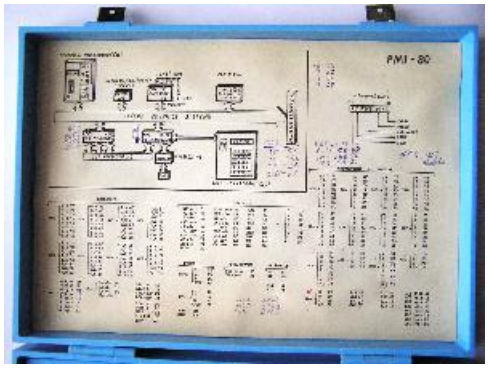
Microcomputer PMI - 80

Introducing the notorious and iconic **PMI-80** after almost 30 years since its introduction probably doesn't make much sense. So, just briefly. The PMI-80 is a classic **school single-board microcomputer** designed for teaching microprocessor technology and programming. A complete computer based on the **8080A** microprocessor is located on a rather tiny double-sided printed circuit board measuring **240 x 145 mm** . If there is an 8080A somewhere, there will also be the **8224** (system clock and RESET generator) and **8228** (system controller - status word decoder and data bus driver) supporters nearby. The processor operates at a clock speed of **1.11 MHz . Although a 10 MHz** crystal is installed , the 8224 circuit always divides the input frequency by nine, so **10/9 = 1.1111 MHz** . The fixed memory is represented by the **8608** circuit (PROM programmed by the manufacturer) with a capacity of 1KB. The service **monitor is located here. The RAM memory consists of two 2114** circuits (each 1024x4 bits, 1 KB in total). The RAM is available to the user for writing their own programs. Only its upper part is reserved for the monitor notebook and the processor stack.

The computer is controlled using a simple **keyboard** (originally from a calculator) with 25 buttons. **The display** is a 9-digit calculator LED seven-segment display with a common cathode. The keyboard and display are connected to the processor bus using an **8255 circuit (3 eight-bit parallel gates). The MH1082** decoder (used specifically in calculators) helps with multiplexing . Other circuits on the board include a **3205** address decoder and auxiliary NAND gates in a **7400** package . PNP transistors are used to drive the display cathodes. Furthermore, a simple interface is installed for connecting **a tape recorder** as an external memory for storing programs. Judging by how many different modifications of this interface have been released in the amars, it was probably not very original. But who would record anything on cassettes today, right? The tape recorder was connected using a classic "five-pin" DIN connector.

The computer came in a small plastic **case** (there were several types and colors) with **a list of the 8080A processor instructions,** including their hexadecimal code, glued to the bottom of the lid. As will be discussed below, programming on the PMI was anything but convenient.



Another note about the case: to protect the components from static electricity, a sheet of aluminum foil was placed under the computer board (this is often seen in photographs). Before you turn on the PMI, don't forget **to remove the aluminum foil** !

Due to the processor used (and also the PROM memory), three voltage levels are required to power the PMI-80: **-5V, +5V and +12V** . This placed increased demands on **the power supply** . Examples of power supplies suitable for the PMI-80 are listed on the **resources** page .

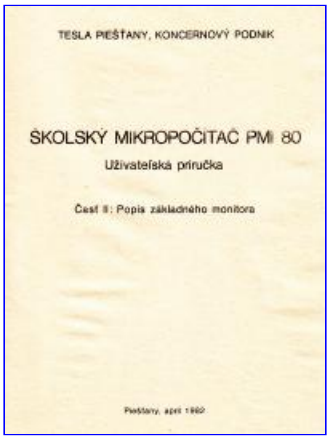
The computer was available in several **equipment levels** . It was possible to add **an additional** 1 KB ROM **8608** or EPROM **8708** to the board , as well as a second **8255** contact circuit , which was used in control applications. The PMI board also contains a small **universal part** for assembling other small accessories. More complex accessories had to be connected to the **FRB** connectors . The longer FRB connector (**62 pins**) contains practically the entire processor bus (data, addresses and control signals) and the computer can be **expanded** almost unlimitedly (provided that the signals are adequately amplified). The shorter FRB connector (**48 pins**) contains three gates of the additional 8255 circuit and gate B of the standard 8255. More about the PMI expansion can be found on the page about **modifications and accessories** . A photo of an almost fully equipped PMI-80 is below. The only thing missing is an additional 1 KB ROM/EPROM memory, but the socket is already installed.

You can also download **the complete documentation** for PMI-80, which was published in Amateur Radio in 1984. The documentation has been supplemented with **a list of errors** that are in it. There is also a file with a description of **additions and modifications** to PMI-80, which were also published in "Amateurs":

PMI-80: description **PMI-80: accessories**

Newly added are scanned **original manuals for PMI-80** . Manual No. 1 - **Technical description** and manual No. 2 - **Monitor description** . Only manual No. 1 has not scanned all the attachments. However, this is not a tragedy, because the attachments are nothing more than datasheets of the circuits used and there are plenty of them on the net. The exception is **PROM 8608** , which is not very common, and therefore it was scanned.

Handbook No. 1



Handbook No. 2

For connoisseurs, here is a real treat. Below you can download the scanned book " **Microcomputer PMI-80: what's up with it?** ", published in 1985 by Svazarm. This book was not intended for sale! It is a complete compendium of information about PMI. Here you will find an annotated monitor source, a complete technical description of the PMI and, most importantly, a lot of applications and programs for fooling around with the PMI-80. In addition to several obligatory **toys** , there is a description of connecting **the Consul printer** to the PMI, the use of PMI in **measuring** various quantities and, especially, a practical **programmer for Eprom 8708** memories ! The original low-quality scans have been retouched and the book is now converted to a professional DTP format. It is possible to copy text (program sources) from it or search for text in it. Simply great! There is also a so-called discarded version intended for double-sided printing. This way you will get an A5-format brochure that will be practically identical to the original!

original book cover

reprinted book

PMI-80: what about it? **PMI-80: what about it? (discontinued version)**

You can download the **annotated extract** of the source text **of the PMI-80 monitor, extracted from the book (from the original scans), below. It is suitable for printing and subsequent study:-)**

PMI-80: commented monitor source code

Below is **the source listing** of the disassembled **monitor** . This is the disassembled binary file **PMI80.ROM** . Disassembling was done with **the DASM 1.30** program and manually "cleaned up" according to the information in the documentation. For example, the tables of displayed texts were searched manually using the table of characters displayed by PMI. The source also contains possible modifications for PMI replicas described elsewhere on these pages. DASM allows the creation of a **SYM** file with symbolic label names. This file is also attached in a ZIP below, if anyone needs to fine-tune it.

PMI-80 monitor

The basic PMI operating program, the so-called **monitor** , is contained in just 1 KB of ROM memory. In addition to basic things such as the ability **to insert, edit and run** your own programs in RAM and record these programs to/from a tape recorder, it can also insert so-called **breakpoints** into programs and view and edit **register states** . The monitor also offers the user a whole range of **subroutines** , for example for operating the display and keyboard, which can be used in your own programs. In addition to simple statements on the display or testing the pressing of a certain button, the monitor also offers subroutines for entering one- and two-byte numbers. All functions, including subroutines, are described in **the documentation** , which is available for download on this page. As mentioned above, the binary code of the monitor (PMI80.ROM) and its disassembled source statement are available for download here.

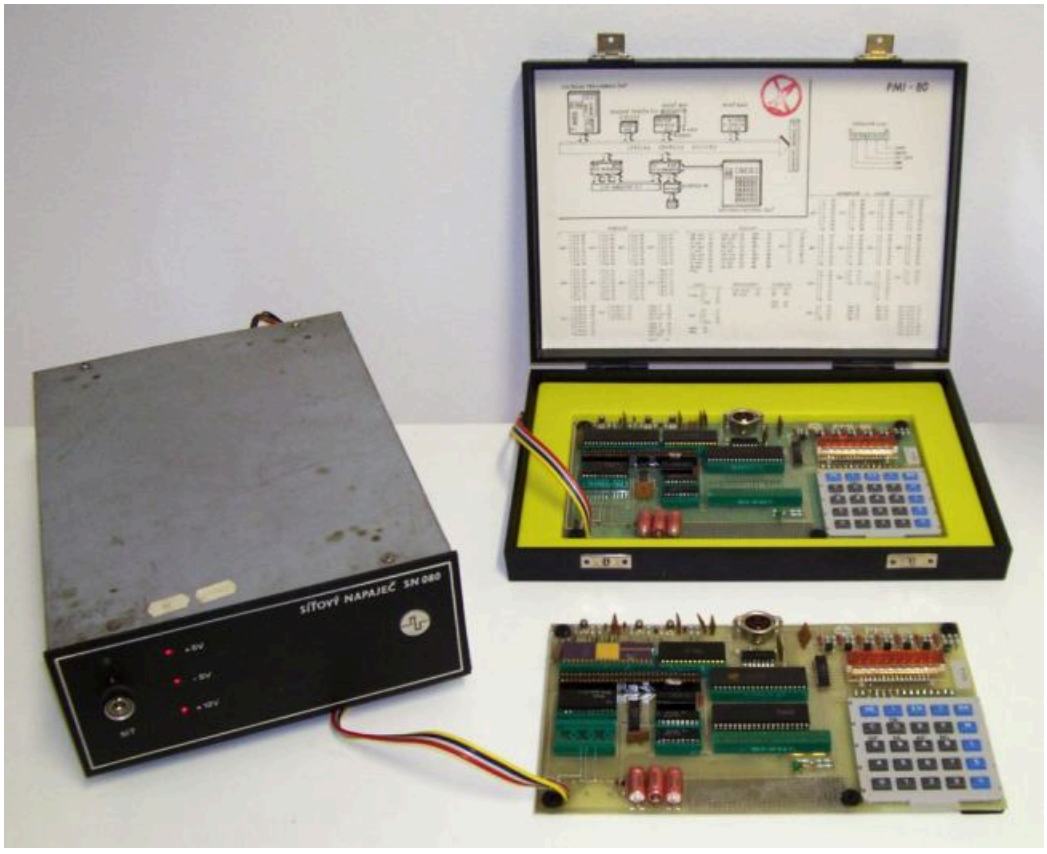
The PMI80.ROM file was downloaded from somewhere on the net, and since it is not good to blindly believe everything that appears on the net, a small experiment (actually two) was performed. The binary file PMI80.ROM was converted to an Intel-HEX file using the **BIN2HEX utility. The HEX file was loaded into the additional 1 KB of RAM memory of the PMI-80 using** PC Loader and a simple program was used **to compare** the contents of the original PMI ROM with the loaded data. The result? **Absolute match** . The file contains a truly **authentic monitor** of the PMI-80 microcomputer. As the icing on the cake, let us say that the PMI80.ROM file worked successfully even after loading it into **the EPROM simulator , which was connected instead of the memory in** the PMI-80r replica .

PMI-80: monitor source code PMI-80: monitor

Working with PMI-80

Working with the PMI-80 must have been hell when it was used as a training computer . Not only is it technically impossible to work with the original PMI (see modifications and additions), but can you imagine creating programs? The program had to be written in assembler on **paper** first . Then you had to **manually** translate it into machine code and manually **type it into the PMI. That would still be possible. But a program rarely works the first time and corrections and modifications were really a pain. What if an instruction is left over, you just put a NOP** instead . But what if an instruction **is missing** ? You can't just insert it. The entire part of the program from the inserted instruction had to **be rewritten** and most importantly, all jumps had to **be redirected** ! And all by hand with only notes on paper, which no one **understood** after a while of "debugging" . And when you had the program ready, you were terrified whether you would be able to save it to a cassette before **the overheated** circuits of the PMI stopped working. Of course, add to this the concern whether the **Emgeton** cassette you had just recorded would be readable in the future...

Fortunately, things are different **today . You can write and compile a program on a PC and, thanks to** the PC Loader, easily load it into the PMI memory. Corrections and recompilation of programs are very easy. Just **fun** :-)



2x PMI-80 and SN 80 power supply

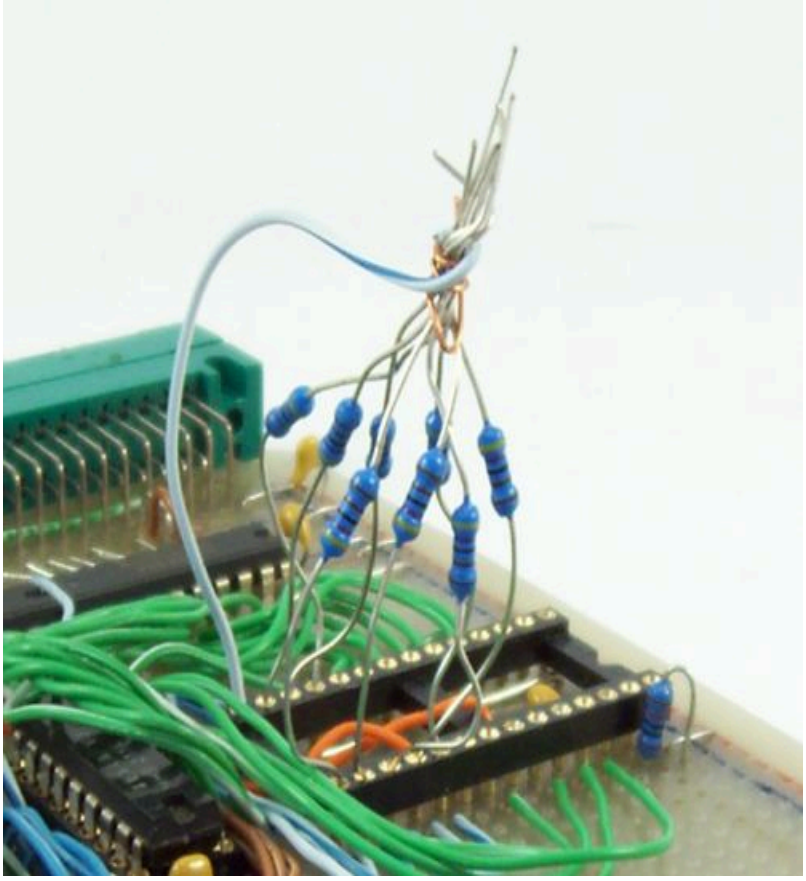
Commissioning of the old PMI-80

The PMI-80 computer was first manufactured in the early 1980s. That's quite a long time ago. Therefore, it's worth clarifying what you should do before turning on an old computer that hasn't been used for years. In the first phase, it's necessary **to clean** the printed circuit board , preferably with a brush or **by blowing it** with compressed air. When cleaning with a brush, be careful of **static electricity ! Before cleaning, it's a good idea to remove** all integrated circuits from their sockets and stick them into a piece of polystyrene covered

with foil . Then it's a good idea to carefully **inspect** the printed circuit board . You can even use a magnifying glass. If you don't think the quality of the soldered connection is good somewhere, or there's a hint of a crack in the connection, repair it **with a micro-soldering iron** . If the circuits are removed, you can connect the board to a suitable **power supply** and measure whether the power is everywhere where it should be (don't forget about the voltages -5V and +12V). Before inserting the ICs back into the sockets (with the power off!) it is advisable to gently clean their **feet** with very fine sandpaper (grit 400 and higher), or with an abrasive non-woven fabric. It is also advisable to slightly **bend** the feet with pliers so that they rest as strongly as possible on the contact surfaces in the socket. After a general check, you can turn on the PMI and it should greet you with the classic message **"PMI-80"** . Of course, operate the PMI only on an appropriate **power supply** !

If the PMI does not report even after pressing the **RE(set)** button , it's **a mess** :-) Then the search for a fault begins. In the first phase, it is necessary to check the presence and quality of all supply **voltages** . It would be advisable to replace **the filter capacitors** as a precaution , but in terms of preserving the "historical value", few people want to do that. All the better, you must have a power supply (you can replace the capacitors there, they are not visible). It is also advisable to use a counter or oscilloscope to check **the clock source** (1.1111 MHz on the corresponding 8224 outputs) and the **RESET** signal (on the 8224 input and output). Also check the **INT** input and other inputs that the PMI does not use, but must have **a defined value** (READY, HOLD). Also check **the configuration jumpers** . These are the pins with wrapped connections above the processor. For the PMI itself, they should be connected according to the diagram in the documentation. You can also orient yourself according to the photos published here.

The operation of the 8080A processor can be easily checked by the following procedure: remove **all** circuits connected to **the data** bus (except the CPU and the 8228 controller), i.e. all memory circuits and 8255 circuits, from their sockets. Now connect the data bus signals **D0-D7** via resistors of about **47 Ohms to GND. Simply insert** one end of the resistors into the data pins of a free socket (for example, from the ROM memory) and connect the other ends together and connect to GND. **The worst** possible way you can do this (however, at the same time the easiest) and how you will probably do it anyway is this:



NOP instruction processor test

This has actually created the absolutely **simplest possible program** . You have forced a single instruction on the processor: **NOP** (No OPeration). After switching on, the processor starts executing the program. `0000h`It loads the first instruction at the address - **NOP** . It does nothing. It just increases the contents **of the address counter** by one. The processor then goes to the address `0001h`and (surprisingly) loads the NOP instruction. And so it goes up to the address `FFFFh`. Then the address counter overflows (resets) and everything starts again from `0000h`. And guess what instruction the processor finds there :-) This has turned a sophisticated processor into a stupid **16-bit counter** . You can now find out that the processor **is working** by placing the tip of a logic probe on the **A15** signal . The probe will blink (alternating log.1 and log.0). Then you place it on **A14** . And it will blink again, only twice as fast. You try **A13** and it will blink again. And again twice as fast as on A14. You should be able to see the flickering by eye on **A12** , on lower address signals the frequency is already too high and can only be checked **with a counter or oscilloscope** . In the described way you can of course also check the operation of the processor when **building a replica** . Only for the **8085** processor with multiplexed data and address bus use resistors of about **4K7** .

Checking other circuits is quite individual. Sometimes you can do with a logic probe (or an LED, resistor and pieces of wire), in other cases you need to use a special **device** (for memory testing), perhaps one made on a non-solderable surface.

If you think your PMI has **a problem** , be sure to check out **the PMI-80 problem** page . You might find **a solution to** your current problem there. But if you (successfully) encounter a different problem, **write** so that we can let other users know! Thanks.

PMI-80 forever, or how to use PMI today

Most people probably think that the only thing you can **use** PMI-80 for today is **to play around** a bit , **mess around** in hexadecimal in front of younger students and then put it back in **the display case** . But PMI can still be quite **useful** after 30 years. You ask how?

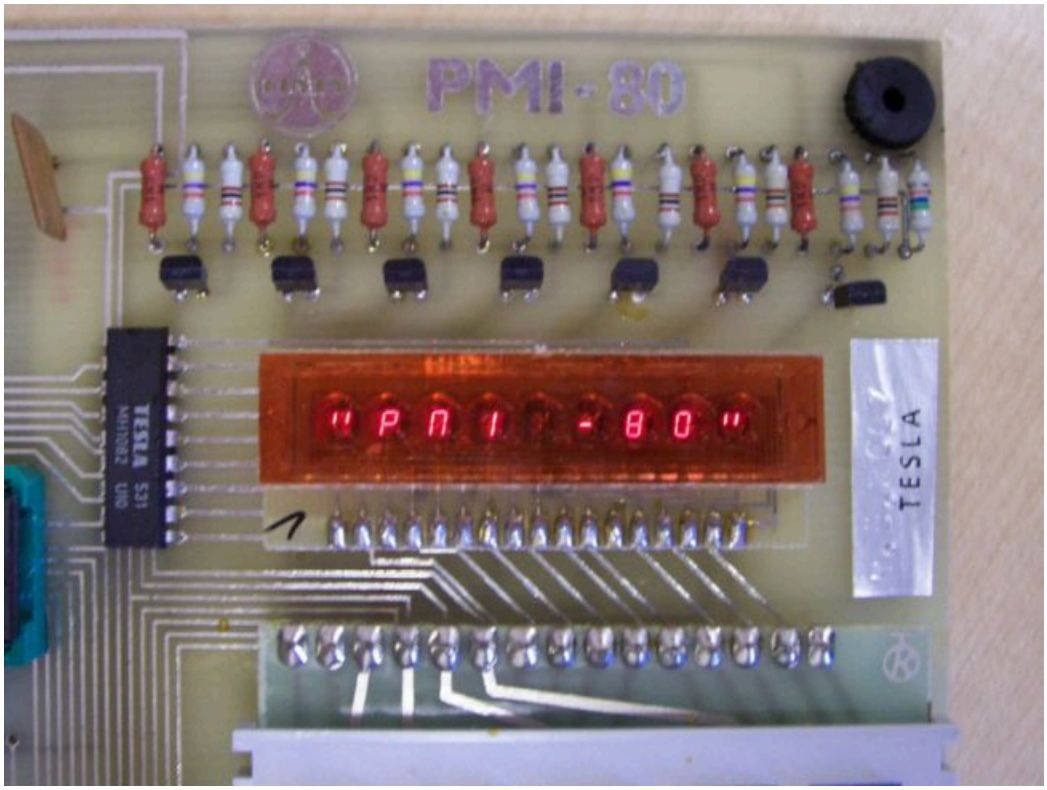
1) PMI-80 was originally intended to **teach programming** in assembler and machine code and to teach microprocessor technology in general. There is nothing to prevent it from **serving the same purpose in the future** . In principle, not much has changed, and knowledge of the basics can encourage future programmers not to create such software **monsters** as are being created today. PMI can be a very illustrative tool, especially for those interested in working with some single-chips.

2) PMI can be used to conveniently **debug** various algorithms that we develop, for example, for a single chip. Of course, there are more advanced tools, such as JTAG, but they are not available everywhere. Thanks to the ability to insert breakpoints and view the contents of registers, debugging algorithms is much more convenient than constantly flashing a single chip and letting some registers out for temporary added LEDs. Algorithms for **the Petr replica** were conveniently debugged on PMI-80 , especially routines for arithmetic (DIV/MOD, MUL and SUBB) and DEC/BIN conversions and vice versa. Although there are software simulators for almost anything, it is not advisable to trust everything...

3) Various fixtures can be constructed for PMI, creating various **single-purpose devices** for occasional use. A prime example is memory and single-chip **programmers** . If you regularly work with a programmable circuit, you probably have the appropriate programmer. But sometimes you need to program something else that your programmer can no longer do. Or you simply program only occasionally and a specialized programmer is not worth it. Then there is nothing easier than building a simple fixture for PMI.

All of the above ideas for using PMI are **enhanced** by the ability to capture, compile, and upload compiled programs to PMI from a PC, as described on the page about **PC Loader for PMI** . Without that, none of this would make sense and would just be a game.

If you can think of **another way to use it** , or if you are already using your PMI for something useful, **write to us** .



Advantages of PMI-80

It only does what you program it to do. **It doesn't mess around with the internet** . It doesn't constantly gnaw at the disk without you knowing what's going on. It doesn't crash. It responds to your instructions faster than those current software-crammed PC shops. PMI is just great :-)

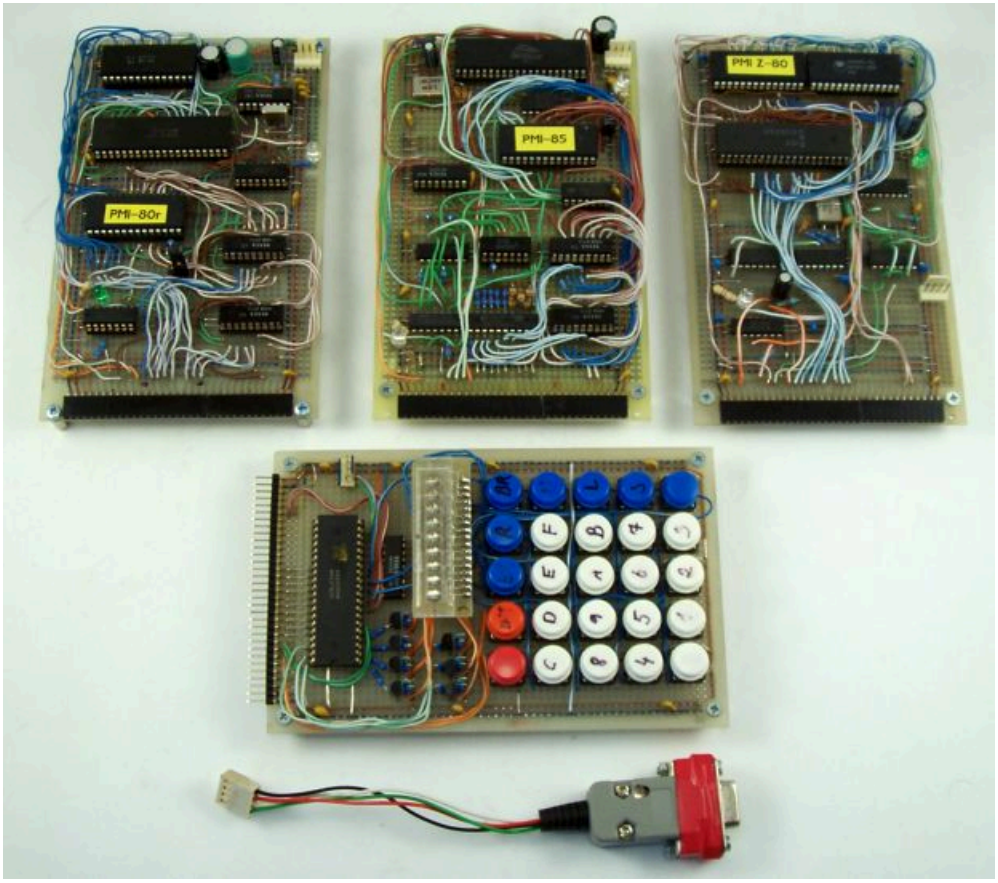
How to get to PMI

If you don't have **your own PMI-80** and would like to play around with it, don't despair. Sometimes a piece appears on Aukra, but **speculators** don't know what to say about the board. And when one does appear at an acceptable price, "collectors" swarm in and will jack up the price anyway, just to have the board sit in their display case. I know what you're talking about. I spent **a lot of money** on that piece in the case ...

Fortunately, there is **another way** to get to PMI - build **a replica** . According to the documentation that is available for download here, this is **no problem** . The monitor is also available and the components are still available. You don't even have to stick with the original 8080A processor and you can use, for example, **an 8085** or **Z-80** , as demonstrated by the replicas called **PMI-85** and **PMI Z-80** from these pages. Using these CPUs has (among other things) a pleasant consequence in the form of a simple **+5V** power supply . In particular , **the PMI Z-80** can be recommended for implementation, because it is **the simplest** in terms of circuitry (read the suggestions for simplification at the end of the article about the PMI Z-80) and the Z-80 CPU is still offered by many companies selling components, but you can usually find it under the designation **Z84(C)00** .

Build your PMI!

If you want to play with a typical **school single-board computer** and try programming **in hexadecimal** , the PMI-80 is **ideal** because it is well **documented** . But if you don't have a PMI-80, or don't want to pay ridiculous amounts to speculators for it, build **a replica** . And a replica with **a Z-80 or U880D** is the ideal solution. The Z-80 processor is still easily available and cheap, and a computer with a suitable design needs only a minimum of additional circuits while maintaining full software compatibility with the PMI-80. **The monitor** is also available and its modification for the Z-80 is **trivial** (if you don't use **interrupts** , no modification is necessary). In addition, you have a whole range of other instructions that the 8080 CPU doesn't have. Simply: **Build your PMI!**



three different PMI replicas with 8080A, 8085 and Z-80 processors

Puzzle for beginners PMI-80

If you are a new user of PMI-80, it may not be immediately clear to you how **to properly connect** and start it. Before you do anything stupid, try to think through what is wrong here:

Will it work like this?

PMI monitor SLOWKEY

Some users of **the PMI-80** computer have complained and are complaining about **the unreliable keyboard** , or rather **the poor handling of flicker** by the PMI monitor. Apparently it depends on the piece by piece or it was a problematic series of keyboards, because I don't have this problem with any of my original PMIs. Nevertheless, there was **a solution** in the form of **a modified monitor** . The modification was made by an unknown author and used **the unused memory** at the end of the original monitor. The modified monitor still fits into a 1 KB EPROM (8708). The button **scanning time has been significantly extended , and thus flicker** is perfectly treated . Unfortunately, the delay is already so long that it is almost **unpleasant** . However, if the keyboard is annoying you and replacing it is out of the question, a modified monitor is the solution. **Everything else works** as it should and the subroutine call addresses have not changed. You can download the modified **PMI-80 SLOWKEY monitor in binary form below in the ZIP archive. I tried** the Slowkey monitor and everything really works.

PMI monitor V2

Quite unexpectedly, another variant of the PMI-80 monitor appeared, called **V2 (version 2)** . Dušan discovered the monitor on his original PMI-80. Comparing the binaries of the original monitor and the V2 monitor, we find **a difference** of only a few bytes in the **ENTRY** area - the entry into the monitor from **a breakpoint** . The difference in the files is marked **in red** in the following image. The photo below also shows the Russian Eprom 2708 with a sticker marked as **PMI-80 V2** , which Dušan has in his PMI-80 (and he also successfully tested it in **the PMI-80 M16** replica).

The difference is in the introduction of the **ENTRY** procedure , which is entered at **a breakpoint** and where the complete state of the processor (all registers) is stored for subsequent viewing. I originally thought that the reason for the change was this:

*The original version first backs up everything possible and only then saves the status word with the flags (**PUSH PSW**). But even before this instruction there is the **DAD SP** instruction , which can affect **the C flag!** In the original version of the monitor, you cannot rely on the fact that the **Carry** flag saved during the breakpoint is really **authentic!** Version 2 solves this by saving **the PSW** first and then backing up the other*

registers. ***Version 2*** *can therefore only be recommended and if you like working with* ***breakpoints*** *, it is even* ***a necessity!*** *But if not, then you don't have to be afraid and you can continue to use* ***the original monitor*** *.*

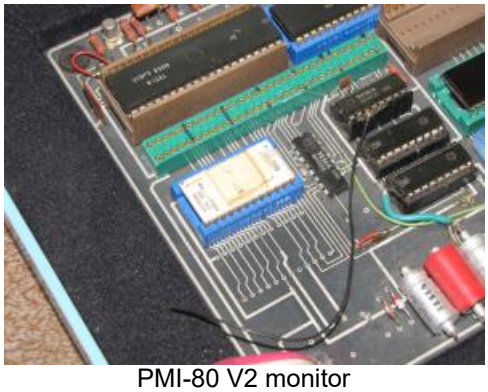
But it doesn't have to be that way! The HL register pair is **zeroed** before being added to SP , so the **DAD SP** instruction (HL=HL+SP) should not affect the **C** flag in any way, and therefore the original version is **fine** . DAD SP actually serves only to **copy SP to HL** . We can only speculate about the reasons for the modification, but it is possible that the unknown author did not realize this and made the modification in good faith.

For completeness, I am attaching a disassembled listing of the changed part between addresses `000Fh - 0020h`. You can see **the original code, for example, in the commented listing of the original monitor, which is part of the great book PMI-80: what's up with it?** , which you can **download** above on this page. As you can see from the listing, only the order of instructions has been swapped.

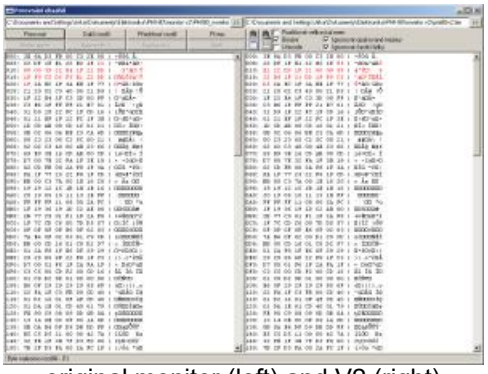
```
;zmenena cast kodu v monitoru V2 na adresach 000Fh - 0020h:
   PUSH PSW
   POP H
   SHLD 1FDDh
   LXI H, 0000h
   DAD SP
   SHLD 1FE4h
   LXI H, 1FDDh
   SPHL
   PUSH B
   PUSH D
```

A small note about the ENTRY procedure : for its full and correct use, it is necessary to enter it with the **RST1** instruction . Sometimes it is seen in programs (and I also do it sometimes) that ENTRY is called with the jump **JMP 0008h . In such a case, however, the state of the program counter PC** is not correctly preserved . However, if we are only interested in the state of the working registers when debugging some software, and if we do not intend to return to the debugged program, then it does not matter at all.

The ZIP archive below contains all **three available monitors** for PMI-80 (original, V2 and SLOWKEY) in the form of binaries designed for EPROM memory type 2708 (8708). The **SLOWKEY** monitor is based on the original version. However, nothing prevents you from modifying it to **the SLOWKEY V2** version ...



PMI-80 V2 monitor



original monitor (left) and V2 (right)

PMI-80: monitor, V2 monitor and SLOWKEY monitor

A series about the 8080 processor and its supporting circuits

Since I received a nicely processed scanned **series about key circuits** used in **PMI-80** by e-mail , I decided to post it here. Hopefully this will make the information about PMI really comprehensive :-) This is a series with a description of the 8080 processor and its supporting circuits. In addition to a detailed **description of the operation of the 8080** processor , here you will find, for example, **equivalent connections of the 8224 and 8228** circuits . Masochists can replace them with several discrete components :-) Of course, there are descriptions of the programmable circuits **8255 and 8251** and also the circuits **8205, 8212, 8214, 8216 and 8226** , which are better known in our country under the designation **3xxx (3205, 3212 to 3226)** . So, enjoy your study :-)

serial 8080

LIP

Optimized for MS IE 8.0 and Firefox at a resolution of at least 1024 x 768 pixels