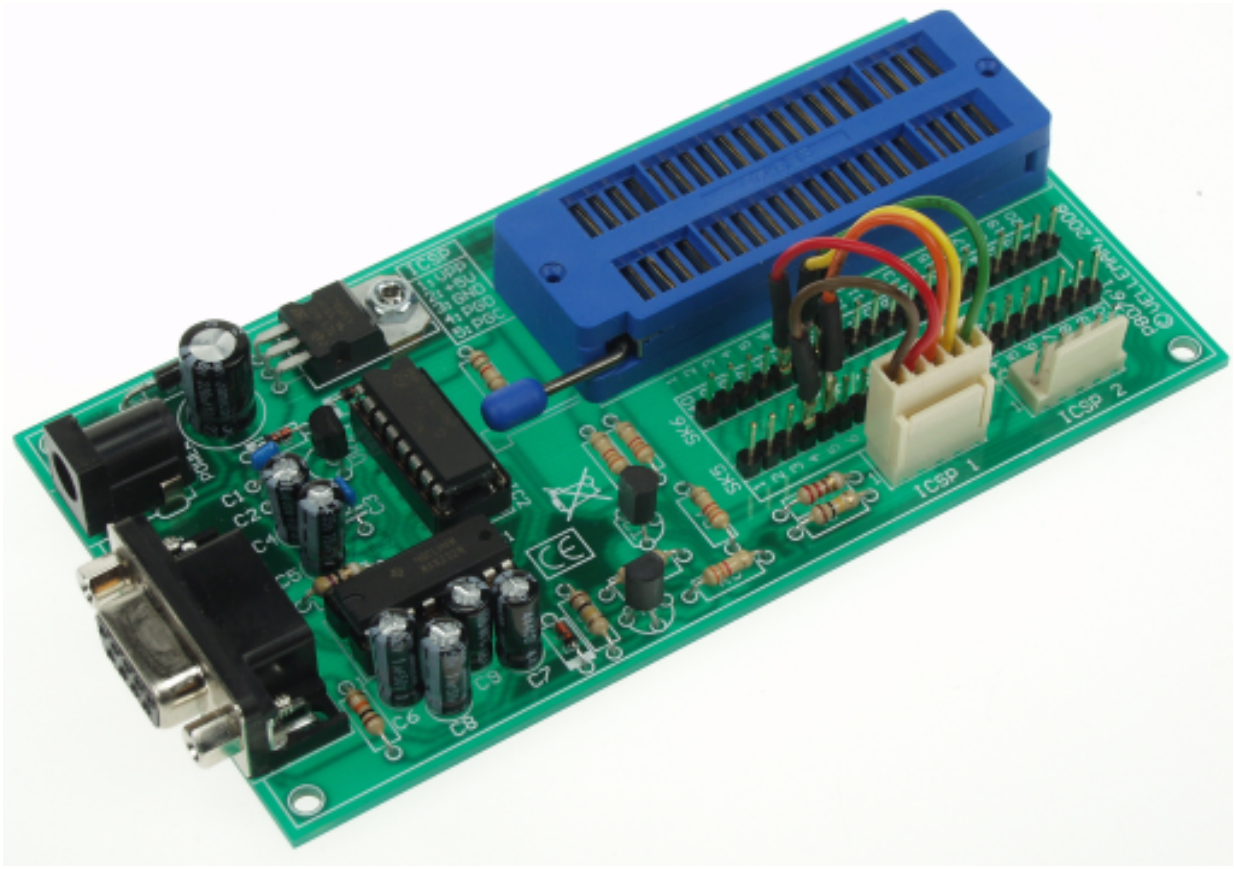



# VM134
## (K8076)

# QUICK GUIDE

velleman® MODULES

# 1 General information

## 1.1 Introduction

Thank you using the Velleman produkts. The VM134 (K8076 kit version) us a multifunctional and instructional programmer, aimed at programming a selection of Microchip ® PIC™ FLASH microcontrollers. These FLASH controllers can be reprogrammed many times, making them an appropriate tool for teaching a PIC programming language. Another advantage of reprogrammable controllers is that software of a device where it is implemented can easily be updated.

The VM134 PIC programmer is a ready-to-use version of our unassembled K8076 programmer. Hence, the VM134 can also be referred as the K8076 in this manual and software.

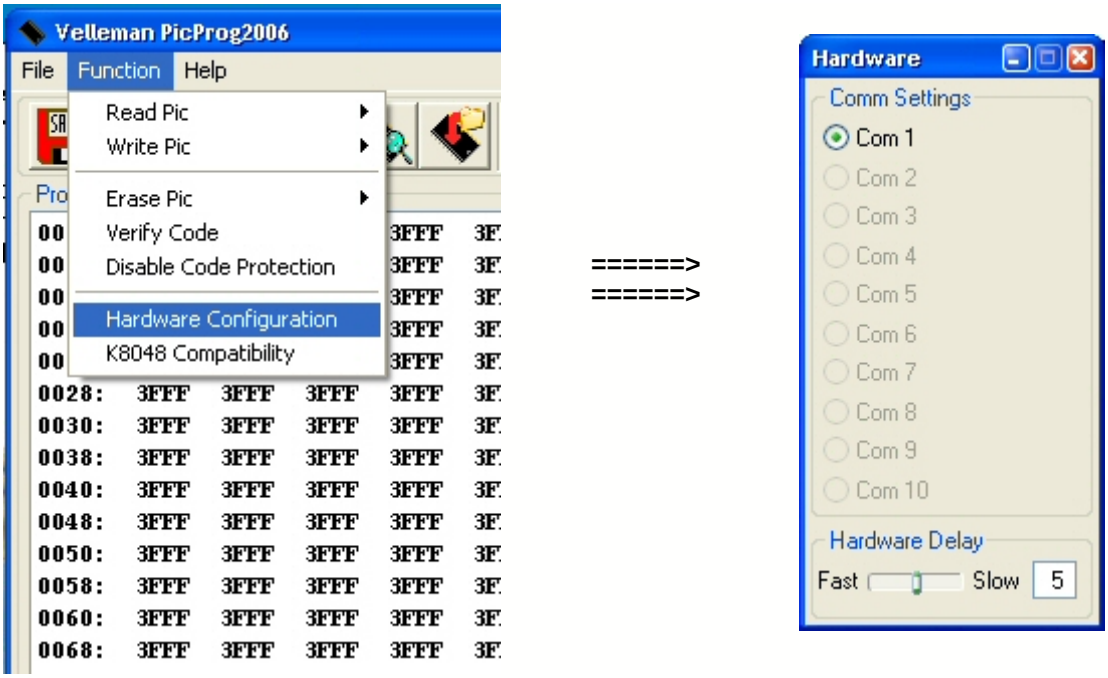
There are 2 important stadiums in the use of this programmer. First, a programme code needs to be written, usually in a graphic environment (IDE). For PICs that is MPLAB(™) by Microchip. This complete software package can be downloaded for free from the website www.microchip.com. An easier method is writing a programme in an ASCII word processing programm like e.g. Notepad, installed on every Windows PC.

Information concerning the commands used in the assembler language for every controller type can be found in data sheets on the Microchip website. An exemple programme is included on CD.

## 1.2 Connection cable

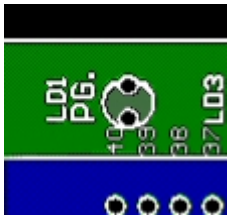
Connect your programmer to a free serial RS232 port of your PC. This serial port must be 100% IBM compatible and 16550 UART compatible. This kit does not use the RS232 protocol but an emulated I2C protocol via handshaking.

A **UART**, *universal asynchronous receiver / transmitter* executes the main tasks in the serial communication of computers. The chip converts incoming parallel information into serial data which can be sent through a communication line. A second **UART** will be used to receive the information. The **UART** executes all necessary tasks like e.g. timing, parity control etc. needed for the communication. The only extra chips needed are the line drivers converting the TLL-level signals into line voltages and vice versa.



## 1.3 connection & test

- Make sure there is no controller in the ZIF socket.
- Connect the serial connector to the serial port.
- Connect a 15VDC power supply. This voltage may or may not be regulated since the PCB is equipped with a voltage regulator (a 12VDC non regulated adapter will work since the terminal voltage is about 15 to 16V).
- When switching on the power supply, LD1 or "Power Good" LED will light. This LED indicates that the programmer is live and that the controller is provided with +5V.




- Start the "PICprog2006" software and click the icon on the upper right, viz "Hardware connections" **(Fig 1.0)**


**FIG. 1.0**

- By clicking the LD3, LD2 and LD4 LEDs with your mouse they should light on the PCB (see PIC2) **(Fig 2.0)**

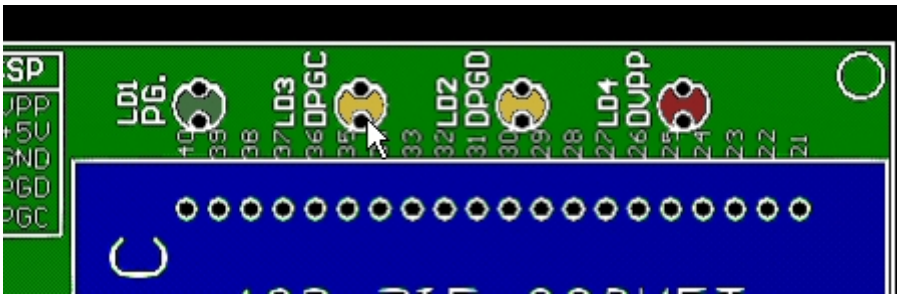



**FIG. 2.0**

- Press "Run hardware datalines test" to start an automatic LED lighting sequence. These LEDs must light synchronous with the screen **(see PIC3)**. Press the button again to stop the testing procedure. Attention: make sure there is no PIC in the socket!.


- You are ready testing the programmer when the test went well. If not, find the hardware error on the print or the incompatibility to the PC to avoid irreversible damage the PIC controllers.

**ATTENTION:** When a communication problem between the PC and the VM134 or a hardware problem with the VM134 occurs, clicking on the test button or the LEDs will not be possible. The following message can appear

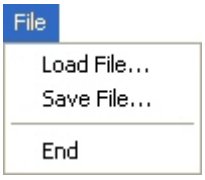

If there is communication in the programmer socket between the VM134 and the PIC, following message will appear:


## 1.4 Disclaimer

Velleman Components NV and the software designer cannot be held responsible for any hardware and/or software failure or damage caused by the use of it

# 2 Menu bar

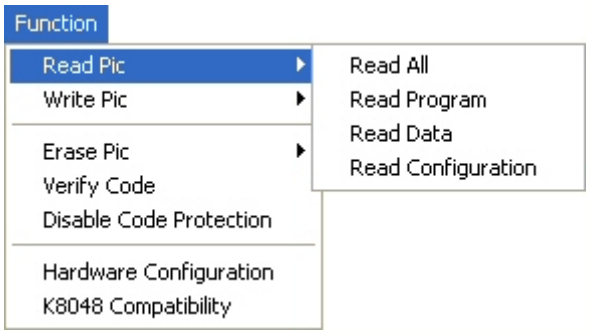
## 2.1 File




"Load File" : load a Hex file
"Save File" : save a Hex file
"End" : end the programme

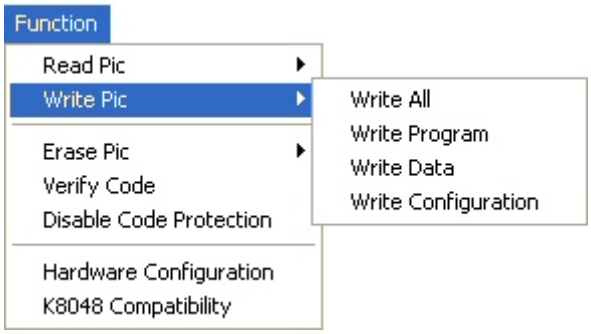
## 2.2 Function


### "Read PIC"



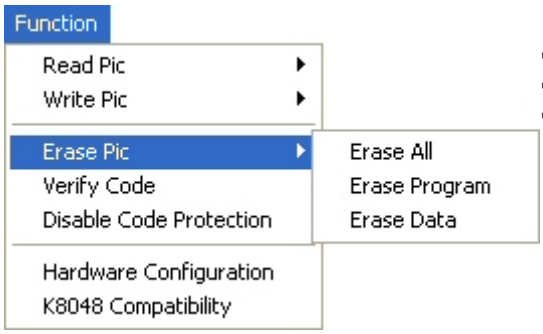

"Read All" : read all available data
"Read Program" : read programme data only
"Read Data" : read EEProm data only
"Read Configuration" : read configuration bits

### "Write PIC"




"Write All" : write available data only
"Write Program" : write programme data
"Write Data" : write EEProm data
"Write Configuration" : write the configuration

### "Erase PIC"




"Erase All" : delete all available data
"Erase Program" : delete programme data only
"Erase Data" : delete EEProm data only

## "Disable Code Protection"


Make the PIC available after it was programmed with a code protection (all data is deleted).

## "Hardware Configuratie"


"Comm Settings" : selection of all available RS232 ports
"Hardware Delay" : if the communication speed is to high.

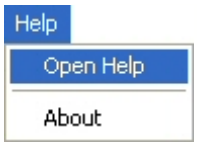
## "K8048 Compatibility"


Lets you use (where possible) the K8048. Not possible with all PICs.

Click the "K8048 Compatibility" option in the "Function" menu to establish the compatibility between the PICprog2006 software and our K8048 (=VM110) PIC programmer and experiment board. Some PICs from the actual list cannot be programmed with the K8048 since the K8048 hardware does not allow it.

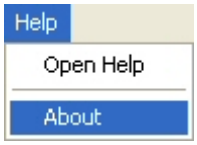
## 2.3 Help menu


### "Open Help"




Consult the kit's help file

### "About"




Call up the programme version

# 3 Button Bar

## 3.1 Button Bar


Click on the button for further information

1. *Save data as HEX file*
   Save the controllers content and save it as INHX8M file onto the hard disk. Attention: controllers with an activated "code protect" bit cannot be read because of the copyright by the manufacturers. Memory dump HEX files of the 18Fxxxx family will be written as INHX32 files.

2. *Load a HEX file*
   Reading of a file from a storage device to the software memory. Attention: the file needs to have an INHX8M, INHX16 or INHX32 format. The compiler (e.g. MPASM) must be configured so it can generate an INHX8M file.

3. *Load the Mpasm editor.*
   Start the included Microchip Assembler. Updates can be found with the complete Microchip "MPLAB" compiler on the website: www.microchip.com.

4. *Write all data to the PIC*
   Write the loaded HEX file to the controller in the ZIF socket.

5. *Load all data from the PIC*
   Reads the controller content and places it into the software buffer memory. Attention: controllers with an activated "code protect" bit cannot be read because of the copyright by the manufacturers.

6. *Write the data from Hex file directly to the PIC*
   Write a HEX file directly to the controller without loading it in the buffer memory first.

7. *Call up a help file.*
   Start the on-line instructions manual of the PICprog2006 software.

8. *Choice bar for the PIC*

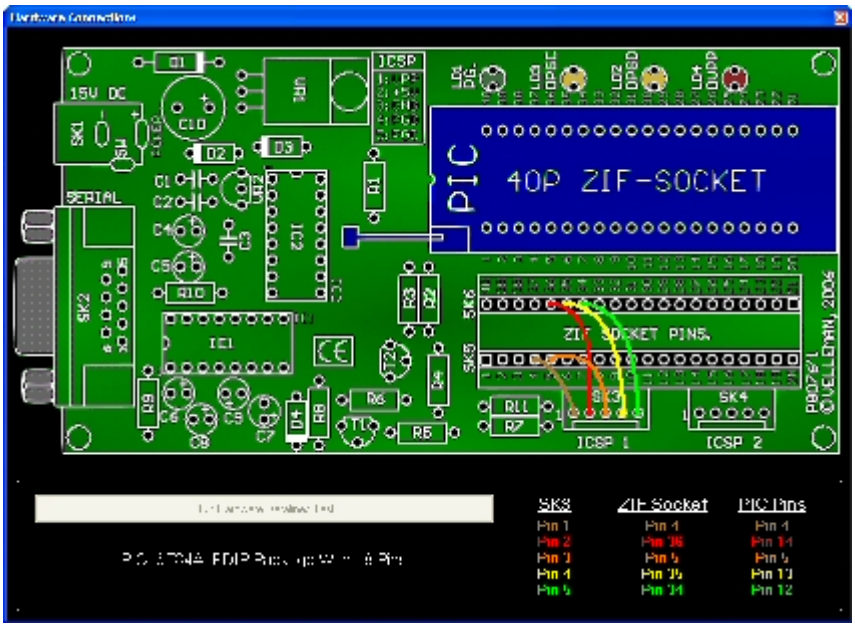

"PIC Family": Select the desired controller family. The family and type configuration has been separated to shorten and to simplify the choice list.


Select the desired controller belonging to the family chosen in point "8". Controllers preceded by a green "V" sign have been tested by Velleman with this PIC programmer whereas controllers preceded by a yellow "X" sign have been implemented in the actual software but not tested with this programmer. When meeting with difficulties, simply send the controller with an explanatory letter to the Velleman main office, attn the support department.

We will do our utmost to solve the problem. We can not offer any guarantee since we depend on the protocol data released by Microchip.

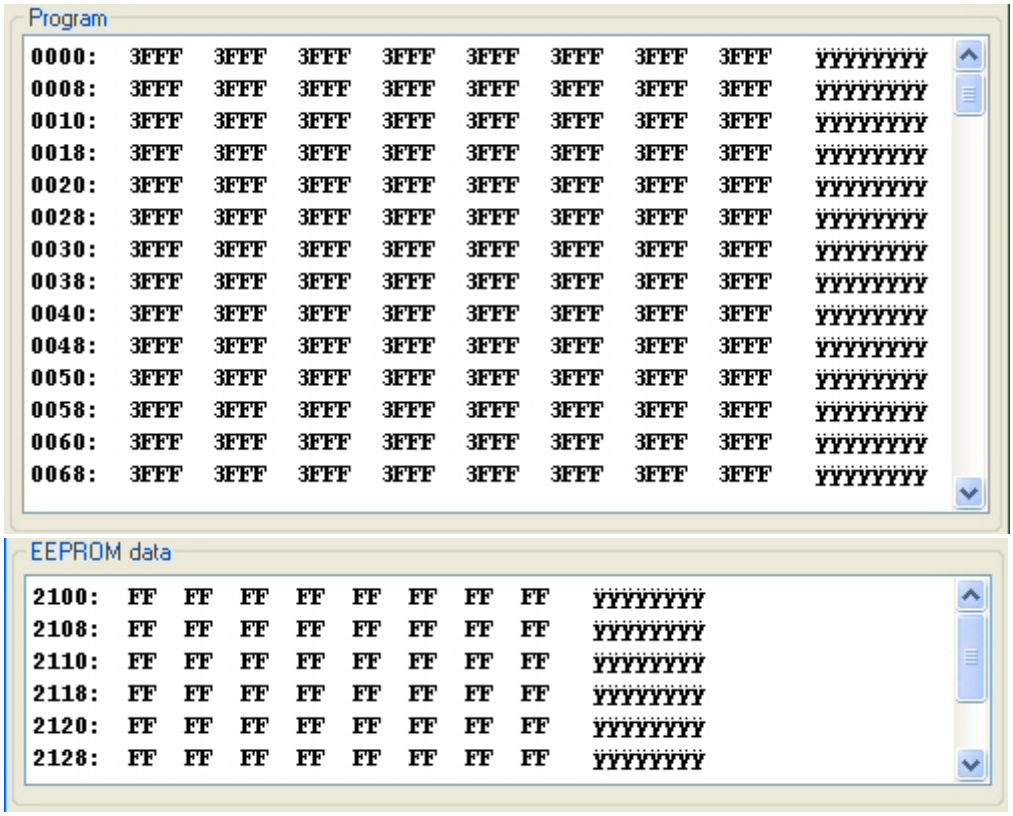
9. *Hardware configuration*

   Visual representation of how to connect the PIC cable in order to program the desired controller



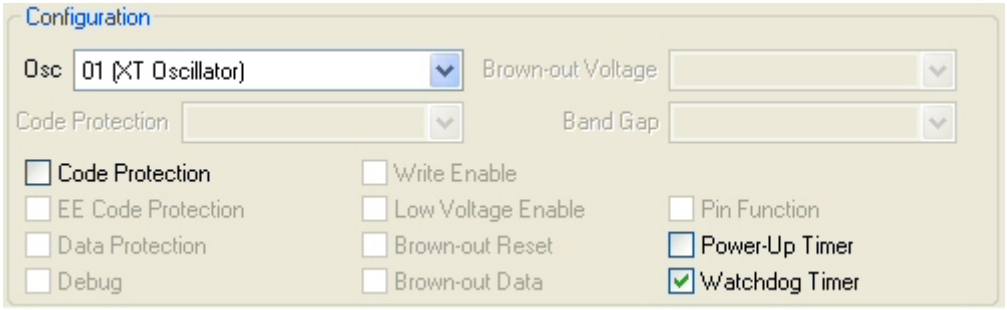

# 4 Windows

## 4.1 Program

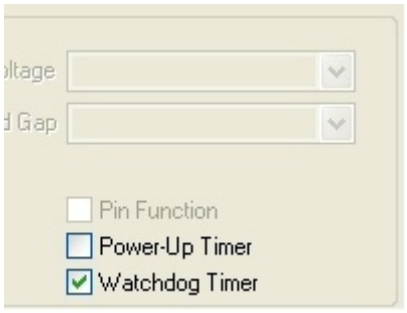



Here you can find the programme code. This is the hexadecimal upcode the controller will execute. You can also see the data code. These are the values in the EEPROM memory of the controller. This window only appears with controllers with an EEPROM memory (e.g. PIC16F627).

## 4.2 Configuration




Consult and modify if necessary the programming options. We recommend executing these configurations directly in the assembler programme through the "__CONFIG" compiler directive. See "BLINKLED.ASM" for an example.



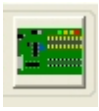

Enable or disable the PIC controller options. You can also set them via the __CONFIG compiler directive in the assembler programme. For more information concerning these options please refer to the data sheets of the used controller on the Microchip website, viz www.microchip.com.

# 5 Use

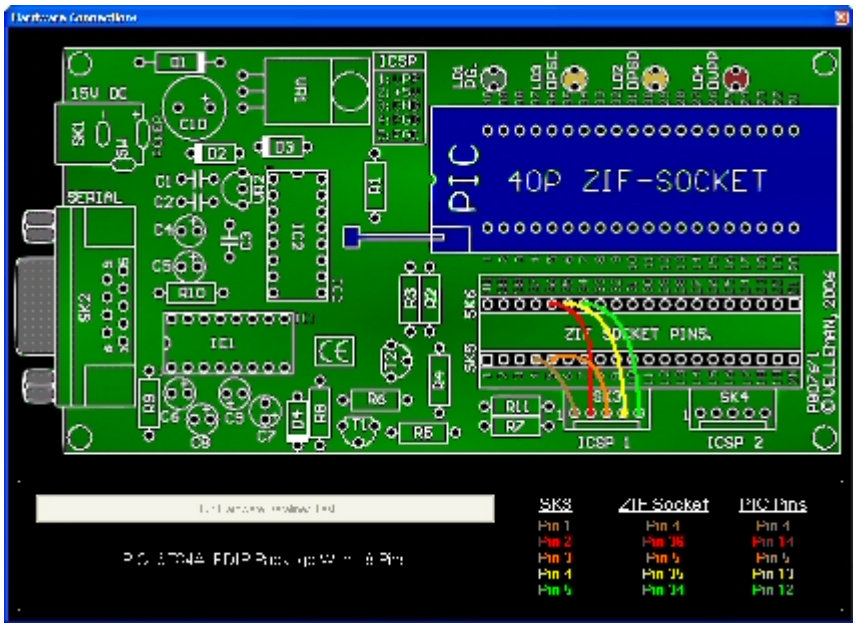
## 5.1 Select PIC

1. Choose the correct PIC family on the upper right corner, e.g. "PIC10F", "PIC16F"...


2. Choose the correct type in the adjacent menu, e.g. "PIC10F200"...

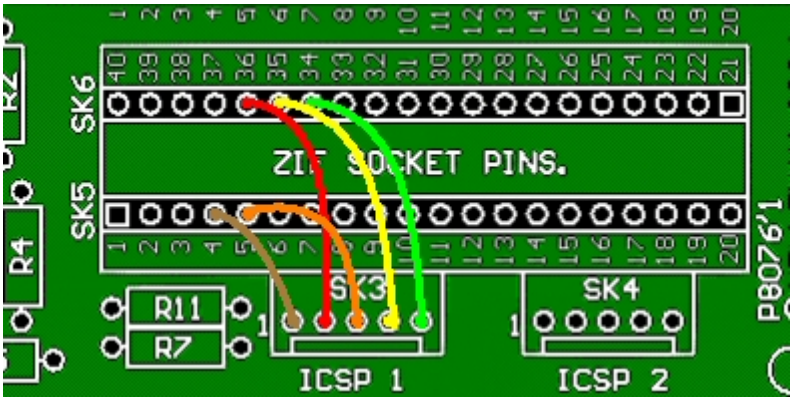
3. Click the "Hardware connections" icon 

- *You can see a picture showing how to connect the PIC configuration patch-cable with the pins from the ZIF socket. When using the included cables, the cable colour code will correspond to the colour code on the display.*




*PIC configuration patch-cable*

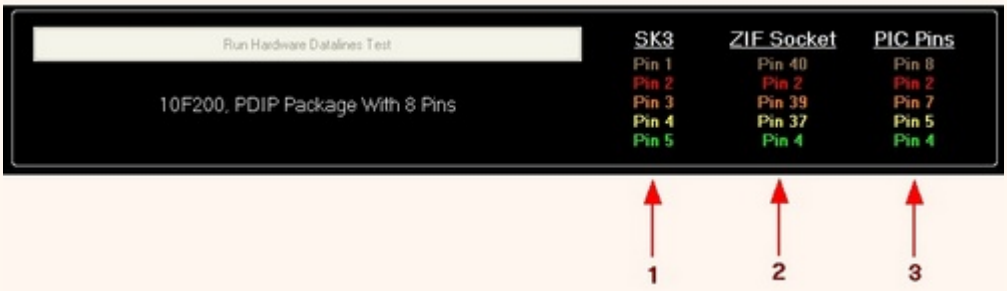
- *The pins from the ICSP1 or ICSP2 connector must be appropriately connected before placing a PIC controller into the ZIF socket*



- *The ICSP1 and ICSP2 connectors can be used arbitrarily since they are identical on the hardware level.*

**Hint**: The ICSP connectors can also be used to program controllers external to the PCB. The cables leading to the print need to be as short as possible (+/- 20cm)

**Hint:** The picture shows how the ICSP connector SK3 and SK4 (1) is connected to the pins of the ZIF socket (2) and the PIC controller (3).




- If everything is appropriately connected, place the controller into the ZIF socket and pull the lever.

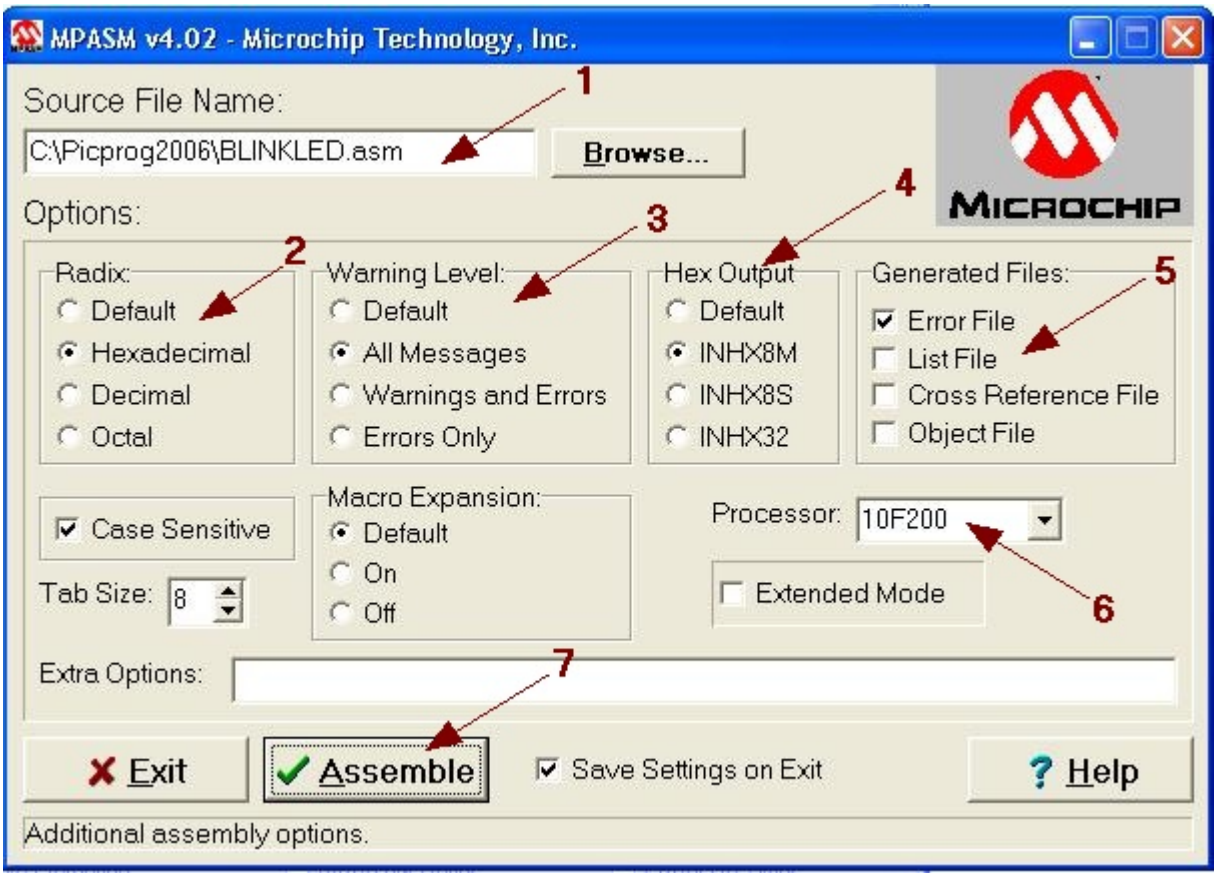
**Hint:** If LD1 turns off when placing a PIC into the ZIF socket, it indicates that an internal short-circuit in the component occurs or the PIC cable is badly connected generating a short-circuit. The VM134 ihas a limited protection against such short-circuits through R10.

## 5.2 Programming the PIC controller

A simple explanatory application will explain how to program and to test a PIC. The application is simply a blinking LED. The used controller in the example is a PIC10F200.

**STEP 1: Compiling your code**

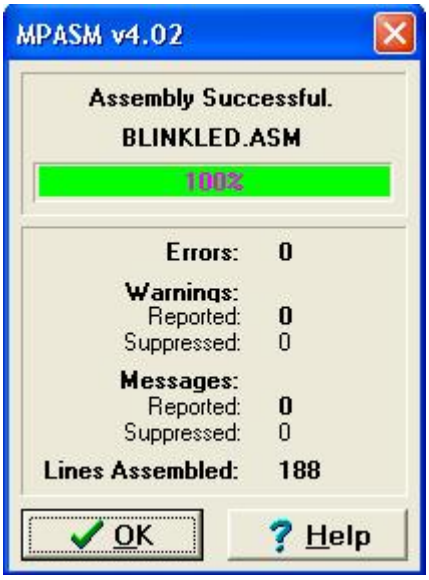
- Start PICprog2006
- Click the "MPASM" icon.
- Read the ".ASM" file.




1. The file to compile is generally of the ".ASM" type.
2. Set the radix that will be standard accepted (setting the radix in the .ASM file will have priority over this setting)
3. Leave the setting on "All Messages" so all error messages and warnings will be recorded in the .ERR or .LST file.
4. Choose the output format. The format can be INHX8M, INHX16, INHX32.
5. Choose which files will be generated by the MPASM, e.g. a file containing error messages…
6. Choose the PIC controller to be programmed.

* For more information please refer to the MPASM help function or the information on the *Microchip.* website

- Press the "ASSEMBLE" (7) button when all setting are correctly implemented (7).
- When the code assembly does not contain any errors, following screen will appear.




Before programming the controller, make sure there are no errors in the assembler code (Errors => 0)

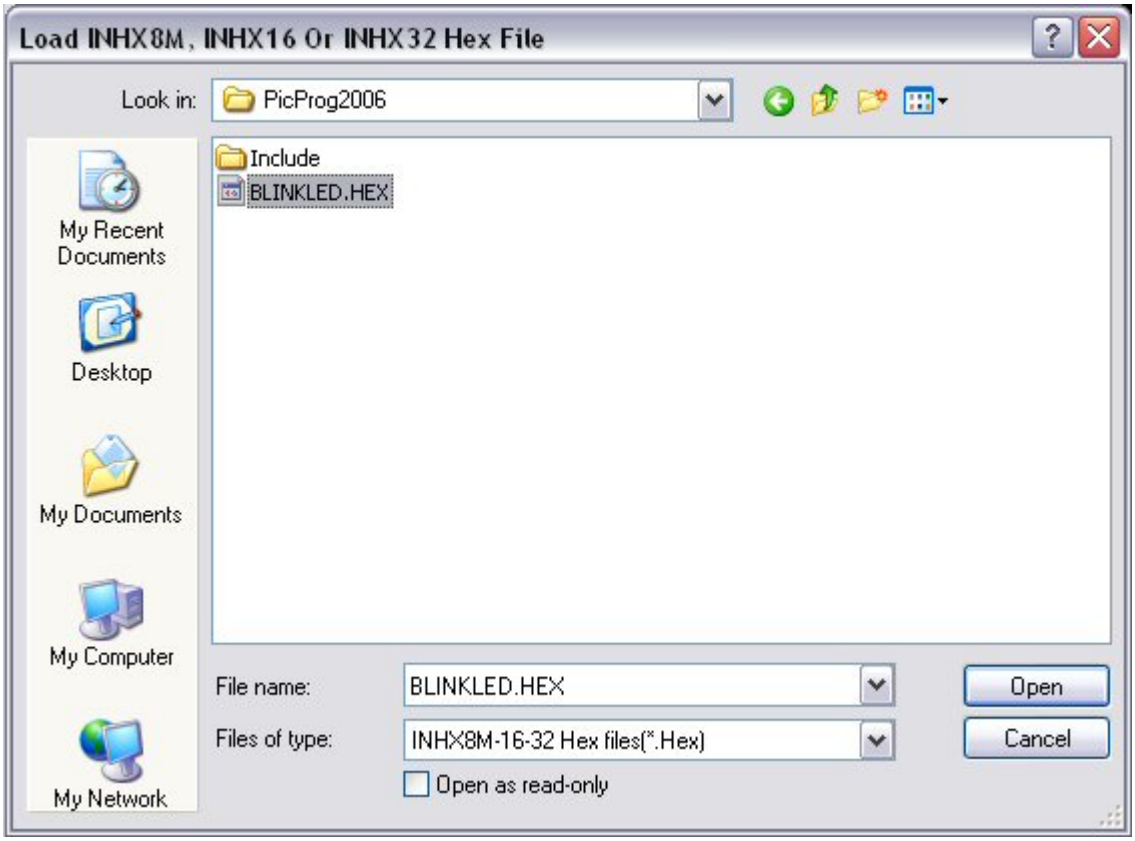
Causes of a communication breakdown:

- PIC controller type does not correspond with the chosen software type
- power supply of the VM134 is too low (15V)
- incorrect PIC selection through patch cables
- defective PIC controller
- PIC controller status cannot be put in program mode

**Remark:** This PIC programmer will not be able to program controllers which simultaneously use the internal oscillator and the MCLR pin as input. Programming such a controller can damage it beyond reuse.

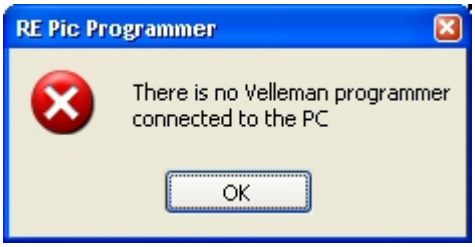
## STEP 2: Programming the controller

- Start PICprog2006
- Click the "LOAD HEX FILE" icon. Following screen will appear:




- Click the desired HEX file (e.g. BLINKLED.HEX)

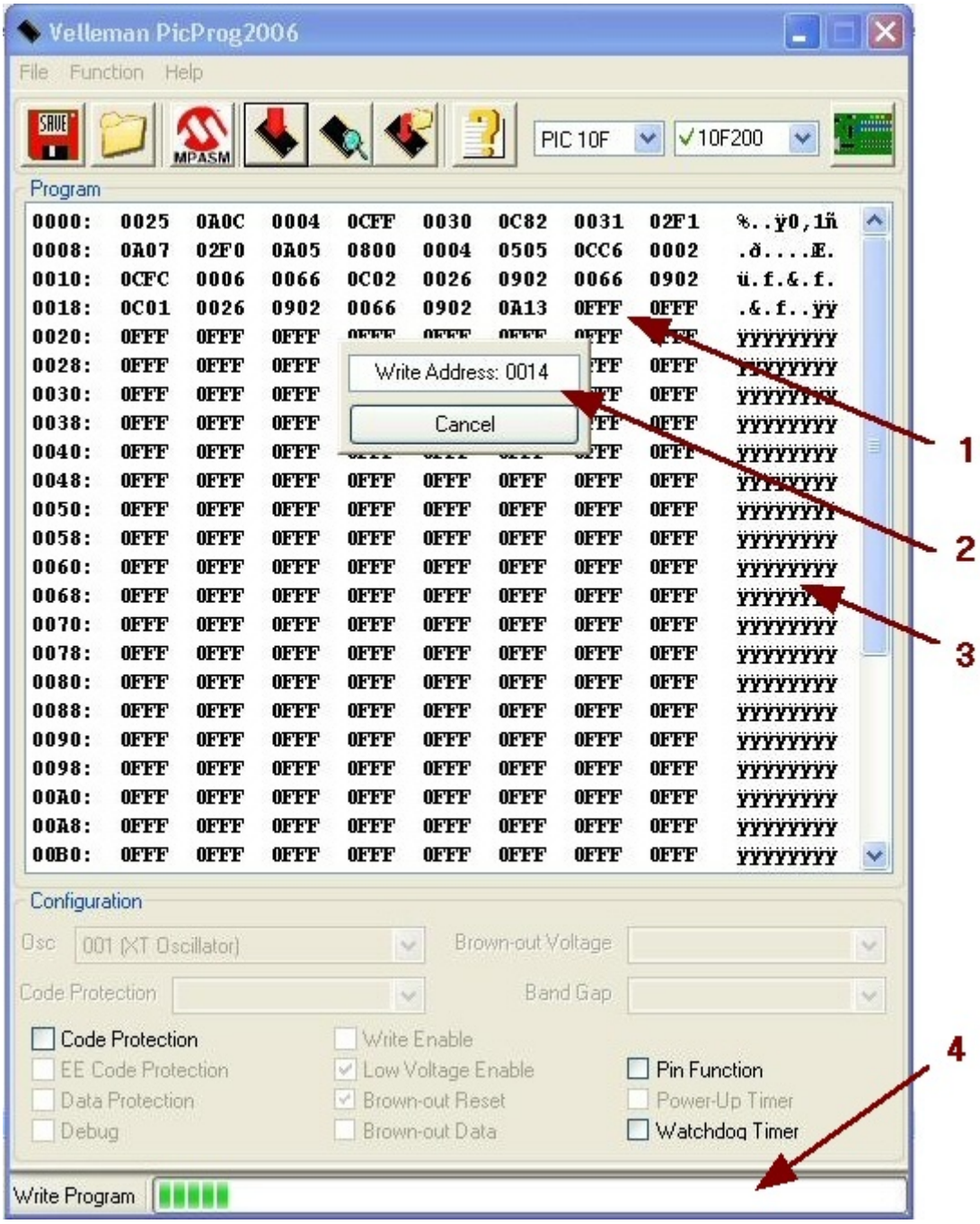
  In case of a hardware error, following message can appear. Check the VM134 and/or the controller selection




- Click the "WRITE ALL DATA TO PIC" icon.

  The programme will ask for a confirmation:
  Click "YES" when you are sure to overwrite the controller.


- You will see the progress of all kinds of actions like e.g. deleting, programming, controlling and setting the controller parameters.




1. The HEX code that will be saved into the controller.
2. Address counter: indicates where in the memory the device is reading or writing.
3. The ASCII version of the code.
4. Progress bar: Visualize the percentage of the programming or the reading process.

- When the programming process is finished, push the lever of the ZIF socket and remove the controller.

---

**The controller can irrevocably be damaged if something goes wrong during the programming process (e.g. connection cut-off, software interrupt van software on the serial port etc.). For more information please refer to the data sheets of the used controller**