

NB0Tool.exe

*Es gibt eine Menge Leute, die sich damit befassen, nachträglich noch diverse Änderungen an den ROM-Images von Windows CE vorzunehmen. Dafür gibt es in den einschlägigen Szenen auch einiges an Tools - zumeist für die Kommandozeile. Aber wie wir alle schon bei Ida gesehen haben, sind Kommandozeilentools zumeist zwar gut gedacht, aber nicht so gut gemacht - Ida kommt hingegen interaktiv und grafisch daher und ist mittlerweile **das** Tool. Man sieht auf der Kommandozeile nicht, was man da **tatsächlich** angerichtet hat. Deshalb habe ich mir für meinen eigenen Bedarf dieses simple, aber grafische und interaktive Tool geschrieben. Man kann mit Nb0Tool.exe in *.nb0 Dateien hineinschauen und einiges darin ändern. Dieses Programm ist derzeit alles andere als ausgereift, aber vielleicht kann es der eine oder andere trotzdem gebrauchen.*

1. Was ist eine .nb0 Datei überhaupt?

Das ist ein ROM-Image, also eine Datei, die zum Schluß beim Platform-Builder von Windows CE herauskommt, wenn alles Kompilieren, Assemblieren, Linken, Zusammenstellen usw. erfolgreich gelaufen ist.

Die Dateiendung kommt von der angenommenen Busbreite. Für einen 32 Bit breiten ROM gibt es nur die .nb0, aber für andere Busbreiten - z.B. für gewöhnliche Flashroms mit 8 Bit Datenbreite - würden die Tools beim Platformbuilder das ROM-Image aufspalten in .nb0 .nb1 .nb2 und .nb3 - für jeden ROM ein Image.

Diese Datei ist also dafür gedacht, in einen ROM gebrannt zu werden, der seinerseits im Hauptspeicher des Zielsystems, also im Adreßbereich der CPU angeordnet sein soll.

Dabei ist die innere Struktur dieser nb0 Datei so ausgelegt, daß zum einen der eigentliche Kern sowie alle Programmteile, die das Windows ständig oder öfter braucht, bereits auf feste Adressen in diesem Adreßbereich gelinkt sind, so daß der Code nicht mehr geladen werden muß, sondern gleich direkt vor Ort von der CPU ausgeführt werden kann. Das nennt sich dann XIP (execute in place). Die so auf feste Adressen gebrannten Programme werden "Blöcke" genannt - im Gegensatz zu 'Files' also normalen Dateien.

Ganz wichtig: 'Files' sind in vollem Umfange gespeichert und können unbeschadet extrahiert oder ausgetauscht werden - für 'Blöcke' gilt dies aber **nicht**! Bei Blöcken ist wirklich nur das gespeichert, was zum Ausführen unbedingt notwendig ist. Alles andere ist abgeschnitten und

unwiederbringlich weg - insbesondere die Relokationsinformationen.

Zum anderen ist die Struktur der nb0 Datei so ausgelegt, daß sich daraus der Grundstock für ein Dateisystem ergibt, so daß man also mit dem Explorer zwischen diesen im ROM gespeicherten Dateien herumschauen kann, als wären es Dateien auf einer Festplatte.

2. Womit es anfang

Eigentlich brauchte ich bloß eine Möglichkeit, einem ROM-Image von Windows CE 5.0 ein paar Treiber nachzurüsten und diese in die Registry einzutragen. Dazu gibt es in den diversen Szenen einiges an Tools, die zum Bearbeiten und Verändern von ROM-Images für Windows CE oder Mobile gedacht sind. Nachdem ich auf 'dumprom' und 'dumpromx' sowie diverse Perl-Skripte 'fdf2reg.pl' und 'reg2fd.pl' gestoßen war, dachte ich, das sei ausreichend für meine Vorhaben, doch weit gefehlt.

All diese Tools haben ihre Meriten, aber auch ihre Kehrseiten. Nach einigen ersten Reinfällen habe ich mich entschlossen, mir mein eigenes Handwerkszeug zu schreiben. Doch zuerst sei all das hier mal erwähnt, worüber ich bei den vorhandenen Tools gestolpert bin.

3. dumprom und dumpromx

Man kann damit sehr schön alle enthaltenen Dateien extrahieren, sowohl Blöcke als auch Files. Aber das hat seinen Haken: dumprom(x) versucht bei Blöcken zwar, sie wieder in eine gültige PE-Exe-Form zu bringen, was auch weitgehend gelingt, aber verlorene Infos, allen voran

die Relokationsdaten kann es nicht wiederherstellen. Deshalb sind alle Dateien aus Blöcken garantiert **nicht** mehr lauffähig. Man kann sie nur noch benutzen, um z.B. mittels IDA hineinzuschauen.

Mit dumpromx kann man angeblich auch Dateien löschen, ersetzen oder hinzufügen. Im Detail funktioniert das aber nicht, denn dumpromx kann Dateien nur hinter dem RomHeader anfügen. Alle davor stehenden Dateien bleiben im Image drin - auch dann, wenn man versucht, sie zu ersetzen.

Noch etwas stört bei den beiden Tools: gelegentlich, aber nicht immer bauen sie beim Datelexport von Blöcken in den selbstkreierten PE-Header einen falschen CPU-ID ein oder vergessen bei DLL's die Exportsektion oder bei EXE's die Importsektion.

4. fdf2reg.pl und reg2fdf.pl

In einem ROM Image wird der Anfangszustand der Registry oft in einer Datei 'default.fdf' gespeichert. Um die Registry zu bearbeiten, muß diese Datei erst einmal in Text decompiliert werden. Dazu dient eines dieser Perl-Skripte. Es macht seine Sache auch ganz gut, strauchelt aber bei Multi-SZ-Einträgen. Ich habe mir deswegen einen eigenen Decompiler RCFDF.EXE geschrieben. Der ist zwar noch nicht wirklich vollständig, reicht aber momentan schon aus. Zum Compilieren benutze ich lieber das Original-Tool 'REGCOMP.EXE' aus dem Repertoire des Platformbuilders.

Ein putziges Detail ist mir dabei aufgefallen:

Wenn man eine 'default.fdf' mit RCFDF decompiliert (oder mit fdf2reg.pl und anschließend verbessert) und das Recompat dann mit REGCOMP wieder compiliert, dann ist das compilierte Ergebnis **nicht** binär gleich zur Ursprungsdatei. Wenn man aber dieses Compilat erneut mit RCFDF decompiliert und dieses zweite Recompat wieder mit REGCOMP compiliert, jaaa.. dann ist dieses letztere Compilat exakt gleich zu der ursprünglichen 'default.fdf' - Byte für Byte.

5. Boooff

Das ist ein grafisch aussehendes Tool vermutlich russischen Ursprunges, was aber zumindest für mich schlichtweg kryptisch und unbenutzbar ist.

6. NB0Tool.exe = mein eigener Senf

Es ist ein grafisches Tool und läuft unter Windows. Man arbeitet mit einem virtuellen ROM von 32 oder 64 MB Größe, wo man eine vorhandene nb0 Datei hineinladen, betrachten und in Grenzen manipulieren kann. Benutzt habe ich es bislang **nur** mit Images von Windows CE 5.0.

Nach dem Start wählt man erstmal die virtuelle ROM-Größe aus. Ebenso kann man jetzt noch aussuchen, ob das Tool beim Auffüllen unbenutzten Platzes mit 0 oder FF auffüllen soll.

Danach wählt man die nb0 Datei aus, das Tool lädt sie in den virtuellen ROM und analysiert sie soweit es geht. Das Ergebnis wird in einer Liste dargestellt: Zuerst der Rom-Header, dann die Module und dann die Files. Hier kann man bereits Files in der Liste anklicken und speichern. Das eventuelle Dekomprimieren erledigt das Tool intern. Man kann jedoch das File auch roh, also so wie ist speichern und dann mit anderen Tools dekomprimieren.

Falls man eine Reihe von Dateien zu komprimieren oder zu dekomprimieren hat, kann man das eingebaute Batch- 'Compress-Tool' benutzen. Es ist ein Frontend zu den bekannten compress-dll's und hat 2 Fenster: ein Quell-Fenster und ein Ziel-Fenster. Für beides sollte man sich getrennte Verzeichnisse auf dem PC aussuchen. Im Quellfenster kann man beliebig viele Dateien markieren. Drückt man dann auf den Start- Knopf, werden diese dann komprimiert oder dekomprimiert.

Die eigentliche Arbeit des NB0Tools wird jedoch in der sortierten Liste gemacht. Diese Liste enthält alle Bereiche des ROM-Images nach Adressen sortiert - einschließlich der 1..3 Byte kleinen Reste, die entstehen, weil alle Bereiche auf geraden DWORD-Adressen (0,4,8,C...) angeordnet sind.

In dieser Liste kommen die festen Blöcke nur als namenlose Bereiche vor, weil man daran mit diesem Tool ja sowieso nichts ändern kann und sollte. Aber bei den Files kann man einiges tun: Dateien exportieren und importieren, Dateien löschen, ersetzen und Attribute ändern. Es gibt auch einen Knopf für beliebig viele Hex-Viewer, mit denen man sich Inhalte ansehen kann.

Ähem.. halt, es geht nicht **alles**. Wenn hinter der betreffenden Datei kein fester Block liegt, dann geht alles, sonst nicht alles.

7. Dateien und Blöcke löschen

Dateien können gelöscht werden, wenn hinter ihnen kein fester Block liegt. Beim Löschen rückt alles nach vorne auf, was hinter der gelöschten Datei liegt. Dateien haben einen Dateinamen als Asciiiz-String, der irgendwo im Image liegen kann - unter Umständen auch innerhalb des Konstantenbereiches eines Programmes unter den Blöcken. Dieser String wird **nur dann** gelöscht, wenn er direkt vor der Datei liegt. In allen anderen Fällen wird er beim Löschen der Datei verwaist.

Man kann auch - mit dem nötigen Willen zur Unat - einen Block löschen. Was man damit für einen Schaden anrichtet, muß man aber selber wissen.

8. Datei ersetzen

Auch hier gilt, daß man immer dann eine Datei ersetzen kann, wenn hinter ihr kein fester Block liegt. Je nachdem, ob die neue Datei größer oder kleiner ist, rücken alle nachfolgenden Daten vor oder zurück. Die neue Datei wird so eingebaut, wie sie ist. Es findet also keinerlei Komprimierung statt - die muß vorher erledigt werden, wenn man sie will.

Der Dateiname bleibt, wo er ist. Beim Directory-Eintrag bleibt auch alles so, wie es war - mit einer Ausnahme: Bei der Real-File-Länge trägt mein Tool erstmal die Länge der gelesenen Datei ein. Bei komprimierten Dateien muß man also von Hand die entkomprimierte 'Real'-Länge nachtragen.

Wenn die zu ersetzende Datei vor einem festen Block liegt, dann darf die neue Datei nicht größer sein als die bisherige. Ist das gegeben, dann kann man auch diese Datei ersetzen.

9. Datei hinzu

Solange noch Platz ist im virtuellen ROM, kann man beliebig viele Dateien zum Image hinzufügen. In jedem Fall sollte man den zugehörigen Directory-Eintrag editieren und bei komprimierten Dateien die entkomprimierte Länge nachtragen.

10. Image speichern

Ja, und zum Schluß sollte man sein editiertes ROM-Image speichern. Automatisch macht es dieses Tool nicht.

Man kann auch die angezeigten Listen als Text speichern.

11. Absonderlichkeiten

Eigentlich sollte man meinen, der Magic 'ECEC' und die 2 darauf folgenden Zeiger zum RomHeader sollten dem WinCE ausreichen - doch damit weit gefehlt. Aktualisiert man bei Änderungen im Image nur diese Zeiger, dann bootet das Image nicht. Der Grund ist, daß es am Anfang der nk.bin nochmal einen Zeiger auf den Rom-Header gibt - und **den** benutzt der Kernel. Ob das bei allen Images so ist, weiß ich nicht, aber bei **meinem** aktuellen Vorhaben brauch ich das. Also hab ich in mein Tool noch eine Suche nach diesem zweiten Zeiger und dessen Aktualisierung eingebaut.

12. Was NB0Tool nicht kann

Ich habe mir dieses Tool geschrieben, um erstmal auf die Schnelle wenigstens irgend etwas zu haben, mit dem ich tun kann, was ich tun will. Deshalb ist eine Menge dessen, was man üblicherweise einbaut, schlichtweg nicht vorhanden: Fehlerprüfungen aller Art, Code-Optimierung usw. Dieses Tool ist also nicht idiotensicher und auch nur mit wenigen ROM-Images ausprobiert.

Eines weiß ich: die beim Simulator von WinCE 6.5 vorhandenen ROM-Images lassen sich nicht mit NB0Tool anschauen. Es findet zwar den ROM-Header, aber dieser ist wohl so sehr anders als der von WinCE 5.0, daß aus allem Weiteren nichts wird.

Ob ich NB0Tool weiterentwickeln werde, weiß ich nicht. Eher ist es wahrscheinlich, daß ich es nicht tue, sobald seine Arbeit getan sein wird.

W.S.

Berlin, im Herbst 2011