Tunguska Manual 0.0.3

Viktor Lofgren <vlofgren@gmail.com>

http://www.acc.umu.se/~achtt315/tunguska/

2008-03-25

Contents

1 Introduction **3**
1.1 The fine print . . . 3
1.2 Foreword . . . 3

I The software **4**

2 The emulator **5**
2.1 Getting started . . . 5
2.2 Command line arguments . . . 5
2.3 The interface . . . 6
2.3.1 Special keys . . . 6
2.3.2 Indicator lights . . . 6

3 The assembler **7**
3.1 Conventions . . . 7
3.2 Command line arguments . . . 7
3.3 Numerical constants . . . 7
3.4 Instructions and addressing modes . . . 7
3.4.1 Addressing modes . . . 8
3.5 Assembler macros . . . 8
3.6 Inline arithmetics . . . 8
3.7 Labels and assembler variables . . . 9

II Introduction to Ternary Computing **10**

4 Numerical representation **12**
4.1 Conventions . . . 12
4.2 Ternary numeral base . . . 12
4.2.1 Balanced ternary . . . 12
4.2.2 Compact representation . . . 13
4.3 Ternary representation on binary computer . . . 13

5 Ternary logic **14**
5.1 Conventions . . . 14
5.2 Ternary logic . . . 14
5.2.1 Logical conditionals . . . 14
5.2.1.1 Material conditional . . . 14
5.2.1.2 Logical biconditional . . . 14
5.2.2 Ternary operators . . . 15
5.2.2.1 $\bigwedge$ (AND) . . . 15
5.2.2.2 $\bigvee$ (OR) . . . 15
5.2.2.3 $\bigoplus$ (XOR) . . . 15
5.2.2.4 ~ (NOT) . . . 15
5.2.3 Non-boolean operations . . . 15
5.2.3.1 Shift . . . 15
5.2.3.2 Shift| . . . 16
5.2.3.3 BUT . . . 16
5.3 Truth tables . . . 16

A Tunguska specifications **17**
A.1 Registers . . . 17
A.1.1 Processor status register specification . . . 17
A.2 Op-code specifications . . . 18
A.2.1 Addressing modes . . . 18
A.2.2 Operations . . . 18
A.3 Reserved addresses . . . 20
A.3.1 Screen . . . 20
A.3.1.1 Vector mode . . . 20
A.3.1.2 Raster mode . . . 20
A.3.1.3 Text mode . . . 20
A.4 Interrupts . . . 21
A.4.1 Disk I/O . . . 21
A.4.2 Auxiliary General Data Processor (AGDP) . . . 22
A.5 Floating point . . . 22
A.6 Notes for 6502 programmers . . . 22
A.7 Debugging . . . 23

B The GNU Free Documentation License **24**

Chapter 1

Introduction

1.1 The fine print

1.2 Foreword

This is a multi-part document covering the use of the Tunguska Emulator, the Tunguska Assembler, a brief introduction to Ternary Computing, and the Specifications of Tunguska.

Tunguska is by no means affiliated with any organization, religion, political movement, nationality, sports team, text editor or pizza topping; mentioned, insinuated, or otherwise referenced. It is non-profit, non-affiliated, and largely only exists for the fun of it.

If you do find some horrible crime against mathematics or grammar in here, first make sure this is the latest version of the document by visiting the Tunguska website, and if it indeed is the latest version you've found the error in, feel free to send me an email with a suggested correction.

Part I

The software

Chapter 2

The emulator

2.1 Getting started

To compile and install Tunguska, simply do the following

```
> cd tunguska-version
> ./configure
> make
> make install
```

To be of any sort of use, Tunguska needs a memory image to load. Starting tunguska without one would be the equiavalent of booting your computer without any sort of hard drive. Don't worry, you don't need to illegally download some proprietary image off some murky backwater website. All you need to get started ships with Tunguska. After compiling and installing Tunguska itself, you need to assemble an image, and this is how you do it.

```
> cd tunguska-version/memory_image
> tg_assembler -o image.ternobj ram.asm agdp.asm stdlib.asm screen.asm
```

You're now ready to load the image with Tunguska

```
> tunguska image.ternobj
```

You can also assemble disk images, and load them with tunguska. To do this, you need to find the source code for some such image, I suggest you download the pong clone from Tunguska's website *http://www.acc.umu.se/~achtt315/tunguska/* and put it in the same directory as the other .asm-files. You're now ready to assemble

```
> tg_assembler -o pong.ternobj pong.asm
```

And here's how to load it in Tunguska

```
> tunguska -F pong.ternobj image.ternobj
```

Now, from the tunguska prompt type

```
> LOADSUB
```

And you're done! That's all there is to it. Now you'll probably want to dive in and make your own programs. Good for you.

The most helpful tip on that (except reading this document) is to look up a 6502-assembly reference or guide on the internet. Tunguska is compatible enough for most of the stuff you'll find there to be applicable. If something doesn't work as expected, head over to **Appendix A** and check so the instruction is still there (some instructions have changed names, some have been dropped).

2.2 Command line arguments

Tunguska takes the following command line arguments:

-h Display help message

-f Full screen

-T Specify floppy image

-Z Don't attempt to actually load the floppy, assume it's zero.

2.3 The interface

The border around the window indicates the size of the raster window (bright) and the text/vector window(dark).

2.3.1 Special keys

Esc	Exit
F9	Enable trace mode, spew the location of the program counter to standard output.
F10	Debug, print all registers to standard output.
F11	Step instruction (when in pause mode).
F12	Send interrupt
Pause	Toggle pause

2.3.2 Indicator lights

Top-left of the Tunguska window is three indicator ligths, labelled R, G, AG. They correspond to

R	Running when lit, paused when not
G	Graphics mode (see specifications)
AG	Auxiliary graphics mode (see specifications)

Chapter 3

The assembler

3.1 Conventions

The following document uses the following conventions, anything inside hard brackets [like this] is optional. Anything inside braces and separated by pipes {like|this} is a situation where you must select one of the options (in the example, either like or this). 'n' means any numeral, 's' means any string, 'a' means any address or label name, 'l' means explicitly any label name.

3.2 Command line arguments

The assembler takes the following command line arguments:

-v Verbose mode

-o Specify output

-h Display help message

3.3 Numerical constants

There are two main numeral classes in the Tunguska assembler , trytes and words. Trytes have trit-width 6, and words have trit-width 12. A tryte can generally be used in place of a word, but a word can not be used in place of a tryte. There are two special operators, 'LOW' and 'HIGH' that allow you to extract the individual trytes of a word, and use that as a tryte.

In the strictest sense, it is sometimes possible to use a word in place of a tryte without the assembler complaining, but it isn't recommended.

There are two allowed numeral bases, decimal and balanced nonary. Balanced nonary allows numbers in the range -4,...,4; but since there is no symbols for negative numbers, the letters A,...,D are used to symbolize -1,...,-4. Balanced nonary is to ternary roughly what octal is to binary.

31 Regular decimal. No prefix required. Since it is within +-364, both a word and a tryte.

1000 Regular decimal. No prefix required. Only a word, not a tryte.

%0DA Nonary triplet. Always a tryte long.

%0DA114 Nonary sextet. Always a word long.

%000000 Nonary sextet. Always a word long, even if it is smaller than 364.

LOW %0DA114 Lower tryte of nonary sextet. Equivalent to %114.

3.4 Instructions and addressing modes

In general, there is no forbidden instructions in Tunguska. Even if you try to pass an unexpected addressing mode to an operator, the whole machine shouldn't crash and burn. There can be unexpected behavior though, so don't do it on purpose. For a complete list of Tunguska instructions, see the machine specifications.

The tunguska assembler expects all instructions to be uppercase, any lower case instructions will be interpeted as labels or variables, and the assembler will complain.

3.4.1 Addressing modes

OP Implicit addressing. No argument.

OP A Accumulator. Whatever operation is done with the accumulator as argument. Strictly speaking, this is the same as implicit addressing.

OP #n Immediate addressing. Whatever operation is done with the directly specified numeral as argument.

OP a Absolute addressing. Whatever operation is done on the memory at address a.

OP a,X Absolute addressing with X offset. Whatever operation is done on the memory at address a+X.

OP a,Y Absolute addressing with Y offset. Whatever operation is done on the memory at address a+Y.

OP (a) Indirect addressing. Whatever operation is done on the memory pointed to by the memory at address a.

OP (a,X) Indirect addressing with X offset. Whatever operation is done on the memory pointed to by the memory at address (a+X).

OP (a),Y Indirect addressing with Y offset. Whatever operation is done on (the memory pointed to by the memory at address a) + Y.

OP X,Y XY-addressing. Whatever operation is done on the memory pointed to by X:Y.

3.5 Assembler macros

The Tunguska assembler supports a series of pseudo-instructions, or macros that do not affect the machinecode itself, but allows the assembler to enter non-generated data, or perform other operations.

As a rule of thumb, all macros begin with an **@**, and are all uppercase.

@DT {n|s}[, {n|s}, ...]

Define tryte. Accepts a comma separated list of numerals and strings. These are entered into the assembled memory output as-is.

@DW {n|s|a}[, {n|s|a}, ...]

Define word. Accepts a comma separated list of numerals, strings or memory addresses (labels).

@REST {n|s} [{n|s} = 0]

Reserve argument1 number of trytes into memory, set them to argument2.

@EQU l {n|s|a}

Set variable argument1 to argument2. For most intents and purposes, this is identical to jumping to argument 2 and declaring a label there.

@ORG {n|a}

Jumps instruction counter to address or label specified. Interpret as "this is where I want the following code to go into memory."

3.6 Inline arithmetics

The Tunguska assembler has support of inline arithmetics, that is, it can calculate pretty much any algebraic function based on values available to it at assembly-time. A magic **$$** token is available, resolving to the address of this (the current) memory position. It works pretty much like you'd expect it to, with regular infix syntax. The only quirk is that you can't use paranthesis for precedence override, instead you must use braces.

This works: {1+2} * 3 - 5 + $$ - somelabel*2

This doesn't: (1+2) * 3 - 5 + $$ - somelabel*2

3.7 Labels and assembler variables

While labels and assembler variables are in many ways interchangable, there are a few differences. Labels can have local child labels and child variables that, from within the label are accessible through .localname and from outside the label through label.localname, but a variable can not.

A label is declared through **labelname:** in the beginning of a line, and accessed through substituting labelname wherever an address is requested.

A variable is declared through **@EQU variable value**, and is accessible in much the same way a label is. Futhermore, it is possible to store not only words, but trytes in variables, which can be accessed for an instance in immediate addressing mode like this: **OP** #**variable**.

A very powerful combination is the **$$** -token and variables. For an instance, if you want to determine the length of a string automatically for use later, you can use a construct like this:

```
mystring:  @DT     'Hello world!', 2, 'How are you doing?'
           @EQU    .length         $$ - mystring
```

The length of **mystring** (which reads: 'Hello world![new line]How are you doing?') will be stored in **mystring.length** at no cost of machine memory.

Part II

Introduction to Ternary Computing

Notes

This is more or less original research, and on some parts it might be downright erroneous. See it as a crash course into the relevant parts of Ternary computing to Tunguska. It's unfortunately a bit esoteric in it's fairly heavy use of mathematical and logical symbols and conventions.

More resources

There is a multitude of resources on Ternary Computing available on the Internet, though they can be difficult to find. Here are some of them:

http://xyzzy.freeshell.org/trinary Trinary Computer Systems. Very long and complete document about ternary/trinary computers.

http://jeff.tk/wiki/Trinary Trinary - Jeff.tk (Some sort of project to buld a ternary computer? Has a lot of useful links, not very clear on what the purpose of the page is.)

http://www.ternary.info/ Ternary.info - special interest group on balanced ternary numeral system and trinary logic (Ternary site mostly in Russian, has helpful people in it's forum.)

http://en.wikipedia.org/wiki/ Lots of information under they keywords "**Balaned ternary**", "**Ternary logic**", "**Ternary numeral system**", ...

http://www.trinary.cc/ Mostly hardware-oriented trinary computer site.

Chapter 4

Numerical representation

4.1 Conventions

I will use subscript to indicate numeral base. n_{10} is decimal, n_3 is ternary, n_{3b} is balanced ternary, n_{9b} is balanced nonary. When nothing else is indicated, assume decimal.

I will use vector form to represent a number, in this fashion:

$$n =< a_{N-1}, a_{N-2}, ..., a_1, a_0 >$$

n will mean arbitrary number, **N** the number of digits in a number, **a** will mean a digit in given number.

4.2 Ternary numeral base

The **regular ternary** numeral system uses base 3, that is, it has three digits: 0, 1 and 2. The value of a ternary number is formally defined as

$$V = \sum_{i=0}^{N} 3^i a_i \tag{4.1}$$

Where a_i is the i:th digit (and a_0 is the least significant digit). More practically, this means that the ternary number 201, in decimal is

$$210_3 = 2 \cdot 3^2 + 0 \cdot 3 + 1 = 19_{10}$$

There really isn't anything special or spectacular with this numeral base, it works much like you'd expect any old base to do. Much like any conventional base, it's range is

$$0 \leq X < 3^N \tag{4.2}$$

where N is the number of digits. At a fixed word size, addition with consideration to spillover is defined by

$$T_0 + T_1 = (T_0 + T_1) \ \boldsymbol{mod}\ 3^N \tag{4.3}$$

4.2.1 Balanced ternary

The **balanced ternary** numeral system is a non-standard base. It still follows equation 1, but the digits are shifted one step down. That is, it's digits are -1, 0, 1; but since there is no symbol for the digit -1, it's conventional to call the digits N, 0 and P. Again, to supply an example:

$$PN0_{3b} = 1 \cdot 3^2 + (-1) \cdot 3 + 0 = 6_{10}$$

An important thing to note is it's range. Note the extreme difference between the range of regular ternary in equation (2), and the range of balanced ternary:

$$-\lfloor \frac{3^N}{2} \rfloor \leq X \leq \lfloor \frac{3^N}{2} \rfloor \tag{4.4}$$

Addition with consideration to spillover is defined by

$$T_0 + T_1 = (T_0 + T_1 + \lfloor \frac{3^N}{2} \rfloor) \ \boldsymbol{mod}\ 3^N - \lfloor \frac{3^N}{2} \rfloor \tag{4.5}$$

Where T_1 is positive. Do note the difference between equations (5) and (3).

The biggest difference between balanced bases and non-balanced "regular" bases is how negative numbers are handled. There is no neat way of handling negative numbers in non-balanced bases, so you have to invent work-arounds. These work-arounds are generally eyesores, either because they allow illegal states (negative zero in one's complement), or because they generate asymmetric ranges (two's complement).

This issue is non-existent in balanced bases, since negative numbers are inherently supported. All you need to do to invert a balanced number is invert all the digits (6).

$$-n = - < a_{N-1}, a_{N-2}, ..., a_1, a_0 > = < -a_{N-1}, -a_N, ..., -a_1, -a_0 > \tag{4.6}$$

4.2.2 Compact representation

There are two major ways of representing ternary numbers in a more manageable way: **Nonary** (Base 9) and **Septemvigesimal** (Base 27). Septemvigesimal is very unpractical, since it requires symbols for 17 additional digits beyond our decimal 0...9.

Nonary is much more manageable. It is very close to decimal in data density and therefore easy to "intuit" the value of. Nonary should be easier to wrap your mind around than say hexadecinal which is more alien to decimal, data-density-wise.

In this document, we're interested specifically **balanced nonary**. It has nine digits, -4, -3, -2, -1, 0, 1, 2, 3, 4. Again, we lack symbols for negative digits, and for this reason, I will introduce the letters A...-D to represent -1,...-4.

It's value is governed by the following formula:

$$V = \sum_{i=0}^{N} 9^i a_i \tag{4.7}$$

Example:

$$3AD_{9b} = 9^2 \cdot 3 + 9 \cdot (-1) + (-4) = 230_{10}$$

Each balanced nonary digit represents two balanced nonary digits.

4.3 Ternary representation on binary computer

There is no such binary word size b that it's possible to represent a ternary word of size t without superfluous states. This follows from the prime uniqueness theorem, which simplified to fit this case states that

$$3^b \neq 2^t \forall b, t \neq 0$$

So the quest is not to find a lossless representation of ternary values on binary computer, but minimizing loss while maximizing usability and speed. Assume ternary word size 6, 729 states. The smallest number of binary bits that can represent that many states is 10, 1024 states. That's 295 meaningless states. Furthermore, there is no real obvious connection between individual bits in the binary representation of the ternary word, and the digits of the ternary word.

Obviously, this is not a good representation. It is small, but quite useless. Instead, if one represents a ternary digit with two bits. This representation squanders states pretty carelessly, but it is by far the most easily used one. There is a clear tie between bits and ternary digits, and it is quite fast.

It is also worth mentioning that it is possible to implement ternary logic with memory pointers, where the value is represented by a pointer to a memory location containing a true-false value, but this memory location can also be some invalid value (typically NULL), meaning the third state. However, this is even worse from a state point of view. On a 32 bit computer, $2^{32} = 4,294,967,296$ states (the size of a memory address) are used to store one state, and the other two states are stored in hopefully one bit, but more likely 8. So, worst case scenario, that's using $2^{32} + 2^8 = 4,294,967,552$ states to store 3 states. Needless to say, this is not something you'll want to use for speed or size. No matter how you twist and turn this, it will never be an effective solution[1].

[1] For creating a ternary computer, emulated or in hardware. There is probably some cases where this sort of construction can be useful in other fields.

Chapter 5

Ternary logic

5.1 Conventions

In this part, I will abbreviate TRUE to T, UNKNOWN to U, FALSE to F; furthermore, I will use standard logic symbols

- $\bigwedge$= AND
- $\bigvee$= OR
- $\bigoplus$= XOR
- ...

5.2 Ternary logic

Boolean logic, TRUE or FALSE-logic, while immensely powerful, is limited due to the fact that it only deals in absolutes. Answer the question "Was it raining in western Beijing 4:39 PM, March 13 1839?" in boolean logic. Unless you're an archeometeorologist, are exceptionally old, or own a time machine; you won't know the answer, and therefore, you will be unable to answer either true or false.

But, note well how your head is not spinning and sparking and you how you are not shrieking "DOES NOT COMPUTE!" repeatedly, because unlike robots in science fiction films from the 50's, you understand ternary logic.

Ternary logic isn't something new and scary, it's something old and mundane. It's, just as boolean logic, a formalization of human logic. Only difference is that ternary logic introduces a third state. There are multiple ideas of what this third state should be; unknown, both, neither, a zen non-answer. None of them are inherently *wrong*, they are all ternary logic. I will be describing true-unknown-false logic.

It's important to realize that this is just one possible interpretation of ternary logic. Even though I will be using "proof"-like devices to to derive the output of the functions, this is merely so that the system of logic formulated is self-consistent, and compatible with booelan logic. This also means the proofs can be more lax, and call on common sense more often than otherwise merited.

5.2.1 Logical conditionals

What does this mean for logical conditionals? Let's take a look at them.

5.2.1.1 Material conditional

Material conditionals don't need that much modification. Assume you have a rule "If P, then Q", or in logical symbols $P \rightarrow Q$, then since modifying existing boolean rules is undesirable, it is necessary to make a rule for the case P is Unknown. It is not hard to convince yourself that if P is unknown, and P implies Q, then Q must also be unknown. We can also protect ourselves against the logical fallacy of affirming the consequent by introducing a rule that states that if P implies Q, and Q is true, then P is unknown.

5.2.1.2 Logical biconditional

Logical biconditionals are also pretty much the same. Assume you hae the rule "P if and only if Q", or in logical symbols $P \leftrightarrow Q$, then the case of concern is when P is unknown. Since all cases involving true or false, for both P and Q, are already taken by boolean logic, then a logical biconditional with either P or Q unknown must imply that the other is also unknown.

5.2.2 Ternary operators

Boolean logic isn't wrong, it works and it arrives at the correct results, so whatever extensions are made to it must preserve backward compatibility with boolean logic, so all standard boolean operators with exclusively true or false values must result in the same values you would expect them to with conventional boolean logic.

5.2.2.1 $\bigwedge$ (AND)

The AND operator can pretty much be intuited from what we know of it's boolean counterpart. We know AND is commutative, so the only cases we need to deal with are:

- T $\bigwedge$ U = U; Since $T \bigwedge F \neq T \bigwedge T$, the result must be U.
- F $\bigwedge$ U = F; Since $F \bigwedge F = F \bigwedge T = F$, the result must be F.
- U $\bigwedge$ U = U; Since both arguments are unknown, nothing is known about the result.

5.2.2.2 $\bigvee$ (OR)

The same logic used to deduce the results of the AND operator can be used to deduce the results of the OR operator. It also is commutative, so the cases are the same:

- T $\bigvee$ U = T; Since $T \bigvee F = T \bigvee T = T$, the result must be T.
- F $\bigvee$ U = U; Since $F \bigvee F \neq F \bigvee T$, the result must be U.
- U $\bigvee$ U = U; Since both arguments are unknown, nothing is known about the result.

5.2.2.3 $\bigoplus$ (XOR)

The same logic used to deduce the results of AND and OR also works for XOR. XOR is also commutative, so these are the cases we must consider:

- T$\bigoplus$U = U; Since T $\bigoplus$ F $\neq$ T$\bigoplus$ T, T$\bigoplus$U=U.
- F$\bigoplus$U = U; Since F $\bigoplus$ F $\neq$ F$\bigoplus$ T, T$\bigoplus$U=U.
- U$\bigoplus$U = U; Both arguments are unknown, nothing can be said about the result.

A small point of interest, if ternary logic is represented with balanced ternary, so that T=1, U=0, F=-1, then $P \bigoplus Q = -(P \cdot Q)$, which makes $\bigoplus$ very useful in ternary computing, since it both does masking and inversion.

5.2.2.4 ~ (NOT)

If nothing is known about a variable, then nothing can be known about the inverted variable. Therefore, ~U = U.

5.2.3 Non-boolean operations

There are **many** possible ternary logical operations. First, consider binary logic. There are 16 possible bivalent logical operations. Ternary logic has as 3^9, or 19683 possible loical operations. 729 of them are commutative.

5.2.3.1 Shift

The shift operator is an operator of convenience. It doesn't correspond to any particular logical relationship. What it does is essentially addition with wrap around. With balanced ternary, the function is defined as

$$S(P,Q) = \begin{cases} 1 & P+Q=-2 \\ -1 & P+Q=-1 \\ 0 & P+Q=0 \\ 1 & P+Q=1 \\ -1 & P+Q=2 \end{cases}$$

5.2.3.2 Shift|

The shift| operator (called permute in Tunguska, from it's ability to permute values) is like shift, without the wrap-around.

$$S|(P,Q) = \begin{cases} -1 & P+Q<0 \\ 0 & P+Q=0 \\ 1 & P+Q>0 \end{cases}$$

5.2.3.3 BUT

The BUT operator is a complement to AND and OR. It does not correspond to any semantic operator. It is part of the same family of operators as AND and OR, with a 5:3:1 distribution of values. AND has 5F:3U:1T, OR has 5T:3U:1F; while BUT has 5U:3F:1T.

There are many other 5:3:1-operators, truth to be told, BUT is only implemented in Tunguska because it is in TriINTERCAL, so not to hang any TriINTERCAL-users out to dry without their favorite operator.

5.3 Truth tables

Here are truth tables for the hitherto mentioned logical operations.

A	**B**	~**A**	$A \bigwedge B$	$A \bigvee B$	$A \bigoplus B$	**A** BUT **B**	**A** shift **B**	**A** shift| **B**
T	T	F	T	T	F	T	F	T
T	U	F	U	T	U	U	T	T
T	F	F	F	T	T	F	U	U
U	T	U	U	T	U	U	T	T
U	U	U	U	U	U	U	U	U
U	F	U	F	U	U	U	F	F
F	T	T	F	T	T	F	U	U
F	U	T	F	U	U	U	F	F
F	F	T	F	F	F	F	T	F

Appendix A

Tunguska specifications

Balanced ternary arithmetic, tryte width 6. Address width, word size 12. Standard notation balanced base-9 (nonary). Accessible address range DDD:DDD$_{b9}$ to 444:444$_{b9}$. Total accessible memory 531_{10} ktrytes. Big endian.

A.1 Registers

Tunguska has relatively few general purpose registers, but the speed of memory access is more or less identical to register access, so this is not that big of a deal. Directly user-accessible registers are indicated by bold.

Name	Purpose
A	Accumulator
X	Address register
Y	Address register
PC	Program counter
S	Stack index
CL	Clock register
P	Processor status

A.1.1 Processor status register specification

The processor status register (P) has the following tritfields, and it is set by most operations.

MST					**LST**
PR	V	B	I	G	C

The flags have the following purpose

Flag	Purpose
C	Carry
G	Comparison. Sign of the result of the last operation
I	No-Interrupt flag
B	Break in progress flag
V	Overflow flag
PR	Parity flag

Note: I and B flags may merge in the future to make better use of ternary capabilities.

A.2 Op-code specifications

The tunguska operation codes are following as following, where A is addressing mode and C is operation. In general, there are no forbidden combinations of addressing modes and operations, specifying an unsupported mode may be unpredictable, but will not crash the system.

MST					LST
A	A	C	C	C	C

A.2.1 Addressing modes

DEC	B9	Symbol	Instruction length	Description
-4	D	ABS	3	Value at memory position directly specified
-3	C	IMM	2	Value immediately following op-code
-2	B	AX	3	ABS with offset X
-1	A	AY	3	ABS with offset Y
0	0	ACC	1	Accumulator
0	0	IMP	1	Implicit – no argument
1	1	INDX	3	Value at (Memory position + X)
2	2	INDY	3	Value at (Memory position)+Y
3	3	INDIRECT	3	Value pointed to by the memory position directly specified
4	4	X:Y	1	Memory at page X, index Y

A.2.2 Operations

DEC	B9	Symbol	Valid addressing mods	Comment
-40	DD	CLV	IMP	Clear overflow
-39	DC	BRK	IMP	Trigger interrupt
-38	DB	RTI	IMP	Return from interrupt
-37	DA	LDA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set accumulator to memory value
-36	D0	LDX	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set Y register to memory value
-35	D1	LDY	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set X register to memory value
-34	D2	STA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Store accumulator's value in memory
-33	D3	STX	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Store X register's value in memory
-32	D4	STY	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Store Y register's value in memory
-31	CD	TAX	IMP	Transfer A to X
-30	CC	TAY	IMP	Transfer A to Y
-29	CB	TXA	IMP	Transfer X to A
-28	CA	TYA	IMP	Transfer Y to A
-27	C0	TSX	IMP	Transfer Stack index to X
-26	C1	TXS	IMP	Tarnsfer X to Stack index
-25	C2	PHA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Push to stack
-24	C3	PHP	IMP	Push processor status to stack
-23	C4	PLA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Pull from stack
-22	BD	PLP	IMP	Pull processor status from stack
-21	BC	AND	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=A$\wedge$memory
-20	BB	EOR	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=A$\oplus$memory
-19	BA	ORA	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=A$\vee$memory
-18	B0	BIT	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set status flag as though AND
-17	B1	ADD	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=A+memory
-16	B2	CMP	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set status flag as though A-memory
-15	B3	INC	ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Increase value by 1
-14	B4	INX	IMP	Increase X register by 1
-13	AD	INY	IMP	Increase Y register by 1
-12	AC	DEC	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Decrease value by 1
-11	AB	DEX	IMP	Decrease X register by 1
DEC	B9	Symbol	Valid addressing mods	Comment

DEC	B9	Symbol	Valid addressing mods	Comment
-10	AA	DEY	IMP	Decrease Y register by 1
-9	A0	ASL	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Shift value left, spillover in carry
-8	A1	LSR	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Shift value right, spillover in carry
-7	A2	ROL	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Rotate left
-6	A3	ROR	ACC,ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Rotate right
-5	A4	JMP	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump
-4	0D	JSR	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump to subroutine
-3	0C	RST	IMP	Return from subroutine
-2	0B	CLC	IMP	Clear carry
-1	0A	CLI	IMP	Clear interrupt flag
0	00	NOP	IMP	No operation
1	01	SEC	IMP	Set carry
2	02	SEI	IMP	Set interrupt flag
3	03	MLH	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=high tryte(A*memory)
4	04	MLL	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A=low tryte(A*memory)
5	1D	DIV	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A = integer($\frac{A}{memory}$)
6	1C	MOD	ACC, IMM, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A = remainder($\frac{A}{memory}$)
7	1B	PLX	IMP	Pull X from stack
8	1A	PLY	IMP	Pull Y from stack
9	10	PHX	IMP	Push X to stack
10	11	PHY	IMP	Push Y to stack
11	12	JCC	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if carry is clear
12	13	JCT	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if carry is positive
13	14	JCF	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if carry is negative
14	2D	JEQ	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag is clear
15	2C	JNE	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag isn't clear
16	2B	JLT	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag is negative
17	2A	JGT	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if comparison flag is positive
18	20	JVC	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if overflow is clear
19	21	JVS	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Jump if overflow is set
20	22	IVC	IMP	Invert carry flag
21	23	PRM	IMM, ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Permutate value (tritwise add 1)
22	24	TSH	IMM, ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	PRM without roll over
23	3D	BUT	IMM, ACC, ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	A BUT Value
24	3C	LAD	ABS, ABSX, ABSY, IND, INDX, INDY	Load ADdress into X:Y
25	3B	PGT	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Copy contents of page A to page mem
26	3A	PGS	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Set contents of page A to mem
27	30	CAD	ABS, ABSX, ABSY, IND, INDX, INDY	Compare X:Y with argument
28	31	XAM	ABS, ABSX, ABSY, IND, INDX, INDY, X:Y	Exchange A with memory value
29	32	XAX	IMP	Exchange A with X
30	33	XAY	IMP	Exchange A with Y
31	34	XYX	IMP	Exchange X with Y
32	4D	RESV		
33	4C	RESV		
34	4B	RESV		
35	4A	RESV		
36	40	RESV		
37	41	RESV		
38	42	RESV		
39	43	PAUSE	PAUSE	PAUSE
40	44	DEBUG	DEBUG	DEBUG
DEC	**B9**	**Symbol**	**Valid addressing mods**	**Comment**

A.3 Reserved addresses

The following memory addresses have special functionality, and should not be used for general purpose code or data storage.

Address range	Purpose
DDD:DDD/C	IRQ, IRQ data
DDD:DDB	Screen mode
DDD:DDA	Disk I/O
DDD:DD0	AGDP Operation
DDD:DD1/2	AGDP Register 1
DDD:DD3/4	AGDP Register 2
DDD:DCD/C	AGDP Register 3
DDC:DDD-DDC:444	Stack
DDB:DDD-DDA444	Screen buffer, text mode
DDB:DDD-DDB:444	Screen buffer, vector mode
DDB:DDD-CAB:444	Screen buffer, raster mode
444:441	Clock interrupt interval[1]
444:442/3	Interrupt handler vector

A.3.1 Screen

The Screen Mode tryte (DDD:DDB) has the following function

MST					**LST**
RES	RES	RES	Aux graphics mode	Graphics mode	Redraw

Screen redraws when Redraw is set to 1, and allowed graphics mode are raster(-1), text (0) and vector (1).

A.3.1.1 Vector mode

Vector mode only uses the first page of the screen buffer (that is, DDB:DDD/444), and stores vector information in sets of three trytes (allowing for a total of 121 interconnected lines in 729 colors). They are drawn in succession, and black lines are not painted (there won't be black spots where black lines intersect visible ones.)

	MST					**LST**
First tryte	MRed	LRed	MGreen	LGreen	MBlue	LBlue
Second tryte	UNUSED	X	Coo	rdi	na	te
Third tryte	UNUSED	Y	Coo	rdi	na	te

A.3.1.2 Raster mode

Raster mode resolution is 324x243[2] and uses a total of 108 memory pages, with two color depths:

- One pixel per tryte, allowing for a total of 729 colors. They are encoded like in the first tryte in vector mode above, addressed according to (1).

$$T(x, y) = DDBDDD_{9b} + 324y + x \tag{A.1}$$

- One trit per pixel, allowing for a total of three gray light intensities, memory addressed according to (2), color intensity specified by (3):

$$T(x, y) = DDBDDD_{9b} + 54y + \lfloor \frac{x}{6} \rfloor \tag{A.2}$$

$$R(x) = x \bmod 6 \tag{A.3}$$

A.3.1.3 Text mode

Text mode resolution is 27 rows by 54 columns (2 memory pages).

[1] Don't set it to zero or less, unpredictable behavior will ensue

[2] Observant people will notice that this is 400_{b9}x300_{b9}

A.4 Interrupts

IRQ	Function	IRQ data
0	Keypress interrupt	Key
1	Clock interrupt	Random value
2	Keyboard break sent	undef.
3	Arithmetic error	undef.
4	Soft interrupt	undef.
5	Mouse motion	relative X : relative Y
6	Mouse event	click = 1, release = -1

A.4.1 Disk I/O

Tunguska features a floppydisk-like virtual disk drive. It is controlled through the Disk I/O tryte (DDD:DDA). It is the same size as the main memory (729^2 trytes). All data transfer is on the block level, and block size is 729 trytes. The following operations are allowed:

Name	Number	Function
NOOP	0	Idle
READ	1	Read from disk to memory
WRITE	2	Write from memory to disk
SYNC	3	Write from virtual disk to physical file
SEEK	4	Set disk position to page
GETPOS	5	Get disk position

SEEK and GETPOS use the accumulator as input. Data is read to and from memory page specified by the Y register.

The disk image format is 729^2 consecutive 16 bit *short integer*s, each storing a single memory cell. To save space, the images are compressed with ***zlib***.

A.4.2 Auxiliary General Data Processor (AGDP)

The AGDP offers block operations and floating point arithmetics. It has three 12 trit registers, AGDP R1-R3; and one memory position that determines it's operation.

Operation	Number	Function	Arguments
NOOP	0	Nothing	-
ITOF	1	Convert Integer to Float	R1 -> R1
FTOI	2	Convert Float to Integer	R1 -> R1
FADD	3	Add two float numbers	R1, R2 -> R1
FMUL	4	Multiply two float numbers	R1, R2 -> R1
FDIV	5	Divide two float numbers	R1, R2 -> R1
FEXP	6	Exponent of R1	R1 -> R1
FLOG	7	Natural logarithm of R1	R1 -> R1
FCOS	8	Cosine of R1	R1 -> R1
FSIN	9	Sine of R1	R1 -> R1
BLT	10	Block transfer R1 to R2	Length R3
BLS	11	Block set R1 to val R2+1	R2+1->R1:(R1+R3)
BLA*	12	Blockwise AND R1 vs. R2	Length R3
BLX*	13	Blockwise XOR R1 vs. R2	Length R3
BLO*	14	Blockwise OR R1 vs. R2	Length R3
BSH*	15	Blockwise "TSH" R1 vs. R2	Length R3
BLP*	16	Blockwise "PRM" R1 vs. R2	Length R3

* Overlap undefined.

A.5 Floating point

Tunguska has support for the equivalent of half-precision floating point numbers, with base 3, a [-40,40] range exponent, and [-3280, 3280] range significand. Where the numbers are packed in two trits in the following fashion.

MST					**LST**	**MST**					**LST**
E	E	E	E	S	S	S	S	S	S	S	S

The mathematics are more or less identical to binary floating point, the only significant difference is that there is no need for bias or a sign bit (sign is determined by the significand S). The mathematics behind it is as follows.

$$F = S \cdot 3^E \tag{A.4}$$

$$S = F \cdot 3^{-E} \tag{A.5}$$

$$E = \lfloor \frac{\ln |F|}{\ln 3} \rfloor \tag{A.6}$$

A.6 Notes for 6502 programmers

There are some major differences between the 6502 and this machine. Beyond the obvious differences in endianness, and processor status flag, some instructions have changed. **PHA** is replaced by a general purpose **PSH** ("*PHA A*" will replicate the behavior of **PHA**), that will push any memory value onto the stack. **PLL** is it's respective pull operation. **ADC** has changed name to **ADD** with no change in functionality. **RTS** has changed name to **RST** with no change in functionality.

X:Y addressing is a new mode that uses both the X and Y register to address a memory location. The **LAD** operator is a quick way to transfer a complex memory location onto X:Y (compare with the x86 "LEA" operation). Tunguska has an address comparison operation **CAD** that compares the argument with X:Y, and sets flags accordingly.

Branching has been abandoned, use jumps instead. The same behavior can be acheived with the $$-token.

There is also a series of convenience operations **XAM**, **XAX**, **XAY**, **XYX** that serve to swap the contents of registers or memory in one atomic instruction.

A.7 Debugging

Tunguska operates some built-in debugging facilities, mainly the **DEBUG** instruction which prints the contents of all registers to stdout, and the **PAUSE** instruction which acts like a breakpoint and halts execution at a specific point in the code, to allow for instruction stepping with the F11 key (execution is resumed by pressing the Pause key.)

Appendix B

The GNU Free Documentation License

GNU Free Documentation License Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the

DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History"

section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version. N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.