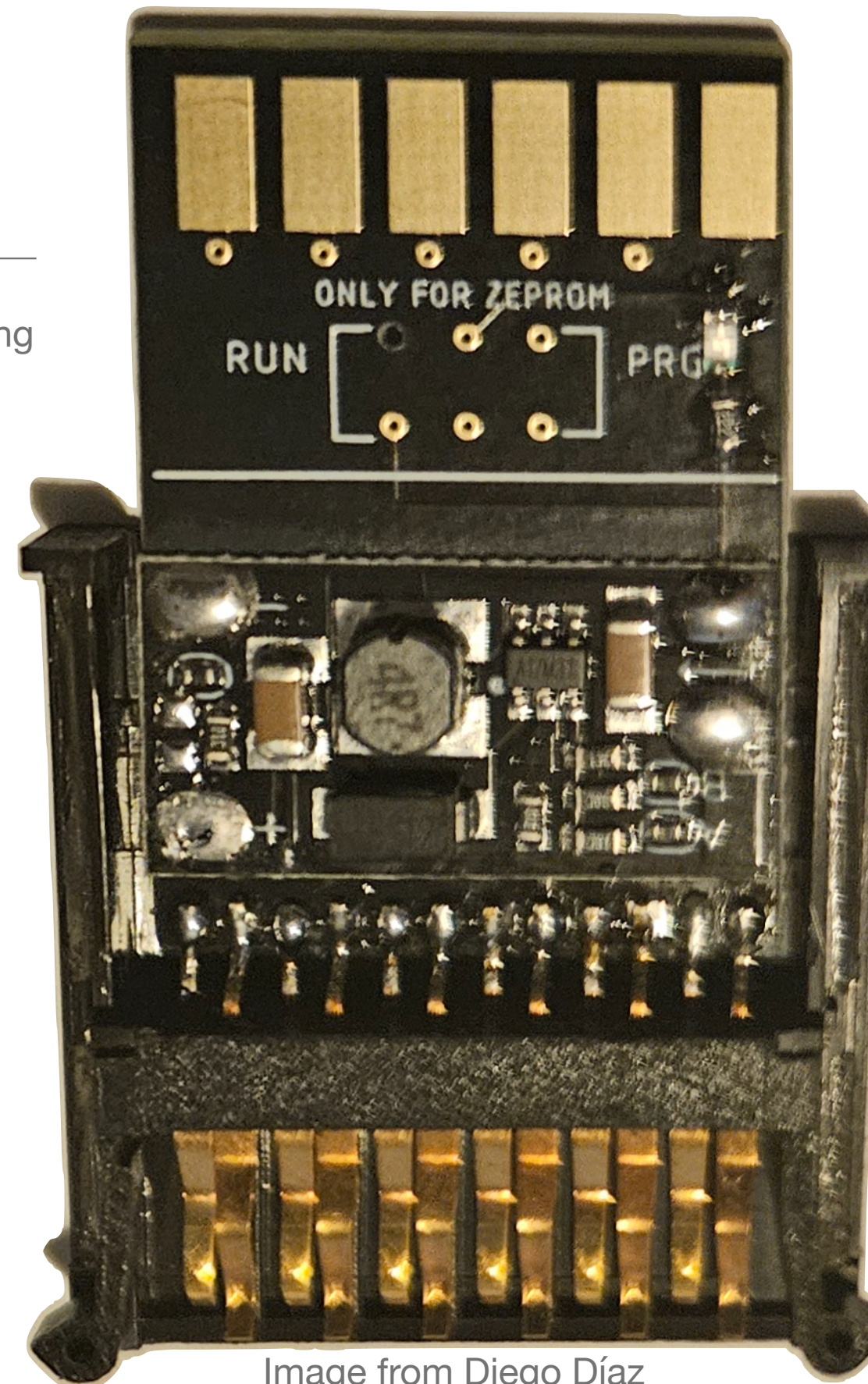


Image from Diego Díaz



ZEPROM Programming

- * The ZEPROM module has a custom chip interfacing the EPROM to the HP-41, it also has a mini-programmer to allow programming by a low-powered, low-cost device (ex.: ZVC, ZVC Replica, etc.) using suitable programming software (ex.: PROGRAMMER ROM, etc.).
- * Because EPROM type devices require a longer write cycle, a special algorithm is needed to ensure that they stay programmed once the programming voltage is removed.

PROGRAMMER ROM Programming Algorithm

```
reset retry counter
DO
  load address and data
  start write cycle (WMLDL)
  DO
    // empty loop
  UNTIL
    peripheral flag 4 clear
  END
  increment retry counter
  read data from address
UNTIL
  data read = data written
OR
  retry counter = 32
END

IF
  data read = data written
THEN
  display burn successful
ELSE
  display burn failure
END
```


ZEPROM Erasing

- * ZEPROM erasure is achieved via a window in the top half of the casing above the EPROM chip by exposing it to an ultraviolet (UV) light source.
- * A dosage of 15W-seconds/cm² is required to completely erase the module.
- * This dosage can be obtained by exposure to a UV lamp of wavelength 2537 Angstroms (Å) with an intensity of 12000 μW/m² for 15 to 20 minutes.



PROGRAMMER ROM



Programmer ROM Introduction

- * The Programmer ROM software has been designed and especially written to provide complete control over the burning of both user code programs and m-code functions into a ZEPROM.
- * Originally, ZEPROM modules were delivered as follow:
 - * Straight 16K configuration.
 - * Cores 0, 1, & 2 erased.
 - * Core 3 burned with the Programmer ROM.
- * The Programmer ROM was never made available as a standalone module.

Programmer ROM Features

- * Compiling and loading user code (RPN/FOCAL) to MLDL/ZEPRROM.
- * 4K page initialization and checksum calculation.
- * 4K page switching support for machine code and for user code.
- * Copy 4k page from/to MLDL/RAM-Box to/from ZEPRROM.
- * Copy 4k page from/to MLDL/RAM-Box/ZEPRROM to/from IL storage device.

Programmer ROM Functions

XROM FUNCTION DESCRIPTION

9, 0	-PROGRAMMER	Programmer ROM header
9, 1	ADDBSW	Add bank-switching code to initialized page
9, 2	ADDMCF	Add M-code bank-switching functions
9, 3	ADDUCF	Add user code bank-switching functions
9, 4	BLANK?	Test if the page in Reg X is blank
9, 5	BNKSW?	Test if page in Reg X is bank-switched
9, 6	BURNUC	Burn user code program from RAM or ROM
9, 7	BURNWD	Burn word in Reg X into address in Reg Y
9, 8	CHKSUM	Compute & return checksum for specified page
9, 9	COMPUC	Compile user code program in main memory
9,10	COPYPG	Copy entire 4K image between pages X & Y
9,11	CPXYZ	Copy data as specified in Regs X,Y,Z
9,12	DECHEX	Convert decimal value in Reg X into hexadecimal
9,13	ENABLEP	Enable primary bank of ROM page in Reg X
9,14	ENABLES	Enable secondary bank of ROM page in Reg X

XROM FUNCTION DESCRIPTION

9,15	FRSPC?	Address & amount of free space of page in Reg X
9,16	HEXDEC	Convert hexadecimal string into decimal number
9,17	ILBURN	Burn page using image from IL mass storage file
9,18	ILSAVE	Save page to HP-IL mass storage file
9,19	'INIT	Initialize page, interactive program
9,20	'INITP	Initialize page, interactive sub-program
9,21	INITPG	Initialize a ZEPROM Page
9,22	PB09	Enable primary bank of Programmer ROM module
9,23	PGX=Y?	Compare two 4K ROM pages for equality
9,24	PRGMLN	Return program length and number of LBLs
9,25	READWD	Read word from ROM page in RegX
9,26	REBURN	Recover from error in burning user code program
9,27	RRBURN	Burn ZENROM ROMREG + format data into ZEPROM
9,28	SB09	Enable secondary bank of Programmer ROM module
9,29	'UCBURN	User code burning, interactive program

SETUP



Hardware

- * 41C/CV/CX/CL calculator.

- * ZEPROM module(s).

- * ZVC or ZVC Replica module.

- * Programmer ROM module.

Originally, all ZEPROMs had the programmer ROM burned into its core 3, so when the time came to burn a ZEPROM, we would simply plug an erased ZEPROM module into the ZVC module and plug a default configured ZEPROM into port 3 to get the Programmer ROM. Today, it is almost impossible to find a default configured ZEPROM, so the Programmer ROM image must be loaded into an extra EPROM/Flash/Q-ROM device, like a: EPROM box/module, Clonix-D module, Q-ROM box/module.

- * HP-IL module.

- * PIL-Box IL/USB interface

- * EPROM eraser.

- * 10-bit RAM Q-ROM device.

*MLDL/QROM box, W&W RAMBox, MAXX module, 41CL (built-in), etc.
Size: 8K for ROM creation, 16K for ROM creation and testing*

- * Computer.

Type: PC, Mac, Raspberry Pi, etc.

Processor: Intel or ARM.

Operating system: Windows, GNU/Linux or macOS.

Software

* Package manager for GNU utilities

- * Windows Cygwin (w/POSIX)
- * macOS Homebrew
- * GNU/Linux built-in

❄ Software *version*

- * lifutils 1.7.12
- * python3.xx 3.11
- * python3.xx-venv 3.11
- * pyILPER 1.8.8
- * perl 5.36

* *GNU utilities*

- * *xxd* *hex dump/upd. file*
- * *unzip* *extract zip file*
- * *wget* *web file getter*

Configuration

- * python venv setup *~/py311*

Directories & Files

- * Start a command-line shell.

- * Windows → Cygwin → bash

- * macOS → Terminal → zsh

- * GNU/Linux → Terminal → bash

- * Build directory structure.

```
mkdir -p ~/Z41/{web,roms,src,temp,pyilper}
```

```
~ (home dir.)
└─ Z41
    ├── web
    ├── roms
    ├── src
    ├── temp
    └── pyilper
```

- * Download ROMs and manuals.

```
cd ~/Z41/web
wget systemyde.com/zip/rom_files_240731.zip
wget systemyde.com/pdf/mem_ref.pdf
wget literature.hpcalc.org/community/hp41-zeprom.pdf
wget literature.hpcalc.org/community/zenrom.pdf
```

- * Extract ROM files.

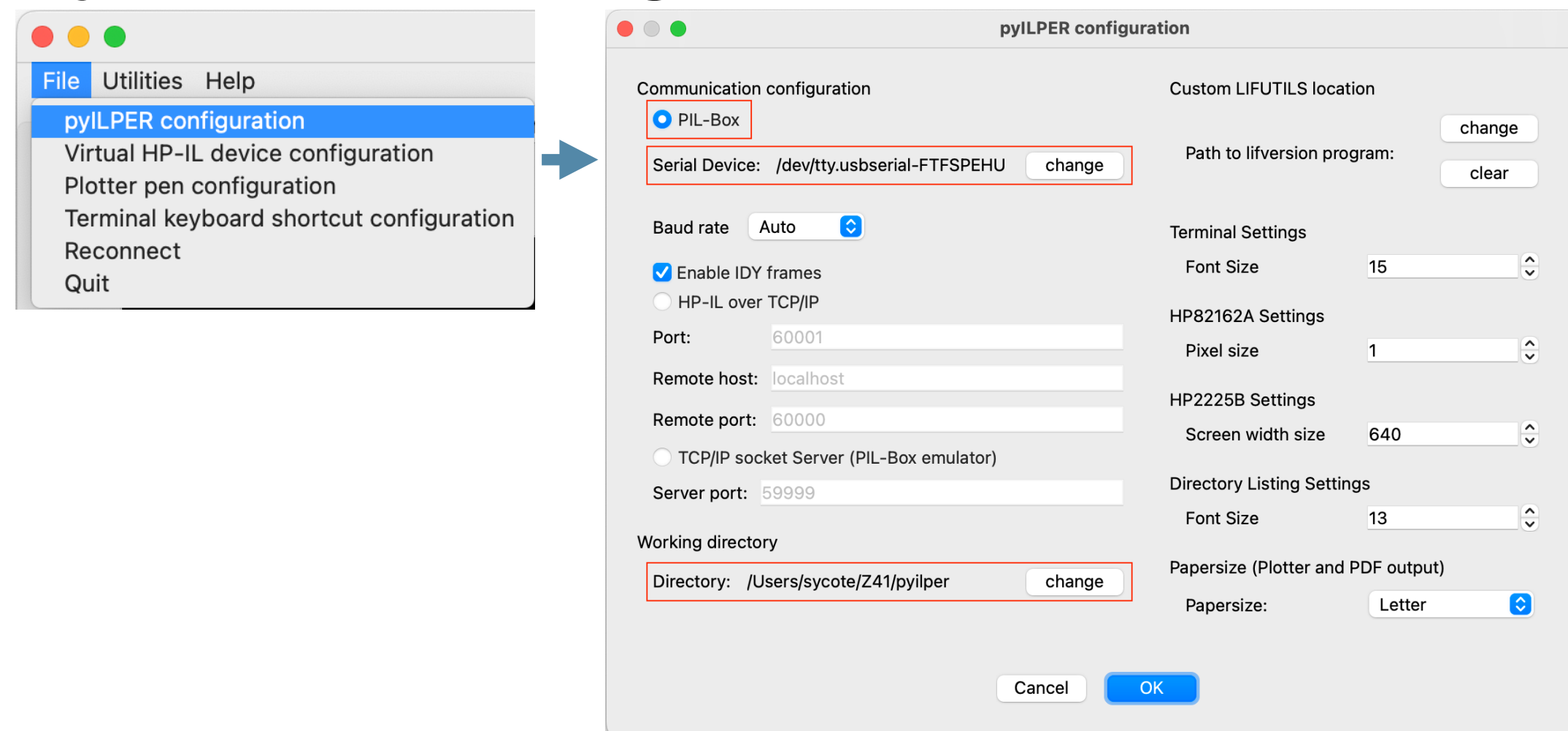
```
cd ~/Z41/roms
unzip ../web/rom_files_240731.zip "*.ROM"
```


pyILPER Configuration

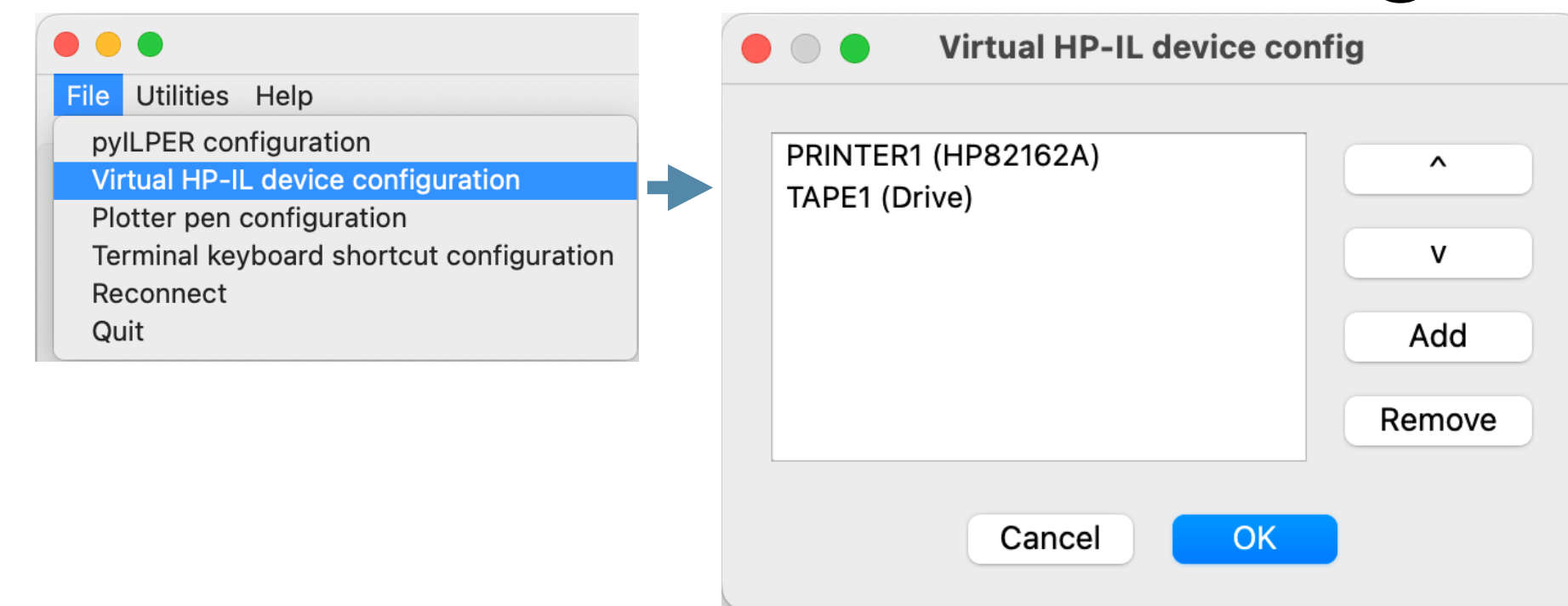
- * Assuming PIL-Box is connected.
- * Start pyILPER with custom. config.

```
cd ~/Z41/pyilper          # go to data folder
source ~/py311/bin/activate # adjust as needed
sudo chmod a+rw /dev/ttyUSB0 # linux-only
pyilper --clean --instance Z41 & # setup new instance
```

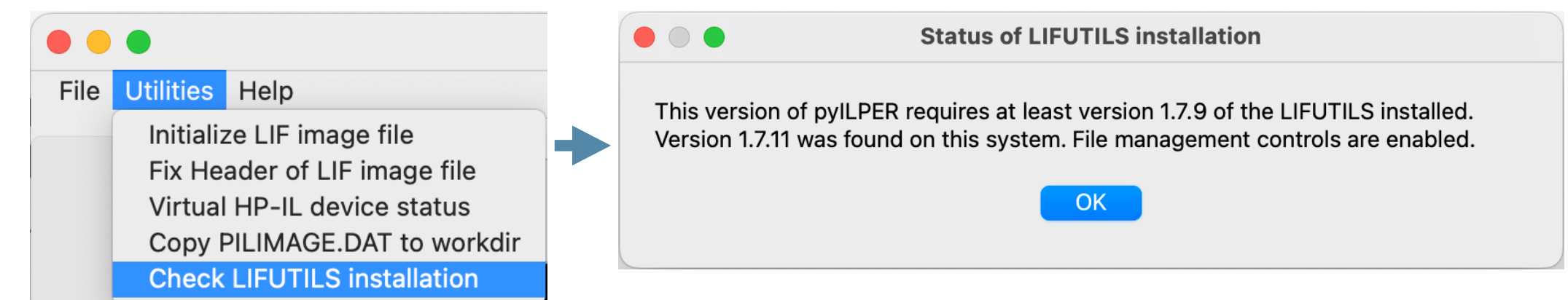
- * pyILPER configuration.



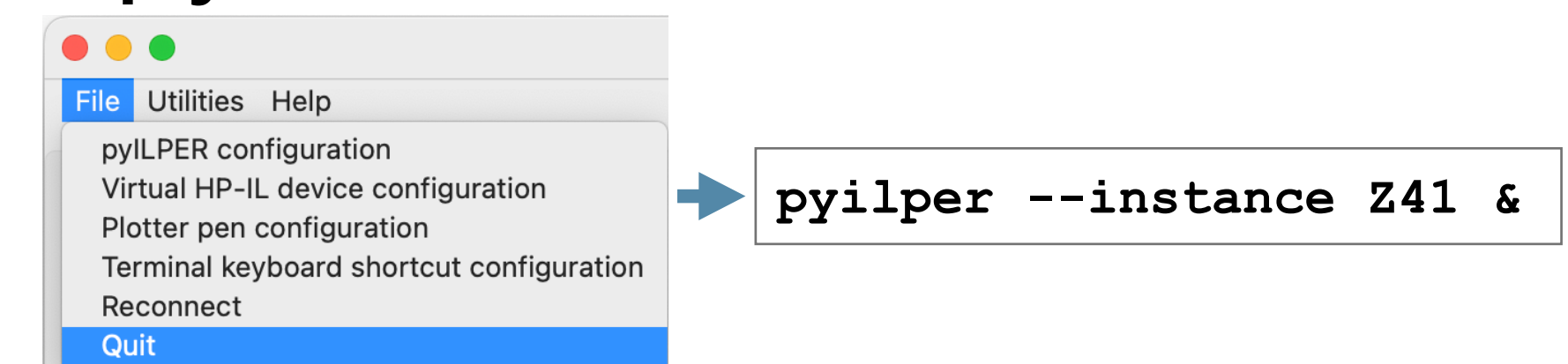
- * Virtual HP-IL devices configuration.



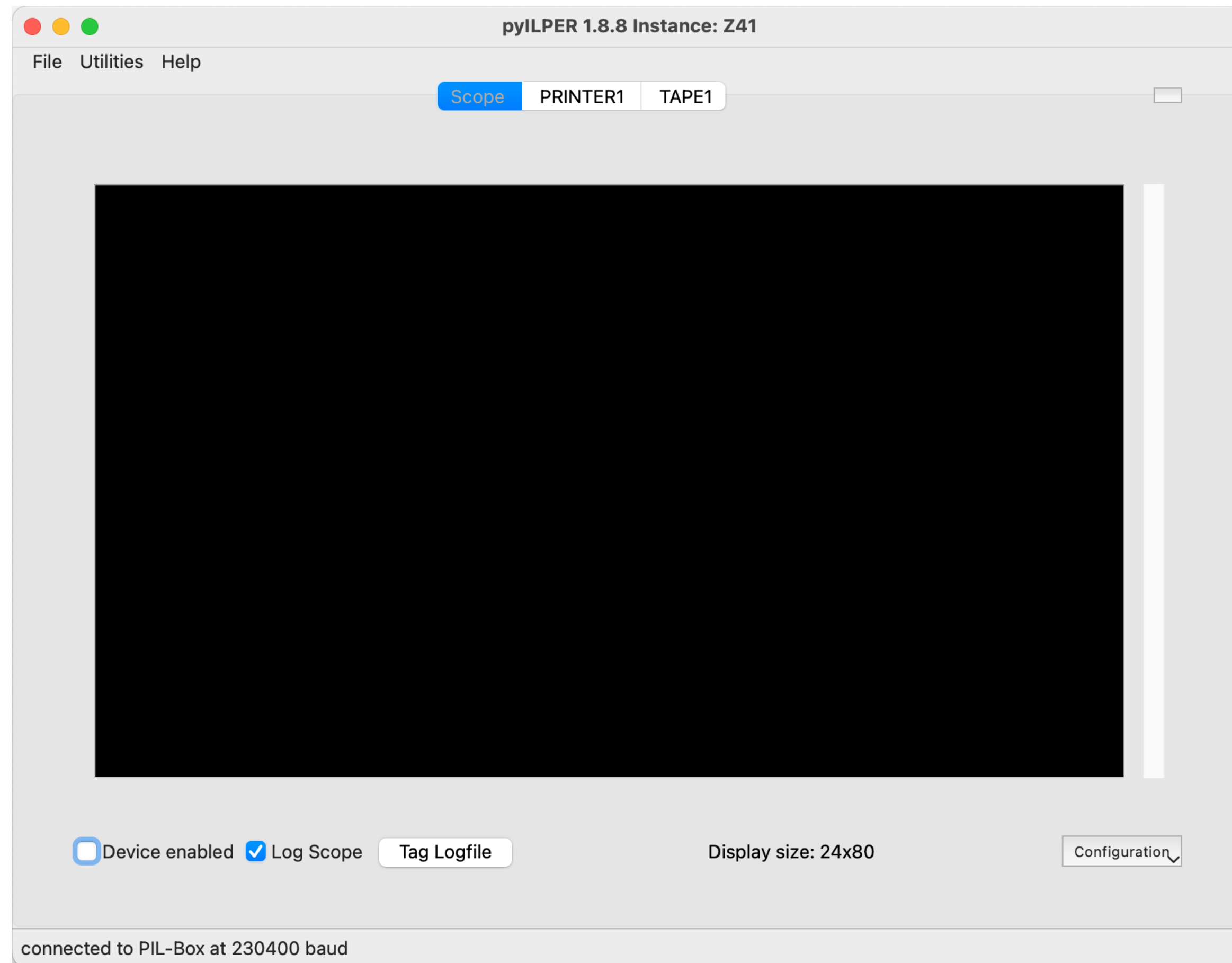
- * LIFUTILS installation status.



- * Restart pyILPER.

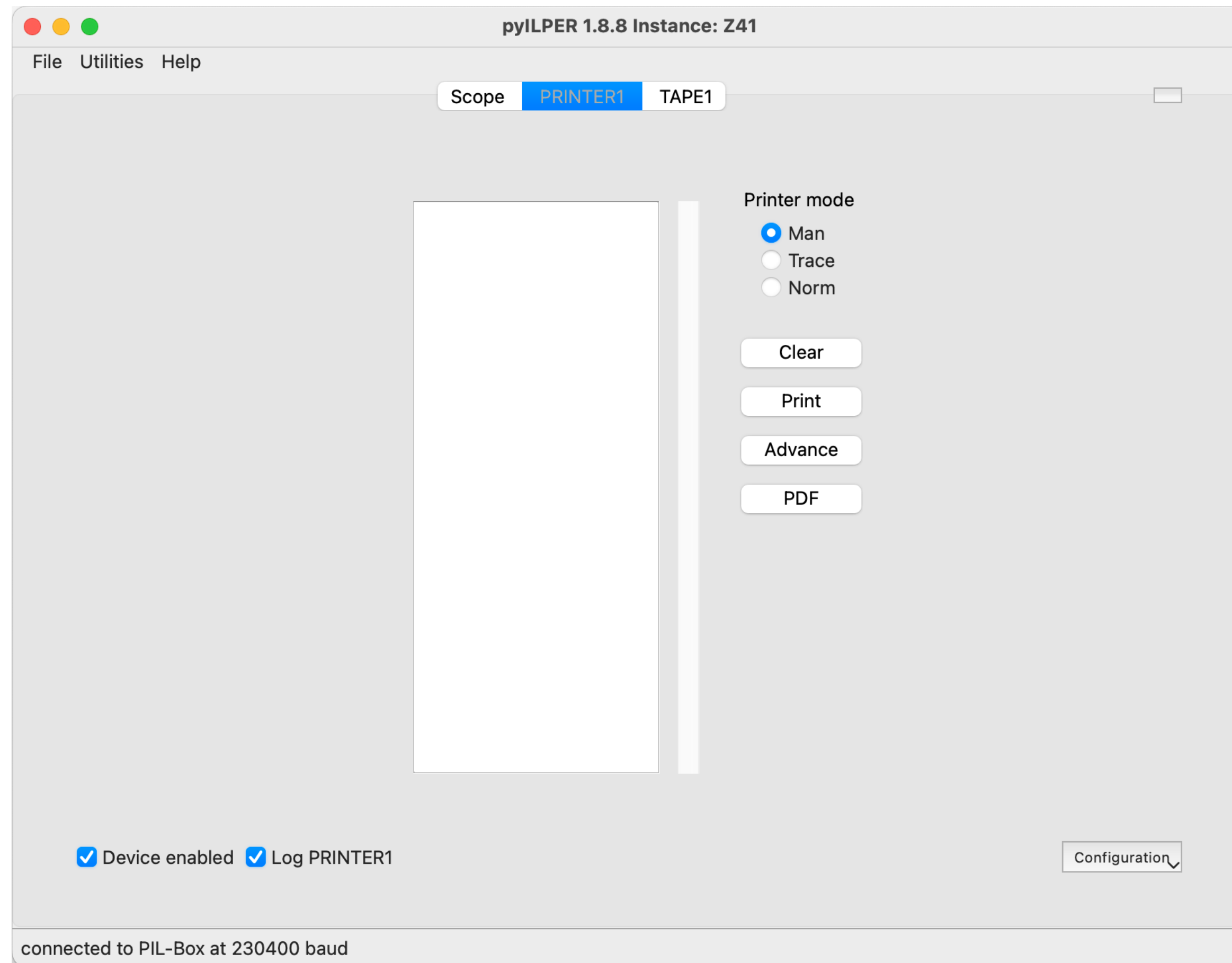


pyILPER Scope Setup



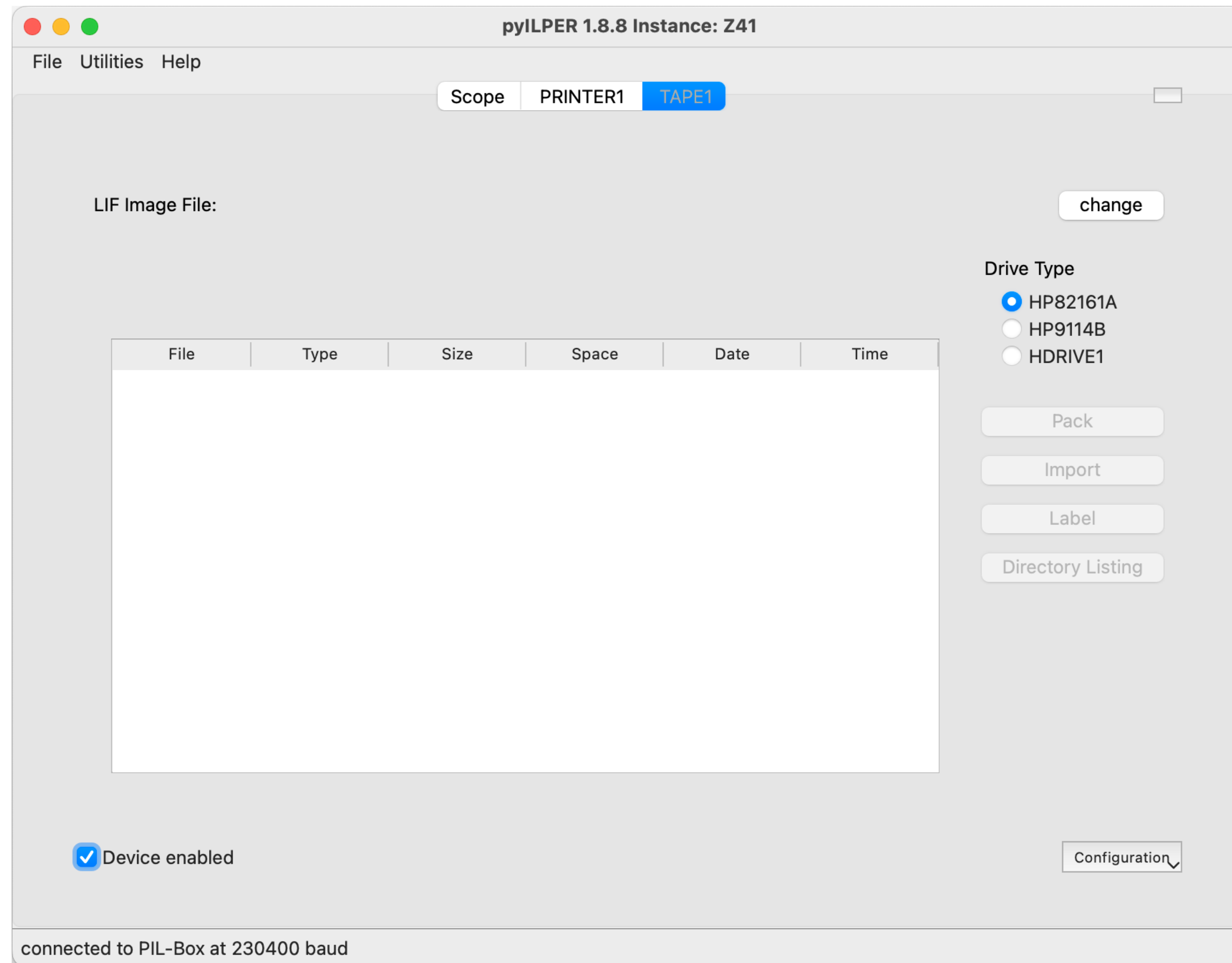
- * Select **Scope** tab.
- * Check **Log Scope** check-box.
- * Uncheck **Device enabled** check-box.
- * **Scope** does not have an interface loop address.

pyILPER PRINTER1 Setup



- * Select **PRINTER1** tab.
- * Set **Printer mode** radio-button to **Manual**.
- * Check **Log PRINTER1** check-box.
- * Check **Device enabled** check-box.
- * **PRINTER1** interface loop address is 1.

pyILPER TAPE1 Setup

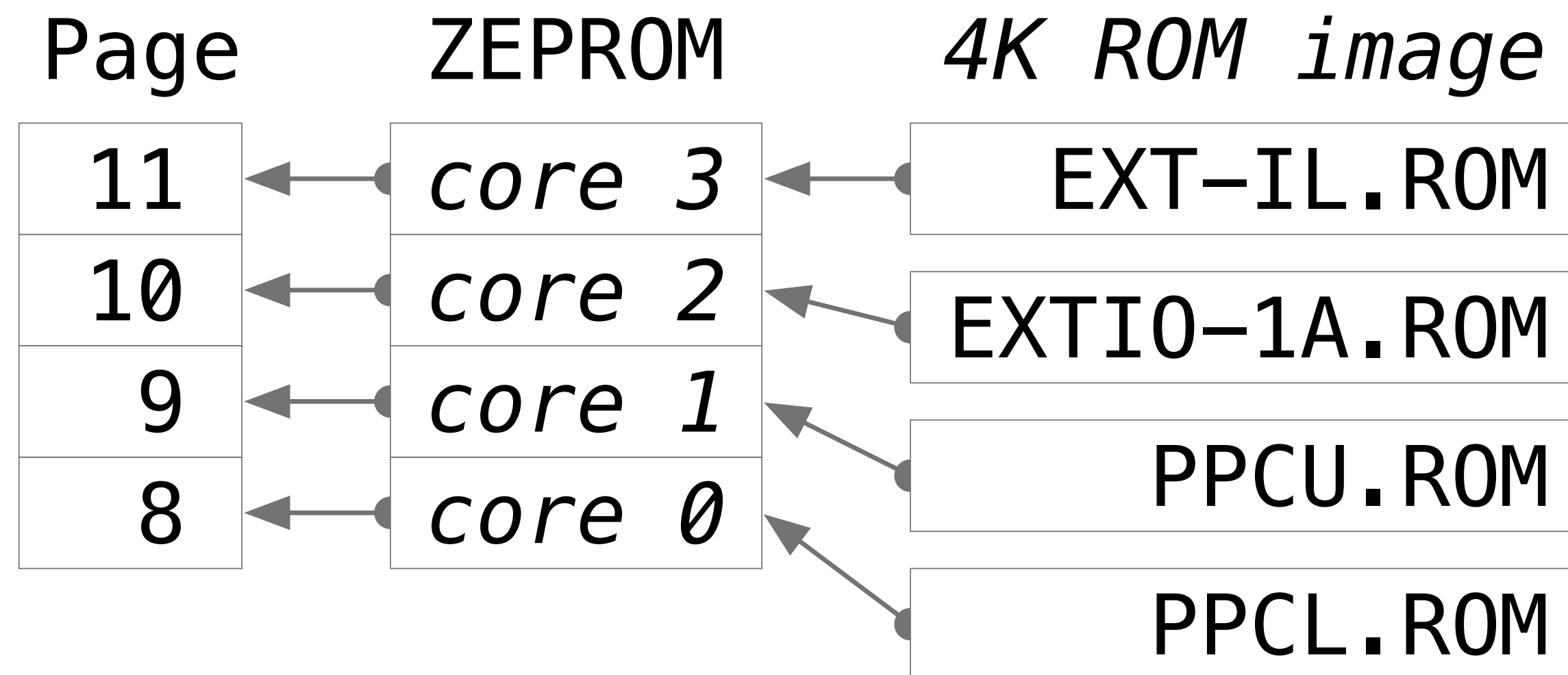


- * Select **TAPE1** tab.
- * Set **Drive Type** radio-button to **HP82161A**.
- * Check **Device enabled** check-box.
- * **TAPE1** interface loop address is 2.

FLAT 16K

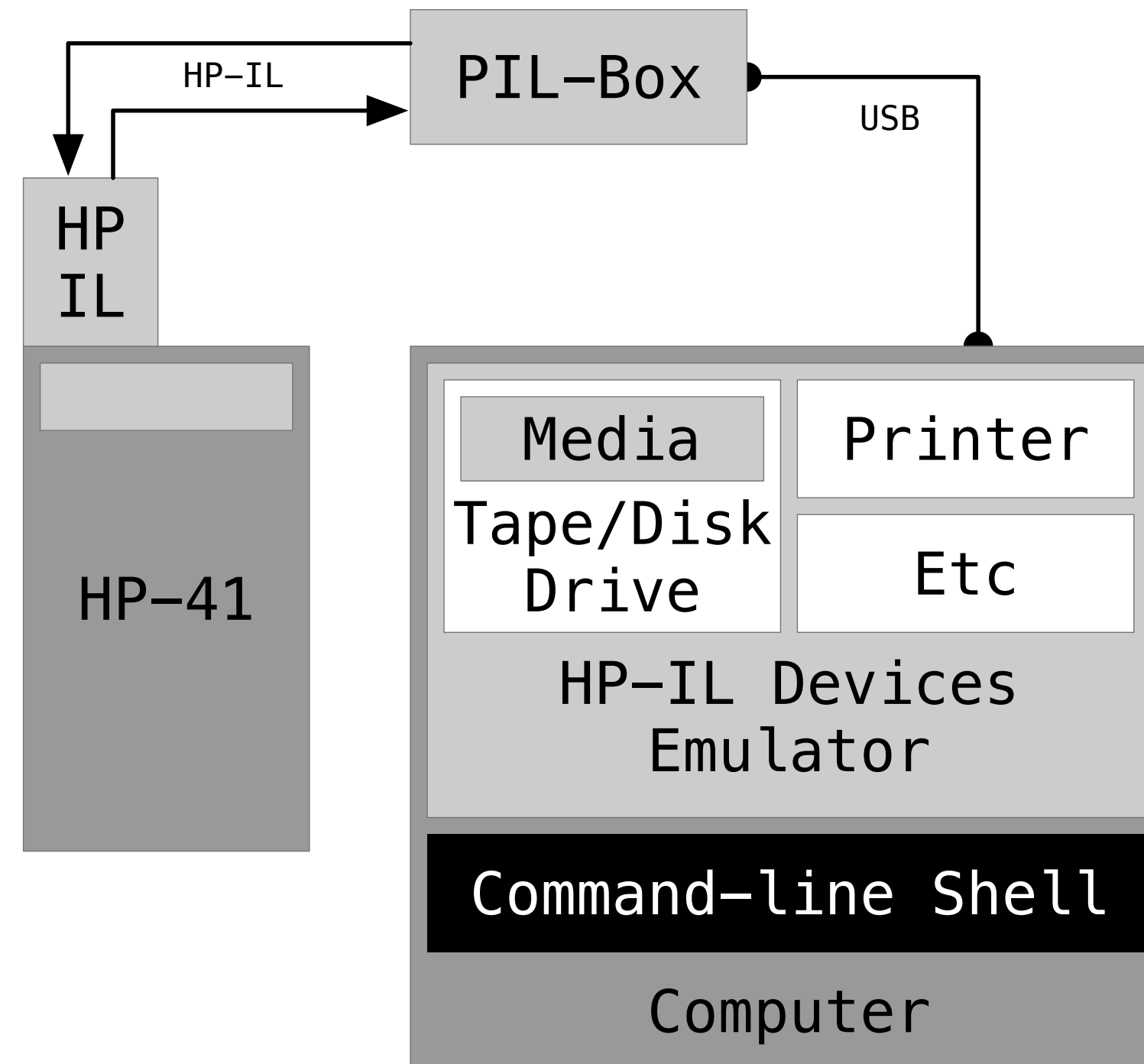
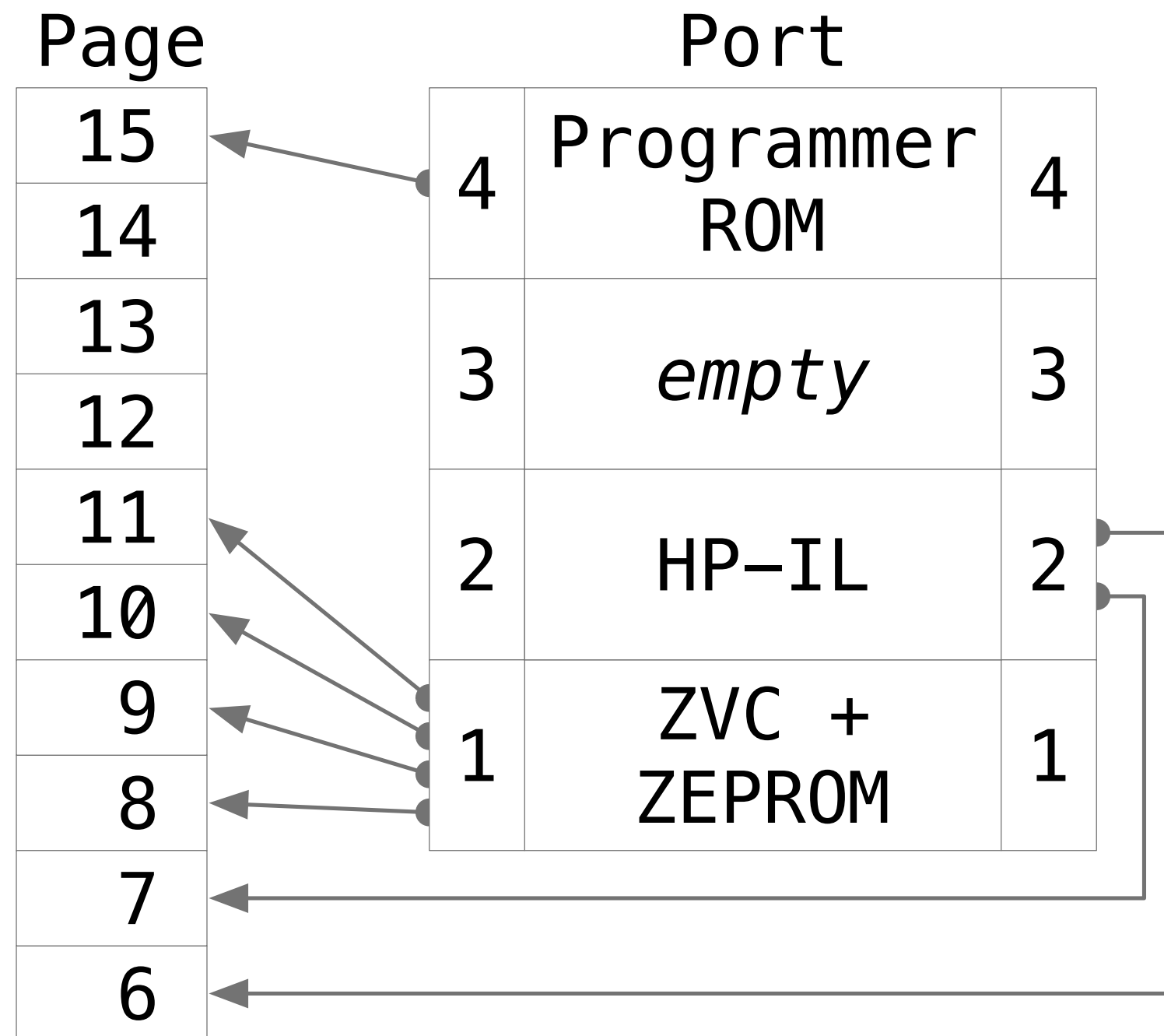
USING STANDARD ROM FILES

Summary



- * In this section, we will create a 16K flat ZEPROM module.
- * Once programmed, the module will contain the following ROM images:
 - * PPC ROM (lower & upper pages).
 - * HP Extended I/O ROM.
 - * Skwid Extended HP-IL ROM.

Setup



- * Erased ZEPR0M inserted in ZVC.
- * HP-41 connected to the PIL-Box.
- * PIL-Box connected to the computer.
- * pyILPER running Z41 instance.
- * Command-line Shell started.

Virtual Media Creation → Z41C1

* Go to workspace.

```
cd ~/Z41/temp
```

* Create LIF volume.

```
lifinit -m cass -z z41c1.dat 64  
liflabel z41c1.dat Z41C1
```

* Add ROM images to LIF volume.

```
cat ../roms/PPCL.ROM      | rom41er PPCL  | lifput z41c1.dat  
cat ../roms/PPCU.ROM      | rom41er PPCU  | lifput z41c1.dat  
cat ../roms/EXTIO-1A.ROM  | rom41er EXTIO | lifput z41c1.dat  
cat ../roms/EXT-IL.ROM    | rom41er EXTIL | lifput z41c1.dat
```

* Export LIF volume.

```
cp z41c1.dat ../pyilper
```

* Exit workspace.

```
cd
```

* Show LIF volume content.

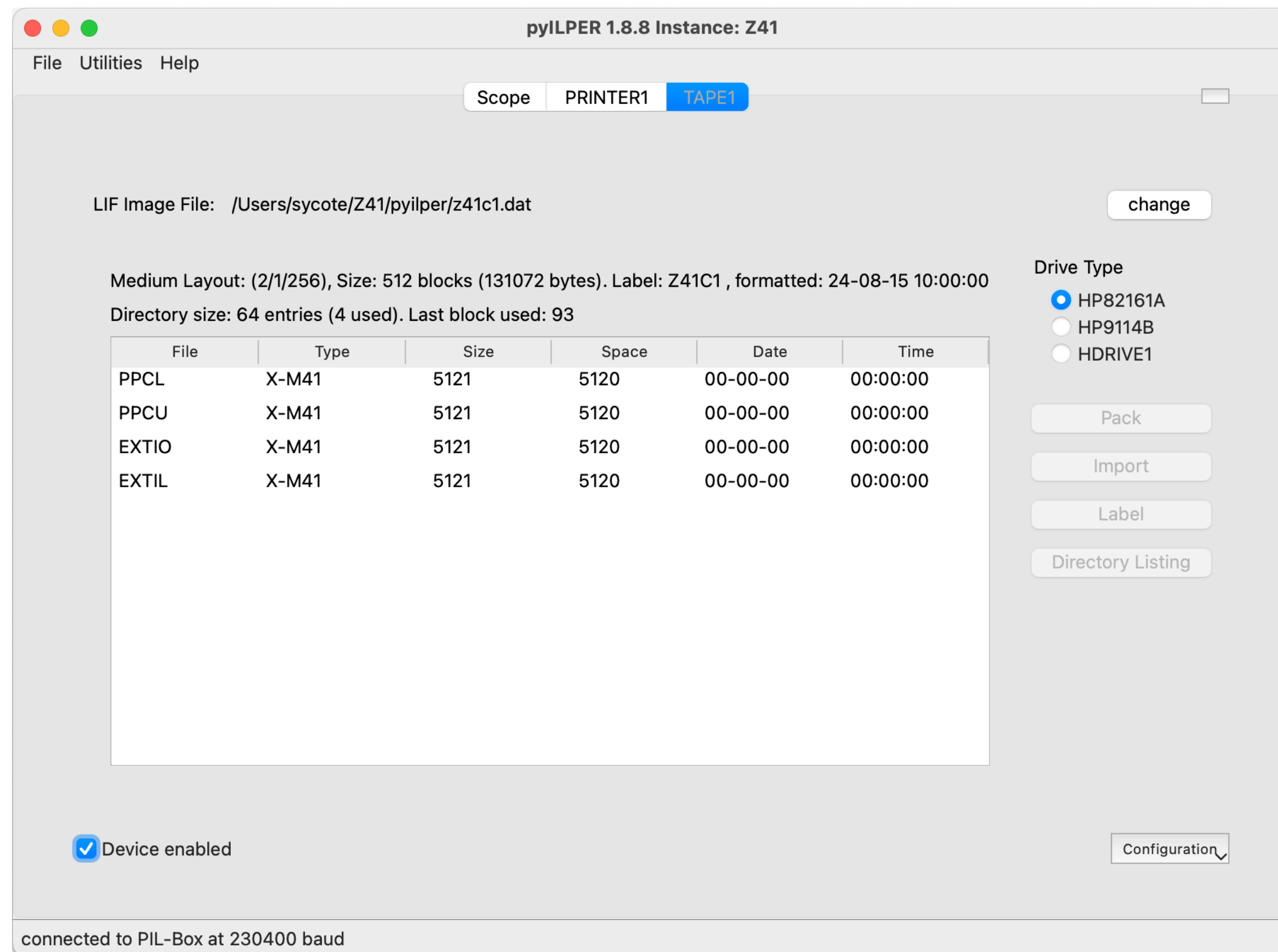
```
lifdir ~/Z41/pyilper/z41c1.dat
```

```
Volume : Z41C1 , formatted : 03/08/24 09:00:00  
Tracks: 2  
Surfaces: 1  
Blocks/Track: 256  
Total size: 512 Blocks, 131072 Bytes
```

PPCL	X-M41	5121/5120
PPCU	X-M41	5121/5120
EXTIO	X-M41	5121/5120
EXTIL	X-M41	5121/5120

```
4 files (64 max), last block used: 92 of 512
```


pyILPER TAPE1 → Z41C1



- * Select **TAPE1** tab.
- * Set **Drive Type** radio-button to **HP82161A**.
- * Click on **change** push-button.
- * In the **Select LIF Image File** dialog-box.
- * Go to **~/Z41/pyilper** directory.
- * Select **z41c1.dat** file.
- * Click on **Open** push-button.
- * Check **Device enabled** check-box.
- * **TAPE1** loop address is 2.

ZEPROM Programming



- * ZVC (Replica) switch set to **DOWN (PRG)**.

- * To burn ROM images into ZEPROM, do the following commands:

```
"PPCL"  
8  
ILBURN
```



```
"PPCU"  
9  
ILBURN
```



```
"EXTIO"  
10  
ILBURN
```



```
"EXTIL"  
11  
ILBURN
```

- * ZVC (Replica) switch set to **UP (RUN)**.

- * **OFF**

- * Remove all modules and insert the newly programmed ZEPROM in port 1.

- * **ON**

- * **CAT 2**



```
...  
C PPC 1981  
-X MASS 1A  
-X EXT FCN  
-X CTL FNS  
-ADV CTL FN  
EXT IL ROM  
PRINT FCNS  
...
```


BANK SWITCHING

Machine Code Instructions

- * Bank Switching Instructions:
 - * 100h : ENBANK1 : Enable Bank #1
 - * 180h : ENBANK2 : Enable Bank #2
 - * 140h : ENBANK3 : Enable Bank #3 : not supported by ZEPROM
 - * 1C0h : ENBANK4 : Enable Bank #4 : not supported by ZEPROM
- * Programming Instruction:
 - * 040h : WMLDL : Write to Machine Language Development Lab

Notes

- * Because the 410S and the NUT CPU was not designed to support bank switching, it is the responsibility of the attached 10-bit memory based device to implement the needed behavior. (RAM, E.E.P.ROM, Flash, etc.)
- * Bank switching and MLDL opcodes are seen as NOP (no operation) by the NUT processor.
- * ZEPROM bank switching is completely manual, once a bank has been selected it will stay selected until another bank is manually selected. Unlike some other bank switching devices, light sleep and/or deep sleep does not reset the active bank to the default one. (bank 1)

Bank-Switch Support

- * ZEPROM hardware scan the two most significant bits of core 1 address xFFD to decide if it is bank-switched or not.
- * If both bits are clear, then the module is considered non bank-switched. (1x 16K)
- * If either or both bits are set, then the module is considered as being bank-switched. (2x 8K)
- * The ZEPROM module is 100% code compatible with HP ROMs. This means that all user or M-code designed to run on an HP ROM will run on ZEPROM, bank-switched or not.



bit-9	bit-8	ZEPROM bank-switch bits (pFFD)
0	0	Page is not bank-switched
0	1	Bank-switched, primary bank
1	0	Bank-switched, indeterminate bank (HP)
1	1	Bank-switched, secondary bank

Bank-Switch Example

Port	Module	Active	Pg
------	--------	--------	----

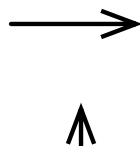
4	ZEPR0M #2 BS-16K	Core 1	15
		Core 0	14

3	ZENROM	ZENROM	13
			12

2	ZEPR0M #1 BS-16K	Core 1	11
		Core 0	10

1	Games		9
		Games	8

ENBANK2 executed from
within ZEPR0M #1 module



Port	Module	Active	Pg
------	--------	--------	----

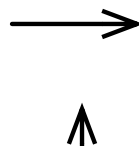
4	ZEPR0M #2 BS-16K	Core 1	15
		Core 0	14

3	ZENROM	ZENROM	13
			12

2	ZEPR0M #1 BS-16K	Core 3	11
		Core 2	10

1	Games		9
		Games	8

ENBANK1 executed from
within ZEPR0M #1 module



Port	Module	Active	Pg
------	--------	--------	----

4	ZEPR0M #2 BS-16K	Core 1	15
		Core 0	14

3	ZENROM	ZENROM	13
			12

2	ZEPR0M #1 BS-16K	Core 1	11
		Core 0	10

1	Games		9
		Games	8

bank
switch
request
ignored

no banks

bank
switch
occurs

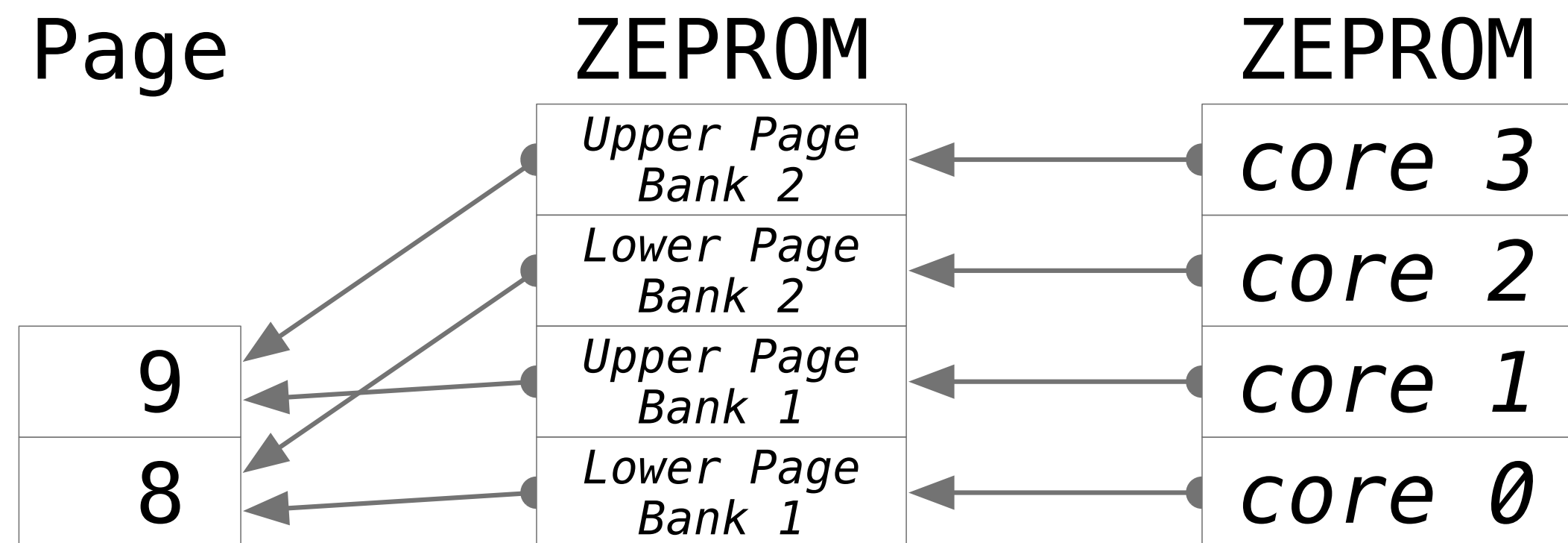
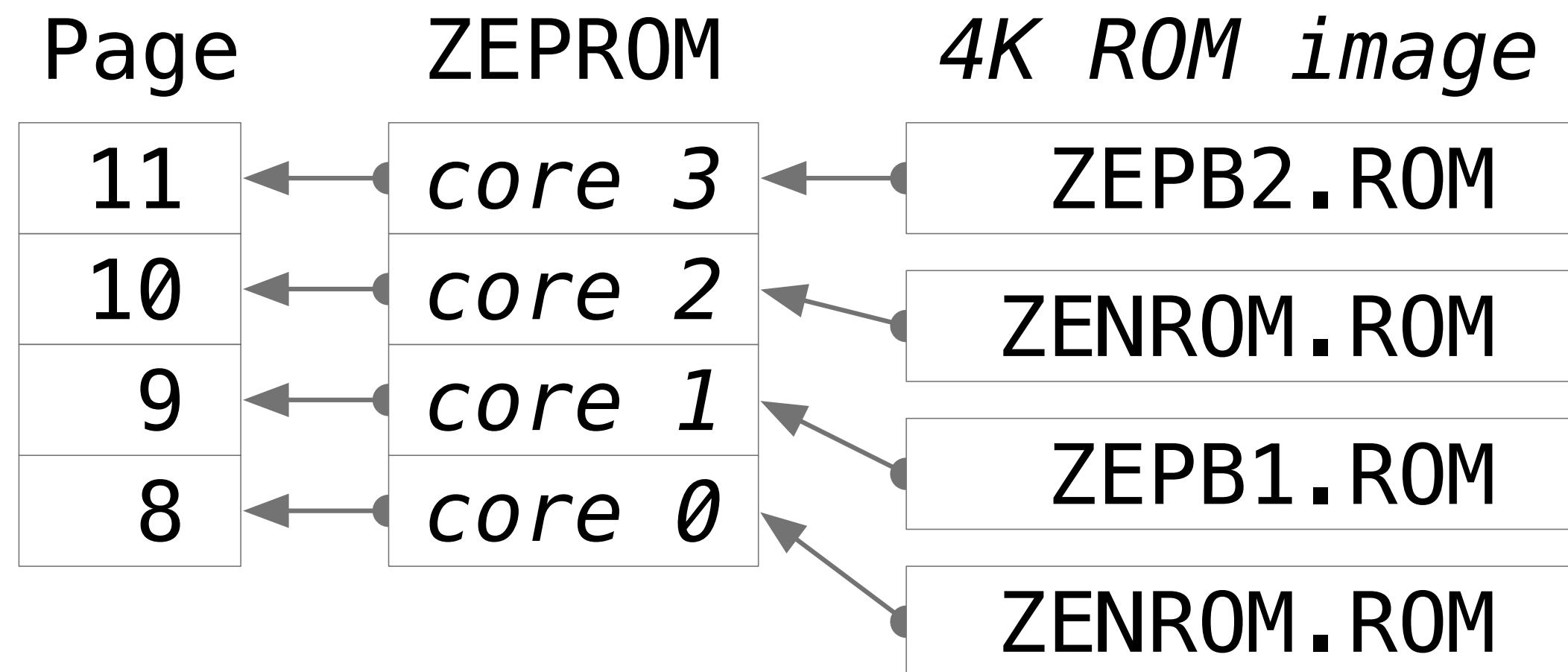
no banks

FLAT 8K

USING SLIGHTLY MODIFIED ROM FILES

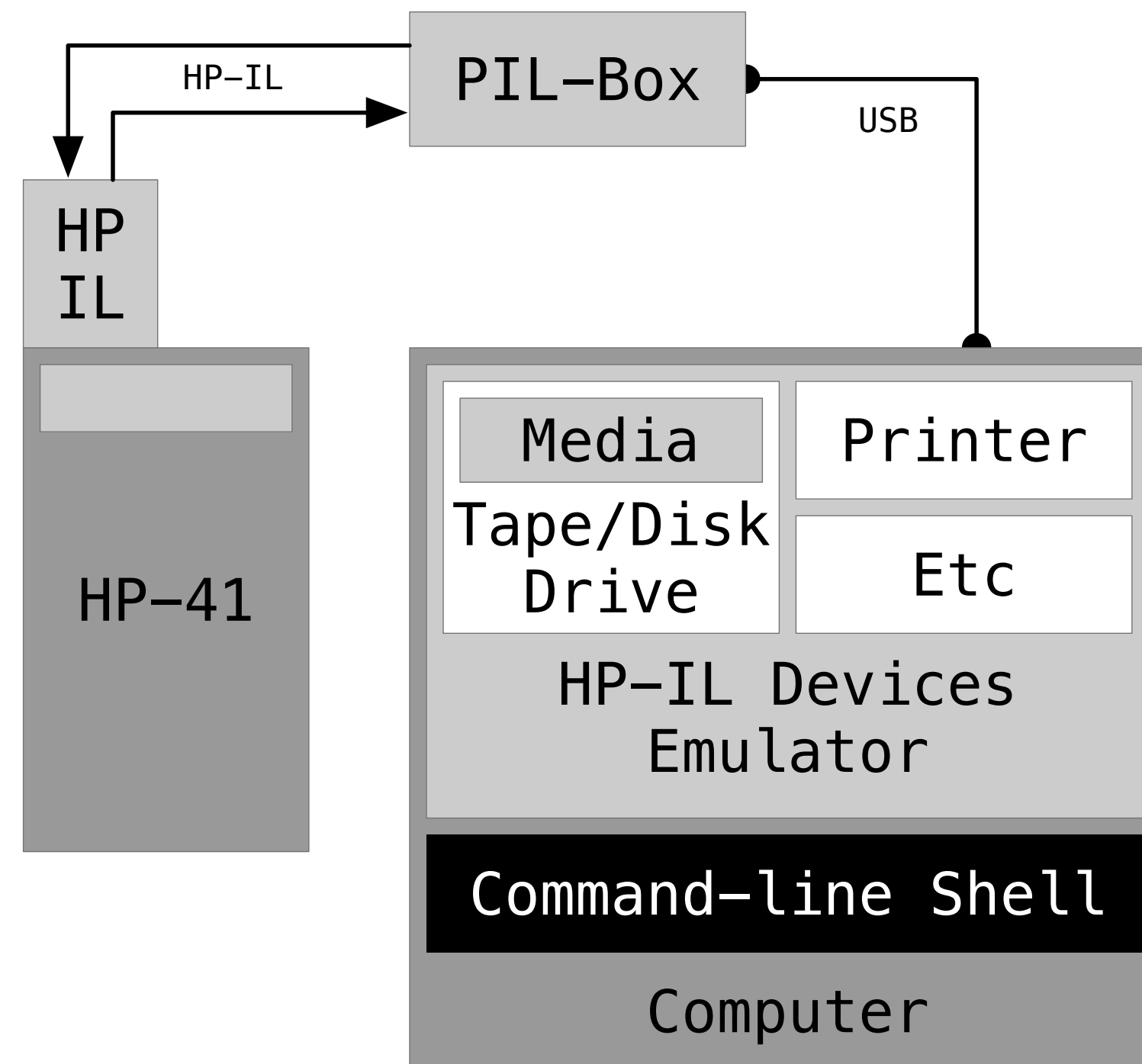
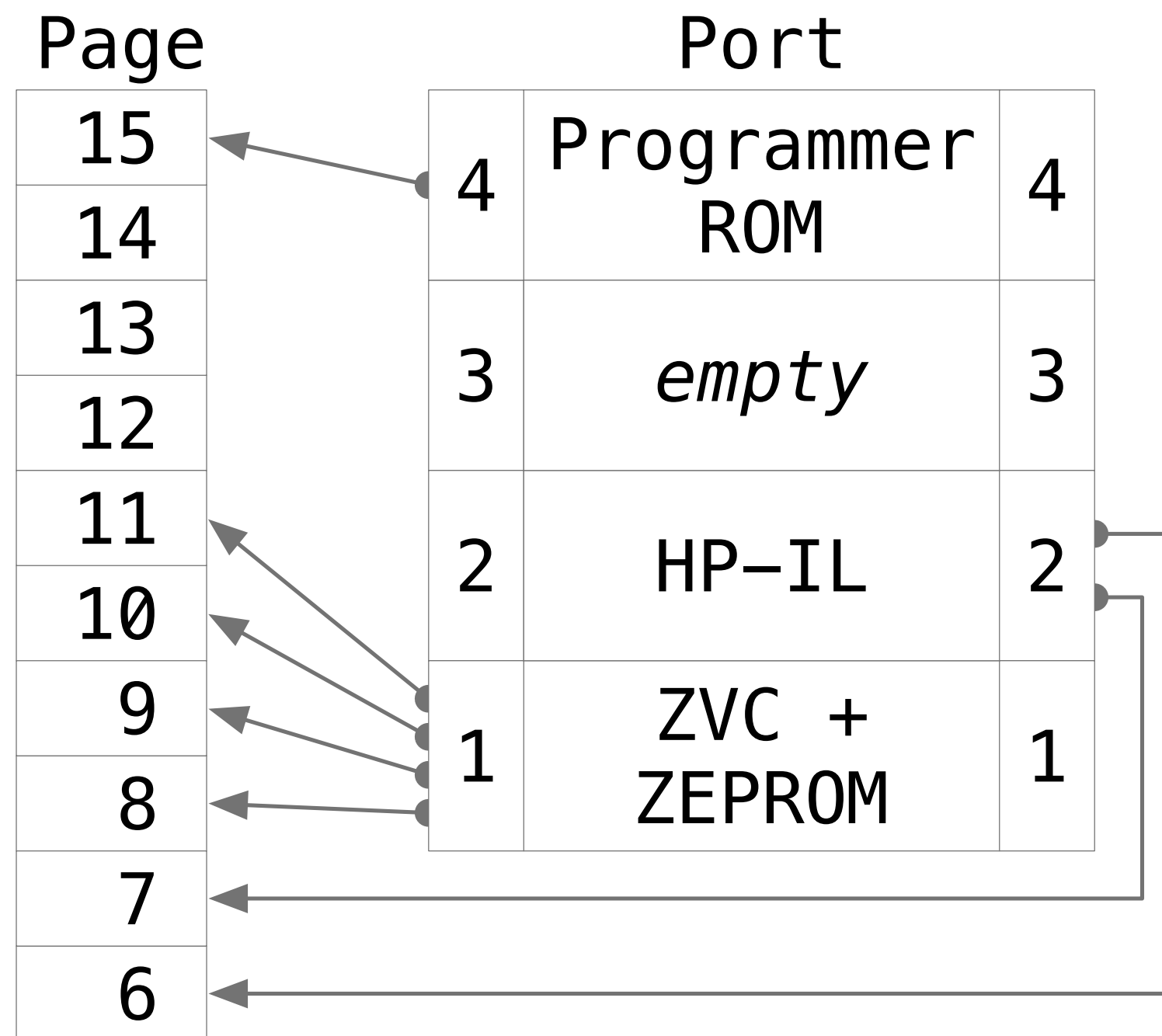


Summary



- * In this section, we will create a 8K flat ZEPROM module.
- * The Programmer ROM image must be modified to enable ZEPROM bank switching mode to get the 8K flat mode.
- * Once programmed, the module will contain the following ROM images:
 - * Programmer (ZEPBx) upper pages
 - * ZENROM lower pages

Setup



- * Erased ZEPR0M inserted in ZVC.
- * HP-41 connected to the PIL-Box.
- * PIL-Box connected to the computer.
- * pyILPER running Z41 instance.
- * Command-line Shell started.

Assign Programmer ROM to Bank 1

✱ Go to workspace

```
cd ~/Z41/temp
```

✱ Apply BANK 1 modification

```
// clone Programmer ROM file
cp ../roms/ZEPROM.ROM ZEPB1.ROM

// read original bank switching bits [FFD: 00B9]
xxd -seek $((16#1FFA)) -p -u -l 2 ZEPB1.ROM

// apply bank 2 modification [FFD= 01]
printf "001FFA: 01" | xxd -r - ZEPB1.ROM

// read updated bank switching bits [FFD: 01B9]
xxd -seek $((16#1FFA)) -p -u -l 2 ZEPB1.ROM
```

✱ Update checksum value (optional)

```
// show original checksum value [FFF: 029F]
xxd -seek $((16#1FFE)) -p -u -l 2 ZEPB1.ROM

// clear checksum value [FFF= 0000]
printf "001FFE: 0000" | xxd -r - ZEPB1.ROM

// validate previous line [FFF: 0000]
xxd -seek $((16#1FFE)) -p -u -l 2 ZEPB1.ROM

// update file checksum [FFF= 019F]
cat ZEPB1.ROM | xxd -p -u -c 2 | perl -e 'my $word = 0; my $prev = 0; my $curr = 0; while(<>) { $prev = $curr; $word = hex($_) & 1023; $curr += $word; if ($curr > 1023) { $curr = ($curr & 1023) + 1; } } my $csum = (-$prev) & 1023; printf "001FFE: %04X", $csum;' | xxd -r - ZEPB1.ROM

// validate previous line [FFF: 019F]
xxd -seek $((16#1FFE)) -p -u -l 2 ZEPB1.ROM
```

bit-9	bit-8	ZEPROM bank-switch bits (pFFD)
0	0	Page is not bank-switched
0	1	Bank-switched, primary bank
1	0	Bank-switched, indeterminate bank (HP)
1	1	Bank-switched, secondary bank



```
cat ZEPB1.ROM | xxd -p -u -c 2 | perl -e '
my $word = 0;
my $prev = 0;
my $curr = 0;
while(<>) {
    $prev = $curr;
    $word = hex($_) & 1023;
    $curr += $word;
    if ($curr > 1023) {
        $curr = ($curr & 1023) + 1;
    }
}
my $csum = (-$prev) & 1023;
printf "001FFE: %04X", $csum;
' | xxd -r - ZEPB1.ROM
```


Assign Programmer ROM to Bank 2

✱ Go to workspace

```
cd ~/Z41/temp
```

✱ Apply BANK 2 modification

```
// clone Programmer ROM file
cp ../roms/ZEPROM.ROM ZEPB2.ROM

// read original bank switching bits [FFD: 00B9]
xxd -seek $((16#1FFA)) -p -u -l 2 ZEPB2.ROM

// apply bank 2 modification [FFD= 03]
printf "001FFA: 03" | xxd -r - ZEPB2.ROM

// read updated bank switching bits [FFD: 03B9]
xxd -seek $((16#1FFA)) -p -u -l 2 ZEPB2.ROM
```

✱ Update checksum value (optional)

```
// show original checksum value [FFF: 029F]
xxd -seek $((16#1FFE)) -p -u -l 2 ZEPB2.ROM

// clear checksum value [FFF= 0000]
printf "001FFE: 0000" | xxd -r - ZEPB2.ROM

// validate previous line [FFF: 0000]
xxd -seek $((16#1FFE)) -p -u -l 2 ZEPB2.ROM

// update file checksum [FFF= 039E]
cat ZEPB2.ROM | xxd -p -u -c 2 | perl -e 'my $word = 0; my $prev = 0; my $curr = 0; while(<>) { $prev = $curr; $word = hex($_) & 1023; $curr += $word; if ($curr > 1023) { $curr = ($curr & 1023) + 1; } } my $csum = (-$prev) & 1023; printf "001FFE: %04X", $csum;' | xxd -r - ZEPB2.ROM

// validate previous line [FFF: 039E]
xxd -seek $((16#1FFE)) -p -u -l 2 ZEPB2.ROM
```

bit-9	bit-8	ZEPROM bank-switch bits (pFFD)
0	0	Page is not bank-switched
0	1	Bank-switched, primary bank
1	0	Bank-switched, indeterminate bank (HP)
1	1	Bank-switched, secondary bank



```
cat ZEPB2.ROM | xxd -p -u -c 2 | perl -e '
my $word = 0;
my $prev = 0;
my $curr = 0;
while(<>) {
    $prev = $curr;
    $word = hex($_) & 1023;
    $curr += $word;
    if ($curr > 1023) {
        $curr = ($curr & 1023) + 1;
    }
}
my $csum = (-$prev) & 1023;
printf "001FFE: %04X", $csum;
' | xxd -r - ZEPB2.ROM
```


Virtual Media Creation → Z41C2

* Go to workspace

```
cd ~/Z41/temp
```

* Create LIF volume

```
lifinit -m cass -z z41c2.dat 64  
liflabel z41c2.dat Z41C2
```

* Add ROM images to LIF volume

```
cat ../roms/ZENROM.ROM | rom41er ZENROM | lifput z41c2.dat  
cat ../roms/ZEPROM.ROM | rom41er ZEPROM | lifput z41c2.dat  
cat          ZEPB1.ROM | rom41er ZEPB1 | lifput z41c2.dat  
cat          ZEPB2.ROM | rom41er ZEPB2 | lifput z41c2.dat
```

* Export LIF volume

```
cp z41c2.dat ../pyilper
```

* Exit workspace

```
cd
```

* Show LIF volume content

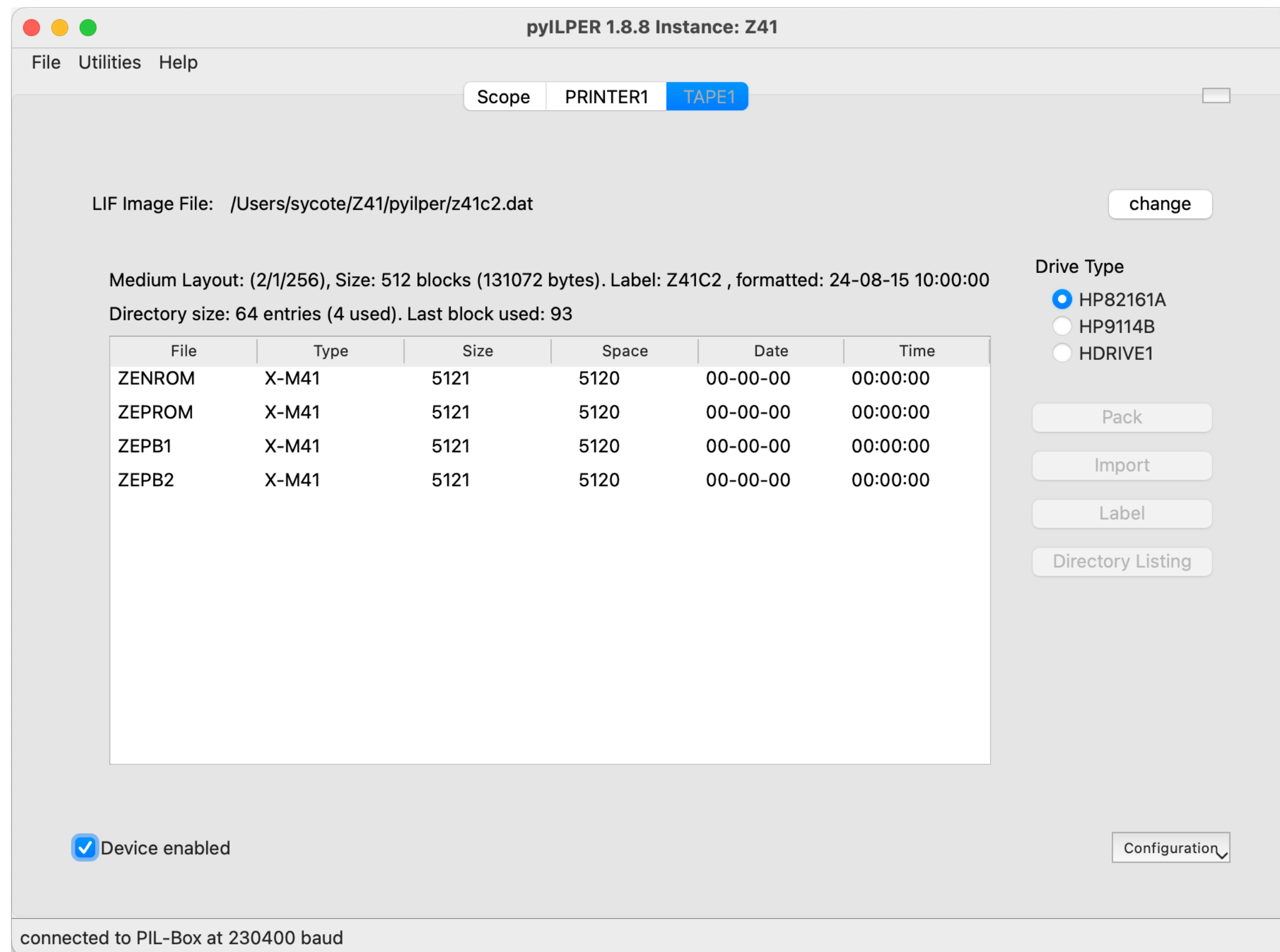
```
lifdir ~/Z41/pyilper/z41c2.dat
```

```
Volume : Z41C2 , formatted : 03/08/24 09:00:00  
Tracks: 2  
Surfaces: 1  
Blocks/Track: 256  
Total size: 512 Blocks, 131072 Bytes
```

ZENROM	X-M41	5121/5120
ZEPROM	X-M41	5121/5120
ZEPB1	X-M41	5121/5120
ZEPB2	X-M41	5121/5120

```
4 files (64 max), last block used: 92 of 512
```


pyILPER TAPE1 → Z41C2



- * Select **TAPE1** tab
- * Set **Drive Type** radio-button to **HP82161A**
- * Click on **change** push-button
- * In the **Select LIF Image File** dialog-box
- * Go to **~/Z41/pyilper** directory
- * Select **z41c2.dat** file
- * Click on **Open** push-button
- * Check **Device enabled** check-box
- * **TAPE1** loop address is 2

ZEPROM Programming



- * ZVC (Replica) switch set to **DOWN (PRG)**.
- * To burn ROM images into ZEPROM, do the following commands:

"ZENROM"
8
ILBURN



"ZEPB1"
9
ILBURN



"ZENROM"
10
ILBURN



"ZEPB2"
11
ILBURN

- * ZVC (Replica) switch set to **UP (RUN)**.
- * **OFF**
- * Remove all modules and insert the newly programmed ZEPROM in port 1.

- * **ON**

- * **CAT 2** →

...
-ZENROM 3B
-PROGRAMMER
...

	bank 1	bank 2
upper	PROGRAMMER	PROGRAMMER
lower	ZENROM	ZENROM

USER CODE FUNCTIONS

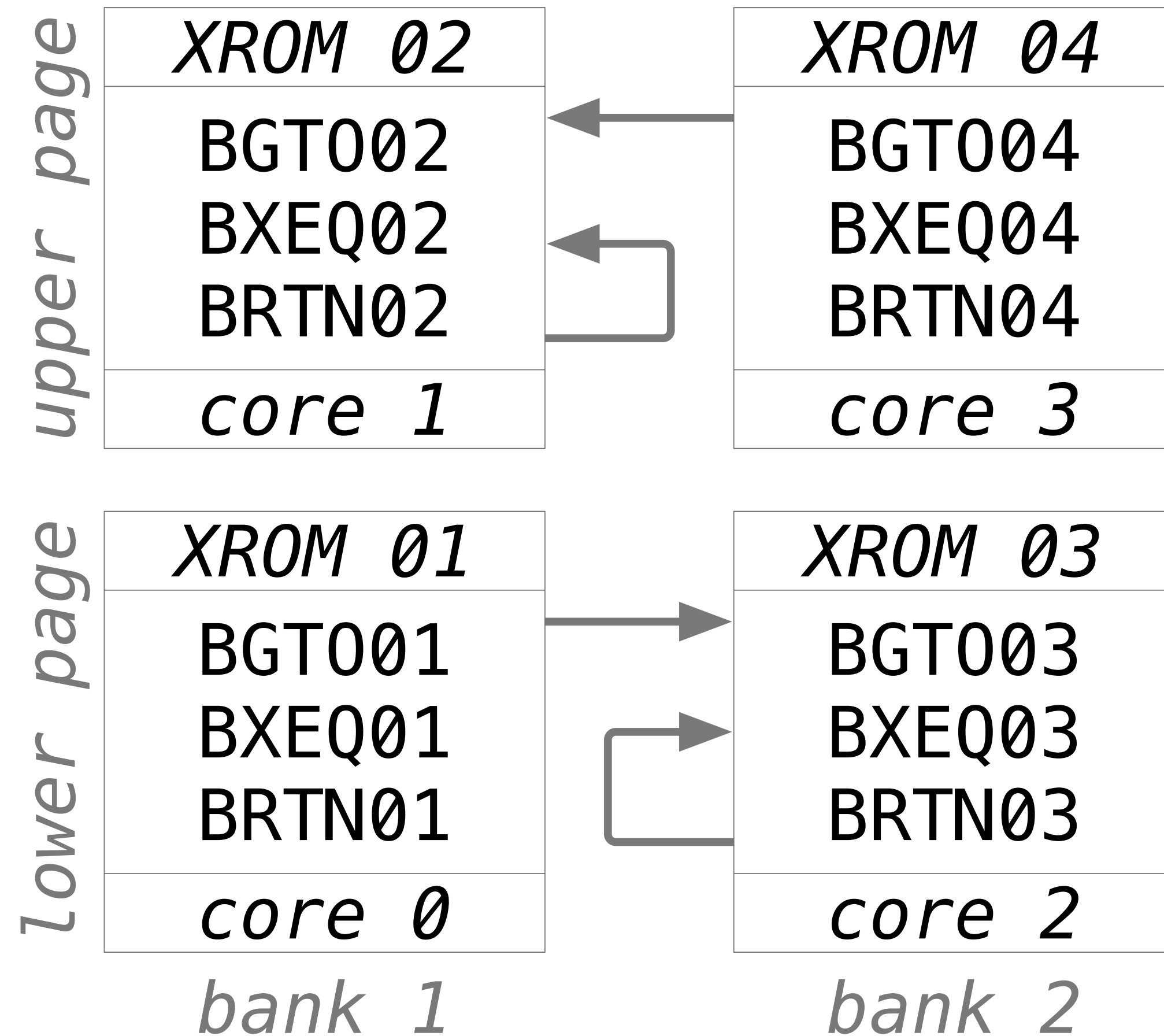
User Code Functions (1/2)

	Description
* BLANK?	→ Verifies that all cells of the 4K page is set to 0. Input → X: target-page Display → "YES" or "NO"
* INITPG	→ Initialize a blank 4K page. (+1 FAT entry) Input → A: "header[8..11],trailer[4]" Input → Y: target-page Input → X: ± xrom-id.number-of-fat-entry {positive mean flat & negative mean bank-switched}
* ADDBSW	→ Adds low level bank switching code to page. Input → X: target-page Output → low level page switching code
* ADDMCF	→ Add machine code bank switching functions to page. (+2 FAT entries) Input → X: target-page Output → high level page switching code: PBxx & SBxx (xx = xrom-id)
* ADDUCF	→ Add user code bank switching functions to page. (+3 FAT entries) Input → X: target-page Output → user programs page switching code: (xx = xrom-id) BGTOxx & BXEQxx → select bank, then do GTO/XEQ IND X BRTNxx → select bank, then do RTN

pFC7 100 ENBANK1
pFC8 3E0 RTN
pFC9 180 ENBANK2
pFCA 3E0 RTN

XROM 01 ↘
pFE9 0B1 1 x
pFEA 030 0 x
pFEB 002 B B
pFEC 010 P P
pFED 2D3 JNC -26 pFC7
pFEE 0B1 1 x
pFEF 030 0 x
pFF0 002 B B
pFF1 013 S S
pFF2 2BB JNC -29 pFC9

BGTOxx, BXEQxx & BRTNxx Behavior



- * Bank 1 is active
 - * Core 0 functions calls ENBANK2.
BANK1 → BANK2
 - * Core 1 functions calls ENBANK1.
BANK1 → BANK1
- * Bank 2 is active
 - * Core 2 functions calls ENBANK2.
BANK2 → BANK2
 - * Core 3 functions calls ENBANK1.
BANK2 → BANK1

User Code Functions (1/2)

Description

- * **COMPUC** → Compile user code program in main memory.
Warning → always execute PACK or GTO .. before executing this function.
Input → A: program-name (or empty for the current program)
- * **PRGMLN** → Return program length and number of alpha labels
Warning → always execute PACK or GTO .. before executing this function.
Input → A: program-name (or empty for the current program)
Output → bbbbbb.ccc (program-length-in-bytes.number-of-alpha-labels)
- * **BURNUC** → Burn user code program from main-RAM/ROM/Q-ROM to ZEPROM/MLDL/Q-ROM.
Input → A: program-name (or empty for the current program)
Input → X: target-page
- * **CHKSUM** → Compute and return checksum for specified page.
Input → X: target-page
Output → Y: checksum-address
Output → X: checksum-value
Output → L: target-page
- * **BURNWD** → Burn word to a specified address
Input → Y: word-address
Input → X: word-value

4K Page Format

ADDRESS	VALUE	DESCRIPTION

INITPG: ROM ID, FAT size & ROM label		
p000	0rr	ROM identifier (XROM ID), rr=01..1F hex
p001	0nn	number (n) of entries in FAT, nn=01..40 hex
p002	t0a	address of the first function or program label
p003	0aa	t=0 (mcode), t=1 (focal), aaa=address_hex
p...		
p(2n)	t0a	address of the last function or program label
p(2n)+1	0aa	
p(2n)+2	000	end of FAT (Function Address Table)
p(2n)+3	000	end of FAT - two consecutive 000 words
p(2n)+4 to pFC2		available for user and/or machine code
pFC3 to pFC6		ENBANK3 & ENBANK4 (not supported by ZEPROM)
ADDBSW: Low level bank switching code		
pFC7	100	ENBANK1 (enable bank 1)
pFC8	3E0	RTN
pFC9	180	ENBANK2 (enable bank 2)
pFCA	3E0	RTN

ADDRESS	DESCRIPTION

ADDUCF: user code bank switching functions	
pFCB to pFE8	BGTOxx, BXEQxx & BRTNxx functions
ADDMCF: machine code bank switching functions	
pFE9 to pFF2	PBxx & SBxx functions
Interrupt vector	
pFE4 to pFFA	polling points, seven (7) callback functions
INITPG: revision string	
pFFB	revision level (char)
pFFC	revision level (char)
pFFD	revision level (char) & ZEPROM bank-switch bits
pFFE	revision level (char)
CHKSUM & BURNWD: ROM checksum value	
pFFF	checksum

bit-9	bit-8	ZEPROM bank-switch bits (pFFD)
0	0	Page is not bank-switched
0	1	Bank-switched, primary bank
1	0	Bank-switched, indeterminate bank (HP)
1	1	Bank-switched, secondary bank



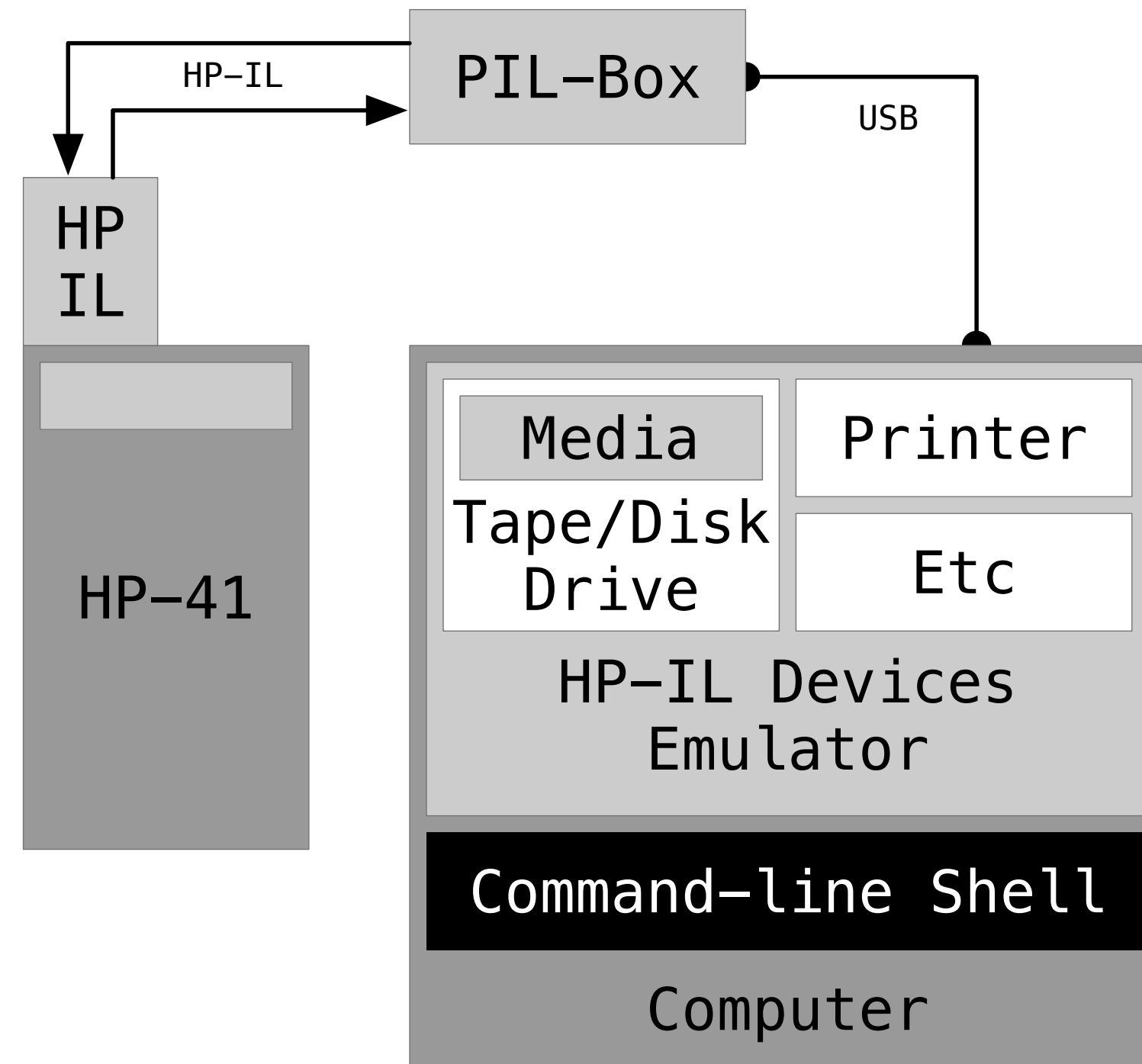
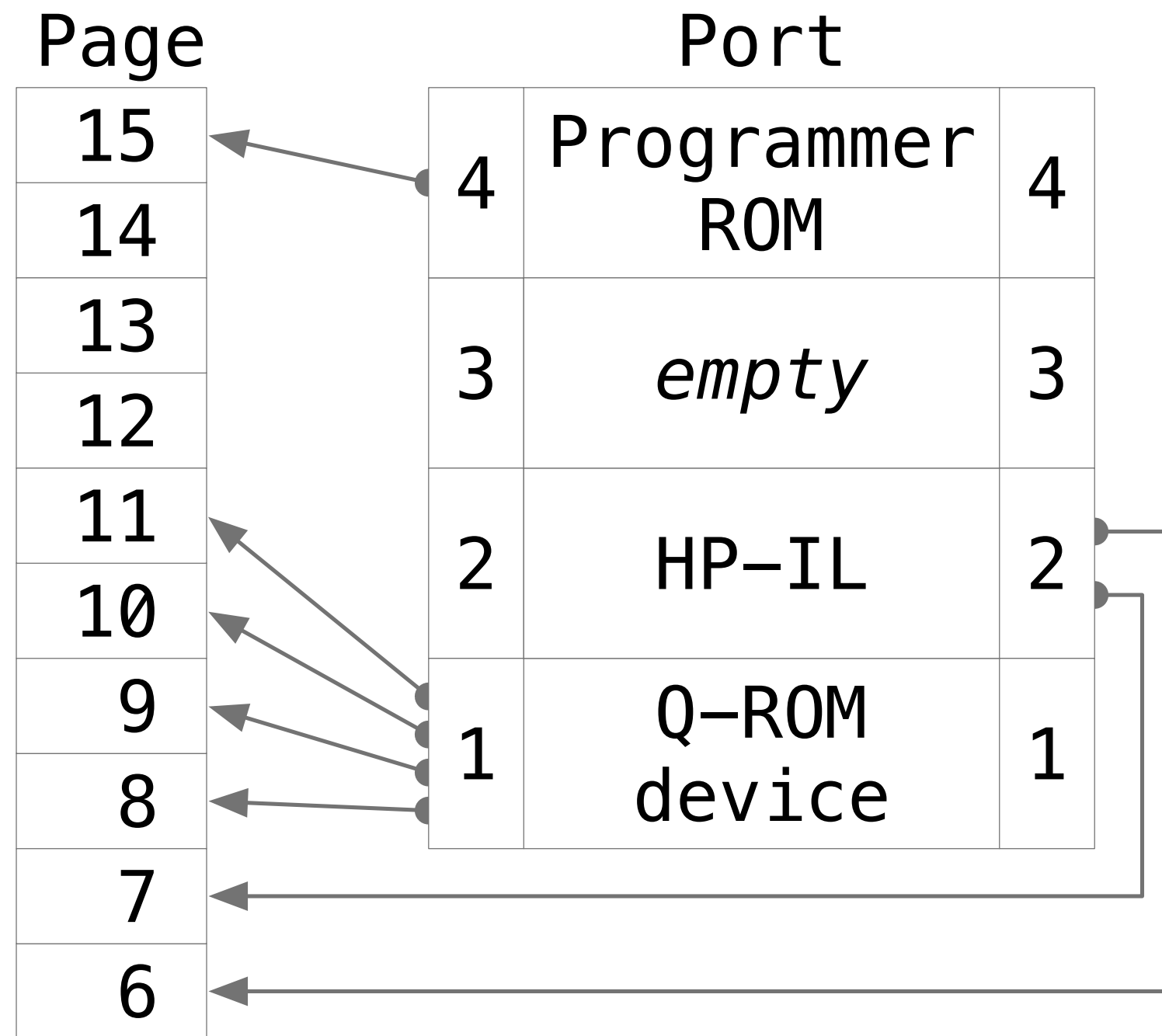
BANK SWITCHED 16K

WITH USER CODE

Summary

- * In this section, we will :
 - * create a 16K bank switched ZEPRON module.
 - * create RPN programs that uses bank switched goto (BGTO), bank switched execute (BXEQ) and bank switched return (BRTN) functions.

Setup with Q-ROM Device



- * 16K Q-ROM device inserted into HP-41
- * HP-41 connected to the PIL-Box.
- * PIL-Box connected to the computer.
- * pyILPER running Z41 instance.
- * Command-line Shell started.

BGTOxx, BXEQxx & BRTNxx Example

* Upper Page - Bank 1 - XROM 02

01 LBL "U1CFG"
02 "U2DISP"
03 ASTO 00
04 "L2BEEP"
05 ASTO 01
06 **BRTN02**
07 END

* Lower Page - Bank 1 - XROM 01

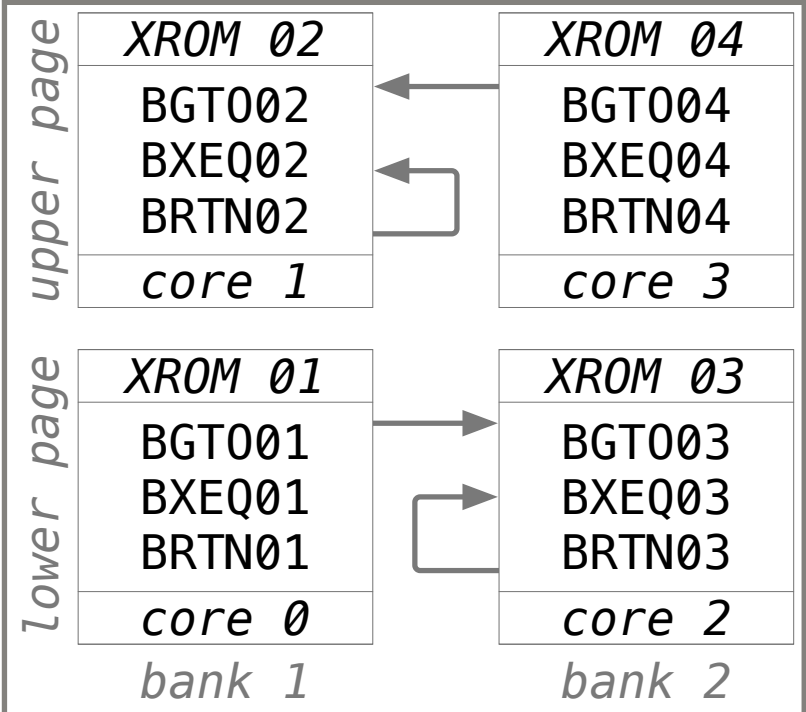
01 LBL "L1MAIN" ← XEQ "L1MAIN"
02 "U1CFG"
03 ASTO X
04 **BXEQ02**
05 "HELLO HHC"
06 RCL 00
07 **BXEQ01**
08 END

* Upper Page - Bank 2 - XROM 04

01 LBL "U2DISP"
02 AVIEW
03 RCL 01
04 **BGTO03**
05 END

* Lower Page - Bank 2 - XROM 03

01 LBL "L2BEEP"
02 BEEP
03 **BRTN04**
04 END



Source Files

~/Z41/src/u1cfg.txt

LBL "U1CFG"
"U2DISP"
ASTO 00
"L2BEEP"
ASTO 01
BRTN02
END

~/Z41/src/l1main.txt

LBL "L1MAIN"
"U1CFG"
ASTO X
BXEQ02
"HELLO HHC"
RCL 00
BXEQ01
END

~/Z41/src/u2disp.txt

LBL "U2DISP"
AVIEW
RCL 01
BGTO03
END

~/Z41/src/l2beep.txt

LBL "L2BEEP"
BEEP
BRTN04
END

~/Z41/src/myrom.xrom

1 0 -MYROM_L1
1 1 PB01
1 2 SB01
1 3 BGTO01
1 4 BRTN01
1 5 BXEQ01
1 6 XROM'L1MAIN'
2 0 -MYROM_U1
2 1 PB02
2 2 SB02
2 3 BGTO02
2 4 BRTN02
2 5 BXEQ02
2 6 XROM'U1CFG'
3 0 -MYROM_L2
3 1 PB03
3 2 SB03
3 3 BGTO03
3 4 BRTN03
3 5 BXEQ03
3 6 XROM'L2BEEP'
4 0 -MYROM_U2
4 1 PB04
4 2 SB04
4 3 BGTO04
4 4 BRTN04
4 5 BXEQ04
4 6 XROM'U2DISP'



Virtual Media Creation → Z41C3

* Go to workspace

```
cd ~/Z41/temp
```

* Create LIF volume

```
lifinit -m cass -z z41c3.dat 64  
liflabel z41c3.dat Z41C3
```

* Add programs to LIF volume

```
cat ../src/l1main.txt | comp41 -x ../src/myrom.xrom | raw41lif L1MAIN | lifput z41c3.dat  
cat ../src/u1cfg.txt | comp41 -x ../src/myrom.xrom | raw41lif U1CFG | lifput z41c3.dat  
cat ../src/l2beep.txt | comp41 -x ../src/myrom.xrom | raw41lif L2BEEP | lifput z41c3.dat  
cat ../src/u2disp.txt | comp41 -x ../src/myrom.xrom | raw41lif U2DISP | lifput z41c3.dat
```

* Export LIF volume

```
cp z41c3.dat ../pyilper
```

* Exit workspace

```
cd ..
```

* Show LIF volume content

```
lifdir ~/Z41/pyilper/z41c3.dat
```

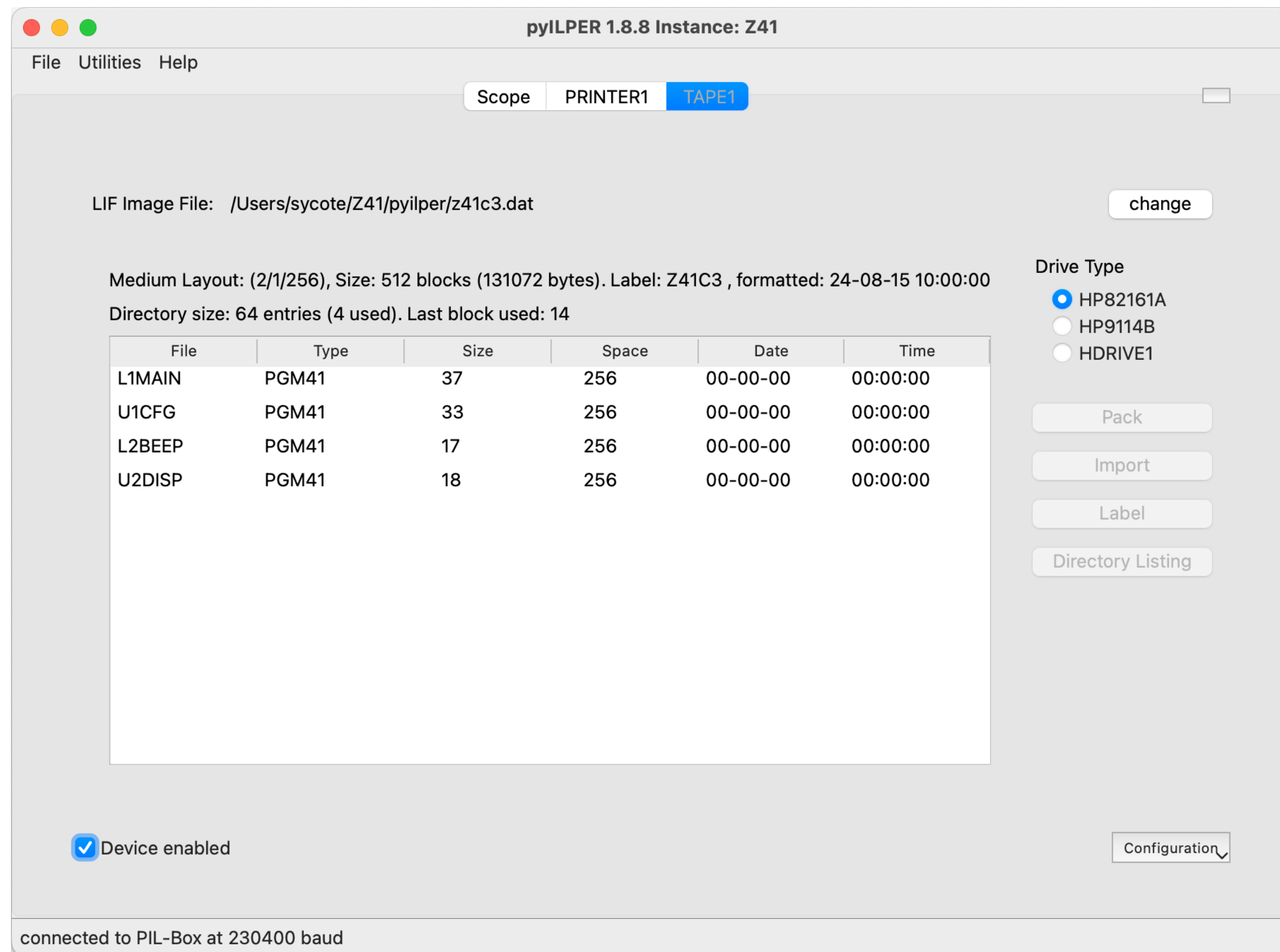


```
Volume : Z41C3 , formatted : 03/08/24 09:00:00  
Tracks: 2  
Surfaces: 1  
Blocks/Track: 256  
Total size: 512 Blocks, 131072 Bytes
```

L1MAIN	PGM41	37/256
U1CFG	PGM41	33/256
L2BEEP	PGM41	17/256
U2DISP	PGM41	18/256

```
4 files (64 max), last block used: 13 of 512
```


pyILPER TAPE1 → Z41C3



- * Select **TAPE1** tab
- * Set **Drive Type** radio-button to **HP82161A**
- * Click on **change** push-button
- * In the **Select LIF Image File** dialog-box
- * Go to **~/Z41/pyilper** directory
- * Select **z41c3.dat** file
- * Click on **Open** push-button
- * Check **Device enabled** check-box
- * **TAPE1** loop address is 2

MYROM Creation

```
8
"-MYROM L1,L11A"
-1.007
INITPG
X<>Y
ADDBSW
ADDMCF
ADDUCF
    "L1MAIN"
    READP
    GTO ..
    COMPUC
    BURNUC
CHKSUM
BURNWD
"MRL11A"
8
ILSAVE
```



```
9
"-MYROM U1,U11A"
-2.007
INITPG
X<>Y
ADDBSW
ADDMCF
ADDUCF
    "U1CFG"
    READP
    GTO ..
    COMPUC
    BURNUC
CHKSUM
BURNWD
"MRU11A"
9
ILSAVE
```

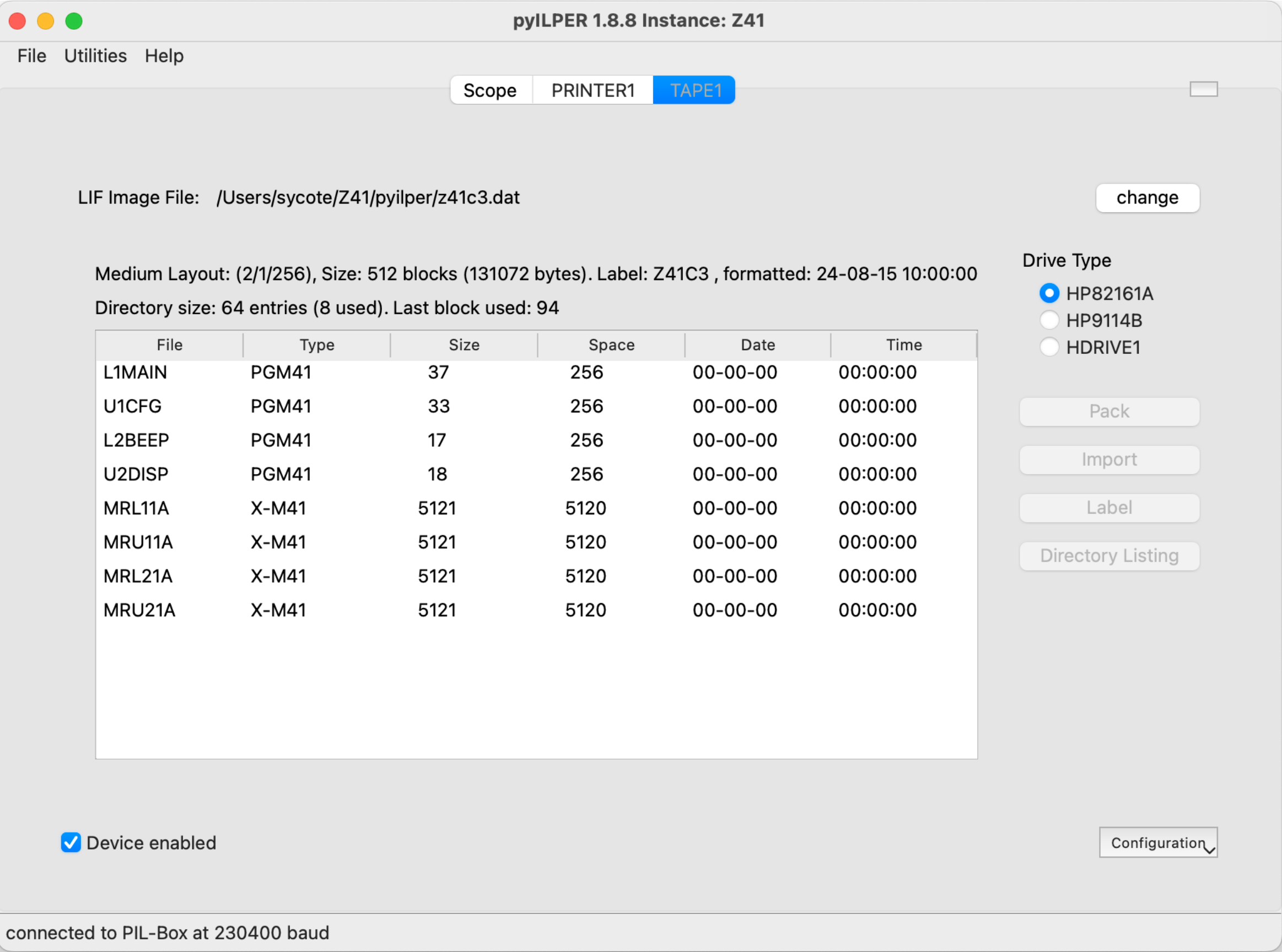


```
10
"-MYROM L2,L21A"
-3.007
INITPG
X<>Y
ADDBSW
ADDMCF
ADDUCF
    "L2BEEP"
    READP
    GTO ..
    COMPUC
    BURNUC
CHKSUM
BURNWD
"MRL21A"
10
ILSAVE
```



```
11
"-MYROM U2,U21A"
-4.007
INITPG
X<>Y
ADDBSW
ADDMCF
ADDUCF
    "U2DISP"
    READP
    GTO ..
    COMPUC
    BURNUC
CHKSUM
BURNWD
"MRU21A"
11
ILSAVE
```

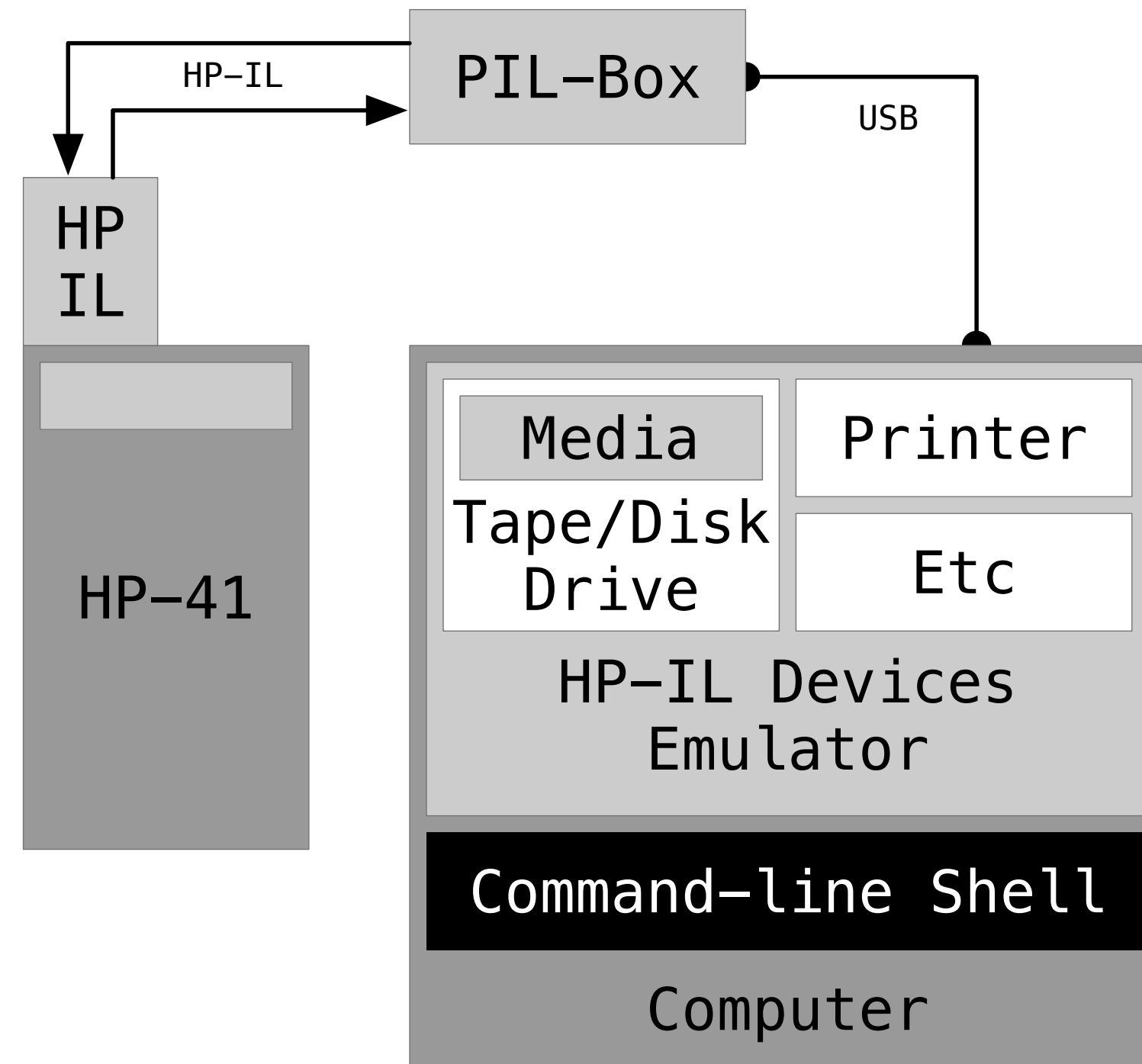
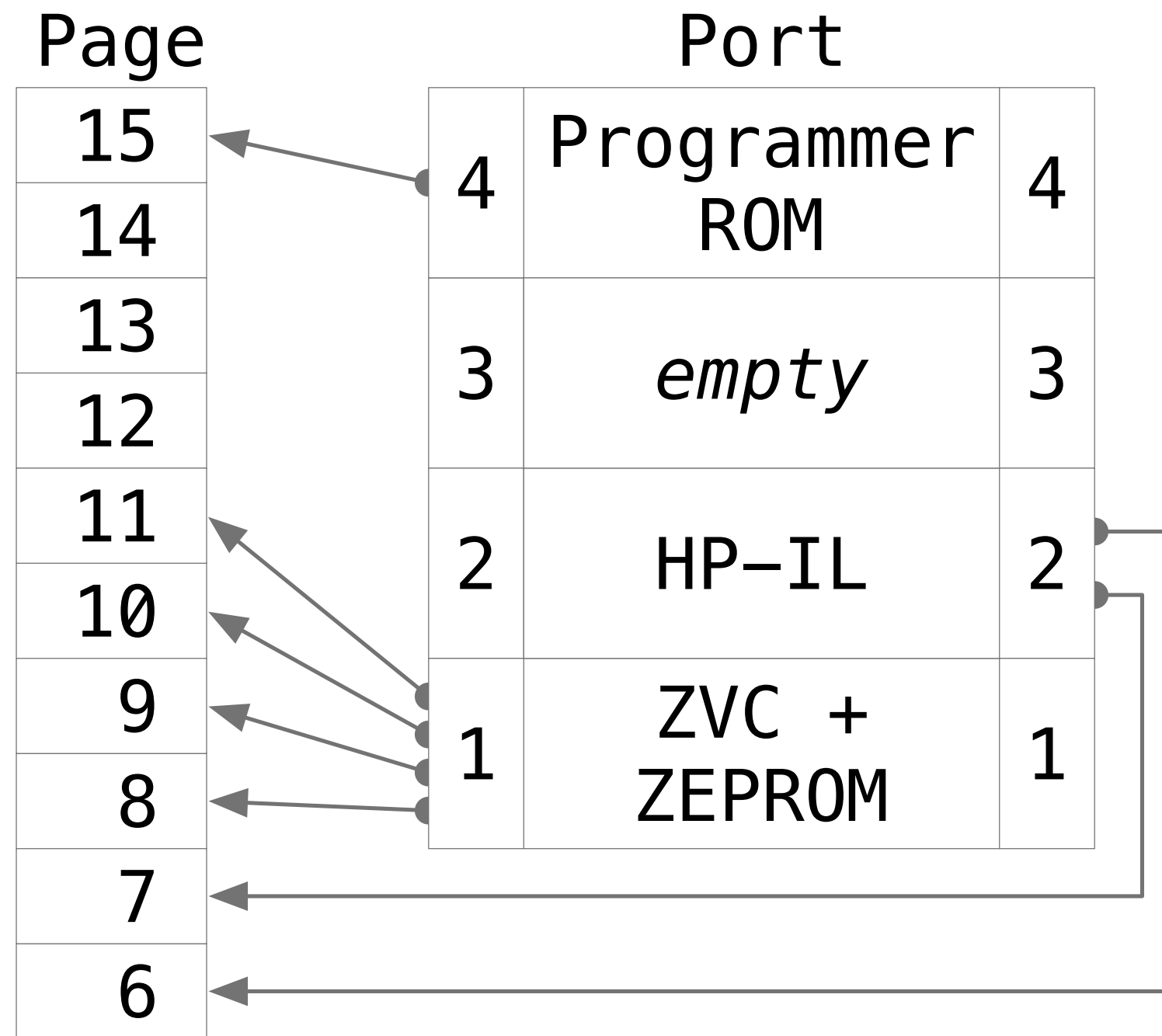

pyILPER TAPE1 → Z41C3 (updated)



* Tape listing

- * L1MAIN MAIN : Lower page : Bank 1
- * U1CFG CFG : Upper page : Bank 1
- * L2BEEP BEEP : Lower page : Bank 2
- * U2DISP DISP : Upper page : Bank 2
- * MRL11A MyROM : Lower page : Bank 1
- * MRU11A MyROM : Upper page : Bank 1
- * MRL21A MyROM : Lower page : Bank 2
- * MRU21A MyROM : Upper page : Bank 2

Setup with ZVC & ZEPRROM



- * Erased ZEPRROM inserted in ZVC.
- * HP-41 connected to the PIL-Box.
- * PIL-Box connected to the computer.
- * pyILPER running Z41 instance.
- * Command-line Shell started.

ZEPROM Programming



- * ZVC (Replica) switch set to **DOWN (PRG)**.

- * To burn ROM images into ZEPROM, do the following commands:

"MRL11A"
8
ILBURN



"MRU11A"
9
ILBURN



"MRL21A"
10
ILBURN



"MRU21A"
11
ILBURN

- * ZVC (Replica) switch set to **UP (RUN)**.
- * **OFF**
- * Remove all modules and insert the newly programmed ZEPROM in port 1.
- * **ON**
- * **CAT 2**

REFERENCES

Price & Availability

- * ZVC Replica
 - * Availability → now!
 - * Prices → include worldwide shipping.
 - * 50€ for a bare board. ⁽¹⁾
 - * 80€ for a ready to use complete module.
 - * Ordering → send an email to Diego Díaz at clonix41@gmail.com
- * ZEPROM
 - * Availability → frequently available on eBay.
 - * Price → varies.
 - * Searching → most are sold as Surveying modules on eBay.

Diego Díaz Products

- * Contact: clonix41@gmail.com
- * Website: www.clonix41.org
- * Products:
 - * ZVC Replica HP-41 ZEPR0M Voltage Converter Replica
 - * 82120D HP-41 LiPo Battery Power Bank
 - * Clonix-D HP-41 Multi-ROM Module
 - * NoV-64d HP-41 Multi-ROM & HEPAX Emulation Module
 - * USB-41 HP-41 Multi-ROM & 82143A Printer Emulation Module
 - * PICkit #2 USB Programmer for Multi-ROM Modules
 - * ClonixConfig Windows Configuration Soft. for Multi-ROM Modules



Web Links (1/2)

- * Manuals

- * ZEPROM Programmer's Manual literature.hpcalc.org/items/1436
- * ZEPROM Programmer User's Manual literature.hpcalc.org/items/1437
- * ZENROM Manual literature.hpcalc.org/items/1434

- * Hardware

- * PIL-Box www.jeffcalc.hp41.eu/hpil
- * smdPILBox News www.hpmmuseum.org/forum/thread-21743.html
- * TULIP4041 News www.hpmmuseum.org/forum/thread-21514.html
- * TULIP4041 Dev. Board www.hpmmuseum.org/forum/thread-22013.html

Web Links (2/2)

- * Software

- * Virtual HP-IL

hp.giesselink.com/hpil.htm

- * pyILPER

github.com/bug400/pyilper

- * LIF Utilities

github.com/bug400/lifutils

- * Cygwin

www.cygwin.com

- * Homebrew

brew.sh

- * Other

- * HP-41 ROM Files

systemyde.com/hp41/archive.html

- * HP-41 Bank Switching

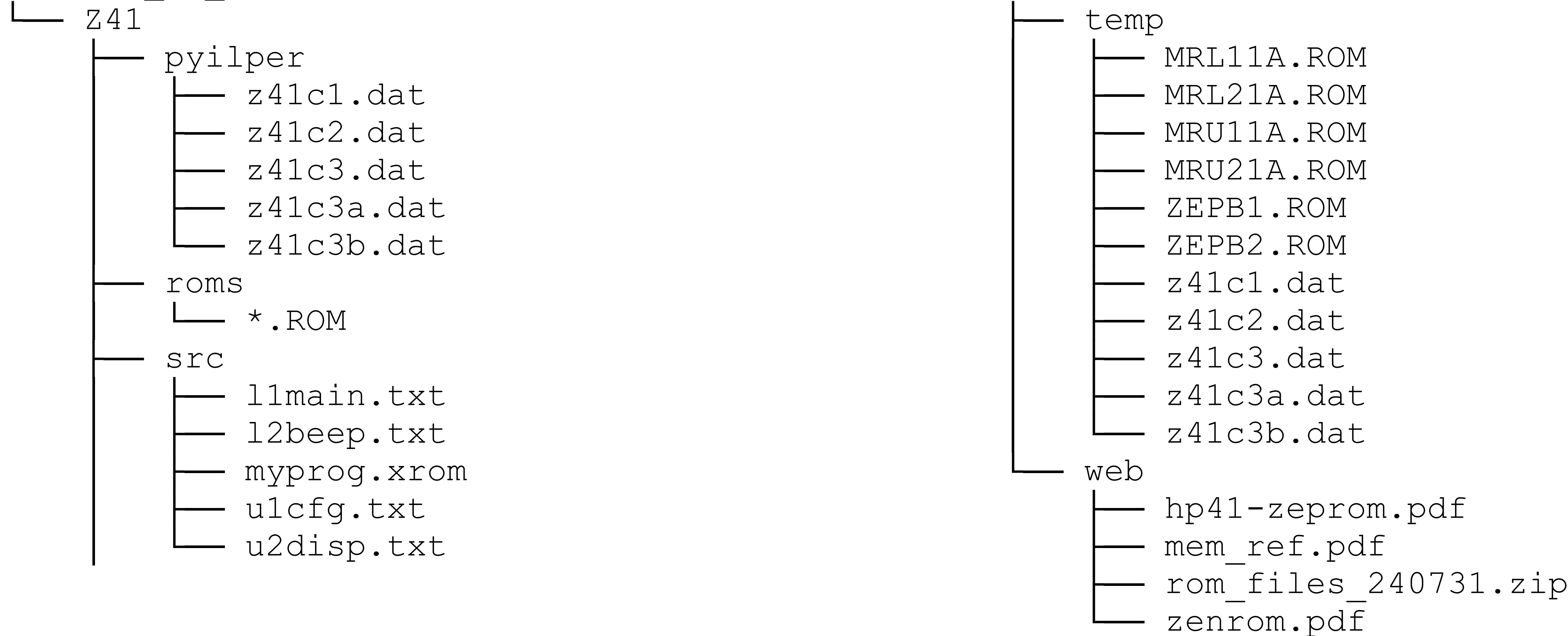
www.hpmuseum.org/forum/thread-14695.html

- * HP-41 Programmer's Handbook

literature.hpcalc.org/items/1618

Support File

HHC2024_SC_SUPPORT.zip



QUESTIONS ?



HHC 2024



THANK YOU !

HHC 2024