issue 93 - June-December 2000

MSX Computer & Club Webmagazine

About MCCW

MCCW is the first MSX webzine. It is made by volunteers. It is the continuation of MSX Computer & Club Magazine. This is the last edition that appeared, due to a lack of articles and time.

Editor in chief

Manuel Bilderbeek

Co-Editor in chief

Maarten ter Huurne

Technical stuff

Maarten ter Huurne, Manuel Bilderbeek

Layout

MCCM (Frank H. Druijff *et al*), Manuel Bilderbeek, Maarten ter Huurne, Arjan Steenbergen, Laurens Holst, Albert Beevendorp

Permanent editors

Manuel Bilderbeek, Albert Beevendorp, Laurens Holst, Antti Silvast, Maarten van Strien, Bas Wijnen, Wammes Witkop, Frank H. Druijff

Other contributors to this issue

David Heremans, Pierre Gielen, Mauricio Braga, Alex Wulms, Robert Wilting, Saku Taipale, Nestor Soriano Vilchez, Jorrith Schaap

Thanks to

Maarten ter Huurne, Geert Josten & Egon Willighagen, Frank H. Druijff, Wammes Witkop, Collin van Ginkel, Albert Beevendorp

Handing in material

You can't send it material anymore. We advise you to send any material you might have to The MSX Resource Center.

Preface 93

All Good Things...

It has taken a long time, but finally edition 93 of MCCW is now finished: I am writing this in February 2004, about four years after the previous edition was released. The biggest part of this preface will be about the reasons for this, what happened during the silence — this seems to be partly a text about history! — and why it will be the very last issue. I apologise on beforehand if the text is a bit messy; it is composed of many little parts that I wrote during several separate days.

The lack of articles I mentioned in the preface of last edition was worse than we expected. But moreover, Maarten and I became very busy after releasing 92. For example, I finished my work on the university and finally got my Master's degree in April 2001. After that I got a job at Océ and since it was already a long time ago, I lost more and more the spirit to work on MCCW.

I regret very much that this happened. When I got the approval of Frank H. Druijff and Wammes Witkop to start this Webmagazine, the conditions were quality and continuity. In my opinion, the quality issue has been satisfied. However, we failed the continuity issue. Of course I did not know if I could really guarantee continuity, that is why I wrote in the preface of issue 91 that we had started MCCW as an experiment. And that we would stop if there would not be enough articles to guarantee continuity.

Do not forget that making MCCW is an incredible amount of work! Maarten is actually the only person I could find that 1) is critical enough to be an editor 2) knows enough about MSX 3) can write good enough in English. In other words: he and I are the only ones who could edit and review the articles we received. This means that we both had to do an incredible amount of work to get the articles on an acceptable level. And we set our standards high.

As you probably have understood by now: we do not have a workable situation. This has led to the very long delay before we released this issue of MCCW. But it also means we cannot continue with MCCW, unfortunately enough. We just do not have the time and spirit to do all that work...

In last issue's preface and even in the articles in this issue a lot of things are promised: more reviews, more episodes of series and so on... They will not be there, I am afraid. In this issue we do not even have Antti Silvast's series continued. Antti told us shortly after the release of last issue that he lost interest in MSX, mainly because of personal reasons.

One of the things I promised for issue 93 was a full report of MSX Den-Yu Land 2000. This fair was held in Tokyo at the time I was doing an internship at Hitachi-Maxell, located in a village near Tokyo. Also that article will never be written, unfortunately. But I can give you a brief summary of what I remember of it here. It was actually a relatively small fair, held in just one large room. At the fair entrance I got a programme booklet, which I saw only a few weeks later in an auction on Yahoo Japan for 1000 yen! It started with a talk by Mr. Kazuhiko Nishi There were a few hundred people in the room. I hope because of the very high temperature some people fell asleep during Nishi's talk... Anyway, he was talking about the one chip MSX and the revival, similar to the talk he held in Tilburg in 2001, I guess. The fair itself was like in Dutch fairs: many tables with people behind them selling and demonstrating stuff. I saw cool MSX T-shirts, NV-Magazine sets, music cd's, DM System 2 books with a LaTeXed layout exactly like the MSX Technical Databook, Mr. Tsujikawa's MSX-on-FPGA design, S.T.A.R. of Matra selling Realms of Adventure and a lot more. At the end there was also a talk of Mr. Yamashita, one of the MSX designers. When he was finished, Mr. Yokoi told us that there had been about 800 visitors that day, if I remember correctly. I had a wonderful day! And now that I am writing about this anyway, I would like to thank my Maxell-colleagues Mr. Inaba and Mr. Ido for joining me, Bernard Lamers for his excellent company, Mr. Suzukawa for his on-line translation during Mr. Yamashita's talk and Mr. Ikeda for his good company.

Of course the visit to MSX Den-Yu Land was not the reason I went to Japan. As Maarten said in last issue's preface, I did not have much time for MCCW during my internship. The working hours were practically from 8 to 9 and I only had access to the Internet at work. I also did not have any other way to communicate than mail. Maarten said he would do most of the editorial work for 93, but he never got to it.

So, what happened during that time and after that? I am browsing through some e-mails now to give you an impression. Collin van Ginkel proposed to start a series of articles about drawing on MSX, dubbed “Pixel Talk”... Unfortunately we have never seen an article. No offence, Collin. The deadline for number 93 was set to June 30, by myself... Oops. Bas Wijnen was planning some articles about GFX9000, but he blew up his own GFX9000 cartridge... Joynet is also something he wanted to write about, as well as several other things, like schedulers; unfortunately he seems to have never got to that. Albert Beevendorp was planning a series about assembly (Assembly Line), the first two parts were supposed to be about calculations. The unfinished first article is in this final issue. Sander Zuidema also offered to write some reviews, starting with Kyokugen and even Moonlight Saga, together with Rieks Warendorp Torringa. The Kyogugen review you can find in this final issue is the first draft he sent us. Laurens Holst was going to write a review of Compass, which was supposed to be released in Bussum 2000. However, the program was still not finished, so it was going to be a preview instead... Also, we wanted to publish interviews with well-known MSX users, finding out what happened to them and how they looked at their own MSX time. Also for this series Sander Zuidema offered to write some articles. I have a very preliminary version of the first article of ‘What has become of..’ or ‘Hoe is het nu met...’, which is about Knightram, one of the ANMA guys. It never became a full article. Lastly, I also found a draft version of a review of Sand Stone, a game of Compjoetania TNG. It is written in Dutch, by Alex Ganzeveld. He recommends the game...

Apart from all these plans, we also received some articles. Most authors of them clearly did not have any experience with writing articles. This means we had to communicate a lot about the articles: tell them what should be improved and getting a new version, proofread again, mailing them again with comment, etc. This is a terrible lot of work, I can tell you. Some articles needed to be half rewritten before they were publishable... We did try to compensate this by attracting a native English speaker to do some editorial work. But this never really became a success: communication was not so smooth with him and I still found quite some errors in the articles that he reviewed. Note that Frank Druijff already warned me about this in his epilogue text of MCCW 91... He was so right.

The few months after I got back, which is around the end of 2000 and the beginning of 2001, Maarten did a lot of work to make the XSL — the style sheet file which is used to generate HTML from the XML that is written by the authors — a lot better. I am still supposed to make some adjustments to our edit guidelines for that. After that, Maarten got really very busy, since he had to work on finishing his Master’s work, while having a job at the same time...

After one year had passed since issue 92, on July 30 2001 9:04, I sent an e-mail to our private mailing list... Here is the contents:
Hi there MCCW people,

I’m sensing a problem. We haven’t published since one year now...

What’s the cause of this?
1) We don’t have people that have time for editorial work (reading incoming articles, giving comment on them, improving the language and style). Especially Maarten used to do this and I also did a part of it (but Maarten is far better at it!). However, it’s not the reason why Maarten joined MCCW, so he got ‘tired’ of it.
2) As the editor in chief, I have a problem since I am not at the university anymore, where I could process MCCW. This will be fixed as soon as I have a pc at home. I’m planning to have one in a couple of weeks. But it means I have far less time to work on MCCW
3) We don’t get enough articles. This could be caused by the fact that we don’t have so many releases... But still, it’s good to have some sort of stock of articles, and at the moment we hardly have anything like that.
4) If this all means we have to quit (which would be a terrible pity), I at least want to publish the (almost) finished articles. About the Maiskoek/Bits and Pieces: lots of those messages are obsolete. I will at least make a note in the preface and in the Maiskoek/B&P itself about this.

So, what do we need? We need editorial staff!
A good editor (like Maarten) has the following ‘properties’:
- is extremely critical (to get a high standard) about style and language (and content) therefore having a lot of knowledge about language (mainly English), how to write an article and about MSX in general
- is well reachable (as editor-in-chief I need good communication with the other editor(s)...)
- has (some) time to do things! (I understand it’s a hobby-thing though...)
- and of course some motivation is necessary.

I think Maarten still has time for the technical support regarding the production of MCCW. Please confirm, Maarten.
Also, Maarten has the tooling to produce MCCW at the moment, in case this would be necessary (i.e. when I don't have my pc yet).

Please share your thoughts about this with me.
This summarises my feelings quite well. And nothing changed after I sent this mail, except that I got my pc at the end of 2001 and made the tools work on it — in March 2002 — so that I could at least generate the MCCW on it. That is all...

Another interesting thing is that we got about a dozen mails of people that just wanted to sell their MSX or wanted to tell us how great they think MSX is. For the former category, we mostly proposed to make an I/O text about them for the Bits and Pieces page...

Luckily, there is still a web site on which one can publish articles: The MSX Resource Center. This site has become the biggest MSX Community site in the world; at least in the non-Japanese world. Please take a look there and send articles to them if you still have some laying around.

So, this is the very final issue of MCCW. All good things come to an end. To be honest, this issue is not of the quality we are trying to make. But, we felt that the articles that we have received that are sort of publishable should be published. We did not want the work of the authors to be done for nothing. However, since we do not have so much time these days, we just made sure that the articles had a minimum quality level, just enough to publish them. Some articles clearly were not — completely — finished yet. Sometimes I put a note with some explanation under those articles, for clarity. The bright side is: there are more articles than ever in one issue! Have fun with it.

Okay, the last thing to say here: thanks to everyone who made MCCW possible!

Manuel Bilderbeek

part 3 - strings and sound design

The wolf and the seven notes

When designing a new sound, perhaps the biggest challenge of all models is FM. A question from one of the readers about that is the central point in this article.

Maarten van Strien

Directory

On my request for questions in the first part of this course, I got the following response from Frederik Boelens:

1. How is it possible to make nice strings with FM voices? I used AM voices so far, but the volume is not adjustable with AM voices in a nice way.
2. How is it possible to keep the music interesting from the start to the end? I used to do a style change, but is it possible to continue with the same style?

Answer 1

Whatever people say about strings: what *you* mean are pads. Strings can be divided in first violins, second violins, violas, celli and double basses. As we use the synthetic FM model, we are talking about pads. In the 80’s people used the old analogue synths — JP-8, Juno 106 etc. — to replace real strings, because the band’s budget usually was not enough to hire 30 real string players, paying $500 per player per day. From that moment, the average synth owner sees the string pad to be real strings, but that is just side-information. We can split your question into two parts, pads for FM-PAC or pads for Music-Module. A nice simple solution for the Music-Module would be by just selecting four ‘strings’, use three channels to build the chord and use the fourth channel to detune the top-line of the chord. A detune value of +1 is good. You could of course also fatten-up all three channels with a detune, but this way you are spending six of nine channels for the chord alone! So you have four channels, do not set them too loud compared to the rest of the channels. Make sure the voice has vibrato of its own and in the patterns also use vibrato, called modulation in Moonblaster. These vibratos — `MOD` — should not be at the same row. See the examples:

example with detune

vol	52	52	52	52
det	+1	+0	+0	+0
1	D 5	D 5	C 5	G 4
2	.	.	.	MOD
3	.	.	MOD	.
4	.	MOD	.	.
5	MOD	.	.	.

example detune and echo

vol	48	52	52	52
det	+1	+0	+0	+0
1	.	D 5	C 5	G 4
2	.	.	.	MOD
3	D 5	.	MOD	.
4	.	MOD	.	.
5	MOD	.	.	.

Your pad-instrument should have the following properties:

- vibrato
- bright sound
- no attack on the modulator
- no attack on the carrier
- longest release on the release of the modulator
- average-long release on the carrier
- no decays on the modulator as well as the carrier: sustain to the max!
- a 1:1 FM-ratio is generally good

Set the modulation-depth in the pattern-editor to 3. Most Muzax 3 tunes were made by using this scheme!

FM-PAC 6 channel mode

It is another story for the FM-PAC using drum mode. Because you have only six channels, wasting four channels for the chord alone is a bit overcooked. It depends on the situation actually: if your idea was to use one bass, one melody and one echo for the melody, then there are three channels left for chords. You could spend three channels for pads, but then without a detuned channel. And again, not too loud: volume 10, 11 or 12. You could also use two channels, so you have a channel left for a detune and write your melody in a way it fills-up your chord too. We are lucky to have an FM-PAC, which sounds a little fatter than the Music-Module, so using only two channels with the hardware-violin does a nice job here also!

Xak 3 has nice string-simulations used for melody. A nice trick is the following. Your melody consists of two hardware violins, detuned respectively as +1 and -1 and with volume 13 and 12. Make your melody use both channels. Now the trick: when you want a musical note-off, put a `SUS` command on the loudest channel. The quieter channel continues and makes a reverb effect. Since the Music Module does not support the `SUS` command, make sure that the Music-Module-instrument has an amount of release in the — carrier —envelope. For FM-PAC you have four channels left for your usual stuff. *If* you use this method, make sure the rest of your channels are ‘filled-up’ pretty well. In case of a solo, better use conventional echo channels instead. An easy way of filling-up your sound, for use with the method above, is by using a harpsichord somewhere.

Answer 2

To make an interesting piece of music of about three minutes is a true puzzle on MSX. Again I am basically talking about sound design, almost boring by now, but sound design is a major basic skill! Sounds on MSX, whether they come from FM-PAC or Music Module, only have little 'evolution' when speaking of sound-colour. Avoid long notes without sound-evolution, unless you are using pad-sounds to make chords. A clear story about melody is a major-sized article, going to the next issue of MCCW, but until that moment here are some quick hints:

- keep the melody simple
- let a friend hear your piece once or twice, if that person can whistle the melody afterwards your melody is good. If that person does not know anymore there's something wrong.
- you can re-use pieces of your basic melody for new themes. The advantage of this is that you already know your melody, this makes your music easy to pick up for others too.
- repeat something = recognise something = structure!
- pay attention to the rhythmic structure! People pick up rhythmic structures more easily than a bunch of different pitches, for an example see the end of this article! Maybe you could listen to Celtic music sometime... a simple rhythmic structure and interesting themes.
- lower the melody an octave, change instrument, and lower the volume. Now make a new melody to this as theme. This method is a part of a technique called counterpoint, a nice subject for a future article.

Fool people with sound design!

As you noticed, I keep repeating the words 'sound design'... designing a sound. What is an instrument actually? Here are some elements. A sound contains:

- pitch
- volume
- sound-colour
- location (when used in an ensemble)
- play techniques

The pitch of an acoustic instrument is not exactly constant... there is always a difference. A lot of instruments have their own out-of-tune level, however, you do not hear this in a full orchestra. When you simulate such an instrument with a synth-model — such as FM — keep this out-of-tune level in mind. A thriller on an oboe rarely — actually never — sounds correct, compared with a thriller you use from a synth. This makes real instruments sound real and a synth sound static.

Volume is important also. Volume — music-technology freaks call it amplitude — when using an FM-model and when used in our trackers, has little to do with real acoustic volume. Our FM volume is rather an amplifier setting instead, since low volumes sound about the same as loud volumes. Acoustic instruments do sound differently when played soft or loud. A crescendo is more than only the volume getting louder, the whole sound-colour evolves too!

Sound-colour is one of the most important elements of an instrument definition. It is the sound-colour you recognise of an instrument, and to start with the good news here, FM is rich of colours, very rich! Even the simplest FM — simple-FM — has a rich amount of colours. When using complex FM — two operator FM with feedback or three or more operator FM — you get a lot more colours. Some instruments can be simulated surprisingly well with FM. Unfortunately, real instruments have small fluctuations in the colour-spectrum, so that a 100% simulation with our FM will not work.

Location is not really an issue on MSX, perhaps on OPL4... when I get questions about this I will come back to it some day...

Play techniques are as important as sound-colour. If a sound is produced 100% perfectly, you can still ruin your result if your play technique fails. A well-known example is the guitar. When you analyse what you hear then you will notice that you will hear a lot of noise and 'mistakes', like fret noise, nails that hit the strings etc. If you make perfect guitar-string samples, and you do not sample all those mistakes too, you can imagine that the result is bad. If you try to simulate a wind instrument such as an oboe or bassoon, you *can* get nice results as long as you take care of all the play techniques/effects those instruments have. All these aspects belong to the profession of a sound designer. A piano like that of the OPL4, not really bad, fails in a tracker notation like that of Moonblaster. Basically because you cannot adjust the volume on the same step as the note. You do have to pay attention to get the piano sound like a real piano; if you have to use eight channels for this... then you have to.

The 80's

The DX7 was the major success of the 80's. Every band of some proportion owned a DX7. In those years, synths did not have built-in effect processors. And the effect processors in fact do a major job on FM, especially when using chorus and reverb. For instance, make an instrument and try to route the output through an effect machine! Or sample the instrument into your pc and use software like Soundforge or Cooledit to add some reverb. You will be surprised about the result because suddenly all instruments sound cool and fat. And suddenly you will recognise all those sounds you heard in the 80's, indeed that was FM too! It is pretty strange it sounds that cool, a piano with reverb sounds 'normal' but the same reverb added to an FM voice is very special and new. Probably this is because you never heard your FM-PAC sound like this in all those years... it is strange to hear it like this. FM in fact is a cool synthesis model... be proud of it! It is an excellent fundamental for your carrier as soundaholic. Whatever people say: remember that FM is cool!

Finally

I promised an experiment! You can do this alone, but together it is more fun! I was talking about the fact that people recognise a rhythmic structure a lot easier than a melodic structure. Do the following: clap you hands twice and let another person clap the third one. You will notice that the timing is correct. The time between the first two claps is obviously that clear that anyone can define when the next clap should come. This can be very fast, but also very slow, the experiment will show how slow it can be. Slower than you think! Next time a theoretic article about analysis.

Questions? E-mail them to support these articles!

A short note from the editors:

Of course there is no need to e-mail anymore, because the articles will not be continued.

The Island

Pentaro Odyssey 2

At the Tilburg 2000 fair, Sunrise did not only release Realms Of Adventure, but also this platform game. It is the sequel to Pentaro Odyssey the Revenge, which was released three years earlier at Tilburg 1997.

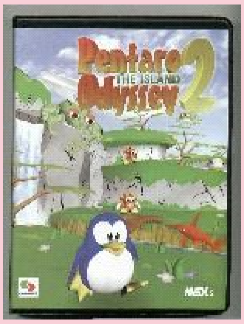
Robert Wilting

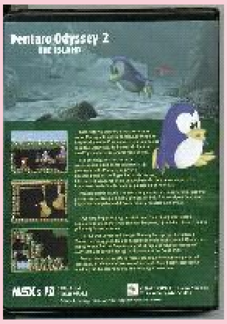
Directory

Ordering information

- Required: MSX2, 2DD, 128kB RAM
- Sold by Sunrise for MSX
- Price: EUR 11.35 for members of the game subscription. Others will also pay for the extra shipping costs. Please contact Sunrise for your exact price.
- The game can also be ordered by Hnostar where it will cost 2000 pesetas plus shipping costs. For Hispasoft members there is a special price.
- For more information look at:
    - Sunrise homepage
    - Hnostar homepage

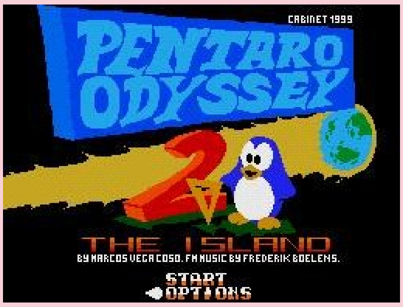
Pentaro Odyssey 2 comes in a nice box made by Club Hnostar, which is in full colour. On the back are some screen shots and you can find the story in Spanish and in English. It contains a nice manual and one game disk with a standard Sunrise coloured quality label. The manual has a coloured cover and nicely printed pages in the quality you expect from Hnostar. The manual is in Spanish and English just like the text on the back. This is a bit strange for me: after seeing the Manual of Realms of Adventure I would have expected also a Dutch version. Not that it is a big problem, because Pentaro Odyssey is a game that is easy to understand. There are some nice black and white screen shots and drawings in the manual to make it all complete.


Front of box


Back of box

The story

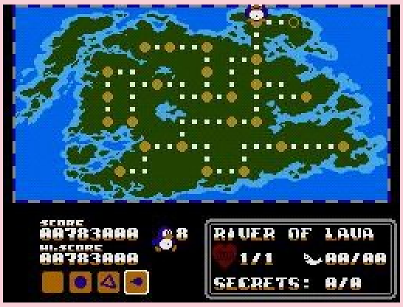
Long ago, the evil emperor Penguin happened to meet one of his subjects, Lucytesse — beauty — and he fell madly in love with her. His love was unrequited, though. She only had eyes for her beloved Pentaro. They were a happy couple but the emperor, in a fit of rage and jealousy, kidnapped Lucytesse, and carried her off to his palace on the South Pole. Pentaro went after it and rescued Lucytesse. They lived happily together for a while, far from the realms of the emperor. Nevertheless, one day the emperor took his revenge and kidnapped Pentaro this time. The emperor locked Pentaro up on an island far away from the South Pole. After Pentaro escaped from the Emperor in 'Pentaro Odyssey the Revenge', he needs to get off the island. Here is where you come in. Note that this story is in the manual; there is no intro demo.



The title screen

The same game again or not?

Most times you see that a sequel is almost exactly the same game idea without any nice improvements. Only graphics and the music have been improved. This is not the case with Pentaro Odyssey 2.



How do I get home?

Pentaro Odyssey 2 is a platform game with horizontal and vertical scrolling just like part one. But Pentaro Odyssey 2 seems to be scrolling a bit smoother than its predecessor. Even when there are a lot of things happening at the same time, there will be no delay and the scroll is still smooth. The main idea is still the same: get Pentaro to the exit and kill or avoid the enemies that are on your way.

There are more improvements: one of the biggest changes is that there is a map and just like in Mario 3 for the NES, you can choose which stage you will do first. In addition, you can go back to stages to pickup items and lives. This is a nice and good option, in my opinion.

This also brings in one new point: you have to think about which stage you are going to do first, because you might need some items for another stage. So, be careful where to go. This is not about being lucky: most of the time it is very logical and for the first stage it is mentioned in the manual.

Another change is that there are no shooting stages anymore, so the game has become more a platform-only game. However, you still will need to take action against the enemies and you will still have to kill the bosses with your weapons. Yes weapons, because another nice change is that you have different weapons. These weapons should be picked up. The original weapon you had in Pentaro Odyssey 1 can still be used without limits, but of the other weapons you have a limited supply and they are more powerful or sometimes handier. It can also be that there is only one weapon that will harm a creature. So, use them wisely.

Pentaro Odyssey 2 also has bigger stages. It is now possible that you will have to go through doors which will take you to another place where you have to get an item to solve a problem in another part of the stage. So always search the stages carefully; some things might be impossible at first sight.

Some spooky cavern

Mission Impossible 2?

The big problem of Pentaro Odyssey was that the game actually was too hard. I do not know one person except the programmer who finished it. So when Pentaro Odyssey 2 came out I was a little bit sceptic: would be impossible again? Pentaro Odyssey 2 is not easy either, but it is always playable. The only problem is that saving is not always possible, because in some levels you will need to get all fish before you have this option. Getting these fish is not easy. Most times they are in a secret room or are heavily protected by some creatures. The same goes for lives that are also often hidden in a secret place. In the next MEGA guide, I will list some spots where you can get some nice items. In Pentaro Odyssey there were *some* secret places, but in Pentaro Odyssey 2 there are a lot more. Luckily, you can see if a stage has them. Also there are more extra lives. With these and the save option, it must be possible to finish it. However, this will not mean that you can finish it in a few hours. The game is still difficult enough. Especially the stage called 'Der Ecke' is hard... I originally thought that it was impossible, but later I found that it can be done.

Easy start. Hmm.. Easy?

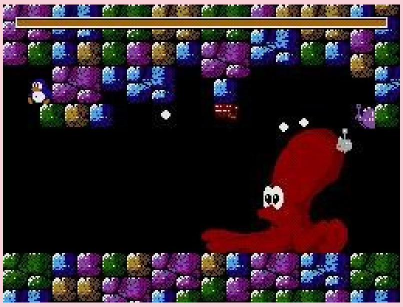
Graphics & Sound

Everything looks good. You can clearly see what creatures you are fighting against and there are some nice animations. The water has some animated waves and that sort of stuff. These animations do not slow down the game. So everything keeps running as smooth as it was. The only part where the graphics are not that great is when fighting the bosses. Some look rather simple, if you compare them to the rest of the game. However they are big and still hard to kill. The graphics of the end demo are very original. The graphical artist has his own style. Not some Japanese Manga imitation or some very simple drawing style, but some real good original graphics which reminded me to some old cartoons and comics. Really nice to see... if you can get to it! For the less advanced players I took a screen shot of it. So you can see what you are missing.

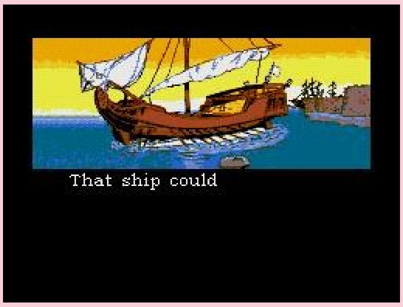
The sound is nice. Some music is from Penguin Adventure as we could have expected, considering the game is based on the story of this Konami hit. The most important thing is that it fits the game. Some songs contain very nice effects. Like a whispering wind voice for a cave level. It is a bit of shame that only the FM-PAC is utilised. What happened to the Moonblaster stereo effect and to Moonsound? They are not used a lot these days, it seems.

Bugs

The game still has some bugs. There are actually three bugs in it. First of all, sometimes it will be impossible to save if you have more than 10 lives in the first levels. Even if you have all the fish. Secondly, the save screen is sometimes messed up. However, the most annoying bug is that Pentaro Odyssey 2 will crash if you have played for a long time and you have used the 'Load' function many times for the same stage. Only then this will happen. However, Sunrise knows about the bugs and is working on a patch. If you own the game already and found some other bugs please let Sunrise know. There will also be a patch for joystick support. The original version can only be played with the cursor keys, but the programmer has made a patch for joystick use already.



Evil Inky, the last bastard



That ship could

Will this ship sail to part three or shall it take Pentaro home?

Conclusion

Pentaro Odyssey 2 is a good platform game with nice graphics and good music. The difficulty level is OK this time. It can be hard sometimes, but it can be finished if you play and thus practice enough. The only problem is the bugs. Nevertheless, knowing Sunrise they will fix it.

There is no real reason why you should not buy it, except if you really hate platform games. Besides, not that many games are released for MSX these days. So, go to Sunrise and order it!

how to control it?

MIDI interface of the GT

One of the least documented and most complex parts of the MSX turbo R GT is the built-in MIDI controller. Therefore, it has cost me a lot of time and effort to gather the data for this article.

Alex Wulms

Directory

References

1. http://www.inter.nl.net/u...

The MIDI link that is used for communication between two synthesizers or one synthesizer and a computer, is a so-called asynchronous serial data link. In the computer industry, there are several serial controllers to build such a link with. Panasonic uses the 8251 Universal Asynchronous Receiver/Transmitter, or UART.

Unfortunately you need more than just a serial interface for a good MIDI link. A MIDI apparatus has to be able to send MIDI data with certain timing, for example to be able to control the tempo of the music. In the Panasonic GT that is done by the 8253 programmable timer controller, which I will simply call the ‘timer’. Both the 8251 and 8253 are also used to control the serial connection and timing of the official MSX RS232C interface.

Connection
The MIDI-controller in the GT is connected to the I/O ports as shown in the following table:

Connections MIDI controller to I/O ports

address	description
0E8h	UART data register
0E9h	UART command/status register
0EAh	timer interrupt flag off
0EBh	timer-counter 0 data register
0ECh	timer-counter 1 data register
0EDh	timer-counter 2 data register
0EFh	timer command register

The UART and the timer are connected with the CPU in several ways, to guarantee optimal cooperation:

- The output of counter 0 of the timer is connected to the clock-input of the UART to get an adjustable baud rate.
- The output of counter 2 of the timer is connected to a so-called flip-flop. This is set when counter 2 becomes 0 and it is reset when any value is written to the I/O register 0EAh.
- The output of the flip-flop and the DTR output of the UART are connected to the input of an AND gate, so the output of the flip-flop is only passed if DTR is set.
- The output of the AND gate is connected to the DSR input of the UART. This DSR signal can be read from the status register of the UART. As a result, the status register shows the status of counter 2.
- The RxRDY output and the RTS output of the UART are connected to another AND gate. The RxRDY signal is thus only passed if RTS is set. The UART uses the RxRDY to signal that MIDI data have been received.
- The outputs of the two AND gates are connected to the Interrupt input of the CPU. Thus, the CPU receives an interrupt request if MIDI data have been received, in which case RxRDY is set, and the CPU also receives an interrupt when counter 2 becomes 0.
- The output of counter 2 is also connected to the input of counter 1.
- The output of counter 1 is not connected to anything, so you cannot generate an interrupt request with counter 1. However, the value of counter 1 can be read through its data register on 0EDh.

See figure 1 for the connections.

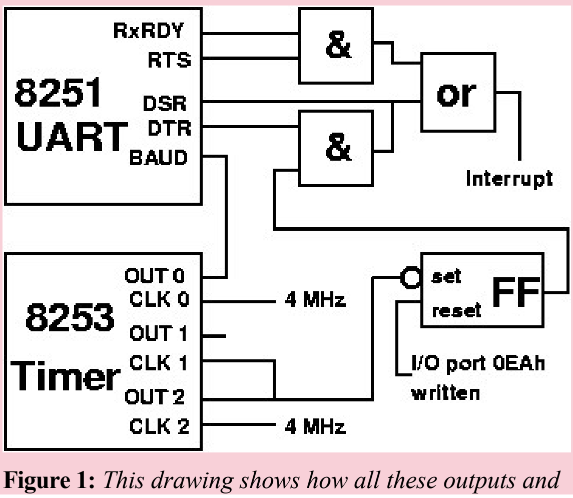


Figure 1: *This drawing shows how all these outputs and*

inputs are connected. Some details that would make the drawing needlessly complex have been omitted.

Setting the UART mode

The UART can work in two modes: a synchronous mode and an asynchronous mode. In fact, the 8251 is a USART; the S standing for Synchronous. However the MIDI controller only needs asynchronous mode. That is the only one I will discuss here.

Before the UART can be used, the right mode has to be set. Before we are able to do this, it will have to be reset. This is done by writing three zeroes and one sixty-four to the UART through the command register at address 0E9h. By the way, the timing is critical; between writing two values to the command register, you will have to wait 4ms (this is about 1 tick on the counter on I/O-address 0E6h.

After the UART has been reset, the mode can be set by writing the appropriate value to the command register. The mode value is composed as follows:

b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
S2	S1	EP	PEN	L2	L1	B2	B1

- S2/S1: number of stop bits:
  - 0 = invalid
  - 1 = 1 stop bit
  - 2 = 1.5 stop bits
  - 3 = 2 stop bits
- EP: even/odd parity generation:
  - 0 = odd parity
  - 1 = even parity
- PEN: parity enable:
  - 0 = no parity check or generation
  - 1 = parity check or generation
- L2/L1: character length:
  - 0 = 5 bits
  - 1 = 6 bits
  - 2 =7 bits
  - 3 = 8 bits
- B2/B1: baud rate factor:
  - 0 = synchronous mode
  - 1 = 1×
  - 2 = 16×
  - 3 = 64×

For MIDI, 1 stop bit is needed, no parity check, eight bits character length and a baud rate of 31.25 kHz. The baud rate is set with counter 0 of the timer and the baud rate factor of the mode value. If, for example, counter 0 of the timer is set to a clock frequency of 500 kHz, the baud rate factor has to be set to 16×. This leads to a binary mode value of 0100110 (decimal value 78). For a complete initialisation of the UART, a MIDI program has to write the following five values to the command register: 0, 0, 0, 64, 78. Between writing two values it has to wait 4 ms.

The UART commands

After a mode has been set, all values that have to be written to the command register are interpreted as so called command values. In fact, resetting the UART with 0, 0, 0, 64 does in fact mean that four successive commands are written to the UART. The first three zeroes are only needed to synchronise the UART with the CPU, just in case the UART is in the middle of a multi byte command mode setting. The fourth value is the actual reset command. Note that these multi byte commands are not used in asynchronous mode, but it could be possible that a previous (non-MIDI) program has set the UART to synchronous mode.

A UART command is composed as follows:

b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
EH	IR	RTS	ER	SBRK	RxE	DTR	TXE

- EH: Enter hunt mode
  - This bit has no function in asynchronous mode
- IR: Internal reset
  - 0 = do not reset UART
  - 1 = reset UART so that mode can be changed
- RTS: request to send. This bit is linked to the RTS output of the UART and in the MSX turbo R GT is used to link the RxRDY to the interrupt request input of the CPU.
  - 0 = MIDI interrupt is off
  - 1 = MIDI interrupt is on
- ER: Error reset
  - 0 = nothing
  - 1 = reset FE, OE and PE status flags
- SRBK: Send break character
  - 0 = normal operation

    - 1 = pull the serial MIDI output down so a continuous stream of zero bits is sent
- RxE = Receive enable
    - 0 = MIDI in is off
    - 1 = MIDI in is on
- DTR: Data terminal ready. This bit is linked to the DTR output of the UART and in the MSX turbo R GT is used to mask the output of counter 2.
    - 0 = counter 2 is ignored
    - 1 = counter 2 can generate interrupts
- TxE = Transmit enable
    - 0 = MIDI out is off
    - 1 = MIDI out is on

To switch MIDI out and MIDI in both to on, the RTS, RxE and TxE bits have to be set. If counter 2 is used as a programmable interrupt generator for timing the recording or transmission of MIDI data, the DTR bit has to be set as well. All other bits can remain reset.

The result is a binary command value 00100111, the decimal value 39. If no programmable interrupt generator is necessary, the binary value 00100101 suffices (decimal 37). And if no interrupts are needed when MIDI data have been received, the binary value 00000101 (decimal 5) is enough.

The UART Status register
The UART also has a status register that can be read through I/O port 0E9h. This status register is composed as follows:

b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
DSR	BRK	FE	OE	PE	TxEM	RxRDY	TxRDY

- DSR: Data set ready
    - This bit is connected to the DSR input of the UART, which in the GT is connected to the output of counter 2 of the timer.
- BRK: Break detect. A break signal is detected by checking the stop bit. If it has been 0 twice in a row, there is a break signal. Normally, the stop bit can only become 0 if the transmitter has set the SBRK bit of the command register. Break signals are not used in the MIDI protocol.
    - 0 = no break signal detected
    - 1 = break signal detected
- FE: Framing error. A framing error is caused by a break signal or an error in the data transfer. With MIDI, a FE is always caused by an error in the data transfer, because break signals are not used.
    - 0 = no framing error
    - 1 = stop bit was 0
- OE: Overrun error
    - 0 = no overrun error
    - 1 = the UART has received a new character while the CPU was not ready reading the previous character.
- PE: Parity error. Since parity is not checked in the MIDI protocol, this bit is always 0 when transferring MIDI data.
    - 0 = no parity error
    - 1 = wrong parity
- TxEM: Transmitter empty. If TxEM is 1, the TxEM will become 0 as soon as the CPU has written a new value to the data port.
    - 0 = the UART is sending data
    - 1 = the UART is ready sending the last byte
- RxRDY: Receiver ready. This status bit is also connected to the RxRDY output of the UART, which in the GT, is combined with the RTS signal and connected to the interrupt input of the CPU.
    - 0 = no byte received
    - 1 = byte received
- TxRDY: Transmitter ready. The TxRDY status bit is always equal to the TxEM status bit. The difference between these two bits is that the TxEM bit is directly connected to the TxEM output of the UART, and the TxRDY bit is masked with two other bits before it is connected to the TxRDY output of the UART. For the MSX programmer, this difference is irrelevant because these two UART outputs are not used in the GT.
    - 0 = the UART is not ready to send a new byte
    - 1 = the UART is ready to send a new byte

Sending and receiving MIDI data
After the UART and counter 0 of the timer have been initialised, it is possible to send MIDI data by sending it to I/O port 0E8h and to receive MIDI data by reading I/O port 0E8h. Before writing a byte to I/O port 0E8H, the program must wait until the TxRDY status bit is set.

Receiving MIDI data can be done by an interrupt routine which is called every time a new byte comes in. Programmers who find interrupt routines too complex can leave the interrupt generation off by resetting the RTS bit to 0 when setting the UART command. Next, they can check when MIDI data arrives by testing the RDY status bit.

The timer
The 8253 programmable timer exists of three independent 16-bit counters. They can count in binary or BCD

mode. The MSB and LSB of a counter can be independently set and read, so that they can also be used as 8 bit counters. They always count down. The mode of the counter can be set by writing a mode value byte to I/O port 0EFh. This command value is composed as follows:

b7	b6	b5	b4	b3	b2	b1	b0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

- SC1/SC0: Select counter
    - 0 = select counter 0
    - 1 = select counter 1
    - 2 = select counter 3
    - 3 = not allowed
- RL1/RL0: Read/load
    - 0 = counter latch operation. See below at ‘Reading the counter’
    - 1 = read/write only the LSB
    - 2 = read/write only the MSB
    - 3 = read/write the LSB followed by the MSB
- M2/M1/M0: Mode (The counter modes are discussed below.)
    - 0 = mode 0
    - 1 = mode 1
    - 2,6 = mode 2
    - 3,7 = mode 3
    - 4 = mode 4
    - 5 = mode 5
- BCD: Binary coded decimal
    - 0 = counter works as 16 bits binary counter
    - 1 = counter works as 4 digit BCD counter

Counter initialisation

A counter is initialised after the right number of bytes (determined by the RL bits) is written to the counter data port. In 16-bits mode (RL1/RL0 = 3) for example, the counter will only be initialised after two bytes have been written to the data port.

Reading the counter

A counter can be read by reading the right number of bytes from the data port. For a reliable result, the countdown must be paused before the counter is read. This can be done by setting the mode of the counter.

To be able to read the counter during countdown, the timer has a special option: there is an internal 16 bits buffer for every counter. The value of each counter is copied to the buffer as soon as the RL1/RL0 bits of the mode register are both 0. The M2/M1/M0 and BCD bits of the mode register are ignored in this case, and directly after copying the counter to the internal buffer, the RL1/RL0 bits are restored to their original state. When the data port of the counter is read the next time, the value in the buffer is shown in stead of the value of the counter itself.

The counter modes

Every counter can work in 6 different modes. These modes can be set with the M bits.

- Mode 0: Interrupt on terminal count
  After setting this mode, the output of the counter is low. After the counter has been set, the output remains low and the counter starts to count down. When the counter reaches 0, the output becomes high and stays high until a new value has been written to the counter or the counter mode as been set again. By the way, the counter will keep counting after reaching 0.
- Mode 1: Programmable one shot
  This mode cannot be used in the GT because the so called gate input of both counters is connected to +5V.
- Mode 2: Rate generator
  With the rate generator you can make a ‘divide by N counter’. After this mode has been set, the output of the counter becomes high and will remain high until the counter has been set with value N. After this, the output of the counter will become low for 1 clock pulse and the output will become high for the next N-1 clock pulses. This process keeps repeating itself until the mode is set again. If on the other hand the counter is set again during countdown, the automatic count-down process will continue. The new setting is only used after the counter has passed 0. This mode can be used for making a programmable interrupt generator using counter 2. If the DTR bit of the UART is set, every time counter 2 reaches 0, an interrupt signal is sent to the CPU. E.g. to make a 200 Hz interrupt generator, counter 2 can be set to mode 2 and to value 20000 (the clock frequency of the counter is 4 MHz).
- Mode 3: Square wave rate generator
  By setting the counter to mode 3, you can make a square wave. If an even number N is written to the counter, the output will be high for N/2 clock pulses and low for N/2 clock pulses. The result is a symmetrical square wave. If an odd number M is written to the counter, the output will be high for (M+1)/2 clock pulses and low for (M-1)/2 clock pulses. This will result in an asymmetrical square wave. Mode 3 is needed to set a clock frequency for the UART, among others. E.g. to provide the UART with a 500 kHz clock, counter 0 can be set to mode 3 and be initialised with the value 8.
- Mode 4: Software triggered strobe
  After mode 4 has been set, the output of the counter goes high. When the counter has been set, it starts counting down and if it reaches 0, the output goes low for 1 clock pulse and the counter will stop

counting.
- Mode 5: Hardware triggered strobe
  This mode can not be used in the GT.

The listing
Below is an example of a MIDI program. It sends all MIDI data that are received through to MIDI out. For this, the program first sets counter 0 to generate a 500 kHz square wave. Next, the UART is set to receive serial data and write data at a 31.25 kHz baud rate. After this, the program will wait for new data to come in and send it through to the MIDI out port.

I have not been able to test this program though, since I do not have enough synthesizer equipment. Here it is:

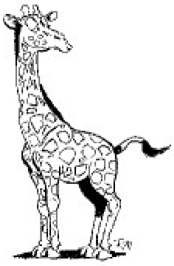
ML-listing: TRMIDI.GEN

```
; Example MIDI program for GT which can be run under MSX-DOS(2)
; Written by Alex Wulms for MCCM
; 26-9-1995

UARTsend: equ         0e8h       ; 8251 data send
UARTrecv: equ         0e8h       ; 8251 data receive
UARTcmd:  equ         0e9h       ; 8251 command/mode register
UARTstat: equ         0e9h       ; 8251 status register

tm_int:      equ      0eah       ; timer interrupt flag off

timer0:      equ      0ech       ; 8253 counter 0 dataport
timer1:      equ      0edh       ; 8253 counter 1 dataport
timer2:      equ      0eeh       ; 8253 counter 2 dataport
tm_cmd:      equ      0efh       ; 8253 command

             di
             in       a,(0aah)
             and      0f0h
             or       7
             out      (0aah),a   ; select row 7 of keyboard

          ld          a,00010110b          ; set timer 0 to
             out      (tm_cmd),a ; generate a square wave
             ld       a,8                  ; set a frequency
             out      (tm_cmd),a ; of 200 kHz

             ld       hl,UARTdata
             ld       b,6
initUART: ld          a,(hl)               ; initialize UART for
             out      (UARTcmd),a          ; MIDI data communication
             inc      hl
             call     waitUART
             djnz     initUART
             in       a,(UARTrecv)         ; reset outstanding flags

main:        in       a,(UARTstat)
             and      2                    ; check RxRDY
             call     nz,midithru          ; data waiting
             in       a,(0a9h)   ; read keyboard
             bit      2,a                  ; check ESC key
             jr       nz,main              ; not pressed

             xor      a
             out      (UARTcmd),a          ; set MIDI off
             ret                           ; thats all

waitUART: in          a,(0e6h)   ; wait 4 micro seconds
             ld       c,a                  ; between two UART operations
waitUART2:in          a,(0e6h)
             sub      c
             jr       z,waitUART2
             ret

midithru: in          a,(UARTrecv)
             ld       b,a                  ; B = midi IN data
midithr2: in          a,(UARTstat)
             and      1                    ; check TxRDY
             jr       z,midithr2 ; UART not ready
             ld       a,b
             out      (UARTsend),a         ; write to MIDI out
             ret

UARTdata: db          0,0,0,64,78,5
```

More articles — including the original Dutch version of this one — can be found on Alex Wulms' home page, see reference [1]

previous:
Pentaro Odyssey 2

next:
De Maiskoek/Bits and Pieces



Page for computer news in general and MSX news in particular

De Maiskoek/Bits and Pieces

De laatste mensen

Er zijn altijd al doomscenario’s geweest. Grappige; zoals de robot die zodanig geprogrammeerd was dat hij geen mens kwaad kon doen; hij werd op zekere dag aangezet en na een kleine pauze begon hij op zoek te gaan. Het speelde natuurlijk in Amerika en daarom zal het geen verwondering wekken dat hij al snel gevonden had wat hij zocht: een pistool. Zij haalde de veiligheidspal eraf en de ontwerpers keken bevreesd toe wat er nu zou gaan gebeuren. Maar ja, de robot richtte het wapen niet op de hoofdontwerper, niet op de bouwer en zelfs niet op de financier, maar op zichzelf. Logisch: hij kon geen mens kwaad doen dus moest hij de mens wel verlossen van de grootste bedreiging voor die mens. Andere verhalen zijn somberder en spelen in een toekomst waar voor mensen geen plaats meer is, of alleen in een beschermende omgeving die een neutraal beschouwer als zeer aangenaam voor de mens zal beoordelen, het is alleen wel een dierentuin, want onze rol als dominerende soort zijn wij kwijt geraakt. Recente schattingen komen tot ergens tussen 2050 en 2060 voor het tijdstip waarop kunstmatige intelligentie die van de mens overstijgt.

Wat valt hier tegen te doen? Van de politici valt niet veel te verwachten: lieden die al eeuwen gewend zijn achter de feiten aan te lopen en pas maatregelen te nemen als het uit de hand is gelopen die zullen ons echt niet beschermen tegen deze ontwikkeling. Integendeel zij zullen juist wel maatregelen nemen en bepaalde — door sommigen gewenste — ontwikkelingen tegenhouden of traineren. Het gevolg zal zoals altijd zijn dat die ontwikkelingen in het geheim en zonder enige algemene controle zullen plaatsvinden. En als het dan eenmaal naar buiten komt zal het misschien te laat zijn.

Is er nog hoop voor de mensheid? Ik moet u teleurstellen, de enige kans op overleven is uitsterven. Dit klinkt nogal paradoxaal, maar ik zal het toelichten. De kansen dat uw nazaten nog enige kans van dominantie over deze aarde zullen hebben zijn klein. Computers zijn over een vijftigtal jaren slimmer dan mensen en veranderen in schrikbarend hoog tempo — nu al elke anderhalf jaar een tweemaal zo snelle generatie — en computers gaan computers ontwerpen en zullen mensen steeds meer bezien als wij de eerste ‘vis’ die aan land

MCCW 93 finally online

As you have noticed, MCCW 93 is finally online. All texts from this issue are written at the end of 2000 or the beginning of 2001, including this Bits and Pieces page... So, you will notice that most of the ‘news’ items on this page are very out of date! However, for history’s sake, we have not removed them. But please keep this in mind! For up-to-date news, we refer to The MSX Resource Center. Since this issue is also the final release, please do not react anymore to the CALL advertisements...

Lucyforge

The German MSX group Programmer’s Affair has changed its name to Lucyforge. The groups has extended its scope to Playstation. They will continue with their project OSR (One Shot Rising), but will also make a version for the Playstation. Their new site is here. The old site seems to be offline.

New products from Matra

“The Matra Corporation” has shown some previews of new products on MadriSX. The first product to be released is “Don’t cock it up!”, a Tetris clone for one or two players, including battle mode! The product after that will be a horizontal shooter, but we have only seen some screenshots of that. We are happy to see that Matra has again new products, they seem to be very alive. Last year (2000) they also released two new games: Ark-A-Noah — an Arkanoid clone in that typical Matra-style with mouse control of course — and Moskow 2024: The Olympic Megashock, which is a Hyper Olympics clone for MSX2. The prices are reasonable for the quality offered: EUR 9 for Ark-A-Noah and Don’t cock it up! (Caligula Budget series) and EUR 15 for Moskow 2024, which even has its own website: `http://www.moskow2024.com`. If someone offers to write a review of these games, expect them to be published here. The previews, screenshots and demonstrations look very convincing!

Carnivore schiet providers in verkeerde keelgat

De FBI blijft graag op de hoogte en daar is op zich voor een inlichtingendienst niets mis mee, maar als dat de eigen bedrijfsvoering bemoeilijkt of in ieder geval belast, is dat toch wat problematischer. In Amerika zijn internetproviders verplicht de FBI toe te staan een aftapsysteem op het netwerk te installeren.

Free Japanese games

The Japanese company Studio Wing has announced a new game for pc, which will be an enhanced version of their MSX title. The interesting thing is that they have now put online the MSX version of this Japanese text adventure. It can be downloaded for free! Together with this they also put online a lot more old games they made for MSX, all freely downloadable. Most are Japanese text adventures, but there is also a nice puzzle platform game. They will put more older material of them online in the future. It can all be found on `http://www.st-wing.net/`. Let us hope other (former) MSX companies do the same.

WWW veel groter

Uit onderzoek van Bright Planet blijkt dat het web veel groter is dan tot nu toe werd aangenomen. De gangbare zoekmachines beperken zich namelijk enorm en het vermoeden bestaat dat zelfs met alle zoekmachines samen nog maar 42% van het hele web wordt gevonden. Het bedrijf schat dat er wel een miljard pagina’s op het web staan. Dat is wel wat meer dan de vijftigduizend die Lycos er 1994 had.

Compaq stopt met Vax

Of het het einde van alle minicomputers betekent weten we niet, maar feit is dat Compaq een einde maakt aan de productie van zijn mini. De Vax is een van de succesvolste systemen aller tijden. Het was de opvolger van de PDP11 en werd net als die geleverd door Digital. Later werd het geheel overgenomen door Compaq, maar aan alles komt een eind en na 22 jaar trouwe dienst moeten sommigen toch even wat wegslikken.

Elektronische verpleegster

Matsushita Communication Industrial heeft op een tentoonstelling in Tokyo een prototype voor een verpleegrobot getoond. Het apparaatje ziet er uit als een soort staand model stofzuigertje — zonder slang — en heet Tamapy. Hij is voorzien van diverse sensoren die temperatuur, bloeddruk en hartslag kunnen meten. Tamapy praat — Japans — en de dokter kan via een mobiel netjes de gegevens opvragen.

Gemakkelijker studeren

Of eigenlijk moeilijker studeren, want Minister Hermans van OC en W heeft toegestaan dat ook die studenten tot de studie worden *klom. Om in zo’n omgeving nog enige kans te hebben zijn er voor de mensheid twee mogelijkheden: integreren of muteren.*

Integreren is onszelf ombouwen tot een androïde of cyborg. De begrippen ontlopen elkaar niet veel. Onder een androïde wordt meestal een vrijwel mensachtige robot verstaan die alleen door zijn uiterlijk en mogelijk — hopelijk? — programmering op een ‘echt’ mens lijkt. Een cyborg is een halfslachtig wezen: half mens half robot, maar de verhoudingen liggen niet vast. Om iemand met een bril nu al een cyborg te noemen zal velen te ver gaan, maar iemand met een pace maker? Computers kunnen nu al veel lichaamsfuncties in de gaten houden en met kleine ingebouwde ‘devices’ reageren. Als de computer over enige tijd uw hersengolven leert interpreteren is de spraakherkenning van dit moment een lachertje geworden. Inplanteer een kleine interface in uw schedel en om de zoveel tijd haalt u het nieuwste model hersenchip. U denkt eens: ‘zou er ergens in de ontwikkeling van het getal pi ook mijn geboortedatem staan?’ en zonder dat u daar verder iets aan hoeft te doen ‘flitst’ iets later ‘Ja, op de 67 923 348e decimaal begint dat.’ door u heen. Wij zijn geen ‘echte’ mensen meer, maar leven en zijn altijd slimmer dan computers. Wij zijn dan namelijk computer plus een steeds meer verwaarloosbaar deel, mens.

Muteren kan echter ook. Het menselijk genoom — zeg maar ons DNA — is vrijwel bekend. Op zich trouwens wel vreemd dat groots wordt aangekondigd dat het haast helemaal bekend is. In een kleine tien jaar is ruim 97% ontrafeld en dan geven ze een persconferentie. Je zou toch zeggen dat je beter nog een maandje kunt doorwerken om dan te verkondigen dat je klaar bent. Of zijn die laatste paar ACGT’s vreselijk moeilijk te plaatsen? Net zoals dat laatste procent huishuidbacteriën dat ook niet is weg te krijgen. Maar hoe het ook zij, in principe wordt het hiermee mogelijk ons eigen nageslacht te componeren. En wij maken onze kinderen natuurlijk slimmer. Jammer dat u bij schaken verliest van uw vierjarig zoontje en dat uw kleindochter u al liggend in de wieg versloeg, maar de mens blijft slimmer dan de computer. Helemaal al omdat steeds slimmere ‘mensen’ het DNA herprogrammeren.

Net zoals steeds slimmere computers nieuwe generaties computers ontwerpen.

Parcellus

I/O

INPUT

Who has some cheap MSX stuff for sale/trade? Please mail me! manuel@msxnet.org. You can also see what I already have on my homepage! At the moment, I have a lot of Konami cartridges for trade only... Check it out.

CALL

Het systeem bestaat uit een ‘gewone’ pc die in het netwerk wordt opgenomen en waar speciale software op draait, zodat de FBI zijn inlichtingen kan vergaren. Iemand die niets te verbergen heeft zou hier niet al te veel problemen mee moeten hebben, maar als — om de vergelijking overdreven door te trekken — zij bij u in de badkamer een videocamera willen monteren om in de gaten te houden of u aan kinderporno doet zult u toch in ieder geval de zekerheid willen hebben dat die camera elektrisch veilig is. En daar zit bij de systeembeheerders nu juist de knop, want de FBI laat niets los over hoe hun pc werkt en hoe hij het net beïnvloedt.
Als het systeem eenmaal werkt kan men van een verdachte alle internetactiviteit bijhouden, e-mail onderscheppen en chats afluisteren. Zelfs de ‘messages’ zijn te volgen en dat veel encryptie is te decoderen ligt voor de hand. In principe mag alleen zo’n actie worden ondernomen als de rechtbank daarvoor toestemming heeft gegeven, maar door alle geheimzinnigheid zijn de providers niet in staat te controleren of de FBI niet buiten die toestemming gaat. En als de FBI iets op internet van u monitort? U bent geen Amerikaan.

Deukdijen

In België hebben veel mensen — vooral vrouwen — die met computers werken deukdijen opgelopen. Of om het wat wetenschappelijker te zeggen: lipoatrofia semicirculas. Het komt vooral voor bij vrouwen die in een modern kantoor met modern meubilair achter een moderne computer zitten. Vermoedelijk door de elektromagnetische straling verliezen de bovenbenen wat onderhuids vet, al valt aan de huid te plekke niets bijzonders te zien. In Nederland is het verschijnsel nog niet gemeld, maar TNO Arbeid stelt een onderzoek in.

Geheugenrek van HP

Hewlett Packard heeft voor de zwaarste toepassingen de HP J6000 Compute Farm. In een kast van twee meter hoog zitten 20 HP Visualize J6000 systemen als kippen in een legbatterij, om in de agrarische sfeer te blijven. Zo’n J6000 is uitgerust met twee PA-8600 processors en beschikt per processor over 8 GB geheugen. Een eenvoudig rekensommetje leert dat het hele systeem in de maximale configuratie over 20×2×8=320 GB geheugen beschikt. Hiermee zijn nieuwe IC’s te ontwerpen en... nieuwe spelletjes.

Heathrow snel

Velen die wel eens gebruik maakten van dit Londense vliegveld zullen bij het lezen van deze kop de wenkbrauwen fronsen. Vertragingen lijken daar een vast bestanddeel van de dagvulling te zijn. Wij hebben het hier echter over een nieuwe chip van het Deense I-data. Zij heeft een 16-poort ethernet switch-on-a-chip gemaakt die 24 miljoen pakketten per seconde kan verwerken. De interne switching engine verwerkt 32 gigabits per seconde. Nu maar hopen dat de chip zijn naam niet waar maakt.

Irridium komt naar beneden

toegelaten die niet Wiskunde B 1 en 2 in hun profiel hadden zitten. Enerzijds wordt daarmee de drempel tot bijvoorbeeld informatica verlaagd, maar anderzijds verhoogd omdat de studenten in het eerste jaar van de studie alsnog aan de eisen moeten voldoen. Dat wordt dus hard aanpoten, zeker als juist een profiel zonder Wiskunde B werd gekozen wegens weinig aanleg voor dat vak.

High met WAP

Via iToke kan de liefhebber in Amsterdam met zijn WAPmobieltje marihuana bestellen. Die wordt zeer milieuvriendelijk in porties van 2 gram per fietskoerier bezorgd. Betalen moet met zogenaamde iTokens die vooraf moeten worden aangeschaft. Deze vorm van prepaid drug moet de wietscene een facelift geven, volgens de Amerikanen die er achter zitten.

Intel heeft weer snelste

Nog voor de Pentium 4 uitkomt en voor de nieuwe Itanium op de markt verschenen is, heeft Intel opnieuw een snellere versie van de Pentium III uitgebracht. Nu is het natuurlijk leuk dat zij die hebben aangekondigd, maar voorlopig komen de versies van boven de 800 MHz ook nog niet in de advertenties. Zij zijn officieel wel op de markt, maar blijkbaar niet echt leverbaar. De nieuwe recordhouder in de Pentium III serie werkt op een 1,13 GHz en is ook gebaseerd op de Coppermine techniek, dus .18-micronbasis. Intel beweert dat zij dus de snelste chip heeft, en wij nu maar hopen dat zij die ook kan leveren. Voor geïnteresseerden wellicht de waarschuwing dat de op de Willamette gebaseerde Pentium 4 die in het najaar van 2000 verwacht werd en op 1,4 GHz of iets in die buurt zal draaien. De vermoedelijk ook aan het eind van 2000 komende Itanium zal ‘maar’ 800 MHz doen, maar door nieuwer ontwerp en 64-bitstechniek toch sneller blijken. Ook duurder, dat wel.

Internetdomein geliefd

Het aantal internetdomeinnamen is in Nederland enorm toegenomen. Kwamen er vorig jaar (1999) de eerste zes maanden 25000 domeinnamen bij, nu waren dat er tien maal zoveel. De score staat nu op 406000 geregistreerde namen. en dan te bedenken dan vorig jaar augustus er nog maar 91000 puntenel — .nl voor de slechte verstaander — waren.

Kleine grote harddisk

Het is een fenomeen dat we vaker in de informatietechnologie aantreffen maar dit verhaal trof ons toch weer. Toen IBM in 1980 zijn eerste harddisk op de markt die 1 GB aan opslagruimte had was dat een fors model ter grootte van een (Amerikaanse) koelkast. En door het gewicht van een slordige 250 kg verplaatste je die ook niet eventjes. IBM vroeg daar destijds veertigduizend dollar voor. Momenteel levert IBM voor nog geen vijfhonderd dollar een harddisk van 1 gigabyte voor kleine computers. Het apparaatje is zeer klein en de schijf zelf is nog kleiner — en dunner — dan een munt van vijf gulden. De hele drive is ietsje groter, maar in de zelfde verhoudingen als de normale pc-drives. Op de

Who wants to write an article about an interesting MSX topic for this webmagazine? Please contact us! mccw@aktu.nl

We want to continue with the Art Gallery! So please send us your MSX drawings! Thanks!

Lief Maiskoekje zoekt leuke I/O'tjes om samen op het net te staan. Mail even naar mccw@aktu.nl.
If *you* have some little advertisement for this page, you can mail us at mccw@aktu.nl.

Motorola heeft toestemming gekregen om de satellieten van het Irridiumnetwerk neer te halen. De eigenaren van het peperdure netwerk voor mobiele communicatie zijn benauwd voor juridische aansprakelijkheid als iemand zo'n satelliet op zijn kop krijgt en laten ze daarom liever zelf in zee storten. Daarmee hebben ze wel ruim vijf miljard dollar in het water gesmeten. Meer zelfs, want het gecontroleerd afvoeren kost op deze wijze ook nog eens iets van tussen de dertig en vijftig miljoen. Overname van het netwerk, zelfs voor een vriendenprijsje van vijftig miljoen, is nog steeds niet gelukt. Maar met slechts enkele duizenden abonnees wordt het natuurlijk niet rendabel. Ook al zakken de prijzen tot een dollar per minuut — toch niet echt duur voor wereldwijd mobiel bellen — blijken er toch minder globetrotters te zijn dan destijds vermoed.

Mobiel blijft groeien

Vinden vooral ouderen mobiel bellen maar duur communiceren, grote groepen zien het juist als de toekomst. De groei van het aantal mobieltjes lijkt die laatsten gelijk te geven, want nu al zijn er 570 miljoen toestellen in omloop. Nemen we even — onterecht trouwens — aan dat ieder mens maar maximaal één mobieltje heeft, dan is nu al tien procent van de wereldbevolking mobiel te bereiken. Nokia die één der grootste producenten is, verwacht dat eind 2002 dat aantal haast verdubbeld is en de score net op een miljard toestelletjes zal liggen.

Pocket PC camera

Intel heeft het ei van Columbus gevonden en een eenvoudige webcam gemaakt die ook los is te gebruiken als digitale fotocamera. De resolutie is met 640×480 niet groots, maar voor internettoepassingen voldoende. De prijs van 149 dollar doet de rest. De camera is daarmee als webcam vrij prijzig, maar heeft wel 8 MB geheugen waarmee 120 foto's zijn te maken of desgewenst twee minuten schokkerige videofilm.

Sneller geheugen

IBM heeft nieuwe geheugenchips ontwikkeld voor gebruik in zijn nieuwe mainframes. Deze SRAM-chips zijn statische geheugenchips die functioneren op een interne snelheid van 2 GHz. In de zogenaamde Freeway-mainframes zullen ze het echter moeten doen met 64-bit Blue Flame processors die voorlopig nog — experimenteel — werken op 1 GHz. Wel sneller dan de Itanium van Intel en in het mainframe zijn er zestien te plaatsen, maar we praten wel over een mainframe met maximaal 128 GB echt geheugen, virtueel kan het zelfs 512 GB zijn.

Steeds meer chips

Ondanks de pc-verkopen wat zakten/stokten is de wereldchipmarkt nog steeds groeiend. Vorig jaar is het aantal verkochte chip maar liefst veertig procent hoger dan het jaar daarvoor. De totale omzet bedroeg 15,9 miljard dollar in mei alleen al. Op basis hiervan voorspelt de SIA dat de totale omzet dit jaar op 195 miljard dollar komt. Voor 2001 wordt zelfs 244 miljard en het jaar daarna 279 miljard in de prognose genoemd. Gezien de stijging die dan steeds lager is een voorzichtige schatting.

redactie is ook een IBM-drive aanwezig, die weliswaar een stuk groter is maar wel ruim 15 GB kan opslaan en net driehonderd gulden kostte.

PC-verkopen omlaag

Voor het eerst in jaren is in Nederland het aantal verkochte pc's gedaald. In het eerste kwartaal werden er zo'n tien procent minder verkocht dan het vorige jaar. Het vreemde verschijnsel kwam niet geheel onverwacht, want met name het bedrijfsleven had vorig jaar uit voorzorg voor de millenniumbug de aanwezige pc's eerder dan gebruikelijk vervangen. Hiermee kwam natuurlijk ook een groter aantal en recentere pc's op de tweedehands markt terecht en dat roomt natuurlijk de thuismarkt weer wat af. Echt zorgen maakt de industrie zich echter niet; het zal wel weer aantrekken.

Ruim een gigabyte per seconde

De meesten denken aan transportsnelheden van 10 Mb per seconde als ze over een snelle verbinding over het netwerk praten. Over modemsnelheden hebben we het maar niet. Met ethernet kan dat met de juiste apparatuur zelfs opgepept worden tot 100 Mb per seconde. De IEEE — Institute of Electrical and Electronic Engineers — is echter al verder en is bezig specificaties op te stellen voor verbindingen tot 10 Gb. Welk apparaat die data zo snel kan aanleveren werd er niet bij verteld. Vreemdgenoeg zal de standaard pas voorjaar 2002 klaar zijn terwijl dit jaar al de eerste producten voor die snelheid op de markt komen.

Te automatisch

Een arts in Pittsfield, Massachussetts moet elfhonderd van zijn patiënten op eigen kosten opnieuw onderzoeken. Hij gebruikte de computer op te slimme manier naar het idee van de Board of Registration in Medicine. Als hij een patiënt onderzocht en tot de conclusie kwam dat er (verder) niets aan de hand was, liet hij zijn patiëntendossier door de computer invullen. Dus ook op de punten waar vermoedelijk niets met de patiënt mis was, maar waar hij zich niet door onderzoek van had overtuigd. Hij had de vakjes oningevuld moeten laten, want nu wekte hij de indruk dat het betreffende wel onderzocht was. Hij verdedigde zich door op te merken dat hij zo meer tijd had voor zijn patiënten en de kwalen waarvoor zij naar hem toe kwamen.

Blind surfen met braille Note

In Nieuw Zeeland heeft men een Windows CE systeem ontwikkeld dat blinden in staat stelt te surfen over internet. Met aangepaste versies van Word en Outlook kunnen blinden communiceren met andere blinden en niet-blinden. Met spraaktechnologie wordt tekst naar spraak omgezet om zo te 'lezen' en navigeren, maar op het speciale 'scherm' kan ook in braille worden gelezen. Invoer gaat met een speciaal brailletoetsenbord met maar negen toetsen die in combinatie gebruikt moeten worden om de karakters aan te geven.

experiences with it

The Ciel MSX2+ Turbo

Would you like to know more about Ademir Carchano's products? Read about Ciel MSX2+ Turbo and see why there are many Brazilian MSX users buying it.

Mauricio Braga

References
1. http://www.carchano.com.br
2. programs.zip

What is it?

What is a Ciel MSX 2+ Turbo motherboard? It is a new MSX2+ motherboard made in 1996, having the following main features:

- A Z84C0010 - CMOS processor working at 7 MHz;
- 4 slots, two inside the computer and two outside (using Brazilian MSX Expert case);
- Up to 4 megabytes of memory mapper using 30 pin SIMM memory found on pc's. You can use 256 kB, 512 kB, 1 MB or 4 MB memory, but you only have one connector for it. The motherboard already includes 256 kB of internal memory, you can choose to use that internal ram or the 30 pin memory using jumpers;
- Clock and calendar maintained with a NiCd rechargeable battery;
- Sound output using FM and PSG mixed. In one channel you have the FM and in the other one you have PSG + drums. There is an option to use a headphone connector too;
- You can use v9958 or v9938 by setting jumpers;
- Three reset connectors located

Introduction

In the golden years of MSX, Ademir was known in Brazil by his great products that sold thousands of units each, like Megaram and his MSX2/2+ add-on boards to Brazilian MSX1 computers. Very useful, because MSX2 was never released in Brazil. Being an old fan of Ademir's products, when I heard about his MSX2+ Turbo motherboard, I did not think twice and I started to search for someone willing to sell it. I had to do this because Ademir did not have one for sale at that time; unfortunately he cannot produce many boards and wait for people to buy it. So it was not easy to find one, because those who have one will not sell it. But Leonard Oliveira got one and sold it to MSX Brazilian team (MBT), and they sold it to me in a MSX Expert plus case. You can learn more about Ademir's products at [1].

Installation

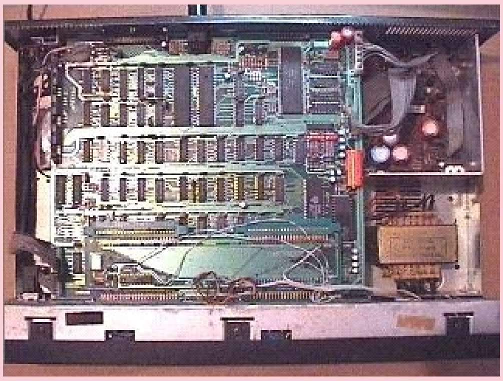
The MSX Ciel is made to be installed in an MSX Expert case. The Expert is an old MSX1 made by Gradiente 15 years ago in Brazil. It was the most popular MSX in Brazil. As a matter of fact, there were only two MSX models made in Brazil: the Expert by Gradiente and the Hotbit by Sharp. When Sharp stopped with the Hotbit, the Expert became the only MSX machine being sold here in Brazil until the production was stopped in 1991. This makes it a natural choice for Ademir's motherboard. You can of course put it in a pc case, for example, but it will not be as easy as using an Expert case.

No soldering is required to install it, unless you want to use the external headphone connector. After taking away the old motherboard it took me only 10 minutes to finish the installation and turn it on. As you can see, it is really easy to mount.

My MSX Ciel mounted in a MSX Expert Plus case.

MSX à la carte

One of the most interesting things about buying a Ciel motherboard — as a matter of fact, any product developed by Ademir — is that you probably will not find another motherboard that is exactly the same as yours, unless you specifically want it. The reason is simple: Ademir is always willing to hear your weirdest desires. So, everyone who has a Ciel motherboard, has it with his own configuration. That is one of the main advantages of buying products from Ademir, besides the quality, of course.


My MSX Ciel opened.

For example, my good friend Daniel Caetano has a pc power connector in his motherboard, so instead of using the power supply of the Expert, he replaced it with a 300 W pc power supply. He also has a 2 gigabyte hard disk and a 3.5" drive inside his computer, using the 2 internal slots. Vinicius Beltrao and Luciano Sturaro have the same configuration. Daniel even has a Sega Master System 3D glasses set connected to the VDP in order to generate 3D images.

The Ciel motherboard, taken from Ademir Carchano's site.

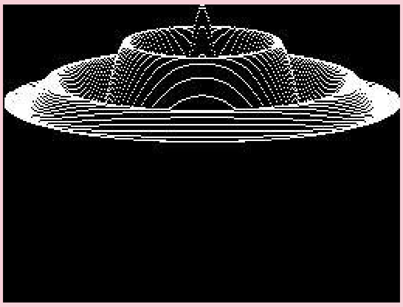
My Ciel does not have these features, but it has switches that allow me to turn on/off the slots without having to unplug the cartridges and a switch that allows me to plug 2 cartridges in the same slot. I need this to use my Megaram cartridge and my SCC at the same time, in order to run the Konami games with SCC sound, without the need of cracked versions. For this I use ExecROM from Adriano Cunha to load the megarom files used in MSX emulators in Megaram. My board also uses AA batteries instead of the NiCd battery, mainly because it is easy for me to change them. Other people have other things in their Ciel computers. The bottom line is, Ademir will modify the motherboard as you wish with little or no cost.

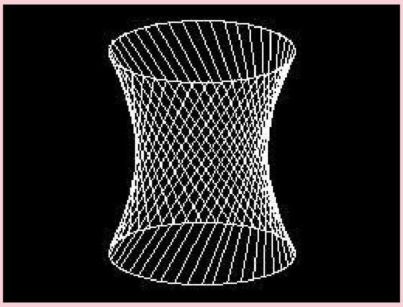
on different parts of the motherboard;
- Connector for video expansions that are in the final phase of the project: image digitising, genlock, connection of multiple VDPs to work with up to 16 million colours (24 bits), etc. I do not know if Ademir will ever make those devices, since people did not seem to want them, as far as I know. But it makes it a lot easier for Ademir to add the new ADVRAM board he developed.
- Main ram can be placed on slot 2 or slot 3, using jumpers;
- Connector to led and Turbo switch available;
- The Turbo mode can be selected using the Turbo switch or via basic, using: turbo on (activates turbo mode), turbo off (deactivates turbo mode) and turbo (toggles the state).

Benchmarks
In order to measure the Ciel speed with turbo on and off, I used some small two-line programs in basic. These programs were taken from an old Brazilian book "Colecao de programas para MSX", written by Renato da Silva Oliveira and published by Aleph. The results were are listed in the following table:

Program	normal mode	turbo mode
Cogumelo	41 m 58 s	19 m 39 s
Cesto	19 s	9 s
Astroide	35 s	16 s

You can see below some photos of these programs running. I tried only these three basic programs because I thought it would be a good way to measure the speed of the computer; if you want me to try any other software to measure the speed for you, just send it to me and I will run it on my computer. The photos were taken using BrMSX, a great MSX emulator written by Ricardo Bittencourt. You can download the programs that I used here [2].


Cogumelo.


Cesto.

A short note from the editors:

Unfortunately Mauricio never handed in the finished version of his article; we apologise for that. However, the description he gives of the system was worth publishing.

the next level in hi-speed programming

Assembly line

Assembly programmers sometimes have problems with calculations. Some of them are quite simple as there are instructions for them, others might be a bit more complicated. Adding and subtracting values are common issue with the Z80. Multiplication and division are not. The R800 has an advantage when it comes to multiplication as it has an instruction for it. And introducing the next step in MSX Assembly resources... The MSX Assembly Page.

Albert Beevendorp

Directory

References

1. DIVIWORD.ASC
2. MULUBYTE.ASC
3. MULUWORD.ASC
4. DIVIBYTE.ASC

Mathematics in another way

Many articles have been written about the math-pack which is incorporated in the main-rom of any MSX computer. I found some routines which can be made easily and a lot faster in assembly. The routine `IADD` (i.e. `HL=DE+HL`) can be done with just one assembly instruction: `ADD HL,DE` but to be conform to what `IADD` does an `EX DE,HL` has to be added. The routine `ISUB` (i.e. `HL=DE-HL`) can be done in very much the same way, with the only exception the register exchange is necessary. The routines `IMULT` (i.e. `HL=DE*HL`, the routine published here isn't the fastest multiplication routine though) and `IDIV` (i.e. `HL=DE\HL`) are included in this little source as well and note that `IMOD` (i.e. `HL=DE MOD HL`) uses the same routine as `IDIV`, with the exception the modulus is returned in HL:

ML-listing: `MATHPACK.ASM`

```
IADD:   EX    DE,HL           ; HL = DE + HL
        ADD   HL,DE
        RET

ISUB:   EX    DE,HL           ; HL = DE - HL
        AND   A               ; Reset Carry
        SBC   HL,DE
        RET

IMULT:  EX    DE,HL           ; HL = DE * HL
        PUSH  HL              ; Keep for subtraction later
        PUSH  HL              ; Counter
        POP   BC
IMLTLP: ADD   HL,DE           ; ADD DE to HL BC times
        DEC   BC
        LD    A,B             ; Check if counter reached zero
        OR    C
        JR    NZ,IMLTLP
        POP   DE              ; Result is DE too high
        SBC   HL,DE           ; so correct this
        RET

IDIV:   EX    DE,HL           ; HL = DE \ HL
        LD    A,E             ; Division by zero?
        OR    D
        JR    NZ,IDIVDO       ; Continue of not
        LD    HL,0
        LD    DE,0
        SCF                   ; Set Carry to indicate an error
        RET
IDIVDO: LD    C,L             ; AC = HL
        LD    A,H
        LD    HL,0            ; HL = Modulus
        LD    B,16            ; 16-bit division
        OR    A               ; Start with carry reset
IDIVLP: RL    C               ; AC = AC * 2
        RLA                   ; (set Carry to bit 0)
        RL    L               ; HL = HL * 2
        RL    H               ; (reset Carry again (HL = 0))
        PUSH  HL
        SBC   HL,DE
        CCF                   ; Carry set when Ok
        JR    C,IDIVDN        ; Modulus cannot be >= Value
        EX    (SP),HL         ; Restore Modulus
IDIVDN: INC   SP              ; Get Modulus from Stack
        INC   SP
        DJNZ  IDIVLP          ; Repeat for all bits
; Shift last Carry-bit to divider
        EX    DE,HL           ; DE = Modulus
        RL    C               ; Carry to C
        LD    L,C             ; HL = AC
        RLA
        LD    H,A
        OR    A               ; Reset Carry (valid result)
        RET

IMOD:   CALL  IDIV            ; HL = DE MOD HL
        EX    DE,HL           ; Modulus in HL
        RET
```

Explanations

The source contains 5 major calls which result in 4 routines. The first two are addition and subtraction. Because of the simplicity of these routines explaining how these work would be nonsense now. The routines for multiplication, division and modulus mathematics are explained right now.

Multiplication using `IMULT`

There are a few methods for multiplying one value with another. In a way, two methods are explained in this first Assembly Line. The first method is quite simple to make and understand. I agree on the fact there are methods which are much faster and a second and faster solution will be provided in the next Assembly Line. In here I have taken a rather lazy approach. The routine actually does `DE * HL - DE`. That is because I didn't reset one of the register pairs first. We need to multiply 123 with 234. The routine does the following, a step by step explanation: We make 123 a counter which is usually put in `BC`. The loop does `HL = HL + DE` exactly 123 times, so `HL = 123 + 234`, so `HL = 357`. Then `HL = 357 + 234`, so `HL = 581`. Then `HL = 581 + 234`, so `HL = 815` and after 123 times the result in `HL` would be 28905 because we started with 123 already in `HL`.

So to correct the result we need to subtract 123 from `HL`. `HL` will become 28782.

Division using `IDIV`

Just like multiplication there are a few methods to divide a value from another and in a way, two methods are explained in this first Assembly Line. First, we check whether we are going to attempt an illegal division by zero. According to correct math only 1 division by zero is legal, but not tested in this routine.

Modulus using `IMOD`

For `IMOD` the exact same routine is called like `IDIV` and by swapping `DE` and `HL` the modulus result is written in `HL`. For more information about the actual division routine, read “Division using `IDIV`”.

The source of `IDIV` and `IMOD` can be found right with the following reference [1] as the original file doesn’t have these routines.

More math appeared with the MSX turbo R, which contains two very useful instructions to do very fast multiplication: `MULUB` and `MULUW`. Note that both instructions are located in the R800 processor and the Z80 processor still needs routines to do the multiplication part. `MULUB` needs two 8-bit registers as input and the 16-bit result is output in `HL`. `MULUW` is a bit different. It needs two 16-bit registers as input and the 32-bit result is output to registers `DE` and `HL` (format `DEHL`). Clicking on the instruction the Z80 equivalents of `MULUB A,B` (reference [2]) and `MULUW HL,BC` (reference [3]) can be found. Both routines check whether they are running on Z80 or R800 CPU, so both the R800 instruction as the Z80 routine which does the same multiplication are included in both sources including the check. I am aware of the fact a little bug remains in the Z80 equivalent of `MULUW HL,BC`.

Another way of dividing

Referring to a telephone conversation I once had with Jan van der Meer — who has passed away a few years ago — of a way to divide a number by 6 the easy way, he came up with a different solution to divide two bytes by each other. The source can be found in this reference [4] and in a nutshell, this is how it works: the routine subtracts the divider from the value you want the value from until you reach 0 or a value which lies below the divider. This will be the modulus. The result is the number of times the divider could go into the value.

MSX Assembly Page

A lot has happened between the release of the last two issues of MCCW. A dedicated website for MSX assembly programmers (either current or future ones) has become a quite worthy place with resources, articles, sources, downloads and links to other pages which have info about MSX assembly quite useful to programmers. Together with the MSX Resource Center development forum the MSX Assembly Page will be the most valuable to all MSX assembly programmers. There is also an MSX development IRC channel. Information and the channel location, as well as a CGI:IRC client to it is available at MSX Banzai!.

I hope to meet many developers on the MSX development IRC channel. I also think we can help each other with a variety of programming problems and solutions, faster routines and other MSX development related issues.

A short note from the editors:

This article isn’t finished at all. We think that it lacks a thorough explanation of what is going on. The author did not have time to improve it, though. It may be beneficial to very experienced MSX programmers anyway. Of course I do not need to mention that there will be no Assembly Line part 2 and beyond.

the shooting star children

Kyokugen

Just like in Europe, amateurs started making their own hard- and software for the MSX system after the professionals had stopped developing them. It took M-kai two years and three months to create this vertically scrolling shooting game.

Sander Zuidema

Kyokugen is a game very similar to the popular MSX games Aleste and Zanac. While controlling a small aircraft, the player will have to destroy as many enemies as possible, dodging their attacks. During the game, the weapons of the aircraft will get stronger and several boss-monsters have to be defeated.

First impressions
The full-colour box and manual already reveal with how much care this game was created. The manual clearly explains all aspects of the game, including the story and some fighting tactics. My expectations have risen to a high level so I decide to boot the game.

After the M-kai logo, a small intro demo is shown. A very fast animation visualises the story in the manual. Everything goes very smooth and the music is well-timed. The intro demo ends in the main menu, which has a stunning multi layer-effect. After a while, a small demonstration of the game is given and the intro demo starts again.

Adventure
In adventure mode, we see how the story continues. First we have to choose which pilot we want to be. Each pilot has his own ship, each ship has different weapons. All ships also differ in speed and defensive power. There are no less than twelve different pilots, so it might take some time to figure out which ship suits you the best.

You follow the rest of the story by reading diaries of the crew-members. The English used in the diaries is sometimes a bit hard to follow, but it is a lot better than Japanese. After two diary entries the first stage begins. Our ship is launched from a ship and we are thrown into the action almost immediately. Every enemy that is destroyed, leaves a power-up. Catching these power-ups will increase the experience points. In the right top of the screen is a gauge which represents your experience. Once the gauge is full, your ship will increase a level. Meanwhile, you can catch other options that increase the strength of your weapons or the speed of your ship.

The game contains a lot of small jokes. For instance, one of the enemies seems to be unbeatable, until he flies into a large wall himself. When a level is finished, we get to read one or more diary entries again. This way, the story continues while playing the game. You actually know what you are doing and why you are doing it.

Time attack
There are also two time attack modes in the game. In three or five minutes your only mission is to gain as many points as possible. Unfortunately, you cannot choose your own ship here. The time attack mode is a great place to improve your skills or to have a small competition with some friends. Both the time attack modes and the adventure mode have their own high score lists.

Graphics and music
When first playing the game, I had a bit of difficulty of understanding what was actually going on. Everything on the screen moves very, very fast and the enemies are sometimes a bit blocky. It is a bit in contrast with the very well drawn backgrounds and boss-monsters. Very soon, however, I got used to the style of the game and its graphics. Some of the graphical effects in the game are really well done, and make a very good impression. The game music is made for MSX-MUSIC. Each level has its own soundtrack which represents the atmosphere of the level very well.

Game play
When you see someone play Kyokugen, you would think it is a miracle that he is able to play it. The entire screen is filled with enemies, bullets and options. Enemies keep coming in at high speed. When you play the game yourself, though, you will discover the game has an incredible game play, something I missed in D.A.S.S. The game play is increased even more by the selections you can make in the main menu. You can double the size of the sprites, you can increase the amount of ships you have from three to five and there are seven different difficulty levels. Some of the enemies do entirely different things when the difficulty level is increased. They move faster, they shoot more bullets or they aim better at your ship. This option guarantees many hours of gaming fun for the novice as well as the advanced gamer.

Conclusion
Kyokugen is a very entertaining game. You need to adjust a bit to the graphics first, but when you are used to the game you will be amazed by the game play of this addictive game. Especially if you like games like Aleste, this game can not be missed in your collection. Kyokugen can easily compete with all games of its genre.

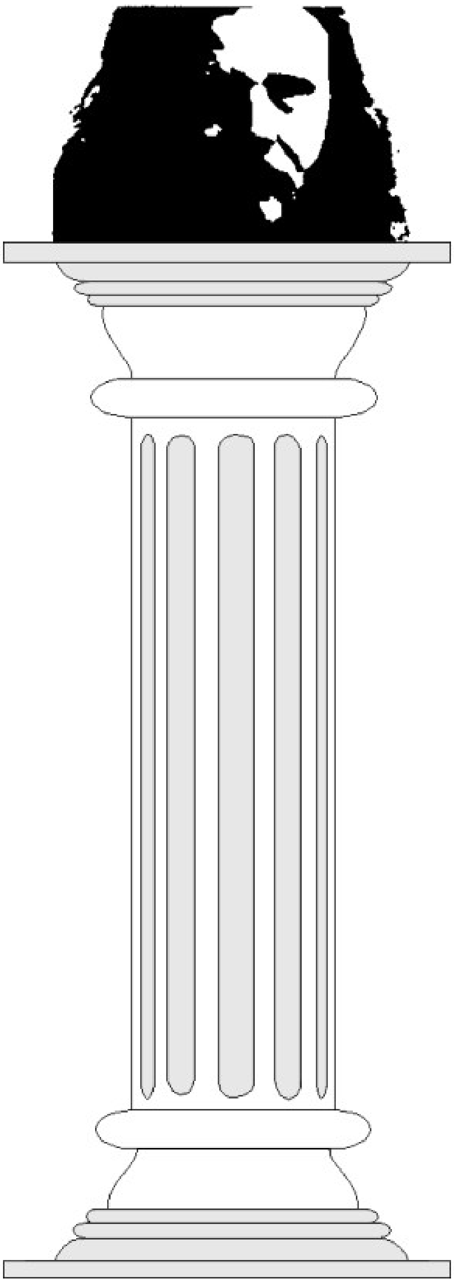
A short note from the editors:

The article you read here is based on the first draft that we recieved from Sander and was not finished entirely...

Unfortunately this article is only available in Dutch.

Wammes' kolom

Impact



Nee, geen verhalen uit de keuken, die houdt u tegoed. Maar wel een misschien persoonlijke terugblik op wat MSX Computer Magazine allemaal betekend heeft voor sommige van zijn lezers.
Zo nu en dan, zeg een keer of acht per jaar, krijg ik er weer één. Een mailtje van een oud-MSX'er. Mensen die ik verder nooit persoonlijk heb gekend, die vele jaren nadat het laatste nummer van MCM onder mijn redactionele verantwoordelijkheid is verschenen, er toch nog eens op terug willen komen. Natuurlijk lees ik die berichtjes graag en ik geef ook altijd trouw antwoord, hoewel soms pas na een week of zo; ik heb tot mijn spij niet echt de tijd om uitgebreid op dergelijke gesprekken in te gaan.

Wat me echter begint op te vallen als ik die mailtjes eens voor mijn geestesoog voorbij laat paraderen, is dat voor bijna al die mensen MSX een heel belangrijk gebeuren in hun leven is gebleken. Dat geldt voor wel meer mensen natuurlijk, mensen die ik wèl kende en waar ik zo nu en dan nog wel eens wat van mag horen: de één is Japans gaan studeren, de ander werd programmeur, weer een ander heeft een eigen bedrijfje opgericht, geënt op die MSX-ervaringen van vele jaren her. Maar naast die groep die ik ben blijven volgen zijn er blijkbaar nog velen die uit MSX de inspiratie hebben opgedaan om hun leven richting te geven.

Vaak schrijven mensen ook letterlijk dat ze, juist door MSX Computer Magazine, een bepaalde richting zijn ingeslagen. En dat zet mij op mijn beurt dan weer aan het denken. Want als ik eerlijk ben, daar hebben we toentertijd nooit bij nagedacht, de mogelijkheid dat een hobbyblaadje zo'n impact zou kunnen hebben op de lezers. Ik spreek niet alleen voor mijzelf, maar ook voor al die anderen die toen aan het blad meewerkten, als ik stel dat wij eigenlijk alleen maar met ons eigen plezier bezig waren. MCM werd niet geschreven voor de adverteerders; dat was een probleem voor de uitgeverij, niet voor ons. En ook over lezersmarketing braken wij ons het hoofd niet, we deden gewoon wat ons zelf wel aardig leek. Veel artikelen werden eigenlijk voor elkaar geschreven...

Om dan, zoveel jaren later, te ontdekken dat dat blaadje voor een aantal van de lezers zo belangrijk is geweest, dat is eigenlijk een beetje eng. Misschien is het maar goed dat we dat niet hebben beseft. Die verantwoordelijkheid was namelijk te groot geweest; de toon en vorm waarmee we zelf zoveel lol beleefden hadden we nooit durven vasthouden als we hadden geweten dat onze schrijfsels zoveel invloed zouden hebben.

Terugkijkend was MSX Computer Magazine een vreemde eend in de bijt. Maar dat wisten we toen niet. Een clubje tamelijke amateurs maakte met veel persoonlijke lol een tijdschrift, dat door een professionele uitgeverij werd uitgegeven. Hoewel, zo professioneel was die eerste uitgeverij nu ook weer niet. Eigenlijk kregen we de vrije hand, zolang de verkoopcijfers maar redelijk waren. Pas toen die toko werd overgenomen door een wel professionele uitgeverij werd het soms knokken tussen redactie en uitgever; de lol werd stukken minder groot. Gelukkig heeft die episode niet al te lang hoeven duren...

Al met al kan ik slechts uitermate tevreden zijn over die periode. Want blijkbaar was onze lol aanstekelijk. En uiteindelijk schuilt in iedere journalist — hoewel die vlag de lading niet dekt, want we waren geen journalisten — ook een soort leraar. En dan te bedenken dat ik er ooit bewust voor heb gekozen geen leraar te worden!

Wammes Witkop

MSX and Laserdiscs

A marvellous amount of extensions have been made for the MSX system. One of the most ambitious extensions was the Laserdisc game system.

Saku Taipale

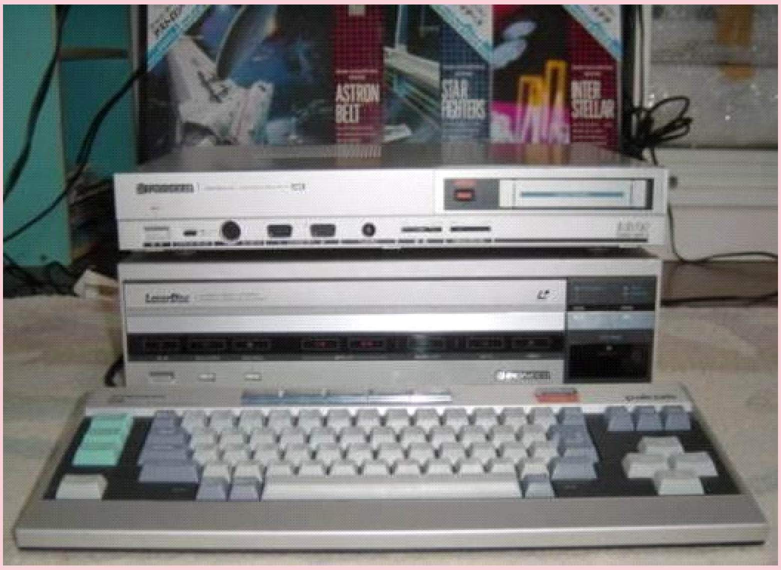
In the early 80's there were lot of companies believing that Laserdisc games would be the next step in the gaming scene. Big projects, mostly for the arcade, were started and several games were released. For home computer systems, Laserdiscs were and are very rare due to the high costs of the equipment and discs. Also, in early stages arcade games revealed that the promises of high quality graphics did not mean Laserdisc games were better than normally programmed video games. Some flaws in the way most games are played were the major cause of this. The Laserdisc standard is quite old: first released in the late 70's. It offers very high quality uncompressed video. This enables adding very high quality backgrounds to a video game, while computer sprites are overlayed. Another way of making a game is animating the player's character in the video images. This creates the problem that scenes are always repetitive and playing is merely just making sequences of movements at the right time. In the end, laserdisc (LD) games never made a big impact on the gaming scene. There are good reasons for this:

1. Arcade machines were very expensive and normally costed more to play than normal video games.
2. Home computers with LD players were very expensive. Pioneer made perhaps the best effort with the MSX standard in Japan.
3. Almost all of the released games were for the NTSC video standard and this prevented penetration into the European market. There were plans to make also PAL games for MSX, but they were cancelled due to poor sales. Only few PAL versions exist for arcade systems.
4. Players did not have a feeling of free control with most LD games.

Needed equipment for playing games

To play Laserdisc games, you need a Laserdisc player, game disc(s) and a computer capable of controlling the player and superimposing the sprites. Another solution is getting a complete arcade game cabinet or all PCB's and the game disc with player from it. Note that every arcade game has different hardware, so for every different game the PCB's and disc is needed. In the MSX case, there are only few computers capable of handling LD's:

- Pioneer Palcom PX-V7 (Japanese, but also a European PX-7 exists, which is a PAL computer)
- Pioneer Palcom PX-V60 (Japanese)
- An MSX in combination with the Pioneer Palcom ER-101
- Victor HC-95 (this one actually uses VHD, a different laserdisc-like standard)
- Sony 900-series (may work with heavy hardware and ROM modifications, but needs really *much* work)



The Pioneer Palcom PX-V7, with the LD-700 Video Disc Player underneath it and some games in the background.

Note that the Sony 900-series does have superimpose and laserdisc control via RS-232, but the built-in for controlling the LD player is not compatible with the Pioneer P-Basic. Also, getting the sound of the game track to the cassette-interface needs some work in the form of hardware modifications. The most difficult problem with the Sony 900 series — and other European MSX computers — is however the PAL-NTSC incompatibility. So far, no one has tried to play games with the Sony 900-series. And it is *not* recommended to try it! There are *no* descriptions of how to do it, only some thoughts how it might be done, so there may be a lot of undiscovered hardware and software problems ahead.

The game background — and in most cases also the player's character — is stored as video on the disc. Also, the disc has two sound tracks: on MSX discs one was used for the game data and the other for sounds. For other systems both tracks were used for audio. MSX is the only known system which used another track for saving data in audio form. All other systems used different media for loading the game program: either it was loaded from floppy disk or in older systems data was stored into ROMs. The way to load the game on MSX is quite similar to loading a game from tape. The `CALL LD` command should be typed in basic and then the computer initialises the player and seeks the starting signal on the audio track. After receiving the signal, the game will be loaded. It would be possible to use arcade discs with MSX as well, but then you need to code the game program yourself.

One may wonder how it is possible to make a game from some played video images, because the player may get 'Game Over' in the middle of a scene. Due to the way of how data is saved on the disc and the random access possible with the players, Laserdisc has some advantages over the other formats. As far as I know every normal LD player has simple functions to access the disc. For example: search for an individual frame on the disc, play backward and forward with variable speeds, jump to chapters programmed onto the disc, show the time, search for a time, mute a selected audio channel, scan the disc frame by frame, etc. And most of these functions can be controlled from the remote controller. Now, when a game player loses his life for example, the game program notifies this and gives a search command to the Laserdisc player to retrieve the correct frame. The frame is searched and played. The game player sees then a nice 'Game Over' animation on the screen. Searching is very fast, maybe 1 second or less.

The most important thing is the video standard the computer uses, because good quality and correct NTSC-PAL transformation is far from easy or cheap. So, the easiest solution for getting a compatible system is to get a Japanese computer. There are European (PX-7) models from Pioneer that use the PAL standard, but they are incompatible with MSX Laserdisc games, because all games produced for MSX are made in NTSC format. A normal MSX (NTSC, Japanese) can perform the same functions as a PX-V7 using a special device attached to the cartridge slot: the Pioneer ER-101 MSX Expansion Processor. It includes the needed circuitry for controlling the player as well as superimposing the picture and also the P-Basic. This expansion is very rare and it does not work even with all Japanese MSX models. It also does not work on MSX machines with disk drive and non-MSX1 models. The reason for this incompatibility is probably the location of P-Basic in the slot. See also the user's guide.

The ER-101 is almost an MSX computer on itself! It sort of makes your MSX a Pioneer PX-7. It includes 'Laser Vision Player Control', 'Superimpose' and 'P-basic (extended MSX-basic)', just like is built in in the PX-7.

Players

For Pioneer you need either an LD-700 (US) or LD-7000 (Japanese) player, or LD-500 (Japanese) or LV-1000. There seem to be a lot of other Pioneer laserdisc players which have I/O-ports similar to the LD-7000, but their compatibility with the PX computers is not tested. The easiest solution is getting an LD-700 or LD-7000, which are tested to work in several systems. Fortunately these two types were popular players both in the United States and Japan for playing movies, so they are still available time by time for example at US eBay and Yahoo Japan.

As said above, the Sony 900 series can also control a laserdisc player. You cannot play games with it, but you can use it for combining Laserdisc images with superimposed images from the computer. You can use any Sony player (maybe also from other manufacturers) which has an RS-232 interface with 900-series. Known suitable models are LDP-1450, LDP-1550, LDP-2000 and LDP-3600. There was once a player specifically for the HB-G900P: the LDP-180P with IF-180 interface. This player and interface seem to be extremely rare nowadays. They are mentioned in the manuals of the G900.

MSX software

A total of 11 games were published for MSX and today they are all rather hard to get. The complete list of games is below. They were never sold in Europe, only in Japan. If you want to find any, then it is better to look for them in Japan, but they are not common there either. The normal price for these games is 5,000-12,000 Yen each, but the rarest titles may go as high as 30,000 Yen. The price depends on the condition as well. Note that the background graphics in all LD games are very good, but the sprites are MSX(1) quality.

Also there were 12 educational software discs published, containing English, Japanese and Mathematic lessons for junior high school students.

Known LD games for MSX	
PG001-12SG	Astron Belt
PG002-12TO	Strike Mission
PG004-11KO	Badlands (by Konami)
SS098-0002	Starfighters
SS098-0003	Umi Yukaba

SS098-0008	Inter Stellar
SS098-0011	Cosmos Circuit
SS098-0019	Esh's Aurunmilla
SS098-0044	Rolling Blaster
LCT001	Mystery Disc 1: Murder, Anyone?
LCT002	Mystery Disc 2: Many Roads To Murder

Prices

The parts for laserdisc systems are not cheap. The computer may cost 60-100 Euro, a player might be 30-60 Euro and game discs about 60-100 Euro each, apart from Konami and some other rarer titles. And these prices are not including shipping costs. For the player alone, shipping costs are easily 100 Euro or more; discs are 20-40 Euro and a computer is 50-70 Euro. But if you are still interested in getting a system, good luck! Some games and computers have been offered lately in eBay by Japanese sellers as well. For players you can try eBay.com; there are cheap players sometimes. And if you buy any player, remember to ask the seller to lock the shipping screw on the unit. At least the LD-700 has it. It is important, because the player may be severely damaged internally if it is not done properly.

Conclusion

If you want keep your money safe, do not buy laserdisc stuff. But if you want to get some really special MSX items, go ahead. Be careful though with parts and ensure that they work and are well packed when you buy them. Benefits of a working system are twofold: a nice rare MSX computer and a laserdisc player which plays nice movies also. The picture and sound quality in discs is still very good, if discs are not too much damaged. Good luck if you decide to buy them!

het ware verhaal...

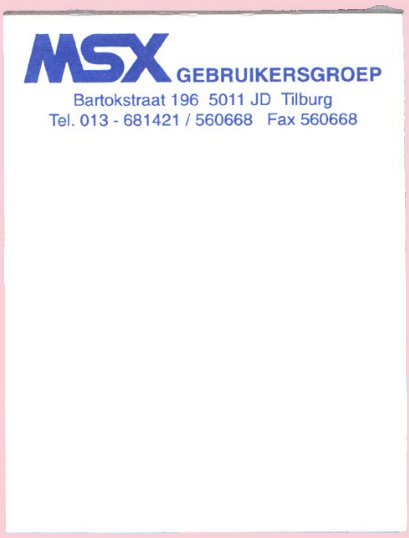
VPB50 MSX notebook - revisited

In MCCM 75 stond op pagina 9 een artikel over een geheimzinnige MSX notebook. Het was een 1-aprilgrap die erg geslaagd was: de MSX gebruikersgroep Tilburg had als relatiegeschenk een klein kladblokje/notitieklappertje met zijn naam laten bedrukken en gaf die aan beursbezoekers en anderen weg. Op de redactie kregen wij er al ruim voor de beurs een paar opgestuurd. Even een leuk verhaaltje schrijven en dat werd lachen. Tot op heden — na dit artikel niet meer hoop ik — blijken er mensen te zijn die nog steeds geloven dat wij iets elektronisch beschreven. Laten we het artikel eens nalopen en zien waar wij u op het verkeerde been trachtten te zetten. Het commentaar is in cursief.

Frank H. Druijff

Inhoud

Het artikel gaat, jawel, over het volgende voorwerp:


MSX GEBRUIKERSGROEP
Bartokstraat 196 5011 JD Tilburg
Tel. 013 - 681421 / 560668 Fax 560668

VPB50...

MSX notebook
Een notitieboekje — kladblokje zo u wil — waar MSX op staat.

Geheimzinnige VPB50
Het enige geheimzinnige is dat u niet begreep waar wij het over hadden.

De MSX Gebruikersgroep Tilburg had de hand weten te leggen op een aantal exclusieve MSX notebooks. Zij kwam er goedkoop aan en op de beurs werden er enige 50 VPB's weggegeven. Een fabrikant kon echter niet worden achterhaald.
Aantal, lekker vaag dus, en exclusief dat klopte wel, want wie gaat nu iets laten bedrukken met de naam van een ander? En een fabrikant kon niet worden achterhaald? Nee, niet door ons, maar de Tilburggroep wist natuurlijk precies waar ze had besteld. Als wij nu niets vragen, kunnen wij 'eerlijk' zeggen dat we het niet konden achterhalen.

Zij schijnen via een dumpcircuit toch op de markt te zijn gekomen.
Tilburg gaf ze gratis weg, als dat geen dump is.
We vermoeden dat de fabrikant, voor er mee op de markt te komen, nog een aantal experimenten wilde uitvoeren en enige onvolkomenheden wegwerken.
pfff
De behuizing is duidelijk onder de maat, lees afwezig.
Ooit een kladblok in doos gezien?
Alleen een stevige bodemplaat.
Ja,van karton om precies te zijn.
En al is die van een milieuvriendelijk materiaal gemaakt, mag dat toch geen volwaardige behuizing heten. Houd hier terdege rekening mee als u ermee aan de slag gaat. Even onbeschermd meenemen in de regen of er een kop koffie over laten omvallen, is desastreus voor dit model.
Papier en karton zijn niet bepaald de grootste vrienden van vocht als ze eenmaal geproduceerd zijn.

Geheugen
Je kan er je (aan)tekeningen op bewaren.
Het geheugen is van een vrij exclusief type dat het best kan worden omschreven als een mengeling van EPROM en RAM. Het is namelijk altijd goed te beschrijven, maar afhankelijk van de schrijfmethode wel of niet wisbaar. Dit vergt van de gebruiker natuurlijk een zekere mate van zelfbeheersing. Het geheugen is te vullen met data die permanent wordt opgeslagen en die niet meer per ongeluk is te verwijderen, maar de totale beschikbare hoeveelheid geheugen is wel beperkt. Door een deel van het geheugen permanent te beschrijven, is de hoeveelheid wisbaar geheugen natuurlijk kleiner geworden.
Als je met potlood schrijft kun je het natuurlijk weggummen en zo heb je de RAM-functionaliteit en als je met pen schrijft is dat niet meer te wissen. Expres werd de term EPROM gebruikt, terwijl het eigenlijk PROM had moeten zijn. Pen is niet wisbaar, maar met de juiste middelen misschien weer wel en dan is EPROM wel goed. We wilden 'duidelijk' aangeven dat het moeilijker was om het te wissen.

Het geheugen kent een indeling zoals bij MSX schermen. Door de term paging denkt men als vanzelf aan de memorymapper, maar die vergelijking gaat toch mank. Bij de memorymapper kiest men vier blokken — vaak foutief met pagina's aangeduid — die samen het totale 'werkgeheugen' vormen. Dit voor de Z80 bereikbare geheugen is 64 kB groot. Bij de schermen werkt men echter in één pagina, naar een tweede kan men eventueel kijken, maar niet in werken. Als er nog twee pagina's zijn — bij scherm 5 en 128 kB VRAM — zijn die onbereikbaar, totdat overgeschakeld naar die pagina's. In dat geval is echter de oorspronkelijke pagina automatisch uitgeschakeld. Het memorymanagement-systeem van de VPB 50 is hiermee goed te vergelijken.
paging, want het ding bestaat toch uit blaadjes? Niet te lang bij blijven stilstaan en gelijk overgaan op memorymapping en wat zaken die daar mee te maken hebben. Mensen lezen iets dat ze kennen, in ieder geval vertrouwd voorkomt, en kunnen dus iets controleren dat wel waar is. De rest van de alinea kan het best worden gelezen als: je kunt de blaadjes omslaan.

Typeaanduiding
U las hiervoor al dat wij het model aanduidden met VPB 50, terwijl de Gebruikersgroep sprak van 50 VPB. Maar net als de fabrikantnaam, ontbreekt ook een duidelijke typeaanduiding. De term VPB, die wij aantroffen, zou net zo goed UP8 of zo kunnen zijn. Wij volgen hierin echter de aanduiding VPB om verwarring te voorkomen. De 50 zetten wij er echter achter! De term is nergens terug te vinden en slaat alleen op de indeling van het geheugen. Om er 50 bij te zetten, vinden wij op zich een goede zaak: duiken er nog meer van deze notebooks op, dan kunnen die best een andere geheugenhoeveelheid hebben en daarom vinden wij de beslissing, om de 50 bij het typenummer te zetten, best goed. Wij vinden echter wel dat de MSX-traditiegetrouw (Sony 700, Philips 8235, NMS 8280) deze getalsaanduiding er achter moet

staan. De processor is alweer zo'n probleem.
Wij hebben noch een Z80, noch een R800 in het MSX'je kunnen ontdekken.
Een hoop onzin, waar wij moeilijk doen over iets dat er totaal niet toe doet, maar de lezer kan wel met ons meedenken. VPB stond er echt op en betekent mogelijk iets als Verenigde Papierfabrieken Brabant en de 50 sloeg op het aantal blaadjes. Geen Z80 of R800 klopt natuurlijk, er zit nooit een processor in een papieren kladblokje.

Praktijk
Toch blijkt hij goed te werken: geen van de door ons ingevoerde instructies gaf problemen, maar toch lijkt deze MSX niet bedoeld om spelletjes te spelen. Een eenvoudig boter, kaas en eieren, oké, maar het display is te traag om leuke actiespellen te doen.
Wetend wat het is, zal dit nu ook wel duidelijk zijn.
Het kleurgebruik zal in de meeste gevallen beperkt blijven tot de blauw-en-wit-standaard, die iedereen die wel eens op F6 drukt zal kennen.
Het was wit papier en het logo van de gebruikersgroep Tilburg was in licht blauw gedrukt.
Het schijnt dat er een adapter leverbaar is, waarmee twee geheugenblokken simultaan te vullen zijn.
Carbonpapier heet die adapter.
Wij waren echter niet in staat hiermee te experimenteren en ook het nut van deze optie ontgaat ons.
Wij hadden geen carbonpapier in huis.
Is een permanent beschreven geheugenblok niet meer nodig is het probleemloos te verwijderen.
Uitscheuren heet die handeling
Het voordeel hiervan is, dat men een overzichtelijker geheugen overhoudt. In de praktijk blijken velen slechts enkele geheugenblokken te gebruiken en de andere zolang mogelijk leeg te laten. Daar steeds maar één memorypage tegelijk is te bekijken/bewerken, is dit inderdaad aanbevelenswaardig. De hoeveelheid data die men op zo'n geheugenpagina kwijt kan, hangt af van de gekozen schrijfwijze. Men kan sterk comprimeren en op die manier veel kwijt, maar ondervindt dan vaak moeilijkheden bij het uitlezen. Niet overdreven comprimeren, luidt ons devies.
In kleine lettertjes schrijven, want dan kun je meer kwijt, maar dat gepriegel is vanzelfsprekend ook moeilijker leesbaar.
Voor degenen die er naast grepen in Tilburg: bedenk dat de beurs volgend jaar misschien al een week eerder is.
Slaat nergens op, de mensen van Tilburg speelden het spel schitterend mee en gaven soms zelfs een notitieblokje weg als troost omdat de beursvoorraad van de VPB50 op was en er alleen nog een paar bij leden thuis lagen.

Frank H. Druijff
uw grijnzende bedrieger

Een aantal gelukkigen ontving in Tilburg een exemplaar van deze MSX notebook. Dat zo'n gegeven paard natuurlijk niet in de bek gekeken moet worden, spreekt voor zich. Wees er dan ook zuinig op en geniet er van zolang hij werkt.

PS. het is niet de enige keer dat wij ons bezondigden aan een 1-aprilgrap en lees de andere artikelen van mijn hand en van vlak voor die datum ook maar eens wat kritischer door.

beyond normal MSX-DOS 2

MSX-DOS 2 version 2.40

After ASCII Corporation stopped with MSX I did not expect a new MSX-DOS 2 version. However, there now is an update available, from an entirely unexpected direction!

Alex Wulms

Directory

Note from the translator

This article is quite old, and therefore might contain some outdated remarks. Out of respect for the author, I have not changed anything, but when necessary I will give comments in italics. In this first comment I would like to start with mentioning that in the meantime there have been some bug fixes and the current version is v2.41. Also, it is not really a new MSX-DOS version, but rather an update of the command-interpreter, `COMMAND2.COM`*.*

Ordering information

MSX-DOS 2.41 used to cost about EUR 10, but has become freeware in 2003. You can download it here. The sources are available here. A special version for FAT16 (by Okei) can be found here.

For the people who read MCCM 77 thoroughly, the news about MSX-DOS version is not really a surprise. In the article about the Multi Mente utilities this new version was already mentioned in a letter of its creator, Fokke Post. Fokke followed MCCM's advise and released the new version as a patch.

Installation

The new MSX-DOS 2 consists out of a new version of the command interpreter `COMMAND2.COM`, and of a new series of (English) help files with in it extensive explanations about all old, changed and new MSX-DOS commands. Thee disk contains an install-program, two data files, a PMA-archive with in it all new help files, a text file with an overview of all modifications and a set of external commands which from now on belong to MSX-DOS 2 by default. With the install-program and one of the two data files the old version of MSX-DOS 2 can be patched to the new version. It is possible to patch MSX-DOS 2 version 2.20 as well as version 2.31, since one of the data files is used for patching version 2.20 and the other datafile belongs to version 2.31. The latest MSX-DOS 2 can easily be installed by copying the correct datafile to a disk with `COMMAND2.COM` and then launching the install-program. It is wise though to save an old `COMMAND2.COM` version separately, in case something goes wrong.

Apart from a new version of the command interpreter you will also get three new external commands. Those the latest versions of `TO.COM`, `TREE.COM` and `RRAMDISK.COM`. Those three programs are already available for some time in the public domain circuit. It is nice though to have them for free with this new MSX-DOS 2 version. Especially for the people who did not have those utilities yet.

Furthermore, there are yet two handy batch files on the disk. It is highly likely there will be more and/or other batch files on the final version because Fokke is still working on MSX-DOS 2.4. One of the batch files is mainly meant to demonstrate the new possibilities and the other sets up several handy *aliases* for you. The use of aliases is one of the new features of this MSX-DOS. The people who already know this term from other operating systems, like UNIX, from which ASCII Corporation already got a lot of inspiration for MSX-DOS functions, will now probably jump in excitement. For the other MSX-freaks an explanation about this oh-so-useful option will follow below.

Changes

MSX-DOS 2 version 2.40 has got a lot of new possibilities compared to the previous versions of MSX-DOS 2. I myself find the progress in user-friendliness from version 2.31 to version 2.40 almost as large as the progress from MSX-DOS 1 to MSX-DOS 2. An overview of the new possibilities can be found in a separate box. Of all those changes I am most pleased with the possibility to use aliases and with the new type of batch files in which commands like `GOTO` can be used. In addition, I find it very useful to be able to access the internal variables and functions of MSX-DOS 2; there clearly is a programmer speaking here. However, the disappearance of some key-combinations like `CTRL-U` for clearing the line are less useful. *(Note: the line can also be cleared by pressing ESC.)*

Aliases

An alias can best be compared with a batch file which consists only out of one line. However, by using the new command-separation mark (`^`) several commands can be put in one alias. The big advantage to batch files however, is that aliases are directly in the computer's ram. That means that they do not have to be loaded from disk first. So in fact you compare an alias with an internal command which is directly executed, without any delay. In this new MSX-DOS 2 an alias can be defined with the new `ALIAS` command. An example of an alias is the following:

```
ALIAS DW = DIR @1 /W
```

If you would now type

`DW A:` MSX-DOS 2 will execute the command

```
DIR A: /W.
```

Another advantage of the aliases is that they can be linked to extensions. An example is the following:

```
ALIAS .TXT = A:\UTILS\TED
```

If you have a file named `REVIEW.TXT` in the current directory, you can start TED with `REVIEW.TXT` as a parameter by simply typing `REVIEW`. Unless TED cannot be found in the given path of course. And if the file `REVIEW.TXT` is somewhere else in the search path, `A:\UTILS\TED` is executed with the complete path of `REVIEW.TXT`.

Internal variables and functions

Internal variables and functions can be used in the same way environment variables can be used in MSX-DOS version 2.31, with `%NAME%` in which `NAME` indicates the variable or function. A number of useful variables is:

`_BG` and `_FG`	screen colours
`_CWD`	current directory
`_CPU`	current processor mode

`_DATE`	current date

A number of useful functions is:

`@ATTRIB`	request a given file its attribute bits
`@DISKFREE`	amount of free space on a disk
`@MID`	comparable with `MID$` in the Basic environment

By combining those internal variables and functions with the new `IFF THEN ELSE ENDIFF` constructions, it is possible to create highly advanced batch files which execute very complex tasks.

Memory

Next to the numerous advantages, the new MSX-DOS also has a small disadvantage: the usage of aliases takes 16 kB of additional memory and the new batch file-format also takes up an extra 16 kB at the moment such a batch file is executed. This is especially a disadvantage for the people who have only 128 kB of memory. Then there is hardly any free memory left for a RAMdisk, MemMan 2.x or other programs which use additional memory. In computer land, memory is often the price you have to pay for the extra possibilities which ease your life. Take a look at the pc, for example, where nowadays even 4 MB is not enough to run modern programs. In that respect MSX still remains a quite cheap computer.

More possibilities and conclusion

Although this update of MSX-DOS 2 already is a huge improvement, I myself see enough space for more possibilities. Especially the batch-language could be greatly enhanced with useful programming constructions like `WHILE`, `FOR` and `CASE`. In this respect Fokke is open to all new ideas, so who knows what will come next if ultimately everybody uses this version of MSX-DOS 2 and passes all ideas they come up with through to Fokke.

Hence my conclusion is that actually everybody using MSX-DOS 2 should buy this new version. The improvements are well worth the money.

Note of the editor: in the MCCM 78 in which the above article was published, there is also an article that shows all differences between MSX-DOS 2.40 and MSX-DOS 2.31. For completeness I will include the English version of that article here too.

New command line editor

- Command history buffer is now 1024 bytes.
- File name completion with `TAB`.
- `CTRL-DEL` clears the buffer.
- `CTRL-INS` puts the current line in buffer without executing it.
- `CTRL-LEFT/RIGHT` goes to begin/end of line.
- `SHIFT-LEFT/RIGHT` goes to previous/next word.
- `SHIFT-DEL` deletes the rest of the line after the cursor.
- `CTRL-RETURN` executes the command without putting it in the buffer.
- The normal control keys — like `^U`: delete line, `^G`: beep — have disappeared.

Length of command line

- You can enter 127 characters at most.
- After expanding aliases a maximum of 255 characters.
- After splitting into commands 127 characters maximum.

New commands

- `FREE [d:]` Shows the amount of free space on drive *d*.
- `BEEP` Sounds a beep.
- `ALIAS [name] [separator] [value]` Sets an alias for a command.
- `RESET` Resets computer.
- `COLOR forclr [bakclr [brdclr]]` Sets the screen colours.
- `INPUT [string] %%envname` Reads a line to an environment variable from keyboard.
- `INKEY [string] %%envname` Reads a key press to an environment variable
- `CPU [n]` Sets or shows CPU mode.
- `CDPATH [[+|-] [d:]path [[d:]path [[d:]path...]]]` Changes search path that is used for commands like `CD`.
- `CDD [d:][path]` Changes to another drive and directory.
- `PUSHD [d:][path]` Pushes current dir to directory stack and changes to new directory specified with *path*.
- `POPD` Pops the directory at the top of the directory stack from the stack and make it the current directory.
- `IFF`, `THEN`, `ELSE` and `ENDIFF` Execute a command conditionally. `IFF` is nestable in 16 levels.
- `HISTORY` Shows the contents of the command history buffer.
- `DSKCHK` Sets or shows the current disk check status.
- `MEMORY` Shows memory usage.

Disappeared commands

- The commands `ERA`, `ERASE` and `RENAME` have been removed.

New environment variables

- The following environment variables have been added: `ALIAS`, `SEPAR`, `EXPAND`, `CDPATH` and `LOWER`.

New default values of environment variables

- `TIME` is now on 24 by default.
- `EXPERT` is now on `ON` by default.
- `PROMPT` is now on `%_CWD%>` by default.

Added internal variables

Internal variables can be queried the same way as environment variables, so using `%name%`. However, they cannot be changed with the `SET` command. The following internal variables are defined:

- `_BG` Current background colour.
- `_BOOT` Boot drive letter, without the colon.
- `_COLUMN` Current cursor column, starting at 1.
- `_COLUMNS` Number of screen columns.
- `_CPU` Current CPU type, Z80 or R800.
- `_CWD` Current directory, format `d:\directoryname`.
- `_CWDS` Current directory, format `d:\directoryname\`.
- `_CWP` Current directory, format `\directoryname`.
- `_CWPS` Current directory, format `\directoryname\`.
- `_DATE` Current date, format dd-mm-yy.
- `_DATEF` Current date, format dd-mm-yyyy.
- `_DIRBUFFER` The directory you were previously and will return to with the commands `CD -` and `CDD -`.
- `_DISK` Current disk drive, without colon.
- `_DOSVER` Current version of COMMAND2.
- `_DOW` Current day of the week (‘Monday’, ‘Tuesday’, etc).
- `_FG` Current foreground colour.
- `_MSXVER` MSX version, MSX-2, MSX-2+ or Turbo-R.
- `_ROW` Current cursor row, starting at 1.
- `_ROWS` Number of screen rows.
- `_TIME` Current system time in the format hh:mm:ss

Added Internal Variable Functions

Internal Variable Functions are similar to Internal Variables. The difference is that they need one or more parameters. They can be used like this: `%name[parameters]%`. The following internal variable functions are defined:

- `@ALIAS[aliasname]`
- `@ASCII[character]`
- `@ATTRIB[filename, attributes]`
- `@CHAR[ASCII value]`
- `@DISKFREE[d:, b|k|m]`
- `@DISKTOTAL[d:, b|k|m]`
- `@DISKUSED[d:, b|k|m]`
- `@EXT[filename]`
- `@FILE[filename]`
- `@FILEATTR[filename]`
- `@FILEDATE[filename]`
- `@FILESIZE[filename]`
- `@FILETIME[filename]`
- `@HEX[number]`
- `@INSTR[number, string1, string2]`
- `@LABEL[d:]`
- `@LEFT[string, n]`
- `@LEN[string]`
- `@LOWER[string]`
- `@MID[string, start, length]`
- `@NAME[filename]`
- `@NEWFILE[filename1, filename2]`
- `@PARSE[filename]`
- `@PATH[filename]`
- `@RIGHT[string, n]`
- `@UPPER[string]`

Miscellaneous changes

- The `DATE` command shows the full name of the day.
- It is possible to set another drive when reloading `COMMAND2.COM`.
- Extra spaces are being printed between the name and the contents of a environment variable when using the command `SET`.
- The command `DEL *.*` now asks ‘Delete all files’ instead of ‘Erase...’.

- There can be more than one command on a line by using the separation character `^`.
- Directories can be changed to without using the `CD` command by just typing in the directory name and adding a `\` at the end.
- The command `CD -` changes to the previous directory.
- You can use the so called aliases in the command line.
- When a directory is not found when using `CD`, all paths that are in the environment variable `CDPATH` will be searched.
- The number of parameters for batch files has been increased. Now a maximum of 255 parameters can be passed using `%n`, instead of only 9.
- In stead of parameters, (internal) variables and functions can be used. Example: `%@DISKFREE[%_BOOT%:]%`. Functions are nestable, as long as the command line is not too long.
- Using the alias mechanism, it is possible to link commands to file extensions. This mechanism is called ‘Executable Extensions’.
- The prompt can be any text. It can be set with the command `SET PROMPT` and can also contain `%name%`. The default prompt is `%_CWD%>`. Note that the ‘greater than’ symbol (>) needs to be double quoted when using it in a `SET` command, or else MSX-DOS2 will process it as if it were a redirection symbol.
- An external command (file) can also contain wild cards. In that case, the first matching file will be executed.
- The `/P` option is now also available for the commands `ALIAS` and `SET`.
- The `MODE` command can now also set the screen height.
- In a batch file one can now also use `%n&` which means ‘all parameters starting at parameter *n*’. The default value for *n* is 1, so `%&` is the same as `%1&`.
- Passing parameters when using aliases is done using `@n`. Similar to batch files, `&` can be added to a parameter of an alias: `@n&`. In a batch file both `%n` as well as `@n` parameters can be passed to an alias.
- When using `IF(F)`, apart from == one can also make use of the operators `EQ` (equivalent), `LT` (less than) and `GT` (greater than) and the logical operators `AND`, `OR` and `XOR`.
- When using `DIR` you can get two-column output by adding the `/2` option.
- There is a new type of batch file with the .btm extension. It can be as big as 16 kB and is loaded to memory in one go. It can contain commands like `GOTO`. The batch language is slowly becoming a real programming language, like this!

Changed memory usage

16 kB of extra memory is used to store the aliases and another 16 kB extra memory is used when a .btm file is loaded. So, in total, 32 kB of extra memory is needed on top of the 32 kB that is used by MSX-DOS2 for internal buffers and things like that.

SCC strikes back

Tunez 2 - Asterix edition

In these days of MSX-Stereo and MoonSound music, it is easy to forget the SCC sound chip. Therefore it is surprising to see a new music disk for this chip.

Jorrith Schaap

Directory

Ordering information

The price is 10 Dutch guilders (EUR 4.54). To order, just contact TeddyWareZ:

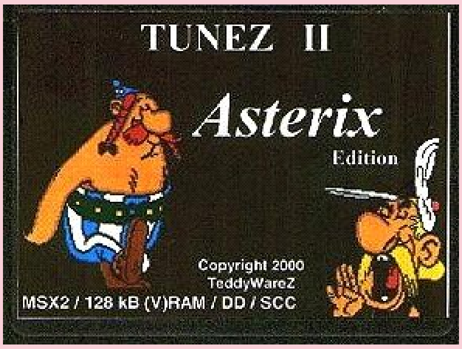
Frederik Boelens
Jan Palachweg 17
9403 JS Assen
the Netherlands

Telephone: +31 592 340197
E-mail: `info@teddywarez.cjb.net`

More information can be found on the TeddyWareZ Homepage.

Note from the editor: nowadays the disk can be downloaded for free. Get it here.

After the release of TeddyWareZ' SCC Blaffer NT, their music program for the SCC sound chip — which can be found in a lot of Konami cartridges — development for this chip has been minimal. To do something about this, and to demonstrate the capabilities of SCC Blaffer NT, TeddyWareZ decided to release a new product with sound support for the SCC chip only. Tunez 2 — the Asterix edition — is almost completely in Asterix & Obelix style, the famous cartoon characters. Not only the replayer and the first intro demo are in this style, but also the disk label contains these two Gallian heroes. This goes even further in the replayer, but more about that later.



Tunez 2 disk label

Once upon a time in Gallia...

The first intro demo consists of some information in typical TeddyWareZ style, which means lots of nonsense written in slang. The font is a bit hard to read, since the letters are black at the top and fading to white at the bottom. This makes the top of the font hardly readable, with the black background. The second part of this intro demo consists of the title screen, with a nice picture of Asterix and a simple logo.

Tunez 2 intro demo

In the meanwhile, the names of the producers will scroll on the screen. Unfortunately this demo cannot be skipped in any way, so you *have* to watch it every time you boot this disk, and that is quite annoying.

The second demo is more spectacular. In this part, the screen is divided in two halves. The top half of the screen shows some green and red coloured bars moving around in some sort of circle behind the TeddyWareZ logo. The lower half of the screen shows a scroll text with the usual TeddyWareZ nonsense, but it also includes some information on what keys you can use in this part. With the cursor keys you can control the speed of the green and the red bars, and this gives a very cool effect!

Panoramix in motion

The replayer is also in Asterix and Obelix style, and shows the druid, Panoramix, cooking up some SCC musics. On the right side of the screen, some information is shown. Here, for instance, the time the song is played is being displayed, but also the frequency and pattern information is shown. Below that, there is a box with the song titles in it. With the cursor keys, you can scroll through the list, the effect you get by scrolling is simple, but it looks good.

Below the song titles the Teleramix can be found. A similar feature can be found in all TeddyWareZ music disks. It is a box that will display information about the song currently playing, and it also contains some nice animations and other effects. Mostly, weird, typical TeddyWareZ texts can be read here. They are quite fun to read, and tell some weird stories, but sometimes some useful information can also be found here. The song info of the first song, for instance, mentions the keys you can use in the replayer. One of the options that is mentioned here is random play, for instance, and there is no mention of it anywhere else.

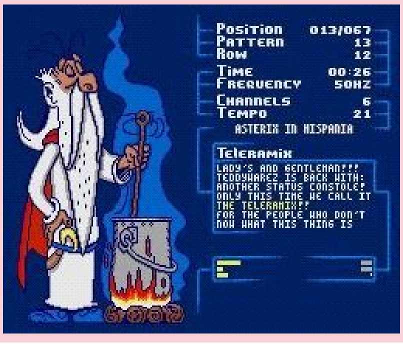
The remaining part of the screen is occupied by an 'equaliser'. This equaliser changes colour from white to grey when a note starts to play, and although it looks good, it is sometimes flickering a bit too much. (*Note from the editor: why do people keep calling these things equalisers? They are just approximations of VU-meters for the different channels of the sound chip! An equaliser is a device that you can use to amplify certain frequency bands in the sound to get it in more balance!*)

Even Gallians do not live forever...

All things end, and so does this music disk. After listening to all songs in the replayer, the only part that is left is the ending demo. In this demo, a vertical scroll is shown with the credits for this product, with a blue and light blue checker board in the background. It is quite a simple demo, also very short. This was a bit disappointing after the good looking intro demos and replayer.

Galliadance!

But... of course I have not mentioned anything about the musics yet. All musics for this music disk were made with TeddyWareZ' own music program for the SCC, SCC Blaffer NT. This program



```
Position    013/067
Pattern          13
Row              12
Time          00:26
Frequency      50HZ
Channels          6
Tempo            21
  ASTERIX IN HISPANIA
```

Teleramix

LADY'S AND GENTLEMAN!!!
TEDDYWAREZ IS BACK WITH:
ANOTHER STATUS CONSTOLE!
ONLY THIS TIME WE CALL IT
THE TELERAMIX!!
FOR THE PEOPLE WHO DON'T
NOW WHAT THIS THING IS

Tunez 2 replayer

enables composers to use not only the five SCC channels, but also the internal MSX sound chip, the PSG. And those two combined can give a very good sound, as Tunez 2 clearly demonstrates.

There are 19 songs present on this music disk, and they are all of good quality, especially when you keep in mind that SCC composing is a bit different from composing for, for instance, MSX-Music. After the SCC musics from Konami, I would say this is the next best thing! The results you get with other music programs for the SCC are not as good, mainly because those programs cannot use the PSG. And, TeddyWareZ shows they know how to use the SCC properly! There are some cover songs on the disk, from Firebird, Vampire Killer, Parodius and Maze of Galious, but the other songs are all original work. Different styles of music can be found, from very relaxing to extremely fast. The cover songs are well done, and it is very nice to hear those PSG tunes rearranged for the SCC. TeddyWareZ' own style can clearly be recognised. Also, it is funny to see that almost all titles, except the titles of the covers, are completely in Asterix & Obelix style as well.

Conclusion

After the SCC seems almost forgotten, it is really nice to see a new product for the SCC, amidst so many products for all the other sound chips. The SCC has a very characteristic sound and, when used correctly, can generate very good songs. The graphics are quite good, the music even better. TeddyWareZ did a good job with this disk, although there are some minor points. For instance, the SCC detection at the beginning of the disk does not work very well, I always had to force the product to use the right SCC slot. Also, the ending is a bit disappointing, with respect to the remainder of the disk. But still, it is a good product well worth your money.

for those who are still scared of NestorBASIC

NestorTIPS for NestorBASIC

This article is about the extension to MSX-basic created by the author: NestorBASIC. It makes a lot of new functionality available to basic programmers. Read ahead for some useful tips.

Nestor Soriano Vilchez

Directory

About the author

I am Nestor Soriano, also known in the MSX world as Konami Man. Maybe you never heard about me, but surely you have seen some of my utility programs for MSX, which can be easily recognised because almost all of them start with the word 'Nestor'. I am a member of the Spanish Club Mesxes and one of the makers of the magazine SD Mesxes; I am also an AAM member and therefore one of the organisers of the MSX meetings in Barcelona. Like most of you, I entered in the computer world with an MSX1 in 1986, but at first I did nothing but playing Konamis; I must admit I am a Nemesis-addict). In 1991 I acquired an MSX2+ and a couple of years after I started to learn assembler. When I changed to MSX turbo R in 1997, I had already developed most of my utilities.

The software mentioned in this article can be downloaded from my homepage: `http://www.konamiman.com/`.

This article is dedicated to the basic extension named NestorBASIC. Many times I hear people complaining about its excessive complexity, but the few people who use it regularly tell me that it is really useful and not so difficult once one is used to it. So with this text I would like to make you lose this fear for the alleged difficulty of programming using NestorBASIC, or at least to show you how useful can it be.

Specifically, I will show you some examples of how NestorBASIC can be used to save basic memory, mainly with the use of one of its most powerful capabilities: the execution of machine code routines stored in mapped memory. This includes various machine code routines and basic sample listings.

NestorPreTer

But first, I need to introduce you to NestorPreTer, because all basic listings are in NestorPreTer format.

Tired of having to remove remarks from my long basic programs when the 'Out of memory' messages started to appear, confused every time I found a `GOSUB 10000` and did not remember what this subroutine did, and thinking about all people who had to call NestorBASIC functions via `usr(x)`, I recently developed NestorPreTer: the MSX-basic pre-interpreter.

Putting it shortly, we can say that while NestorBASIC (NB from now on) is a run-time help tool, NestorPreTer (NPR) is a design-time help tool. In more detail: NPR parses an input text file containing a special formatted basic program, and generates a normal MSX-basic executable — or maybe we should say 'interpretable' — file. This is why we call it a pre-interpreter. And what is this 'special' format? It is the same as the usual MSX-basic format, except that you can:

- Forget line numbers — NPR will generate them automatically — and use line labels when necessary. NPR will convert labels to the appropriate line number when a branch instruction is found. For example: `GOSUB ~ASK_USER` or `RESTORE ~MAIN_DATA`.
- Use as much comment — with the `REM` statement — as you want; NPR will not include it in the destination file.
- Use macros to name constants, variables and code. For example `LOCATE @ROW, @COLUMN` will be converted to `LOCATE X,Y`; or `POKE @INT_HOOK,@RET` will be converted to `POKE &HF349,&HC9`.
- Use indentation to make the entire program more readable.

...so using NestorPreTer, you make very clear, easy-to-read basic code. Isn't that nice? Listing 1 is a NPR format code example which shows how to load NB and do some system initialisation. You can see that we create an array, named `D`, in which we define all the variables we will use in the program; I will explain later why we do this.

As usual with all NestorWare, NPR is free and you can download it from my home page, see the box on the left. You better get it and read the complete manual — do not worry, it is not as long as the one of NB! — because all the listings in this article are in NPR format. Of course, also NB is available on my page.

Basic-listing: `LIST1.ASC`

```
'///
'/// Listing 1: NestorBASIC load / sample of NestorPreTer format code
'/// Summarising: NPR converts this listing into a normal MSX-basic file.
'///

'*********************************
'*                               *
'*  First we define some macros  *
'*                               *
'*********************************

'---  Constants  ---

@define TRUE  -1
@define FALSE  0
@define ON @TRUE
@define OFF @FALSE

'@REQ_SEGS defines the minimum number of ram segments we will need,
'including the five first ones, which are used by NB itself:
'When NB is loaded, the availability of at least REQ_SEGS segments
'is checked; if not available, an error message is prompted and
'NB is uninstalled.

@define REQ_SEGS 6

'---  Variables  ---

'We will centralise all variables in an array named D,
'which will be created after loading NB (see tip 2 for more details).
'Exception is made for loop counters, which must be simple variables.
'Also, we define simple variable @ERROR for NB functions error recovering.

@define NUM_VARS 3   'Total number of variables we will use
                           '(except @ERROR and loop counters)
@define ERROR e
@define NUM_SEGS d(0) 'Total number of available segments
```

```
@define FILE_HANDLE d(1)         'Identifier for any open file
@define ANY_DATA d(2)
'...
'define here all variables you will use
'and do not forget to set NUM_VARS appropriately
'...

'We define also all the loop counters we will use
'It is better to use b, b1, b2... so we can easily initialise all
'with just a DEFINT b

@define LOOP b
@define LOOP2 b2

'---  NestorBASIC functions ---

'Using macros to name NB functions you do not need to remember
'the numbers for all functions (and vice versa, when you find
'a function call in your listing, you do not need to remember which is
'the function for that number)

@macro     NB_UNINST e=usr(0)    'Uninstalls NestorBASIC
@macro     NB_INFO e=usr(1)      'Gets info about NestorBASIC
@macro     R_SEG      e=usr(2)   'Reads a byte from a segment

'...
'define here all functions you will use, or define all
'in an external macros file (this is better).
'In my HP you can find a file with all NB functions defined as macros.
'...

'---   Samples of useful macros  ---

'Clear keyboard buffer:

@macro     CLEAR_KEY_BUFFER defusr1=&h156: @ERROR=usr1(0)

'Check if ENTER is being pressed (if @PRESS_ENTER then...)

@macro     PRESS_ENTER (peek(&HFBEC) and 128) = 0

'Uninstalls NB freeing basic memory, and ends.
'BEWARE: Do not uninstall NB from inside a turbo block!

@macro     FINISH     p(0)=@YES: @NB_UNINST: end

'--- To identify the listing ---

@remon

'list1.bas

@remoff


'**********************
'*                    *
'*  NestorBASIC load  *
'*                    *
'**********************

~          maxfiles=0:                                'This saves about 250 bytes
           keyoff: screen 0: width 80:
           color: color 15,0,0:
           ?"--- Loading NestorBASIC... ---": ?:
           bload"nbasic.bin",r:
           defint @ERROR:                   'NB error variable
           @ERROR= 0:
           if p(0) >4 then
                 goto ~LOADOK    'No error is P(0) is at least 5
           else
                      ?"ERROR: ";

'*** Error? Then show message and finish.

~          if p(0)=0 then
                      ?"No mapped memory or only 64K found!":
                      end

~          if p(0)=1 then
                      ?"Disk error when loading NestorBASIC!":
                      end

~          if p(0)=2 then
                      ?"No free segments!":
                      end

~          if p(0)=3 then
                      ?"NestorBASIC was already installed.":
                      @NB_INFO:
                      goto ~ALR_LOAD

~          if p(0)=4 then
                      ?"Unknown error.":
                      end

'*** Jumps here if NestorBASIC loaded successfully.

~LOADOK:   ?"NestorBASIC loaded successfully!":
                      ?"Available segments:"; p(0): ?

'*** Jumps here if NestorBASIC was already loaded.

~ALR_LOAD

'*** Checks if we have at least REQ_SEGS segments, else ends.

~          defint d: dim d(@NUM_VARS):      'Creates variables array
           if p(0)< @REQ_SEGS then
                      ?"ERROR: Not enough free segments!":
                      ?"I have"; p(0) ;" segments and I need at least";
                      @REQ_SEGS; "segments.":
                      @FINISH
           else
                      @NUM_SEGS= p(0)
'...
'Put your program from here. Suggestion for the beginning:

_turbo on (p(), d(), e)
dim f$(1): defint @LOOP: @LOOP=0: @LOOP2=0
```

NestorTIP 0: general considerations

Imagine that you are an assembler programmer — I suppose you actually are — and you need to make an editor for a game you are developing, with maps, sprites, etc. For such a purpose the most practical way is to use basic, because you do not have a real need for speed and coding it in machine code directly could take you as much time as the game itself.

But soon you bump into the eternal problem of basic programs: the lack of memory. We have a 128, 256, even 1024 or 4096 kB machine, but we can use only 23 kB, and if we use Turbo-basic it is even worse: just 10 kB!

NB solves this problem partially: we cannot make bigger programs, but we can use mapped memory to store data. So from now on do not forget this: *do not store in your basic program whatever you can store in mapped memory!* So from now on it is 'forbidden' — you choose the prohibition level — to put in your programs:

- Strings. It is very easy to spend, without realizing it, 3 or 4 kB with just `PRINT "Welcome to my amazing program which is the best one in the world"`, `PRINT "Please enter the name of the desired filename"` and so on.
- Data for coordinates, key combinations, etc... For example "`IF X>34 AND X<100 THEN 1000 ELSE IF X<34 THEN`..."
- `DATA`s. Making a loop for reading data from mapped memory is almost as easy as making a `READ` loop.

We can also save basic memory — and gain speed as a side effect — if we perform some tasks in assembler. For example, read the mouse and cursor keys and then check if the pointer does not move beyond the screen limits. Of course these assembler routines will also be placed in mapped memory so no basic memory is spent.

And maybe you think 'I also save memory if I do not use comments'. If you use NestorPreTer you do not need to worry about this, otherwise you must of course minimise the number of comments and their length in order to save memory.

Last but not least: now that you will use mapped memory, take paper and pencil and draw a map of all the segments, i.e., what you will put in them and on what address. This will make the further coding process much easier. Years of experience have taught me this.

And after these introductory words, let us start with some more detailed tips.

NestorTIP 1: sharing variables

This is not exactly a tip for saving memory, but it is also very useful.

You already know, I hope, that NB has functions for storing basic programs in ram segments, and to execute them without losing current variables. But maybe for any reason you cannot — due to a lack of segments, for example — or do not want to use these functions. Well then, we are not lost yet: there is a way to load another basic program in the usual way — `RUN"PROGRAM"` — without losing variables. How? This is what I will explain now.

The tip is just to save all variables in mapped memory, then load the new program, and then recover the variables. It sounds easy... and actually it is, if we have all variables in one single array; that is why we define array `D` in listing 1, as I explained in the introduction text.

More detailed explanation: if we put all variables in one array, then all of them are stored consecutively in memory and therefore we can copy all of them easily to any ram segment, using the NB function for memory block transfers, which is function number 10. But how do we know where the array is stored in basic memory? That is what the basic instruction `VARPTR` is for! Add the fact that basic memory has segment number 255 assigned when using NB functions and we have listing 2a.

Once we have saved all variables, we load the new program with a common `LOAD`, we create again array `D`, we recover variables with the same NB function, swapping source and destination parameters and we can continue our task. See listing 2b.

Basic-listing: LIST2A

```
'///
'/// Listing 2a: storing all variables (array D) in a mapped memory segment.
'/// Execute it before loading another basic program.
'/// We suppose that all variables are of integer type.
'///

@macro LDIRSS e=usr(10)         'Memory block transfer function

'Segment and address where variables will be saved:
'define it as you want, I just use example values.

@define DATA_SEG 6
@define DATA_DIR &H100

'Now we save variables. We suppose that NUM_VARS is appropriately defined,
'for example with listing 1.

~          p(0)= 255: 'basic memory segment number
           p(1)= varptr(d(0)):   'Address in basic memory of the array
           p(2)= @DATA_SEG: 'Destination for transfer
```

```
          p(3)= @DATA_DIR:
          p(4)= @NUM_VARS * 2:  'Block size: each integer var = 2 bytes
          p(5)= @NO: 'Or @YES, you set it as you want
          p(6)= @NO: 'Idem
          @LDIRSS:

          run "nextprog.bas"        'Now we can execute another basic program
```

Basic-listing: LIST2B

```
'///
'/// Listing 2b: Recovering the variables previously saved with listing 2a.
'/// Execute it at the beginning of the program which must recover
'/// the variables from the "parent" program.
'///

@remon
'nextprog.bas
@remoff

@macro LDIRSS e=usr(10)        'Memory block transfer function

'Segment an address where variables are saved.
'Again you can define it as you want, but of course you must use
'the same values used in listing 2a!
'And again, we suppose @NUM_VARS already defined

@define DATA_SEG 6
@define DATA_DIR &H100

'First we define array P again so we can use NB functions.

~          defint p: dim p(15): define @ERROR: @ERROR=0:

'Now we create again data array D. Again, we suppose
'@NUM_VARS already defined.
'Suggestion: define DATA_SEG, DATA_DIR and NUM_VARS in an external macros
'file, and use this file when processing with NPR both parent and child basic
'programs.

          defint d: dim d(@NUM_VARS):

'Now we have D again, we recover variables:

          p(0)= @DATA_SEG:
          p(1)= @DATA_DIR:
          p(2)= 255:
          p(3)= varptr(d(0)):
          p(4)= @NUM_VARS * 2:
          p(5)= @NO:
          p(6)= @NO:
          @LDIRSS

'...and from here, life goes on:

_turbo on (p(), d(), e)
dim f$(1): defint @BUCLE: @BUCLE=0: @BUCLE2=0

'etc...
```

NestorTIP 2: when coordinates are bothering

Let us continue with the example of the editor for the game. Surely it will have a graphical environment including icons and mouse control. Then, when a mouse button click is detected, you must find out which icon the pointer is pointing at. How do you do this?

The normal way — actually I cannot think of another — is to have a table with start and end coordinates for each icon and with a given current pointer coordinates, we scan the table checking if these current coordinates are or are not inside the coordinates range for each icon. That is, if an icon is located from (X0,Y0) to (X1,Y1), we must check if X0<X<X1 and Y0<Y<Y1. Simple and easy.

Of course you can do this in plain basic, storing the table of coordinates in a `DATA` line. But if we use NestorBASIC we can use assembler, which is faster, easier and consumes less memory.

Let us see how. In a ram segment we will save an assembler routine, together with the coordinates table for all the icons we have on the screen. When calling this routine with function 59, we just pass current mouse coordinates as parameters, then the table will be scanned, and we get the icon number as a result. It saves an incredible amount of memory and we also gain speed.

So here you have three more listings. Listing 3 is a universal routine for loading a file into a segment; we will use it to load the ML routine and the icons table. I will also use it in the other tips. Listing 4a is the table scanner in assembler and listing 4b is some basic sample code which calls the assembler routine previously loaded.

Basic-listing: LIST3

```
'///
'/// Listing 3: Loading a entire file in a segment
'/// (file must have a maximum length of 16K, of course)
'/// NOTE: @ERROR and @FILE_HANDLE are defined in listing 1
'///

@macro     F_OPEN     e=usr(31)  'Open a file
@macro     F_CLOSE e=usr(32)     'Close a file
@macro     F_READ     e=usr(33)  'Read from a file

'Segment and address where we will load file, define it as you want.
'FILE_SIZE is the amount of bytes we will try to read from the file.

@define FILE_SEG 6
```

```
@define FILE_DIR 0
@define FILE_SIZE 16384-@FILE_DIR

'Name and path for file to be loaded

@define FILENAME "c:\routines\anything.bin"

'We open file, if error, we jump to a routine ~ERROR,
'which we suppose defined anywhere.

~List_3:
           F$(0)= @FILENAME:
           @F_OPEN:
           if @ERROR<>0 then ~ERROR else
           @FILE_HANDLE= p(0)

'Now we try to read 16K from file. If we obtain error 1 or 199 (for DOS 1 and
'DOS 2 respectively) we ignore it, because this error means just "End of
'file", that is, file is smaller than 16K

~          p(0)= @FILE_HANDLE:
           p(2)= @FILE_SEG:
           p(3)= @FILE_DIR:
           p(4)= @FILE_SIZE:
           p(6)= @NO:
           @F_READ:
           if (@ERROR<>0 and @ERROR<>1 and @ERROR<>199) then ~ERROR

'All done, now we just close the file.

~          p(0)= @FILE_HANDLE:
           @F_CLOSE:
           return
```

ML-listing: LIST4A

```
;--- Listing 4a: Icon scanner in assembler
;    With a given pair of coordinates, it scans a coordinates table
;    corresponding to icons positions, and it returns the icon number
;    which contains the given coordinates.
;    Input:   P(3) = BC = X coordinate
;                P(4) = DE = Y coordinate
;    Output:  P(5) = HL = Found icon identifier (-1:none)

           org        #8000

           push       bc,de      ;So P(3) and P(4) are not modified
           call       CHKICON
           pop        de,bc
           ret

CHKICON:   ld         ix,TABLE_ICON

LOOPICON:   ld         a,(ix)
           cp         #FF
           jr         z,ENDICON

           ld         l,(ix+1)
           ld         h,(ix+3)
           ld         a,c
           call       RANGE
           jr         nz,NOICON

           ld         l,(ix+2)
           ld         h,(ix+4)
           ld         a,e
           call       RANGE
           jr         nz,NOICON

YESICON: ld            l,(ix)
           ld         h,0
           ret

NOICON:    inc        ix
           inc        ix
           inc        ix
           inc        ix
           inc        ix
           jr         LOOPICON

ENDICON:   ld         hl,-1
           ret

;--- Subroutine: RANGE
;      Checks a byte for being or not inside a given range
;    INPUT:            H = Upper value of range (inclusive)
;                      L = Lower value of range (inclusive)
;                A = Byte
;    OUTPUT:           Z = 1 if inside range (Cy = ?)
;                      Cy= 1 if above range (Z = 0)
;                      Cy= 0 if below range (Z = 0)
;    MODIFY:           AF

RANGE:     cp         l          ;Smaller?
           ccf
           ret        nc

           cp         h          ;Bigger?
           jr         z,R_H
           ccf
           ret        c

R_H:       push       bc         ;=H?
           ld         b,a
           xor        a
           ld         a,b
           pop        bc
           ret


;--- Icons coordinates table
;    Format: identifier + start x + start y + end x + end y
;    (1 byte each one)

;    NOTE: identifier #FF is reserved for the end of table mark (mandatory)
;    Assuming that all icons are placed in a rectangular array starting with
;    base position (BX, BY), and icons have a size TX x TY, we can use
;    the macro "icon":

BX:        equ        0          ;Sample values
BY:        equ        200
TX:        equ        11
TY:        equ        11

icon:      macro      @num,@xi,@yi
           db         @num,BX+@xi*TX,BY+@yi*TY,BX+@xi*TX+TX-1,BY+@yi*TY+TY-1
           endm

;Sample: 5 x 2 icons table:
;01234
;56789

TABLE_ICON:
_0:        icon       0,0,0
_1:        icon       1,1,0
_2:        icon       2,2,0
_3:        icon       3,3,0
_4:        icon       4,4,0
_5:        icon       5,0,1
_6:        icon       6,1,1
_7:        icon       7,2,1
_8:        icon       8,3,1
_9:        icon       9,4,1
END:       db         #FF
```

Basic-listing: LIST4B

```
'///
'/// Listing 4b: Sample of the use of the icons scanner from NestorBASIC,
'/// in a short, useful, fast and elegant way. (-v-)v
'///

@macro     ASSEMB_EXE  e=usr(59)

'First we load the file containing the routine in any segment.
'NOTE: Load address (@FILE_DIR) must be the same where the routine has been
'assembled (ORG directive on listing 4a):

@define ORG_DIR &H8000

@define FILE_SEG 6
@define FILE_DIR @ORG_DIR
@define FILE_SIZE 16384-(@ORG_DIR-&H8000)

~          gosub ~List_3

'...
'... Any code in which, for example, we detect a mouse click
'... in coordinates (X,Y)
'...

'Now we get the clicked icon number depending of the coordinates:

~          p(3)= x:
           p(4)= y:
           gosub ~CALL_ASSEMMB:
           if p(5)= -1 then ~NO_ICON
           else on p(5) gosub... 'Or: goto ~PROCESS_ICON

'Subroutine for calling the assembler routine

~CALL_ASSEMB:
           p(0)= @FILE_SEG:
           p(1)= @FILE_DIR:
           @ASSEMB_EXE:
           return
```

NestorTIP 3: restore things to the state in which you found them
This is not a very impressing tip and it does not use assembler routines at all, but it can be useful.

Let us suppose your program is using text mode. It sets `SCREEN 0`, `WIDTH 80` and black and white colours. It also performs a `KEY OFF` and switches off the key click sound. You do all of this at the beginning of the program and it works, but... what happens when your program finishes? The screen state remains as you have set it, that is, the initial state is not restored.

Solution: save the screen's initial state, which we will get by looking at some system variables, before setting the desired screen mode (listing 5a). Then, when our program finishes, we can retrieve these initial settings (listing 5b). And of course, we will save this state in mapped memory, so no single byte of basic memory is lost.

Basic-listing: LIST5A

```
'///
'/// Listing 5a: saving screen state in a ram segment
'///

@macro     R_SEGI     e=usr(3)   'Reading from a segment with auto increment
@macro     WSEGI      e=usr(7)   'Idem for writing

'Segment and address where we will save state
'As usual, you must define it as you want

@define STA_SEG 6
@define STA_DIR 0

@define VARIABLE v    'For a loop

'System variables we will save

@define LINLEN &HF3B0'Current WIDTH
```

```
@define CRTCNT &HF3B1'Number of lines on screen
@define CLIKSW &HF3DB'Key click sound, (0=no, other=yes)
@define CNSDFG &HF3DE'KEY ON (0) / OFF (other)
@define FORCLR &HF3E9'Text color
@define BAKCLR &HF3EA'Back color
@define BDRCLR &HF3EB'Border color
@define SCRMOD &HFCAF'Current SCREEN

'We save state with a simple loop
'(OK, I said before that it is better to not use DATAs in order to save
'memory... but this is just a sample! Besides you surely will put this code
'in a short initialisation program rather than in the main program)

~          p(0)= STA_SEG:
           p(1)= STA_DIR:
           restore ~VARIABLES

~          read @VARIABLE:
           if @VARIABLE<>0 then
           p(2)= peek (@VARIABLE):
           @W_SEGI:
           goto ~~

~DONE: 'Here life goes on...

..

'Here you have the used system variables, be careful to not modify
'the variables order in the DATA line, else listing 5b will not work!

~VARIABLES:
           data       @SCRMOD, @LINLEN, @CRTCNT, @CLIKSW, @CNSDFG,
                      @FORCLR, @BAKCLR, @BORCLR, 0
```

Basic-listing: LIST5B.ASC

```
'///
'/// Listing 5b: Restoring initial state, saved with listing 5a
'/// NOTE: execute this code out of turbo blocks in order to avoid
'/// problems with some incompatible instructions
'///

~          p(0)= STA_SEG:
           p(1)= STA_DIR:

           @R_SEGI:   'SCREEN
           screen p(2):

           @R_SEGI:   'WIDTH
           width p(2):

           @R_SEGI:   'Lines in screen
           poke @CRTCNT, p(2):

           @R_SEGI:   'Key sound
           if p(2)=0 then screen ,,0
           else screen ,,1

~          @R_SEGI:   'KEY ON/OFF
           if p(2)=0 then keyon
           else keyoff

~          @R_SEGI:   'Colours
           color p(2):
           @R_SEGI:
           color ,p(2):
           @R_SEGI:
           color ,,p(2)
```

NestorTIP 4: press any space key

This tip is similar to tip 2. If we use assembler to detect icons, why we cannot we do same to detect key pressing? The idea is simple: instead of using `INKEY$`, `INPUT$` and similar basic instructions, we save in any segment a table with the keys we are interested in plus an assembler routine which will detect what key or key combination is being pressed, returning its associated identifier.

Advantages: as in the case of the icons scanner, we save basic memory — one single `USR` is enough to scan all the keys in the table and detect the key/combination being pressed — get more speed and make it easy to detect 'difficult' keys such as `SHIFT`, `CTRL`, `SELECT`, `ESC`...

Maybe you are asking yourself how a key table can be stored. Well, as you may already know, when working in assembler, the keyboard is seen as an 11×8 array, in which every key has a row number and a column number assigned: see Figure 1.

Figure 1: Row and column number for each key

	7:	6:	5:	4:	3:	2:	1:	0:
0:	7	6	5	4	3	2	1	0
1:	;	]	[	\	=	-	9	8
2:	B	A	ACCENT	/	.	,	‘	‘
3:	J	I	H	G	F	E	D	C
4:	R	Q	P	O	N	M	L	K
5:	Z	Y	X	W	V	U	T	S
6:	F3	F2	F1	CODE	CAPS	GRPH	CTRL	SHIFT

7:	RET	SEL	BS	STOP	TAB	ESC	F5	F4
8:	RIGHT	DOWN	UP	LEFT	DEL	INS	HOME	SPACE
9:	4	3	2	1	0	/	+	*
10:	.	,	-	9	8	7	6	5

The assembler routine we will use (list 6a) in order to simplify things, detects just the pressed keys, or keys pressed together with SHIFT and/or CTRL. Therefore, for each key we have the following in the table: an identifier, SHIFT/CTRL required combination, row and column. Apart from this, the operating system itself scans the keyboard at each clock interrupt (50 or 60 Hz) and stores information about the status for every key in the system's work area. So, we just need to read this system area zone in order to know if a given key is being pressed or not.

Last but not least: in addition to the key or key combination currently being pressed, this routine also returns the key or key combination that was being pressed in the previous call. This is very useful to detect a key press only once, even if the key is still pressed: we consider the key pressed only if it is pressed currently *and* it was not pressed in the previous call. Alternatively, we can also detect a key release: a key is not being pressed but it was pressed in the previous call.

Listing 6b is a basic sample of the use of the routines of listing 6a.

ML-listing: LIST6A

```
;--- Listing 6a: Checks if any key of the keys table is pressed
;    Input:   -
;    Output:  BC = P(3) = Key/combination currently pressed (-1 = none)
;                DE = P(4) = K/C pressed in the previous call

          org        #8000

;NEWKEY is the system work area where keyboard status is saved by OS.
;It is 11 bytes long, each byte is a row, and for each byte, each bit
;is the state for the corresponding column (1=not pressed, 0=pressed)

NEWKEY: equ          #FBE5

CHKEY:    ld         ix,TABKEY

LOPKEY: ld a,(ix)
          cp         #FF
          jr         z,NOKEY

          ld         e,(ix+2)
          ld         d,0
          ld         hl,NEWKEY
          add        hl,de
          ld         a,(hl)     ;A = Row state
          cpl

          ld         b,(ix+3)
          inc        b
LOPFIL: srl          a
          djnz       LOPFIL
          jr         nc,NEXTKEY ;Cy = Key state (1=Pressed)

          ld         a,(NEWKEY+6)
          cpl
          and        3
          cp         (ix+1)
          jr         nz,NEXTKEY ;Checks for SHIFT and CTRL

OKKEY:    ld         c,(ix)     ;Combination pressed?
          ld         b,0
          ld         de,(OLDKEY)
          ld         (OLDKEY),bc
          ret

NEXTKEY:  inc        ix         ;Next combination
          inc        ix
          inc        ix
          inc        ix
          jr         LOPKEY

NOKEY:    ld         bc,-1      ;No key or combination pressed
          ld         de,(OLDKEY)
          ld         (OLDKEY),bc
          ret

OLDKEY: dw -1        ;For storing previous combination

;--- Keys table
;    Format: identifier + SHICT + row + column
;    (1 byte each one)
;    SHICT = &B000000CS
;    C=1 if CTRL pressed is required
;    S=1 if SHIFT pressed is required
;    In other words:
;    SHICT=0 for key alone
;    SHICT=1 for key + SHIFT
;    SHICT=2 for key + CTRL
;    SHICT=3 for key + SHIFT + CTRL
;    See figure 1 for the corresponding row and column for each key

;NOTE: identifier #FF is reserved for end of table mark (mandatory)

TABKEY: ;Example table
          db         0,0,7,2    ;ESC
          db         1,2,7,2    ;CTRL+ESC
          db         2,3,7,2    ;CTRL+SHIFT+ESC
          db         3,1,7,2    ;SHIFT+ESC
          db         4,0,6,5    ;F1
          db         5,3,6,3    ;CTRL+SHIFT+CAPS
          db         #FF
```

Basic-listing: `LIST6B`

```
'///
'/// Listing 6b: Detects if any of the keys on the example table in listing 6a
'/// is being pressed (pressing and releasing is detected)
'///

@macro    ASSEMB_EXE  e=usr(59)

'First we load the file with the routine and the table.
'See note about ORG_DIR in listing 4a

@define ORG_DIR &H8000

@define FILE_SEG 6
@define FILE_DIR @ORG_DIR
@define FILE_SIZE 16384-(@ORG_DIR-&H8000)

~         gosub ~List_3

'Some initialisation

~         screen 0,,0:
          width 40:
          color 15,0,0:
          keyoff

'We create a strings array to show information about the key being pressed.
'OK, OK, I said you "do not store strings in the basic program", but...
'THIS IS JUST A SAMPLE!!

@define STRINGS c$

~         dim @STRINGS(6):
          @STRINGS(0)= "ESC":
          @STRINGS(1)= "CTRL+ESC":
          @STRINGS(2)= "CTRL+SHIFT+ESC":
          @STRINGS(3)= "SHIFT+ESC":
          @STRINGS(4)= "F1":
          @STRINGS(5)= "CTRL+SHIFT+CAPS"

'Infinite loop for detect keys and show which one is pressed

@macro    BEEPEA    beep:beep:beep

~INFINITE:
          gosub ~CALL_ASSEMB:

          if p(3)=-1 and p(4)<>-1 then   'A key is released?
          ?"Release ";@STRINGS( p(4) );" !!": @BEEPEA else

          if p(3)<>-1 and p(4)=-1 then   'A key is pressed?
          ? @STRINGS( p(3) ): @BEEPEA

~         cls:
          goto ~INFINITE

'Subroutine for calling the assembler routine

~CALL_ASSEMB:
          p(0)= @FILE_SEG:
          p(1)= @FILE_DIR:
          @ASSEMB_EXE:
          return
```

NestorTIP 5: BIOS can also help you

We have spoken a lot about the use of our own assembler routines, but we forgot that we have also a good set of ready-to-use routines in the ROM of our machines: the BIOS. Of course you can use the BIOS via the `(DEF)USR` instruction; you do not need NestorBASIC at all. However, if you use `USR` you cannot set the input registers, nor look at the output registers, while if you use NestorBASIC function 58 you can.

Most of the BIOS routines are not more than assembler versions of basic instructions, but there are some that will be very useful for us. For example `CHGCPU` (&H0180) and `GETCPU` (&H1083), which respectively change and get the current processor on the Turbo-R. See listing 7.

Basic-listing: `LIST7`

```
'///
'/// Listing 7: Getting and setting the processor on the Turbo-R using BIOS
'///

@macro    BIOS      e=usr(58)  'Function to execute BIOS routines

@define GETCPU      &H180
@define SETCPU      &H183
@define Z80         0          'CPU modes
@define R800ROM 1
@define R800DRAM    2

'MSXVER= MSX version: 0=MSX1, 1=MSX2, 2=MSX2+, 3=MSX Turbo-R

@macro    MSXVER    peek (&H2D)

'Getting current CPU: after calling GETCPU, we have AF in p(2); current
mode is stored in A, therefore:

~         if @MSXVER<3 then ~NO_TURBOR else
          p(0)= 0:
          p(1)= @GETCPU:
          @BIOS:

          p(2)= (p(2)\256) and 255: 'Note: integer division (inv. bar or yen)
          if p(2)= @Z80 then...
          else if p(2)= @R800ROM then...
          else...

'Setting CPU: to set 0, 1 or 2 in A, we put &H8000, &H8100 or &H8200 in AF,
'that is, in p(2). The 8 is to update the processor led appropriately; if we
```

```
'change it into a 0, led will not change.

~          if @MSXVER&lt;3 then ~NO_TURBOR else
           p(2)= @R800ROM: 'or any other
           p(2)= p(2)*256:
           p(2)= p(2) or &H8000:'If we want led update

           p(0)= 0:
           p(1)= @SETCPU:
           @BIOS
```

More examples: `CHGCAP` (&H0132) allows you to change the CAPS led state (see listing 8). Or `GETPLT` (&H0149) in the subrom, which allows you to get the palette data for a given color number (list 9). As you can see there are a lot of possibilities: it is up to you to scan the BIOS routines listing in order to find the routine you were searching for.

Basic-listing: LIST8

```
'///
'/// Listing 8: Changing CAPS led state with CHGCAP
'/// Use: set value 0 (led ON) or any other (led OFF) before calling CHGCAP
'///

@define OFF          1
@define ON 0

@define BIOS         e=usr(58)
@define CHGCAP       &H132

~          p(2)= @ON*256:            'or @OFF
           p(0)= 0:
           p(1)= @CHGCAP:
           @BIOS
```

Basic-listing: LIST9

```
'///
'/// Listing 9: Getting palette data information with GETPLT
'///

@define COLOR c
@define RED r
@define GREEN g
@define BLUE b

@define BIOS         e=usr(58)
@define GETPLT       &H149

'GETPLT works in the following way:
'Input:  A = Color to get information for (p(2)=AF)
'Output: BC = p(4) = GREEN + 256*BLUE + 4096*RED

~          @COLOR= 7  'Sample color
           p(0)= 1:   'Now we call SUB-BIOS
           p(1)= @GETPLT:
           p(2)= @COLOR*256:
           @BIOS:

           @RED= (p(3)\4096) and 15:
           @GREEN= p(3) and 15:
           @BLUE= (p(3)\256) and 15:
           print @COLOR; "="; @RED, @GREEN, @BLUE
```

NestorEND: summarising...

Basic programs are easy to make, but slow and very limited; assembler programs are a lot more powerful, but even the simplest task becomes a very time consuming and complicated thing. NestorBASIC wants to be an intermediate solution, as well as a good alternative to the standard hybrid programming: placing your assembler routines in mapped memory enables you to save a lot of the always scarce basic memory and also you can easily set and read all the registers.

Although probably I will not continue the development of NestorBASIC, I continue developing assembler extensions and I have noticed other people are doing the same. Remember to visit my home page from time to time and if you have any doubt or suggestion, do not hesitate to mail me.

I hope you find these tips useful. Thanks for your time and... enjoy programming!

let us play

MEGA Guide

MEGA Guide is back and even better than the last ones in MCCM. It is possible that you never saw it, because MCCM was a Dutch magazine. That is why I explain a lot in this first MEGA Guide. People who already read the old MCCM should also read it, because some things have changed.

Robert Wilting

Directory

MEGA Guide is the part of MCCW for tips and help on playing games. This first part was just written by me but I hope that I will receive some material of our readers. So, please mail me if you got a great cheat or tip. But also when you are stuck in a game. Yes, this part of MCCW totally depends on the input from the readers. Of course you will get credit and fame for all published material that you handed in. In MCCW the reward for your input is the eternal fame of having your name in our web magazine. In any case: please send me material, or this part of MCCW will end very soon.

This new MEGA Guide will have a 'direct help' service. This means that you can always send me an e-mail with questions about games. If I can, I will answer you within three days. If I cannot find the answer in that time, I will search for it. When successful, I will publish the answer in the MEGA Guide. So, do not hesitate to e-mail me, because your input in the form of questions is also material for this part of MCCW and the more I get, the more likely it is that I can continue with MEGA Guide.

Every game hint that I receive will be checked, so all of them should work. However, if something goes wrong with this, please mail me so that I can put it right in the next Guide. For this issue, we are going to put some things right, first. Some game tips are published with errors in them. It is not a problem if the errors are corrected later, but most of the time this does not happen and the erroneous tip is copied and recopied and many people complain that it does not work. So here are a few corrected tips with an explanation about what went wrong, if possible.

Arkanoid, Taito (MSX1, ROM)
A great break out clone from Taito, which was released for many systems. And sometimes this can be very handy: there are a lot of Japanese MSX games released in English for some consoles or even for the pc. However, most of these versions have small differences. In issue 70 of MCCM someone mentioned the password `DSIMAGIC`. A few MCCMs later, someone said that it did not work. Well it does work in the pc version of Arkanoid, but not in the MSX version.

As you see this is a nice lesson for the future. It can sometimes be very handy to play an MSX-game on another system. But remember that if you give tips the game, that they must work in the MSX version. So, if you want to translate some menus easily, by just taking over the English menu text in the console version, do not forget to look if it is exactly the same menu in the MSX version. I have seen it myself with Ys 3: three different versions means three different menus.

To come back to Arkanoid: there is a cheat mode in the MSX version. Just hold `GRPH` at the beginning and push the cursors up and down a few times and you will get infinite lives.

Dynamite Dan, Mirrorsoft (MSX1, cas)
Another old tip which is wrong: many people claim that you if you push `F9` and `F10` together, you will get infinite lives. Wrong! If you press these two keys you will go directly to the safe and have enough Dynamite. Now, there is a nice trick you can do with these buttons: if you walk against the door and press the two keys quickly enough, you will be teleported to the place on the right side of the screen and it will not cost you any lives for blowing up the door. Dynamite Dan also has a cheat mode for infinite lives: press `SHIFT`, `A`, `S`, `D`, `Z`, `X`, `C` all simultaneously. If you do this for a second time, you will no longer have infinite lives.

The next tip is for a newer game, called 'The Lost World'. This game is a nice RPG from UMAX, which looks like Dragonslayer 6. I think it is a great game, except for the ending. I will explain two things about this game that have been asked to me many times.

The Lost World, UMAX (MSX2, 4*2DD, Stereo)
Question 1: what is the cheat mode and were can I enter it? Enter this in the title screen: `A`, `A`, `R`, `A`, `B`, `Z`, `A`, `R`, `A`, `Q`. If you typed it correctly, the game will go to the start-menu. Now choose 'NEW GAME' and you will start with Joshua at level 75 with some nice spells and the mirrorshroud, so that you have a really easy time playing the game. This also brings me to a big question, which I have been asked for several times: where is the Altar of Eternal Power? It is in the desolate waste of Northland. There are some caves in this area and in one of the caves there is the Altar of Eternal Power. Put the three skulls on the Altar and a door will open. Now walk to the north until you find a stone on which the password is written.

Livingstone, I Presume?, Opera soft (MSX1, cas)
And now a tip for an old game again. A long time ago, there were still many commercial European game makers. One of them was the Spanish company Opera Soft. They did not only make Ease, but also some great games like 'La Abadia del Crimen'. This tip is about two of their other games; both having exactly the same cheat. It is known that you can cheat with The Last Mission MSX2, with the following (this does not work in the MSX1 version): press `STOP`, now type `OPERA`, press `RETURN` and press `STOP` again. You now have infinite lives.

Only a few people know that it also works with 'Livingstone, I Presume?'. Note that it only works in the MSX1 version of the game. The MSX2 version has a different subtitle and was released only in Spanish.

Anyway, the game will be a lot easier with this tip.

Remember the Game Builder? The old game making tool, published by MSX Club Magazine. It is a nice program, but many people made the big mistake to only use the routines of the example game 'The Castle'... and so all games made with Game Builder are very similar. Some games do not have this problem: The ATP, The Castle of Blackburn and Lizard. However, most were just new Castle games. Consequently, Game Builder does not have a good reputation as a game creation tool. I think that it can be useful, though, as long as you make new game routines. It can be very frustrating to play almost the same game ten times: why don't we just load the end demo too see if the programmer did make something nice in the end? The next tip is how you can do this for one of the Castle clones.

Anger, MSX Club Magazine (MSX2, 1DD)
If you want to see the end demos you have to do the following. Start your MSX up in basic and type:
`RUN"anger.014"` for the end demo with the bad end
`RUN"anger.015"` for the one with the happy end.

Bomberman Special, Hudson Soft (MSX1, ROM)
Warning: this is not the Paragon MSX2 game, but a game of Hudson Soft which was released quite some time ago. There should also be a cassette version around. This would then be the officially released version for Europe. But I have never seen the cassette itself.

Here are the passwords for the first 13 stages:

STAGE 02:	`IHOLBABABAHIHIMNMNMA`
STAGE 03:	`MNNMKJNMNFLOLOBABABH`
STAGE 04:	`ABCPIOBABLFEFEKGKGKN`
STAGE 05:	`DJFEMIMNMAHIMNDDJDJL`
STAGE 06:	`HIABEEKGKJIHABKEFEFL`
STAGE 07:	`JDKGDPPCPNNMIHKEFEFL`
STAGE 08:	`HIFEMOHIHCLOBADNMNMC`
STAGE 09:	`JDPCOLKGKLCPPCBKGKGH`
STAGE 10:	`HICPIKLOLJABOLPEFEFN`
STAGE 11:	`BACPIHOJDFHIMNIKGKGG`
STAGE 12:	`MNEFPGPBAFMNJDOLOLOA`
STAGE 13:	`BAGKKNDOLANMIHPPCPCH`

Another tip: when a bomb explodes and another bomb is laying in the explosion line, it explodes together with the first one. So check if your in a safe position, before you blow yourself up. This multi bomb explosion can be very useful sometimes, though! And the last tip: the ghosts can walk through walls, but not through bombs.

Nemesis 3, Konami (MSX2, 2MEGAROM, SCC)
Most people know that Nemesis 3 has four passwords, which can be used when the game is paused, after pressing `F1`. Now most people already know what `FIND` does. For the people who do not: it will detect the weapons and maps and other stuff for you. The other two are also known. `GOOD` increases the difficulty level by one, while `HARD` decreases it by one. Now let me explain what the last known password does, since I have not seen any explanation so far. `EXPAND` just extends the shield duration: your shield will last twice as long. All passwords are typed in using `F1`password`RETURN` `F1`.

Ninja-kun, HAL/UPL (MSX2, 1MEGAROM)
Sometimes you will encounter stages that have two exits. The trick is to finish the level not via the easy exit, otherwise you won't get the extra weapons as bonus. If you solve scene 3 by going through the difficult exit you will get boomerangs. For scene 6 you will get grenades. Now in one of the levels close after the second one with two exits, the bombs will be very useful to you. I think it was in scene 9 that you have to defeat two big jumping enemies. If you have the grenades, then just throw two grenades on both creatures. If you do not have grenades then I wish you good luck, because defeating them with ninja stars or boomerangs is much more difficult.

Super Rambo, Pack-In-Video (MSX2, ROM)
Are you such a sick guy, that you count dead bodies when playing this game? There is a nice cheat for all those freaks. With this cheat you will no longer have to count the bodies yourself. Just press `F1` for the 'Body Count Mode': the computer counts the dead bodies and you can concentrate on making as many dead bodies as you want.

Puyo vs Gorby, Compile/S.T.U.F.F. (MSX2, 2DD, stereo)
Everybody wants to have the high score. Using the next tip you will see that you will get a lot more points in the same time compared to normal, which enables you to get the second place in the hall of fame by only playing the first stage. Shoot all Puyos until they are red. The Puyos that will come through, will form nice structures giving you lots of points. However, there is a bug in the game, which will cause that you never get the highest score. Every time you have the highest score, Carbun will get the same score as yours and stay on top of the list. So you will always be second...

Jagur, Compile & Hudson Soft (MSX1, ROM)
And now for some longer tips! From now on we will try to place a complete solve for a game in each Guide. I

will start with a nice MSX1 game that is a crossover between an action game and an RPG. Since I do not know the story of this game at this time, I will only give you the solution itself.

Let me first explain the controls. If you play with keyboard, you can use all weapons except the bombs with `SPACE` or `SHIFT`. Bombs are used differently: first place a bomb with `SPACE` or `SHIFT` and then use `Z` to let them explode. In all other cases, the `Z` is used to enable the options menu. Selecting is done with `Z` and to exit the menu you can use `SPACE`. The items of this menu are explained in the next paragraph. In the shops, use `LEFT` and `RIGHT` to select other products. With `UP` you can talk about other things. This way you can get weapons at some stores or men for your army.

The options menu has the following items:

1. Leader
   With this option you can select a team member as leader, who is the character for whom you can select another weapon/item. It is also the one who will fight against the boss. Each character has different weapons: Anne has the flame thrower, Roger (my personal favourite) has grenades, Jeff has a gun, Ukon has ninja stars and Guadic has a bazooka.
2. Formation
   With formation you can choose different formations for your group.
3. Item
   Can be used for selecting other weapons. Other items such as drugs which return your HP are automatically used when needed. During a boss fight this will not work, however.
4. Status
   Just shows the status.
5. Contact
   Gives you a code which you can use for playing the game on a later time, starting at the point you you were. For this option you need the receiver, costing $1200.

With `UP` you can talk about other things. This way you can get weapons at some stores or men for your army.

You start with Jef in a little town. Now go to the highest house on the left side, where you will get some money. The same you can do with the highest house on the right side. See also the map in Figure 1.

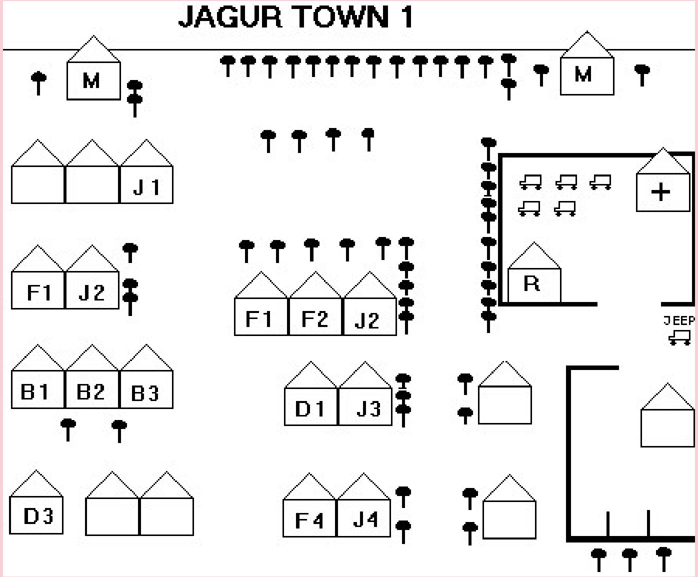


Figure 1: *Map of town 1 in Jagur.*
Legend:
M = place where you can get money
J = junk
R = rent a jeep ($500)
D = drugs
F = food
B = bar (B1: also has a sound test)
+ = hospital
J1/D3 = sells machine gun and grenades
J3 = sells receiver and transmitter

Now go to the most left bar *B1*. Just talk, using `SPACE` and they will ask you to pay $1000. Press `SPACE` to pay and you will get another character to play with. You should also find the other team members. Because they are randomly placed in the game, I can only give you the following three tips:

1. They should be in a shop: junk, food or drugs. Check J, F, and D on the map.
2. However, they are not in the shops which sell machine guns and grenades.
3. You will need to swap a machine or some other equipment for a person.

You can buy the following things in the town:

- weapons: grenades, machine gun
- drugs: medicine A, medicine B
- food: A, B; I can’t remember using food actually
- items: transceiver (needed for saving), receiver (I don’t know what it is for, just buy one to be sure.)
- bullets: additional bullets for machine gun
- body armor: this can be bought for any team member and increases defence
- health: in bar 2 and the hospital you can refill your hp points for $400 per person.

Note: most stores let you switch to a second item with `LEFT` and `RIGHT`!

If you do not have five characters at this point, visit all stores again and just buy some different equipment, like grenades, a machine gun, a transceiver and a receiver. Talk with the aforementioned store owners again, until you have all five. So now we have our private army. What for? Well, we are going to the warzone!

To go to the warzone, go to the house (*R* on map) near the hospital and pay the $500. Then walk to the jeep and step in it. For this, just move down when you are ‘above’ the jeep. Now you will drive to a new screen, with some buildings. Some contain enemy soldiers and others have paths to other places. Here is a simple ‘map’:

```
* Temple * Forrest * Town 2 * Base * Rocks * End-temple *
   *The area between the Town 1 and other places*
                 * Town 1 *
```

What you have to do first is buy a bazooka. You can get it on a house on the right side in this area, for $2250. Close to this house there is a house in which you can refill your health for $400 per person. Use this if you need it. After you got the bazooka, you can start with getting the dolls. You will need five dolls before you can go to the end temple. Let us start with the rocks. Just walk through the opening which leads to them, see the map. Blow up some rock with the bazooka and walk through the rocky area. You will find some statue with water on the left. Blow up the rock with the bazooka and enter it. The other rocks that are before some caves can also be blown up. When fighting the boss, jump up or duck when he shoots at you. When you have defeated him, you will get a statue.

So, that is one area cleared. Now lets go to the base. Just enter every building until you get the flame thrower. (A red bar with fire on it.) As soon as you get it, select this item and go to the flower fields. Destroy all the flowers with the flame thrower. You should find a hole. Select your fighter and enter the hole for the next boss. Just kill him in the same way as the first boss.

After this, go to the forest. Walk until you find six red holes in the ground. Search the area for six big statues and put them on the holes. The only problem that you will have, is that there are some trees in your way. However some trees can be blown up with your bazooka, so that you can place the statues on the holes. When completing this, two statues before a house will move apart. Walk through the hole and blow up some trees. Enter the house. Here is the next boss, defeat him the same way as the other two.

After defeating him, go to the temple. Here you must find five red scrolls. First enter the three houses, which will give you three scrolls. Now you must blow up some parts of the temple. On the left there are three sort of towers on a row. Blow up the second one with the bazooka and enter the hole. Here you find the fourth one. The fifth is in a tower on the left. If I am right, it is the left one in the second row. Enter this hole and you will find the fifth scroll. Go up and enter the big temple door for the fourth boss. Guess how to defeat him... yeah, just like the other bosses.

The next thing you should do is visit all the houses. There is one house with a guy who wants to sell you a letter (which you cannot see) for $800 dollars. Buy it. Next, make sure to have $12000 and go the second town. The person in the house before the bridge will let you pass because of the letter. Go the big house above the bridge. There, you will see a lady with green hair. She will join you. Next, buy all new items in town and enter every house and hut until you meet a lady with purple hair. At this point the music will change and you will have to go back to the big house above the bridge. Here you will fight the fifth boss. After defeating it, go to the end-temple. Walk as soon as you can. Enter the only entrance and you will have to finish off the end boss. Just kill him the same way as the other bosses and you have finished Jagur: you will see an end demo and a credits scroll.

That is it for this first issue. In the next issue we will have new short tips, for at least Pentaro Odyssey 2. We will also have the perfect solve of King Kong 2. I have received so many questions about this game and finally there will be one perfect English solve. It will also contain some other short tips. I hope you enjoyed it and please give me some feedback.

A short note from the editors:

Of course there will never be a next issue, unfortunately...

the secrets behind a well known demo effect

The zooming rotator

One of the most used 2D effects ever has to be the camera that rotates and zooms in and out above a, mostly static, picture. This so called zooming rotator can be used in many forms and the techniques upon which it is based can be used for much more. It is the simplicity of the final algorithm that is the beauty of it all. So for all you demo programmers beginners out there, wet your appetite before you plunge into the undiscovered depths of source code.

David Heremans

Directory

If you ever get in contact with the current batch of business IT developers, you can test their knowledge of demo programming by asking them to make a zooming rotator. Since this is not about e-business or about handling a lot of — hopefully structured — data in a database, you will probably find a lot of them giving forfeit, a lot of them will turn to the most obvious solution at hand. They will start to deform circles. Do not be surprised if their first attempt will behave a lot like the following basic code.

Look mama, no optimisations

basic-listing: ROT1.ASC

```
10 screen 8,0:set page 1,1
20 zf=1:za=0.4:al=0.2
30 pi=4*atn(1)
40 for r=1to50:for i= 0 to 2*pi step pi/32
50 setpage,1:c=point(128+r*cos(i),106+r*sin(i))
60 setpage,0:pset(128+zf*r*cos(i+al),106+zf*r*sin(i+al)),c
70 next i,r
```

This is obviously a simple solution. What happens here is that we are rotating around the middle of the screen. We stand in the middle and look around us in steps of ± 10 degrees each time and with a radius *r*. We store the colour of this point in *c* and calculate were this point should be if we rotate it some degrees further and alter the radius by our zoom factor *zf*. For the mathematicians: we do not use the normal screen coordinates, instead we use polar coordinates to calculate the new position of the point (radius, angle) to (radius×zf, angle+offset).

Disadvantages

Although the reasoning used in the previous program is mathematically correct, everybody will agree that in practice it is insufficient. You could make a zooming rotator in this way, if you would translate *all* of the *infinitely small* points of a mathematical plane. This thing can not be done very well in the not so abstract computer bitmaps. First of all, in the mathematics point of view every point has dimension zero. This means that to cross even the smallest distance we need an infinite number of points. Rotating them will therefore take an infinite amount of time, which we do not have.

OK, the above statement was just an other theoretical remark. Here are some real flaws of the program:

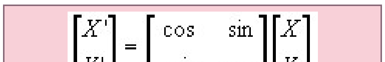
1. The program calculates too many or too few points.
   For every radius it takes the same amount of points on the circle, on the smaller circles a lot of points will overlap so too many points are calculated. The circle will be drawn completely but it is slow. On the greater circles there will not be enough points calculated to fill the entire circle and you will see giant gaps appear.
2. You will calculate a filled circle
   In most case however we want to have a rectangle as end result of the zooming rotator. If for example the purpose would be to fill the screen, you would have to calculate a lot of points, which would be discarded afterwards because their final position is on an off screen coordinate.
3. Empty points
   A computer has to store its numbers internally with a finite number of digits — this in itself already introduces rounding errors — something the used mathematical solution did not take into consideration. Later these imperfect numbers are again submitted to truncating, introducing even more possible rounding errors. Concretely, this results in points on screen which will never be filled with an other colour, introducing a moiré effect.

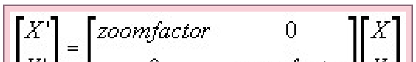
A lot of these effects can be corrected in a very crude way:

If we would take into account how many points we would have to calculate for a given radius *r* this argument can be circumvented. We could for example store all these values in a simple array.
You could of course test if the rotated point is inside the box (x1,y1)-(x2,y2). You could check even if it is inside a triangle or some other weird shape. On the other hand these extra tests will slow down the process even more.
Instead of drawing a single point, you could draw small boxes. These boxes will overlap each other, but they will also overlap the empty holes in the end result.

A more elegant approach

The previous solution really was a first attempt, now let us start to think about what we want to achieve. We want to take an original picture and rotate the points so that we get a rotated and possibly zoomed version of it. So it makes more sense to take every original point and calculate the new position of it. People who like mathematics will certainly be aware of the rotation and scaling formulas, see Figures 1 and 2.





$$\begin{bmatrix}Y'\end{bmatrix} \quad \begin{bmatrix}-\sin & \cos\end{bmatrix}\begin{bmatrix}Y\end{bmatrix}$$

$$X' = X.\ \cos + Y.\sin$$
$$Y' = X.(-\sin) + Y.\cos$$

Figure 1: *Standard rotation formula*

$$\begin{bmatrix}Y'\end{bmatrix} \quad \begin{bmatrix}0 & zoomfactor\end{bmatrix}\begin{bmatrix}Y\end{bmatrix}$$

$$\begin{bmatrix}X'\\Y'\end{bmatrix} = zoomfactor\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}X\\Y\end{bmatrix}$$

$$X' = zoomfactor\ *X$$
$$Y' = zoomfactor\ *Y$$

Figure 2: *Standard scaling formula*

$$\begin{bmatrix}X'\\Y'\end{bmatrix} = \begin{bmatrix}zoomfactor & 0\\0 & zoomfactor\end{bmatrix}\begin{bmatrix}\cos & \sin\\-\sin & \cos\end{bmatrix}\begin{bmatrix}X\\Y\end{bmatrix}$$

$$\begin{bmatrix}X'\\Y'\end{bmatrix} = zoomfactor\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}\cos & \sin\\-\sin & \cos\end{bmatrix}\begin{bmatrix}X\\Y\end{bmatrix}$$

$$\begin{bmatrix}X'\\Y'\end{bmatrix} = zoomfactor\begin{bmatrix}\cos & \sin\\-\sin & \cos\end{bmatrix}\begin{bmatrix}X\\Y\end{bmatrix}$$

$$\begin{bmatrix}X'\\Y'\end{bmatrix} = zoomfactor\begin{bmatrix}X.\ \cos + Y.\sin\\X.(-\sin) + Y.\cos\end{bmatrix}$$

$$X' = zoomfactor\ \ (X.\cos + Y.\sin\ )$$
$$Y' = zoomfactor\ \ (X.(-\sin) + Y.\cos\ )$$

Figure 3: *Scaling and rotating combined*

If you combine them, you get the formulas of Figure 3. So:

```
X' = zoomfactor ( X.cos + Y.sin )
Y' = zoomfactor ( X.(-sin) + Y.cos )
```

Now that we have these formulas, let us create a second program. We will take every point of the image we want to deform and calculate its new coordinates. Also, at this point we can already introduce great speed gains. For example, all the points we are going to rotate we need to calculate the sines and cosines of angle alpha only once. We will store this value in a variable. If we store the scaling factor in another variable we can again gain some speed. Here is the first version of our enhanced program.

basic-listing: ROT2.ASC

```
10 screen 8,0:set page 1,1
20 bload"picture.sr8",s
30 zf=1:al=0.2
40 co=cos(al):si=sin(al)
50 for x=-50 to 50:for y=-50 to 50
60 xa=(x*co+y*si)*zf
70 xy=(y*co-x*si)*zf
80 setpage,1:c=point(128+x,106+y)
90 setpage,0:pset(128+xa,106+ya),c
100 next x,y
```

This solution is much better.

- We do not have to calculate the sines and cosines over and over again. We just calculate it once and then we can store it in some variables. This increases speed a lot.
- Every point in our source picture is calculated only once. We will not do the math for the same point over and over again.

Of course some problems are still there. If we zoom in too much we will still have gaps, and if we zoom out too much we will again overdraw most of the points.

An other perspective

Up until now we always started out from our known picture and tried to calculate the points for the unknown picture. We have noticed in our examples we keep having problems because some points in our final picture are not filled in, or some points of the picture are overdrawn. Another problem is that we do not know exactly where our new picture will end up on the screen.

Maybe it is time to change our way of looking upon the problem and reverse our question. We are not longer interested in "Where will the original points end up in the final picture" but the more accurate question should be "what should be the colour of our points in the final picture". If we look at the problem this way, we solve both problems mentioned in the previous paragraph in one go.

Adjusting a previous example

Let us take our last basic program. Instead of taking a rectangle of 100 by 100 and rotate this original, let us calculate the points we should need to fill up a rectangle of 100 by 100. We are going to take every point in our final picture and calculate its position in the original one. We could take out formulas:

```
X' = zoomfactor ( X.cos + Y.sin )
```

```
Y' = zoomfactor ( X.(-sin) + Y.cos )
```

and recalculate them for X and Y, but it is much simpler to say

```
new rotation angle = -(original angle)
new zoomfactor = 1/(original zoomfactor)
```

This means that if we do not change these two parameters we will just change the rotation direction and instead of zooming in we will be zooming out. Keeping this in mind we can simple change line 80 and 90

basic-listing: ROT2FIX.ASC

```
80 setpage,1:c=point(128+xa,106+ya)
90 setpage,0:pset(128+x,106+y),c
```

And abracadabra! We have a zooming rotator. One that is free of all the major disadvantages we had in our previous examples, there are no gaps, no doubly calculated points and we now exactly which place to reserve on screen for the end result.

The above solution is a fast and clean piece of program that is easy to understand and rather gracefully combines the pure mathematical approach and the real computer implementation of pixels. For demo programmers there still is a faster solution.

And now for the really speedy solution

Most demo programmers will tell you that the best way to achieve fast results is to use only integer — or fixed point — calculation and that one should use as simple instructions as possible because they execute the fastest. In our case this means that we should get rid of all those multiplications.

Let us have a look at Figure 4.

Again we change our strategy just a little. Instead of taking every point separately through the rotation formula we could say we start at the upper left corner of the on screen drawing and take a step to the left: in vector notation we take a step (1,0). You can read this as $x=x+1$ and $y=y+0$. If our original picture is not zoomed and rotated, this would also be a step of (1,0) in our original image. If we would have a zoom factor of 2 this would mean that for every step (1,0) in our result we would take a step of (1/2,0) in the original. If we would have rotate our picture over 90 degrees instead of zooming, we would take steps of (0,1) in our original for every (1,0) step in the final on screen result. What we are going to do is the following. Instead of calculating every point, we are just going to calculate the start of our line. We are also going to have to calculate what the size and direction of our step is going to be. This is very simple: we will rotate (1,0) by using our formulas and the corresponding (x,y) is exactly the step we need.

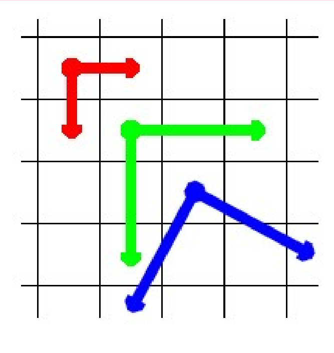


Figure 4: *Red: this is a normal stepvector*
Green: If we zoom out we start to skip points
Blue: Zoomed out and rotated

Of course, if we can take steps of (1,0) for our horizontal line we can also take steps of (0,1) to change our vertical lines. So what we need to calculate using our formulas are 3 points:

1. The upper left corner after rotation
2. The corresponding steps we need to make to step 1 to the left in our result
3. The size of the step needed to descend a line

Here is the final basic program:

basic-listing: ROT3.ASC

```
10 screen 8,0:set page 1,1
20 bload"picture.sr8",s:set page 0
30 zf=1:al=0.2:zs=0.2    REM Zoomfactor and rotation angle
40 co=cos(al):si=sin(al):REM on time (co)sine calculation for current frame
50 x=-50:y=-50                REM coordinates of upper left corner
60 xc=(x*co+y*si)*zf+128
70 yc=(y*co-x*si)*zf+106 REM XC,YC the coordinates in original picture
80 REM Calculating step (0,1)
90 xv=(si)*zf+128
100 yv=(co)*zf+106    REM XC,YC the coordinates in original picture
110 REM Calculating step (1,0)
120 xh=(co)*zf
130 yh=(-si)*zf      REM XC,YC the coordinates in original picture
140 REM getting trough the picture
150 for i=0to100
160 x=xc:y=yc
170 xc=xc+xv:yc=yc+yv REM XC,YC placed one step vertical step lower
180 for j=0to100
190 setpage,1:c=point(x,y)
200 x=x+xh:y=y+yh
210 setpage,0:pset(j,i),c
220 next j,i
230 al=al+0.01
240 zf=zf+zs:if ((zf=1) or(zf=5)) then zs=-zs
250 goto 40
```

The way this program works is rather easy to understand.

- In the first lines we set up the screen and load the original picture
- We initialise the zoom factor (*zf*), the rotation angle alpha (*al*) and the zoom speed (*zs*)
- Calculating sines and cosines takes a lot of time so we calculate it once and then store the result in a variable; this will speed up the basic program.
- We rotate the upper left corner and store these values in *xc* and *yc*, our short notation of x-corner and y-corner.
- We calculate the step size for a vertical step and store it in *xv* and *yv*.
- And we do the same for a horizontal step in *xh* and *yh*.

Then we walk through the resulting picture from left to right (horizontal) and from top to bottom.

Since we need the corner coordinates later we first copy their values in some more convenient variables, *x* and *y*.
For every horizontal step, represented by variable *j*, we add a (rotated and zoomed) horizontal step to *x* and *y* in our original picture.
After having walked over a horizontal line we still have the coordinates of our corner in *xc* and *yc*. For our next horizontal line we will need start one vertical line below the original corner. So we add a vertical step to our corner coordinates from this step on we can draw a new horizontal line.
Once our entire result has been calculated we change the rotation and zooming factor and start all over again, calculating *xc*, *yc*, *xv*, *yv*, *xh* and *yh* and redrawing the picture.

Except for the occasional `DEFINT` and `DEFDBL` this is as fast as it can get in basic. The demo programmer will use assembly and luckily the last solution can be very easily ported to it.

Possible optimisations in assembly
Here are some hints for the assembly programmers if you want to produce some really fast code. I will not give the entire solution since it is more fun to make one yourself.

- The calculation of the sines and cosines will be very fast if one uses a look up table.
- Using fixed point you can easily use the `ADD HL,DE` instruction. Use `H` as the integer part and `L` as the fractional part and you have the perfect way to make some fast code.
- Try to make the original picture have a width of 32, 64, 128 or even better 256. This will make it very simple to look up the colour value of the original point.
- Remember, a Z80/R800 has a double set of registers.

If you want to be creative do not look at this example. If you are not intending to write one yourself, but you would like to see a fast inner loop, a nice example is given below. This inner loop is the equivalent of the basic lines 180-220.

ML-listing: INNRLOOP.ASC

```
; The picture is stored at #0000 and has dimensions (256, ??)
; One byte is one pixel
; VRAM write is set to the correct address
; B is the number of horizontal pixels
; C is #98
; HL is x
; DE is xh
; HL' is y
; DE' is yh

LOOP:
 ADD HL,DE ; X=X+XH
 LD A,H
 EXX
 ADD HL,DE ; Y=Y+YH
 LD C,L
 LD B,A
 LD A,(BC)
 EXX
 OUT (C),A
 DJNZ LOOP
```

Same idea, other result
With the fixed-point-step-trough-a-picture method described above you have seen that it is very easy to rotate a picture, scale a picture or do both at the same time. This is also the basis for simple texture mapping. There are however some other nice effects to create. You could add a sine to what we have called *xv* and *yv* or both at the same time. This would create a wave effect in the final result. Or you could play with the values of *xh* and *yh* during the tracing of a horizontal line. If you do this it is possible to create all kinds of lens like effects. The few pictures below give you some results of what can be done when these variables are altered during the calculation instead of keeping them fixed. *Editor's note: we have never recieved the pictures David mentions here and after asking him, he could not find them back either... So, we apologise for the missing pictures!*

A simple idea and a little implementation that opens a whole range of possibilities when used with a little imagination. Great demos are made with great music, great graphics, great programming and some very creative imagination. I wish you all a lot of fun and creativity when playing with these routines, it could well result in some new great demos.