Tutoriel	**AmigaCracking : Turrican 3**
Protection	**MFM (multi file)**
Auteur Original	**aLpHa oNe**
Soumit par (sur flashtro.com)	**aLpHa oNe [2004-09-12]**
Version	**19/03/2018 Gi@nts**
Vérification/Correction	**V2, Testé et fonctionnel de A à Z**

* TURRICAN III CRACK TUTORIEL *

Table des matières

Matériels nécessaires

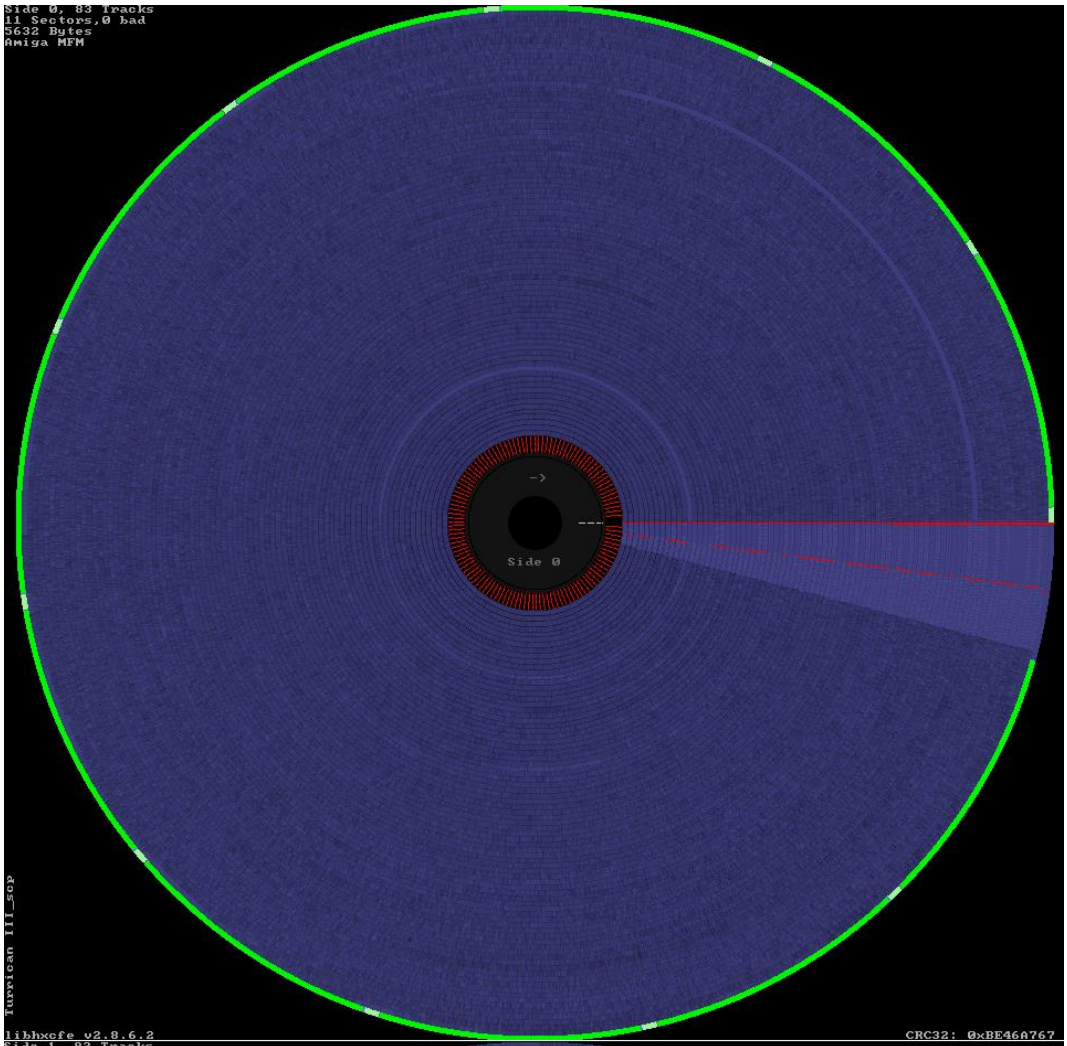
1) Un AMIGA ou l'émulateur WINUAE avec un configuration mémoire conseillé de 2MB de Chip*
   Les 2MB vont nous servir à ripper en un seul passage un nombre maximum de donnée.
   Il est toujours possible d'effectuer ce crack avec moins de mémoire mais cela nécessitera plus de manipulation et une adaptation des instructions du tutoriel.
2) Une Carte ACTION REPLAY (ou ça ROM Image) selon configuration utilisé.
3) Le jeu Original turrican III ou son image CAPS (SPS 0596 ou SPS 2633)
4) Le logiciel Xcopy-pro en disquette ou son image disk.
5) L'assembleur ASM-One
6) Les connaissances pour utiliser un l'assembleur ci-dessus !
7) Le trackloader d'AlphaOne de 2004, taille du binaire : 404 Octets (Disponible dans ce tuto sous forme Hexa)
   HARDWARE-DISKLOADER (c) ALPHA ONE 2004, ; EASY VERSION -> WITHOUT TRACKCOUNTER.
   /!\ La version 2005 du TrackLoader d'AlphaOne 'MULTIDRIVE VERSION' ne fonctionnera pas avec ce tuto

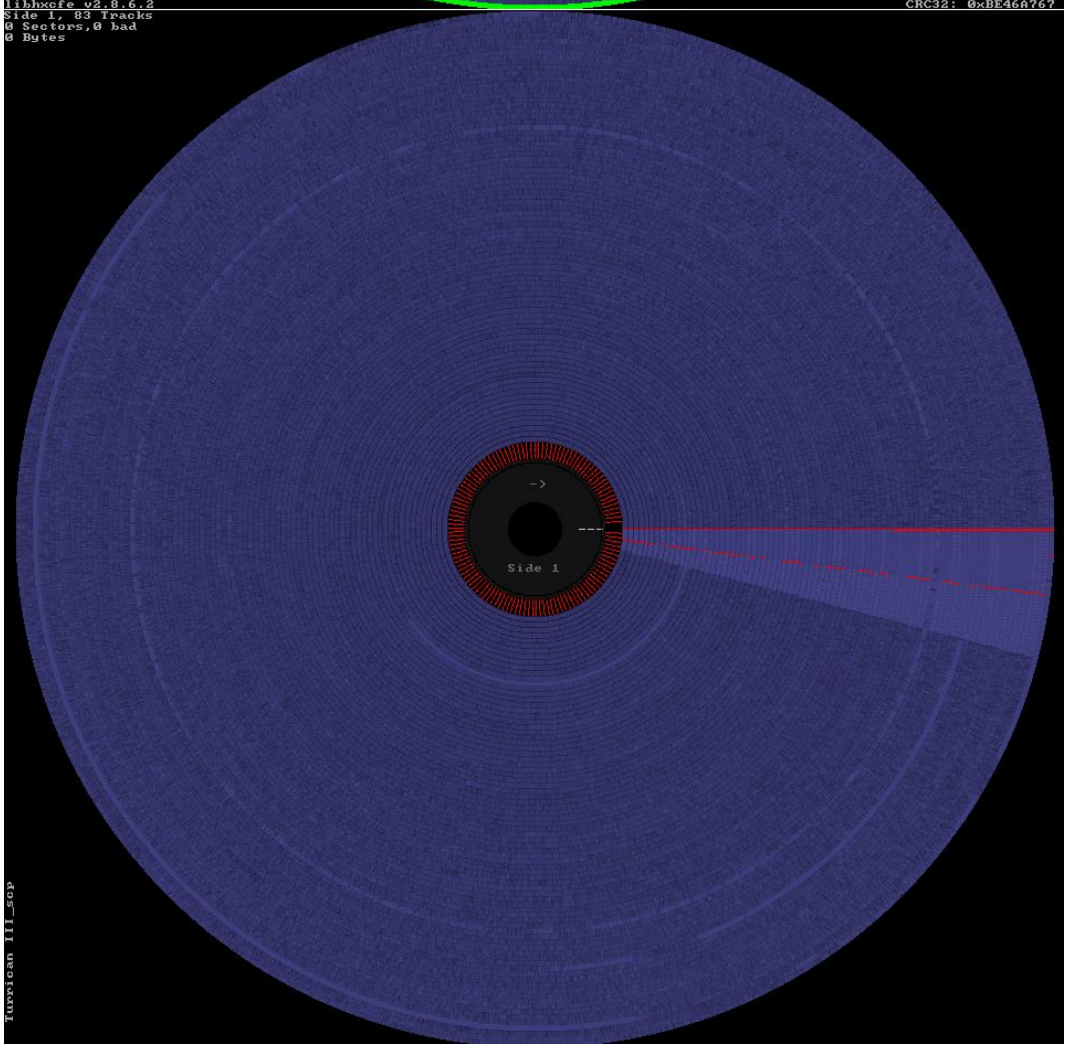
Général Info

Ce tutoriel Français est basé sur le tutoriel original de aLpHa oNe.
Ce document n'est pas une traduction mot par mot de celui-ci mais plus une nouvelle version.
Suivit pas à pas avec des nouvelles informations.

Bon Tuto.

Gi@nts


Side 0, 83 Tracks
11 Sectors,0 bad
5632 Bytes
Amiga MFM
->
Side 0
Turrican III_scp
libhxcfe v2.8.6.2
CRC32: 0xBE46A767

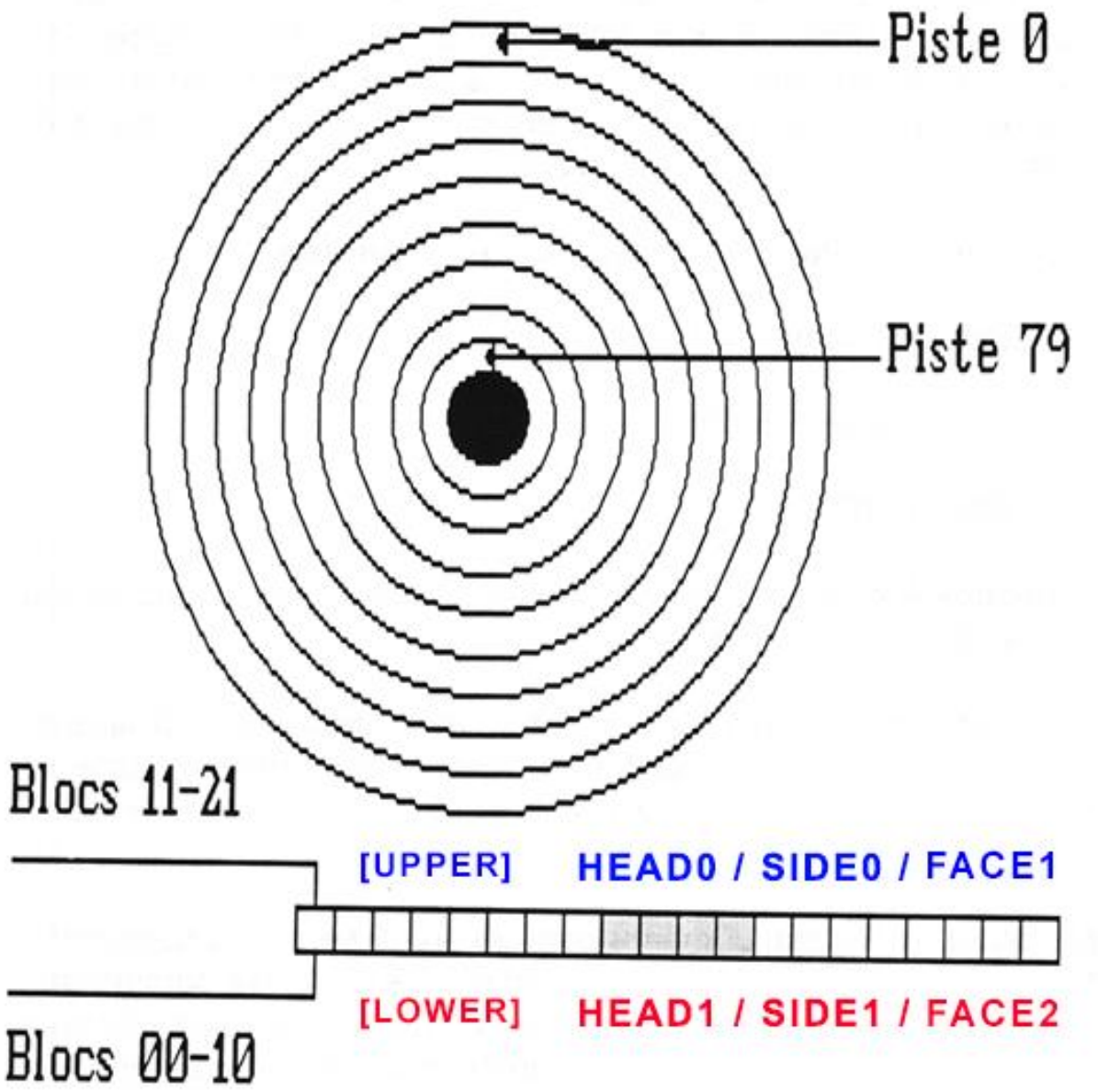

Side 1, 83 Tracks
0 Sectors,0 bad
0 Bytes
->
Side 1
Turrican III_scp

Agencement des disquettes Amiga v1.1

En France :

On utilise des termes comme : *piste, bloc, secteur, face*…
Piste : 0 à 79 Certaine disquette pousse jusqu'à 81 voire 82 pistes mais le standard reste quand même 80 pistes (de 0 à 79)
Face : 0/1 ou **1/2** ou **A/B**, Dessus ou dessous tout simplement. Sur Amiga nous avons deux faces utilisées sur 99% des jeux.

Chaque piste, pour un format standard 'AmigaDOS' est composée de plusieurs *bloc ou secteur*, en général 11 **par face**.
Le terme piste peut désigner l'ensemble d'une piste (les deux 'side' du disque), ou uniquement une 'side' d'une piste.
Une piste standard *amigados* est découpée en plusieurs partie appelé **bloc**, **secteur**, **sector**.



Dans d'autre pays :

On utilisera d'autre terme, comme **sector, keys, tracks, cylindre**, **head**…

Le terme **track** par exemple que l'on aurait vite fait de traduire 'piste' ne colle pas forcément à notre description française.
En général, le terme **tracks** désigne toujours une position sur la disquette mais **elle va de 0 à 159** (soit 160 **tracks**)
Le maximum étant 160 et non 80 car on a deux faces bien sûres, en fait, elle correspond à une piste sur une face.

Il peut néanmoins arriver que l'on utilise dans des tuto anglo-saxon le terme *tracks* dans le sens 'piste' en français
(donc de 0 à 79 et non de 0 à 159). Mais en règle générale, il a plutôt une plage de 0 à 160.

C'est le terme **cylindre** qui 'colle' plus à notre définition française de **piste.**
En effet, il est courant d'utiliser le terme **cylindre** pour désigner une position sur la disquette de 0 à 79.

Le terme **sector** ou **key** quant à lui corresponds au terme français **bloc** ou **secteur.**

Sur une disquette au format ***Amigados***, nous avons 880ko et nous avons 11 secteurs par face, par piste.

La taille d'une piste ayant une valeur physiquement maximum.
Le nombre maximum de **sector** sur une piste dépends assez logiquement de la taille de ses **sectors.**

Pref…beaucoup de terme qui ne sont pas forcément utilisé dans leur sens propre, le mieux est de lire un tuto et de comprendre quel sens l'auteur a voulu leurs donner.

Il existe aussi un autre type d'appellation utilisé par exemple par le logiciel **MFM-Warp** de Ferox*
**C'est un programme qui scan le disque en bas niveau et essaye d'en réaliser une copie.*

Track	Calcul	Résultat	Format utilisé sous MFMWarp
0	0/2	0 et pair	0.0
1	1/2	0 et impair	0.1
2	2/2	1 et pair	1.0
3	3/2	1 et impair	1.1
156	156/2	78 et pair	78.0
157	157/2	78 et impair	78.1
158	158/2	79 et pair	79.0
159	159/2	79 et impair	79.1

On notera que :

Le premier secteur (secteur 0) appelé aussi *bootbloc* commence sur la *lowerSide* en piste 00
et se fini en piste 79 sur le *upperside*

En *tracks* c'est le même système sauf que l'on terminera en Track 179 et non 79.

La piste Zero est celle situé le plus à l'extérieure du disque.

Le 1er secteur logique, donc le premier bloc sur la disquette, se trouve **piste 0 secteur 0**
Les *bloc* se suivent physiquement mais ne sont pas forcément ordonnée, on parle aussi d'entrelacement.

Le bloc 11 (si on part de 0 bien sur) n'est pas le 1er secteur de la seconde piste mais le 1er secteur *de la face suivante.*
(voir image ci-dessus)

En format **Amigados**, **la taille d'un secteur est de 512 octets**
Ce qui nous donnes comme taille disponible : 512*11 secteurs*80 pistes*2 faces = 901 120 octets soit 880Ko
Une 'track' AmigaDos a une taille de 512 * 11 = **5632** en décimal soit ***$1600*** *octets*

Mise en application sous l'AR :

Il existe deux commandes sous l'AR qui permettent de charger sauver des pistes, à savoir : **RT** et **WT**
Elles fonctionnent pareil.
L'une permet la lecture, l'autre l'écriture.

#**RT** alias Read Track. Permet le chargement de donnée située sur la disquette vers la mémoire.
#la première valeur sera la track de **départ** [0 à 159] à indiquer en hexa. **/!\ ne pas confondre avec piste**

#La seconde valeur sera le nbr de demi track à copié à partir de là.

#**WT** alias Write Track. Permet la sauvegarde de donnée située en mémoire vers la disquette.

Exemples :

```
RT 20 1 50000
```
Start Track = $20 et taille à lire = 1
On copiera donc la piste !16 (en décimal) side 0 en mémoire $50000

Oui car **20** est donné en hexa, ce qui nous donnes !32 en décimal **mais** il indique une track (de 0 à 159) **PAS en piste.**
Pour avoir l'équivalent en piste on divisera donc par 2 (car deux faces).

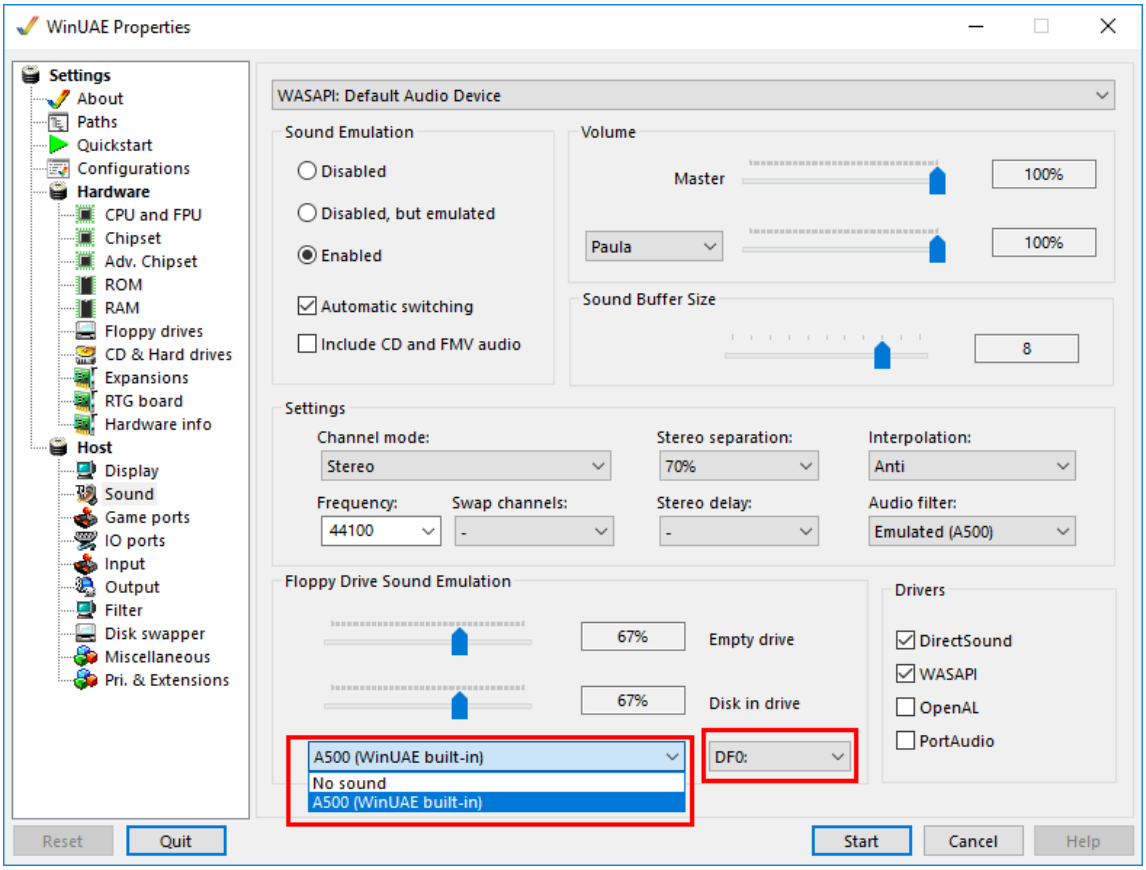
$20/2=$10 = !16 (en décimal donc) et comme il n'y a pas de retenu on est sur la face0.

```
RT 21 2
```
Start Track = $21 et taille à lire = 2
On copiera la piste !16 side 1 et la piste !17 side 0 en mémoire 50000

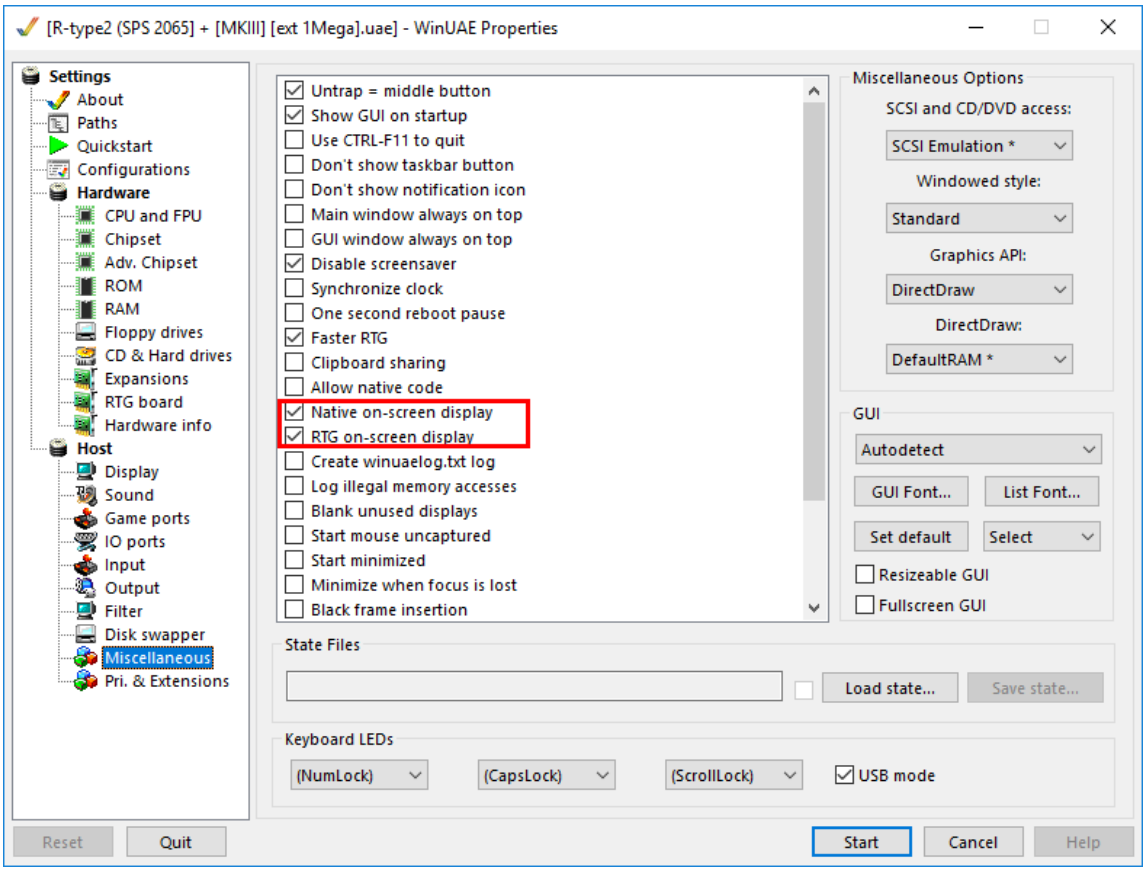
21 est donné en hexa **donc $21 = !33** en décimal.
33/2 = 16.5 ,donc piste 16 side 1 et comme on continue à lire/copier les donnes (**taille à lire =2**), on continue la copie.
On change donc de track car on est déjà sur la face 1 (il existe que 2 faces sur une disquette)
On arrive donc sur la prochaine track à savoir, piste 17 en side 0 puisque que c'est la première face au niveau structure la side 0.

WinUAE

Pour ceux qui utilisent ***winUAE*** pour ces tutoriels (j'imagine, la plupart des personnes), Je vous conseille fortement d'activer le son des lecteurs de disquette histoire d'entendre ce que le lecteur effectue comme accès.
HOST -> SOUND -> FLOPPY DRIVE SOUND EMULATION - > DF0 Built-In

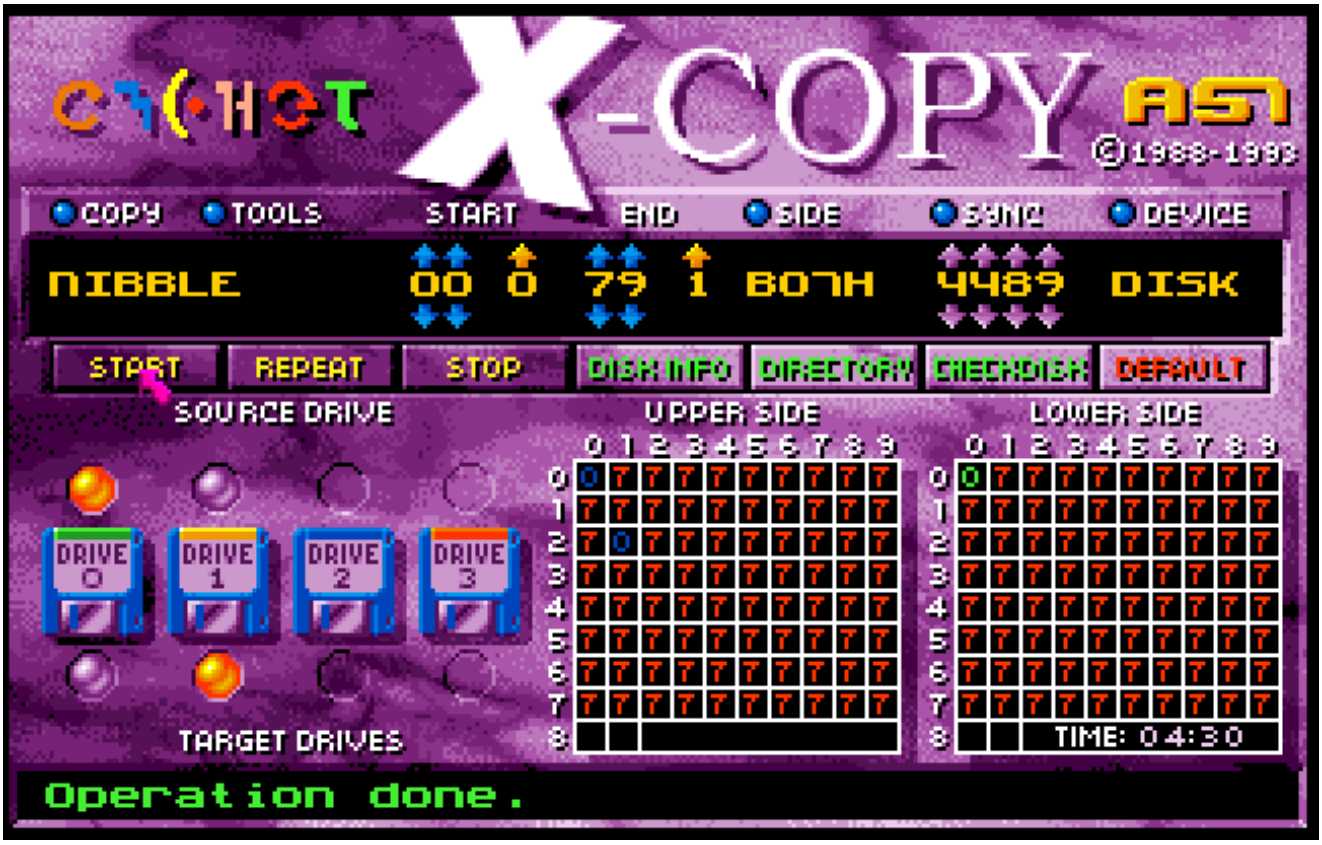


Voir même, pour plus d'information. Par exemple afficher sur qu'elle face l'on se trouve, d'activer :
Host -> Miscellaneous -> Native on-screen display AND RTG on-screen display



Part 1 X-copy

Time to check the protection
On va dans un premier temps essayer de copier la disquette originale à l'aide de Xcopy Pro.



Rappel des codes d'erreur de Xcopy :

1. *Less or more than 11 sectors*
2. *No sync found*
3. *No sync after gap found*
4. *Header checksum error*
5. *Error in header/format long*
6. *Data block checksum error*
7. *Long track*
8. *Verify error*

Part 2 Analyse de l'image IPF

FILENAME	0596_Turrican3.ipf
TYPE	Floppy_Disk
ENCODER	CAPS(V1)
FILE	596(V1)
DISK	0
TRACK	00-83
SIDE	0-1
PLATFORM	Amiga
REVOLUTION	2

Concernant la piste 00.0 rien de bien spécial : 11 secteurs de 512 bytes, standard AmigaDOS

TRACK		Data Length (bytes)		Data (bits)				CRC32 of the complete Extra Data Block			Adress
Data block Description	Sector ID	Data		bytes/sector	GAP		Codage	GapDef	DataOff		Adresse
		MFM bits	bytes		MFM bits	bytes			MFM bits	bytes	
[T00.0]		6446		51568				DC2CEDF7			13576-20021
#0		8704	545		0	1	MFM	0352	0352	15	13576-13607
#1	10	8704	545	512	0	1	MFM	0906	0906	57	13608-13639
#2	0	8704	545	512	0	1	MFM	1460	1460	92	13640-13671
#3	1	8704	545	512	0	1	MFM	2014	2014	126	13672-13703
#4	2	8704	545	512	0	1	MFM	2568	2568	161	13704-13735
#5	3	8704	545	512	0	1	MFM	3122	3122	196	13736-13767
#6	4	8704	545	512	0	1	MFM	3676	3676	230	13768-13799
#7	5	8704	545	512	0	1	MFM	4230	4230	265	13800-13831
#8	6	8704	545	512	0	1	MFM	4784	4784	300	13832-13863
#9	7	8704	545	512	0	1	MFM	5338	5338	334	13864-13895
#10	8	8704	545	512	4299	269	MFM	5892	5892	369	13896-13927

Ensuite, **toutes les pistes** se ressemblent. A savoir pas de secteur standard, un codage MFM et une taille de $6707 au niveau Data Track
Et ce, jusqu'à la piste 79.1

TRACK		Data Length (bytes)		Data (bits)				CRC32 of the complete Extra Data Block			Adress
[T01.0]		6707		53656				7D095846			26271-32977
Data block Description	Sector ID	Data		bytes/sector	GAP		Codage	GapDef	DataOff		Adresse
		MFM bits	bytes		MFM bits	bytes			MFM bits	bytes	
#0	N/A	106656	6667	N/A	4496	282	MFM	0032	0032	2	26271-26302

Sauf… La piste **21.1** qui elle a une taille de $6193 au niveau Data Track.
Elle contient à coup sur des données différentes des autres pistes et sera surement décodé par un code spécifique.

TRACK	Data Length (bytes)	Data (bits)	CRC32 of the complete Extra Data Block	Adress
[T00.0]	6446	51568	DC2CEDF7	13576-20021
[T00.1]	6193	49544	EFBE51BF	20050-26242
[T01.0]	6707	53656	7D095846	26271-32977
[T01.1]	6707	53656	56937FD4	33006-39712
[T02.0]	6707	53656	8B9A33FD	39741-46447
[T02.1]	6707	53656	133F897C	46476-53182
[T03.0]	6707	53656	21B2CD6B	53211-59917
[T03.1]	6707	53656	389FE5FA	59946-66652
...				
[T19.1]	6707	53656	5339226E	275466-282172
[T20.0]	6707	53656	93798D3E	282201-288907
[T20.1]	6707	53656	93CF6A56	288936-295642
[T21.0]	6707	53656	24C47FC4	295671-302377
[T21.1]	6193	49544	C1E46487	302406-308598
[T22.0]	6707	53656	22150EE6	308627-315333
[T22.1]	6707	53656	D660123E	315362-322068
[T23.0]	6707	53656	09304846	322097-328803
[T23.1]	6707	53656	9316D5F1	328832-335538
[T24.0]	6707	53656	6A711398	335567-342273
[T24.1]	6707	53656	D0C4C079	342302-349008
...				

Ce qui, au passage, valide ce que l'on a déjà vue sous Xcopy :

Part 3 Let's Rock'n Roll

Allumer votre Amiga sans disquette et **Entrer dans votre AR**
Insérer votre original de Turrican III dans le lecteur est **Taper** le texte suivant :
#RT alias Read Track, permet le chargement de donnée à partir de la track 0 et d'une taille de 1
#On lie donc ici uniquement la 1er face de la piste 0 et on copie le tout à l'adresse mémoire 50000

```
rt 0 1 50000
```

Commencer par le désassemblage du bootcode **Taper** le texte suivant et défiler vers le bas, comme dans l'image ci-dessous.
#D, alias Désassemble

```
d 5000c
```

```
Ready.
rt 0 1 50000
Disk ok
d 5000c
~05000C MOVEA.L 00000004.S,A6
~050010 MOVE.W  #2,1C(A1)
~050016 MOVE.L  #400,2C(A1)
~05001E MOVE.L  #58000,28(A1)
~050026 MOVE.L  #1200,24(A1)
~05002E JSR     -1C8(A6)
~050032 MOVE.W  #9,1C(A1)
~050038 CLR.L   24(A1)
~05003C JSR     -1C8(A6)
~050040 LEA     00DFF000,A5
~050046 MOVE.W  #7FFF,9A(A5)
~05004C MOVE.W  #7FFF,96(A5)
~050052 LEA     5006C(PC),A0
~050056 LEA     0007D000,A1
~05005C MOVE.W  #324,D0
~050060 MOVE.B  (A0)+,(A1)+
~050062 DBF     D0,00050060
~050066 JMP     0007D000
;=====================================
~05006C LEA     0004D000,A7
```

Les premières choses intéressantes apparaissent ici... (en rouge dans l'image ci-dessus)
$1200 octets de données sont chargées en utilisant le '*trackdisk-device position*' à l'adresse **$58000**.

C'est la partie ou le logo "**Rainbow Arts**" disparait alors rappelons-nous de ripper cet endroit de la mémoire durant les prochaines étapes du déplombage.

À Titre de comparaison, les mêmes informations pour Turrican 2 et Turrican3

	T3	T2
28(A1) = mémoire destination	58 000	60 000
24(A1) = longueur de ce que l'on va copier	1200	800
2c(A1) = début de la position de lecture	400	400

Continuons notre désassemblage sur quelques lignes.

```
~050060 MOVE.B  (A0)+,(A1)+
~050062 DBF     D0,00050060
~050066 JMP     0007D000
;=====================================
~05006C LEA     0004D000,A7
~050072 MOVE.W  #0,00DFF180
~05007A LEA     50380(PC),A4
~05007E BSR     00050328
~050082 MOVEQ   #0,D0
~050084 BSR     000502B2
~050088 BSR     000501EC
~05008C BSR     00050278
~050090 MOVEQ   #0,D0
~050092 BSR     000501D2
~050096 MOVE.B  #0,B(A4)
~05009C BSR     000502C0
~0500A0 MOVE.L  #7E800,0(A4)
~0500A8 MOVE.L  #1800,4(A4)
~0500B0 BSR     000500C4
~0500B4 BSR     0005029A
~0500B8 MOVE.W  #7FFF,96(A5)
~0500BE JMP     0007E800
;=====================================
~0500C4 MOVE.W  #1002,9A(A5)
~0500CA MOVE.W  #8210,96(A5)
```

On dirait bien qu'il y a un petit chargeur de piste (*trackloader*) planqué dans le code de boot
qui charge **$1800** octets de données à l'adresse **$7E800**, suivi par un saut (jmp) en **$7E800**.

C'est probablement... naan !! ... **c'est** le chargeur principal (*mainloader*) du jeu.
Donc comme toujours on va remplacer le *'jmp'* avec une boucle de branchement (*branch-loop*)
pour aller au contact avec les data chargée en **$58000** et **$7E800**

Pour commencer, **insérer** une disquette vierge et sauvons tout ça, **Taper** :
#format, Permet de formater une disquette au format AmigaDOS
#SM, alias SaveMemory permet donc de sauver une zone mémoire vers un fichier.

```
format Turrican3_1
Ready to format disk in drive DF0: (y/n)? y
Disk ok

SM bootblock, 50000 50000+400
SM loader, 50400 50400+1200
```

Réinsérer votre original de Turrican III dans le lecteur et modifier maintenant comme prévu le code et insérons notre (*branch-loop*).

Taper :
#A, alias Assemble, Instruction qui va permettre de taper du code assembleur.

```
a 500be
^0500BE BRA 500BE
^0500C0 NOP
^0500C2 NOP
^0500C4
```

On re-calcul le checksum de boot et on re-écrie le tout sur la disquette.
Pas de panique, même si on est sur un disque original, on ne ré-écrie que la piste AmigaDos R/W présente sur la disquette.
En clair, on peut revenir en arrière sans aucun soucis 😉

#BOOTCHK, alias BootChecksum. Permet de calculer le checksum d'un bootblock en mémoire.

#WT alias Write Track, permet de copier une zone mémoire sur la disquette. On indique une track de départ et une taille
#ici on va copier les donnée mémoire située en $50000 sur la track 0 side0

```
bootchk 50000
wt 0 1 50000
```

```
a 500BE
^0500BE bra 500BE
^0500C0 nop
^0500C2 nop
^0500C4

bootchk 50000
Old checksum was 35562C3D, now is set to E6ECB3C6
wt 0 1 50000
Disk ok
_
```

Une petite vérification aux adresses **$58000** et **$7E800** avant de relancer le tout.
Taper :
#M alias memory read. Permet de voir les données en mémoire au format HEXA/ASCII.
`m 58000`
`m 7E800`

Bien, complément vide !

Redémarrer votre amiga.

Après un très court chargement (au bruit, 1 piste), l'écran passe et reste au noir, nous devrions être dans notre boucle.
Entrer dans votre AR.
Taper : `d`

Effectivement, c'est le cas, nous somme dans notre boucle.

Jetons un coup d'œil de plus près aux adresses **$58000** et **$7E800**. Entrer dans votre Action replay et **taper :**
Taper :

```
m 58000
m 7e800
```

```
M 7E800
:07E800 33 FC 00 20 00 DF F1 DC 61 00 04 0C 42 B8 03 00 3ü. .ß.üa...B...
:07E810 08 F9 00 01 00 BF E0 01 20 78 00 04 51 F8 02 22 .........x..Q.."
:07E820 0C 28 00 7F 01 29 66 04 50 F8 02 22 33 FC 7F FF .(.▒.)f.P.."3ü▒.
:07E830 00 DF F0 96 42 B8 02 00 21 FC 00 07 EB E6 00 80 .ß..B...!ü......
:07E840 4E 40 4F F8 05 80 4E B9 00 07 F0 00 70 06 61 00 N@O...N.....p.a.

M 58000
:058000 60 00 00 06 60 00 00 4E 41 FA 10 F4 43 F9 00 05 `...`..NA...C...
:058010 81 9A 45 F9 00 05 AE 4E 61 00 01 18 41 FA 01 04 ..E....Na...A...
:058020 42 98 42 98 42 98 42 98 61 00 00 88 4B F9 00 DF B.B.B.B.a...K..ß
:058030 F0 00 41 FA 01 22 2B 48 00 80 3B 7C 83 80 00 96 ..A.."+H..;|....
:058040 21 FC 00 05 80 80 00 6C 3B 7C C0 20 00 9A 41 FA !ü.....l;|. ..A.
```

Des choses semblent s'être chargées alors sauvons-les sur notre disquette de sauvegarde.
Les données en **$58000** ont une longueur de **$1200** octets et les data en **$7E800** font **$1800** octets.
(on le sait grâce au code du boot, voir plus haut.).

Réinsérer votre disquette de sauvegarde et **Taper**

```
SM 7E800, 7E800 7E800+1800
```

Disk ok

```
SM 58000, 58000 58000+1200
```

Disk ok

```
sM 7E800, 7E800 7E800+1800
Disk ok

SM 58000, 58000 58000+1200
Disk ok
```

Allons **jeter** un coup d'œil maintenant au code chargé en *$7E800*, **Taper** :

```
d 7E800
```

```
d 7e800
~07E800 MOVE.W  #20,00DFF1DC
~07E808 BSR     0007EC16
~07E80C CLR.L   00000300.S
~07E810 BSET    #1,00BFE001
~07E818 MOVEA.L 00000004.S,A0
~07E81C SF      00000222.S
~07E820 CMPI.B  #7F,129(A0)
~07E826 BNE     0007E82C
~07E828 ST      00000222.S
~07E82C MOVE.W  #7FFF,00DFF096
~07E834 CLR.L   00000200.S
~07E838 MOVE.L  #7EBE6,00000080.S
~07E840 TRAP    #0
~07E842 LEA     00000580.S,A7
~07E846 JSR     0007F000
~07E84C MOVEQ   #6,D0
~07E84E BSR     0007EB3A
~07E852 MOVE.L  #7EB4E,00000080.S
~07E85A TRAP    #0
~07E85C TST.B   00000224.S
~07E860 BEQ     0007E866
~07E862 CLR.L   00000200.S
~07E866 LEA     00000580.S,A7
~07E86A MOVE.L  #7EBE6,00000080.S
~07E872 TRAP    #0
~07E874 MOVEQ   #7,D0
~07E876 BSR     0007EB44
~07E87A TST.B   0007EC15
~07E880 BEQ     0007E8AC
~07E882 MOVE.L  #7EB70,00000080.S
~07E88A TRAP    #0
~07E88C MOVE.W  #7FFF,00DFF09A
~07E894 MOVE.W  #7FFF,00DFF09C
~07E89C MOVE.W  #7FFF,00DFF096
~07E8A4 JSR     00058000
~07E8AA BRA     0007E8B4
;=====================================
~07E8AC BSR     0007EA66
~07E8B0 LEA     00000000.S,A0
~07E8B4 MOVEQ   #8,D0
~07E8B6 JSR     0007F004
~07E8BC TST.B   0007EC15
```

On peut voir quelque d'intéressant en **$7E8A4**
On saute dans un sous-programme situé en **$58000** (notre seconde floppé de DATA chargée), puis continue sa routine en **$7E8B4**
Change **D0** et sautes dans un sous-programme en **$7F004**
Et en **$7F004** saute encore pour continuer sa routine en **$7F020** et le code continue…

```
d 7f004
~07F004 BRA     0007F020
;=====================================

d 7f020
~07F020 CLR.L   0007FDFE
~07F026 CLR.L   0007FE40
~07F02C LEA     00DFF000,A5
~07F032 MOVE.W  2(A5),D7
```

Pourquoi tant de cachoterie ??? Humm, regardons de plus prêt.

Taper :
`d 7F004` suivi de `d 7£020`

```
d 7f004
~07F004 BRA     0007F020
;--------------------------------------

d7f020
~07F020 CLR.L   0007FDFE
~07F026 CLR.L   0007FE40
~07F02C LEA     00DFF000,A5
~07F032 MOVE.W  2(A5),D7
~07F036 ORI.W   #8000,D7
~07F03A MOVE.W  D7,0007FE0E
~07F040 MOVE.L  A0,D7
~07F042 BNE     0007F048
~07F044 LEA     7FE9A(PC),A0
~07F048 MOVE.L  A0,0007FE06
~07F04E MOVE.L  A0,80(A5)
~07F052 MOVE.B  D0,0007FE2E
~07F058 BTST    #6,D0
~07F05C BEQ     0007F062
~07F05E MOVE.B  7FE2E(PC),D0
~07F062 ANDI.W  #1F,D0
~07F066 MOVE.W  D0,D2
~07F068 LEA     7FCBA(PC),A0
~07F06C ASL.W   #4,D0
~07F06E MOVE.W  D0,D1
~07F070 TST.L   C(A0,D1.W)
~07F074 BNE     0007F1EE
~07F078 MOVE.L  8(A0,D1.W),D0
~07F07C MOVE.L  7FE3C(PC),D2
~07F080 BEQ     0007F09C
~07F082 MOVE.L  D2,D3
~07F084 NEG.L   D2
~07F086 ADD.L   7FE38(PC),D2
~07F08A CMP.L   D2,D0
~07F08C BGE     0007F09C
~07F08E MOVE.L  D3,D2
~07F090 ADD.L   D0,D3
~07F092 MOVE.L  D3,0007FE3C
~07F098 BRA     0007F16A
;--------------------------------------
```

En fait, ici, nous avons un chargeur de piste qui semble fonctionne que ceci :

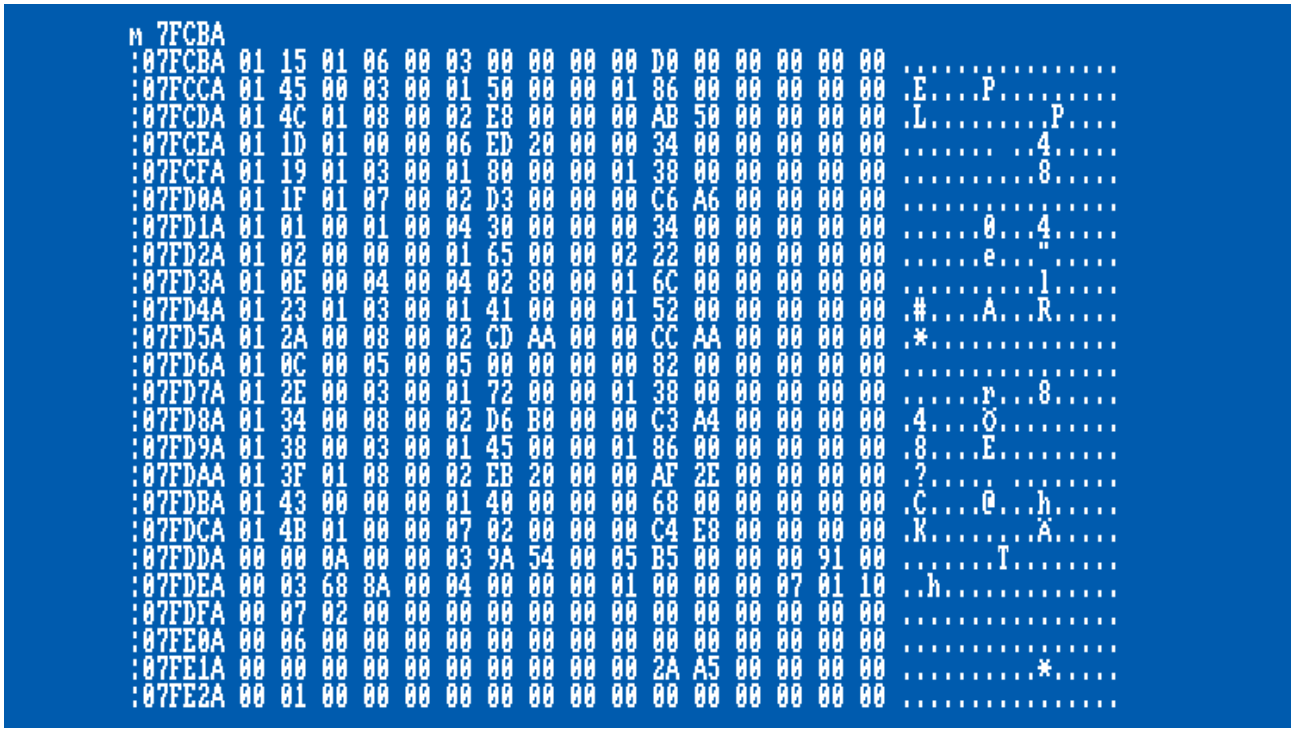
Le chargeur de piste (***trackloader***) est stocké en **$7F020**, fonctionne avec une table de fichier qui est située en **$7FCBA**
et elle est appelée avec une table d'entrée de fichier en **D0**.

Cette valeur est alors multipliée par !**16** avec l'instruction présente en **$7F06C**
asl.l #4,d0

Ainsi, logiquement, chaque entrée dans la table des fichiers devrait avoir une taille de **!16** octets.

Allons donc jeter un coup d'œil dessus sur cette '*filetable*'

Taper :
`M 7FCBA`



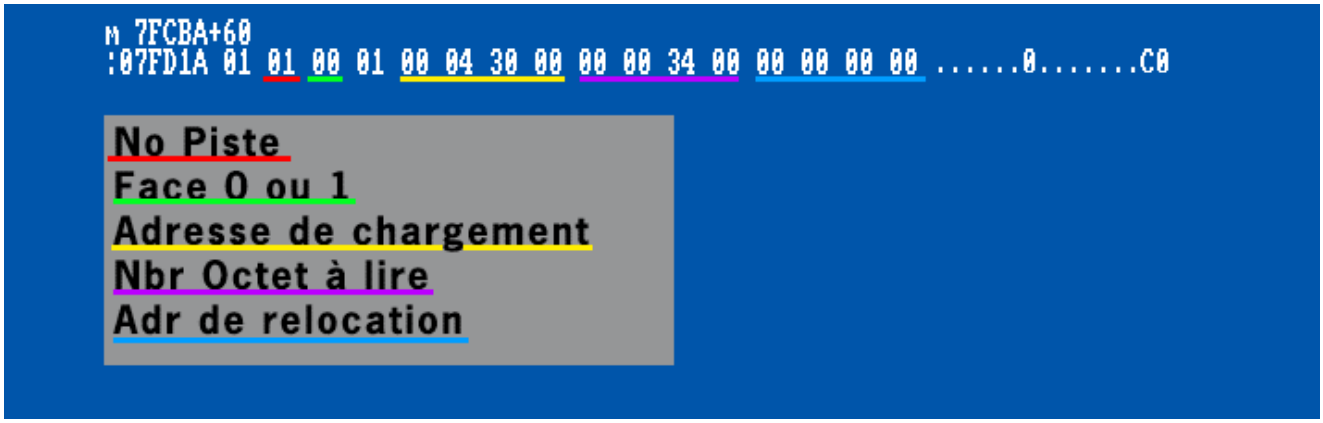
Position (Offset) :

$0.b = Inutile
$1.b = Numéro de piste en hexa
$2.b = Face 0 ou 1 de la piste
$3.b = Aucune idée…
$4.l = Adresse de chargement du Fichier en hexa.
$8.l = nbr Octets à lire en hexa.
$C.l = Adresse de relocation en hexa.

Si vous avez une extension mémoire le jeu alloue la mémoire haute et charge les données à cette adresse au lieu d'utiliser celle à l'offset **$4.l** dans la table.

Cette nouvelle adresse de chargement est stockée ici, suivie par la copie des octets à la bonne adresse à l'offset **$4.l**

Comme ça le jeu vérifie cette valeur plus tard, si ce n'est pas zéro, il copie juste les octets de la mémoire au lieu de charger à nouveau les pistes depuis la disquette. Super idée ! :-)



D'accoordd... utilisons ce que nous savons maintenant pour ripper toute la disquette !
Mettons un breakpoint à l'adresse du trackloader en **$7F020** et changeons certains paramètres de chargement.

Part 4 Ripage des fichiers

Important :
Si vous utilisez une autre configuration mémoire que celle préconisé plus haut, à savoir 2 Mo de Chip et 2 Mo de Fast.
Les adresses peuvent différer de celles utilisées dans ce tutorial.

Ainsi si vous utilisez une autre config vous devrez sauver les pistes chargées depuis un autre endroit de la mémoire. (selon config)

Quelques exemples :

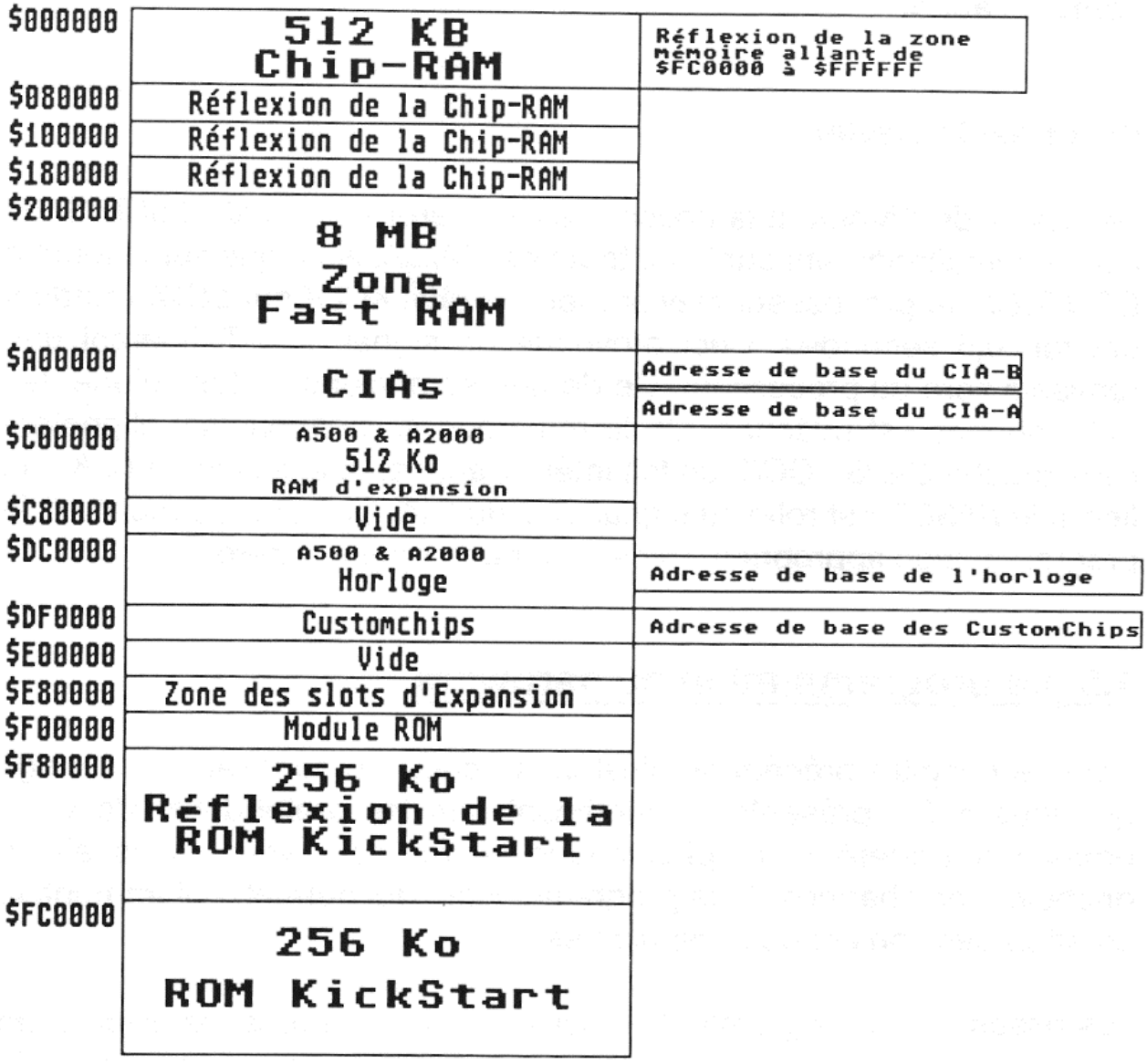
Config avec 512MB Chip uniquement	➔ Data chargée en adr '*filetable*'	*/!\ ne permet pas le rip en une passe.*
Config avec 1024MB Chip uniquement	➔ Data chargée en adr '*filetable*'	*/!\ ne permet pas le rip en une passe.*
Config avec 512MB Chip + 512MB Slow	➔ Data chargée en adr '*filetable*'	*/!\ ne permet pas le rip en une passe.*
Config avec 512MB Chip + 1.8 Mega Slow	➔ Data chargée en $C01809	*_Rip possible en une passe_*
Config avec 2048MB Chip uniquement	➔ Data chargée en $80000	*_Rip possible en une passe_*
Config avec 512MB Chip + 512Slow + 512MB Fast	➔ Data chargée en $C0000 + adr '*filetable*'	*_Rip possible en une passe_*
Config avec 512MB Chip + 512MB Slow + 2048 MB Fast	➔ Data chargée en $C0000 + adr '*filetable*'	*_Rip possible en une passe_*

Petite piqure de rappel concernant l'organisation mémoire de l'amiga

1.5.1 ORGANISATION DE LA MEMOIRE

La Bible AMIGA

Configuration normale



/!\ RETIRER LA DISQUETTE DE BACKUP ET **RE-INSERER** LA DISQUETTE ORIGINAL DE TURRICAN III

Taper :
#BS, alias BreakPoint. Permet, dès que l'adresse mémoire indiquée est atteinte, d'effectuer un arrêt du code.
`bs 7f020`
Breakpoint inserted Ready.

On retourne à l'exécution normal du programme, **Taper** :
#G comme GO, démarre le code à l'adresse indiquée.
`g 7E800`

Après un bref instant de chargement, le *Breakboint* est atteints et on est de nouveau dans notre AR.
Regardons un peu les registres pour voir quelle entrée dans la table de fichier le jeu veut charger.

Taper :
#R permet d'afficher tous les registres du 68000 et/ou de changer des valeurs des registres.
`r`

```
bs 7f020
Breakpoint inserted
Ready.

g 7e800
No known virus in memory!
Ready.
Breakpoint raised at address: 0007F020

r
D0=00000006 00000001 FFFFFFFF FB1EDE68  00F80000 00000003 0000FFFF 0000FFE8
A0=00000000 00060000 0007EC98 00080000  00050380 00DFF000 00000676 000004BA
PC = 0007F020 USP = 0000057C SR = 0000 T=0 S=0 I=000 X=0 N=0 Z=0 V=0 C=0
```

Comme vous le voyez en **D0** ça doit être l'entrée **#6**, donc regardons la position de la table de fichier **$6** * **!16** = **$60**
Une petite vérification, **taper** : `m 7fcba+60`

```
m 7fcba+60
:07FD1A 01 01 00 01 00 04 30 00 00 00 34 00 00 00 00 00 ......0...4.....
```

Il veut charger **$3400** octets à l'adresse **$43000**, commençant au piste **01**, face **00**.
C'est une super position de départ donc pas besoin de changer ça.
Mais comme vous l'avez deviné on ne veut pas ripper seulement **$3400** octets alors changeons ce paramètre,
ex : en **$F0000** (!**983040**) octets.

Retaper :`m 7fcba+60` Et maintenant **modifier** la valeur comme indiqué dans la l'image ci-dessous.

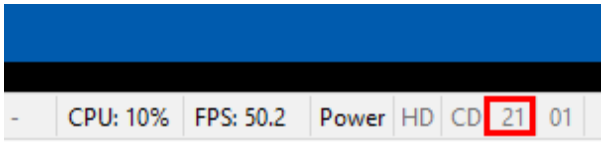
```
m 7fcba+60
:07FD1A 01 01 00 01 00 04 30 00 00 00 34 00 00 00 00 00 ......0...4.....

m 7fcba+60
:07FD1A 01 01 00 01 00 04 30 00 00 0F 00 00 00 00 00 00 ......0...4.....
:07FD2A 01 02 00 00 00 01 65 00 00 02 22 00 00 00 00 00 ......e..."......

x
No known virus in memory!
Ready.
```

Fais-le en pas à pas comme montré dans l'image au-dessus et continuer le jeu en quittant l'AR.

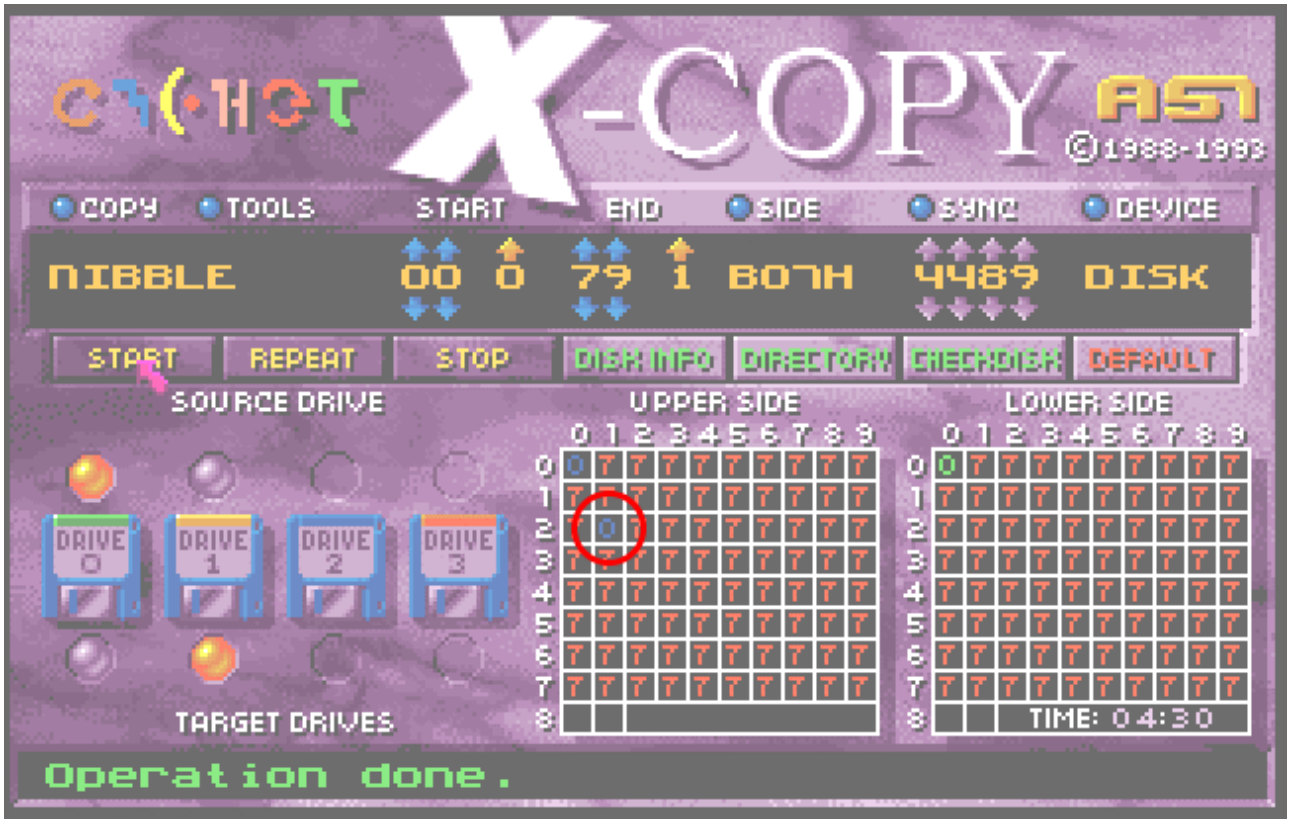
Et on continue l'exécution du programme, taper donc : `x`

Ainsi après avoir lu quelques pistes le jeu se bloque au cylindre **21** et part en boucle. *(À moins d'avoir une configuration mémoire qui ne permettra pas de charger autant de donnée, dans ce cas, un message d'erreur apparaitra et il faudra adapter le code pour riper moins de données et joindre les fichiers binaires plus tard.)*

(on peut aussi compter les accès disque à l'oreille sur un vrai amiga 😉.
On en compte 20, et comme il '*essaie*' de lire la 21, il bloque dessus.



On n'arrive donc pas à lire pour l'instant, plus loin que la piste 20 donc.
Ce qui concorde assez avec le résultat visuel de la copy via X-Copy d'ailleurs et l'hypothèse que l'on avait faite plus haut concernant ce sujet.



Pas besoin de s'inquiéter pour ça, c'est surement la piste des *highscores* qui doit être dans un autre format, c'est assez courant.
Entrer dans l'AR à nouveau pour sauver les pistes **1-20** sur notre disquette de sauvegarde.

Due au fait que le registre d'adresse *A3* est utilisé comme pointeur mémoire, on n'aura pas besoin de chercher la fin des données chargées à la main parce que *A3* pointe directement dessus. :-)
Comme on utilise une extension mémoire de 2MB en CHIP, l'adresse de destination sera *$8000 'Voir organisation mémoire plus haut.'*

Mais en fait, il y a encore plus facile pour savoir où ont été copié ces données, il suffit de re-regarder dans le *filetable* 😉
Taper :

```
r
m 7FCBA+60
```

```
r
D0=00000025 14151114 00082AA5 3E3DDAD7  00000000 00000000 55555555 0000FFF1
A0=0007FE20 00060006 0007EC98 000C1000  0007D314 00DFF000 00000676 000004BA
PC = 0007F64A USP = 00000568 SR = 0000 T=0 S=0 I=000 X=0 N=0 Z=0 V=0 C=0

m 7FCBA+60
:07FD1A 01 01 00 01 00 04 30 00 00 0F 00 00 00 08 00 00 ......0.........
```

On voie clairement dans **A3** que les données ont une longueur de **$0C100** et qu'ils ont été relogé à l'adresse mémoire **$080000**
D'ailleurs ont peu le vérifier facilement, on va juste regarder un peu avant cette adresse pour être sûr de son début.
Taper :

```
m 80000-20
```

```
m 80000-20
:07FFE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
:07FFF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
:080000 54 50 57 4D 00 00 AE 42 00 60 00 00 16 60 00 01 TPWM...B.`...`..
:080010 72 0A 60 00 02 6E 00 08 5A 00 0C 50 40 00 00 01 r.`..n..Z..P@...
:080020 4B FA FF FA 2A 88 00 2A 48 1B 40 08 07 51 ED 20 K...*..*H.@..Q.
:080030 08 05 00 04 06 20 4D D1 FC 00 00 00 F0 E6 2B 48 ..... M.ü.....+H
:080040 08 00 00 4D F9 00 DF F0 00 46 FC 00 27 00 3D 7C ...M..ß..Fü.'.=|
:080050 7F FF 00 9A 80 02 06 96 41 FA 06 90 21 C8 02 00 ▒.......A...!...
:080060 6C 41 FA 07 22 00 08 68 80 00 08 AA 2D 48 00 80 lA.."..h....-H..
:080070 61 00 02 07 68 3D 7C C0 18 01 28 86 00 70 00 96 a...h=|...(..p..
```

Voilà, c'est l'limpide, les données ont bien été relogé en **$80000**

/!\ Retirer la disquette Original de Turrican 3 et **insérer** la disquette de backup
Et sauvons le tout, **taper :**

```
sm piste1-20,80000 c1000
```

```
sm piste1-20,80000 C1000
Disk ok
```

On va faire la même chose pour les pistes allant de **21-79**.
Replacer le disque original **et Réinitialiser** votre machine.
Entrer dans l'AR, et on recommence nos anciennes manipulations.
`bs 7f020` puis `G 7E800`

Le *breakpoint* est atteint très rapidement et on entre automatiquement dans l'AR

On modifie encore notre filetable en **$7FCBA+60** mais cette fois, **non pas pour la piste suivante** mais pour la **face suivante**.
Peut-être qu'il n'utilise qu'une face pour les *highscores*, c'est ce qui semble être le cas au vue de l'écran de copie de Xcopy.

On change les paramètres pour charger **$F0000** octets du cylindre $15 (!21), piste 1
(rappelez-vous la face 0 de la piste $15(!21) est la piste des Highscores) on va donc essayer la face 1.
Taper : `M 7FCBA+60`
Et modifier comme indiqué dans l'image ci-dessous.

```
m 7fcba+60
:07FD1A 01 01 00 01 00 04 30 00 00 00 34 00 00 00 00 00 ......0...4.....

m 7fcba+60
:07FD1A 01 15 01 01 00 04 30 00 00 0F 00 00 00 00 00 00 ......0...4.....
:07FD2A 01 02 00 00 00 01 65 00 00 02 22 00 00 00 00 00 ......e..."......
```

On retourne à l'exécution de notre programme qui va donc charger le reste des données.
Taper : `X`

Le jeu continue son chargement, on avait raison ☺
Le chargement des données se termine en piste 80 avec un message d'erreur à l'écran :

```
SERIOUS DISKERROR.CLICK MOUSE TO RETRY.
```

Entrer à nouveau dans l'AR et regardons tout ça.

Comme vous le savez, A3 pointe vers la fin des données, dans ce cas-ci **$13E200**.
On peut vérifier ça avec la même manipulation que tout à l'heure, taper : `R`

```
r
D0=00002900 11401105 00082AA5 BB4AB9A5  00000000 00000000 55555555 0000FF74
A0=0007EE32 0007FE70 0007EEC4 0013E200  0007D314 00DFF000 00000676 000004BA
PC = 0007FB58 USP = 00000568 SR = 0008 T=0 S=0 I=000 X=0 N=1 Z=0 V=0 C=0
```

Bon, aucune chance de faire rentrer ça dans notre disquette de backup déjà bien remplie.
Insérer donc une nouvelle disquette vierge, **taper :**
`format Turrican3_2`
Ready to format disk in drive DF0: (y/n)? y
Disk ok

On sauve le tout, **taper** : `sm piste21-79, 80000 13E200`
On sauve exactement $BE200 = $13E200-$80000 = !778752 octets !

```
format Turrica3_2
Ready to format disk in drive DF0: (y/n)?
y
Disk ok

sm piste21-79,$80000 $13E200
Disk ok

dir
Directory of (Turrica3_2)
      778752  piste21-79
0137 blocks free, 92.2 % of disk used
Disk ok
```

Part 5 Taille d'un cylindre custom

Il serait intéressant de connaitre la taille custom que font ses pistes.
Noter que cette section n'est pas obligatoire, vous pouvez la passer et aller directement à la prochaine étape.

Replacer le disque original **et Réinitialiser** votre machine.
Entrer dans l'AR, et on recommence nos anciennes manipulations.
`bs 7f020` puis `G 7E800`

Le *breakpoint* est atteint très rapidement et on entre automatiquement dans l'AR

On modifie encore notre filetable en **$7FCBA+60** mais cette fois on va charger des données à partir de la piste 0 face 1 (et non face0)

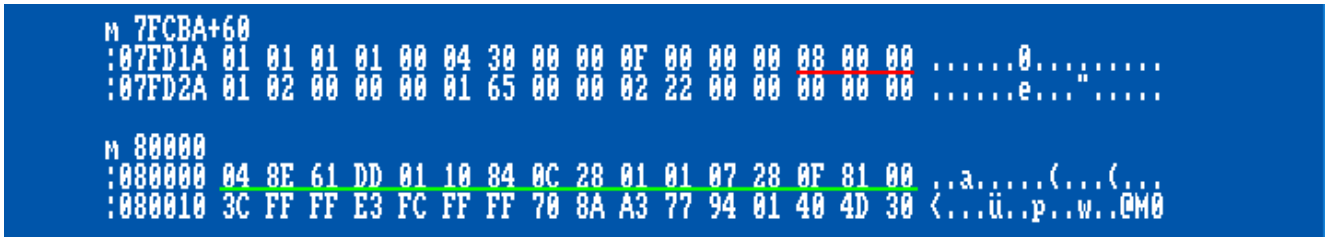
Taper : `M 7FCBA+60`
Et modifier comme indiqué dans l'image ci-dessous.

On retourne à l'exécution de notre programme qui va donc charger le reste des données.
Taper : `X`

Le jeu continue son chargement, on avait raison ☺
Le chargement des données se termine en piste 80 avec un message d'erreur à l'écran :

Entrer à nouveau dans l'AR et regardons tout ça.

On va vérifier que les données sont bien chargées en *$80000* comme précédemment.
Taper : `M 7FCBA+60`

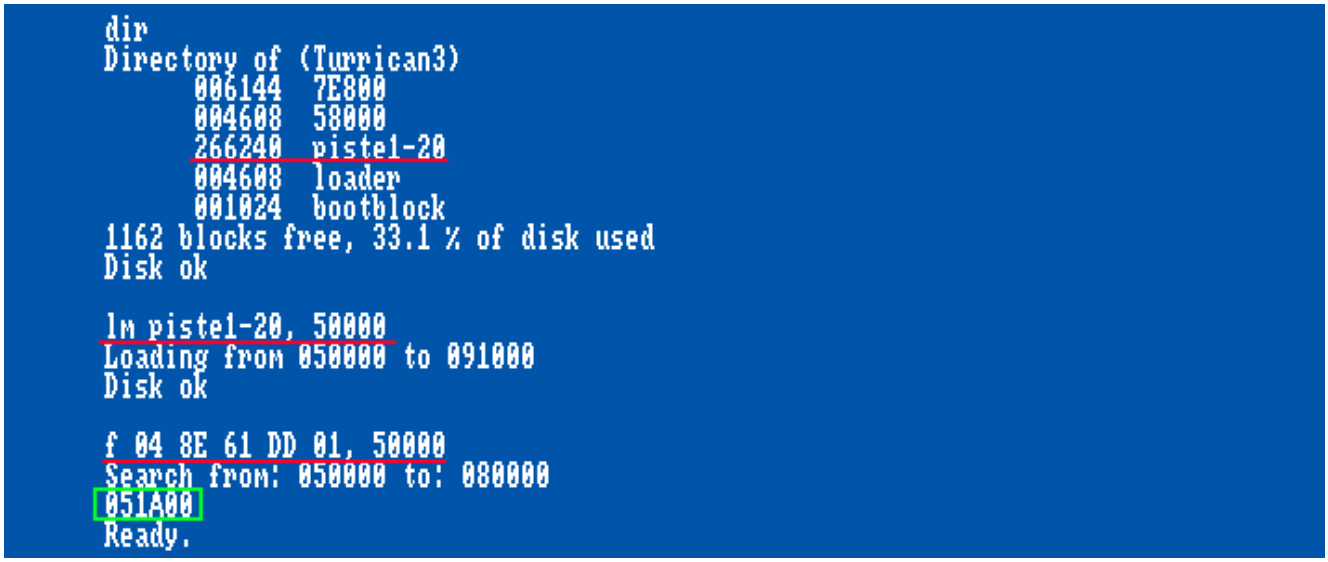


Bon, rien de neuf au soleil, tout va bien, les données ont bien été relogé en $80000
Allons voir tout ça et noter ce début de piste, **TAPER :** `M 80000` comme indiqué dans l'image ci-dessus.
On note sur un coin de feuille les débuts des datas en 80000 qui correspondent au début des datas de la piste 1 face 1
04 8E 61 DD 01 10 84 0C 28 01 01 07 28 0F 81 00

Replacer la 1ère disquette de backup dans votre machine, celle contenant le fichier préalablement sauvé : **piste1-20**
On va charger ce fichier et chercher dedans cette chaine hexa, **taper :**

```
lm piste1-20, 50000
```

```
f 04 8E 61 DD 01, 50000
```



Bingo, trouvé en *$51A00*
Ce qui nous donnes $51A00 - $50000 = ***$1A00***
On a donc ici une taille de *cylindre* custom de ***$1A00*** au lieu de $1600, une belle performance 😊

Part 6 On range Tout

Bien félicitations, on va pouvoir ranger notre disquette originale mais avant cela, on va remettre le *bootblock* d'origine.
Replacer le disque original **et Réinitialiser** votre machine, **entrer** directement dans l'AR.
Taper :

```
rt 0 1 50000

a 500be
^0500BE JMP 7E800

bootchk 50000
wt 0 1 50000
```

```
rt 0 1 50000
Disk ok

a 500BE
^0500BE JMP 7E800
^0500C4

bootchk 50000
Old checksum was E6ECB3C6, now is set to 35562C3D

wt 0 1 50000
Disk ok
```

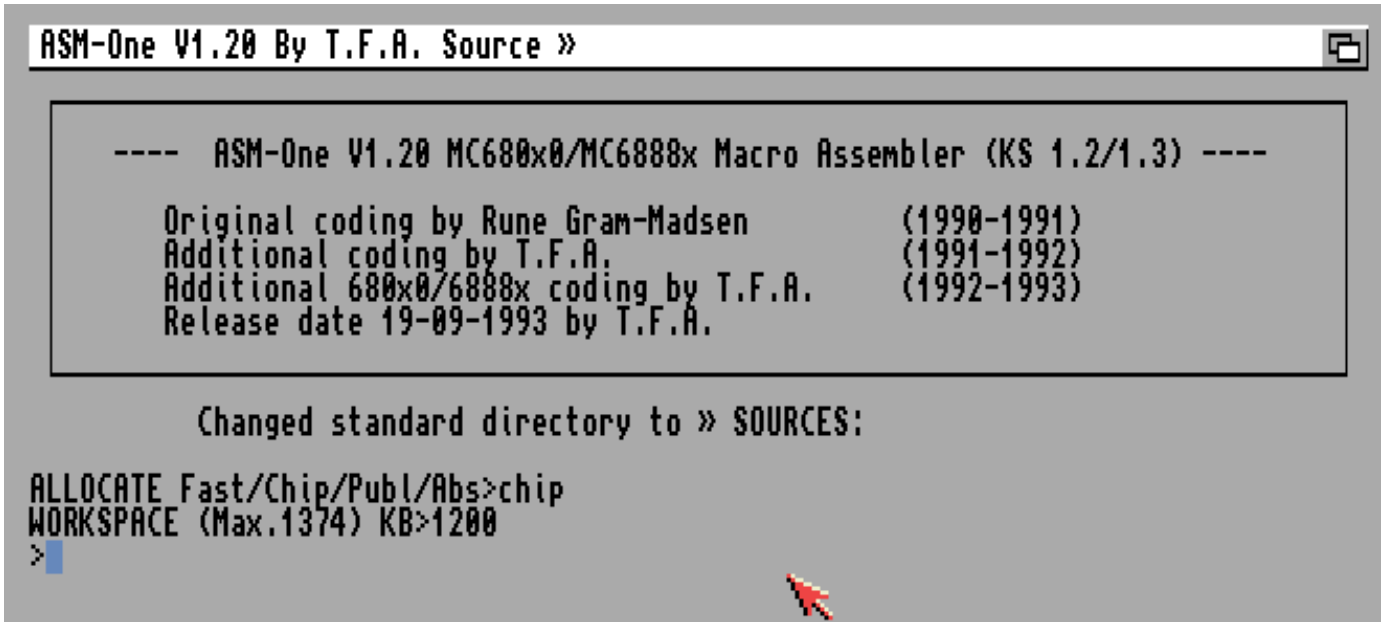
Vous pouvez si vous le souhaiter rebooter votre amiga et vérifier que la disquette originale fonctionne maintenant comme au début.
C'est-à-dire, elle lance le chargement du jeu et ne rentre plus dans notre ancienne *loop*

Voilà, c'est officiel, **vous pouvez ranger** votre disquette originale dans votre **tiroir** 😉

Part 7 Code

Maintenant avec ces informations ça devrait être facile de patcher le chargeur pour gérer nos pistes, hé !
Bootez sur votre Assembleur préféré, ASM-One dans ce cas précis, et commençons par écrire nos nouvelles images disques.

On mettra toutes les données sur 2 disquettes puisqu'elles ne tiendront pas sur une disquette 'standard' Amiga-Dos formatée.
Réserver 1200 ko de mémoire chip



Appuyer sur la touche **ESC** pour passer en mode édition et **taper** le code suivant :
Adapter si nécessaire les chemins dans les includes.

Crackdisk.s ➔

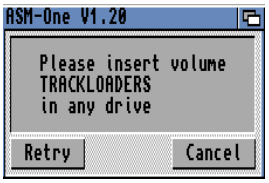
```
                MOVE.W #$4E75,LOADER+$806           ; mets un -rts- sur le Branchement "Load Highscore".
                MOVE.L #$4E714E71,LOADER+$1132      ; 'Nope' une routine qui demande la disquette originale.
                MOVE.W #$4E71,LOADER+$1136
                MOVE.L #$11805000,LOADER+$D4C       ; Désactive le diskstepper de Turrican 3
                MOVE.W #$600E,LOADER+$D50

                LEA NEWLOADER(PC),A0                ; écrase simplement une partie du loader original avec le nôtre.
                LEA LOADER+$AAC(PC),A1              ; chargeur+$AAC est $7F2AC plus tard en mémoire.
                MOVE.L #(NEWLOADERENDE-NEWLOADER)-1,D0

REPLACELOADER:
                MOVE.B (A0)+,(A1)+
                DBF D0,REPLACELOADER
                RTS

NEWLOADER:
                MOVE.L $7FDFE,A0                    ; Lire l'adresse
                MOVE.L $7FE02,D0                    ; Octets à lire
                MOVEQ #0,D1                         ; Vider les registres désirés.
                MOVEQ #0,D2
                MOVE.B $7FE20,D1                    ; Obtenir le numéro de Cylindre.
                LEA DISKSTATUS(PC),A2               ; Adr. du word de statuts "quel disk est inséré?"
                                                    ; 0=Disk1, 1=Disk2.

                CMP.B #21,D1                        ; Doit être lu au-delà du cylindre 21
                BGE.B SECONDDISK                    ; Oui... c'est sur la disquette 2.
                TST.W (A2)                          ; Première disquette dans le lecteur ?
                BEQ.B CORRECTDISK                   ; Où ? tout est ok alors.
                BSR.W FLASHCOLORS                   ; Autrement flash de couleurs !!!
                CLR.W (A2)                          ; Changer le statut de la disquette en "Disque 1 insérée".
                BRA.B CORRECTDISK

SECONDDISK:
                SUB.B #21,D1                        ; Tout ce qui commence depuis le cylindre 21
                                                    ; sur la disquette originale
                                                    ; Commence maintenant de zéro à nouveau sur la disquette 2.
                TST.W (A2)                          ; Seconde disquette dans le lecteur.
                BNE.B CORRECTDISK                   ; Où ? continuer le chargement de pistes.
                BSR.W FLASHCOLORS                   ; Autrement couleurs qui flashent...
                MOVE.W #1,(A2)                      ; Et changez le statut de la disquette en "Disque 2 insérée".
```

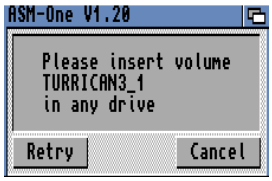
```
CORRECTDISK:
                LSL.L #1,D1                             ; Numéro de cylindre * 2 = numéro de piste ...
                ADD.B $7FE1C,D1                         ; ... + numéro de piste sur le cylindre=piste correcte!
                MULS #$1A00,D1                          ; Multiplier avec la taille de piste du jeu=position d'octet
                                                        ; correct sur la disquette.

                DIVS.W #$1600,D1                        ; Diviser par une taille de piste dos normale
                SWAP D1                                 ; d1 = numéro de piste correcte sur le disque déplombé.
                MOVE.W D1,D2                            ; Reste de la division = position d'octet sur la piste.
                CLR.W D1                                ; Déplace la position de l'octet en d2.
                SWAP D1                                 ; Vide la position de l'octet en d1 et ...
                MOVE.L $7FE0A,A2                        ; Le swap de nouveau, donc d1 = numéro de piste.l de nouveau.
                LEA $DFF000,A6                          ; Adresse de chargement MFM.
                BSR.B TRACKLOADER                       ; Adresse de base des puces Custom en a6
                                                        ; nécessaire pour le chargeur de piste.
                LEA $DFF000,A5

                RTS

FLASHCOLORS:
                MOVE.W D7,$DFF180

                SUBQ.W #1,D7
                BTST #6,$BFE001
                BNE.B FLASHCOLORS
                MOVE.W #$0,$DFF180
                RTS

DISKSTATUS:
                DC.W 0

TRACKLOADER:
                INCBIN "TRACKLOADERS:TRACKLOADER_AlphaOne[2004].bin" ; On charge le Trackloader d'AlphaOne

NEWLOADERENDE:                                          ; Image disque.

DISK:                                                   ; Disque 1 = !50 pistes!
                DC.B "DOS",0
                DC.L 0
                DC.L 0
                MOVE.W #$02,$1C(A1)             ; Utilise le trackdisk-device pour charger les données en $7E800
                MOVE.L #$7E800,$28(A1)
                MOVE.L #$400,$2C(A1)
                MOVE.L #$1800,$24(A1)
                JSR -$1C8(A6)
                MOVE.W #$02,$1C(A1)
                MOVE.L #$58000,$28(A1)
                MOVE.L #$1C00,$2C(A1)
                MOVE.L #$1200,$24(A1)
                JSR -$1C8(A6)
                MOVE.W #$7FFF,$DFF09A           ; Flingue les Interruptions & les DMA et saute dans le chargeur du jeu.
                MOVE.W #$7FFF,$DFF096
                MOVE.B #$1,$7FE14
                JMP $7E800

BOOTENDE:                                       ; Remplit le reste du code de boot avec des zéros, comme ça
                                                ; le chargeur du jeu commence en position impair en !1024 ($400).
                BLK.B $400-(BOOTENDE-DISK),0

LOADER:
                INCBIN "Turrican3_1:7E800"      ; Position sur disque = $400.
                INCBIN "Turrican3_1:58000"      ; Position sur disque = $1C00.
                BLK.B $600,0

GAMEDATA:
                INCBIN "Turrican3_1:piste1-20" ; Position sur disque = $3400,
                                                ; (Cylindre 1, piste 0 sur la disquette de jeu original).
                BLK.B $A00,0                    ; Juste pour faire les !50 pistes entières. :-)

DISK2:                                          ; Disque 2 = !140 pistes!
                BLK.B $1A00,0                   ; Données de jeu sur le disque 2 commence sur le Cylindre 0, piste 1
                INCBIN "Turrican3_2:piste21-79"; (calculé avec une taille de piste de $1A00 bien sûr).
                BLK.B $C00,0                    ; Juste pour faire les !140 pistes entières.
```

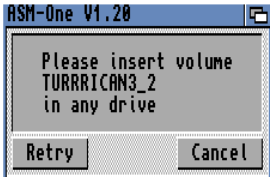
Part 8 Assemblage

Il est maintenant temps d'assembler notre code fraichement taper.
Appuyer sur la touche **ESC** pour passer en mode command line et **taper** A
Insérer comme demander à tour de rôle les disquettes préalablement crée :

- **TRACKLOADERS** #disquette contenant le fichier binaire du TrackLoader d'AlphaOne v2004 (404 octets)
- **Turrican3_1** #disquette fraichement crée dans ce tuto.
- **Turrican3_2** #disquette fraichement crée dans ce tuto.
-







Cela prend un certain temps puis, normalement se termine avec un message '**No errors'**.
Toutes les données ont été chargé.

```
ASM-One V1.20 By T.F.A. Source » crackdisk.s

   ----  ASM-One V1.20 MC680x0/MC6888x Macro Assembler (KS 1.2/1.3) ----

      Original coding by Rune Gram-Madsen          (1990-1991)
      Additional coding by T.F.A.                  (1991-1992)
      Additional 680x0/6888x coding by T.F.A.      (1992-1993)
      Release date 19-09-1993 by T.F.A.

         Changed standard directory to » SOURCES:

ALLOCATE Fast/Chip/Publ/Abs>chip
WORKSPACE (Max.1397) KB>1200
>R
File length =       4019 (=$00000FB3 )
>a
Pass 1..
Pass 2..
Incbin      : "TRACKLOADERS:TRACKLOADER_ALPHAONE[20 =       404 (=$00000194 )
Incbin      : "TURRICAN3_1:7E800"                   =      6144 (=$00001800 )
Incbin      : "TURRICAN3_1:58000"                   =      4608 (=$00001200 )
Incbin      : "TURRICAN3_1:PISTE1-20"               =    266240 (=$00041000 )
Incbin      : "TURRICAN3_2:PISTE21-79"              =    778752 (=$000BE200 )
No Errors
>
```

Exécutez le code avec la ligne de commande `J`

```
ALLOCATE Fast/Chip/Publ/Abs>chip
WORKSPACE (Max.1397) KB>1200
>R
File length =       4019 (=$00000FB3 )
>a
Pass 1..
Pass 2..
Incbin      : "TRACKLOADERS:TRACKLOADER_ALPHAONE[20 =       404 (=$00000194 )
Incbin      : "TURRICAN3_1:7E800"                   =      6144 (=$00001800 )
Incbin      : "TURRICAN3_1:58000"                   =      4608 (=$00001200 )
Incbin      : "TURRICAN3_1:PISTE1-20"               =    266240 (=$00041000 )
Incbin      : "TURRICAN3_2:PISTE21-79"              =    778752 (=$000BE200 )
No Errors
>j

D0: 00001FFF 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0: 000A7822 000A88E4 00000000 00000000 00000000 00000000 00000000 000853C4
SSP=000863C0 USP=000853C4 SR=0000 -- -- PL=0 ----- PC=EOP      VBR=00000000
>
```

Maintenant que le chargeur est patché en mémoire il est temps de sauver les pistes sur de nouvelles disquettes.
Enlever toutes les disquettes du/des lecteurs et Insérer une nouvelle disquette vierge dans le lecteur
Ca sera notre disque 1 de Turrican3 Cracké
Taper : `WT pour écrire les pistes.`
Indiquer :
DISK pour **RAM_PTR**
0 pour **DISK_PTR**
50 pour **LENGTH**

Les données sont enregistré sur cette nouvelle disquette
Taper : `CC pour calculer et enregistrer le checksum de boot.`

```
>WT
RAM_PTR>DISK
DISK_PTR>0
LENGTH>50
>cc
```

Enlever la disquette du lecteur, **étiqueter la : DISK1 TURRICAN3**

Insérer une nouvelle disquette vierge dans le lecteur, c'est au tour de la disquette 2

Taper : `WT pour écrire les pistes.`
Indiquer :
DISK2 pour **RAM_PTR**
0 pour **DISK_PTR**
150 pour **LENGTH**

```
>wt
RAM_PTR>DISK2
DISK_PTR>0
LENGTH>150
>
```

Enlever la disquette du lecteur, **étiqueter la : DISK2 TURRICAN3**

Rebooter votre Amiga, **insérer** la disquette 1 et profitez de ce super jeu.

Part 9 Binaire du HARDWARE-DISKLOADER (c) ALPHA ONE 2004

Pour ceux qui n'ont pas réussi à trouver le binaire ou les sources en question, voilà la version hexa.

```
00000000  49 F9 00 BF D1 00 4B F9 00 BF E0 01 7E 00 D0 82  Iù.¿Ñ.Kù.¿à.~.Ð,
00000010  18 BC 00 7D 4E 71 4E 71 18 BC 00 75 61 00 01 6E  .¼.}NqNq.¼.ua..n
00000020  08 15 00 04 67 22 08 D4 00 01 08 D4 00 00 4E 71  ....g".Ô...Ô..Nq
00000030  4E 71 08 94 00 00 4E 71 4E 71 08 D4 00 00 61 00  Nq.”..NqNq.Ô..a.
00000040  01 40 61 00 01 48 60 D8 83 FC 00 02 48 41 4A 41  .@a..H`Øƒü..HAJA
00000050  67 06 08 94 00 02 60 04 08 D4 00 02 48 41 4A 01  g..”..`..Ô..HAJ.
00000060  67 24 08 94 00 01 08 D4 00 00 4E 71 4E 71 08 94  g$.”...Ô..NqNq.”
00000070  00 00 4E 71 4E 71 08 D4 00 00 61 00 01 04 61 00  ..NqNq.Ô..a...a.
00000080  01 0C 51 C9 FF DA 61 00 01 04 3D 7C 82 10 00 96  ..QÉÿÚa...=|,..-
00000090  3D 7C 7F 00 00 9E 3D 7C 85 00 00 9E 3D 7C 44 89  =|...ž=|….ž=|D‰
000000A0  00 7E 3D 7C 40 00 00 24 2D 4A 00 20 3D 7C 99 00  .~=|@..$-J. =|™.
000000B0  00 24 3D 7C 99 00 00 24 3D 7C 00 02 00 9C 08 39  .$=|™..$=|...œ.9
000000C0  00 01 00 DF F0 1F 67 F6 3D 7C 40 00 00 24 7A 00  ...ßð.gö=|@..$z.
000000D0  22 4A 28 3C 55 55 55 55 0C 59 44 89 66 FA 0C 51  "J(<UUUU.YD‰fú.Q
000000E0  44 89 67 F4 26 11 22 29 00 04 C6 84 C2 84 E3 83  D‰gô&.")..Æ„Â„ãƒ
000000F0  86 81 E0 9B B6 05 67 08 D3 FC 00 00 04 3E 60 D8  †.à›¶.g.Óü...>`Ø
00000100  D3 FC 00 00 00 38 2C 3C 00 00 00 7F 22 29 02 00  Óü...8,<....")..
00000110  26 19 C6 84 C2 84 E3 83 86 81 B4 87 6E 06 48 43  &.Æ„Â„ãƒ†.´‡n.HC
00000120  30 C3 48 43 54 87 B4 87 6E 02 30 C3 54 87 B0 87  0ÃHCT‡´‡n.0ÃT‡°‡
00000130  6F 00 00 40 51 CE FF D6 52 05 0C 05 00 0B 66 90  o..@QÎÿÖR.....f.
00000140  08 14 00 02 67 08 08 94 00 02 60 00 FF 3A 08 D4  ....g..”..`.ÿ:.Ô
00000150  00 02 08 94 00 01 08 D4 00 00 4E 71 4E 71 08 94  ...”...Ô..NqNq.”
00000160  00 00 4E 71 4E 71 08 D4 00 00 61 00 00 14 60 00  ..NqNq.Ô..a...`.
00000170  FF 16 18 BC 00 FD 4E 71 4E 71 18 BC 00 E7 4E 75  ÿ..¼.ýNqNq.¼.çNu
00000180  28 3C 00 00 25 00 51 CC FF FE 4E 75 08 15 00 05  (<..%.QÌÿþNu....
00000190  66 FA 4E 75                                      fúNu
```

```
49 F9 00 BF D1 00 4B F9 00 BF E0 01 7E 00 D0 82 18 BC 00 7D 4E 71 4E 71 18
BC 00 75 61 00 01 6E 08 15 00 04 67 22 08 D4 00 01 08 D4 00 00 4E 71 4E 71
08 94 00 00 4E 71 4E 71 08 D4 00 00 61 00 01 40 61 00 01 48 60 D8 83 FC 00
02 48 41 4A 41 67 06 08 94 00 02 60 04 08 D4 00 02 48 41 4A 01 67 24 08 94
00 01 08 D4 00 00 4E 71 4E 71 08 94 00 00 4E 71 4E 71 08 D4 00 00 61 00 01
04 61 00 01 0C 51 C9 FF DA 61 00 01 04 3D 7C 82 10 00 96 3D 7C 7F 00 00 9E
3D 7C 85 00 00 9E 3D 7C 44 89 00 7E 3D 7C 40 00 00 24 2D 4A 00 20 3D 7C 99
00 00 24 3D 7C 99 00 00 24 3D 7C 00 02 00 9C 08 39 00 01 00 DF F0 1F 67 F6
3D 7C 40 00 00 24 7A 00 22 4A 28 3C 55 55 55 55 0C 59 44 89 66 FA 0C 51 44
89 67 F4 26 11 22 29 00 04 C6 84 C2 84 E3 83 86 81 E0 9B B6 05 67 08 D3 FC
00 00 04 3E 60 D8 D3 FC 00 00 00 38 2C 3C 00 00 00 7F 22 29 02 00 26 19 C6
84 C2 84 E3 83 86 81 B4 87 6E 06 48 43 30 C3 48 43 54 87 B4 87 6E 02 30 C3
54 87 B0 87 6F 00 00 40 51 CE FF D6 52 05 0C 05 00 0B 66 90 08 14 00 02 67
08 08 94 00 02 60 00 FF 3A 08 D4 00 02 08 94 00 01 08 D4 00 00 4E 71 4E 71
08 94 00 00 4E 71 4E 71 08 D4 00 00 61 00 00 14 60 00 FF 16 18 BC 00 FD 4E
71 4E 71 18 BC 00 E7 4E 75 28 3C 00 00 25 00 51 CC FF FE 4E 75 08 15 00 05
66 FA 4E 75
```