MIX Manual

version 2.0, 30 October 2009

Sergey Poznyakoff.

Short Contents

Table of Contents

1 Introduction to MIX

Plain MIX is a set of tools for assembling, running and debugging programs, written in MIXAL, an assembly language for an imaginary computer, MIX, used in "The Art of Computer Programming" of D. Knuth.

This package provides a development platform for those who wish to try out examples and exercises from the book. It includes the following programs:

`mixal` A MIXAL assembler, i.e. a program which translates a MIXAL source file into a program that can be run on a MIX machine.

`mixsim` MIX machine simulator. There is no real, hardware, MIX machine, but you may use `mixsim` to emulate it and to run the programs, prepared by `mixal`. The `mixsim` utility also provides a *terminal mode* with a debugger, which is useful for finding and fixing bugs in your programs. It is also handy for educational purposes, as it allows to trace progam execution.

`mixrun` A utility to run MIXAL programs, without creating an intermediate object file.

This manual assumes the reader has some basic notions about MIX and MIXAL and that he has a copy of "The Art of Computer Programming" (referred to in this book as TAOCP) at hand. The page references to TAOCP assume the Addison-Wesley edition, 1968 (Library of congress catalog card no. 67-26020).

2 mixal – MIX Assembler.

A MIX assembler is called `mixal`. The utility assembles its standard input (or a named file), which should be a valid MIXAL program, and writes the resulting object code to the standard output or to another file.

This chapter describes how to use `mixal`. The examples in this chapter assume that the file '`hello.mix`' contains the following example program:

```
* ''HELLO, WORLD'' PROGRAM
PRINTER EQU  18
        ORIG 3000
HELLO   OUT  TEXT(PRINTER)
        JBUS *(PRINTER)
        HLT
TEXT    ALF  HELLO
        ALF  , WOR
        ALF  LD
        END  HELLO
```

2.1 Assembling MIXAL Programs.

The simplest way to assemble a MIXAL program is to give it as an argument to `mixal`:

```
$ mixal hello.mix
```

The `mixal` utility assembles the program, and prints the resulting object code on the standard output. The object code is formatted as a *card deck*, as described in *TAOCP, 1.3.1, p.141, ex. 26*, therefore in this book we use the terms *object file* and *deck file* as synonyms.

Each line in the deck file corresponds to a single punch card. First two cards are always the same — they contain a loader routine, responsible for loading of the entire deck into MIX memory and passing control to the program entry point. The following lines, up to the last one, contain the program code, formatted as described in the following table:

Column	**Meaning**
1–5	Ignored.
6	Number of consecutive words to be loaded on this card (between 1 and 7, inclusive).
7–10	The location of word 1 (always greater than 100).
11–20	Word 1.
21–30	Word 2.
31–40	Word 3.
41–50	Word 4.
51–60	Word 5.
61–70	Word 6.
71–80	Word 7.

For example, the card:

```
HELLO6300007872196210786695330000000013301355825440687973395021915238 4
```

contains 6 words to be loaded starting from address 3000. These words are:

Address	Word
3000	0787219621
3001	0786695330
3002	0000000133
3003	0135582544
3004	0687973395
3005	0219152384

The deck ends with a special *transfer card*, which contains information in format '`TRANS0`*nnnn*', where *nnnn* is the address of the program entry point. For example, '`TRANS03000`' means "start execution from address 3000".

To illustrate this, here is the deck file produced for '`hello.mix`' (the first two cards are omitted):

```
HELLO6300007872196210786695330000000013301355825440687973395021915238 4
TRANS03000
```

The card deck, produced by `mixal` can be executed by the MIX simulator, as described in Chapter 3 [mixsim], page 9. In the simplest case, you can directly feed the deck to the standard input of `mixsim`:

```
$ mixal hello.mix | mixsim
```

However, for more complex programs, it is common to store the produced card deck in a file for further use by `mixsim`. To do so, use '`--output`' ('`-o`') command line option, as shown in the example below:

```
$ mixal --output=hello.deck hello.mix
```

2.2 Program Listing

To obtain more details about the generated object deck, use '`--list`' ('`-l`') command line option. This option generates a *listing file*. The file name for this file is constructed by appending '`.lst`' suffix to the base name of the input file. For example, the following invocation will store the program listing in file '`hello.lst`'.

```
$ mixal -l hello.mix
```

If explicit input file is not given, e.g. when assembling the standard input, the listing is named '`mixal.lst`'.

These naming conventions can be overridden, by specifying the listing name explicitly, with '`--list-file`' option. This option implies '`--list`', so you need not give the two options together. For example, the following invocation will store the listing in file '`out/hello.list`':

```
$ mixal --list-file=out/hello.list hello.mix
```

The program listing contains, for each line of the input source, the address of the corresponding MIX cell and the assembled cell contents, as shown in the table below:

Column	**Meaning**
1–4	MIX cell address.
5	A semicolon
7–21	Cell contents.
23–27	Source line number.
28 and others.	Source line.

The cell contents (columns 7–21) is formatted as described in *TAOCP, 1.3.1, p.124, Instruction format*:

Column	**Meaning**
7	Sign.
8-12	Address part.
14–15	I-field.
17–18	F-field.
20–21	Opcode.

The following example shows `mixal` listing for the '`hello.mix`' program:

```
                          1 * ``HELLO, WORLD'' PROGRAM
                          2 PRINTER    EQU  18
                          3            ORIG 3000
3000: + 3003  0 18 37     4 HELLO      OUT  TEXT(PRINTER)
3001: + 3001  0 18 34     5            JBUS *(PRINTER)
3002: +    0  0  2  5     6            HLT
3003: +  517 13 13 16     7 TEXT       ALF  HELLO
3004: + 2624 26 16 19     8            ALF  , WOR
3005: +  836  0  0  0     9            ALF  LD
                         10            END  HELLO
```

After the listing comes a *symbol table*, which, for each symbol used in the program, shows its name, location and a source line, where it was defined. For example:

```
Symbol     Value  Line
PRINTER       18    2
HELLO       3000    4
TEXT        3003    7
```

The '`Value`' column contains a MIX location, corresponding to that symbol, except in case of macro-definitions (`EQU`), where the actual value of the macro is printed (see '`PRINTER`' in the example above).

The symbol table contains not only user-defined symbols, but also any literals and local labels used in the program. For literals, the '`Symbol`' column contains their computed *w-expressions*, surrounded by equals sings. For example, the line

```
        MUL  =2*25+1=
```

will produce the symbol name '`=51=`', e.g.:

```
Symbol     Value  Line
=51=        1101   18
```

Local labels are displayed as two integer numbers, separated by a dash and surrounded by vertical bars (‘|’). The first number corresponds to the number of the local label, the second one means its ordinal number in the program text. For example, the MIXAL fragment below:

```
1H      INCX 0
        DECA 11
        JANP 1B
1H      INCX 1
```

will produce the following two entries in the symbol table:

```
Symbol     Value  Line
|1-1|       1026    7
|1-2|       1029   10
```

An additional statistics about the input source can be obtained using ‘`--xref`’ (‘`-x`’) option, which instructs `mixal` to print a cross-reference table of all used symbols. A *cross-reference table* is added as an additional column to the symbol table, described above. The contents of this column lists the lines where the symbol was referenced. The following example shows a cross-reference table for the ‘`hello.mix`’ program:

```
Symbol     Value  Line  Referenced on line
PRINTER       18    2    4    5
HELLO       3000    4   10
TEXT        3003    7    4
```

If ‘`--xref`’ is used without ‘`--list`’ (or ‘`--list-file`’), the symbol table is printed on the standard error.

2.3 Raw Object Code.

Sometimes you may need to assemble a MIXAL program into a raw sequence of bytes, without composing a proper load deck. In particular, this becomes necessary if you wish to develop your own loading routine. The ‘`--raw-output`’ (‘`-r`’) allows you to do that. When called with this option, `mixal` outputs assembled code as is, without converting it to object card format and without prefixing it with loader routine. Currently this option assumes that the produced code will be read using device 16 (card reader), so the output byte stream is formatted in blocks of 16 words (80 bytes) delimited by newlines.

A particularly interesting implementation of this feature would be to produce a loader code for another type of input device, e.g. to load programs from magnetic tapes or disks. This, however, requires some further work on `mixsim` and will be implemented in future versions of the package.

2.4 `mixal` option summary.

Usage:
`mixal [`*`options`*`] [`*`file`*`]`

The following table summarizes the available command line options:

‘`--list`’
‘`-l`’ Produce a source listing. Unless ‘`--list-file`’ (see below) is used, the default listing file name is constructed by appending ‘`.lst`’ suffix to the base name of the input file. If standard input is used, the listing file is named ‘`mixal.lst`’.

See Section 2.2 [listing], page 4.

‘`--list-file=`*`file`*’
Set listing file name. Implies ‘`--list`’.

See [mixal-list-file], page 4.

‘`--force`’
‘`-f`’ Force generating object deck (and, eventually, listing) even if there were errors during assembly.

‘`--output=`*`file`*’
‘`-o`’ Set output file name. By default, object deck is printed on the standard output.

See [mixal-output], page 4.

‘`--raw-output`’
‘`-r`’ Produce raw object output.

See Section 2.3 [raw output], page 6.

‘`--xref`’
‘`--cross-reference`’
‘`-x`’ Output a cross reference.

See [mixal-xref], page 6.

‘`--debug-gram`’
Enable parser debugging output.

‘`--debug-lex`’
Enable lexical analyzer debugging output.

‘`-d` *`level`*’
Set debug level. This option is for compatibility with previous versions. The ‘`-dy`’ option is equivalent to ‘`--debug-gram`’, ‘`-dl`’ is equivalent to ‘`--debug-lex`’, ‘`-dyl`’ (or ‘`-dly`’) is equivalent to both.

‘`--help`’
‘`-h`’ Print a concise help summary.

‘`--version`’
‘`-V`’ Print program version and license information.

3 mixsim – MIX Simulator.

The MIX simulator, `mixsim`, allows you to execute object code assembled by `mixal`, and to inspect the machine state at any stage of the execution. The simulator also includes a debugging facility, useful when analyzing the examples and solving problems from TAOCP.

The simulator is written in compliance with the recommendations found in *TAOCP, 1.4.3.1, p.198, MIX Simulator.*

3.1 A MIX Machine Implementation.

The simulator implements the MIX machine as described in *TAOCP, 1.3.1, p.120, Description of MIX*. The machine is shipped with all external equipment, except the paper tape unit[1], and the floating point feature.

Each I/O device is bound to a particular UNIX file. By default the card reader is connected to stdin, the printer to stdout and the card punch to stderr. The typewriter is connected to stdin. The paper tape unit is not yet implemented.

The tape units use files '`tape0`' to '`tape7`' and the disk units use files '`disk0`' to '`disk8`' in the current directory. None of these files is required to exist: they will be created on demand, when the first output operation on the given unit takes place.

The disks are always opened for update without truncation, so that old data is not destroyed until it is overwritten. Note that big disk files will never shrink unless they are deleted.

In contrast, the tapes are always opened with truncation, so that any existing data is lost after the first `OUT` instruction is executed.

3.2 Executing MIX Programs.

When run without command line options, `mixsim` reads a load deck from device 16, loads and executes it, writing any error messages to stderr. On completion, a dump of the machine state is written to stderr. If `mixsim` is interrupted during execution, the machine state will be dumped to stderr.

At most one argument can be given. It is treated as a file name to be assigned to the card reader device. Thus, there are three ways to execute a load deck previously stored in a file:

a. Redirect the file's contents to the `mixsim` stdin:

```
$ mixsim < hello.deck
```

b. Give the file name as an argument to `mixsim`:

```
$ mixsim hello.deck
```

c. Assign the file to device '`#16`', using '`-a`' option (see below):

[1] It will be implemented in future releases

```
$ mixsim -a 16=hello.deck
```

The default device assignments can be changed using '`--assign-device`' ('`-a`') command line option. It takes a single argument in the form *dev*=*file*, where *dev* is the MIX device number and *file* is the name of file to be assigned to it. For example, the following invocation tells `mixsim` to connect the card puncher (device 17) to file '`punch.out`':

```
$ mixsim --assign 17=punch.out
```

3.3 Terminal Mode and Debugger.

When given the '`--terminal`' ('`-t`') option, `mixsim` starts in *terminal mode*. The terminal mode provides a custom shell for executing, tracing and debugging MIX programs. When you start `mixsim` in this mode, you will see:

```
$ mixsim -t

MIX TERMINAL STATION READY FOR COMMUNICATION

MIX> _
```

The '`MIX>`' string at the end of the screen is the *terminal prompt*, inviting you to enter a command. The syntax of terminal commands is similar to that of shell: each command consists of a *command verb*, optionally followed by one or more *arguments*, separated by any amount of whitespace. A command ends with a newline character.

3.3.1 Mixsim Commands

All `mixsim` command verbs (hereinafter referred to as *commands*) are case-insensitive. Each command has a full and contracted form, the latter one being designed to save extra typing.

If the package is configured with `readline` support enabled (which is the default, if `readline` is present on the system), you can abbreviate a command to the first few letters of the command name, if that abbreviation is unambiguous. A flexible command line editing facility is also available in this case. See Section 3.3.1.1 [completion], page 10, for a detailed desctiption.

You can test if an abbreviation is valid by using it as an argument to the `help` command (see Section 3.3.2 [help], page 11).

A blank line as input to GDB (typing just *RET*) means to repeat the previous command.

Any text from a '`#`' to the end of the line is a comment; it is ignored. This is useful mainly in command files (see Section 3.4 [command files], page 21).

3.3.1.1 Command Completion

This subsection discusses the command completion facility, available if `mix` is configured with `readline` support. Using this feature is recommended. If you don't have GNU `readline`, we suggest you to get it from the Readline Home Page prior to compiling `mix`.

See Section "Command Line Editing" in *GNU Readline Library*, for more information about the library.

`Mixsim` can fill in the rest of a word in a command for you, if there is only one possibility; it can also show you what the valid possibilities are for the next word in a command, at any time.

Press the *TAB* key whenever you want `mixsim` to fill out the rest of a word for you. If there is only one possibility, it will replace the partial word and will wait for you to finish the command. This *command completion* is context-dependent. For example, if you type:

```
MIX> br_TAB
```

this command will immediately be expanded to `break`, since it is the only possibility in this context:

```
MIX> break
```

If there is more than one possibility to complete the word you are typing, `mixsim` rings a bell. You can either supply more characters and try again, or just press *TAB* one more time, in which case `mixsim` will display all the possible completions for that word. For example, the `address` command (see Section 3.3.6 [breakpoints], page 13) can be followed by several keywords, so, if you type *TAB* after `address`, `mixsim` rings a bell and waits for your further actions. If you press *TAB* second time, you'll see:

```
MIX> ADDRESS _TAB
A bell sounds. Pess TAB again to see:
CB          DELETE      ENABLE      INFO        LIST
CLEAR       DISABLE     IGNORE      LB          PASSCOUNT
MIX> ADDRESS _
```

To save you typing, it is also possible to view the list of alternatives in the first place. To do so, type *M-?*, instead of pressing *TAB* twice. *M-?* means either to hold down a key designated as the *META* shift on your keyboard (if there is one) while typing *?*, or to press *ESC*, followed by *?*.

If the context requires you to type a file name, `mixsim` will complete file names. For example:

```
MIX> ASGN 16 ../examples/_M-?
mystery.mix    hello.mix      init.mix       p.mix
easter.mix     load.mix       tsort.mix
```

Similarly, if the context requires a name of an executable file, you will see the appropriate completion, made using your `PATH` environment variable setting:

```
MIX> ! ca_TAB TAB
cat             cal
```

3.3.2 Obtaining On-line Help

`HELP`
`?` To obtain help about all available commands, type `HELP`.

`HELP` *cmd*
`?` *cmd*
: When used with an argument, it displays usage summary for the given command verb, e.g.:

```
MIX> HELP ASGN
ASGN <DEVICE> <FILE>
```

3.3.3 Quitting the Terminal

`QUIT`
`Q`
: To exit the terminal, use the `QUIT` (abbreviated `Q`), or type an end-of-file character (usually *C-d*).

An interrupt (*C-c*) does not exit from `mixsim`, but rather stops the program being executed (see Section 3.3.7 [stopping], page 16), or, if no program is being run, cancels the current input line.

3.3.4 Assigning and Listing Devices

`ASGN` *device file*
`A` *device file*
: Assign *file* to the given MIX device. *device* specifies the device number. In the effect of this command, output operations on *device* will write the data to *file*, and input operations on *device* will read data from *file*.

For example, to assign file '`hello.deck`' to the card reader, type:

```
MIX> ASGN 16 hello.deck
```

To obtain information about MIX devices, including their current assignments, use the following command:

`INFO IO [`*device*`]`
`LIST IO [`*device*`]`
`LI [`*device*`]`
: Without arguments, lists all devices. With a numeric argument, shows information about that particular device. For more uses of `INFO` command, see Section 3.3.6.5 [list breakpoints], page 15.

The information is displayed in tabular form and contains the following columns:

UNIT
: Unit number.

IOT
: I/O time per block.

SKT
: Seek time.

ADDR
: Memory address for the pending I/O operation.

POS
: Device position for the pending I/O operation. This column is meaningful only for tape and disk devices.

OP
: Opcode or '`N/A`', if no operation is pending.

CLOCK Clock time when the I/O will occur. To examine the current clock time, see Section 3.3.6.5 [list breakpoints], page 15.

ASGN File assigned to that device.

The example below shows the information about card reader, obtained after the first loader card was read:

```
MIX> info io 16
UNIT    IOT    SKT  ADDR  POS   OP  CLOCK ASGN
  16  10000      0    16    0   IN  15000 easter.obj
```

3.3.5 Running a Program

`GO`
`G`
`RUN`
`R` Emulates MIX GO button.

This command does the following:

a. A single card is read into locations '`0000--0015`';
b. When the card has been completely read and the card reader is no longer busy, a `JMP` to location '`0000`' occurs. The J-register is also set to 0.

3.3.6 Breakpoints

A *breakpoint* makes your program stop whenever a certain location in the program is reached. When the program stops at a breakpoint we say that this breakpoint has been *passed* or *hit*.

For each breakpoint, two numbers are defined: *ignore count*, which specifies the number of passes before that breakpoint is enabled, and *pass count*, which specifies the number of passes after which the breakpoint will be deleted.

Each created breakpoint is assigned a *sequence number*, which can be used to refer to that breakpoint in other commands. Another way to refer to a breakpoint is by the program location it is set at. Thus, each command described in this subsection has two forms: a form which uses breakpoint numbers, and a form that uses program locations. The second form is constructed by prefixing the command with `ADDRESS` keyword (abbreviated `AD`). For example, the following command deletes breakpoint number 2 (see Section 3.3.6.2 [delete breakpoints], page 14):

```
MIX> DELETE 2
```

In contrast, the following command deletes all breakpoints set at address 1000:

```
MIX> ADDRESS DELETE 1000
```

3.3.6.1 Setting Breakpoints

Breakpoints are set with the `BREAK` command (abbreviated `B`).

`BREAK` *location*
`B` *location*
: Set a breakpoint at the given *location*. Both ignore and pass counts are set to 0. A sequence number assigned to the breakpoint is displayed, as shown in the example below:

```
MIX> BR 1000
BREAKPOINT 1 IS SET AT ADDRESS 1000
```

This sequence number can then be used to refer to this breakpoint.

`BREAK TEMP` *location*
`BT` *location*
`TB` *location*
: Sets a temporary breakpoint at the given *location*. A *temporary breakpoint* is a breakpoint which is deleted after a single hit. In other words, it has pass count set to 1.

This command is equivalent to:

```
BREAK location
ADDRESS PASSCOUNT location 1
```

See Section 3.3.6.4 [configure breakpoints], page 15, for information about `PASSCOUNT` command.

You can set any number of breakpoints at the same place in your program. This feature is not very useful at the moment, it is reserved for future use.

3.3.6.2 Deleting Breakpoints

It is often necessary to eliminate a breakpoint which has done its job and is no longer needed. This is called *deleting* the breakpoint. A deleted breakpoint no longer exists, and its sequence number is returned to the pool of available numbers.

It is not necessary to delete a breakpoint to proceed past it. `Mixsim` automatically ignores breakpoints on the first instruction to be executed when you continue execution without changing the execution address.

`DELETE [`*number-list*`]`
`D [`*number-list*`]`
: Delete specified breakpoints. *number-list* is a whitespace-separated list of breakpoint numbers. If no argument is specified, delete all breakpoints (in this case `mixsim` asks confirmation).

When prefixed with `ADDRESS` (abbreviated `AD`), *number-list* is treated as a list of addresses, instead of breakpoint numbers.

Examples:

1. Delete all breakpoints:

```
MIX> DELETE
```

2. Delete breakpoints 1, 3 and 5:

```
MIX> DELETE 1 3 5
```

3. Delete breakpoints set at addresses 1000 and 3000:

```
MIX> ADDRESS DELETE 1000 3000
```

3.3.6.3 Disabling and Enabling Breakpoints

Instead of deleting a breakpoint, you might prefer to *disable* it. A disabled breakpoint continues to exist, but is ignored. You may enable it again later, if the need arises.

To disable and enable breakpoints, use `ENABLE` and `DISABLE` commands, described below. To obtain information about existing breakpoints and their status, use `INFO BREAK` command (see Section 3.3.6.5 [list breakpoints], page 15).

`DISABLE [`*`number-list`*`]`
`DIS [`*`number-list`*`]`
: Disable breakpoints, specified by *number-list*. If no arguments are given, disables all breakpoints.

`ENABLE` *`number-list`*
`ENA` *`number-list`*
: Enable breakpoints, specified by *number-list*. If no arguments are given, enables all breakpoints.

Both commands may be prefixed with `ADDRESS` (abbreviated `AD`), to specify breakpoints by MIX addresses, rather than by breakpoint numbers.

3.3.6.4 Configure Breakpoints

Each breakpoint has two numbers associated with it: *ignore count*, which specifies the number of passes before that breakpoint is enabled, and *pass count*, which specifies the number of passes after which the breakpoint will be deleted.

`IGNORE` *`number count`*
`I` *`number count`*
: Set ignore count for breakpoint *number* to *count*.

`PASSCOUNT` *`number count`*
`P` *`number count`*
: Set pass count for breakpoint *number* to *count*.

Both commands may be prefixed with `ADDRESS` (abbreviated `AD`), in which case their first argument, *n*, is treated as a MIX location where the breakpoint is set, rather than its number. For example, the following command sets pass count to 1 for all breakpoints set at location 1000:

```
MIX> ADDRESS PASSCOUNT 1000 1
```

3.3.6.5 Listing Breakpoints

`INFO BREAK [num]`
`LIST BREAK [num]`
`LB [num]` Without argument, lists all existing breakpoints. With an argument, lists only breakpoint *num*. When prefixed with `ADDRESS` (abbreviated `AD`), treats its argument as a MIX location and lists breakpoints set at that location.

Following is an example of a breakpoint listing:

```
NUM  LOC  ENB  CNT  IGN  PAS
  1 1000    Y    1    0    0
  2 1040    N   10    8    0
  3 1230    Y    0    0    1
```

The columns and their meanings are:

Column	Meaning
NUM	Breakpoint number.
LOC	Location.
ENB	Is the breakpoint enabled or not.
CNT	Number of passes registered this far.
IGN	Ignore count.
PAS	Pass count.

3.3.7 Stopping and Continuing

The principal purposes of using a debugger are so that you can stop your program before it terminates; or so that, if your program runs into trouble, you can investigate and find out why.

Inside `mixsim` a program may stop either because it hit an active breakpoint or because it reached a new line after `NEXT` or `STEP` command (see below). Additionally, you can stop a running program at any moment, by pressing *interrupt* (usually `C-c`).

`CONTINUE`
`C` Continue program execution, at the address where it last stopped.

A typical debugging technique is to set a breakpoint (see Section 3.3.6.1 [set breakpoints], page 13) at the location where a problem is believed to lie, run your program until it stops at that breakpoint, and then step through the suspect area, examining the variables that are interesting, until you see the problem happen.

The following two commands are useful with this technique.

`NEXT [count]`
`N [count]`
Execute next instruction and stop again. With an argument, execute next *count* instructions. If any of the executed instructions is a function call, that function is not descended into.

`STEP [`*`count`*`]`
`S [`*`count`*`]`
: Execute next instruction and stop again. If the instruction is a function call, descend into that function. With an argument, execute next *count* instructions.

3.3.8 Executing Shell Commands.

`SHELL [`*`command`*`]`
`! [`*`command`*`]`
: Execute given shell command, by running `/bin/sh -c` *`command`*. For example, to see the listing of the current working directory, do:

```
MIX> ! ls -l
```

If no arguments are supplied, execute a subordinate shell.

3.3.9 Examining Data and Registers

`DUMP`
`DU`
: Dump MIX registers and memory contents. The output format is described below.

`DUMP MEMORY [`*`from`*` [`*`to`*`]]`
`DM [`*`from`*` [`*`to`*`]]`
: Dump MIX memory. When used without arguments, dumps entire address space. When used with one argument, dumps the memory contents starting from location *from*. When used with two arguments, dumps the contents of memory between the locations *from* and *to*, inclusive. Both locations are rounded to the nearest word boundary.

The output is formatted in blocks, each one containing 5 machine words. Each block is preceded by the location of its first word. Each word is printed in three forms, located in rows. The first row is the decimal value of the word, the second row is its representation in instruction format (*TAOCP, 1.3.1, p.124, Instruction format*), and the last one gives its printable representation. For example, the words '`3000--3004`' of '`hello.mix`' code look as follows:

```
3000 +0000787219621 +0000786695330 +0000000000133
     +3003 00 18 37 +3001 00 18 34 +0000 00 02 05
     '*d Q7'        '*b Q4'        '   BE'

     +0000135582544 +0000687973395
     +0517 13 13 16 +2624 26 16 19
     'HELLO'        ', WOR'
```

The above example is split in two groups due to printing restrictions.

If several blocks of memory contain the same data, only first of them is displayed, the rest being replaced by a message, similar to the following:

```
Lines 3015 to 3995 are the same.
```

`DUMP REGISTERS`
`DR` Dump contents of MIX registers. The following example illustrates the output format:

```
Registers A/X    +00000000000 +00041363636
                 +          0 +    8775582

Index Registers  +00000  +00000  +05670  +00000  +00000  +00000
                 +    0  +    0  + 3000  +    0  +    0  +    0

Jump Register    +00015     Overflow toggle:      OFF
                 +   13     Comparison Indicator: EQUAL

Clock = 262436 u. Location = 3001, M 3003, I 0, F 18, C 37,
inst = + 5673002245
```

`DISASSEMBLE [`*from* `[`*to*`]]`
`UNASM [`*from* `[`*to*`]]`
`U [`*from* `[`*to*`]]`
Dump a range of memory *from–to* as MIX instructions. If *to* is not given, disassemble first five words starting at *from*. If no arguments are given, disassemble first five words starting from the current instruction pointer:

```
MIX> disassemble 0
0       IN    16(16)
1       IN    29(16)
2       LD1   0(0:0)
3       JBUS  *(16)
4       LDA   30
```

3.3.10 A Summary of Terminal Commands.

For convenience, this section lists all available terminal commands in alphabetical order, along with a short description and a reference to their detailed description.

`ADDRESS` *`breakpoint-command`* [Terminal Command]
breakpoint-command is any of the following commands with appropriate arguments: `DELETE`, `ENABLE`, `INFO BREAK`, `DISABLE`, `IGNORE`, `PASSCOUNT`.
The `ADDRESS` prefix makes *breakpoint-command* to refer to breakpoints using memory locations they are set as, rather than breakpoint numbers.
See Section 3.3.6 [breakpoints], page 13.

`ASGN` *`device file`* [Terminal Command]
Assign *file* to the given MIX device.
See Section 3.3.4 [devices], page 12.

`BREAK` [*TEMP*] *`location`* [Terminal Command]
Set a breakpoint at the given *location*. If `TEMP` is given, set a temporary breakpoint, active for one pass only.

See Section 3.3.6 [breakpoints], page 13.

`BT` *`location-list`* [Terminal Command]
`TB` *`location-list`* [Terminal Command]
Shortcut for `BREAK TEMP`.

`DELETE` [*`num-list`*] [Terminal Command]
Delete specified breakpoints. *num-list* is a list of breakpoint numbers or, if `ADDRESS` prefix is used, their addresses. Without arguments, delete all breakpoints.

See Section 3.3.6.2 [delete breakpoints], page 14.

`CONTINUE` [Terminal Command]
`C` [Terminal Command]
Continue program execution, at the address where it last stopped.

See Section 3.3.7 [stopping], page 16.

`DISABLE` [*`num-list`*] [Terminal Command]
Disable breakpoints. *num-list* is a list of breakpoint numbers or, if `ADDRESS` prefix is used, their addresses. Without arguments, disable all breakpoints.

See Section 3.3.6.3 [disable breakpoints], page 15.

`DISASSEMBLE` [*`from`* [*`to`*]] [Terminal Command]
`UNASM` [*`from`* [*`to`*]] [Terminal Command]
`U` [*`from`* [*`to`*]] [Terminal Command]
Dump a range of memory as MIX instructions.

See Section 3.3.9 [data], page 17.

`DUMP` [Terminal Command]
`DU` [Terminal Command]
Dump MIX registers and memory contents.

See Section 3.3.9 [data], page 17.

`DUMP` *REGISTERS* [Terminal Command]
`DR` [Terminal Command]
Dump contents of MIX registers.

See Section 3.3.9 [data], page 17.

`DUMP` *MEMORY* [*`from`* [*`to`*]] [Terminal Command]
`DM` [*`from`* [*`to`*]] [Terminal Command]
Dump MIX memory.

See Section 3.3.9 [data], page 17.

`ENABLE` [*num-list*] [Terminal Command]
Enable breakpoints. *num-list* is a list of breakpoint numbers or, if `ADDRESS` prefix is used, their addresses. Without arguments, enable all breakpoints.

See Section 3.3.6.3 [disable breakpoints], page 15.

`GO` [Terminal Command]
`RUN` [Terminal Command]
Run a program. See Section 3.3.5 [running], page 13.

`HELP` [*command-verb*] [Terminal Command]
`?` [*command-verb*] [Terminal Command]
Display a short usage summary about *command-verb*. Without arguments, display all available commands.

See Section 3.3.2 [help], page 11.

`IGNORE` *number count* [Terminal Command]
Set ignore count for breakpoint *number* to *count*. *number* is a breakpoint number or, if `ADDRESS` prefix is used, its address.

See Section 3.3.6.4 [configure breakpoints], page 15.

`INFO` *BREAK* [*num*] [Terminal Command]
`LIST` *BREAK* [*num*] [Terminal Command]
`LB` [*num*] [Terminal Command]
Without argument, lists all existing breakpoints. With an argument, lists only breakpoint *num*. May be prefixed with `ADDRESS` to use breakpoint address instead of number.

See Section 3.3.6.5 [list breakpoints], page 15.

`INFO` *IO* [*num*] [Terminal Command]
`LIST` *IO* [*num*] [Terminal Command]
`LI` [*num*] [Terminal Command]
Without arguments, list all devices. With a numeric argument, show information about that particular device.

See Section 3.3.4 [devices], page 12.

`NEXT` [*count*] [Terminal Command]
Execute next *count* (default is 1) instructions and stop again.

See Section 3.3.7 [stopping], page 16.

`PASSCOUNT` *number count* [Terminal Command]
Set pass count for breakpoint *number* to *count*.
number is a breakpoint number or, if `ADDRESS` prefix is used, its address.

See Section 3.3.6.4 [configure breakpoints], page 15.

`SOURCE` *filename* [Terminal Command]
`SO` *filename* [Terminal Command]
Execute the command file *filename*.

See Section 3.4 [command files], page 21, for more information about command files and their execution.

`QUIT` [Terminal Command]
Quit the terminal. See Section 3.3.3 [quitting], page 12.

`SHELL` [*command*] [Terminal Command]
`?` [*command*] [Terminal Command]
Execute given shell command. See Section 3.3.8 [shell commands], page 17.

`STEP` [*count*] [Terminal Command]
Execute next *count* (default is 1) instructions and stop again. If a function call is encountered, descend into the function.

See Section 3.3.7 [stopping], page 16.

3.4 Command Files

A *command file* is a file containing `mixsim` commands, one per line. Comments (lines starting with '`#`') and empty lines are also allowed. An empty line in a command file does nothing; it does not mean to repeat the last command, as it would from the terminal.

Command files are useful to store sequences of `mixsim` commands for executing them later. There are two ways to execute a command file: explicit, by using `SOURCE` command, or implicit, by naming the file '`.mixsim`' and storing it in the current working directory.

`SOURCE` *filename*
`SO` *filename*
Execute the command file *filename*.

The lines in a command file are executed sequentially. They are not printed as they are executed. An error in any command terminates execution of the command file and control is returned to the console. However, any `mixsim` command that prints a diagnostic message saying what it is doing, continues to do so even when called from a command file.

Commands that would ask for confirmation if used interactively proceed without asking, as if an affirmative answer was obtained.

When started in terminal mode (see Section 3.3 [terminal], page 10), `mixsim` searches for file named '`.mixsim`' in the current working directory, and, if this file exists, executes it. This file can be used to provide necessary default settings. For example, the following '`.mixsim`' file assigns '`easter.dck`' to the card reader device and sets breakpoint at address '`1000`':

```
asgn 16 easter.dck
br 1000
```

3.5 `mixsim` option summary.

This section summarizes `mixsim` command line options.

Usage:
`mixsim [options] [deck-file]`

‘`--assign-device=dev=file`’
‘`-a dev=file`’
: Assign *file* to the MIX device *dev*.
See Section 3.2 [exec], page 9.

‘`--terminal`’
‘`-t`’
: Run in terminal mode.
See Section 3.3 [terminal], page 10.

‘`--help`’
‘`-h`’
: Print a concise help summary.

‘`--version`’
‘`-V`’
: Print program version and license information.

4 mixrun

The MIX package provides a utility for assembling and executing a MIXAL file in one run. The utility is called `mixrun`. In its simplest form, it is called with the name of MIXAL source file as an argument, e.g.:

```
$ mixrun hello.mix
HELLO, WORLD

Registers A/X    +00000000000 +00041363636
                 +           0 +    8775582

Index Registers  +00000  +00000  +05670  +00000  +00000  +00000
                 +    0  +    0  + 3000  +    0  +    0  +    0
...
```

By default, a dump of the machine state is produced at the standard error. To direct it to another file, use '`--dump`' ('`-d`') command line option, e.g.: `mixrun --dump=hello.dump hello.mix`.

To suppress the dump, use '`--no-dump`' command line option.

You can also request producing a listing file. To do so, use '`--list`' ('`-l`') option. By default, the name of the listing file is constructed by appending '`.lst`' suffix to the base name of the input file. To use another file, give its name as an argument to '`--list`', as in the example below:

```
$ mixrun --list=my.list 'input'
# or:
$ mixrun -lmy.list 'input'
```

Notice, that an argument to '`--list`' option must be separated from it by an equals sign, with no whitespace on any side of it. Similarly, when a short form, '`-l`', is used, its argument must follow the option immediately.

As any other MIX program, `mixrun` understands two informational options. The option '`--version`' ('`-V`') displays the program version and a short licensing information, and the option '`--help`' ('`-h`') shows a short usage summary.

5 How to Report a Bug

Email bug reports to gray@gnu.org.ua.

As the purpose of bug reporting is to improve software, please be sure to include maximum information when reporting a bug. The information needed is:

- Version of the package you are using.
- Compilation options used when configuring the package.
- Conditions under which the bug appears.

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

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.

The "publisher" means any person or entity that distributes copies of the Document to the public.

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 under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

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. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

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 1.3 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.

Concept Index

This is a general index of all issues discussed in this manual.

I

L

M

N

O

P

Q

R

S

T

U

V

X