

# **MixEmul Documentation**

**Version 0.1.5**

**© Copyright 2008, Rutger van Bergen**

## Table of Contents

1	What, why and how .....	2
1.1	What is MixEmul?.....	2
1.2	But why write a MIX emulator (and in .NET even)? .....	2
1.3	But why not write MMixEmul?.....	3
1.4	What do I need to run MixEmul? .....	3
1.5	Will it actually work? .....	3
1.6	What about the legal mumbo-jumbo?.....	3
1.7	Can I have the source code? .....	4
1.8	Who are you anyway? .....	4
2	Running and using MixEmul .....	5
2.1	Starting MixEmul .....	5
2.2	The main window.....	5
2.2.1	Overview.....	5
2.2.2	Control strip.....	6
2.2.3	Registers section .....	7
2.2.4	Symbols section .....	8
2.2.5	Memory section.....	9
2.2.6	Devices section .....	10
2.2.7	Messages section.....	11
2.3	Using the assembler.....	12
2.3.1	Overview.....	12
2.3.2	MIXAL input format.....	12
2.3.3	Invoking the assembler .....	12
2.3.4	Assembly result window .....	13
2.4	Using devices .....	14
2.4.1	Device implementation.....	14
2.4.2	The device editor .....	14
2.5	Teletype window .....	18
2.6	Preferences window .....	19

# 1 What, why and how

## 1.1 What is MixEmul?

MixEmul is an emulator for the MIX computer that is described in The Art of Computer Programming (TAOCP) series of books from D.E. Knuth. MIX is a mythical, non-existent computer with features similar to those of real computers of the 1970s. It - or more accurately, its assembly language MIXAL - is used as the foundation for the text of aforementioned books.

MixEmul completely emulates the entire MIX instruction set. With that I mean that MixEmul not only performs the actions that an instruction should - like multiplying when it encounters a MUL instruction -, but it does it in the way MIX would. For example, the time unit ("tick") counts that are included in TAOCP Volume 1 are implemented as well. Also, as TAOCP specifies, I/O operations are performed in the background.

Speaking of I/O, all 21 MIX I/O devices (tapes, disks, card reader, card punch, printer, teletype and papertape) are implemented in MixEmul.

MixEmul lets you edit the contents of MIX's registers and memory before, during and after program execution using a number of editor types and review device status in a single glance. It includes a breakpoint feature and allows programs to be run in the background. It's even mildly configurable.

Oh, I'd almost forget, it incorporates a MIXAL assembler too. This means that you can write MIXAL programs using any text editor you like and then load them into MixEmul to debug and run them.

In a few words: MixEmul contains all features that I think I need to be able to better absorb the contents of the TAOCP books.

## 1.2 But why write a MIX emulator (and in .NET even)?

Well, I actually wrote MixEmul for a number of reasons, being:

1. I believed a good MIX emulator would allow me to get more joy and knowledge from reading TAOCP;
2. I didn't think there was a sufficiently usable MIX emulator available for the Windows platform until I completed MixEmul;
3. I had a long vacation and not much to do;
4. It was a lot of fun to write.

So, the fact you might be able to use it wasn't actually one of the reasons. However, if you do I won't hold it against you either.

I wrote it in .NET because I like the characteristics of the .NET platform and the C# language. I used to program in Java which is another great platform *and* language. However, I think the creators of C# managed to take some of the quirks out of Java and include new bits and pieces that make using it just that much more pleasant.

### 1.3 But why not write MMixEmul?

There is a number of reasons for that as well:

1. The prints of the books I recently purchased are still based on MIX, and only mention MMIX (a more modern replacement for MIX) as a future development;
2. MMIX is a lot more powerful and complex than MIX and therefore its emulator would need to be also. Writing MMixEmul would therefore be a major task that might start to feel like work. Not that I mind programming for work, but I already have a job;
3. My vacation wasn't *that* long.

Having said that, I did attempt to write the code for MixEmul in such a way that it might be used as the foundation for an MMIX emulator. Of course, I do realize that I have probably made all the wrong decisions while designing MixEmul's structure, which means that that will prove impossible if and when I actually try to do it. But hey, I've tried, haven't I?

### 1.4 What do I need to run MixEmul?

This:

1. A computer running Windows;
2. The Microsoft .NET Framework version 2.0 (or higher, I guess);
3. A directory to unpack the contents of the ZIP archive you found this document in;
4. Enough disk space for the device files of those devices you choose to use. The size of the device files for all devices except the disks depend on yourself and/or your programs. The device files for the disks are 2,457,600 bytes each.

Regarding requirements 1 and 2 I should add that it might be possible to run MixEmul on other platforms than Windows using Mono. However, I haven't tried that and am not planning to either. If you do try it please let me know what your findings are.

### 1.5 Will it actually work?

To be honest, I don't know, because I haven't yet used all instructions in the MIX instruction set. Nevertheless I decided it was time to (have other people) start using MixEmul. Of course, I did do *some* tests and I have been able to run a number of MIXAL programs on it. Amongst those are the Primes (TAOCP section 1.3.2), Easter (TAOCP 1.3.2 exercise 14) and Permutations (TAOCP section 1.3.3) programs. These programs have been included in the ZIP file, by the way.

I've also successfully performed some I/O operations on all MIX devices.

In short, everything I've used myself so far seems to work.

If you do find a bug and would like me to fix it please let me know and I'll attempt to set it straight.

### 1.6 What about the legal mumbo-jumbo?

There isn't any.

MixEmul is completely free and you may use it in any way you like. If you manage to build a 6 billion dollar corporation on top of it without sending me as much as a thank-you e-mail then that's fine by me. (However, in that case I would be more than happy to add some improvements to the program for just 10% of your company's shares.)

For the price you pay for MixEmul you get an appropriate amount of warranty: none. In other words, MixEmul is provided as is and if you do choose to use it then you do so at your own risk. If it doesn't do anything useful, erases your hard disk or eats your dog you're on your own. However, in the first case you could drop me a note in which case I'll see what I can do.

## **1.7 Can I have the source code?**

Maybe. You will have to tell me what you want to use it for and then I'll decide if I think that's a good enough reason. Improving or extending MixEmul would be a good reason, but then I'd first like to agree with you on how you'll be sending the improvements that you've made back to me. I'm just a bit of a quid pro quo guy with this sort of thing.

Also, please be aware that the source code is almost completely liberated of any documentation. Bad programming practice maybe, but I did this largely for the fun of it.

## **1.8 Who are you anyway?**

I'm Rutger van Bergen, a 30 year old computer science undergraduate born and living in The Netherlands.

I have an e-mail address that you can reach me on, it's [rbergen@xs4all.nl](mailto:rbergen@xs4all.nl). Because I also have a job and a life I can't guarantee any response times but I do guarantee that any requests to do your school assignments for you will be happily ignored. So don't even think about sending them. Don't do it. Really.

If you contact me to report a bug, please check that you are using the latest version that is available. The full version number is shown in the About window, which can be opened from the Help menu. The current version of MixEmul is available from the MixEmul web page, which is located at <http://www.xs4all.nl/~rbergen/mixemul.html>.

## 2 Running and using MixEmul

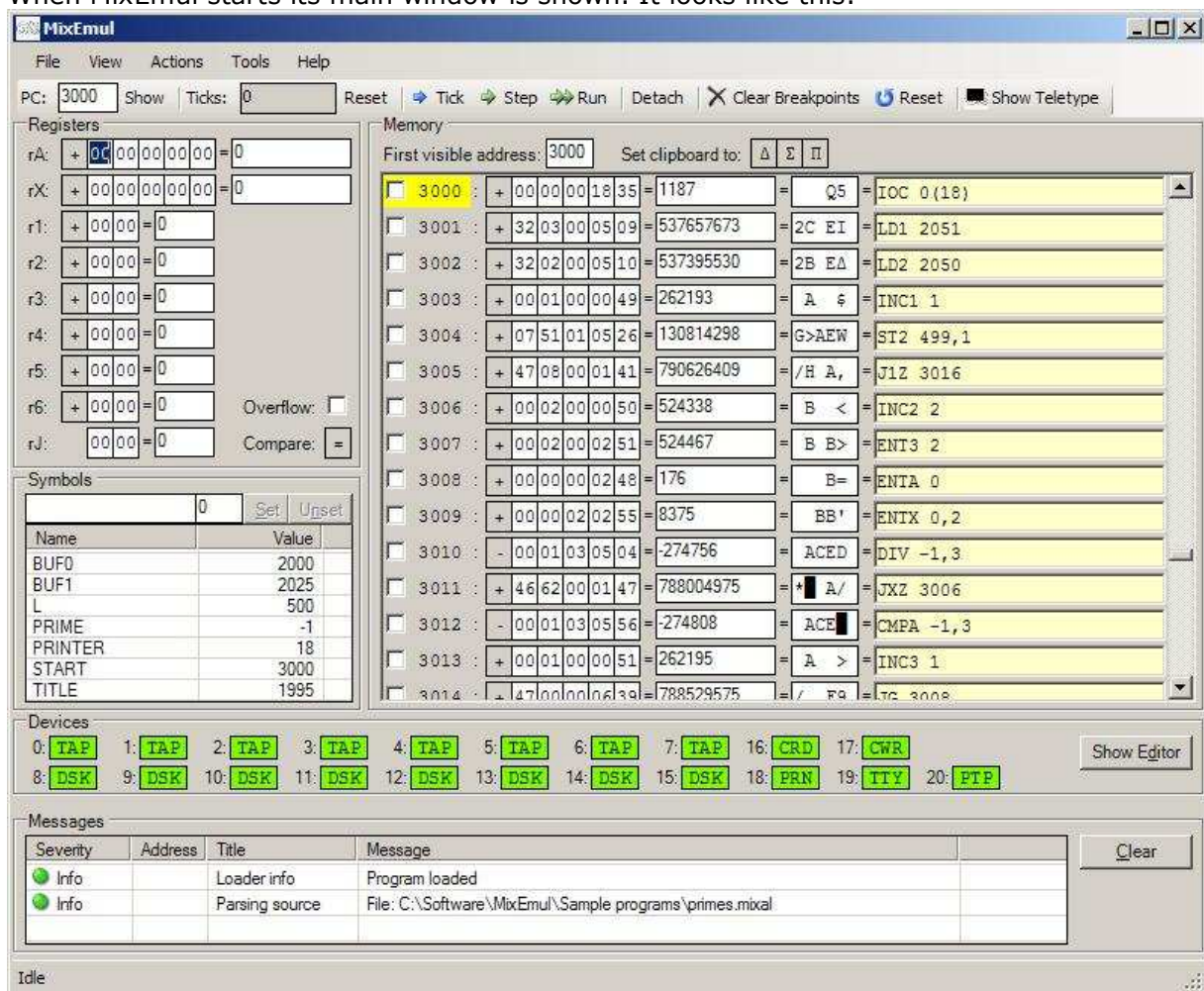
### 2.1 Starting MixEmul

MixEmul can be started by double-clicking on MixEmul.exe, after unpacking the ZIP file you got this document from.

### 2.2 The main window

#### 2.2.1 Overview

When MixEmul starts its main window is shown. It looks like this:



The main window consists of a number of sections which are discussed in the following text.

## 2.2.2 Control strip

The control strip is located at the top of the main window. It looks as follows:



The control strip contains the following items:

- An editable field (PC) that shows the current value of the program counter, i.e. the memory address from which the instruction to execute (the current instruction) will be loaded if the Tick, Step or Run buttons are pressed. You can change the program counter value by editing the field's contents and pressing Enter. The program counter memory address can be made visible in the memory section (see below) by clicking the Show button next to it.
- A read-only field that shows the current value of the tick counter, i.e. the number of time units that have passed since program startup or the last system or tick counter reset. The tick counter can be reset to 0 by clicking the Reset button right next to it.
- A Tick button. When this is pressed MixEmul runs for precisely one time unit. Depending on the current instruction and the status of Mix this may have the same effect as clicking the Step button, or have no visible effect at all.
- A Step button. Pressing this will run MixEmul until the current step is finished. A step is finished when the program counter changes from the value it has at the time of pressing Step to another value. Note that the current instruction might execute multiple times within a single step. An example of this is when the current instruction is JBUS \*(18) and the printer is busy.
- A Run button. Pressing the Run button will run MixEmul until one of the following conditions occurs:
  - The HLT instruction is executed. In this case MixEmul, or rather the tick counter, continues running until all busy devices are ready.
  - Teletype input is required but not available (see the discussion on the teletype window, below).
  - A breakpoint is reached.
  - A runtime error occurs.
  - The Stop button is pressed.

While MixEmul is running after clicking Step or Run, the Run button changes into a Stop button. Clicking this will stop execution after completing the then current tick.

- A Detach/Attach button. This button can be used to change MixEmul's execution mode. MixEmul can run in either of the following modes:
  - In Attached mode the main window is regularly updated when MixEmul is running. The controls, registers, memory and devices sections are updated to show the current status of the respective parts of the emulator. Using this mode allows you to monitor the effects of your programs as they are running. However, because the updating of the main window takes time, programs run slower than in Detached mode.
  - In Detached mode MixEmul runs the programs it executes in the background. In this mode the main window is not updated until execution stops, but programs will run faster.

The execution mode can be changed when MixEmul is running.

- A Clear Breakpoints button. Clicking this button will clear any breakpoints that are currently set.

- A Reset button. Clicking this will Reset MixEmul. All register and memory values are set to 0, as are the program and tick counters. All devices are reset as well.
- A Show/Hide Teletype button, which does exactly what its name implies. More information about the teletype can be found below.

### 2.2.3 Registers section

The registers section looks like this:

The screenshot shows a window titled "Registers". It contains a list of registers with their current values and editors. The registers are rA, rX, r1, r2, r3, r4, r5, r6, and rJ. Each register has a sign (+/-) and a value editor. rA and rX have 8-digit editors, while r1-r6 and rJ have 4-digit editors. All values are currently 0. There are also checkboxes for "Overflow" and "Compare".

Register	Sign	Value Editor	Current Value
rA:	+	00 00 00 00	0
rX:	+	00 00 00 00	0
r1:	+	00 00	0
r2:	+	00 00	0
r3:	+	00 00	0
r4:	+	00 00	0
r5:	+	00 00	0
r6:	+	00 00	0
rJ:		00 00	0

Overflow: ☐ Compare: ☐

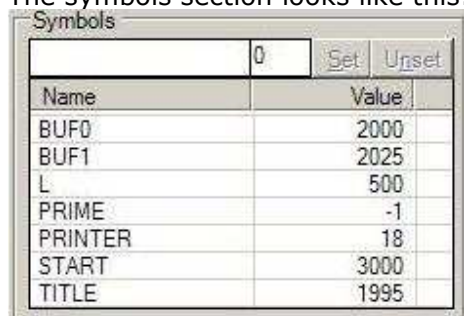
It contains the following items:

- Value editors for the registers. Values can be edited per byte (left of the equal sign) or as a composed value (right of the equal sign). The editors for rA, rX and the index registers support negative values. In the byte editor the sign can be changed by clicking it. A byte or composed value that has been edited but not yet applied is shown in blue. Changes are applied by pressing Enter or leaving the field in question and can be aborted by pressing Escape.
- An overflow indicator/editor. The box is checked when the overflow flag is set.
- A comparison indicator/editor. The current value of the comparison register is shown, it can be changed by clicking the indicator.



## 2.2.4 Symbols section

The symbols section looks like this:



Name	Value
BUF0	2000
BUF1	2025
L	500
PRIME	-1
PRINTER	18
START	3000
TITLE	1995

Upon loading a MIXAL program into MixEmul, the symbols section shows the named symbols that were included in said program.

It is possible to perform the following actions using the symbols section:

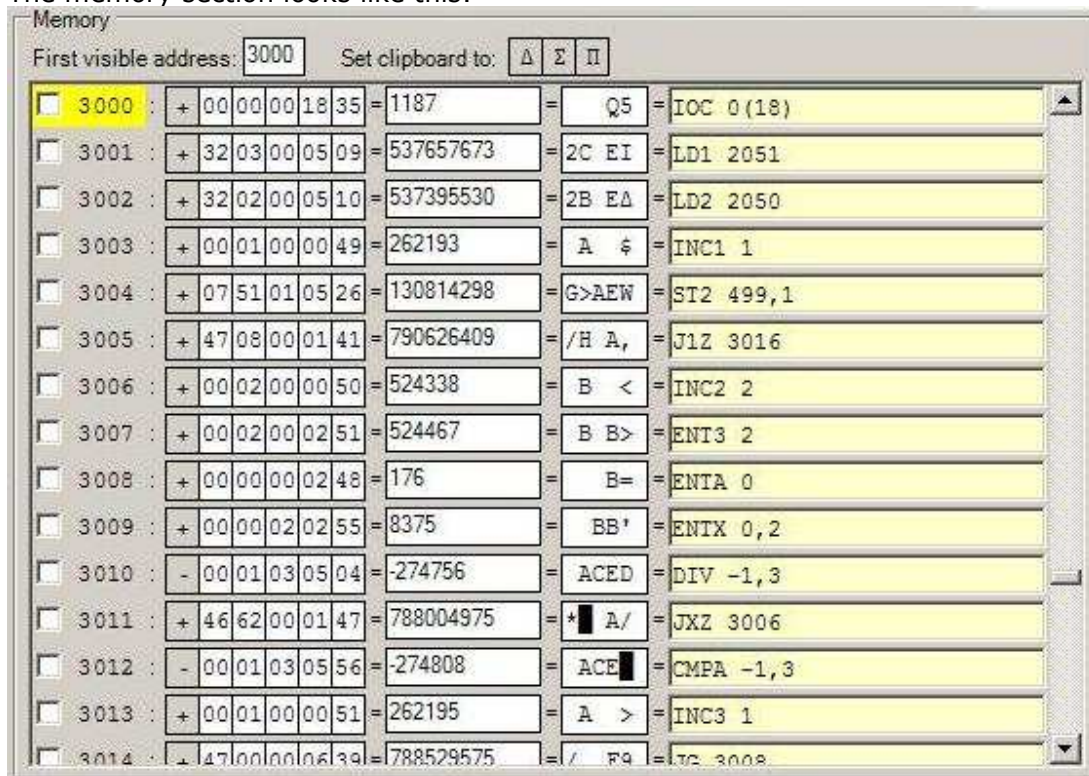
- Adding a new symbol, by entering a unique, valid, symbol name and the value it should have, and pressing the Set button.
- Modifying an existing symbol, by entering the symbol's name or selecting it in the list, entering the symbol's new value, and pressing the Set button.
- Removing an existing symbol, by entering the symbol's name or selecting it in the list, and pressing the Unset button.

The symbols that are included in the symbols section, regardless if they are loaded from a program or entered manually, can be used in instructions that are entered into the instruction editors in the memory section (see below).

For those symbols of which the value is a valid memory address, the address in question can be made visible in the memory section (see below) either by double-clicking the symbol or by selecting it and pressing Enter.

## 2.2.5 Memory section

The memory section looks like this:



It contains the following items:

- An editable field that contains the address of the first visible memory word.
- Three buttons with the delta ( $\Delta$ ), sigma ( $\Sigma$ ) and pi ( $\Pi$ ) characters, which I could not find on my keyboard. Clicking any of these will put the character that is shown on the button on the clipboard.
- A value editor for each of the memory words. Each of these consists of the following:
  - A checkbox that can be used to mark the address as a breakpoint;
  - The memory word's address. If the address is equal to the program counter, the address and checkbox are marked in yellow;
  - Byte and composed value editors that are identical to those for rA and rX (see the discussion of the Registers section, above);
  - A character editor. Each character corresponds to one byte of the memory word. Byte values for which no character is available are shown as a vertical block (■);
  - An instruction editor.

The instruction editor merits a slightly more detailed discussion. The instruction editor for a memory word shows, if possible, which instruction the word's value represents. Values that do not represent an instruction are indicated with the text *Not an instruction*.

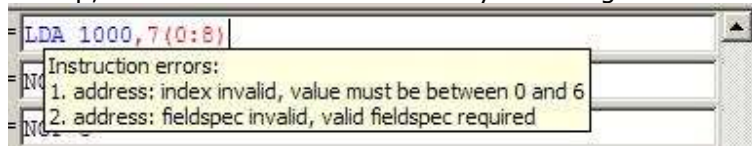
Being an editor, it is also possible to enter the instruction you want the memory word to contain. Instructions entered in the editor must comply with MixEmul's MIXAL format (see the discussion of MixEmul's assembler, below), with the following restrictions:

- The location field is not supported and must not be specified;
- Only MIX instructions are supported. That means that the following instructions cannot be entered: EQU, ORIG, CON, ALF and END;

- In the address field, the only symbols that are supported are those included in the symbols section (see above), and \*.

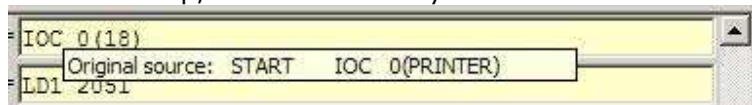
As with the other editors, any changes made can be applied by pressing Enter or leaving the editor.

Any invalid parts of the instruction (both in case of a word value being shown or an instruction being parsed) are marked in red. The actual errors are shown in the editor's tooltip, which can be made visible by hovering the mouse cursor over the editor:



Please note that an invalid instruction will not be applied. This means that if the editor is left while it contains an (edited) invalid instruction, the changes that were made are lost.

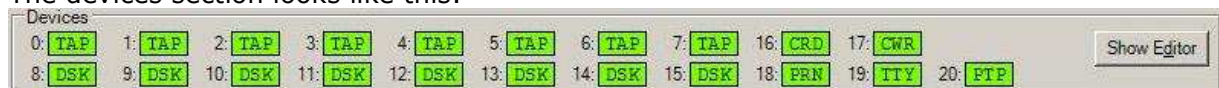
If a program has been loaded into memory, the memory addresses that contain instructions or values (CON and ALF) that were assembled and loaded from the source file are marked by a different background color. The actual instruction that was loaded is then shown in the editor's tooltip, in addition to any instruction errors that were found:



Right-clicking an instruction editor shows a menu through which the memory address that is referred to in the instruction's address field can be made visible in the memory section. Through another option in the same menu this can be done with indexing applied. Either option is only available if the address (and index, if applicable) is in fact valid.

## 2.2.6 Devices section

The devices section looks like this:



It contains indicators for each of MixEmul's devices.

Each of these shows the following:

- The device number, as used in the F-field of the I/O instructions.
- An abbreviated name of the device. The following abbreviations are used:
  - TAP: Tape
  - DSK: Disk
  - CRD: Card reader
  - CWR: Card punch
  - PRN: Printer
  - TTY: Teletype
  - PTP: Paper tape

- The device's status. For idle devices the abbreviated name is shown on a green background, busy devices on a yellow one. More information on the device's status is shown in the indicator's tooltip, which is made visible by hovering the mouse cursor over the indicator:



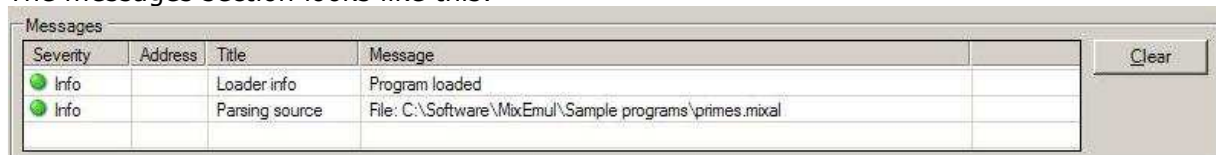
Right-clicking an indicator opens a menu that shows the following:

- If the device supports input, output or both;
- An option to reset the device to its initial state.

Right next to the indicators is a button with which the device editor window can be opened or closed. More information about the device editor can be found below.

## 2.2.7 Messages section

The messages section looks like this:



Messages that are generated by MixEmul or its devices are shown in this section, newest message first.

If the message includes an address then that address can be made visible in the memory section by double-clicking the message, or by selecting the message and pressing Enter.

The messages section can be emptied with the Clear button and resized with the splitter that is located directly above it.

## 2.3 Using the assembler

### 2.3.1 Overview

MixEmul's built-in assembler can be used to parse MIXAL programs written using an external editor and then load them into MixEmul.

At the moment MixEmul doesn't allow you to assemble your programs into binary executables. The reasons for this are that (a) I don't think that there would be much use for them besides loading them into MixEmul, and (b) the assembly of typical MIXAL programs doesn't take enough time to merit avoiding it.

The assembler is a two-pass assembler, which performs the syntactic checks during the first pass and the semantic checks during the second. If your program is correct you won't notice anything of this, but you will if it contains both syntactic and semantic errors. In that case only the syntactic errors are reported.

### 2.3.2 MIXAL input format

The MIXAL assembler accepts the terminal input format that is discussed in TAOCP section 1.3.2 with the following exceptions:

- Any characters that are included in an ALF address field but are not known to MIX are parsed as a byte with value 63.
- The address field of the ALF instruction starts with the first non-blank character after the blank(s) that follow the op field. To allow the ALF address field to start with a blank, double quotes (") can be used around the address field. The last double quote of the instruction line that is followed by a blank is considered to be the closing quote for the address field. This allows for double quotes to be included in the address field without the need for "escaping" them. (Not that there's much use in doing so, because the double quote isn't part of the MIX character set. But anyway...)

Consider the following examples:

- ALF FIRST and ALF "FIRST" both yield a word containing the characters FIRST (06|09|19|22|23).
- ALF FIVE and ALF " FIVE" yield different words. The first word contains FIVE followed by a space (06|09|25|05|00), the second a space followed by FIVE (00|06|09|25|05).
- ALF "QUO"TE yields a word containing 18|24|16|63|23
- ALF "QUO" TE yields a word containing 18|24|16|00|00, and a comment TE.
- ALF "QUO" TE" yields a word containing 18|24|16|63|00
- The rule that the END instruction is the last instruction of a program is enforced. Any instructions following an END instruction yield a warning and are treated as comment lines.

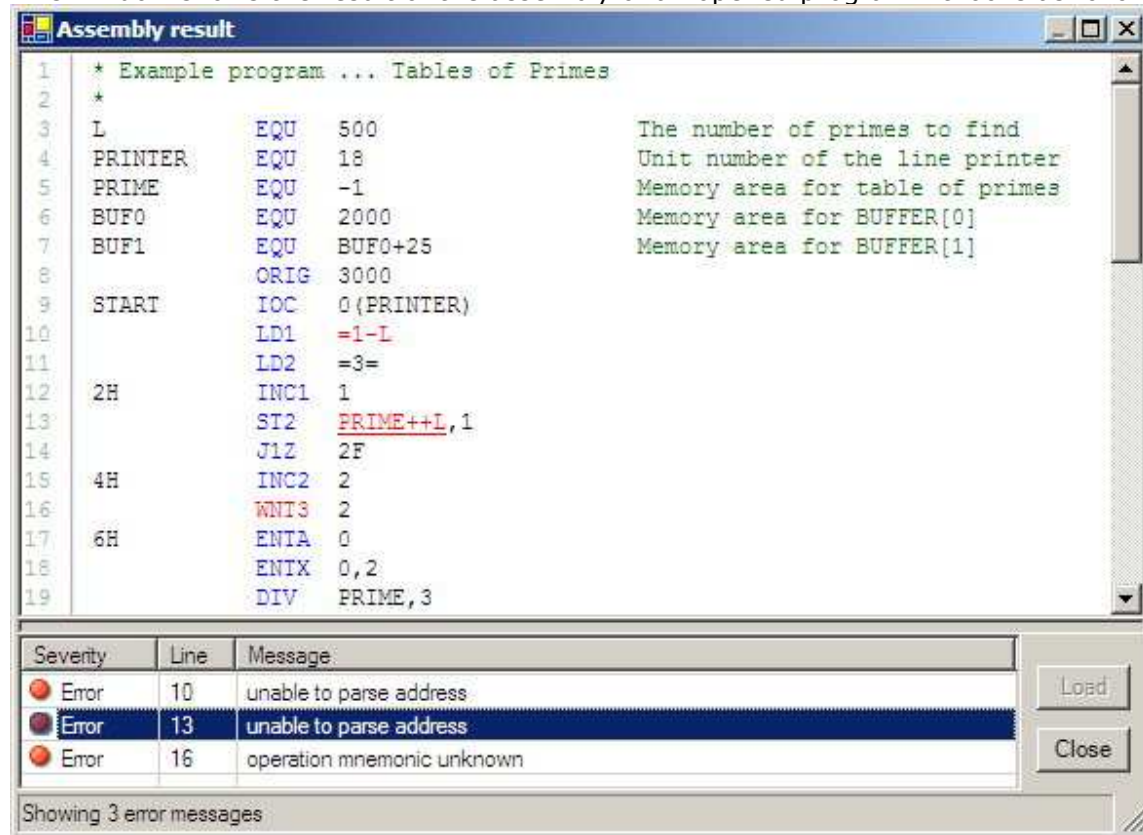
### 2.3.3 Invoking the assembler

MixEmul's assembler is activated by opening a MIXAL program. This can be done using the "Open program" option in the File menu.

The assembler is invoked automatically when the MIXAL program has been read from disk. After assembly completes the assembly result window is shown.

### 2.3.4 Assembly result window

This window shows the result of the assembly of an opened program. It looks as follows:



The lower part of the screen shows the messages that were generated during assembly. The upper part of the screen shows an evenly indented version of the MIXAL program that was assembled, with syntax coloring applied. Warnings and errors are also color-marked in the program source code.

In this case three syntactical errors were found. If a message is selected in the message list, the appropriate part of the source code is made visible and underlined.

If no errors were encountered during the assembly then the assembled program can be loaded into MixEmul's memory by pressing the Load button.

## 2.4 Using devices

### 2.4.1 Device implementation

A few words about the way the devices are implemented are in order. The I/O operations (IN, OUT and IOC) are implemented as a series of steps. The general structure of these is as follows:

- IN: Initialization, Open file (if applicable), Read from file, Close file (if applicable), Copy read words to memory
- OUT: Initialization, Copy words to write from memory, Open file (if applicable), Write to file, Close file (if applicable)
- IOC: Initialization, Seek.

The number of ticks spent on the Initialization step is configurable per device (see the discussion of the preferences window, below), as is the seek time per record/sector for those devices that support seeking. Reading from and writing to memory is done one word per tick.

On binary devices (tapes and disks) words are stored as six bytes. The first byte contains the word's numeric sign (+ or -) and is followed by the five bytes of the word.

Text-based devices (card reader, card punch, printer, teletype and paper tape) are read from and/or written to one line at a time. For example, if the instruction IN 1000(16) is executed then one line is read from the card reader device file (default: crdin16.mixdev). If a line from an input file is longer than the device's record byte size (80 in case of the card reader) then the rest of the line is discarded. If a line is shorter then the remaining bytes in the record are set to 0.

On input any characters that are not known to MIX are parsed as a byte with value 63.

On output trailing spaces are dropped before a line is written.

On startup or after a (device) reset the location pointer is at the following position:

- For tapes, at the end of the device file;
- For disks, at the beginning of the device file (sector 0);
- For the card reader, at the beginning of the device file;
- For the card punch, at the end of the device file;
- For the printer, at the end of the device file;
- For the paper tape, at the beginning of the device file.

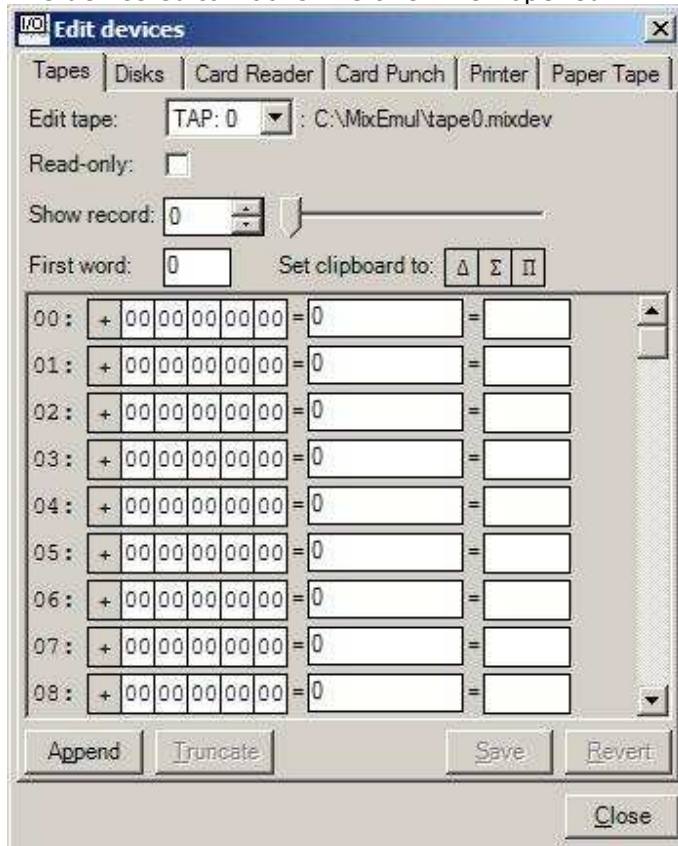
### 2.4.2 The device editor

The device editor can be opened and closed by clicking the Show/Hide Editor button in the Devices section of the MixEmul main window, or the Show/Hide Device Editor option in the View menu.

Through the device editor, the contents of all of MixEmul's devices except the teletype can be viewed and, if applicable, edited. The teletype device has its own window, which is discussed below.



The device editor looks like this when opened:



The device editor contains 6 tabs, one for each type of device.

For the tapes and disks, a binary device editor is used. Through this editor, one record/sector can be edited at a time.

It is made up of the following parts:

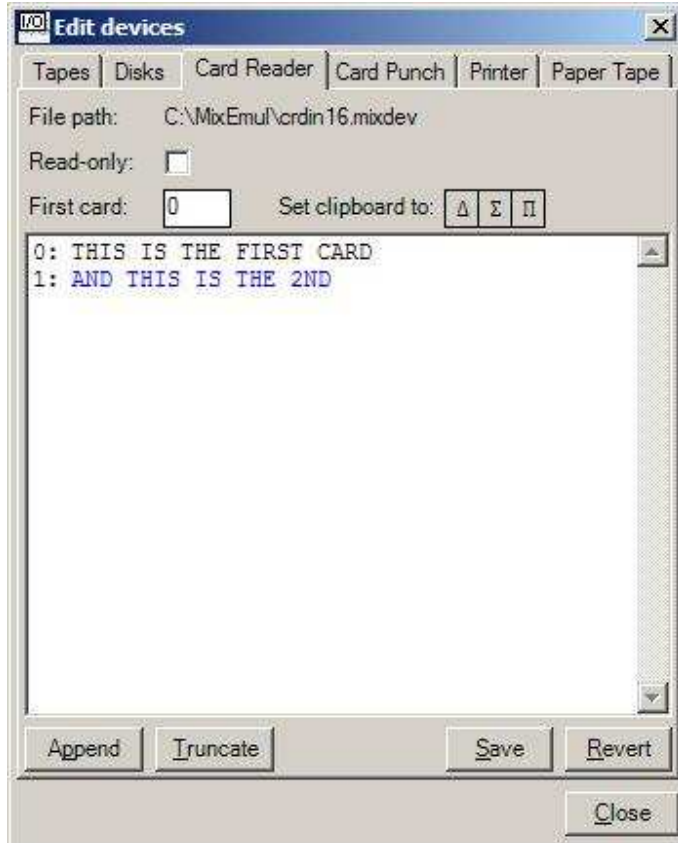
- A selection control to choose which tape/disk to edit. Next to it, the selected device's file path is shown.
- A checkbox to switch the device editor to read-only (to prevent unintended modifications of the device's contents).
- A text field and slider to select the record/sector to edit.
- A text field that shows/sets which word is shown at the top of the record/sector editor.
- Buttons to load Mix-specific characters onto the clipboard. These work the same way the buttons in MixEmul's main window's memory section do.
- A list of editors for the words in the record/sector. These are identical to the editors in MixEmul's main window's memory section, except that there is no instruction text field in the device editor.
- For tape devices, buttons to append or truncate records to/from the end of the tape.
- Save and revert buttons to save or undo the changes to the current record.

Note that upon changing to another record/sector or selecting another device (through the aforementioned selection control) or device type (by clicking the appropriate tab), any changes to the current record/sector will be automatically saved.



For the card reader and paper tape, a text device editor is used. Each of these editors allows all of the device's contents to be edited at once.

It looks like this:



The editor is made up of the following parts:

- An indication of the selected device's file path is shown.
- A checkbox to switch the device editor to read-only (to prevent unintended modifications of the device's contents).
- A text field that shows/sets which card or record is shown at the top of the device content editor.
- Buttons to load Mix-specific characters onto the clipboard. These work the same way the buttons in MixEmul's main window's memory section do.
- A list of editors for the device's contents. These are identical to the character editors in MixEmul's main window's memory section, except that the maximum number of characters per editor is equal to the respective device's record length.
- Buttons to append or truncate cards/records to/from the end of the device.
- Save and revert buttons to save or undo the changes to the current device.

Note that upon selecting another device type (by clicking the appropriate tab), any changes to the current device will be automatically saved.

For the printer and card punch, a read-only viewer is used. These are identical to the text editors for the card reader and paper tape, except the following controls are not shown:

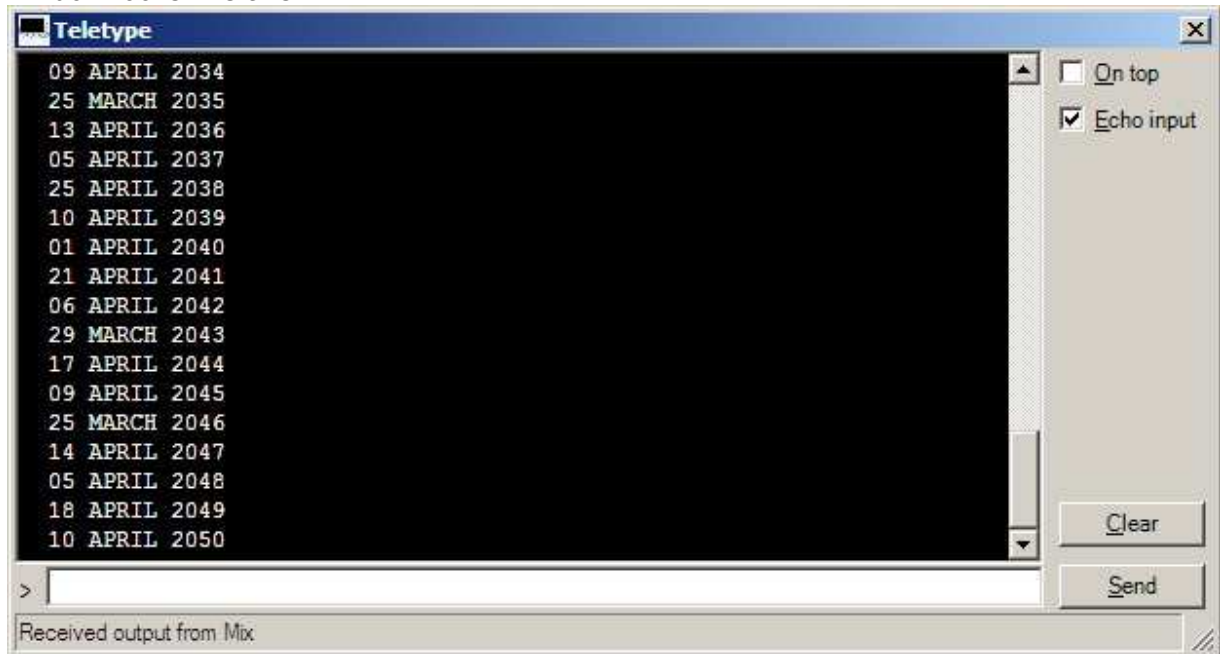
- The read-only check box
- The Mix character clipboard buttons
- The Append, Truncate, Save and Revert buttons

Besides the aforementioned Hide Editor button and menu option, the editor can also be hidden using the Close button at the bottom right of the editor window.

Whenever a device editor is visible – the device editor window is visible and the device editor tab in question is selected – the respective device's file is monitored for changes. When a change is detected, because the device was written to by MixEmul or manual modifications were made, the device editor's contents will be automatically reloaded. The minimum interval between these reloads is configurable using MixEmul's preferences window (see below).

## 2.5 Teletype window

The teletype window can be shown or hidden using the Show/Hide Teletype button on MixEmul's main window or the Show/Hide Teletype option in the View menu. The teletype window looks like this:



The upper part of the screen shows data that the teletype has received from MIX (i.e. the data that has been transferred using OUT ...(19) instructions). The output area can be cleared with the Clear button.

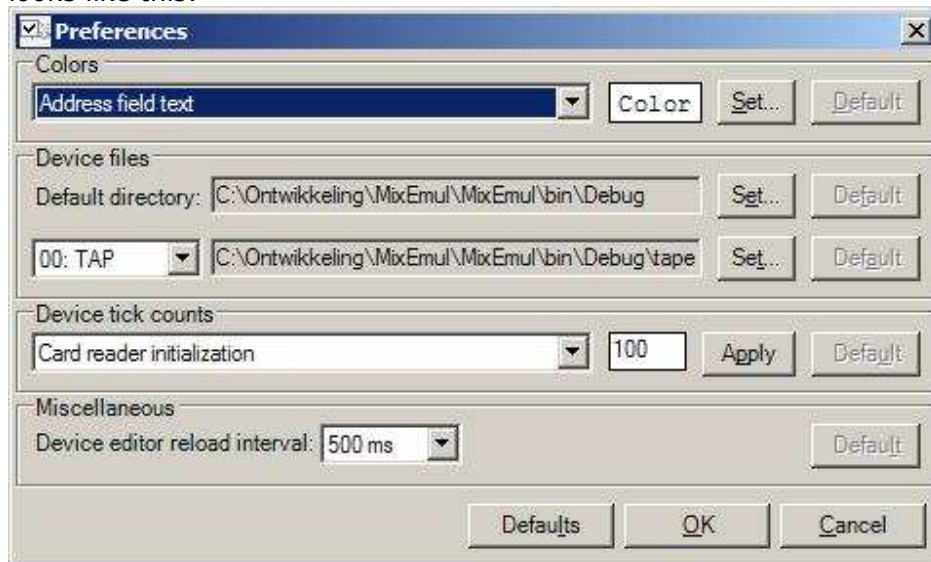
Input can be entered in the field at the bottom of the screen. If the "Echo input" option is checked then input that has been sent (by pressing Enter or clicking the Send button) is echoed in the output section, preceded by the prompt character (>). Input is stored in an input buffer, where it is kept until it is retrieved using the IN ...(19) instruction.

If an IN instruction is executed while the input buffer is empty then program execution is halted so that teletype input can be sent.

If the "On top" option is checked then the teletype window is kept in front of all other windows.

## 2.6 Preferences window

The preferences window can be accessed via the Preferences option in the Tools menu. It looks like this:



The following preferences can be changed:

- A selection of the colors that are used throughout MixEmul. Most of the color names speak for themselves, but the following might need some explanation:
  - Address text vs. Address field text. The Address text color is used for the address numbers in the Memory region of the main window. The Address *field* text color is used for the address fields of the MIXAL programs as they are shown in the assembly result window. Similarly, the Location field, Op field and Comment text colors are used for the respective fields of the assembly result window.
  - The Rendered text color is used for text that displays information (e.g. in editors) which has not yet been edited. The editing text color is used for text that has been edited but the changes to which have not yet been applied.
  - The loaded instruction text and background colors are used for instruction editors into which a MIXAL instruction has been loaded by the assembler (see also section 2.2.5).
- The default directory for device files. This directory is used to store all device files for which a specific file hasn't been selected using the preference that is discussed next. Note that any existing device files are not automatically moved if this directory is changed.
- The device files for individual devices. Note that an existing device file is not automatically moved or renamed if this setting is changed.
- Tick counts for a number of I/O operation steps (see also the discussion of the Devices region, above). This enables you to modify the speed at which the MixEmul devices operate.
- The minimum interval between reloads of the contents of the visible device editor, upon modification of the device's file on disk.

For each preference it is possible to revert to the default setting using the respective Default button at the right. All preferences can be reset to defaults at once with the Defaults button at the bottom of the window.