benoetigte programme:

•resedit am besten mit forker erweiterung
•resorcerer
•ein grafikprogramm

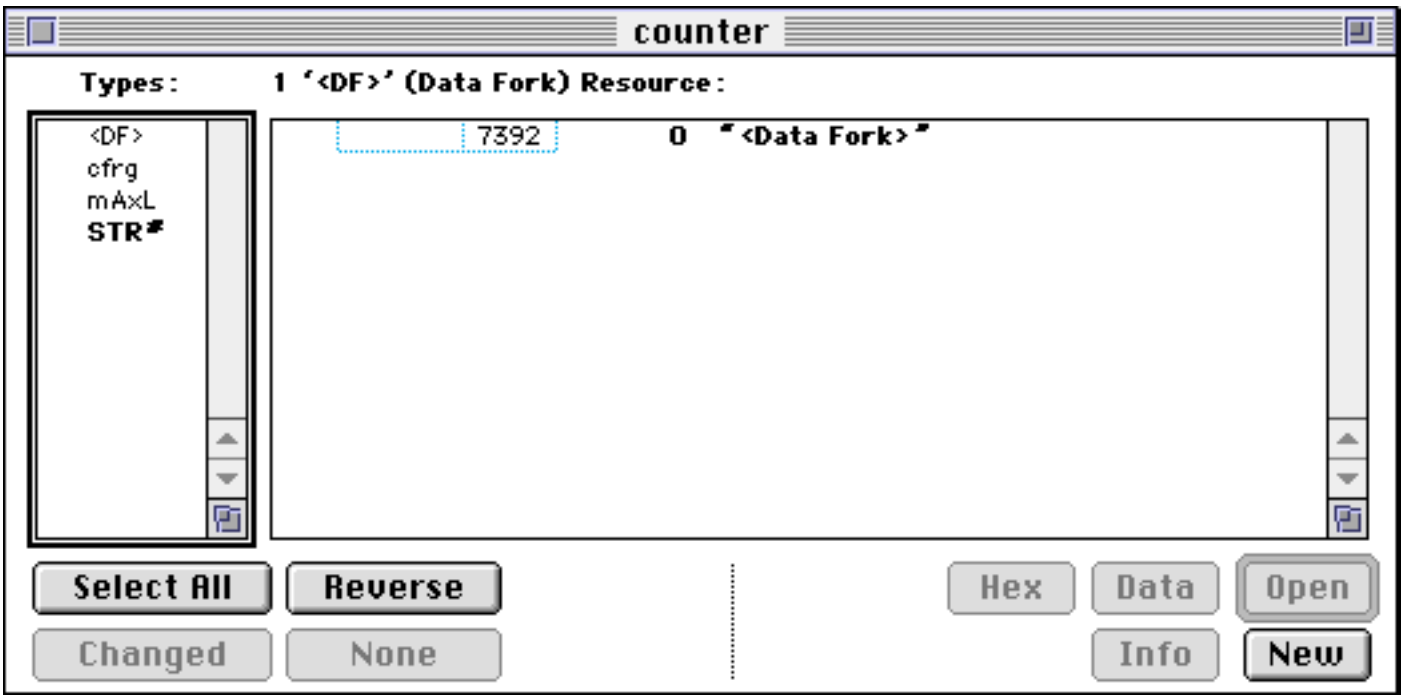
resedit kann nur files bis zu einer bestimmten groesse verarbeiten. ich benutze daher resorcerer.
die programme kann man sich im falle von resedit ‘legal‘ aus dem netz besorgen, resorcerer gibts ueber www.carracho.com oder www.hotlinesw.com. wer das nicht kennt ist selber schuld.

vor dem knacken

man sollte sich irgendwo einen ordner z.b. ‘ƒknackzeugs’ anlegen, in dem die kopien der zu knackenden plugins, die template files und die fertig geknackten plugins abgelegt werden.
es werden folgende template files benoetigt. eins fuer externals, eins fuer patcher, eins fuer patcherastext. dafuer wird von jeder art eine kopie in diesem ordner gelegt. diese template files werden dazu benoetigt, um die entsprechenden daten aus dem vst plugin file, in sie zu kopieren.

wenn du die template files in dem ordner hast und die kopien der pluggo plugins, die du knacken willst, bist du soweit.
wer noch nicht mit resorcerer gearbeitet hat sollte hier weiterlesen. sonst s.4 weiter.
zuerst schlage ich vor, dass du die template files in resorcerer oeffnest. du wirst dann ungefaehr so etwas sehen. ungefaehr, weil die externals mit unterschiedlichen resources bestueckt sein koennen.

abb.1 external



im linken fenster siehst du die verschiedenen resource typen, die das file enthaelt.
im falle dieses externals sind das <DF>, cfrg, maxL, STR#.
•<DF>, steht fuer data fork
•cfrg, steht fuer configuration
•maxL, enthaelt 68k code
STR#, steht fuer strings

wenn man links auf einen resource typ klickt, wird im rechten fenster angezeigt, was alles in dieser resource enthalten ist.

abb.2 patcher

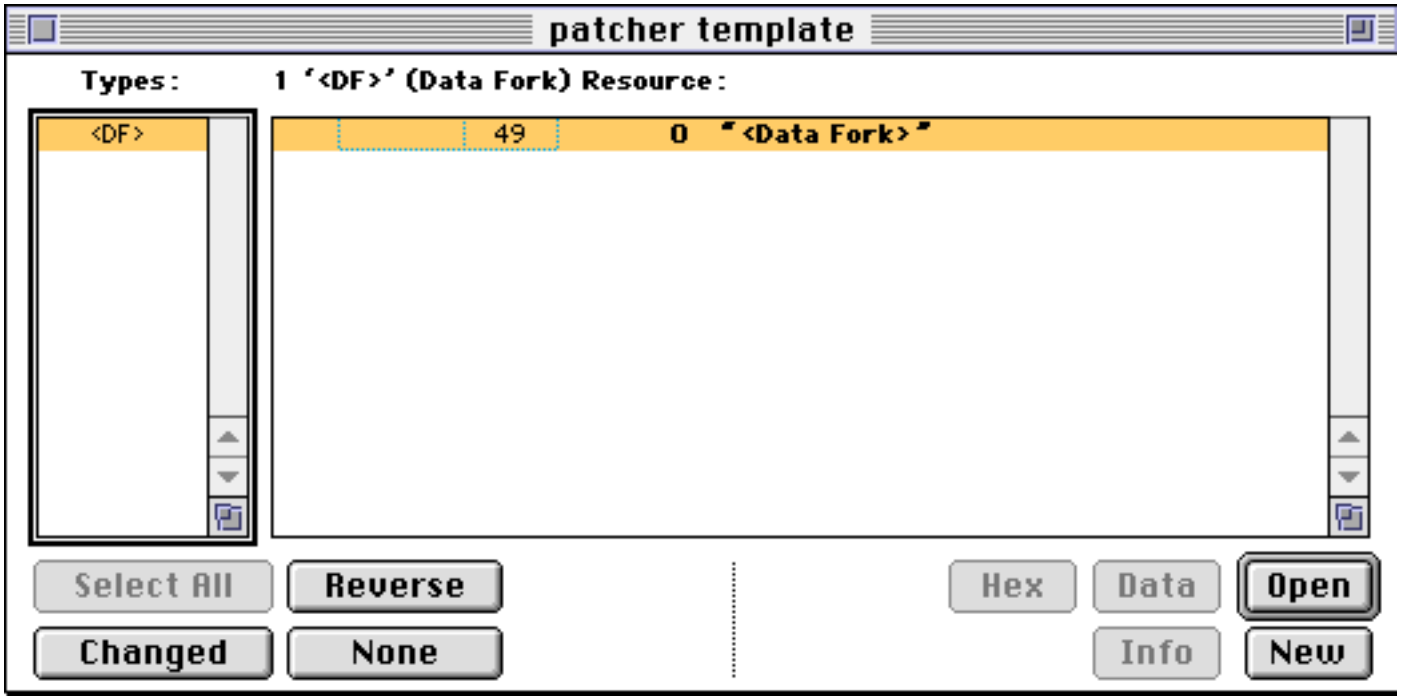
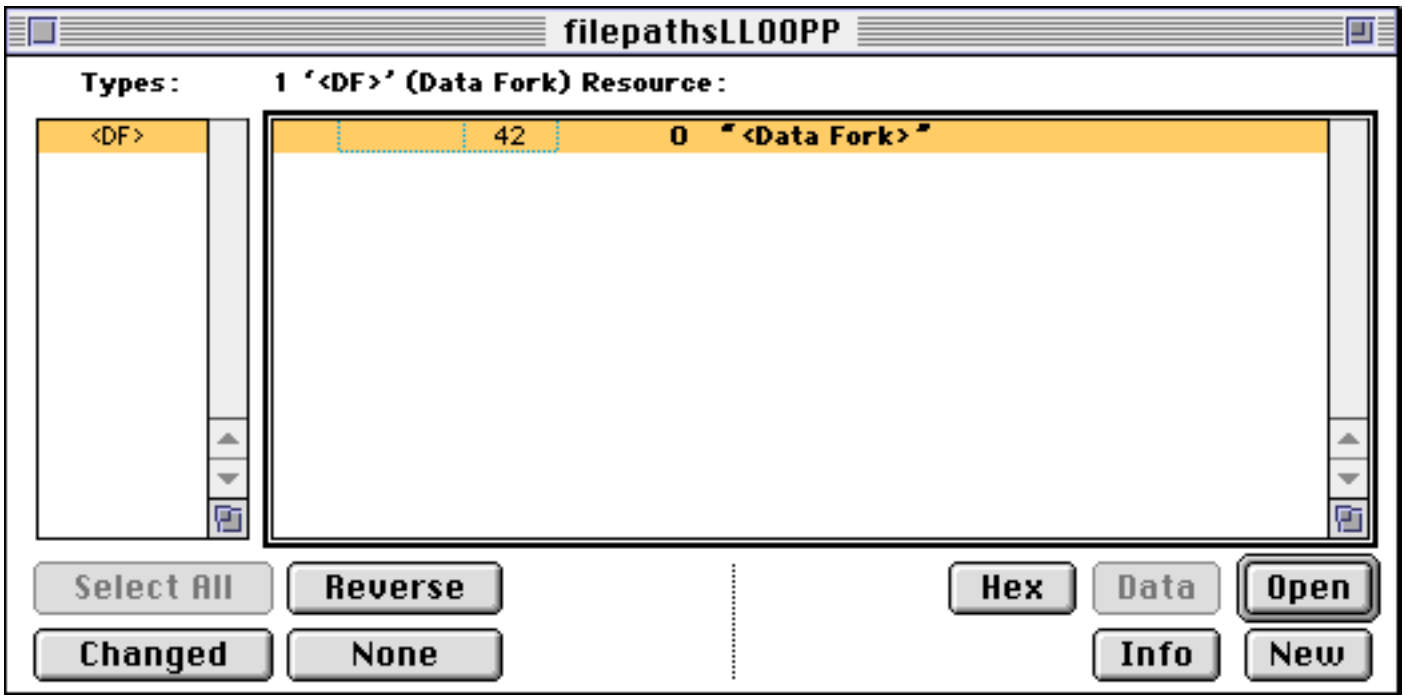
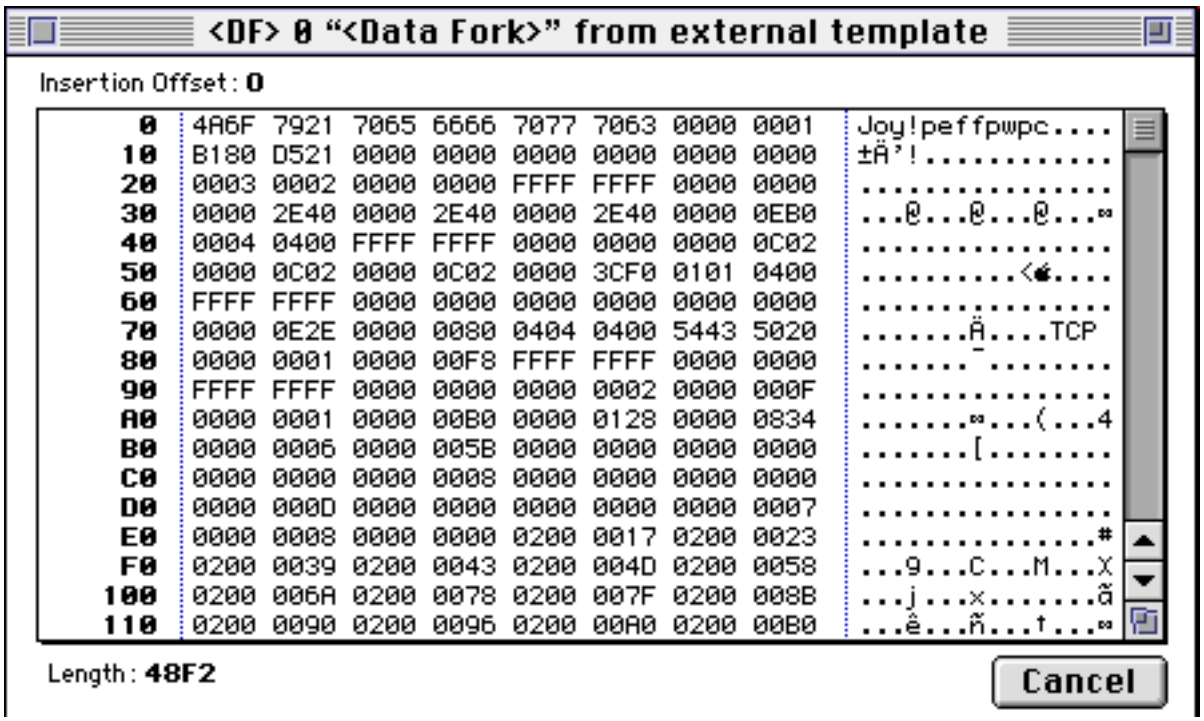


abb.3 patcher as text



wenn man im rechten fenster den inhalt der resource doppelklickt, werden die daten im hex und ascii format angezeigt.

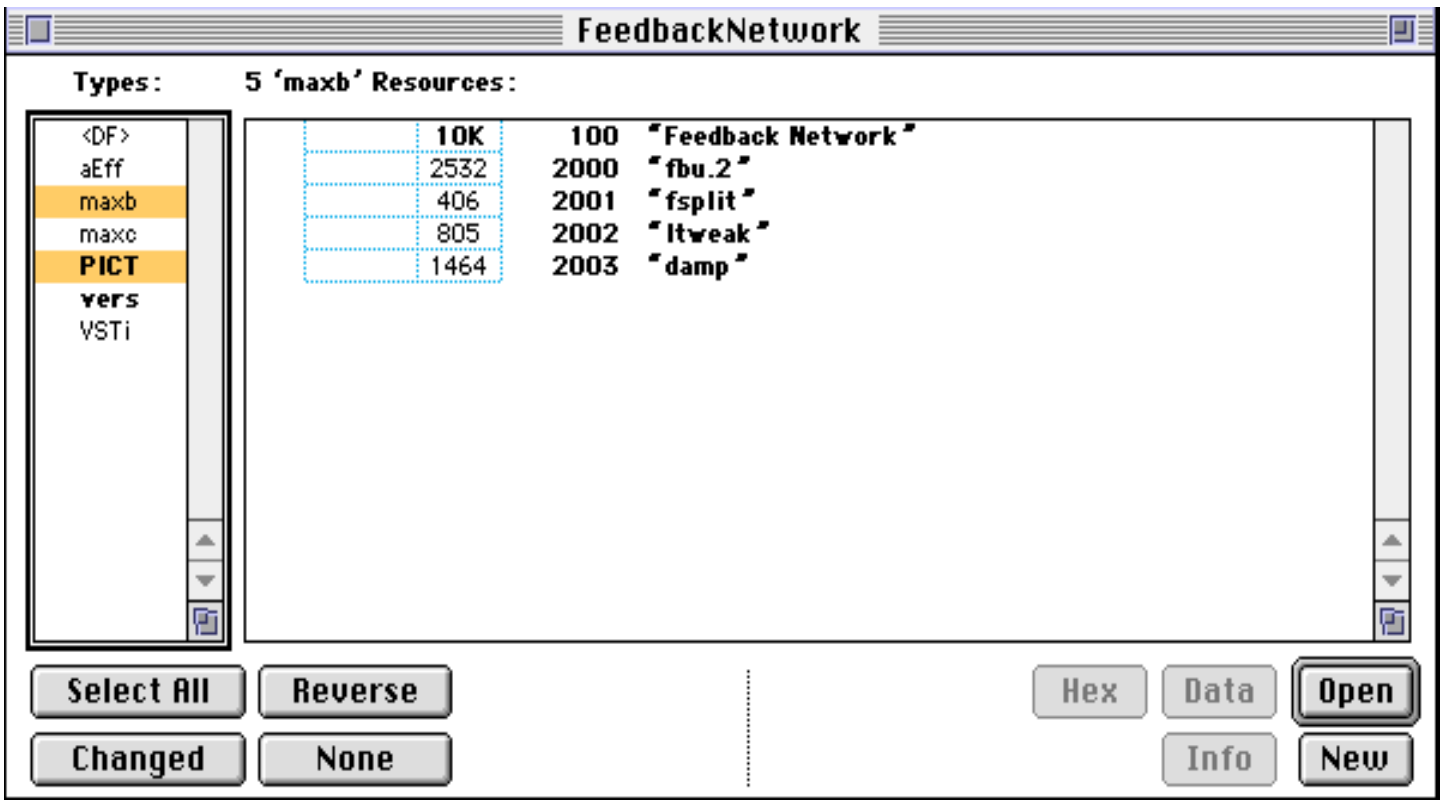
abb.4 teil der daten der <DF> resource von external



das sollte erstmal reichen, um sich in resorcerer zurechzufinden.

jetzt beginnt der eigentliche knack teil.
bitte die kopie des zu knackenden plugins in resorcerer oeffnen. ich habe hier das pluggo plugin feedbacknetwork genommen.

abb.5 maxb und PICT resource markiert.



am besten einen ordner z.b. ƒfertige plugins anlegen, in dem dann wiederum die ordner eines jeden plugins liegen, die alle patches und bilder enthalten, die das plugin benoetigt.

abb.6

bei diesem plugin sind nur die maxb in die PICT resource zu kopieren.
in der maxb resource sind 5 teile enthalten. das sind die patcher aus denen das plugin besteht.
also wird das template patch 5 mal kopiert und genauso benannt wie es in resorcerer zu sehen ist.
dann doppelklicken im rechten fenster auf die erste resource. in diesem fall “Feedback Network”.
ein weiteres fenster oeffnet sich, in dem sind die daten in hex und ascii zu sehen. nun den cursor an den anfang des rechten feld, den ascii teil, setzten und mit apfel-a alles markieren und mit apfel-c die daten
kopieren. nun das entsprechend beschriftete template patch in resorcerer oeffnen. das sieht dann so aus wie abb.2 . im linken fenster die <DF> resource selektieren und im rechten fenster doppleklicken.
den cursor wieder ganz rechts an den anfang setzen und mit apfel-a alles selektieren, loeschen und mit apfel-v die daten einfuegen.
mit apfel-s speichern. das file kann nun mit apfel-w geschlossen werden.
genauso mit den restlichen 4 resourcen im <DF> verfahren.
das noch offene resource fenster, des plugins mit klick auf cancel oder apfel-w schliessen.

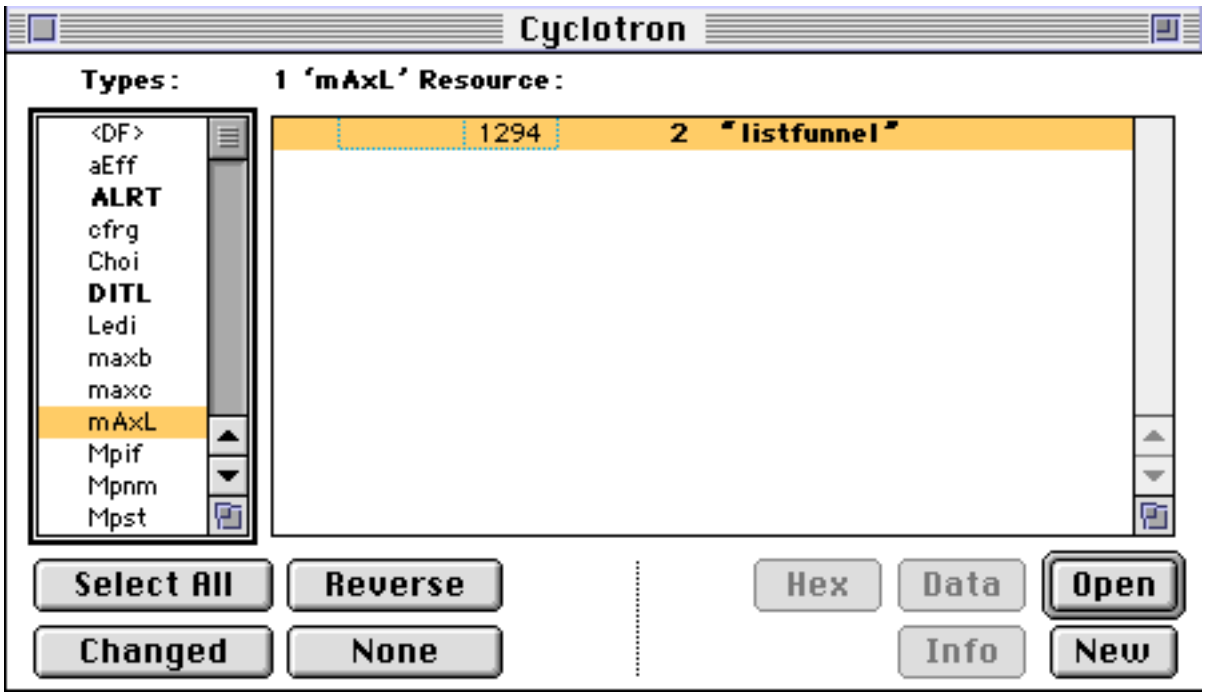
nun im linken fenster des plugin files die PICT resource selektieren und sich das bild heraussuchen, das nach einem userinterface aussieht. oft sind hier auch info bilder enthalten. die braucht man aber nicht fuer das patch.
im rechten fenster das bild doppelklicken, kopieren und im grafikprogramm ein neues bild oeffnen, in das der clipboard inhalt, die PICT resource, mit apfel-v eingefuegt wird. das bild muss dann so wie in resorcerer angegeben gespeichert werden.

wenn jetzt alle files, die das plugin patch braucht in einem ordner liegen, kann man das hauptpatch, in diesem fall [Feedback Network] in max oeffnen. ;)

ein ende.

jetzt erklaere ich noch, was zu tun ist, falls ein plugin eine resource typ maxL hat.

abb.7 maxL enthaelt external “listfunnel“



die maxL resource enthaelt nicht die eigentlichen daten des externals, sondern 68k code, der bewirkt, dass, falls dieses external auf einem 68k computer verwendet wird, eine message erscheint, dass dieses objekt - hier “listfunnel“ - nicht 68k prozessor compatibel ist. an der maxL resource laesst sich aber einfach ueberpruefen, ob das plugin externals enthaelt. man kann dann am besten nach den angegebenen namen bei sich im externals ordner suchen. falls sie gefunden werden, glueck gehabt, weniger arbeit. falls nicht gibts noch ein bisschen billigen copy und paste spass.

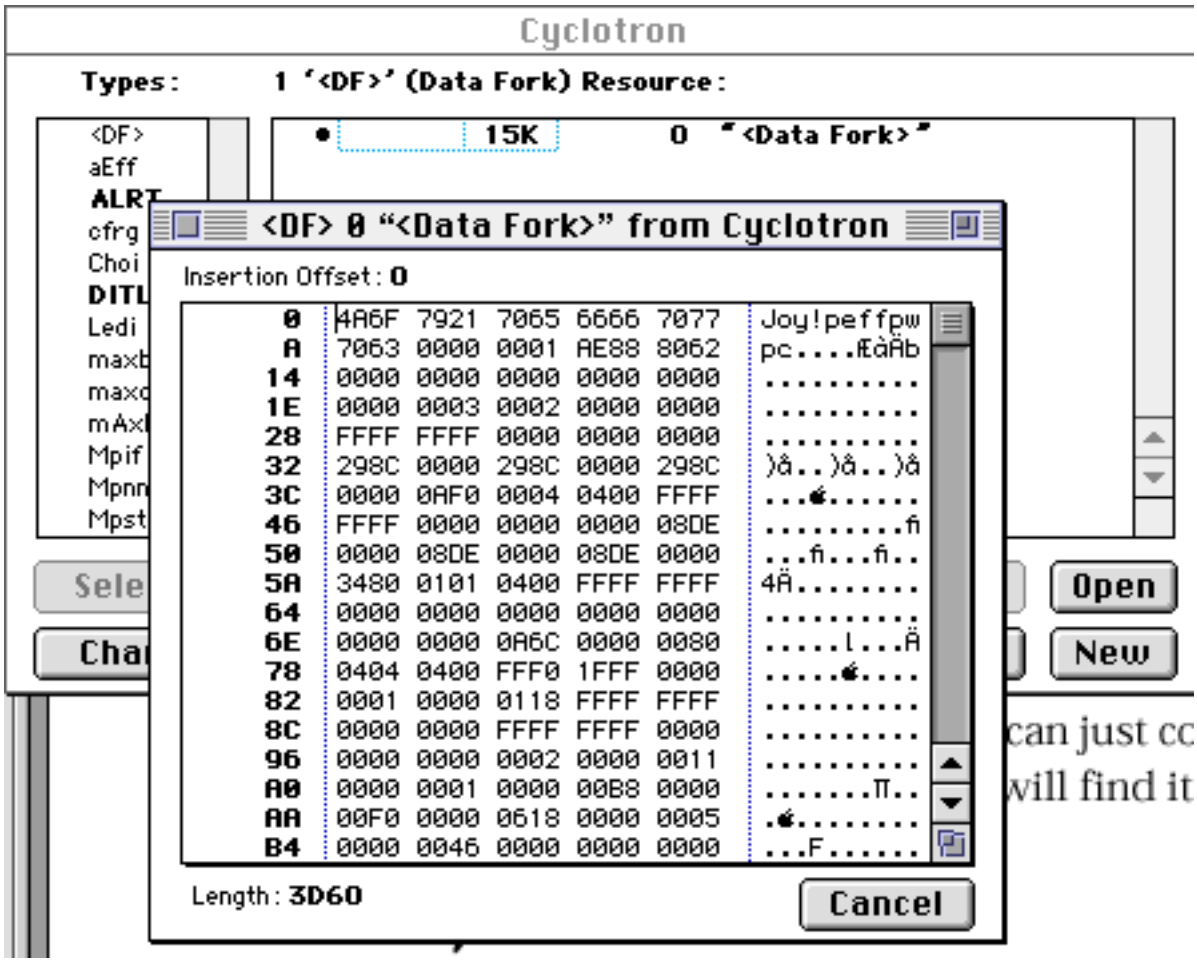
aus folgenden resources muessen alle daten oder teile daraus kopiert werden:
•<DF>
•cfrg
•maxL
•STR#

am besten mit der maxL resource beginnen, die wird naemlich komplett kopiert. d.h. sie kann im linken fenster selektiert und kopiert werden. dann ins linke fenster des geoeffneten external templates klicken und die daten einfuegen.

als naechstes kommen die daten aus dem <DF> dran. hier verhaelt es sich so, dass der datenblock, der relevant ist fuer das external mit “Joy!peffpwpc...“ beginnt. und zwar fuer jedes vorkommende external im plugin ein solcher datenblock. in unserem fall also nur einer, naemlich der fuer “listfunnel”. sollten mehrere externals vorhanden sein, muss man den entsprechenden “Joy!peffpwpc“ datenblock kopieren.

das geht am einfachsten wenn man vorher nochmal in maxL schaut, an welcher stelle welches external liegt. die reihenfolge ist dann im <DF> dieselbe. man kann nun bei geoeffnetem resource fenster mit apfel-g die funktion ‘goto/find/replace‘ nutzen. man kreuzt ascii an und gibt “Joy“ ein. nach dem ersten suchdurchgang wird das erste gefunden und markiert. beim zweiten das zweite usw.
das macht es leichter die korrekten datenbloecke zu kopieren.

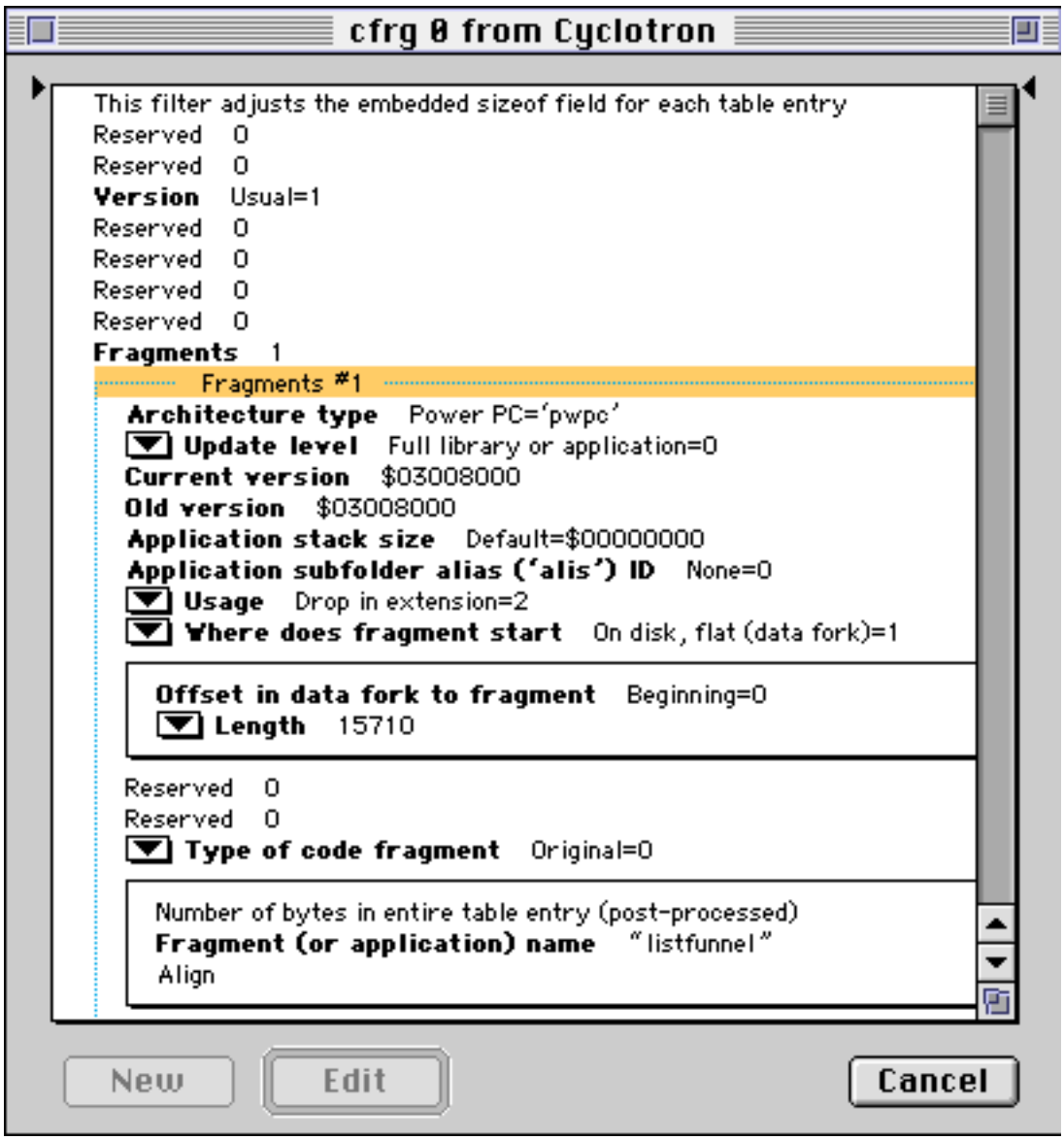
abb.8 geoeffneter <DF>



im fenster des geoeffneten <DF> den cursor im ascii teil an den anfang setzen alles selektieren und kopieren. die daten wie gehabt in den <DF> des external patches einfuegen und speichern.
natuerlich hat man sich vorher wie bei den patchern auch ein file angelegt, dass aus einem kopierten und umbenannten external template file besteht.

weiter mit den relevanten teilen aus der cfrg resource.

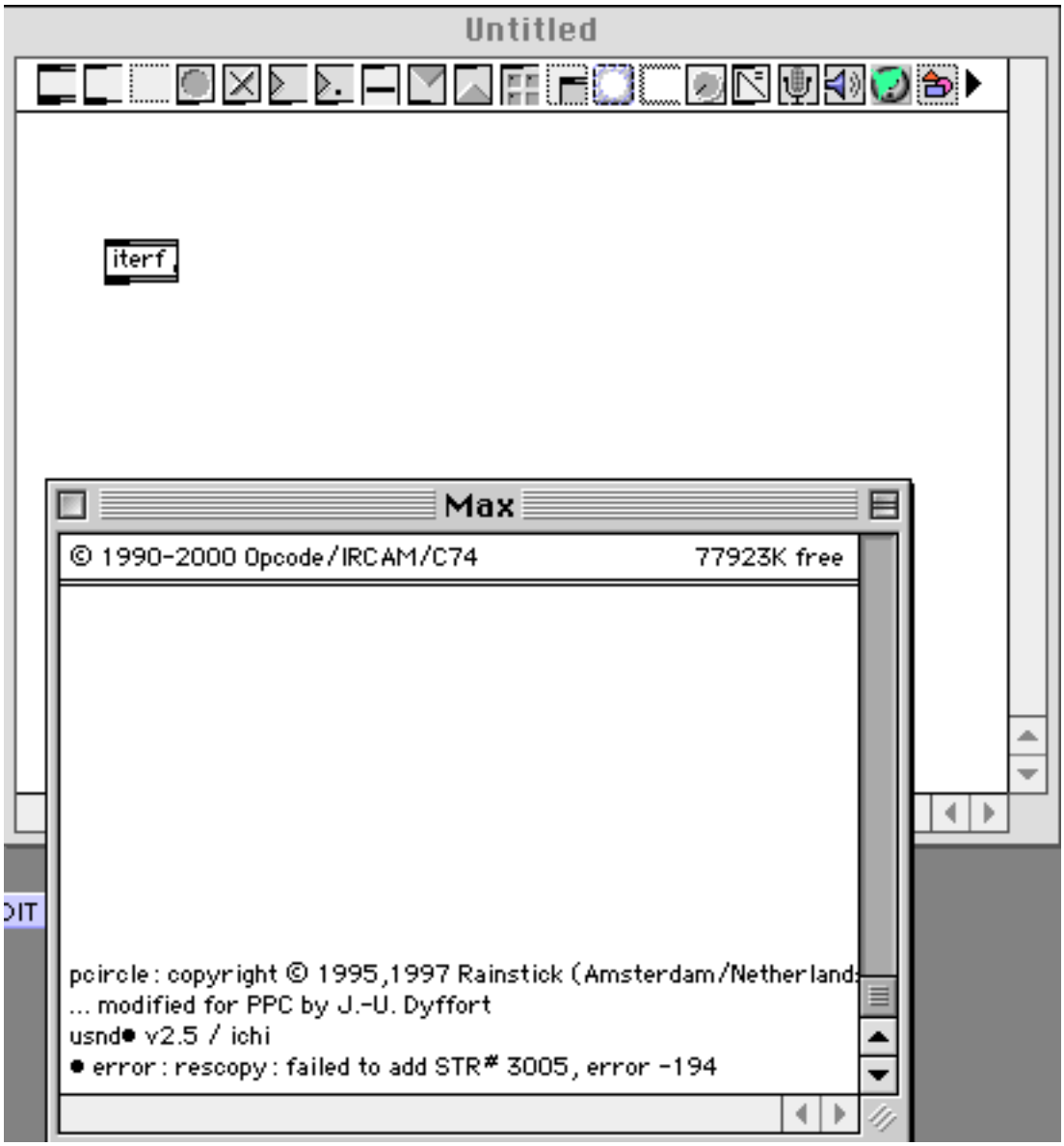
abb.9 geoeffnete cfrg



fuer jedes external sind in der cfrg fragmente enhalten. hier nur eins, da nur ein external da ist. d.h. man kann die komplette cfrg resource kopieren. das geht dann wie bei der maxL (abb.7).
sind mehrere externals vorhanden bzw. fragmente, muss fuer jedes external file, das entsprechende fragment kopiert werden. zuerst kopert man auch hier die komplette cfrg, dann loescht man die fragmente, die nicht gebraucht werden. indem man den text “Fragments #” auf dem blau gestrichelten rahmen selektiert und mit delete loescht bis das benoetigte fragment uebrig ist . dann speichert man und schliesst das fenster.

eigentlich kann man das external file jetzt schon mal testen. dazu wird es in den externals ordner von max getan. dann einen neuen patcher machen und ein objekt machen, das namen des gebastelten externals hat. entweder tauchen keine fehlermeldungen auf, eher nicht, oder aber welche, die sagen, dass STR# nicht kopiert werden koennen.

abb.10 so sieht's aus, wenn eine STR# resource fehlt.



ist also eine einfache moeglichkeit genau herauszufinden was fuer eine STR# fehlt. man kann sich natuerlich in resorcerer die in STR# enthaltenen strings anschauen, aber das ist zu muehselig. dann kopiert man einfach genau den string, der fehlt und dann ist alles gut.

ENDE