Entis CSX: Cotopha Image file
http://honyaku-subs.ru/forums/viewtopic.php?f=17&t=95 Page **1** of **1**

Author: **Proger_XP** [May 30, 2009, 00:50]
Post subject: **Entis CSX: Cotopha Image file**

So with ... Recently dismantled one format, I would like to save it somewhere to help future generations Thank you for sending WinKi

- all numbers in the text - hex.
- if I write "usually", which means that in all / most cases in my file there was this value.
- [DW] - number of type DWord; [S] - a sequence of characters (string); [S8] - eight characters; [US] - Unicode string (see below)
- **Program for translation** will be posted here: http://www.solelo.com/p4s

Entis CSX: Cotopha Image file

(Wanko to Lily, system \ wanko.csx, 3,813,713 bytes)

Code:

```
The general format of CSX-file:
.00 .. 40 - CSX file header
.41 + - Sections

CSX file header:
.00: [S] "Entis" # $ 1A # 00 # 00
...
.0 C: [DW] Should be zero
.10: [S] "Cotopha Image file"
...
.38: [DW] Contents size
...
```

where:

- Should be zero - field must be zero, otherwise the engine will report an incorrect file header. Maybe this is the version zarezirvirovanoe field extension code format or something.
- Contents size - the size of the entire file, except for its title (in other words, the value + 40 should be exactly 1 byte larger than the whole file).

CSX-file consists of sections. They have five

- image - the actual script commands and text. Largest section.
- function - this function table, and apparently the entry points or something else. Is exactly where you have the right shift.
- global - did not go into the details, does not affect the possibility of transfer.
- data - the same thing here.
- linkinf - I do not know exactly what it is, because in the game, this section was empty ...

Section has the following format:
Code:

```
.00: [S8] ID
.08: [DW] Size
.0 C: [DW] Unknown (usually 0)
```

where:

- ID - type of sections (one of five). Additional space is padded with spaces (20, no zeros).
- Size - size of the section without header.

Some sections are optional reading [DW] or more.

Unicode string

The file uses a common format for all strings, which, incidentally, all Unicode (except ID sections that are ASCII). Here I will be denoted [US]:
Code:

```
.00: [DW] Character count
.04: [S] Unicode string
```

where:

- Character count - the number of **characters** (not bytes). To get the length of the string must multiply it by 2.
- Unicode string - for example, 0030 BF7D ...

Linkinf section

Since no interest in this section, I can only say that it has a title (in addition to the normal cell):
Code:

```
.00: [DW] Count 1
.04: [DW] Unknonwn
.08: [DW] Count 2
.0 C: [DW] Unknown
```

Linkinf has two parts, each of which has a number of "something."

Data section

Also not the most interesting section.

Code:

```
Has a header of 4 bytes
.00: [DW] Records count

Record:
.00: [US] Name 1
+00: [DW] Unknown (usually 0)
.04: [DW] Unknown (usually 80)
.0 C: [US] Name 2
```

Examples of (Name 1 => Name 2):

Code:

```
input => InputFilter
screen => Window
frameskin => ResourceManager
```

Global section

This section is, again, not particularly interesting, since it does not affect the translation. But here's what we know about her:

Code:

```
Has a header of 4 bytes
.00: [DW] Records count

Record:
.00: [US] Name
+00: [DW] Unknown
.04: [DW] Unknown 2
```

It is possible that Unknown 2 is the length of one more Unicode-string.

Function section

But this is the most interesting place. It stores pointers into the section image, so here we need Fix shifts when the script.

Section consists of two parts. The first was simply stored offset in the second - a table with the names and offsets.
All addresses in this section are relative to the section image. So to get the absolute URL, we add to them the address of the start image (usually 50).

The first part
Code:

```
Has a header of 4 bytes
.00: [DW] Records count

Record:
.00: [DW] Relative offset
```

where:

- Relative offset - otnositelnyy address section image.

The second part
Code:

```
Has a header of 8 bytes
.00: [DW] Unknown (usually 0)
.04: [DW] Records count

Record:
.00: Relative offset
.04: [US] Name
```

where:

- Relative offset - otnositelnyy address section image.

Image section

Messages are two functions: Talk and Mess. Waiting for the reaction function HitretNewPage user and displays the following message (in a new screen).

Talk feature use and Mess is that Talk (apparently) is always used for messages, not the speech, *but* not always Ondo message will only function Talk. The message is usually divided so that Talk displays the first 26 characters (the maximum that can fit on the screen in one line), and the rest is output functions Mess (apparently also one function on one line).
It is interesting that any long string displayed normally even one function. It is not clear why the script is broken down in this way.

At the same time Mess used for speech output (without the initial call Talk). You can divide a message with catching HitretNewPage.

The procedure for the parser for the script (this is a screen enclosure of RTF)

Attachment:

Cotopha parser jumps.rar [03.03 KB]

Downloads: 288

Now a few words about each function.

Talk

Code:

```
.00: [US] "Talk"
+00: [DW] Unknown (usually 01 02 00 06)
.04: [US] Message
```

Mess

Code:

```
.00: [US] "Mess"
+00: [DW] Unknown (usually 01 02 00 06)
.04: [US] Message
```

HitretNewPage

Code:

```
.00: [US] "HitretNewPage"
```

This is enough to change the strings.

Where and what to change?

If you change the line in the image section to change the following places in the CSX-file:

- @ 38 - the size of CSX-Failla without a title.
- "Image" section:
- - @ 08 - the size of the section;
- - Unicode-length string in the script.
- - Correct distances in short jumps, see below.
- "Function" section:
- - Change the address in the first part of the section;

- - To do the same in the second part.

Short hops

The script may meet function jumps. Usually there is a jump indicates the relative distance (ie, number of bytes to skip.) Because of this, it is necessary not only to change the address in the offset table (functions, see above), but also keep track of these jumps in the script and edit distance in them if necessary.

For example, a function *IsGameClear,* in which the distance is recorded here: [us] *IsGameClear* [w] [dw] *distance* (that is, 2 unknown bytes after the function name, and then offset the size DWord). The same syntax have similar functions: *ChkFlagOn, ChkSelect* and *OnFlag.*

Function names, as usual, in the Unicode (for the English names of the symbol is simply the code + 0x00 for the 2nd byte).

One more thing. Functions hops can sometimes refer to a special code, which in this context is also bound. This code - one byte = *0x06* followed the 4-byte range dyal jump: [b.] *0x06* [dw] *distance.* So such "jumps" are also needed to monitor and tinker.

In general, the script jumps abound - for example, in *Yosuga no Sora* only 4 choices, but the script is already 22 jump.

In conclusion ...

The good news is that the engine always uses Unicode, so the problems with any language (including our favorite Cyrillic) no. Very curious fact that, if you turn off hard returns in the Japanese style in a small hack, the engine itself will carry Russian and English words and even add them to the transfers.
By the way, the engine is running and without installing the Japanese locale.

That's all.