Amadeus

µC based USB-programmer for Windows Computers for selected PIC and AVR controllers

Table of Contents

1 General Information

This project is based on a Microchip PIC18F242/252 microcontroller and Future Technology's FT232BM USB-to-Serial converter.

Currently, the software provides four programming algorithms for selected PIC and AVR controllers.

2 Putting the programmer into operation

This chapter describes what to do, to make Amadeus work. Maybe it is a good idea to glance over the entire document at first.

First of all, you need to order all necessary components. Find a list of all required components in the partlist on page 22.

Next you need to produce your own PCB. There's an external file, named "Amadeus_PCB.gif", with a higher resulution than the graphics in this document. The schematics can be found on page 20. Feel free to create your own layout, if you like. But be aware that at least the USB controller is only available as SMD package.

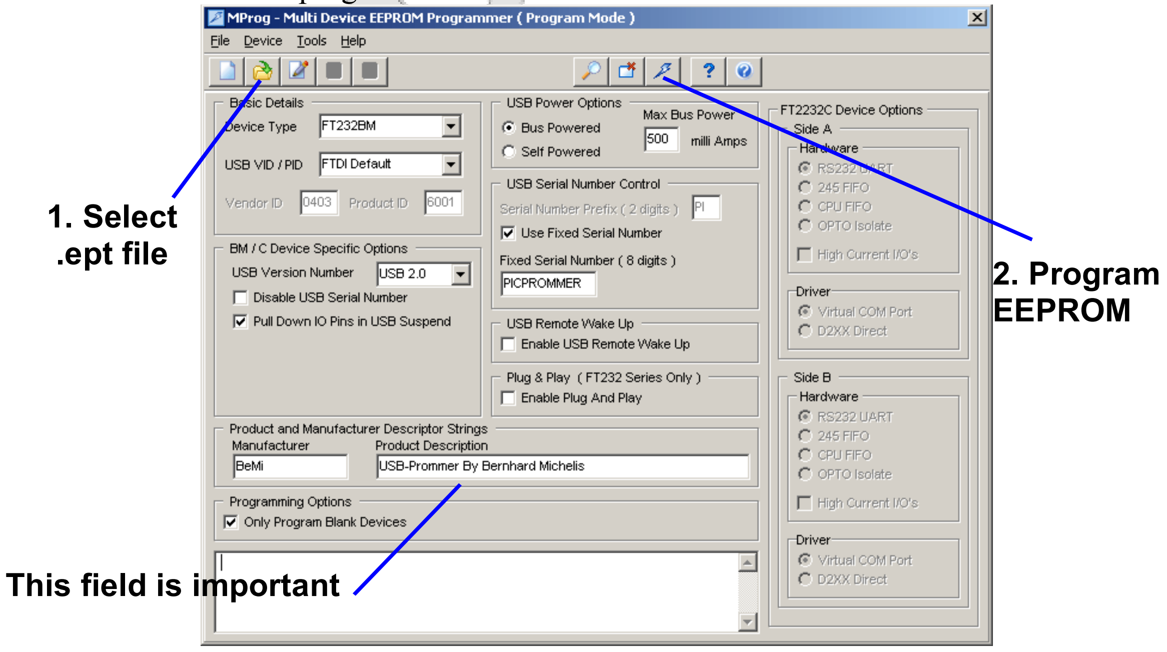
Than you have to assemble your PCB. See chapter "Assembly" on page 23 for details.

When the hardware is ready to work you have to do some programming.

Download two file from the internet. First of all the ***D2XX* driver package for the FT232BM (http://www.ftdichip.com/Drivers/FT232-FT245Drivers.htm)** and a tool named ***MProg* (http://www.ftdichip.com/Resources/Utilities.htm#MProg)**. Both can be found on the webside of Future Technologies.

Connect the programmer hardware to your computer. Windows should find a new USB device. Install only the D2XX driver not the VCP driver.

After rebooting, start ***MProg***. Load the programming package "Amadeus.ept" provided with this documentation and program it into the USB-EEPROM.



MProg - Multi Device EEPROM Programmer (Program Mode)

Basic Details
Device Type: FT232BM
USB VID / PID: FTDI Default
Vendor ID: 0403 Product ID: 6001

BM / C Device Specific Options
USB Version Number: USB 2.0
[ ] Disable USB Serial Number
[✓] Pull Down IO Pins in USB Suspend

USB Power Options
[● Bus Powered] ○ Self Powered
Max Bus Power: 500 milli Amps

USB Serial Number Control
Serial Number Prefix (2 digits): PI
[✓] Use Fixed Serial Number
Fixed Serial Number (8 digits): PICPROMMER

USB Remote Wake Up
[ ] Enable USB Remote Wake Up

Plug & Play (FT232 Series Only)
[ ] Enable Plug And Play

Product and Manufacturer Descriptor Strings
Manufacturer: BeMi
Product Description: USB-Prommer By Bernhard Michelis

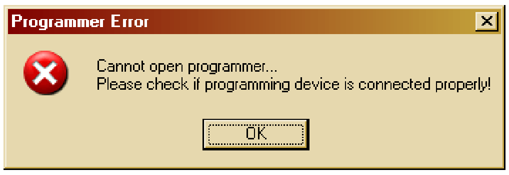
Programming Options
[✓] Only Program Blank Devices

FT2232C Device Options
Side A
Hardware: [● RS232 UART] ○ 245 FIFO ○ CPU FIFO ○ OPTO Isolate
[ ] High Current I/O's
Driver: [● Virtual COM Port] ○ D2XX Direct
Side B
Hardware: [● RS232 UART] ○ 245 FIFO ○ CPU FIFO ○ OPTO Isolate
[ ] High Current I/O's
Driver: [● Virtual COM Port] ○ D2XX Direct

Illustration 1: MProg UI

Disconnect the programmer from the computer and reconnect it. Probably Windows will find a new device, because we reprogrammed the USB-Controller. If you are requested to istall a driver, install the ***D2XX*** driver again.

Now it's time to start ***Amadeus***, the programming interface software provided with this documentation. When the message box below appears, the programmer cannot be found. If the EEPROM programming worked before, the EEPROM values might not be programmed correctly. See above, the **Product Description** must be written exactly this way.

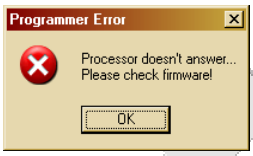

Programmer Error

Cannot open programmer...
Please check if programming device is connected properly!

OK

Illustration 2: Error Message if no hardware was found

If everything works correctly, you get the following Message, telling you, that the MCU is not programmed.

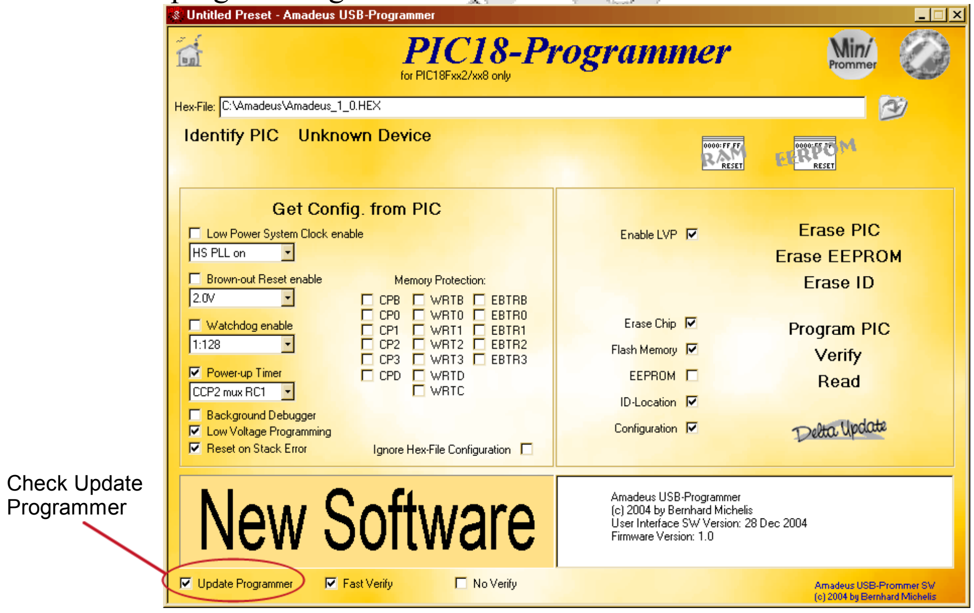

Programmer Error

Processor doesn't answer...
Please check firmware!

OK

Illustration 3: Firmware is missing in the programmer

The last step, before you can use your new programmer is to program the firmware. Therefore go to the PIC18 programming interface.



Untitled Preset - Amadeus USB-Programmer

PIC18-Programmer
for PIC18Fxx2/xx8 only

Hex-File: C:\Amadeus\Amadeus_1_0.HEX

Identify PIC   Unknown Device

Get Config. from PIC

- [ ] Low Power System Clock enable
- HS PLL on
- [ ] Brown-out Reset enable
- 2.0V
- [ ] Watchdog enable
- 1:128
- [✓] Power-up Timer
- CCP2 mux RC1
- [ ] Background Debugger
- [✓] Low Voltage Programming
- [✓] Reset on Stack Error

Memory Protection:
- [ ] CPB [ ] WRTB [ ] EBTRB
- [ ] CP0 [ ] WRT0 [ ] EBTR0
- [ ] CP1 [ ] WRT1 [ ] EBTR1
- [ ] CP2 [ ] WRT2 [ ] EBTR2
- [ ] CP3 [ ] WRT3 [ ] EBTR3
- [ ] CPD [ ] WRTD
- [ ] WRTC

Ignore Hex-File Configuration [ ]

Enable LVP [✓]

Erase Chip [✓]
Flash Memory [✓]
EEPROM [ ]
ID-Location [✓]
Configuration [✓]

Erase PIC
Erase EEPROM
Erase ID
Program PIC
Verify
Read
Delta Update

New Software

Amadeus USB-Programmer
(c) 2004 by Bernhard Michelis
User Interface SW Version: 28 Dec 2004
Firmware Version: 1.0

[✓] Update Programmer   [✓] Fast Verify   [ ] No Verify

Amadeus USB-Prommer SW
(c) 2004 by Bernhard Michelis

Illustration 4: PIC18 interface is also for firmware udpates

Select the firmware file ***Amadeus.hex*** and check ***Update Programmer***. Click on **Identify PIC**. If the programming interface is working correctly, the software realizes the PIC18F242 (or 252). When this is not working, check your soldering.

Now click **Program PIC**. It will take a little time, then the programmer should display SUCCESS.

If the controller was realized, but you got a FAIL, don't be sad and try again. This can happen with some USB interfaces and has no effect on the later stability of the programmer. If it fails again have a look at the log window. In case, only Flash Memory was FAIL, click **Delta Update**, once, twice,.... Otherwise remove the **Erase Chip** checkmark and unckeck all successful steps. Try again **Program PIC**. *If **Erase Chip** is unckecked, it is possible to program each part of the firmware seperately.*

The problem with this firmware update is, that the FT232BM has no flow control in *BitBang* mode. Sending data always works fine, but verifying might be a problem. If it won't work at all, check **No Verify**. Now the verification will be skipped. When the hardware is working correctly, this is ok.

The last step is to disconnect the programmer form the USB port and close Amadeus.

Drumroll.....

Connect the hardware again and start Amadeus. This time the firmware must be read correctly.

That's it! Now have fun and enjoy fast software updates. No more waiting....

2.1 Supported Devices

µC	*Additonal Information*
ATtiny11	High-voltage serial programming
ATtiny12	High-voltage serial programming only - Tested
ATmega8	xxx
ATmega16	Not Tested
ATmega32	xxx
ATmega64	Tested
ATmega128	Not Tested
PIC12F629	Not Tested
PIC12F675	Tested
PIC16F630	Not Tested
PIC16F676	Not Tested
PIC18F242	Tested
PIC18F248	Not Tested
PIC18F252	Tested
PIC18F258	Not Tested
PIC18F442	Test
PIC18F448	Tested
PIC18F452	Tested
PIC18F458	Not Tested

Table 1: Supported MCUs

3 Amadeus Software Description

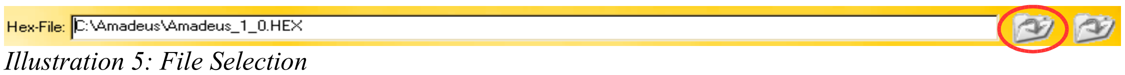
3.1 Basics

First of all, select the desired microcontroller family. There's a dedicated interface for each type of microcontroller. Despite of differences, there are some constant elements, identical for all MCU types supported by Amadeus. To shorten this documentation, they are summariesed in the following paragraphs.

Slight differences might exist between several implementations.

3.1.1 File Selection

Select the hex-file to be programmed. Therefore click the folder symbol right beside the hex-filename edit field and select a file. Files must be in *Intel-Hex* format.



Hex-File: C:\Amadeus\Amadeus_1_0.HEX

Illustration 5: File Selection

For *Microchip* devices, all information (program, EEPROM, ID and configuration) can be contained in one hex-file.

For *Atmel* devices, normally the hex-file only contains the program. Therefore there are two folder buttons on the right side. Click the left one to select the program file and the right one, if EEPROM values have to be programmed. You can deselect the EEPROM file by canceling the EEPROM-file selection.

3.1.2 Identify-Button

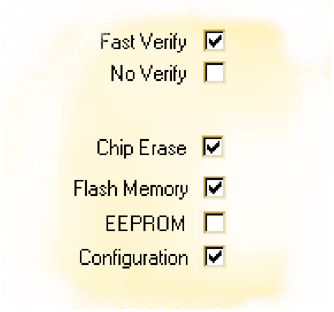
Click to query MCU information. When the MCU is supported by the selected interface and the connection for in-circuit serial programming is correct, the MCU's name will be displayed.

3.1.3 Get Configuration From-Button

Reads the current configuration settings form the connected MCU and displays them.

3.1.4 Program, Read, Verify

Programs, reads or verifies the MCU. Depending on the MCU, there are some settings available, to give more detailed control.



Fast Verify [✓]
No Verify [ ]
Chip Erase [✓]
Flash Memory [✓]
EEPROM [ ]
Configuration [✓]

Illustration 6: Action Selection

Fast Verify speeds up verification by skipping all unprogrammed areas. To test, if unprogrammed areas are really empty (0xFF), disable **Fast Verify**. This is also valid for **Read!** In case of read, only memory areas are read, that are defined by the selected hex-file.

No Verify suppresses all read operations. In this case the verification result is always *successful*.

Erase Chip, **Software**, **Flash Memory**, **EEPROM**, **ID-Location** and **Configuration** enable or disable the conrresponding functionallity for programming, verification or reading.

Later verification and reading will fail, if code protection is activated!

Ignore Hex-File Configuration (PIC only) uses the displayed configuration settings instead of the settings stored in the hex-file.

Some additonal functionallity might be available for some MCUs.

3.1.5 RAM/EEPROM Read Reset

Is a little debugging feature. As far as possible, it reads RAM (PIC18 only) or EEPROM content via the programming interface and displays them as hex dump. In all cases a reset accompanies this request.

3.1.6 Connection and Power Switching

Because the programming interface goes into high-impedance mode (see Electrical Characteristics on page 18) after programming, in most cases there's no need to disconnect the programmer from your target system after programming. That rises the wish to reset the whole target system after programming, because some external components might got upset during programming.

Therefore the programmer can switch an external power supply. See Electrical Characteristics on page 18 for details. In case that you need to switch voltages above about 20V you should design your own power switch and use the internal Power Switch (N4 and N5) or EXTVCC signal to control it. EXTVCC is somewhere between 4.6V and 5.2V during programming and can be used as power supply during programming for a stand alone programming interface (MCU, oscillator only).

Caution: Don't use the relay for high voltage switching.

Power switching can be activated on the MCU selection view. You have the choice between inactive, manual and two versions for some predefined times.

Power switching is evoked, if active, every time the programmer accesses the target MCU. You can choose *power on* or *off* after every MCU access.

3.2 Debugging Features

Amadeus is *not* an in-circuit-debugger. However it offers at least one simple debugging feature.

The EEPROM-Read-Rest give you the opportunity to display data as hex-dump. So it is possible to read back data from you MCU, even if no other communication way is implemented. Just write the desired data to the EEPROM. This can be done by a macro, that you can develop and test inside the simulation environment of MP-Lab or AVR-Studio.

Feel lucky, if you are using a PIC18. Amadeus can read PIC18 RAM directly, so no EEPROM write macros are necessary for debugging.

In all cases, a reset accompanies the reading.

3.2.1 How Do I Use this feature

Imagine, you are trying to program the UART interface and nothing will work. So maybe you will ask yourself sometime, did the byte arrive? Okay, you can switch an LED on or off, or do something else, when the RX-IRQ occures. But you never know, if the byte was received correctly. Think of the mistakes, you could make. Wrong baudrat, wrong parity...

In such cases, just copy the received byte to EEPROM/RAM and *loop forever*. Now use the Read-Reset to see the byte. In all cases the complete EEPROM/RAM will be hex-dumped.

It might happen, that the MCU starts executing some instructions before the memory can be read out. Insert some NOPs at the beginning.

The other way to debug your software in-ciruit step by step is to:

- Use endless loops instead of breakpoints.
- Copy all interested information to EEPROM/RAM before the endless loop.
- Perfom the Read-Reset.
- Set the copy macros and the endless loop to the next *breakpoint* and reassable, reflash and restart your code.
- And so on...

Special Function Registers can also be copied.

3.2.2 Suggestion

TIP: If the MCU choice is your's and saving some hours or days is worth investing additional 4€, take a PIC18. It's RAM-Read-Reset makes in-circuit debugging much easier than using the EEPROM variation.

- On every read, all veriables are displayed.
- When you fill the complete RAM with 0xFF after reset, you can see, if there's a function wildly spreading data all over the memory.
- You don't need to copied interesting data to the EEPROM, only SFRs.

3.3 ATtiny 11/12

When programming one of these devices, the programmer works in high-voltage mode, only. The low-voltage serial programming for ATtiny12 is not supported.

Both MCUs offer 1kByte program memory (512 instructions). Additionally, the ATtiny12 offers 64Byte EEPROM.

3.3.1 How To Connect

For electrical details see chapter “Electrical Characteristics“ on page 18.

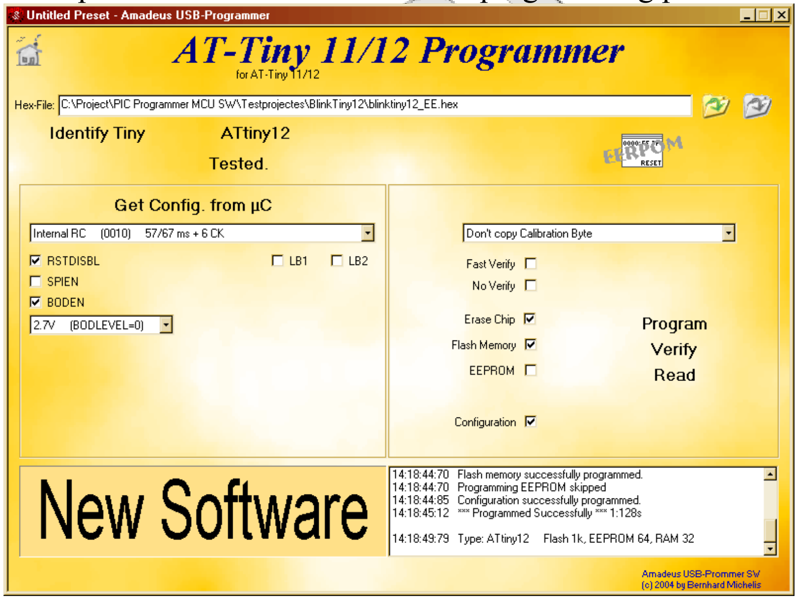
6 lines are required for programming. 🟨

Amadeus side	***MCU side***	***comment***
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit.
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming.
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables.
MCLR/RST	PB5	Reset (about 12V during programming)
PGM/PDI	PB0	Serial Data Input
PGD/PDO	PB1	Serial Instruction Input
PGC/SCK	PB2	Serial Data Output
SCL	PB3	Serial Clock Input

Table 2: ATtiny programming adapter definition

3.3.2 Programming

Have a look at the chapter “Basics” for details on the programming procedure.



Hex-File: C:\Project\PIC Programmer MCU SW\Testprojectes\BlinkTiny12\blinktiny12_EE.hex

ATtiny12

Tested.

Internal RC (0010) 57/67 ms + 6 CK

- [x] RSTDISBL
- [ ] SPIEN
- [x] BODEN
- [ ] LB1
- [ ] LB2

2.7V (BODLEVEL=0)

Don't copy Calibration Byte

- [ ] Fast Verify
- [ ] No Verify
- [x] Erase Chip
- [x] Flash Memory
- [ ] EEPROM
- [x] Configuration

```
14:18:44:70  Flash memory successfully programmed.
14:18:44:70  Programming EEPROM skipped
14:18:44:85  Configuration successfully programmed.
14:18:45:12  *** Programmed Successfully *** 1:128s

14:18:49:79  Type: ATtiny12    Flash 1k, EEPROM 64, RAM 32
```

Illustration 7: Tiny 11/12 Programming Interface

3.3.3 Additionlly Features

Atmel stores an oscillator calibration byte inside the MCU, that cannot be reached by software. Therefore Amadeus gives you the opportunity to copy this calibration byte either to the end of flash memory (high byte) or to the beginning or end of EEPROM (ATtiny12 only).

3.4 ATmega 8/16/32/64/128

See chapter “Supported Devices“ on page 5 for a list of supported devices. On the contrary to other MCUs, ATmegas need a valid clock to be programmed. The internal 1MHz/8MHz oscillator is enabled by default. See documentaion of your MCU.

3.4.1 How To Connect

For electrical details see chapter “Electrical Characteristics“ on page 18.

5 lines are required for programming.

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit.
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming.
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables.
MCLR/RST	RESET	Only low voltage programming.
PGM/PDI	PDI/PE0 (ATmega64) / MOSI	MOSI (different ATmegas seem to use different pins for programming. Have a look at their datasheets, chapter “Memory Programming/SPI Serial Programming Pin Mapping”)
PGD/PDO	PDO/PE1 (ATmega64) / MISO	MISO (different ATmegas seem to use different pins for programming. Have a look at their datasheets, chapter “Memory Programming/SPI Serial Programming Pin Mapping”)
PGC/SCK	SCK/PB1	SCK (different ATmegas seem to use different pins for programming. Have a look at their datasheets, chapter “Memory Programming/SPI Serial Programming Pin Mapping”)

Table 3: ATmega programming adapter definition

Be aware, that different MCU types of the ATmega series use different pins for programming.

3.4.2 Programming

Only additional information on programming are given in the paragraph. Have a look at the chapter “Basics” on page 6 for the basic programming procedure.

Before you start programming a device you should set up the programming speed correctly. Currently you can choose from 6 speed options.

Option	*Description*
10000kHz Normal (2ms)	Select this option if you clock your MCU below 1MHz and internal oscillator does not offer internal 8MHz clock. PT64: 20.6s
10000kHz Fast (2ms)	Select this option if you clock your MCU below 1MHz and the internal oscillator offers 8MHz. PT64: 5.9s

Option	*Description*
1MHz Normal (2µs)	Select this, when MCU is clocked at 1MHz or faster. Use only when the MCU's oscillator does not offer internal 8MHz clock. PT64: 19.8s
1MHz Fast (2µs)	Select this, when MCU is clocked at 1MHz or faster. (Default) PT64: 4.3s
6,7MHz Fast (300ns)	Select, when MCU is running above 6,7MHz. I.e. the internal oscillator@8MHz or a crystal, or ... PT64: 4.1s
10-12MHz and >15MHz (200ns)	High speed programming mode. Requires 10-12MHz or at least 15MHz. The range from 12 to 15 is not specified. PT64: 3.8s

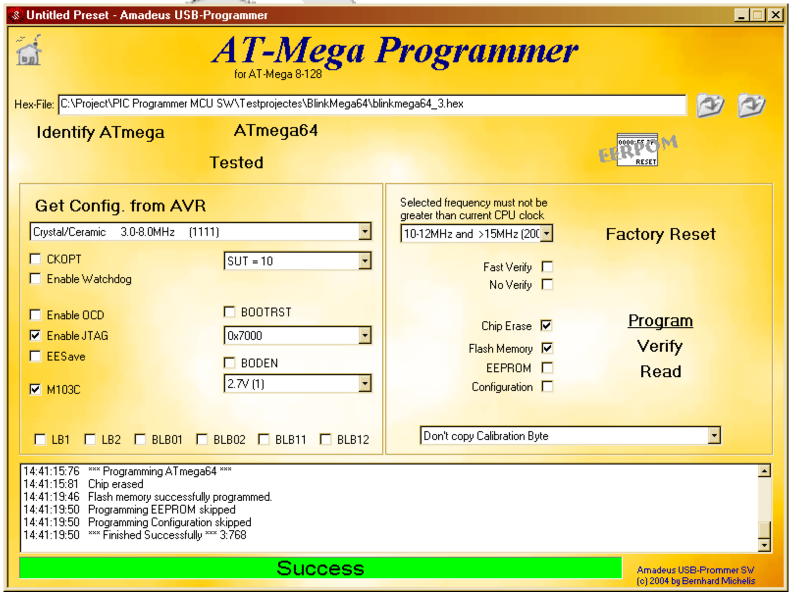
Table 4: Programming Speeds

PT64 indicates the time that is needed to program the complete Flash memory of an ATmega64 with 0x00 and verify it. An ATmega128 will need about twice as long. Smaller Flash memories will be programmed faster.

During development it is recommended to select *Fast Verify*.

If your program is smaller than the MCU's Flash memory, empty areas will be skipped. This speeds up programming further.

If you don't need to update the EEPROM select the *EESave* fuse and disable *EEPROM* for programming. Then the EEPROM will not be erased by *Erase Chip* and reprogrammed. And programming is faster.



Hex-File: C:\Project\PIC Programmer MCU SW\Testprojectes\BlinkMega64\blinkmega64_3.hex

ATmega64
Tested

Crystal/Ceramic 3.0-8.0MHz (1111)
☐ CKOPT
☐ Enable Watchdog
SUT = 10
☐ Enable OCD
☑ Enable JTAG
☐ EESave
☑ M103C
☐ BOOTRST
0x7000
☐ BODEN
2.7V (1)
☐ LB1 ☐ LB2 ☐ BLB01 ☐ BLB02 ☐ BLB11 ☐ BLB12

Selected frequency must not be greater than current CPU clock
10-12MHz and >15MHz (200
Fast Verify ☐
No Verify ☐
Chip Erase ☑
Flash Memory ☑
EEPROM ☐
Configuration ☐
Don't copy Calibration Byte

```
14:41:15:76   *** Programming ATmega64 ***
14:41:15:81   Chip erased
14:41:19:46   Flash memory successfully programmed.
14:41:19:50   Programming EEPROM skipped
14:41:19:50   Programming Configuration skipped
14:41:19:50   *** Finished Successfully *** 3:768
```

Success

Illustration 8: ATmega Programming Interface

Warning

There are some ATmega MCUs, with a Reset Disable (RSTDISBL) option. Be aware, that it is theoretically possible to write this fuse *(for instance by accident, if you select the wrong programming speed)*. After that, serial programming is no longer possible. The software does not explicitly offer the possibility to program this fuse, but if something goes wrong during programming, only parallel programmer is able to reanimate the MCU.

Or for some devices, with a little luck, if you use the Power Switching feature set to 1ms. In this case it might be possible, that the MCU is still in power-on-reset state and accepts programming commands.

3.4.3 Factory Reset / Reanimate Function

A fault during programming can be, that the wrong clock source was selected. In such a case, the programmer can no longer communication with the MCU.

This can be handle with the help of the Factory Reset function. To reanimate the MCU, a special clock frequency might be necessary. Therefore connect the programmers SCL pin to the MCU's XTAL1 pin and click **Factory Reset**. When Factory Reset was successful, disconnect the external clock (SCL).

3.4.4 Oscillator Calibration Byte

When using an internal oscillator, you might need a calibration value for the oscillator. This calibration value cannot be accessed from a program. Therefore it is necessary to copy the value into the Flash memory or EEPROM.

The programmer itself does *not* offer a calibration routine, but if you otherwise managed to calibate your MCU, and the calibration value is stored in the last Flash memory cell (high byte), it is possible to preserve this value during flash programming.

Hint: In case of **Preserve Calibration Byte** never erase the chip without reprogramming the Flash memory, or you will lose your manual calibration. This is also true for the Factory Reset.

3.4.5 EEPROM Programming

For the ATmega family, there's a *specail* way Amadeus handles EEPROM programming. If the **EESave** fuse is set, or you don't select **Erase Chip**, only EEPROM values given in the hex-file are updated. If there's a rage of EEPROM addresses missing in the hex-file, those EEPROM cells will not be updated (no 0xFF will be written there). Keep this in mind, if you store your manual calibration value inside the EEPROM.

3.5 PIC12F 629/675 / PIC16F 630/676

The four MCUs have identical programming features. They contain 1kWord (14bit) of flash memory (0x000 to 0x3ff) and 128Bytes of EEPROM. Further more an ID-location of 4x14bit words and a set of configuration bits.

Internal clock is 4MHz (1MIPS). Externally, up to 20MHz (5MIPS) are possible.

At flash address 0x3ff (hex-file 0x7fe), Microchip stores the Oscillator Calibration Byte in form of a RETLW xx instruction. So the user's software must end at address 0x3fe. The programming software restores the original Calibration Byte after every programming/flash erasing.

Another important calibration value is the Band Gap Calibration. It occupies two bits in the configuration word. They are also restored after every programming/flash erasing.

Amadeus programs the application software area (flash memory), EEPROM, ID-location and configuration, when they are contained in the hex file.

Hex-File-Address	***Contents***
0x0000 to 0x07fd	Software for MCU (14bit/word)
0x07fe to 0x07ff (will be ignored)	Oscillator Calibration Byte
0x4000 to 0x4007	4-14bit words programmed to ID-location
0x400e to 0x400f	Configuration (bits 0-8, BG bits 12/13 will be ignored)
0x4200 to 0x42ff	EEPROM 128Bytes (every second bytes will be programmed, every other byte in the hex-file contains 0x00). For details, see MP-Lab help.

Table 5: PIC12/16 and compatible hex-file definition

Besides the contents of the hex-file, it is possible to edit the configuration-bits. When **Ignore Hex-File Configuration** is marked, the hex-file's configuration is replaced.

3.5.1 How To Connect

For electrical details see chapter “Electrical Characteristics“ on page 18.

4 lines (+ power supply switching) are required for ICSP.

Amadeus side	***MCU side***	***comment***
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit.
EXTVCC !!!	VCC	**For stand alone programming only!!!** Do never connect EXTVCC for in-circuit programming.
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables.
MCLR/RST	MCLR	High Voltage programming.
PGD/PDO	ICSPDAT	Serial Data
PGC/SCK	ICSPCLK	Serial Clock

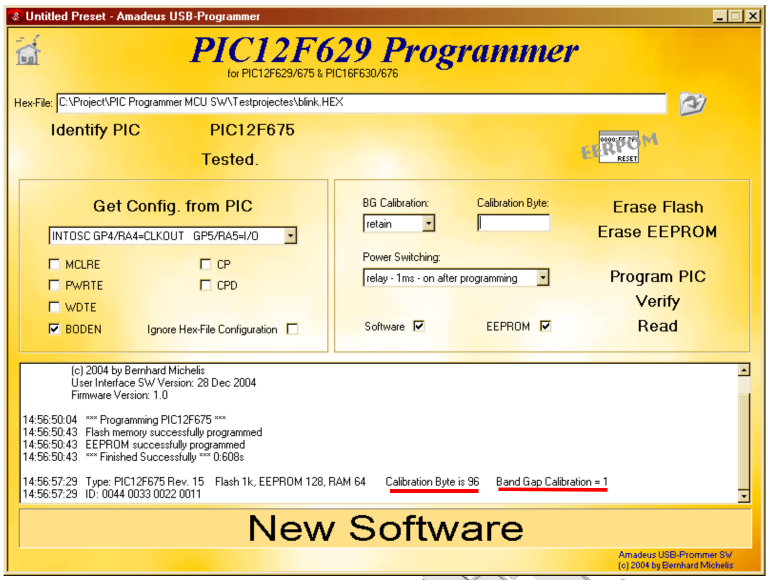
Table 6: PIC12F/16F ICSP adapter definition

The programming specification says, that it is necessary to switch power (MCU supply voltage) on, after VPP has been applied. Practical tests from my side showed (at least for my PIC12F675), that power switching was not necessary.

However, the programmer can handle this situation, if required. For details on Power Switching see page 7.

3.5.2 Programming

Only additional information on programming are given in the paragraph. Have a look at the chapter “Basics” on page 6 for the basic programming procedure.



Hex-File: C:\Project\PIC Programmer MCU SW\Testprojectes\blink.HEX

PIC12F675

Tested.

INTOSC GP4/RA4=CLKOUT GP5/RA5=I/O

- ☐ MCLRE
- ☐ CP
- ☐ PWRTE
- ☐ CPD
- ☐ WDTE
- ☑ BODEN
- Ignore Hex-File Configuration ☐

BG Calibration: retain

Calibration Byte:

Power Switching: relay - 1ms - on after programming

Software ☑ EEPROM ☑

```
        (c) 2004 by Bernhard Michelis
        User Interface SW Version: 28 Dec 2004
        Firmware Version: 1.0

14:56:50:04   *** Programming PIC12F675 ***
14:56:50:43   Flash memory successfully programmed
14:56:50:43   EEPROM successfully programmed
14:56:50:43   *** Finished Successfully *** 0:608s

14:56:57:29   Type: PIC12F675 Rev. 15    Flash 1k, EEPROM 128, RAM 64    Calibration Byte is 96    Band Gap Calibration = 1
14:56:57:29   ID: 0044 0033 0022 0011
```

Illustration 9: PIC12F629 Programming Interface

Click on **Identify PIC**. The software tries to identify the target MCU. When the connection is working and the target device is powered up, there should be a message, displaying the MCU's name. Furthermore the *Oscillator Calibration Byte* (0-255) and the *BG Calibration* (0-3) will be displayed. Although they are restored automatically, it is suggested to write them down somewhere, to restore them manually, if something went wrong.

When the checkbox **Software** is checked, Flash memory, ID-location and configuration will be programmed. When **EEPROM** is checked, the EEPROM area will be programmed.

Erase Flash erases the program memory, ID-location and configuration and **Erase EEPROM** erases the EEPROM.

3.5.3 Calibration Recovery

If, for any reason, one of the calibration values vanished, it is possible to restore them manually.

Therefore you need the two values, you got, when you first clicked **Identify PIC**. Enter the value in the field **Calibration-Byte** and select the correct **BG-Calibration** value. Now click on **Erase Flash**. That's it. When **BG-Calibration** shows **retain**, **Erase Flash** doesn't touch any of the saved values.

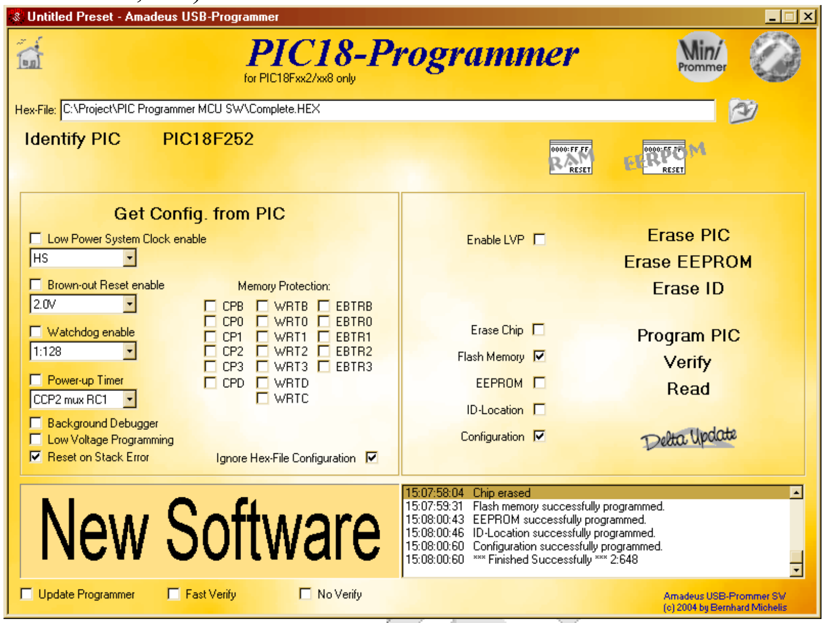
3.5.4 EEPROM Read Reset

See “RAM/EEPROM Read Reset“ on page 7 and “Debugging Features“ on page 7.

3.6 PIC18F 242/248/252/258/442/448/452/458

At the moment, only these eight PIC18F MCUs are supported. They contain 16kBytes or 32kBytes Flash memory, 256Byte EEPROM, an 8Byte ID-Location and 14 configuration bytes.

There's no internal clock. Externally, up to 40MHz (10MIPS) are possible or in HS-PLL mode up to 10MHz (10MIPS!!!, too).


Untitled Preset - Amadeus USB-Programmer

PIC18-Programmer
for PIC18Fxx2/xx8 only

Hex-File: C:\Project\PIC Programmer MCU SW\Complete.HEX

Identify PIC PIC18F252

Get Config. from PIC

☐ Low Power System Clock enable
HS
☐ Brown-out Reset enable
2.0V
☐ Watchdog enable
1:128
☐ Power-up Timer
CCP2 mux RC1
☐ Background Debugger
☐ Low Voltage Programming
☑ Reset on Stack Error

Memory Protection:
☐ CPB ☐ WRTB ☐ EBTRB
☐ CP0 ☐ WRT0 ☐ EBTR0
☐ CP1 ☐ WRT1 ☐ EBTR1
☐ CP2 ☐ WRT2 ☐ EBTR2
☐ CP3 ☐ WRT3 ☐ EBTR3
☐ CPD ☐ WRTD
☐ WRTC

Ignore Hex-File Configuration ☑

Enable LVP ☐
Erase Chip ☐
Flash Memory ☑
EEPROM ☐
ID-Location ☐
Configuration ☑

```
15:07:58:04  Chip erased
15:07:59:31  Flash memory successfully programmed.
15:08:00:43  EEPROM successfully programmed.
15:08:00:46  ID-Location successfully programmed.
15:08:00:60  Configuration successfully programmed.
15:08:00:60  *** Finished Successfully *** 2:648
```

☐ Update Programmer ☐ Fast Verify ☐ No Verify

Illustration 10: PIC18F Programming Interface

There are no calibration values, that have to be preserved.

3.6.1 How To Connect

For electrical details see chapter “Electrical Characteristics“ on page 18.

4/5 lines are required for ICSP. 🟨

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit.
EXTVCC !!!	VCC	**For stand alone programming only!!!** Do never connect EXTVCC for in-circuit programming.
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables.
MCLR/RST	MCLR	High Voltage programming. Enable LVP must be unchecked.
PGD/PDO	PGD	Serial Data
PGC/SCK	PDC	Serial Clock
PGM/PDI	PGM	Only required for Low Voltage Programming. Enable LVP must be checked.

3.6.2 Programming

This dialog also contains all features, necessary, to update the programmers own MCU. For details see chapter “Putting the programmer into operation” on page 3.

The PIC18Fxxx family offers a wide range of programming possibilities, that offer very fast programming. During development/debugging programming time can be reduced, when you keep the following hints in mind.

- Always select **Fast Verify**.
- The currently supported devices offer multi-panel flashing. So it is faster when different parts of the SW start at *org 0x0000* and *org 0x2000* and for 32kBytes devices in addition *org 0x4000* and *org 0x6000* instead of one big program chunk.
- PIC18F devices don't need to erase the complete chip, before they can reprogram the application software. So remove **Erase Chip** during development and disable all areas (EEPROM, ID-location and configuration) that don't need an update. Empty areas will be skipped during programming.

Action	*Duration*
Programming and verifying 16kBytes Flash (full)	2.5s
Programming and verifying 32kBytes Flash (full)	2.7s
EEPROM programming and verifying 256Bytes (full)	1s

Table 7: PIC18Fxx2/xx8 programming durations

For super fast software updates during development (down to 200ms programming time) you can use the **Delta Update** function. It is only available for the program flash memory. It can update single flash segments, while others aren't touched. To prevent large parts of the program to be shifted in memory (takes time), it is strongly recommended to leave some space after a function block. Use the **org** directive to give ISR, program main loop, functions for this, functions for that fixed addresses. Otherwise it is slower then normal programming.

Important: Before using **Delta Update**, you must have programmed the MCU normally (at least for one time), because Amadeus does not compare new data in hex-file with the data in the MCU, but with the data form last programming. If you restart Amadeus, the data history is all 0xFF.

In case the programmer cannot find the PIC, check that **Enable LVP** is set correctly.

3.6.3 RAM/EEPROM Read Reset

See “RAM/EEPROM Read Reset“ on page 7 and “Debugging Features“ on page 7.

4 Amadeus Programmer Hardware

4.1 Electrical Characteristics

4.1.1 Serial Programming Interface

At the moment, the complete hardware is designed for 5V programming. Every output can drive about 20mA.

If you plan to program MCUs working with less than 5V you might follow the following idea (***on your own risk***). Nearly every pin of nowadays MCUs have internal ESD protection diodes, that pull down the voltage to ca 0.6V above VCC. VCC might rise a little, too, depending on the load of other components on VCC. With a resistor in the programming cable this might work for you (***On your own risk!!!***). Also keep in mind, that some timings might shift depending on VCC, so that programming might be impossible.

I, personally, tried an ATmega64 @3V with the upper mentioned method. Worked so far.

Have a look on the following tables. They show, where to find every programmer pin.

Pin on Programmer Hardware	***Description***
Pin 1: SCL	High-impedance during stand-by. Can drive up to 20mA during programming.
Pin 2: GND	Ground
Pin 3: SDA	High-impedance during stand-by. Can drive up to 20mA during programming.

Table 8: N2

Pin on Programmer Hardware	***Description***
Pin 1: PGM/PDI	High-impedance during stand-by. Can drive up to 20mA during programming.
Pin 2: GND	Ground
Pin 3: PGD/PDO	High-impedance during stand-by. Can drive up to 20mA during programming.
Pin 4: GND	Ground
Pin 5: PGC/SCK	High-impedance during stand-by. Can drive up to 20mA during programming.
Pin 6: MCLR/RST	High-impedance (0.1mA@5V/ca 55kΩ) during stand-by. See 74LS05 datasheet for details on open-collector output. Keep in mind, that this pin rises to ca 12V during high-voltage programming when designing your target device. If the MCU pin has reset functionallity, use a 100-200kΩ pull-up resistor. If the *reset* is deactivated, the connected components must not draw more than 1mA form this line. Otherwise the 12V breaks down.
Pin 7: EXTVCC	Switched power-supply for standalone programming. Do never connect to any other power supply. Do not draw more than 100mA.

Table 9: N3

Pin on Programmer Hardware	*Description*
N4(Pin1) to N5(Pin1)	Relay (50mΩ; ca 500ns). Use only for voltages below 20V
N4(Pin2) to N5(Pin2)	Relay (50mΩ; ca 500ns). Use only for voltages below 20V

Table 10: N4 and N5

4.2 Schematics

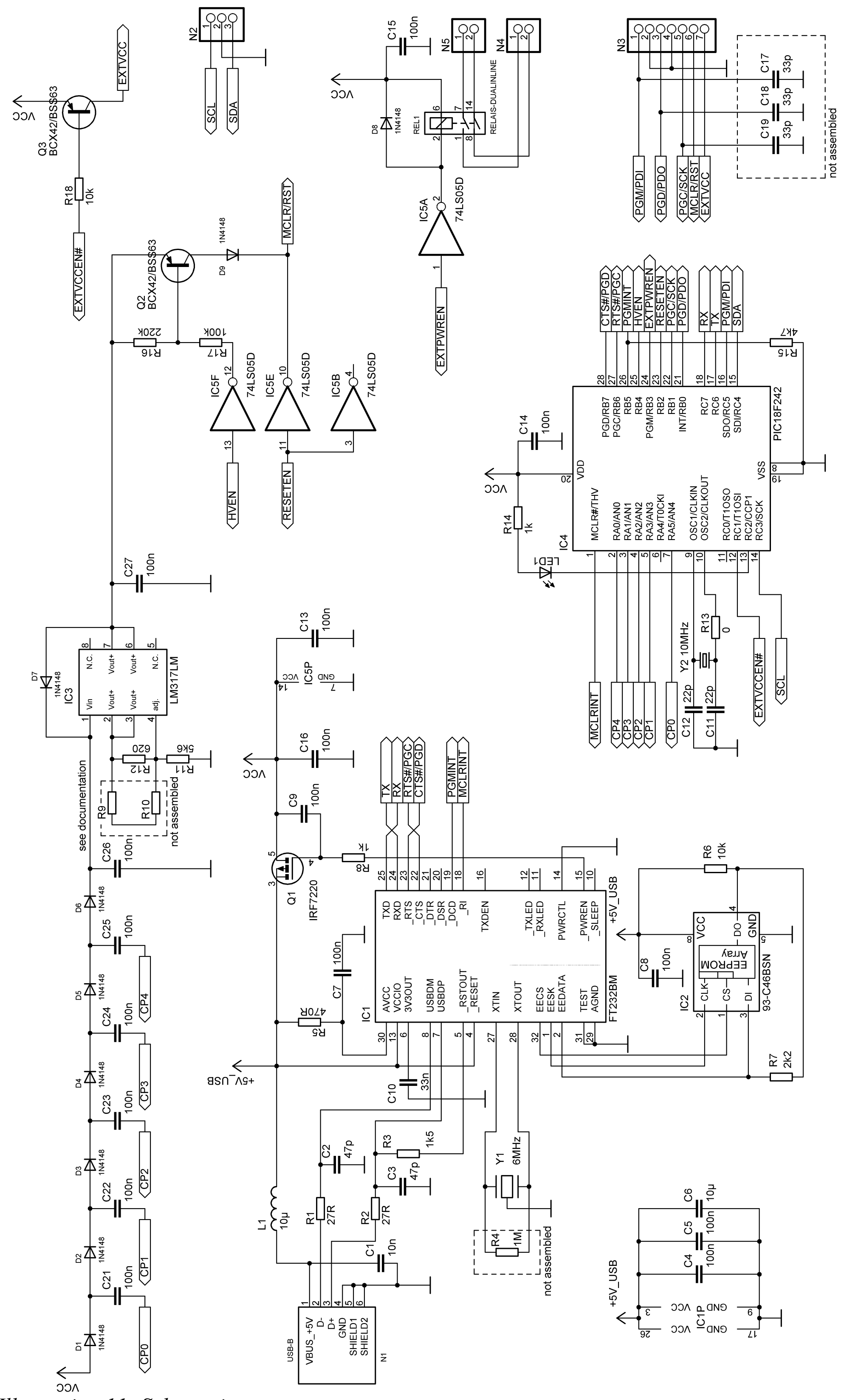


Illustration 11: Schamatics

4.3 Partlist

For people living in Germany: Currently (Dec. 2004) all parts are available at *www.reichelt.de*.

Position	*Value*	*Package*	*Comment/Reichelt ordercode*
C1	10n	C0805	
C10	33n	C0805	
C11	22p	C0805	
C12	22p	C0805	
C13	100n	C1206 or C0805	
C14	100n	C0805	
C15	100n	C0805	
C16	100n	C0805	
C17	Not Assembled	C0805	
C18	Not Assembled	C0805	
C19	Not Assembled	C0805	
C2	47p	C0805	
C21	100n	C0805	
C22	100n	C0805	
C23	100n	C0805	
C24	100n	C0805	
C25	100n	C0805	
C26	100n	C0805	
C27	100n	C0805	
C3	47p	C0805	
C4	100n	C0805	
C5	100n	C0805	
C6	10µ	C5,8X3,2MM	
C7	100n	C0805	
C8	100n	C0805	
C9	100n	C0805	
D1	1N4148 SMD	MINI-MELF	1N 4148 SMD
D2	1N4148 SMD	MINI-MELF	
D3	1N4148 SMD	MINI-MELF	
D4	1N4148 SMD	MINI-MELF	
D5	1N4148 SMD	MINI-MELF	
D6	1N4148 SMD	MINI-MELF	
D7	1N4148 SMD	MINI-MELF	
D8	1N4148 SMD	MINI-MELF	
D9	1N4148 SMD	MINI-MELF	
IC1	FT232BM	LQFP32	FTDI FT232BM
IC2	93-C46BSN	SO-08	
IC3	LM317LM	SO-08	LM317LM
IC4	PIC18F242 or PIC18F252	SO-28W	PIC18F242 (or PIC18F252 might be interesting for Updates) *Important: If you are recycling an old PIC, Low Voltage Programming must be enabled.*

Position	*Value*	*Package*	*Comment/Reichelt ordercode*
IC5	74LS05D	SO-14	
L1	10µ	L0805	JCI 2012 10µ
LED1	LED5MM	LED5MM	
N1	USB_SOCKET	USB-BW Assmann	USB BW for print assembly
N2	1X03	pinhead	
N3	1X07	pinhead	
N4	1X02	pinhead	2,54mm or 0.1"
N5	1X02	pinhead	
Q1	IRF7220	SO-08	
Q2	BCX42/BSS63	SOT-23	
Q3	BCX42/BSS63	SOT-23	
R1	27R	R1206	
R10	Not Assembled	R1206	You can assembled other resistors (R9+R10||R12) to adjust the accuracy of VPP (12V)
R11	5k6	R1206	
R12	620R	R1206	
R13	0R	R1206	
R14	1k	R1206	
R15	4k7	R1206	
R16	220k	R1206	
R17	100k	R1206	
R18	10k	R1206	
R2	27R	R1206	
R3	1k5	R1206	
R4	Not Assembled	R1206	
R5	470R	R1206	
R6	10k	R1206	
R7	2k2	R1206	
R8	1k	R1206	
R9	Not Assembled	R1206	You can assembled other resistors (R9+R10||R12) to adjust the accuracy of VPP (12V)
REL1	RELAIS-DUALINLINE	DIL-14-RELAIS	MEDER electronic "DIP 7221-L 5V" www.meder.com
Y1	6MHz	RESONATOR	CSTCC 6,00
Y2	10MHz	Crystal HC49U-/S or -HC18	10-HC18

Table 11: Partlist

4.4 Assembly

Here is the assembly instruction for the programmer hardware. As you can see, it's only a sigle side PCB.

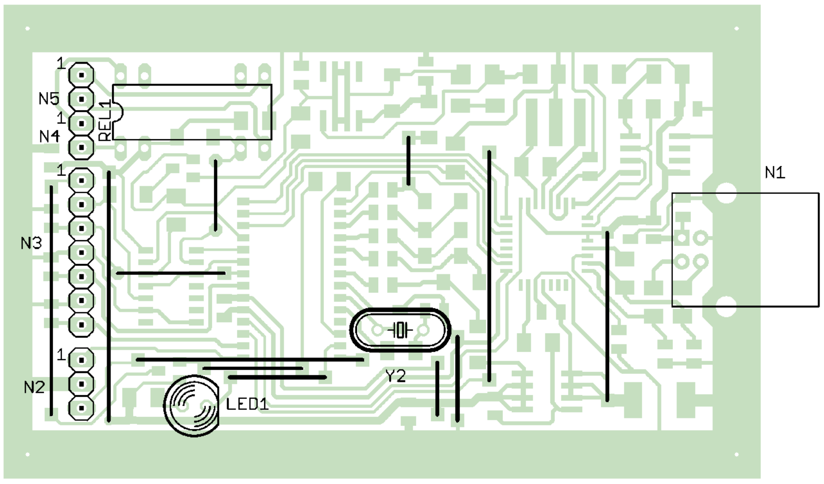


Illustration 12: PCB top view

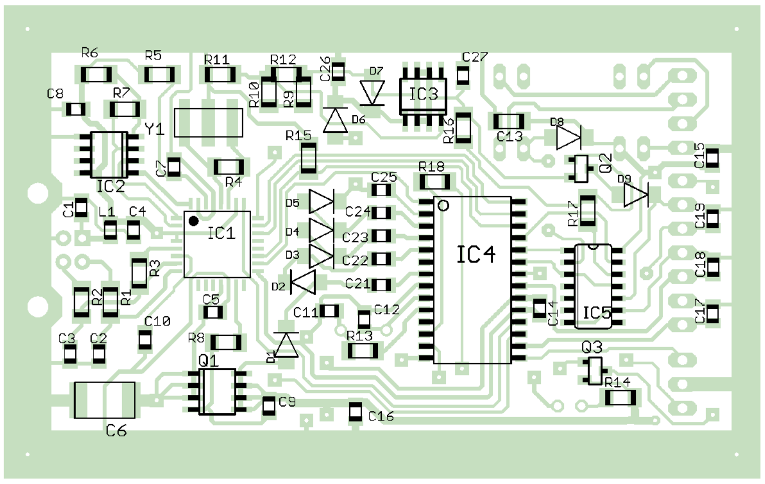
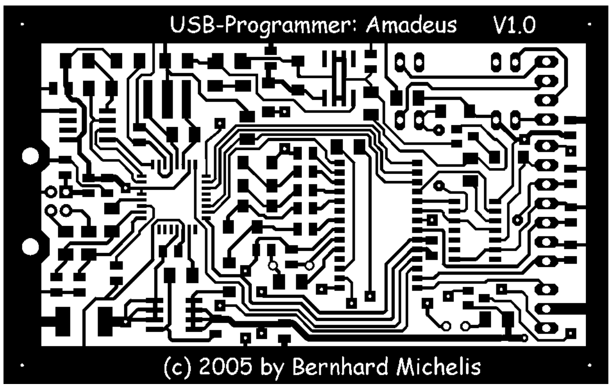


Illustration 13: PCB bottom view

4.5 Layout

The layout is available as a seperate file named “Amadeus_PCB.gif” with a higher resulution. After printing the layout must be 80mmx50mm of size.


USB-Programmer: Amadeus V1.0
(c) 2005 by Bernhard Michelis