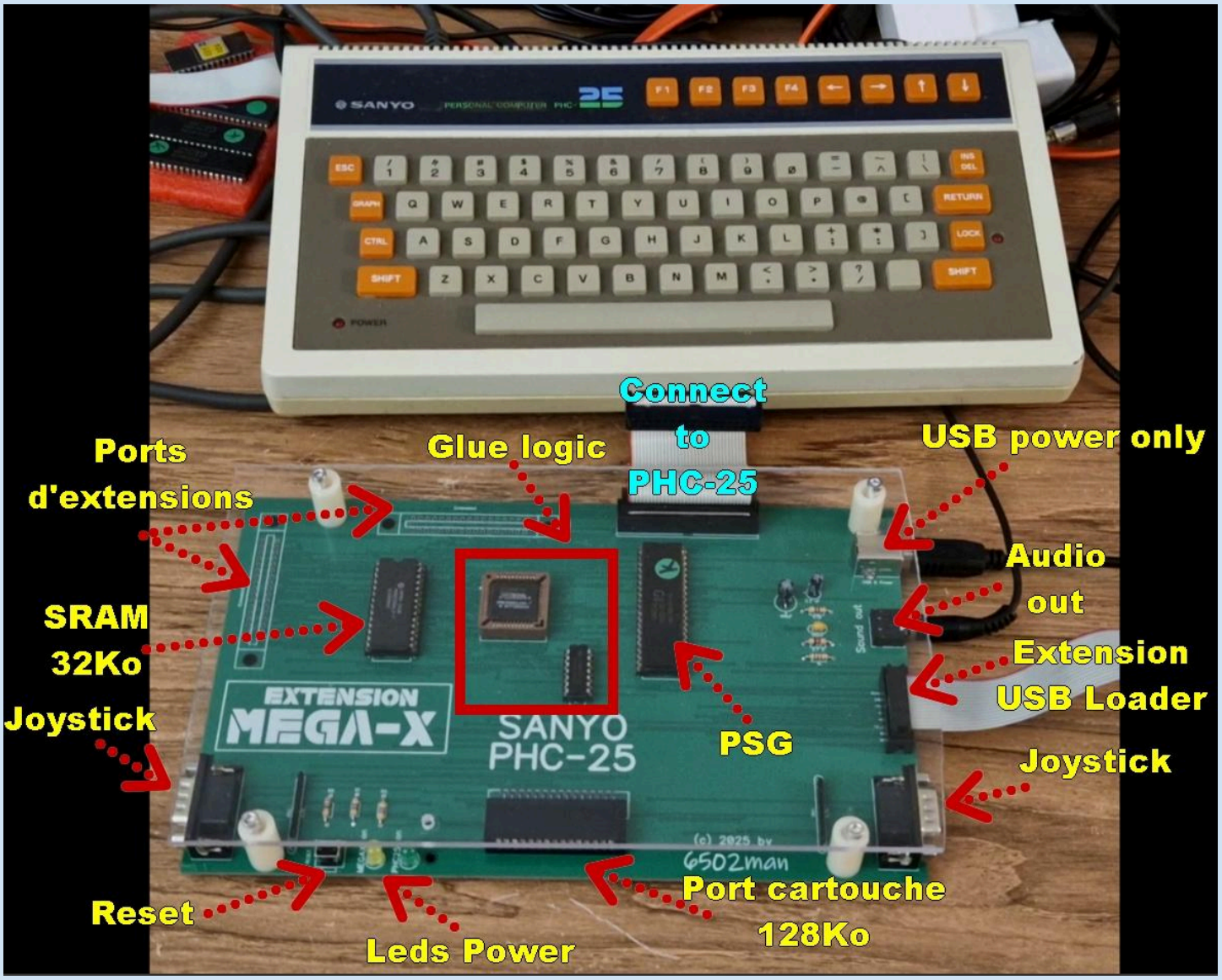
Sanyo PHC-25 Extension MEGA-X
© 2025 by 6502man



Dimensions plexiglas : 26 X 14.5 il faut prévoir de percer 5 trous pour les fixations.
Le patron avec les quotes est fourni dans les pièces jointes.

Cette extension ajoute:

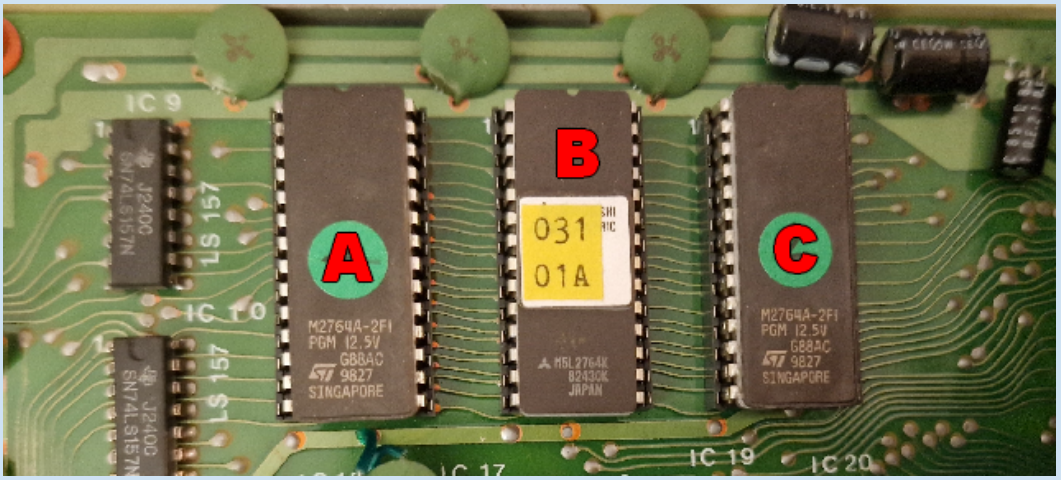
- RAM 32 Ko
- Cartouche ROM 128 Ko
- Connecteur USB loader
- PSG AY-3-8910
- 2 ports Joystick
- Bouton reset
- 2 bus d'extensions supplémentaires
- Ajout dans le BIOS de l'ordinateur de nouvelles fonctionnalités. (uniquement pour la version PAL)

Le mot ROM utilisé dans ce manuel désigne les programmes qui sont stockés sur l'eprom de la cartouche, ce qui est différent du BIOS de l'ordinateur.

Installation :

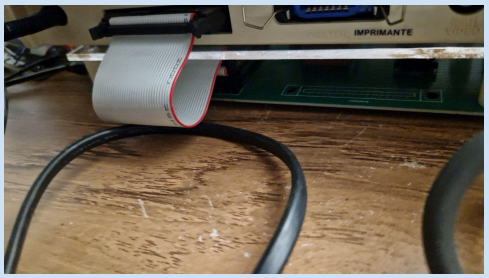
1) le BIOS

Deux nouvelles eproms sont fournies avec de nouvelles fonctionnalités ajoutées, cela ne supprime ou ne modifie aucune fonction du BIOS originale.
Cela est très simple à installer: il vous suffit de dévisser les vis de l'ordinateur, une fois ouvert vous verrez 3 eproms portant une étiquette, il vous suffit d'enlever celle à droite et celle à gauche en vous aidant d'un petit tournevis (avec précaution sans appuyer sur la carte mère), et ensuite d'insérer à la place les eproms A et C (en respectant le sens de l'eprom) comme sur la photo :



2) l'extension

Connecter l'extension MEGA-X au PHC-25 à l'aide de la nappe plate, Les connecteurs sont munis de détrompeurs il n'y a pas de risque d'erreur de sens.

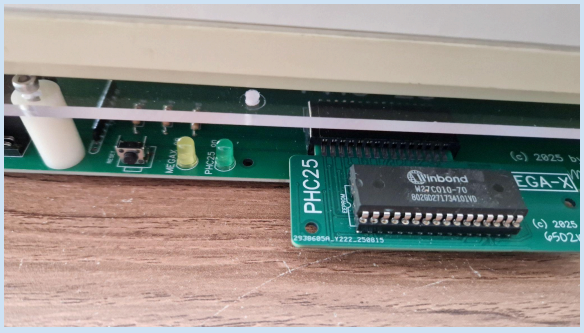


⚠ **ATTENTION**: ne jamais connecter ou déconnecter l'extension si l'ordinateur ou l'extension est connecté au courant, donc toujours débrancher les câbles d'alimentation de l'ordinateur et de l'extension.

Ensuite connecter le câble USB pour alimenter l'extension (une LED s'allume sur l'interface), vous pouvez allumer l'ordinateur (la deuxième LED s'allume sur l'interface).

Vous pouvez utiliser l'ordinateur normalement ainsi que les nouvelles fonctionnalités que cela soit depuis le Basic ou en Assembleur.

3) La cartouche



⚠ Si vous voulez utiliser une cartouche, vous devez impérativement éteindre l'ordinateur et l'extension avant toute insertion où retrait de la cartouche.

La meilleure solution est de vérifier que les 2 leds soient éteintes, pour éviter tout dommage.

La cartouche ne peut s'insérer que dans un sens, donc il n'y a pas de risque d'erreur.
Il est possible de créer des cartouches autoboot ou non, s'il y a présence d'un flag "K" en premier octet de la cartouche l'ordinateur boot automatiquement sur la cartouche à l'adresse $8001, sinon l'ordinateur boot normalement sur le Basic (vous pouvez toujours accéder à la cartouche depuis le Basic ou Assembleur).

Cartographie de l'extension :
RAM et ROM = $8000 - $9FFF
PSG = PORT $C0 et $C1
Sélection et Bank switching RAM et ROM = PORT $F0 - $F2
USB Loader = $F8 (datas) $F9 (statuts)

Utilisation :

- PSG

Sous Basic l'usage est identique qu'avec l'extension PSG-01 d'origine, donc toutes les commandes fonctionnent normalement (SOUND, PLAY, STICK,).

En assembleur utiliser les ports $C0 et $C1

Pour les manettes sous BASIC les commandes normales fonctionnent pour la manette 1 (STICK, STRIG).
La manette 2 n'est accessible qu'en assembleur.
Une routine a été ajoutée au nouveau BIOS pour pouvoir simplement interroger les 2 manettes (détails plus loin).

- RAM/ROM

L'espace adressable est compris entre $8000 et $9FFF.
32 KO pour la RAM partitionné par bloc de 8 KO.
128 KO pour la ROM partitionné par bloc de 8 KO.
Les 2 sont placés aux mêmes adresses, donc il faut sélectionner soit la RAM soit la ROM selon les opérations que l'on veut faire.

Au démarrage de l'ordinateur par défaut, la ROM est sélectionnée sur la page 0.

Pour choisir les pages de RAM et ROM il faut utiliser le port $F0, en tenant compte de l'organisation des bits suivante :

Bits	7	6	5	4	1	2	1	0
Usage	x	RAM	RAM	x	ROM	ROM	ROM	ROM
Valeur du bit (DEC)	128	64	32	16	8	4	2	1
Valeur du bit (HEX)	$80	$40	$20	$10	$08	$04	$02	$01

En Basic :
OUT (&HF0),34 (sélectionne page 1 de la RAM et page 2 de la ROM)

Pour sélectionner la RAM il suffit d'utiliser le port $F1, pour sélectionner la ROM il suffit d'utiliser le port $F2, dans ces 2 cas-là la valeur n'a aucune importance.

Donc en Basic cela se fait simplement par la commande suivante :
OUT (&HF1),0 pour sélectionner la RAM
OUT (&HF2),0 pour sélectionner la ROM

Voici un tableau pour vous aider :
Les valeurs indiquées sont en décimal suivie de la valeur hexadécimale, il suffit de choisir soit l'une ou l'autre.
Bien penser à additionner les valeurs pour la RAM et la ROM, sinon l'une ou l'autre sera sur la page 0,
exemple :
Si l'on veut utiliser la page 3 de la RAM et la page 10 de la ROM il suffit d'écrire la valeur 96 + 10 sur le port $F0, ou $60 + $0A en hexadécimal.
Et si juste après l'on veut utiliser la page 1 de la RAM sans changer la page de la ROM, il faut bien penser à garder la valeur de la page ROM, donc il suffirait d'écrire la valeur 32+10 ($20+$0A), cela permet de sélectionner la page 1 de la RAM en gardant la page 10 de la ROM sélectionnée.

	RAM	**ROM**
PAGE 0 : (valeur à écrire dans le port $F0)	0	0
PAGE 1 :	32/$20	1/$01
PAGE 2 :	64/$40	2/$02
PAGE 3 :	96/$60	3/$03
PAGE 4 :	*n'existe pas*	4/$04
PAGE 5 :	*n'existe pas*	5/$05
PAGE 6 :	*n'existe pas*	6/$06
PAGE 7 :	*n'existe pas*	7/$07
PAGE 8 :	*n'existe pas*	8/$08
PAGE 9 :	*n'existe pas*	9/$09
PAGE 10 :	*n'existe pas*	10/$0A
PAGE 11 :	*n'existe pas*	11/$0B
PAGE 12 :	*n'existe pas*	12/$0C
PAGE 13 :	*n'existe pas*	13/$0D
PAGE 14 :	*n'existe pas*	14/$0E
PAGE 15 :	*n'existe pas*	15/$0F

- **ROM (Cartouche)**

Vous pouvez créer vos propres cartouches en y logeant programmes ou donnés jusqu'à 128 ko (partitionné en page de (8 Ko) .
Les programmes ou données sont vus par le CPU sur la plage de ($8000 à $9FFF), accessible par le Basic ou assembleur.
La cartouche peut être rendue auto-bootable pour cela il suffit de rajouter la lettre K en tout début.
Sinon la cartouche ne sera pas auto-bootable mais accessible quand même par le Basic ou assembleur aux adresses $8000-$9FFF.

- **USBLoader**

C'est une petite carte additionnelle qui est connectée à la MEGA-X par une petite nappe, cela permet de charger des programmes directement depuis un PC par le port USB.

Il est possible d'utiliser l'USB loader directement en Basic ou Assembleur par les ports $F8 et $F9,
Une routine est intégrée dans le nouveau BIOS pour charger des programmes (détails plus loin).

Un programme Windows créé spécialement pour l'USB loader permet de transférer des programmes facilement,
Il intègre aussi le moyen de convertir un programme binaire pour le charger par USB.

Les nouvelles routines du BIOS : (PAL uniquement)

En basic vous pouvez directement utiliser la commande ULOAD pour charger un programme avec l'USBloader.

En assembleur vous pouvez aussi utiliser ces routines à ces adresses :

$5FD0 = **INPUTJOY**
Interrogation des joysticks le résultat est inscrit dans le registre BC.
B = Joystick 1
C = Joystick 2
Bit 0-3 = directions Bit 4-5 = boutons

$5FD3 = **USBLOAD**
Pour un programme à charger il doit y avoir une entête avant les données utiles,
L'entête est dans ce format :
octet 0 : type de fichier (binaire exécutable, exécutable Basic, vidéo, …) 01 = programme 20 = vidéo
octets 1 à 2 : adresse de chargement (MSB LSB)
octets 3 à 4 : longueur des données à charger (MSB LSB)
octets 5 à 6 : adresse d'exécution (MSB LSB)

Par exemple si vous avez un programme assembleur chargeable en $C200, exécutable en $DFFF de longueur $1DEF
l'entête sera : 01 C2 00 1D EF DF FF

Pour les fichiers vidéo l'entête ne comporte qu'un octet les données utiles commencent directement à l'octet 1.

En cas d'anomalies :
Si la LED MEGA-X ne s'allume pas, vérifiez que vous avez bien connecté le câble USB à une alimentation 5V (1A suffit largement).

Si lorsque vous allumez l'ordinateur vous obtenez un écran vide avec un fond coloré, ou des glitchs à l'écran c'est que l'extension n'est pas allumée:
éteindre l'ordinateur, alimenter l'extension (une LED doit s'allumer), puis rallumer l'ordinateur.

Si vous n'obtenez pas de son, vérifiez que vous avez bien connecté des enceintes ou haut-parleurs sur la prise audio de l'interface, et quelles soient bien actives.

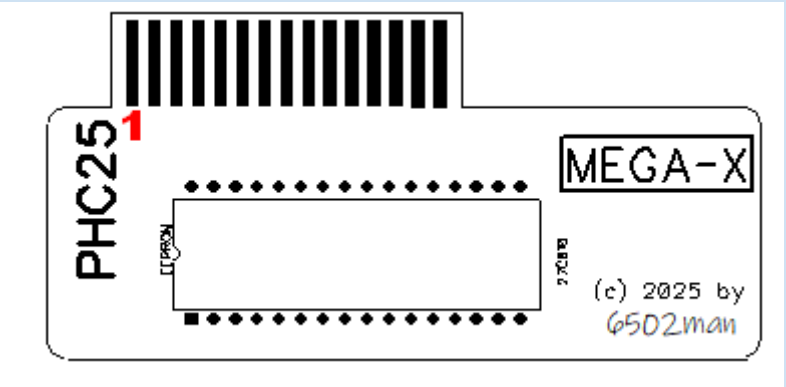
Si les manettes ne réagissent pas vérifiez que vous utilisez des manettes du type Atari.

Version 20260129

ANNEXES

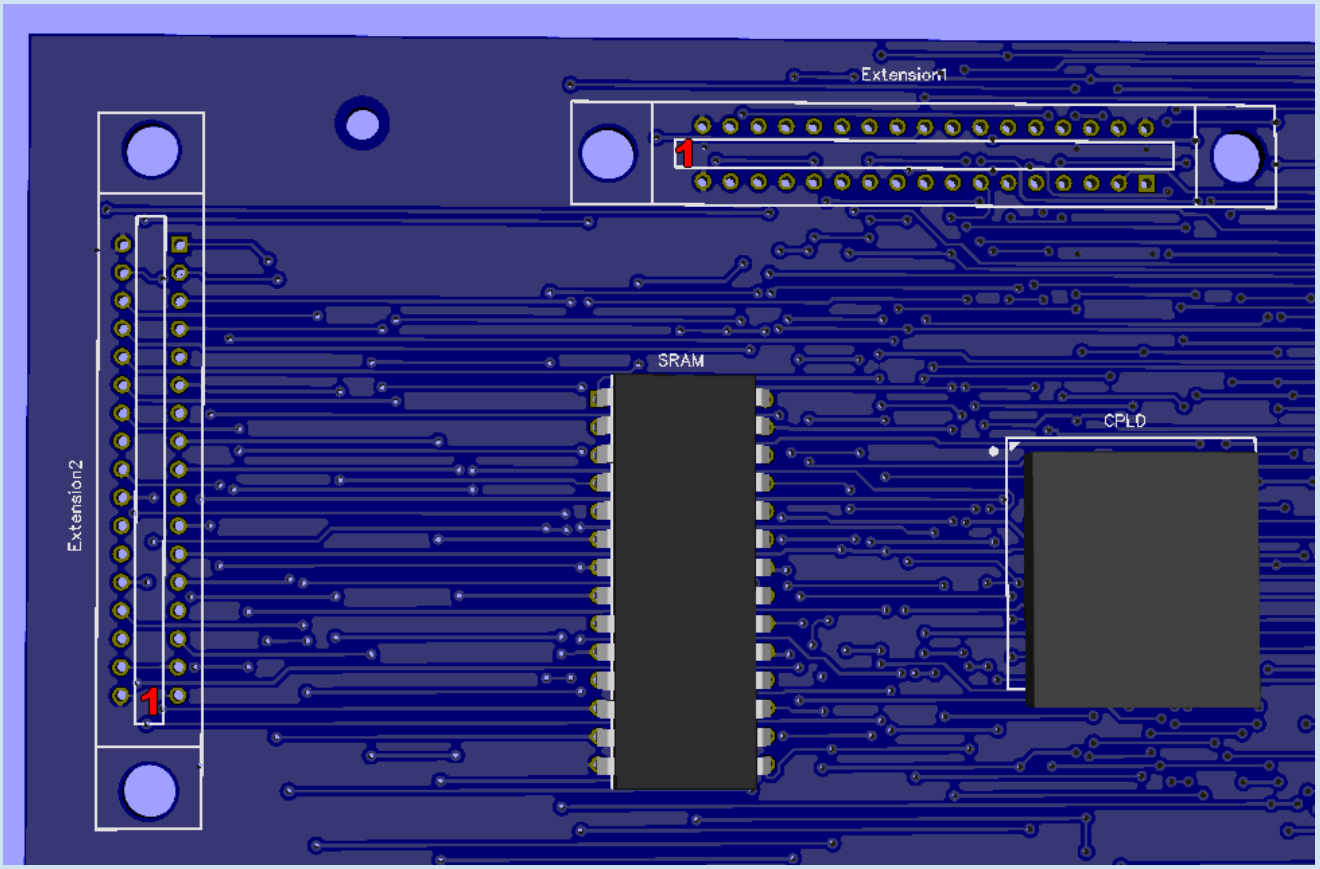
PINOUT cartouches

Haut	/CS	D0	D2	D4	D6	A0	A2	A4	A6	A8	A10	A12	CA13	CA15	NC
pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bas	VCC	D1	D3	D5	D7	A1	A3	A5	A7	A9	A11	/RD	CA14	CA16	GND



PINOUT bus extension (MEGA-X)

Haut	GND	VCC	Reset	/IORQ	D4	D6	IOCS4	A0	A2	A4	A8	A6	A10	A12	/ROMRAMCS	D2	D0
pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Bas	GND	VCC	/MEMRQ	/RD	D5	D7	Clk4	A1	A3	A5	A7	A9	A11	/WR	D3	D1	GND



Les sources assembleurs des nouvelles routines :

Une table de vecteurs à était ajouté en fin de ROM, pour l'instant elle ne contient que 2 jumps :

$5FD0 = **INPUTJOY**

```
        JP $5FB0        ;[ 1e] $5FD0  Joystick states (4 directions + 2 boutons)
```

$5FD3 = **USBLOAD**

```
        JP $5E40        ;[168] $5FD3  USB load
```

INPUTJOY

```
PSGREG          .EQU $C1
PSGDAT          .EQU $C0

        .ORG $5FB0

        ; IN: null
        ; OUT : B=joy1 C=joy2
        ; destroy  A, B ,C

                LD A,7
                OUT (PSGREG),A     ; write registre
                IN A,(PSGDAT)
                AND $3F          ;OR $C0
                LD B,A
                LD A,7
                OUT (PSGREG),A     ; write registre
                LD A,B
                OUT (PSGDAT),A
                ;
                LD A,$0E
                OUT (PSGREG),A    ; write registre
                IN A,(PSGDAT)
                LD B,A
                ;
                LD A,$0F
                OUT (PSGREG),A    ; write registre
                IN A,(PSGDAT)
                LD C,A
        RET
```

USBLOAD

```
USB_Data        .EQU $F8
USB_Status      .EQU $F9

        .ORG $5E40
        DI
        LD ($FF80),SP
        LD SP,$FFEF

                PUSH AF
                PUSH BC
                PUSH DE
                PUSH HL

        CALL ClearScreen

        LD HL,ConnectPC
        LD DE,$6000
        CALL PrintTexte
WaitConnectPC
        IN A,(USB_Status)
        AND $01         ; Interface connectée
        CP $00
        JR NZ,WaitConnectPC

        LD HL,Texte
        LD DE,$6020
        CALL PrintTexte
WaitStartingDatas
        IN A,(USB_Status)
        AND $04         ; données disponible
        CP $00
        JR NZ,WaitStartingDatas

        LD HL,Receiv
        LD DE,$6040
        CALL PrintTexte

        CALL WaitRXF
        IN A,(USB_Data)
        CP $01          ; executable binaire
        JR Z,LoadExec
        CP $20 ; video
        JP Z,Video
        JP ERRORHEADER

LoadExec
        CALL WaitRXF
        IN A,(USB_Data)
        LD D,A                  ; adresse de chargement
                CALL WaitRXF
        IN A,(USB_Data)
        LD E,A                  ; adresse de chargement
                CALL WaitRXF
        IN A,(USB_Data)
        LD B,A                  ; longueur des données à charger
                CALL WaitRXF
        IN A,(USB_Data)
        LD C,A                  ; longueur des données à charger
                CALL WaitRXF
```

```
        IN A,(USB_Data)
        LD H,A                  ; adresse d'exécution
                CALL WaitRXF
        IN A,(USB_Data)
        LD L,A                  ; adresse d'exécution

LoopLoading:
                CALL WaitRXF
        IN A,(USB_Data)
        LD (DE),A

        INC DE
        DEC BC
                LD A,B
                OR C
        JR NZ,LoopLoading



        LD A,H
        LD B,L
        OR B
        JR Z,BasicReturn
        JP (HL)
BasicReturn
                LD SP,($FF80)
                POP HL
                POP DE
                POP BC
                POP AF
                ;EI
                RET



ERRORHEADER
        LD HL,HeadFault
        LD DE,$6040
        CALL PrintTexte
VideCacheF245
        IN A,(USB_Data)
        IN A,(USB_Status)
        AND $04         ; données disponible si RXF => Low
        CP $00
        JR Z,VideCacheF245
Key
                in A,($81)              ; colonne 1
                BIT 05,A
                JR NZ,Key
        JR BasicReturn

ConnectPC
        .DB "CONNECT USBloader TO PC"
        .DB 00
Texte
        .DB "WAITING PC"
        .DB 00
Receiv
        .DB "RECEIVING DATAS"
        .DB 00
```

```
HeadFault
        .DB "NO HEADER  PUSH RETURN"
        .DB 00

PrintTexte
        LD A,(HL)
        INC HL
        LD (DE),A
        INC DE
        CP $00
        JR NZ,PrintTexte
        RET


ClearScreen
        LD HL,$6000
        LD D,$20
        LD BC,20*32
LoopClr
        LD (HL),D
        INC HL
        DEC BC
        LD A,B
        OR C
        JR NZ,LoopClr
        RET


WaitRXF
        IN A,(USB_Status)
        AND $04         ; données disponible si RXF => Low
        CP $00
        JR NZ,WaitRXF
        RET


;-------------------------------------------------
;  read Video file
Video
        LD A,$F6
        OUT ($40),A     ; graphique mode 256x192
        CALL ClearScreenHR

WaitStartingVideoDatas
        IN A,(USB_Status)
        AND $04         ; données disponible
        CP $00
        JR NZ,WaitStartingVideoDatas

LoopFrame
        LD IX,$6000             ; start ecran
LoopDatas
        CALL WaitRXF
        IN A,(USB_Data)
        CP $00
        JR Z,FIN
        CP $FF
        JR Z,WaitSync
                        LD D,0
                        LD E,A
                        ADD IX,DE
```

```
        CP $FE
        JR Z,LoopDatas
DeplacAndWrite
        CALL WaitRXF
        IN A,(USB_Data)
        LD (IX),A
        JR LoopDatas

FIN
        JR FIN



;--------------------------------------
; octet $FF => synchro frame
WaitSync
   IN A,($40)
   BIT 4,A
   JR NZ,WaitSync
IntreSTART
   IN A,($40)
   BIT 4,A
   JR Z,IntreSTART
        JP LoopFrame


; efface l'écran graphique
ClearScreenHR
        LD HL,$6000
        LD D,$00
        LD BC,$1800
LoopClrHR
        LD (HL),D
        INC HL
        DEC BC
        LD A,B
        OR C
        JR NZ,LoopClrHR
        RET
```