http://www.iki.fi/hyvatti/pic/picprog.html

Jaakko Hyvätti Picprog 1.7 documentation

2004-04-29

PIC16 and PIC18 microcontroller programmer for Linux and Windows/Cygwin.

Translations of this document: Dutch

PICmicro microcontrollers, or MCUs, are fine chips that are especially easy to program with a simple device attached to a parallel or serial port. Because of the EEPROM or Flash memory, they are also easy and fast to erase and reprogram without need for UV equipment. This makes them very popular among electronics hobbyists.

At the moment this is the second implementation of a PIC programmer for Linux that works with the very simple and cheap serial port programmers. The first one I know was made by Ralph Metzler in 1996. My programmer was originally designed for PIC16C84 and PIC16F84 chips back in 1997, and since then I have implemented other chips without access to most of them. I have tested PIC16F628, PIC16F676, PIC12F675, PIC16F88, PIC16F876A, PIC16F76, PIC18F1320 and PIC18F458. Others have used many other models.

1. Background reading

You really should take a look at the Microchip www-pages and read the device datasheets and programming specifications there.

Maybe the best source for PIC information is the home page of PICLIST discussion group. Also historically a good collection of links and software for PIC was in David Tait's PIC links page and in GNUPIC pages. I have also documented here the software I took a look at back in 1997. I have focused into Linux support, so I have never used any DOS software mentioned.

2. Requirements

Serial port pic programming hardware
: See the hardware section. This device is connected to a usual serial port of your PC, and is the same device as used with many DOS PIC burning programs.

C++ compiler (g++)
: This program is written in C++, so you need a C++ compiler to compile it. This you should already have installed on your Linux system or Windows/Cygwin system.

Linux kernel version 2.0.32 or 2.1.45 or later.
: This programmer needs some functionality in Linux serial driver that as of kernel versions 2.0.32 and 2.1.45 is available in standard kernels. The programmer uses TIOCSBRK and TIOCCBRK ioctl to control the state of TxD serial port output accurately. These ioctl's are standard on BSD flavor unixes, like SunOS 4, but they still are unimplemented on many serial drivers in Linux kernel. Only the standard serial port is known to work by me.

Cygwin DLL version 1.5.8 or later
: (For Windows installations only) Earlier Cygwin versions do not contain the necessary TIOCSBRK and TIOCCBRK ioctl functionality.

A compiler for PIC
: Your assembler, or C compiler, or whatever, should produce either Intel IHX32, IHX16, or IHX8M format hex files. For assembler on Linux I recommend GNU PIC Utilities gpasm.

3. Hardware

Use a serial port programmer device with the following pinouts:

TxD
: Programming voltage, pin /MCLR

RTS
: Clock pulse, pin RB6

DTR (output), CTS (input)
: Serial data, pin RB7

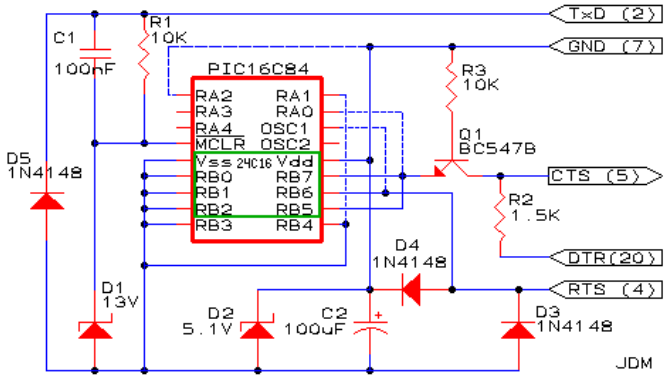
A very good programmer like this is for example PIC-Programmer 2 designed by Jens Madsen. I have heard that other programmers work also, for example TE20 and Olimex PG2C have been tested with Picprog.

The description below is based on old version of JDM hardware. There is a new version of the PCB available on Jens Madsen site. It supports more chips without jumper wires: PIC-Programmer 2. Please use that one instead. Note however that his PCB does not support for example PIC16F628A that need to have pin 10, RB4/PGM grounded. Please modify the PCB to ground this pin if your chip has PGM on pin 10.

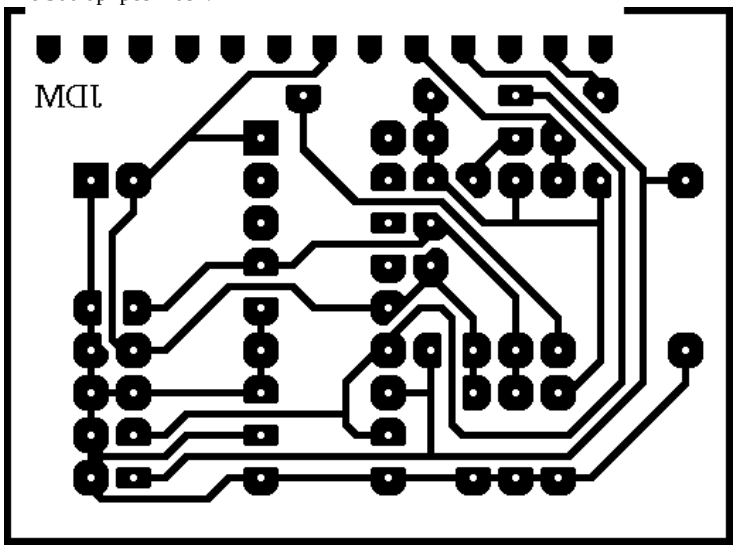
I made a minor modification to the jdm84v23 schema and pcb mask, because I thought D4 was stressed on positive clock pulses - it short circuits the rs-232 RTS pin to GND. I added a 10k resistor between D4 and D3. But it also is not absolutely required, as the clock pulses are short and rs-232 is protected for short circuits anyway. Now I have noticed that this resistor makes the programmer less reliable, especially with later PIC chips, like PIC16F76 and PIC18 family. If you have built the hardware I previously suggested, and it does not work, please short circuit that resistor or build the new PCB from Jens Madsen site.

To support for example PIC16F628 and PIC16F88, which have a low voltage programming mode, the circuit was again modified to ground the RB3 pin 9 and the RB4 pin 10. This prevents the chip from entering the low voltage programming mode. To support for example PIC12F675, PIC16F630, and PIC16F676, pin 1 was connected to Vdd. Other jumpers need to be installed as described below.

The schemantics diagram:



The 300 dpi pcb mask:



To support for example PIC16F876A and 16F76, which have different pinouts for programming signals, you need an adapter that connects to the external connector of the above programmer. Alternatively you could redesign the pcb layout, but maybe it is easier to just solder the adaptor. Solder the pins like this:

```
pcb       1 -- clk  -- 27 (RB6)      ic
connector 3 -- data -- 28 (RB7)      socket
          5 -- Vss  -- 8,19,24
          7 -- Vdd  -- 20
          9 -- Vpp  -- 1
```

To support for example PIC12F675, PIC16F630 and PIC16F676, which have different pinouts for programming signals but do not conflict with PIC16C84 pins, you can solder jumper wires on the

above pcb. You need to connect pin 16 on the socket to clock (same as pin 12), pin 17 to data (same as pins 11 and 13), and pin 18 on the socket to Vss (same as pins 5-10). Note that pin 1 is already connected to Vdd in the above PCB mask. If this connection is missing on your older PCB, connect pin 1 to pin 14 with a wire.

Always check the correct pinouts for your chip from the datasheets also!

4. Installation

Linux

There are binary packages available that have been prepared by helpful users and operating system vendors. I know of the following, but cannot offer any support for them. They might be for older versions, but even if they are, check for the 10. Changes section below if you really need a later version. Some new versions of Picprog only fix small specific bug, which may not affect you at all.

- FreeBSD includes Picprog port.
- Gentoo has Picprog as a package.
- Jan Wagemakers made a Debian package.
- Linux i386 rpm compiled on Red Hat 9 by Maarten Blomme

To install from source, download the picprog-1.7.tar.gz package, if you do not already have it.

Check your system against the requirements mentioned above.

Untar the archive and change to the source file directory. You should only have to type:

```
make dep
make
```

and the program should compile without errors or warnings. If it does not, please check that your compiler, c and c++ libraries and utilities like make are of a reasonably recent, bugfree and compatible version.

After compilation you can, as a root user, just type:

```
make install
```

to install the program and manual page to /usr/local. Or just copy the files `picprog` and `picprog.1` manually.

Windows/Cygwin

The Linux emulator for Windows, Cygwin, allows Picprog to be compiled on Windows plattforms.

To install Cygwin, go to www.cygwin.com and follow the instructions there. In short, download setup.exe and run it. You need to install at least the Developer packages Gcc C compiler, Gcc C++ compiler, Make, and Binutils in addition to the default install.

To install Picprog on Cygwin, follow the above instructions for Linux compilation from source.

Windows 2000 installations that I have tried work fine. Windows 98 installations worked sometimes, and I suppose the timing routines of Picprog do not work well on Windows 98, they even lock the computer sometimes. I suggest using Windows 2000 or later.

5. Usage

To get information about the usage of the program, just type the program name. These options give information about the program:

--warranty, --copyright, --help
: Display warranty or copyright information or the help text with list of supported chip types.

--quiet, -q
: Do not display the copyright notice.

The actual operation of the program is controlled by the options --input-hexfile and --output-hexfile. If the former is present, the program acts as a burner. If the latter is specified, the program will read the contents of the PIC device eeprom memories. Both may be specified on the same command line, in which case the chip is first programmed and then read.

6. Burning PICs

Simple instructions:

1. Compile your program into a hex file.
2. Insert the pic16c84 or other pic chip into the socket in the programmer, or connect the in-circuit programming cable to your device.
3. Connect the programmer device to a serial port.
4. If the device contains calibration information, like OSCCAL word in program memory or BG bits in configuration word in PIC12F675, it is a good idea to save these for later if this is the first time you program the chip. See next chapter for more on reading chip, but this should be enough for saving the calibration:

   ```
   picprog --output saved-cal-chip1.hex --skip-ones --pic /dev/ttyS1
   ```

5. Burn the program with command:

   ```
   picprog --burn --input file.hex --pic /dev/ttyS1
   ```

6. If the above command produces error output that suggests that the chip was in the code protection state, or that the chip must be completely erased before programming, retry with the following command:

   ```
   picprog --erase --burn --input file.hex --pic /dev/ttyS1
   ```

7. Wait for the program to complete. Burning 8192 program locations on 14 bit devices may take 3 minutes. I know better programming algorithms would speed this up significantly, but I have not had time to fix that. PIC18 family programming algorithm is significantly faster.

The burning options are:

--erase
: To be able to reprogram a PIC device that has previously been programmed into Code Protection state (for example in pic16c84 Control Word fuse bit 0x10 cleared), it is necessary to bulk erase the chip. Also some PIC devices do not automatically erase each location as they are programmed, and these devices must always be bulk erased first. It is done by adding this option to the command line. The default is not to bulk erase the chip.

--burn
: Actually program the device. Without this option only the syntax of input files and command line options is checked.

--input-hexfile *path*, -i *path*
: Specifies the input hex file. The file can be either in IHX32, IHX16, or IHX8M formats, the format is automatically recognized.

--cc-hexfile *path*, -c *path*
: Only necessary for debugging. Outputs the same data as was read from the input hex file.

--force-calibration
: Force reprogramming the OSCCAL word and BG bits in control word. Default is to read the calibration values off the chip before erasing and preserving their values. Use this option if you have accidentally erased the values on chip, and you reprogram them from your saved copy which was read off the chip before the values were lost there. In general it is a really good idea to first read the empty chip and save the file somewhere if the calibration data gets lost later, either by accident or because of some bug in Picprog.

--pic-serial-port *device*, -p *device*
: The device name of the serial port the programmer is connected to. Default is /dev/ttyS0.

--device *chipname*, -d *chipname*
: The chip type. Currently supported by code are:
auto, pic16c84, pic16cr83, pic16cr84, pic16f83, pic16f84, pic16f84a*, pic16f87*, pic16f88*, pic16c61, pic16c62, pic16c62a, pic16c62b, pic16c63, pic16c63a, pic16c64, pic16c64a, pic16c65, pic16c65a, pic16c65b, pic16c66, pic16c66a, pic16c67, pic16cr62, pic16cr63, pic16cr64, pic16cr65, pic16c620, pic16c620a, pic16cr620a, pic16c621, pic16c621a, pic16c622, pic16c622a, pic16f627*, pic16f627a*, pic16f628*, pic16f628a*, pic16f648a*, pic16ce623, pic16ce624, pic16ce625, pic16c641, pic16c642, pic16c661, pic16c662, pic16c71, pic16c710, pic16c711, pic16c712, pic16c715, pic16c716, pic16c717, pic16c72, pic16c72a, pic16cr72, pic16c73, pic16c73a, pic16c73b, pic16c74, pic16c74a, pic16c74b, pic16c76, pic16c77, pic16f72*, pic16f73*, pic16f74*, pic16f76*, pic16f77*, pic16c432, pic16c433, pic16c781, pic16c782, pic16c745, pic16c765, pic16c770, pic16c771, pic16c773, pic16c774, pic16f870*, pic16f871*, pic16f872*, pic16f873*, pic16f873a*, pic16f874*, pic16f874a*, pic16f876*, pic16f876a*, pic16f877*, pic16f877a*, pic16f818*, pic16f819*, pic16c923, pic16c924, pic16f630*, pic16f676*, pic12c671*, pic12c672, pic12ce673, pic12ce674, pic12f629*, pic12f675*, pic18f242*, pic18f248*, pic18f252*, pic18f258*, pic18f442*, pic18f448*, pic18f452*, pic18f458*, pic18f1220*, pic18f2220*, pic18f4220*, pic18f1320*, pic18f2320*, pic18f4320*, pic18f6520*, pic18f6620*, pic18f6720*, pic18f8520*, pic18f8620*, pic18f8720*, pic18f6585*, pic18f8585*, pic18f6680*, pic18f8680*, pic18f2515*, pic18f2525*, pic18f2585*, pic18f4515*, pic18f4525*, pic18f4585*, pic18f2610*, pic18f2620*, pic18f2680*, pic18f4610*, pic18f4620*, pic18f4680*, pic18f6525*, pic18f6621*, pic18f8525*, pic18f8621*, pic18f2439*, pic18f2539*, pic18f4439*, pic18f4539*, pic18f2331*, pic18f2431*, pic18f4331*, pic18f4431*, pic18c242, pic18c252, pic18c442, pic18c452, pic18c658, and pic18c858.
The devices marked with a star (*) can be autodetected, so they need not to be specified. If code protection is active on the chip, autodetection may not work.

I do not know if all the chips work or if any other than pic16c84, pic12f675, pic16f676, pic16f76, pic16f88, pic16f876a, pic16f628, pic18f1320, and pic18f458 work, these I have tested myself. Default is to autodetect the device by reading configuration memory location 0x2006. If no device id is present, the default is pic16c84. If reading location 0x2006 with 14 bit programming algorithm fails, the PIC18 programming algorithm is used to read configuration memory locations 0x3ffffe and 0x3fffff. To add a new supported chip type, just edit the table in file hexfile.cc.

The hex file addresses (in IHX16 format) used are the ones specified by Microchip. This example is for pic16f628:

0x0000-0x07FF
: Program memory, 2048 words * 14 bits.

0x2000-0x2003
: ID locations.

0x2006
: Device id (not present on older chips), not present in hex file

0x2007
: Control word fuses

0x2100-0x217F
: Data memory, 128 bytes * 8 bits.

The addresses in IHX32 and IHX8M files are not word addresses but byte addresses. Divide those addresses by 2 and you get the same addresses as in the example above.

This example is for pic18f1320. These addresses are byte addresses. IHX32 is the only option for saving PIC18 family programs in hex files.

0x000000-0x001fff
: Program memory, 8192 bytes of 8 bits, or 4096 words of 16 bits.

0x200000-0x200007
: ID locations.

0x3ffffe-0x3fffff
: Device id, not present in hex file

0x300000-0x30000d
: Control word fuses

0xf00000-0xf000ff
: Data memory, 256 bytes * 8 bits.

Before interfacing with the PIC chip, Picprog calibrates its delay loops by checking the clock speed of the CPU and whether the CPU supports the TSC feature. On Linux, /proc/cpuinfo is read. On Windows/Cygwin, /proc/cpuinfo is read and CPU clock frequency is estimated. Therefore the clock frequency displayed on Windows is not necessarily exactly the true clock frequency.

If Picprog is run as root user, or otherwise gets capabilities to define itself as a real time task, a different timing method is used. The system call nanosleep() is used for delays.

7. Reading PICs

Simple instructions:

1. Insert the PIC chip into the socket in the programmer, or connect the in-circuit programming cable to your device.
2. Connect the programmer device to a serial port.
3. Read the device with command:

```
picprog --output ofile.hex --pic /dev/ttyS1
```

The reading options are:

--output-hexfile *path*, -o *path*
: Specifies the output hex file. The file will be written in IHX16 format, unless otherwise specified by the --ihx8m or --ihx32 options. For PIC18 family devices, file will be written in IHX32 format.

--skip-ones
: When reading the PIC device, do not consider the all-ones memory locations to be programmed, and skip them in the hex file output. For 14 bit devices, this skips the program memory locations that have hex value 0x3FFF, and for PIC18 family devices, this skips byte values 0xFF. In data memory locations that have hex value 0xFF are skipped.

--ihx32, --ihx16, --ihx8m
: Select the output hex file format to be either ihx16 or ihx8m, respectively. The default is ihx32 for PIC18 family devices and ihx16 for others.

--pic-serial-port *device*, -p *device*
: The device name of the serial port the programmer is connected to. Default is /dev/ttyS0.

8. Exit values

Exit values are as defined in <sysexits.h>:

EX_OK, 0
: no error

EX_USAGE, 64
: command line option syntax error

EX_IOERR, 74
: file or serial port io error, or after-programming verification failed

EX_DATAERR, 65
: input file syntax error, not in IHX8M, IHX16, or IHX32 format

EX_NOINPUT, 66
: unable to find input file or file open failed

EX_UNAVAILABLE, 69
: user or signal interrupted programming

9. Internals

Source files and their contents:

picport.cc, .h
: class picport: manipulates the serial port hardware. With this class you can execute programming commands like read a word, program a word, increment address etc. Look at picport.h for details.

hexfile.cc, .h
: class hexfile: contains a PIC memory image. You can load and save the contents of this class to a file, and you can program and read it from the PIC chip. Programming uses class picport.

program.cc, h
: class program: just some generic option handling.

main.cc
: Just the main () to parse command line and call class hexfile to do its job.

10. Changes

This document has not changed much since it was first released with the 1.0 programmer. The changes include some information about new software and more accurate links to PIC information. New options to select type of device other than pic18c84 are also present.

2003-08-10 version 1.2
With help from Taneli Kalvas changed the schemantics diagram and pcb mask to ground the RB4 pin, selecting high voltage programming on for example pic16f628.
Implemented preservation of OSCCAL and other calibration data.
Added automatic detection of devices based on location 0x2006.
Merged Bart Goossens's changes to implement PIC16F73. I hope it works.

2003-08-21 version 1.3
Autodetect more chips. Fix programming of chips with OSCCAL. Fix erasing some chips - erasing and resetting code protection is now performed the hard way: all methods are tried regardless of chip type. --erase now works also without --input-hexfile flag.

2004-01-02 version 1.4
Add option --force-calibration to program OSCCAL and BG bits. Implement programming algorithms for 16f87/16f88 and 16f87Xa. Revise some timings on programmer reset to avoid operating voltage to dip. More verbose output on how many locations actually were burned.

2004-03-02 version 1.5
Fix PIC16F87xA configuration word burning. Remove the 10k resistor added by me from the PCB and schema. Add support for PIC18 family. Make the DTR be held low as long as possible. This may improve the reliability and limit stress on RS-232 port.

2004-03-19 version 1.6
Now compiles on Windows with Cygwin DLL version 1.5.8 and later. More accurate timings result in shorter programming times compared to previous versions. These timings use the CPU RDTSC instruction on x86 and AMD64 plattforms. PIC16F87/88 second configuration word programming implemented.

2004-04-28 version 1.7
Fixed 16c OTP and UV erased parts EPROM programming to use 100µs programming/overprogramming pulses. Use real time priorities and nanosleep() for delays if run under root priviledges. Relax timings so that they work out of the box with longer cables. Fixed PIC12F629 / PIC12F675 / PIC16F627a / PIC16F628a / PIC16F648a / PIC16F630 / PIC16F676 programming with >1.3GHz i386 CPU.

11. Other available programmers

PiKdev
: PiKdev is a simple graphic IDE for the development of PIC-based applications. PiKdev is developed in C++ under Linux and is based on the KDE environment. It includes a programming engine which allows programming various flavors of PIC microcontrollers via classic (ie: D. Tait or JDM compatible) programming hardware connected to the parallel port or to the serial port.

PP06
: Linux and Windows programming software that knows about 83 pic's and 6 programmers.

XWisp
: Programmer software written in Python by Wouter van Ooijen.

serp-0.5 (serp-0.5.tgz)
: A serial port programmer software for Linux, written in c++, author Ralph Metzler. It directly handles the serial hardware, standard 16450/16550 compatible uarts, and needs root priviledges for that. Unmaintained since 1996.

jdm84v23 (jdm84v23.zip, pgm84v23.zip)
: A serial port programmer, schema (gif) and DOS software. The hardware manages with rs-232 interfaces with low voltage output, even as low as ±7V is fine. This is the programmer I use with linux with my own software.

  I modified the circuit to include connections to pins that are needed for programming some PIC microcontrollers.

  There is a new version of the PCB available on Jens Madsen site. It supports more chips without jumper wires: PIC-Programmer 2.

pip-02/com84 (pip-02.zip)
: A serial port programmer, schema (gif) and DOS software. Needs +12V rs-232 positive voltage level.

prog84-3
: A parallel and serial port programmer, and software for Linux and dos, written by Wim Lewis and Frank Damgaard. There is some experimental code for an USB parallel port device.

dvtait84, pic84faq (dvtait84.zip, pic84faq.zip)
: A parallel port programmer, schema (ascii) and DOS software with basic and turbo-C sources included. Author David Tait. He has a lot more stuff, and some new designs to program other PICs in his links page.

mjcox84 (mjcox84.zip)
: A parallel port programmer, no schema, written in assembler for DOS with 486/33 timings. Very limited. Author: Mark J Cox, m.j.cox@bradford.ac.uk.

ngoodw84 (ngoodw84.zip)
: A parallel port programmer and disassembler, no schema, seems to use pins DATA1 = data and DATA2 = clock and needs external programming voltage. From Everyday Practical Electronics, February 1996, author Derren Crome. Disassembler by Nigel Goodwin nigelg@lpilsley.demon.co.uk.

Minimized PIC16C84 Programmer
: Parallel port programmer, DOS software, needs external 13V power source. Author Stephen M. Nolan.

12. Available languages

Free compilers

GNU PIC Utilities
: gputils have assembler, linker, disassembler and library utilities much like MPASM. They support all PICs.

Yappa
: A graphical development environment for PIC16F84 by Mark Colclough at the University of Birmingham. Yappa combines into a single application the editor, assembler and programmer interface that are needed to program a PIC. Picprog is used as the programmer backend.

picasm112b (picasm112b.tar.gz)
: Assembler in ANSI-C by Timo Rossi. Outputs both IHX16 and IHX8M. You can also find disassemblers for 12 bit and 14 bit PIC's on Timo's site.

asm_c84 (asm_c84.zip)
: Assembler in ANSI-C by James Cleverdon, jamesc@sequent.com. No INCLUDE, no IF, outputs IHX8M. The manual.

SIL
: I have heard of free SIL language (something like Pascal/M2) compiler for PIC.

JAL
: Just Another Language, a Pascal like high level language. *"... I wanted a HLL which is better mached to the PIC architecture, to my programming habits, and which I could explain to the kids of the local electronics club without giving a full course on computer architecture."*

Commercial demos

HI-TECH Software C compiler
: A demo is available of their compiler.

elabtronics CoreChart
: CoreChart (formerly named bitset) is an icon-based development tool. 30-day trial available.

13. Other software

GPSIM
: PIC simulator.

14. History

Picprog was first written and released in May 1997. Around that time I briefly experimented with microcontrollers. I found no Linux software for the cheap serial programmer hardware by Jens Madsen, and I wanted to use that one as it was so simple to build. Only later did I learn about serp-0.5. Anyway, that one directly programmed PC style serial port hardware while I wanted to use standard UNIX methods of accessing serial ports. Linux was missing an IOCTL to force BREAK condition (steady +12V) on the serial data transmit line. This kind of functionality existed for example in Solaris. No problem, I created a patch for Linux kernel versions 2.0.30 and 2.1.42, submitted it, and it was included in mainline kernel versions 2.0.32 and 2.1.45.

Since the time I first wrote the software I did not work with microcontrollers at all for years, though I

maintained Picprog by fixing obvious compilation problems and updating documentation. Version 1.0.1 was put together in May 2001 and included mainly documentation fixes. In June 1997 I had worked on adding support for different memory sizes of different PIC chips, and these changes and again documentation updates were released as version 1.1 in February 2002. I also found the programmer hardware I thought I lost a few years back, and was able test that it still works.

Picprog-1.0 was ported to FreeBSD and included in the distribution around September 1999. MIT MASLab 6.186, a student-run robotics course, seems to have used it since January 2001. Recently this documentation page has attracted steadily over 1000 visits per month, so I guess someone is finding it useful.

Nowadays I mostly test Picprog with new chips, and sometimes I start a new project like KanSat satellite or 8-PIN PONG, and never finish them..

15. Copyright notice

The author may be contacted at:

Email: Jaakko.Hyvatti@iki.fi
URL: http://www.iki.fi/hyvatti/

Please send any suggestions, bug reports, success stories etc. to the Email address above. To avoid my spam filters, please put the word 'picprog' somewhere on the subject line.

If you want to support the developement of picprog, please make a donation to Jaakko via PayPal (click on the button).

Jaakko Hyvätti /Jaakko.Hyvatti@iki.fi /+358 40 5011222