Table of Contents

	Foreword	0
Part I	Black Magic Composer	3
Part II	Many Thanks	3
Part III	Preamble	3
Part IV	Controlling the BMC	4
Part V	Loading a demosong	4
Part VI	The Main Sequencer	4
Part VII	The Sequence editor	5
Part VIII	The Sample Sequence editor	6
Part IX	The Instrument editor	7
Part X	Playing music	8
Part XI	Loading and managing samples	9
Part XII	The Info Menu	10
Part XIII	The Songs Menu	10
Part XIV	Disk Operations	11
Part XV	The Comp Menu	11
Part XVI	The Special Menu	12
Part XVII	Using songs in homebrew software	13
Part XVIII	Using songs in BASIC	13
Part XIX	Using songs in Assembly	14
Part XX	Keys you can use	15
Part XXI	Finally	15

Index 0

1 Black Magic Composer

2 Many Thanks

Black Magic Composer - Many Thanks

Before I start with the program description, I would like to thank the following:

... My friend and partner Thorsten Winkler, who kindly loaned me his computer (and hopefully will do that again...)

... BenjySoft, who weren't especially amazed by our editor, but who gave me the incentive to make one myself (I'm curious how your editor will look when it's finished! Good luck!)

... Ulf Petersen for publishing the demo version on his diskmagazine, and for buying the finished version from us.

... And last but not least Chris Huelsbeck, of whom I took some ideas.

Greetings go to:

- Peter Sabath, who's an excellent programmer, but who must have a terrible taste in music.

- Willem Walraven from Breda, for sending us the table for 16-Bit sound (hartelijk bedankt!).

- and everyone else, who helped keeping the Atari 8-bit alive.

3 Preamble

BMC is a program that enables everyone with a little musicality, to create technologically sound music pieces, and to incorporate them in their own programs. BMC utilises a special user interface and produces high quality music, that makes other music editors pale in comparison. The most important features include:

16-Bit sound, Sine waves, Sawtooth waves, integrated samples, up to four songs in one module (important for games). The songs can be saved in an executable format.

I've tried to write this manual in a way that even amateurs can understand, but I refrained from explaining music theory like "portamento" in detail, because this would make this manual a lot bigger. People that like to use their songs in homebrew programs, should at least be a little familiar with hexadecimal numbers. Help on that is available in a lot of magazines, books on programming etc. This manual also won't showcase all the tricks and possibilities, because only the limitations of the hardware apply. To make better sense of this manual, it's preferential to have the BMC loaded while reading. Also, it's highly recommended to take a close look at the demo-songs included.

4 Controlling the BMC

BMC has a comfortable graphical user interface. Controlling it is mainly done by Joystick, or a mouse. The joystick should be plugged into port #1. A mouse should be plugged into Port #2. Choosing between an ST or Amiga mouse, can be done by pressing <select>, in which case, the cursor will briefly change color to acknowledge the change. (If you're not sure what mouse type you have, it won't hurt to try changing types.)

The icons on the lower part of the screen can be used to switch between the main parts of the program, by moving the cursor over it and clicking the button. Each icon will open a specific window, which I'll describe furtheron.

There are however more options, which you'll find in the pull-down menus on the top of the screen. To select any of them, click and hold the button, while pulling down, selecting the option, then release the button. Some of these menu items toggle functions, in which case a tick after the menu option will show it being active.

Windows can usually be closed by clicking the little box-icon on the top left. This will also cancel some functions, like loading or saving. Arrows in the corners are always used to increase or decrease a value. To change a specific entry, it suffices to click the value. A cursor appears, to prompt keyboard input. **During most keyboard input, the mouse will not function.** Keyboard input has to be completed before any other function can be chosen. In some cases, there is more data than can be shown in the window. In those cases, there will be a scrollbar on the right or lower part of the window.

Note by the translator: To continue loading when the title screen is displayed, press START.

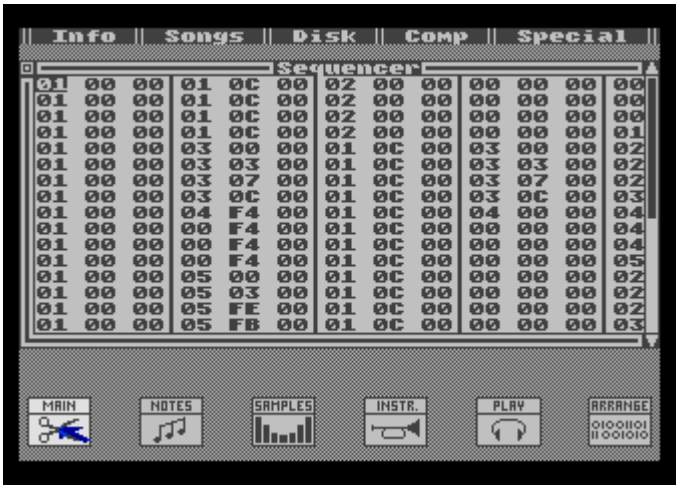
5 Loading a demosong

To load a song, choose the option "MLOAD" from the menu "DISK" and enter the filename. Don't forget the extension, which is ".MSC" for songs. You do however not include any drive indication. See "The Special Menu" for details on how to set the drive number for file operations. If you don't know the name of the song, you can find out by using the option "DIR". Is the song loaded, clicking the Play-icon will open up the player. The buttons on the player resemble those of a casette player. Click the righthand side button first, then the play button.

If a song has samples, the according pack (unless it has the same file name as the song) has to be loaded seperately. These will have the extension ".DRP" and are loaded by the menu option "DLOAD".

6 The Main Sequencer

The main Sequencer is the part of the editor which holds the order of the sequences, and what voice plays which sequence. To open up the sequencer, click the "MAIN" icon. A window filled with hexadecimal numbers appears.



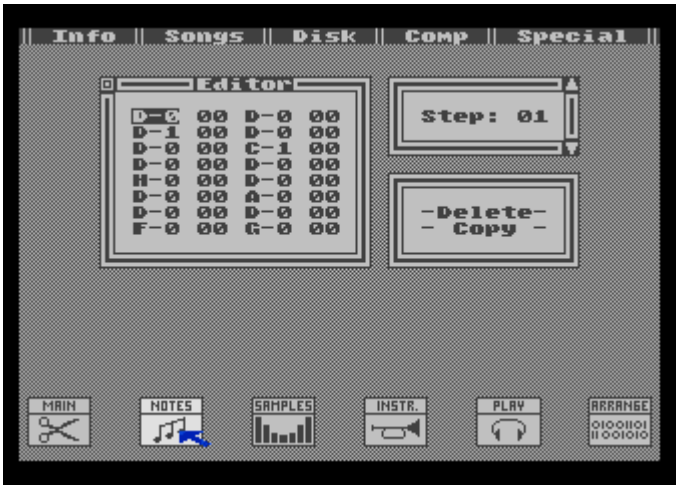
The window is coarsely divided into 5 columns. Column 1-4 are for the voices, the 5th is reserved for samples (digital sounds, usually drums). For each voice, there are 3 entries. The first of these holds the number of the sequence to be played. The second number holds a transpose value, in half-note steps. (01,02 etc will transpose upward, ff, fe etc will transpose down). The third number holds a value that will be added to the instrument numbers in the sequence. Combining these, makes you able to (re-)use one sequence in multiple pitches and with different intruments. (See also the sequence editor, and the instrument editor.)

The table has 128 lines. You can choose what lines to display by using the scrollbar on the righthand side of the window. The cursor can be positioned by the cursor-keys aswell as the joystick or mouse. With the keys <Insert> and <Delete> you can insert or delete rows in the table.

To inform the player routine what the end of a song is, it needs to be specially marked. By entering <Control-R> it'll repeat the song from the start in an endless loop, while entering <Control-E> will make the song end.

7 The Sequence editor

In this window, all the notes are entered, that make up a sequence. You can open up this window by clicking the "NOTES"-icon. It doesn't use normal music notation, instead it has a simpler, more technially oriented input method. Notes are listed in table form which will be played from top to bottom at a certain speed. If a note exists at the location in the table, it will be played.

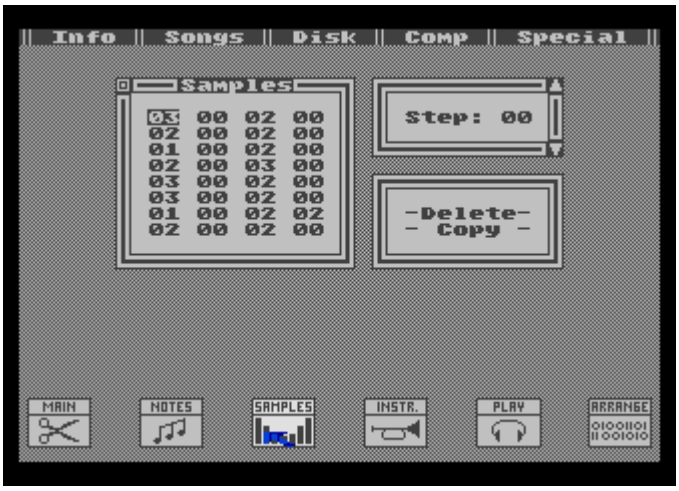


Notes are entered in a "Note-Octave" notation. First enter the note name, then the octave. To obtain sharp notes, press <Control> while entering the note. For 8-bit sound, only notes between C-0 and D-3 are useful. For 16-bit sound, this is C-0 till C-9. To delete a note, simply press <Space>. After the note, the instrument number is entered. Which sequence is edited, can be chosen with the arrows on the top right.

Clicking "DELETE" will delete the entire sequence. Use with caution. This is an unrecoverable action. To copy a sequence, you first need to select the sequence to copy TO, then click and hold on "COPY" and move up/down to select the sequence to copy FROM. Then release the button.

8 The Sample Sequence editor

This window can be activated by selecting the "SAMPLES"-icon. It has a very comparable interface as the sequence editor, with a slight difference: No notes are entered, but instead the number of the sample is. In the same time one note can be played, two samples can be played, hence this table has twice the number of entries.



Important: When a sample is played, there can't be any notes played by the 4th voice.

9 The Instrument editor

The Instrument editor can be called forth by clicking the "INSTR."-icon. This window is used to define instruments to be used in the song. The top left holds parameters that affect the entire intrument.

- **Vibrato min-max:** This first number enables vibrato. Strongest vibrato is accomplished with the value 1F, slightest vibrato with F1. (Both values give the maximal difference between the original sound. With F1, this boils down to -1 to 1.)

- Vibrato delay/speed: This value incorporates the delay (left value) and the speed of the vibrato. (For 8-bit sound, 1 is the speediest, F the slowest. For 16-bit sound, this is reversed for technical reasons.)

- Glide: This entry can be used to hav the note glide up or down smoothly and rapidly (A.k.a. "Glissando"). Values of FF, FE, FD etc will make it glide upwards in pitch, values of 01,02,03 etc will glide it down.

- Detune: This value enables a slight pitch change on the original tone, which might be used to give a "floating" effect.

- 16Bit on/off: A value of 01 will enable 16 bit mode. This links voice 0+1, and/or 2+3 into one voice. To use this function, both voices should henceforth play the same sounds. (A good example of this can be found in the demosong BIT.MSC.)

- Transpose off: For some sounds, specially drum sounds, transposing is not desirable. Entering a value other than 0 will stop the main sequencer from transposing this instrument. (See the main sequencer for details on transposing options there.)

- Sound Control: This gives some options to manipulate the sounds. (For people that know the Pokey chip more intimately, this is NOT related to the Sound Control Register $D20F.) This value can only be used with voices using the distortion value "A" (clear tones).

Values for the Sound Control entry are:
- 01: Detune (with a difference). Voice 0+2, or 1+3 are connected, playing the same note, with a slight detune between them, giving a floating effect.
- 03: Voices 0+2, or 1+3 are connected through a special filter, giving an interesting effect. (play with it)
- 02: Combines both of the above.

- 10: Sinewaves. Voice 0+2 are combined into a single sound register. Two very high frequency notes are combined to give one tone in a sine wave. This method is not very exact and will make most notes sound detuned.
- 11: Sinewaves, with double the frequency. This option theorethically expands the sinewave output scale to 4 octaves!
- 13: Sawtooth waves. The same notes apply as for the Sinewaves.
- 12: Sawtooth waves, with double the frequency.

Important: Whenever two voices are combined, they both have to play the same notes at the same time!

The envelope (output strength, volume) of the instrument can be edited on the lower right side of the editor screen. The top value is the volume, the lower one is the distortion. (See the Atari Basic manual!) The values can be changed with the keyboard, or in the graphical representation with the mouse.

Each instrument table is a maximum of 128 entries long, and can be scrolled with the scrollbar below. There is a red and a green marker in the instrument definition. The first marks the end of the instrument, the other marks the entrypoint where the instrument loops to when it's played longer than the defined length. Setting these markers is possible by pressing "<" and ">".

Arpeggios are entered in the lower left part of this editor. These 4 entries show transpose values to rapidly change the pitch of the tone. Entering 00 four times, will not change the note. Values like 00/03/07/0C will rapidly arpeggiate though one octave. (This sounds a bit complicated but can sound pretty darn well!) If you'd like to use an arpeggio with less values, the remaining, unused values should be set to 80.

Setting the instrument number, aswell as the copy function, work the same as in the sequence editor.

10 Playing music

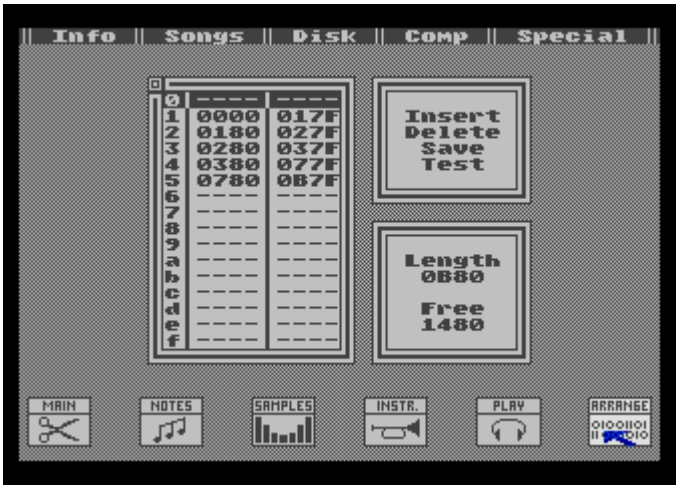
Logically, clicking the "PLAY"-icon opens up the player.

The 4 bars above show the current volume of the voices. The buttons look and function like a casette player, with the exception of the "eject"-button, which is used to initialise the song. (Use this after disk operations, or any change in the song, or you'll suffer distortions during playback.) If the song uses digitised sounds, you will not be able to move the mouse-cursor. It will switch to the "stop" position though, so a single click will stop the music, relinquishing control back to the joystick/mouse. Spooling

forwards/back only works on a per-sequence basis.

11 Loading and managing samples

To be able to use samples in a song, you'll need to load them beforehand. To enter that editor, click the "ARRANGE"-icon.



To the left, you'll see a listing of all the samples currently loaded. (To a maximum of 15 samples.) Sample number 0 can't be used, as the value 0 signifies "none".

To insert a digi-sound, just click on the first free entry, (with an empty table that would be entry 1) then select "INSERT" and fill in the filename. On the original disk, we've included several drum samples with the ".SMP" extension. If an entry is already in use, all consecutive entries move up one place, and the new sample is loaded in the selected spot.

With the function "DELETE", a single sample can be removed. "SAVE" gives the oportunity to save a single sample. "TEST" is in there so you can make sure you actually loaded the right sample in the spot. Using "TEST" gives a slightly lower output quality in comparison to the quality on playback of the song.

If a sample is deleted within a song, for example because it's not used anymore in the piece, this would pose a slight problem, because all the sequences woud have to be changed to reflect the change in order. No need! The BMC will automaticly change all sample numbers in the Sequencer on the fly!

The total amount of space taken by samples can't exceed the magic number of $2000 bytes (4096 bytes). The remaining free space for samples is therefore shown on the bottom right side. If a sample will not fit in the remaining space available, its loading process will be aborted.

All 4-bit samples, saved in the ".COM" format, can be loaded. These samples should have been created with a 7.7kHz frequency, which boils down to one sample every 2 scanlines.

For ease of use, the whole sample pack can be saved and loaded in one operation. See Disk-operations for details.

12 The Info Menu

This part, aswell as the following, explain the menu options on the top line of the GUI. Beginning wth the Info menu.

This menu is there to inform you of the editor version, and/or player version you're working with. Each of the windows can be closed by clicking the "close window" -box on the top left.

13 The Songs Menu

This menu holds all options that are global to a certain piece of music. One song module can hold up to 4 songs. Therefore, there are also quasi 4 distinct sequencer tables. Using this, one song module can, for example, hold the title music, in-game music, aswell as the game-over tune, for a game. This means you can play them all, without loading separate songs for each instance.

SELECT: This function will enable you to set the tune in the song package to edit. The following points then apply only to the tune currently selected.

DIGIS: Here you can enable/disable digisounds for this tune. If you do use digital sounds in a piece, the program running it from, will halt until the song is stopped. The entire CPU capacity is needed to

reproduce the digitised sounds.

15kHz: This option will switch the entire player to 15kHz base frequency, instead of the normal 64kHz. This doesn't apply to 16-bit voices. It does however enable a sort of "Bass-mode" for 8-bit channels.

DELETE: This will delete the entire song (sequencer table) from the current file. You might need to do some additional cleaning to save space though.

14 Disk Operations

This menu denoted "DISK" holds most options you should know from DOS. The main options should be pretty obvious.

MLOAD: This loads the entire song module from disk. (Don't forget to add the ".MSC" extension!) If there's a digi-pack with the same name (but with extension ".DRP") it'll automaticly be loaded aswell.

MSAVE: This saves an entire song module to disk. In contrary to MLOAD, any digi-pack has to be saved seperately by using the DSAVE option.

DLOAD: This loads a digi-pack from disk (Usual extension ".DRP").

DSAVE: This saves a digipack from memory to disk. Separate digi's can only be saved from the "ARRANGE" window.

You could use your own extensions to save your stuff, but we recommend using these:
.COM for compiled music modules
.MSC for song modules.
.DRP for module based sample packs.
.SMP for single samples.

All disk operations can be aborted by closing the window for filename entry.

15 The Comp Menu

This menu only has one option: "PACK". This enables you to save the song pack as an executable.

WARNING: Before you invoke this, SAVE your work! After using this option, you will need to restart the editor by cold boot! You will loose all your editing and sequencing if you don't save before!

Using these executable song modules in your own programs, will be discussed later.

16 The Special Menu

This menu has some options that affect the editor or the play module.

Set Speed: This will set the global speed (or delay) used by the player routine. Fastest setting is "1", slowest is "9".

Set Drive: This option selects which drive to use for operations. You MAY use the ramdisks offered from Dos 2.5 aswell as BiboDos, but this does need a special setup. (The original disk doesn't install these. You need either a Dos 2.5 disk including ramdisk.com, or a disk with Dos.sys from BiboDos.)

Clear All: This option is most coveted by frustrated composers and will clear out all song and digi pack information and will basicly leave you with a freshly started composer.

17 Using songs in homebrew software

Before you can use the songs, created by this program, in your own homebrew software, you'll need to use the "PACK" option in the "COMP" menu. When you click this option, it'll ask you if you're sure.

THIS IS IMPORTANT! Because, when you execute this function, it'll erase the entire composer from memory, and compiles the song module into an executable format. There's no returning to the editor, except when you reboot the disk!

WHEN you answered "YES" to this question, you're asked how many songs there are in the module, to stop the compiled song module from getting overly large. It will then optimise the song, which will take a few seconds. Unneeded data is cut out, instruments are optimised, aswell as translating the digitised sounds. Also, the definative length of the executable is calculated. This process takes a few seconds.

Now, you're asked what starting adress should be used to load the compiled module. After that, the routines are modified to reflect the adress, and you'll be prompted for a filename. This will then be the end of your editing session. The song can be saved multiple times, but there's nothing else to be done except rebooting your Atari.

18 Using songs in BASIC

On the original disk, a little basic program is included, called "PLAYER.BAS", which uses some data-lines to setup two small machine language routines. The first is the load routine, the second is the player invoker. You can use his program as a template to incorporate the sound modules into your own basic program.

The module file should be loaded at the start of the program. The supplied program uses the data starting at line 20000 for that and the call X=USR(1536,1,7,ST,EN-ST). In Turbo-Basic, this could be replaced by a BLOAD command.

The secundary machine language routine is then installed and used, to start a song in your module by calling X=USR(1536,Startadress,Songnumber). If the song doesn't have any digitised sounds, the basic program will run on after starting the song. Otherwise, it'll be halted until <Start> is pressed. A running tune can be stopped by the command X=USR(1536,Startadress,0) otherwise.

But how to come up with a good starting adress to use for the compiler? Well, it's not overly complicated, yet has it's downside. First, you check the top of memory by doing "? PEEK(106)*256" then converting it to hexadecimal. This adress will give you the first byte of memory that you can't use anymore, and thus the end of your memory space. In Turbo-Basic this can be easily retrieved by "? HEX$(PEEK(106)*256)". Whatever the result, note it down.

After that, start the composer, load your song package, and select "PACK". Select the number of songs, let it cook, and you'll get the songmodule-length. Subtract this from the value you had before to get your optimum starting adress.

For example: The highest adressable location you got first is $A000, the song-module shows a length of $2000, so your optimum location would be $8000.

Enter the optimum location, and save the module.

Once that's all done, and you want to use the module, first set locaton 106 to the page where your optimised sound starts, devided by 256, then issue a graphics command. In the example above that would be: "POKE 106,128:GRAPHICS 0" Or use any other graphics mode. The only reason for the

graphics mode change is to inform Basic about the lowered memory top, and to free the memory area by moving the graphics data to it's new, lower, location, making the area above it free to use. Make sure your basic program will still function with the lesser memory!

This way works for Atari-Basic, aswell as for Turbo-Basic. (But Turbo-Basic will give you a higher memory location, more memory, and higher overall execution speed.)

19 Using songs in Assembly

Using songs within Assembly, is somewhat easier than using it in basic, because you can use any starting adress, aslong as it goes along with system equates and your program.

Calling the sound routine will work best from the VBlank interrupt. But for good order, a call to it every 50th of a second should do. Before running a tune, it should be initialised anyway, which will also select the song to play.

If a song has samples, the according routine should also be called from the main program. It'll play the digi sounds untill a flag called "STOP" is set to a non-zero value. This flag is located at the module start adress+9. The only thing to keep in mind otherwise, is that the quality of the samples degrade by lessening the CPU load, aswell as the complexity of the display list.

A simple assembly program to call the sound module would be something like this, in BIBO-Assembler:

```
00010 ; Starting adress of the music module = $4000
00020 ; Song #1 has samples!
00030 ;
00040           .or $0600
00050 ;
00060 INIT      LDA #1          ;Initialise music routine, song is=1
00070           JSR MROUT
00080           LDA #VBI        ;Set Vertical Blank Interrupt
00090           STA 548
00100           LDA /VBI
00110           STA 549
00120           LDA #0          ;Stop flag. Clear before running a song.
00130           STA STOP
00140           JSR MROUT+6 ;Play samples (Until STOP is set)
00150           LDA #$E4        ;Reset VBI vector
00160           STA 549
00170           LDA #$62
00180           STA 548
00190           LDA #0          ;Silence tone generators.
00200           JSR MROUT
00210           RTS
00220  ;
00230  VBI      JSR MROUT+3 ;Call music routine
00240           LDA 53279
00250           CMP #6                  ;Start pressed?
00260           BNE .1          ;Local label use
00270           LDA #1
00280           STA STOP
00290  .1       JMP $E462       ;Continue with the original VBI routine.
00300  ;
```

```
00310            .OR $4000
00320  MROUT          .DF "D:SONG.COM"
00330  STOP   = MROUT+9
```

KEEP IN MIND: The player routine makes use of the memory locations in zeropage locations 231 till 255, so dont use them in your machine language program.

Also here, keep in mind that more CPU capacity available, will enhance the music. Also, the less complicated your Display List is, the better it'll work, quality wise.

20 Keys you can use

BMC as initially made to utilise a mouse. But not everyone has one, so we've added some shortcut keys. To use any of these, hold SHIFT + CONTROL and then the key as mentioned.

R	Load song module	W	Write song module
P	Save Digipack	E	Set work-drive number
D	Show Directory	S	Activate song from module
H	Set main tact to 15kHz	I	En/Disable Digi's
1	Main Sequencer window	2	Voice sequencer window
3	Sample Sequence window	4	Instrument editor window
5	Play window	6	Arrange window

Arrow keys: Up/Down controls the arrow functions in several windows.

21 Finally

Now the end of the manual is there, don't panic. You'll be able to control this software with a little bit of practice. We hope you'll have a lot of fun using this high-end music sequencer for the Atari and will only mention now, what it took to make this:

One Atari 800XL with a speedy OS, 256kB ramdisk, Turbo-Freezer, Internal sound sampler (by own design), Atari 1050 diskdrive with Speedy D, Bibo-Dos, Bibo-Assembler, Design Master, Turbo-Basic XL, a multitude of nervous breakdowns, constant nagging from financial supporters, a lot of cola, 20 empty diskettes, 103kB of sourcecode, Amiga and ST mice, Commodoe Amiga with Pagestream (for the manual (Sidenote by Alphasys: Hence I made this file. It's totally illegible/uncopyable)), my monitor, Thorstens monitor, MTV, loads of cables and power supplies, good weather, bad weather, a casio FX-85M calculator, a casio FX-8000G calculator, more than 4 packs of chained paper, an Atari 1029 printer, a Star LC24-200 printer, an Epson LQ400 printer, multiple kilo's of chips (silicon, germanium, and potato), lots and lots of valuable time, one Vertical Blank interrupt, 24 zero-page adresses, one pizza delivery, 2 ears and many, many pencil leads.

Much fun wished by

Sven Tegethoff

Thorsten Winkler

(Translation (from german to english) and PDF with screenshots by Alphasys. Screenshots taken with opusx.msc loaded.)