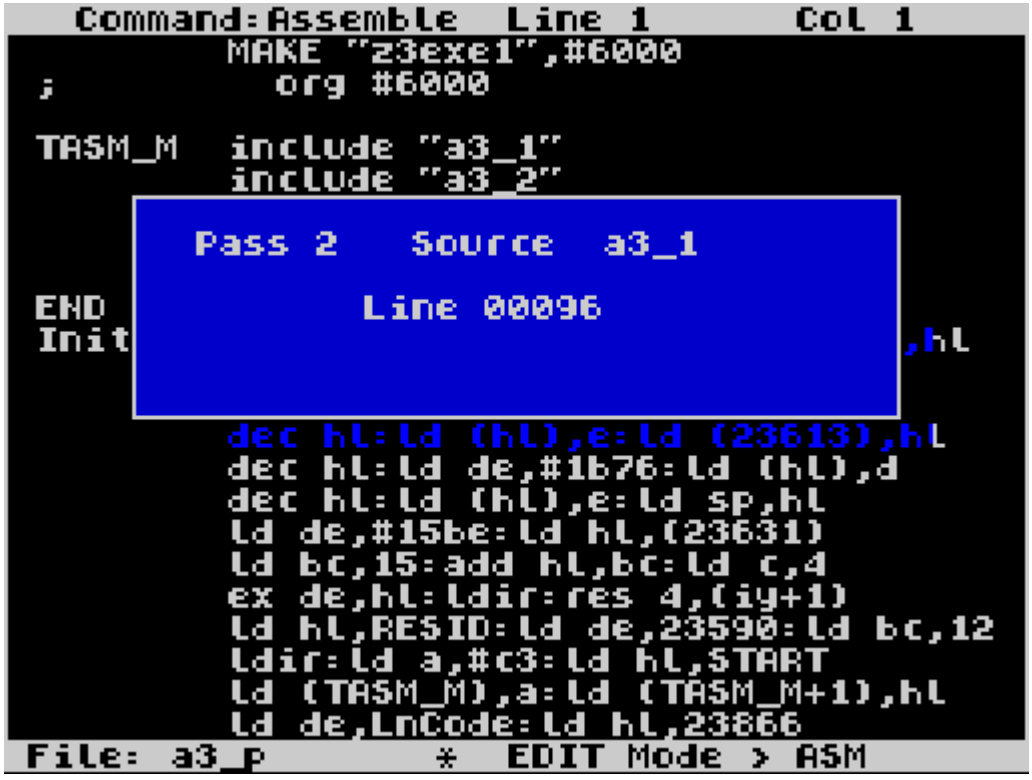
ZX-TURBO ASSEMBLER 3.0 FOR MB02+



Length of works: 14.12.2008-14.01.2009- about 30 intensive days

Date of release: 12.01.2008

Author of modification: Hood, email: hood@znojman.cz

http://hood.speccy.cz

Phone: +420-777-192-191

CONTENTS

FOREWORD OR HOW IT ALL HAPPENED

Esteemed MB02+ users and sympathisers. God and fate wanted it, and you are receiving another piece of my work- MB02+ version of Russian program ZX- TURBO ASSEMBLER 3.0 (further referred to as "zasm"). How it all originated?

All has been caused by +GAMA, to whom I want to give thanks. On JHCON 2008 he was writing in zasm. After I saw the program, I had felt in love with it. And, I wanted to have him on my MB02+ immediately. And as things happen with me, many good things arise from emotion and enthusiasm, zasm was no exception. Ready to work, I began to explore the code, I alarmed the Russian forum, I sought the author, who luckily responded, and after revealing that zasm uses standard trdos calls, I got into remaking zasm for MB02+. The longest and probably hardest thing was to orientate in the code of zasm, this was notably quickened by the author Vladimir Rubcov, who upon my request agreed to open source code of zasm 3.0 to the public. After I oriented myself basically in the code, it was vital to find free space for my routines. It was a bit problem with this, and after I had to move it three times, MB02+ disk operations had slowly started to physically emerge out of my head. From time to time, hard moments had come, but all were finally successfully overcome. A good portion of time was also consumed by finalisation, zasm was functional already in the New Year, but testing, repairing bugs, refining my source code and writing this documentation took up lots of time.

Non- MB02+ users might be ripped off at me, that I did not make disk operations universally from BASIC. This was, however, impossible, as zasm is also working with sectors when loading/saving.

The documentation you are now reading, I have written merely as a basic orientation for complete novices. It should not be considered as a manual for zasm, at all. Generally, I can say, you get used to work with zasm very fast and easy. The first hint for you is EXT+ h for help. Original manual is in Russian language, regrettably. I think a good description you will find in ZX Magazín 3-4/99 written by +Gama- although it is in Czech, very useful for acquainting with zasm. You will find the Russian ReadMe manual in enclosed TAP or MBD image. There should be online version of it on the homepage of zasm http://zasm.by.ru/index.html. You will also see zasm3.10 on this page, which is, I must say, a delicacy for ZX Spectrum user. It means, that the only English text dealing with zasm is this documentation.

ZASM 3.0- ASSEMBLER, EDITOR

Distribution of 128k pages and how ZASM is working with them

It is first to say, that zasm will not work on 48k machines. In ZX Magazín, as well as in Russian manual, there is some info about 128k pages, and I am adding below further ones:

- page 3- buffer or so called pool is placed here. This buffer is exploited by INCLUDE and INSERT functions. You can change its length in the setup. It is strange, that when I maximised the pool and the code to assembly was fairly long, assembly process ended up with NOT ENOUGH MEMORY REPORT. This is, however, also in the original zasm, and you eliminate this simply by lowering pool's length. Right after pool's end, MB02+ disk routines start at f412h.
- page 7- here is by default present STS monitor and debugger, starting at d000h (valid only for version sts 6.22+ distributed and specially modified for zasm), other sts versions start at db00h. From c000h till sts start a second buffer is present, you cannot change its length, and it serves for MAKE function.
- page 0- assembled machine code is placed here
- page 6- source text is placed here
- page 1- zasm code from 8000h- bfffh is place here on quit to BASIC
- page 4- zasm code from 6000h- 7fffh is placed here at c000h do dfffh on quit to BASIC.

How does it work, shifting the content of pages on quit to BASIC? As described above, simply everything from 6000h to bfffh hides into pages 1 and 4. In BASIC then, the page 0 is attached, even with assembled machine code in it.

What ZASM can do- closer look

- the source text begins at 894fh and goes on till ffffh, in page 6, its maximum length is 76b1h/30385 bytes.
- I guess, unlimited length of labels
- line length is 128 characters, displays 40 characters
- commentaries before and after instructions beginning with semicolon
- more instructions on one line, divided with colon (superb thing)
- import/export of txt files (another superb thing)
- uses graphical characters
- possibility to change fonts
- hot keys
- conditional assembly, libraries- IFDEF, IFNDEF, IFUSED, IFNUSED, ELSE ENDIF
- INSERT, INCLUDE functions- written directly in the text and both are executed on assembly. You can input a file of desired name. Again, a superb thing. INCLUDE inputs another zasm source text, INSERT inputs a binary file. If you

would find 30kB of source text insufficient, you must realise, that using both of these commands can extend your source text practically without limits. INSERT function can also work as file linker on MB02+ version, if you input:

MAKE“target“,address

INSERT“file1“

INSERT“file2“

The file “target” consists of file1+2. But only files up to total target 65kB can be linked in this way. It is simply because I designed it like that, a trial to link into bigger file will fail. Even though, it could be done to save and link into bigger files, but it would consume a lot o disk space and also primarily, this function serves for zasm, not for MB02+ linkage. Look into re-work of zasm chapter (MAKE function) for better understanding.

What ZASM cannot do, or comparison with Prometheus assembler

I picked Prometheus, the Czech assembler for comparison, because I know it and use it. Some differences emerge already from the previous chapter. Generally, we can say operating in zasm resembles Prometheus. I re-worked PageUp/Down keys the same as in Prometheus. Zasm is, indeed, more comfortable, it uses a tabulator, you can place instructions anywhere on the line.

A better thing about Prometheus is, that it checks syntax right after entering the instructions. Whereas zasm checks the syntax only on assembly, both programs assemble in two runs.

What a user can really miss in zasm is the list of all labels. You can forget smooth alphabetically ordered list of labels, like in Prometheus, it is only in version 3.10.

What I have been missing right away was Prometheus’s PUT function, that is assembly of machine code with another address, than it physically lies in RAM and in Prometheus 128k version also assembly into 128k pages exists, but not in zasm. This is however solved by zasm differently. Do not use ORG but MAKE”name”,address command, which you write straight away into the source text. After assembly starts, something like Prometehus’s PUT is carried out, but this time into a file on your disk. If you want to assembly into pages, you have to write your own code for that.

In the contrary, zasm has SAVE OBJ function- save of assembled machine code onto a disk, starting on the address in ORG. Beware! A safe zone for the user is 6000h and above- in this range of memory, everything is backed up and on this address also code of zasm starts. The zone below 6000h is not protected by zasm, and if you place some of your routines here, using ORG, very easily, you can rewrite e.g. BASIC system variables area, etc. Remember that! At the same time, you cannot assemble to VRAM. The first address which you will be successful with, will be 5b00h/23296. Here and from here you can load/save your code. Assemble into VRAM or ROM is however solved with MAKE function.

What else cannot be done in the zasm (and it is rather a domain of sts monitor), is disassembly of machine code back to source code. This is really a shame.

MB02+ version specialities

MB02+ version of zasm is a full version with all functions open and also working same way as the original for betadisk. The only thing I did not bother about (literally) is DISK CATALOGUE and DISK SELECTION. Well, disk catalogue is present but the files are not neatly organised into the window as in the original, but is the style, like you know from MB02+. Disk selection is not present at all. Both functions were not re-worked due to lack of time. If I had re-worked them, I might have been sitting on it still now. It is several years ago, when I did remake of two ProTracker 3 versions for MB02+ and I did catalogue and disk selection same as in the original, but it was a very exhausting job. To put it short, not only for this reason, I implemented catalogue and disk selection in my nmi menu for MB02+, so give it a go and use it also in zasm.

Also, zasm had to be re-worked to be able to work with 1024b sectors instead of betadisk 256b.

I corrected a minor mistake. Already mentioned buffer in the page 7 is 6912 bytes long in the original version. It means, that once MAKE function is activated, it rewrites the beginning of sts 6.22+ in the memory. I lowered buffer to 4096 bytes, which means the user can use also sts 6.22+.

Zasm can load practically any monitor/debugger (ideally sts, but I also tried Devastace monitor and it is working) on the fixed position address. But sts 6.22+ is a bit bigger and loads lower than other stses. Sts 6.22+ is by default loaded at the start of zasm in the original as well as in my version, but once deleted, it cannot be loaded back. I fixed this minor mistake. So, MB02+ version can load even sts 6.22+.

STS- MONITOR, DEBUGGER

Yeees, wanted by all non-betadisk users, a myth, a legendary sts has fallen to us form the heaven. Alas! Do not rejoice, dear MB02+ user! No MB02+ operations were implemented by me. Please, count it with me, sts uses the following disk functions: catalogue, help, load/save file, load/save sector, setup. What is necessary out of this list, indeed? Catalogue, load/save file, right, my nmi menu can do these functions. Help is included in the TAP and you can read it in zasm, setup- hopefully, we can live without that thing. Maybe, the good and useful thing would be load/save sector, but in the original, there is no input field for side selection. How should this be solved on MB02+? I have evaluated relevance (better to say irrelevance) of these disk functions, and I said to myself, that re-working of sts for MB02+, I would loose a lot of sweat and maybe also blood, and I am not going to do so, at the moment. I had prepared at least help load from sts, but one routine had kept falling down, so now after zasm is loaded, and also in the TAP file, you have sts 6.22+mb version with all betadisk inputs blinded. It is because, if you would use non-modified sts and press any betadisk service key just by chance, the program could freeze, and you work would go to abyss. The only thing I do regret, is that I had to blind also G key (Z80 tacts counter), which also touches very low routines. I hope, once I will have a chance and time to open this function.

ATTENTION!!! On the disk, or TAP, there are also another sts versions. None of these is modified to work with MB02+ and most probably they will freeze on pressing a betadisk operation key.

MB02+ VERSION- MODIFICATIONS

The following part is aimed for those of you, who will want to remake zasm also for their disk system. You will find here what really is to be re-worked and also principles of my routines. We go function after function, address after address, byte by byte:) In addition to new routines, zasm code had to be modified on some places also. For detailed examination I advise you to consult my source code, where individual functions are separated in a clear way.

In general, every jumps to my routines are directed to one point, where the routine decides according to service number in register c, what function it is, and what to do and where to jump. As I have mentioned, my code is placed right behind the pool in page 3 on f412h. CALL or JP to trdos was replaced by call or jump to 8438h, where page 3 is paged in, and jump to f412h comes after that. Then, backup of approximately 1000 bytes from 7000h is carried out, and my code is moved down, instead. It is placed intentionally down, because many disk operations are working with pages. After an operation is done, everything is then returned to its place. Regrettably, From 6000h till the source text start is not much of free space, I found only about 15 bytes, and the rest had to be placed on the stack, which is bringing us to the question...

WHERE DO I FIND FREE SPACE FOR MY ROUTINES IN ZASM?

A lot of place can be found in pages and you may achieve this just by lowering of maximum length of some of buffers. However, down below you need to have your own paging routines for 128k. The only zone is 6000h- 894eh. Above, the source code begins, and below no protection against assembly is applied. There are 11 bytes starting with 8438h, that is the routine which quits to trdos. A very clever trick brought forth by Velesoft- before zasm starts, it is necessary to rewrite 4 places, where starting SP is initialised, to lower numbers and thus a space for your use is ensued. In this way, I cut off about 15 bytes for my purpose. If such a trick would not exist, the code in lower memory is so dense, that I would probably had to use SRAM in MB02+, otherwise, I would not have found any spare place.

HOW TO CHANGE SECTORS, IF TRDOS USES A LENGTH OF 256 BYTES FOR A SECTOR?

This is an essential thing and you must solve it as first, otherwise no modification is possible. You must force zasm to work with sector length of your disk system. Basically, you have to limit both buffers to a whole multiple of your sector length. For INCLUDE/INSERT pool you have to call your own routine on c947h which will count trdos sectors into whole

multiples of your disk system sectors. Limitation of the second buffer are placed on 65d9h and 6619h in hl register pair.

LOAD/MERGE

both functions use the same path. The call itself is on 85e5h. It is a service number eh- load a file, hl and de contain start and length. Easy.

SAVE/SAVE SETUP/SAVE BLOCK

again, the same path for all. On 84cbh there is file erase, you can omit this one if you do not want to erase a file and 84d6h there is save itself, service number bh. Again hl, de registers and this time also an important address 5cddh- file names and disk system identification are placed or taken from here. Otherwise, easy.

SAVE OBJ

A bit harder nut to crack. In principle, this function saves assembled code from 5b00h or above, on the disk. (How is zasm dealing a problem with SP is nothing I know of, do not ask me, if I had inspected that, I would have been reworking zasm still now). What you have to do, is to place your saving routine in a way, so that it ends up on 57ffh. Well, this is, what you find in the original, whose save routine has about 15 bytes and is moved to 57f1h. I am almost sure you can go even higher, because assembly is probably not done into attributes part of VRAM. And in the contrary, I myself put the start of my saving routine a bit lower than the original has it (on 57eah). But this is the principle. Many ldirs is awaiting you, not a pleasant function to remake.

LOAD FONT

Ultra easy. Font loads on a fixed d6e7h to page 4 and has also a fixed length of 800h bytes. The routine is on ca19h.

LOAD STS

Easy. Stses or other debuggers are loaded on a fixed place db00h into page 7. In my routines, there is a modification, that manages to load even sts 6.22+, which begins already on d000h. You will find this load on 891bh.

INCLUDE/INSERT

So, we are going to deal with sectors, now. Both functions are reading sectors from the disk and modification is a bit harder. Reading from the disk into memory is going on in page 3 in the buffer, its length can be set in the setup. On 66e3h it is found, whether the file exists on the disk, on 66aeh service number 8 is going on, I omit this one, and sector loading is on

6718h, where on the input we have in reg. b number of trdos sectors and start in hl (i.e. always pool beginning). You just entrap this routine to find out how many sectors to load from the given file, and that's it.

MAKE

Alongside with SAVE OBJ this is the hardest part of your remake. It is about saving of sectors from buffer in page 7 to a disk. Here, trdos is called 5 times, on MB02+ we can do with only 3 times. A harsh problem on MB02+ is, that it cannot save into a file of an unknown length. At first, I wanted to force zasm to assemble twice, and first run would only return final file length without saving, however zasm resisted that attempt of mine (regardless of time consumption for the user of this solution) and so, Velesoft, and also independently on him Busy:) thought out a method which I finally used. MB02+ version of zasm is most probably the first MB02+ software which uses backup file. The principle is totally simple. A "zasm3.0bak" file of 65535 bytes length must be first created, MAKE function is saving into this bak file, and after the last save is executed, we already know the final length of the file and we may now create a new file of proper length and copy a content of bak file into it. Than bak file can be erased or left alone, the program creates it, or saves again into it once MAKE function is activated. Easy, isn't it? But before you invent it...65b2h (omitted, trdos opens the file), 65b9h finds out free space on the disk, if free space is ok, bak file is created, unless it already exists, 6618h saving sectors into bak file, 6664h moving bak contents into a new file with the correct length, 6676h (omitted, trdos closes the file).

DISK CATALOGUE

Do not ask me, how to modify zasm to display the catalogue into its window. I do not know that. I simply bypassed this with bsdos catalogue service. It resulted into a lot of free space in the original trdos catalogue routine in page 4, and I use this space for my own needs. The catalogue starts on c6b7h, and I left the first call unchanged, than ld c,64h follows and jump to my central routine. I put this value into c intentionally, so that I can easily find out it is catalogue, we are dealing with.

ERASE FILE

mega easy. Bsdos has a service for that, the name is taken again from 5cddh, routine starts on c8c5 in page 4.

CHANGES IN THE ZASM CODE

A list of addresses follows with changes, so that the program works correctly on MB02+ and cooperates with my own routines (the list especially for me:)

LOAD, MERGE- 85e5h call 8438h

LOAD FONT- page. 4- ca19h call 8438h

LOAD STS- 891bh call 8438h

SAVE/SETUP/BLOCK- 84cbh 3xnop, 84d6h call 8438h

SAVE OBJ- 8564h 3xnop, 859ah jp 8438h, 853fh ld c,101: call 8438h- handles free space on the disk

- 8578h, 8583h ld de,57f1h change to 57eah

- 8598 ld a,13h: ld bc,7ffdh: xxxxx, on xxxx my code ldirs the following: out (c),a: ld c,bh: jp f412h

MAKE- 65b2h 3xnop, 6664h call 8438h, 6676h ret, 6618h jp 8438h, 65b9h call 8438h,

65d9h, 661dh- ld hl,sector multiply- changes MAKE buffer length

INCLUDE/INSERT- 66aeh 3xnop, 6718h call 8438h, 66a3h call 8438h- finding out if the file exists

CALL TRDOS- 8436h jr 841eh- redirect quit to trdos to BASIC

ERASE FILE- c8c5h jp 8438h

CATALOGUE- c6b7h call cb00h: ld c,100: call 8438h: xor a: ret page 4, c6d6h- c7ceh is probably free space, in case you cancel trdos way of catalogue display

If you erase the screen, the original displays badly left and right vertical lines, this will fix the problem:

- c3ceh jp c6c1 i.e. right after ret, page 4, place this one: ld (hl),b: inc h: ld (hl),b: inc h: ld (hl),d: jp c3d1h

- c3d8h jp c6c9h, page 4, place this one: ld (hl),d: inc h: ld (hl),b: exx: ret

To carry out displaying of 10 characters by assembly INCUDE/INSERT functions, the following 4 things have to be done:

- cf79h 5 ->3, page 4
- 7788h 1bh ->19h
- 77b7h 2xnop instead of ldir
- 7795h rewrite to „Memory“,6,4,0- where 6 stands for a spacer, and after that the number of spaces follows, but the place is maximum for 9 characters!!

swapping of PgUp/PgDwn keys- 6aedh 68h, 6aef 7dh

extends INPUT of the name to 10 characters- c8a4h ld c,10, page 4, 67b8h ld b,10

POOL- 894d 18h, the routine count 1024 multiples, essential for the change of pool's length- c947h call c6d6h- page 4, c950h cp 35h- page 4, limits maximum size of the pool

inputs the length in the head- omit it, it rewrites 10th character of the name - 6635h 3xnop

modification of display of the information line, page 4:

- cef3h „ File:“

- 6926h 3xnop- displays trdos drive, and we do not want this

- c8cbh ld de, cefah

- c8cfh 8 -> 10, c893h ret, c894-9h free, page 4- displays 10 characters of the name down in the information line on the screen

modification and repair of SetUp3.0 name to 10 characters, page 4:

- ca41h leave ld hl,5ce6: ld de.... and write jp c6ceh, insert ld (hl),32: ld hl,892fh: jp 84c7h

- ccc0h 20h

cancellation of trdos disk drive selection, page 4:

cb88h ret- a lot of free space follows, until PUTPoint

creating a free space on the SP for your own routines before zasm start:

- 69d6h, 6942h, 77ddh, 697dh- these are the instructions, where the starting SP is handled, originally, here is 61c4h, I rewrote it on 61a8h, and I place here 128k paging routines. They look like push af: ld a, page_number: jp page, see my source code.

The main entry point into my routines is on 8438h:

call 61abh, page 3

jp f412h, jump into page 3

push af, in this place, my routines are giving control back to zasm

jp 6511h and a page is paged, which I set in a register

CHANGES IN STS6.22+MB

In sts the following changes have been made:

e7f8h catalogue (ret), e7b0h help (loads on d000h), setup, key G (ret), e66eh load of a file (ret), e60ah save of a file (ret), e301h load of a sector (ret), e2fch save of a sector (ret), e465h drive selection (ret), eb99h 28h, eb9ch 1bh- swaps PgUp/PgDwn, f6a1h- to defdh ld a,0, to df02h ld de,0- quit to DOS- blind flange

ACKNOWLEDGEMENTS

- Vladimir Rubcov (RubtsOFF): the author of zasm- for consultations and publication of zasm3.0 source codes
- guys from zx.pk.ru conference
- Velesoft for testing, hints at finalisation and sending me two routines with shifts, and for idea of bak file
- +GAMA for hints and for showing me zasm
- Busy for hints+ in duplication for idea of bak file, too:)