

p1Monitor

Qu'est-ce qu'un moniteur

Le mot **moniteur** peut prendre plusieurs sens. Dans le contexte actuel il s'agit d'un petit programme intégré dans une mémoire persistante et qui s'exécute au démarrage. Dans les années 70s au tout début de l'informatique personnelle. Les mémoires RAM et ROM coûtaient cher et les petits ordinateurs de base en possédaient très peu. Par exemple l'Apple I ne possédait qu'une ROM de 256 octets. Quel programme peut-on installer dans si peu d'espace? Le programme inclus habituellement dans ces petites ROMs ne permettait que de faire les opérations de base suivantes:

1. Examiner le contenu de la mémoire.
2. Modifier le contenu de la mémoire RAM.
3. lancer un programme machine.

Ce petit programme s'appellait un moniteur. **monitor** en anglais.

Le moniteur du **POMME-I** est inspiré du **Wozmon**, c'est à dire le moniteur du Apple I créé par Steve Wozniak en 1974. Ses fonctionnalités et son fonctionnement de base sont identique au **Wozmon** mais il comporte des ajouts.

Au démarrage

Le moniteur est l'application qui est lancée automatiquement au démarrage du **POMME-I**. La version du moniteur est indiquée à la suite de la version du **firmware du noyau**. Ensuite s'affiche le symbole **#** pour indiquer que le moniteur est prêt à recevoir une commande.

```
pomme I version 1.4R0 Copyright Jacques Deschenes, (c) 2023,24
Fcpu= 16Mhz

pomme I monitor version 1.4R0 Jacques Deschenes (c) 2023,24

#
```

Fonctions de bases

D'abord il faut savoir que toutes les entrées et sorties numériques sont en hexadécimal. Il n'y a cependant aucun préfixe comme **h**, **0x** ou **\$** pour indiquer qu'il s'agit d'entiers hexadécimaux. C'est inutile puisqu'il n'y a que la base hexadécimale d'utilisée.

Pour connaître la valeur de l'octet à une adresse donnée il suffit d'entrer l'adresse suivit de la touche **ENTER**.

```
#8000
```

```
8000: 82
```

```
#
```

Pour afficher le contenu d'une plage mémoire il faut indiquer l'adresse début et l'adresse fin de la plage séparées par un point. 8 octets sont affichés par ligne.

```
#6000.607F
```

```
6000: 9B AD 0C 25 19 CE 48 7E
```

```
6008: A3 55 AA 27 11 20 16 C6
```

```
6010: 80 00 A1 82 27 06 A1 AC
```

```
6018: 27 02 99 81 98 81 C6 48
```

```
6020: 00 A1 AA 26 09 5F 4F 4B
```

```
6028: 28 86 AC 00 80 00 AD 5A
```

```
6030: 72 10 50 C0 4F C7 50 C6
```

```
6038: B7 97 72 10 50 C1 AE 75
```

```
6040: 30 5A 27 10 72 03 50 C1
```

```
6048: F8 35 01 00 97 AE 8F 23
```

```
6050: BF 95 AD 47 72 18 50 03
```

```
6058: 72 1C 50 12 3F 8E A6 01
```

```
6060: B7 99 B7 9A CD 61 7A 72
```

```
6068: 06 00 8E 02 20 B7 35 56
```

```
6070: 50 62 35 AE 50 62 35 AE
```

```
6078: 50 64 35 56 50 64 A6 79
```

```
#
```

Pour modifier le contenu de la mémoire RAM, il faut indiquer l'adresse de début suivit d'un double point :. On peut indiquer plusieurs valeurs sur la même ligne de saisie, elles seront enregistrées à des adresses successives. Après le **ENTER** le moniteur affiche toujours le contenu original de l'adresse indiquée dans la commande.

```
#100: A6 9 AE 3C 80 83 81
```

```
0100: 00
```

```
#100.107
```

```
0100: A6 09 AE 3C 80 83 81 00
```

```
#
```

Si un programme a été chargé en mémoire RAM et qu'on veut l'exécuter il faut saisir l'adresse d'exécution du programme suivit de la lettre **R**. On peut relancer le programme une autre fois simplement en tapant la lettre **R** suivit de **ENTER** car l'adresse est conservée. Cependant si cette adresse est modifiée par une autre commande elle devra être saisie à nouveau. Les programmes exécutés par cette commande doivent se terminer par une instruction machine **RET**, de code hexadécimal **81** de sorte qu'à la sortie on revient dans le moniteur.

```

0100: A6 09 AE 3C 80 83 81 00
#100R

0100: A6
15488
pomme I monitor version 1.4R0  Jacques Deschenes (c) 2023,24

#R

15488
pomme I monitor version 1.4R0  Jacques Deschenes (c) 2023,24

#

```

Fonctions supplémentaires.

Chaîne de caractères ASCII

une adresse suivit du caractère **"** permet d'assembler une chaîne ASCII. La chaîne est terminée par un **0**.

```

#200"BONJOUR  CHEZ-VOUS!

0200: A6
#200.220

0200: 42 4F 4E 4A 4F 55 52 20
0208: 43 48 45 5A 2D 56 4F 55
0210: 53 21 00 21 0A 00 00 00
0218: 00 00 00 00 00 00 00 00
0220: 00

```

désassembleur

Une adresse suivit du caractère **@** liste un désassemblage du code à partir de l'adresse. La touche **ESPACE** permet de continuer le désassemblage et toute autre touche retourne au moniteur.

```

#6000@

6000: 9B
6000      9B          SIM
6001      AD 0C      CALLR 600F
6003      25 19      JRC 601E
6005      CE 48 7E    LDW X,487E
6008      A3 55 AA    CPW X,#55AA
600B      27 11      JREQ 601E
600D      20 16      JRA 6025
600F      C6 80 00    LD A,8000
6012      A1 82      CP A,#82

```

```
6014      27 06      JREQ 601C
6016      A1 AC      CP A,#AC
6018      27 02      JREQ 601C
601A      99        SCF
601B      81        RET
601C      98        RCF
601D      81        RET
601E      C6 48 00   LD A,4800
6021      A1 AA      CP A,#AA
6023      26 09      JRNE 602E
6025      5F        CLRW X
6026      4F        CLR A
6027      4B 28      PUSH #28
6029      86        POP CC
602A      AC 00 80 00 JPF 8000
#
```

Appels système

La commande ? affiche une carte de référence rapide des appels systèmes disponible.

KERNEL SERVICES				
CODE	FUNCTION	INPUT	OUTPUT	DESCRIPTION
0	RESET	NONE	NONE	reset computer
1	TICK	NONE	X=MSEC	return msec counter
2	PUTCHAR	X=CHR	NONE	print char
3	GETCHAR	NONE	A=CHAR	get char from term
4	CHAR?	NONE	A=0, -1	char received?
5	CLS	NONE	NONE	clear term screen
6	DELBACK	NONE	NONE	delete last char
7	GETLINE	X=line length	A=ln len X=buffer	read line from terminal
8	PUTS	X=STR	NONE	print string
9	PRT_INT	X=INT		
		A=SGN	A=LEN	print integer
A	SET_TMR	X=INT	NONE	set countdown timer
B	TIMEOUT?	NONE	A=0, -1	check time out
C	TONE	X=MSEC		
		Y=FREQ	NONE	generate tone
D	FILE OP	X=FCB	X=FCB	file operation
E	RAND	NONE	X=UINT	get random #
F	SEED	X=0, n	NONE	seed prng

Une adresse suivit du caractère **K** permet d'assembler un appel système. **K** pour **kernel**.

```
#100K 1

0100: 00
0103

#103K 9 ]

0103: 00
0106
0107
#100.107

0100: A6 01 83 A6 09 83 81 00

#100R

0100: A6
8269
pomme I monitor version 1.4R0 Jacques Deschenes (c) 2023,24

#R

12295
pomme I monitor version 1.4R0 Jacques Deschenes (c) 2023,24
```

Dans cet exemple on assemble 2 appels système

- le code d'appel **1** charge le registre **X** avec le compteur de millisecondes du système.
- le code d'appel **9** imprime la valeur du registre **X**.

Lorsqu'on assemble un appel système la prochaine adresse libre est indiquée, ici **103**. On assemble donc le 2ième appel à cette adresse.

- le caractère **]** assemble l'instruction **RET** qui met fin au programme.

Voici le désassemblage du programme qu'on vient de créer.

```
#100@

0100: A6
0100      A6 01          LD A, #01
0102      83            TRAP
0103      A6 09          LD A, #09
0105      83            TRAP
0106      81            RET
```

L'instruction machine **TRAP** est utilisée pour faire un appel système.

Sauvegarde et chargement d'un fichier binaire.

Depuis la version **1.4** le p1Monitor permet de sauvegarder et charger un fichier binaire.

- **adrS nom taille** permet de sauvegarder une plage mémoire dans un fichier.
- **adrL nom** permet de charger un fichier binaire en mémoire RAM.
- **CTRL+D** Affiche le aliste des fichiers disponibles.

Par exemple pour sauvegarder le programme qu'on vient de créer à l'adresse 100 on fait.

```
#100S TICKS.BIN 8

0100: A6operation completed
```

On fait **CTRL+D** pour s'assurer que le nouveau fichier est bien là. On va maintenant recharger ce programme mais à l'adresse 200 et l'exécuter.

```
#BONJOUR.BIN      32 bytes
BONJR.BIN         32 bytes
TICKS.BIN         8 bytes

3 files
3 sectors used

#200L TICKS.BIN

0200: 00operation completed

#200R

0200: A6
5580
pomme I monitor version 1.4R0  Jacques Deschenes (c) 2023,24

#R

7851
pomme I monitor version 1.4R0  Jacques Deschenes (c) 2023,24

#
```

Touches rapides

CTRL+D Affiche la liste des fichiers.

```
#BONJOUR.BIN      32 bytes
BONJR.BIN         32 bytes
TICKS.BIN         8 bytes
```

```
3 files
3 sectors used
#
```

CTRL+E Pour effacer tous les fichiers.

```
#BONJOUR.BIN      32 bytes
BONJR.BIN         32 bytes
TICKS.BIN         8 bytes

3 files
3 sectors used

#
Do you really want to erase all files? (N/Y)
#
0 files
0 sectors used
#
```

CTRL+X redémarre l'ordinateur.

```
pomme I version 1.4R0 Copyright Jacques Deschenes, (c) 2023,24
Fcpu= 16Mhz

pomme I monitor version 1.4R0  Jacques Deschenes (c) 2023,24

#
```

CTRL+B lance **p1BASIC**.

```
#
pomme BASIC version 1.2R0  Jacques Deschenes (c)2023,24
5504 bytes free
>
```

CTRL+F lance **p1Forth**.

```
#
p1Forth version 5.1R3  Jacques Deschenes (c) 2023,24
```

p1BASIC et **p1Forth** peuvent-être quittés avec **CTRL+X** ou bien la commande **BYE** ce qui a pour effet de redémarrer l'ordinateur.