Amadeus

µC based USB-programmer for Windows computers for selected PIC and AVR microcontrollers with UPAL support

DISCLAIMER

AMADEUS IS PROVIDED TO YOU "AS IS," WITHOUT WARRANTY. THERE IS NO WARRANTY FOR THE PROGRAM, ITS DOCUMENTATION NOR ITS HARDWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM, ITS DOCUMENTATION AND ITS HARDWARE IS WITH YOU. SHOULD ANY PART OF THIS PROJECT PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT THE AUTHOR WILL BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR SUSTAINED LOSSES BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAM), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

... or simplified:

“Whatever the future will bring, you are resonsible. No *Risk* no *Fun*!”

HAFTUNGSAUSSCHLUSS

AMADEUS SOFTWARE, DOKUMENTATION UND HARDWAREBESCHREIBUNG WERDEN SO WIE SIE SIND ("AS IS"), OHNE JEGLICHE GARANTIE UND HAFTUNG ZUR VERFÜGUNG GESTELLT. ALLE SICH AUS DER NUTZUNG ERGEBENDEN RISIKEN OBLIGEN DEM BENUTZER. DIESES SCHLIESST VERLETZUNGEN DER RECHTE DRITTER EIN.

UNTER KEINEN UMSTÄNDEN IST DER AUTOR DES PROJEKTES FÜR SCHÄDEN JEGLICHER ART HAFTBAR. DER NUTZER VERPFLICHTET SICH, ALLE SCHÄDEN AUF EIGENEN RECHNUNG ZU BEHEBEN.

DIESES GILT AUCH DANN, WENN DER AUTOR ÜBER MISSTÄNDE INFORMIERT WURDE.

...oder einfach:

"Was die Zukunft auch bringt, Du (der Nutzer) bist schuld. Kein Risiko, Kein Vergnügen!"

Table of Contents

1 General Information/Allgemeine Informationen

This project is based on a Microchip PIC18F242/252 microcontroller and Future Technology's FT232BM (FT232BL is the lead free version of the FT232MB, see http://www.ftdichip.com/FTProducts.htm#FT232BM for more information) USB-to-Serial converter.

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

Dieses Projekt basiert auf einem PIC18F242/252 Mikrocontroller und dem FT232BM USB-Seriell-Wandler von Future Technologys (FT232BL ist die Bleifrei Variante des FT232BM, siehe auch http://www.ftdichip.com/FTProducts.htm#FT232BM).

Derzeit gibt es vier Programmieralgorithmen für verschiedene PIC und AVR Mikrokontroller.

1.1 Supported Devices/Unterstützte Mikrocontroller

For a list of supported MCUs see Website[1].

Für eine Liste der Unterstützten Mikrocontroller bitte auf der Website schauen.

2 Putting the programmer into operation/Inbetriebnahme

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

Dieses Kapitel beschreibt, wie die Programmer Hardware in Betrieb genommen wird.

First of all, you need to order all necessary components. Find a list of all required components in the partlist on page 55.

Als erstes sollten alle notwendigen Komponenten bestellt werden. Eine Liste gibt es auf Seite 55.

Next you need to produce your own PCB. There's an external file, named “Amadeus_PCB.gif” (single side layout), with a higher resulution than the graphics in this document. The schematics can be found on page 51. 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.

Als nächstes gilt es, sich eine Leiterplatte zu fertigen. Die Begleitdatei “Amadeus_PCB.gif” enthält ein Layout in hoher Auflösung. Den Schaltplan findest Du auf auf Seite 51, sowie den Bestückungsplan auf Seite 56.

Then you have to assemble your PCB. See chapter “Assembly/Bestückungsplan“ on page 56 for details.

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

Nachdem die Hardware aufgebaut ist, muss die Schaltung installiert und programmiert werden.

Download two file from the internet. First of all the *D2XX driver package for the FT232BM (FT232BL) (www.ftdichip.com/Drivers/FT232-FT245Drivers.htm)* and a tool named *MProg (www.ftdichip.com/Resources/Utilities.htm#MProg)*. Both can be found on the website of Future Technology's.

1 Website: home.arcor.de/bernhard.michelis

Wir brauchen den D2XX-Treiber für den FT232BM (FT232BL) (http://www.ftdichip.com/Drivers/D2XX.htm), sowie ein Programmiertool mit dem Namen MProg.exe (www.ftdichip.com/Resources/Utilities.htm#MProg). Beides gibt es auf der Seite von Future Technologys.

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

Verbinde die Programmer Hardware mit Deinem Computer. Windows sollte ein neues USB-Gerät finden. Installiere nur den D2XX-Treiber und nicht den VCP-Treiber.

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

*Nach dem Neustart **MProg** starten. Lade die Begleitdatei "Amadeus.ept" in MProg und programmiere sie in die Schaltung.*

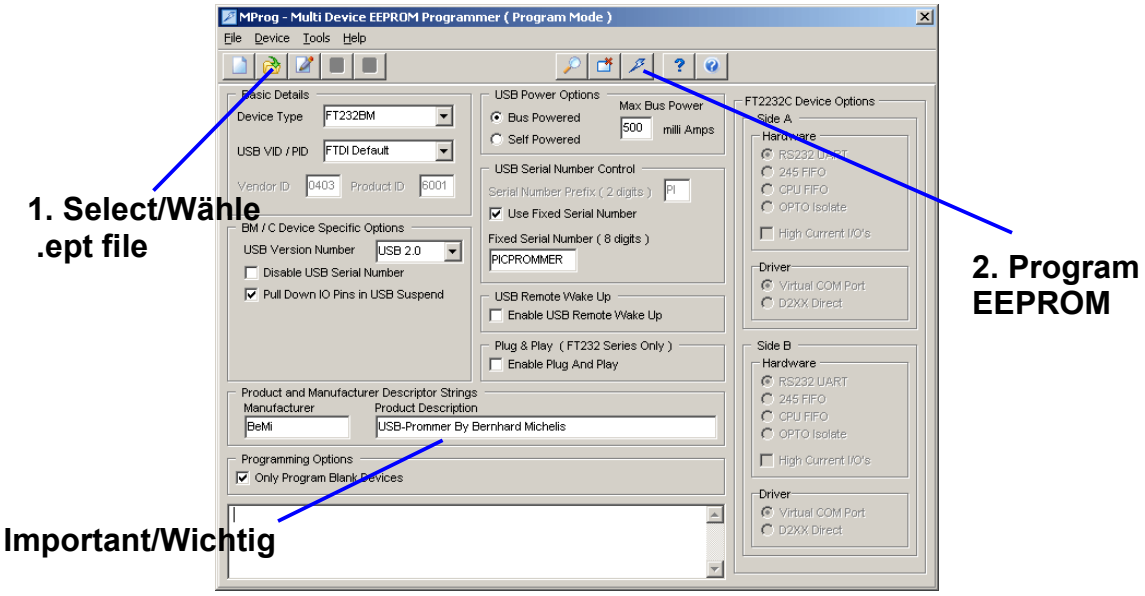

Illustration 1: MProg UI

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.

Den Programmer nach dem Programmieren vom Computer trennen und neu einstecken. Warscheinlich wird Windows jetzt ein neues USB-Device finden, da der USB-Chip umprogrammiert wurde.

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.

*Jetzt ist es Zeit **Amadeus**, die Benutzersoftware zur Hardware, zu starten. Sollte die folgende Meldung erscheinen, ist warscheinlich etwas bei der EEPROM Programmierung schief-gegangen. Darauf achten, dass das Feld **Product Despription** genauso, wie auf dem Bild, geschrieben ist.*

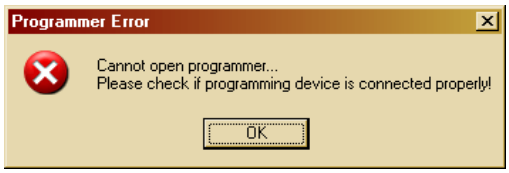

Illustration 2: Error Message if no hardware was found

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

Nachdem der USB-Controller nun korrekt programmiert ist, erscheint folgende Meldung, die uns sagt, dass die MCU im Programmer noch keine Firmware hat.

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.

Der letzte Schritt, bevor Du den Programmer einsetzen kannst, ist die Programmierung der Firmware.

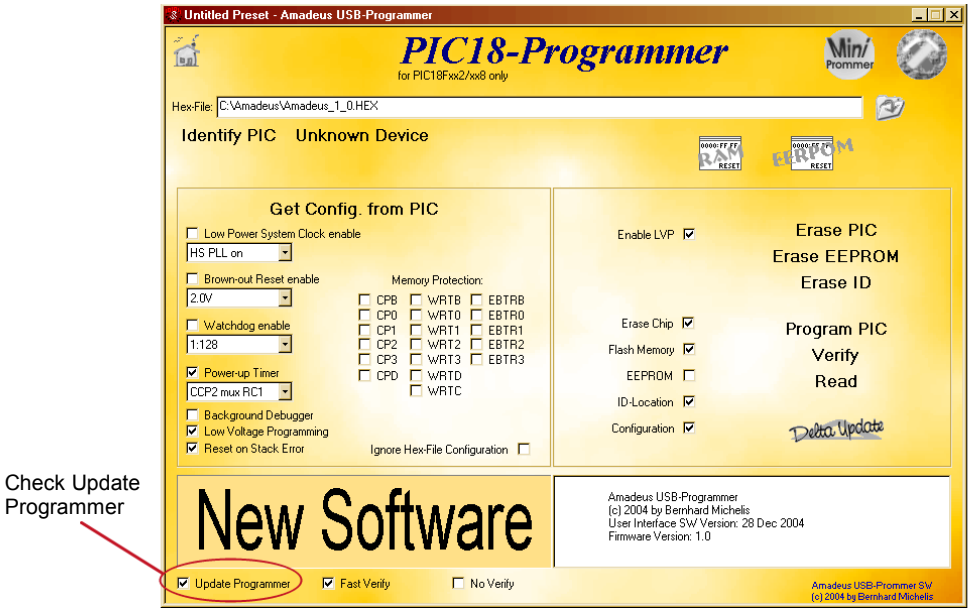

Illustration 4: PIC18 interface is also for firmware udpates

Select the firmware file ***Amadeus.hex*** and check ***Update Programmer***. Everything, except the path, should look like on the picture. 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.

*Wähle die Firmware-Datei **Amadeus.hex** und setze das Häkchen bei **Update Programmer**. Klick auf **Identify PIC**. Wenn alles in Ordnung ist, sollte ein PIC18F242 (oder 252) erkannt werden. Wenn nicht, bitte den Aufbau nochmal kontrollieren.*

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

*Mit einem Klick auf **Program PIC** wird das Software-Update gestartet. Es dauert eine Zeit und dann sollte **SUCCESS** erscheinen.*

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.*

*Wenn der Controller erkannt wurde, aber trotzdem ein FAIL erscheint, hat das nichts zu bedeuten. Das kann bei einigen USB-Schnittstellen der Fall sein, hat jedoch keinen Einfluß auf die Zuverlässigkeit im späteren Betrieb. Einfach nochmal **Program PIC** anklicken. Sollte es auch wiederholt scheitern, bitte einmal einen Blick auf das Log-Fenster werfen. Ist nur Flash-Programming fehlgeschlagen (fail), dann so oft auf **Delta-Update** klicken, bis PASS erscheint.*

*Anderenfalls alles deselektieren, was **successful** war. In diesem Fall unbedingt auch das **Erase Chip** Häkchen entfernen, und erneut **Program PIC**. Solange wiederholen, bis alles Pass ist.*

*Das Problem mit dem Firmware-Update resultiert aus der fehlenden Flußkontrolle beim FT232BM im BitBang Modus. Das Programmieren funktioniert eigenlich immer einwandfrei, nur das Vergleichen scheitert. Sollte also garnichts helfen, kann ein Häckchen bei **No Verify** gesetzt werden. Jetzt wird kein Vergleich nach dem Schreiben mehr durchgeführt.*

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

Nun die Programmer Hardware vom USB-Port entfernen.

Drumroll..... Trommelwirbel.....

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

Und wieder anschließen und die Amadeus-Software erneut starten. Dieses mal muss die Firmware korrekt ausgelesen werden.

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

So, das war's! Viel Spaß und genießt die schnellen Software-Updates. Nie wieder langes Warten.

3 Amadeus Software Description/Beschreibung

3.1 Basics/Grundlagen

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 summarised in the following paragraphs. Slight differences might exist between several implementations.

Als erstes wähle bitte die Mikrocontroller-Familie aus. Jede Familie hat ihre eigene Benutzeroberfläche. Von wenigen Elementen abgesehen gibt es viele Gemeinsamkeiten, die in den folgenden Abschnitten beschreiben werden.

3.1.1 File Selection/Dateiauswahl

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.

Wähle die zu programmierende Hex-Datei. Klicke auf den Ordner rechts neben dem Dateieingabefeld. Alle Dateien müssen im Intel-Hex Format vorliegen.

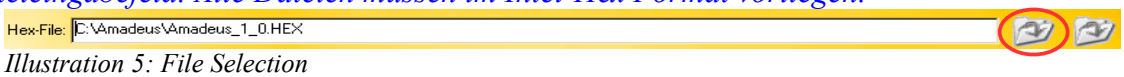

Illustration 5: File Selection

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

Für PIC-Controller können alle Information (Programm, EEPROM, ID und Konfigurationdaten) in einer Hex-Datei zusammengefasst sein.

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.

Bei Atmel enthält die Hex-Datei nur das Programm. Deswegen gibt es eine weitere Ordner-Schaltfläche rechts neben der Dateiauswahl-Schaltfläche für den Inhalt des EEPROMs. Abwählen kann man die EEPROM Datei, wenn man das Auswahlfenster öffnet und die Auswahl dann abbricht.

3.1.2 Identify-Button/Schaltfläche

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.

*Ein Klick auf die **Identify** Schaltflächen zeigt die angeschlossene MCU an, sofern sie von der gewählten Benutzeroberfläche unterstützt wird. Bei einigen MCUs werden erst nach erfolgreicher Erkennung die Konfigurationsdaten angezeigt.*

3.1.3 Get Configuration From-Button/Schaltfläche

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

Ließt die aktuellen Konfigurationsdaten aus der MCU und zeigt sie an.

3.1.4 Program, Read, Verify/Programmieren, Auslesen, Vergleichen

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

Die drei Schaltflächen Program (programmieren), Read (auslesen) und Verify (vergleichen) lösen die entsprechenden Aktionen aus.

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

Illustration 6: Action Selection

*Wenn **Fast Verify** angewählt ist, werden beim Vergleichen alle Bereiche übersprungen, die in der Hex-Datei nicht definiert sind. Es findet also keine Prüfung auf 0xFF statt. Beim Lesen sollte diese Option inaktiv sein.*

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

*Mit **No Verify** wird jeglicher Vergleich unterdrückt. Es wird immer SUCCESS (Erfolg) angezeigt.*

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

*Die Optionen **Erase Chip**, **Software**, **Flash Memory**, **EEPROM**, **ID-Location** und **Configuration** aktivieren oder unterdrücken die entsprechende Aktion.*

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

*Wenn **Ignore Hex-File Configuration** (nur PIC) angeharkt ist, werden eventuelle Konfigurationsdaten aus der Hex-Datei ignoriert und stattdessen die angezeigten Daten programmiert/verglichen.*

Some additional functionality might be available for some MCUs.

Je nach MCU kann es noch weitere Funktionen geben.

3.1.5 RAM/EEPROM Read Reset

This is a little debugging feature. As far as possible it reads RAM (PIC18/dsPIC30 only) or EEPROM content via the programming interface and displays them as hex dump. In all cases a reset accompanies this request.

Diese ist ein kleines Debugging-Feature. Es ließt RAM- (nur PIC18/dsPIC30) oder EEPROM-Inhalte aus und zeigt diese als Hex-Dump an. Das ganze wird von einem Reset begleitet.

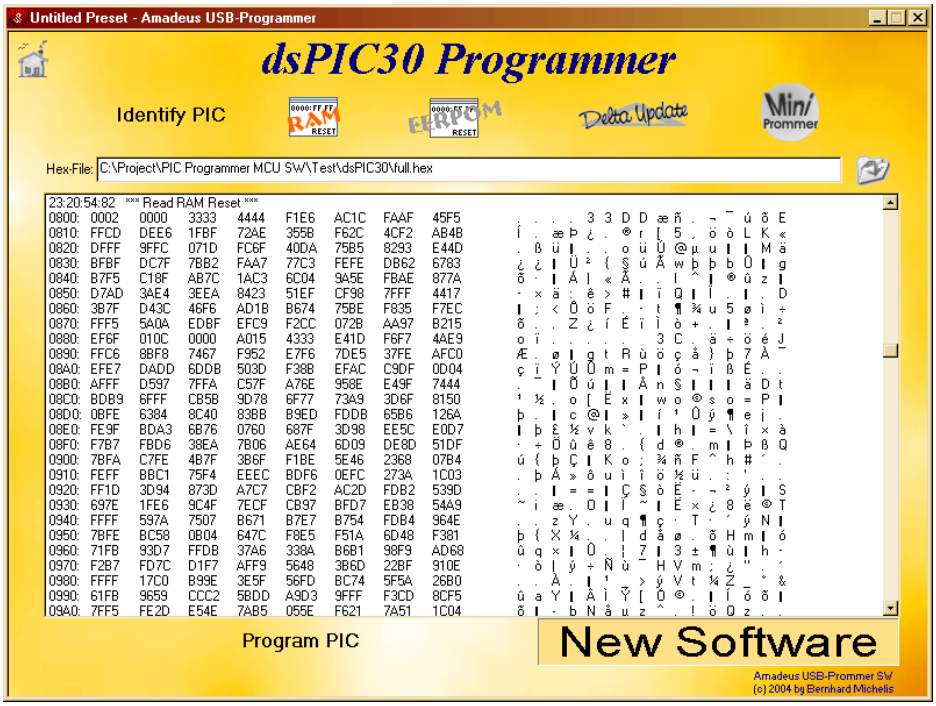

Illustration 7: dsPIC30F6014 RAM Hex-Dump grouped by 16bit words

3.1.6 Connection and Power Switching/Schalten der Versorgungsspannung

Because the programming interface goes into high-impedance mode (see Electrical Characteristics/Elektrische Eigenschaften on page 47) 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.

Da die Programmierschnittstelle nach dem Programmieren hochohmig wird (siehe auch Seite 47), ist es im allgemeinen unnötig den Programmer zu entfernen, damit die Zielschaltung normal arbeiten kann. Hier ist es dann natürlich interessant, die gesamte Schaltung Zurückzusetzen, da ja einige Teile durch die Programmierung durcheinander geraten sein können.

Therefore the programmer can switch an external power supply. See Electrical Characteristics/Elektrische Eigenschaften on page 47 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).

Dafür bietet der Programmer die Möglichkeit die Spannungsversorgung des Zielsystem aus- und wieder einzuschalten. Man sollte allerdings darauf verzichten Spannungen von mehr als 20V mit dem Relais zu schalten. Zum Schalten von Netzspannungen mußt Du dir eine eigene Schaltung entwerfen, die dann vom Relais oder EXTVCC angesteuert werden kann.

EXTVCC liefert während der Programmierung ca. 4,6-5,2V und ist als Spannungsversorgung für Standalone Programmiergeräte gedacht.

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

Achtung: Relais auf keinen Fall zum Schalten hoher Spannungen verwenden.

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.

*Das Schalten der Spannungsversorgung mittels Relais (**Power Switching**) kann auf der MCU Auswahlseite eingestellt werden.*

*Neben **kein Schalten** (inactive), **Handbetrieb** (manual) gibt es eine Reihe vordefinierter Zeiten. Hier kann man dann auch entscheiden, ob nach dem Programmieren die Spannung ausgeschaltet (off) oder aus und wieder eingeschaltet werden soll (on).*

3.2 Debugging Features/Fehlersuch Möglichkeiten

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

Amadeus ist kein In-Circuit-Debugger. Trotzdem bietet es einige einfache Möglichkeiten der Fehlersuche im System.

The EEPROM-Read-Rest gives you the opportunity to display data as hex-dump. So it is possible to read back data from your 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.

Der EEPROM-Read-Reset gibt Dir die Möglichkeit ein Hex-Dump des EEPROMs abzurufen. Selbst wenn noch kein anderer Weg zur PC–Mikrokontroller Kommunikation eingerichtet ist, kann man so Informationen aus dem Mikrokontroller zurücklesen. Schreibe die interessanten Daten einfach in das EEPROM. Ein Macro hierfür kannst Du in MP-Lab oder AVR-Studio entwickeln und testen.

Feel lucky, if you are using a PIC18 or a dsPIC30. Amadeus can read PIC18/dsPIC30 RAM directly, so no EEPROM write macros are necessary for debugging.

Sei froh, wenn Du einen PIC18 oder dsPIC30 verwendest, denn Amadeus kann das RAM des PIC18/dsPIC30 direkt lesen. Das Aufbereiten der Daten für das EEPROM kann entfallen.

In all cases, a reset accompanies the reading.

Alle Leseaktionen werden von einem Reset begleitet.

3.2.1 How do I use this feature/Was fang ich damit an?

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

Stell Dir vor, Du versuchst einen UART zu programmieren und nichts funktioniert. Vielleicht fragst Du Dich irgendwann, "Kommen meine Daten überhaupt an?". Natürlich könnte man in der RX-ISR eine LED anschalten, wenn ein Byte empfangen wird, aber ob das Byte richtig ankam, erfährst Du so nicht. Denke an falsche Baudrate, falsche Parität...

In so einem Fall kopiere das empfangene Byte ins EEPROM/RAM gefolgt von einer Endlosschleife. Jetzt verwende die EEPROM/RAM-Read-Funktion.

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

Es kann vorkommen, dass einige MCUs bereits einige Befehle ausführen, bevor der Speicher gelesen werden kann. In solchen Fällen am Anfang es Programmes ein paar NOPs einfügen.

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.

Der andere Weg Deine Software in der Schaltung zu Debuggen ist:

- *Verwende Endlosschleifen anstelle von Breakpoints.*
- *Kopiere alle interessanten Daten vor der Endlosschleife ins EEPROM/RAM.*
- *Führe den Read-Reset aus.*
- *Setze die Kopiermakros und die Endlosschleife zum nächsten interessanten Punkt, reassembliere und programmiere Deinen Schaltung neu.*
- *Und so weiter...*

3.2.2 Suggestion/Empfehlung

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. If you need a powerful MCU take a dsPIC30.

- 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.

Tipp: Wenn die Entscheidung, welche MCU verwendet werden soll, bei Dir liegt und zusätzliche 4€ nicht weh tun, nimm einen PIC18, da der RAM-Read-Reset einige Vorteile bietet. Wird mehr Rechenleistung benötigt, bieten sich die dsPICs an.

- *Bei jedem lesen werden alle Variablen angezeigt (Hex-Dump)*
- *Wenn beim Programmstart der komplette Speicher mit 0xFF gefüllt wird, kannst Du leicht sehen, wenn eine Funktion wild im Speicher rumschreibt und alles durcheinander bringt.*
- *Es brauchen keine Daten ins EEPROM kopiert werden, abgesehen von SFRs.*

3.2.3 Tracing/Tracen

Meanwhile, there is a second way to debug your target MCU. Next to the Read-Resets Amadeus offers a way to trace the path of program execution inside the target MCU.

What does tracing mean? Imagine your program is executed. You press a key and something should happen, but nothing happens. Now you can add some trace information to your program code. Look for some key execution pathes inside your program. There you should add Amadeus trace macros. As soon as the MCU executes these program paths, the trace macros will be executed, too. Now the trace macros send some data (a number for every position, maybe with some additional information like the current contents of the RAM) to Amadeus and the Amadeus software shows these information in the trace window.

In case of the upper mentioned example, you might see that trace point 1 was passed. Trace point 1 was inside the ISR that sampled the key action.

Trace point 2 might be the entry point of an event loop in your program. Therefore, this trace number should appear in the trace window to show, that is point of program execution was passed. Next, there could a key recognition function that branches according to the key code. Trace point 3 would be in the handler function of your key. So when the trace points 1 and 2 were passed, but trace point 3 will not be passed, you have to look for your mistake between trace point 2 and trace point 3.

Now, that you know that the problem is between point 2 and 3, you could add more trace points between them or you could add a trace point, that sends additional information, such RAM contents, to the trace window. So you might see, that the compared bytes differ in one bit position. I hope this gives you an idea of the possibilities of tracing.

Mittlerweile bietet Amadeus neben den Read-Resets eine weitere Möglichkeit der Fehlersuche. Das so genannte Tracen.

Was bedeutet Tracen? Beim Tracen geht es darum, dass man sein Program ganz normal in der Zielschaltung laufen lässt und dabei die Programmausführung mitverfolgt, ähnlich wie mit einem In-Circuit-Debugger. Allerdings läuft das Programm immer weiter. Es gibt keine Möglichkeit das Programm zu stoppen. Nun gibt es einige markante Punkte in der Programmausführung, deren Ausführung uns darüber Aufschluß gibt, was das Programm gerade macht.

Beispiel: Wir haben ein Programm, das beim Drücken einer Taste eine Aktion auslösen soll, was es aber nicht tut. Um den Fehler zu suchen, überlegen wir uns, welchen Pfad das Programm nehmen müsste, wenn es richtig arbeiten würde. Wir setzen an den markanten Stellen dieses Ausführungspfads Trace Macros ein.

Wenn wir das Programm erneut laufen lassen, wird z.B. die ISR durchlaufen, die die Taste abfragt. Da wir in diesem Programmteil unseren Trace-Punkt 1 eingebaut haben, sendet dieser nun eine Nachricht an den Programmer und wir sehen im Trace-Fenster, dass Trace-Punkt 1 soeben durchlaufen wurde. Wir wissen nun, dass die Taste abgefragt wurde. Unser Programm hat z.B. noch eine Programmschleife zur Verarbeitung aller hereinkommenden Nachrichten. Hier befindet sich Trace-Punkt 2. Wird auch dieser im Trace-Fenster angezeigt, wissen wir, dass der Tastendruck offensichtlich bis hierhin gekommen ist. Jetzt sollte das Programm, je nach gedrückter Taste, unterschiedlich verzweigen. Trace-Punkt 3 befindet sich in der Behandlungsroutine für die Taste. Wenn Trace-Punkt 3 aber nicht im Trace-Fenster

dargestellt wird, wissen wir, dass wir den Fehler zwischen Punkt 2 und Punkt 3 suchen müssen.

Als nächstes fügen wir zwischen Punkt 2 und 3 weitere Trace-Punkte ein. Es ist auch möglich sich nicht nur einen Trace-Punkt anzeigen zu lassen, sondern auch den Inhalt einiger RAM-Zellen mitzuschicken. Hier z.B. der Nachrichtencode unserer Taste. Und bei der nächsten Programmausführung würden wir vielleicht feststellen, dass wir zwar die Taste erkannt haben, aber wir mit dem falschen Code verglichen haben...

Ich hoffe, dass Ihr nun eine Vorstellung davon habt, was man mit der Trace-Funktion so alles anstellen kann.

3.2.4 What do I need to use the trace function? Was brauche ich, um die Trace-Funktion zu nutzen?

In general, the trace functionality is available for every type of MCU. Even MCUs, not supported by Amadeus, could use this feature. All you need is to do is to connect two pins of your MCU (for PIC and dsPIC you can directly use the programming pins) with the programmer. There is one clock and one data line. Those have to be connected to the SCL/SDA pins of the programmer hardware. Further more both lines need to be pulled up to VCC. In case you are using the programming pins of the target MCU to trace, SCL/SDA are in parallel to PGC/PGD (Illustratio

n 8). In this case you can activate the internal weak pull-up resistors of Amadeus. Just click **Enable Pullups** before you activate **Trace**. In case you have to use different pins for tracing (Illustration 9), SCL/SDA need a 10k pull-up resistor to EXTVCC of Amadeus. **Enable Pullups** must not be active in this case, so that PGC/PGD will remain high impedance.

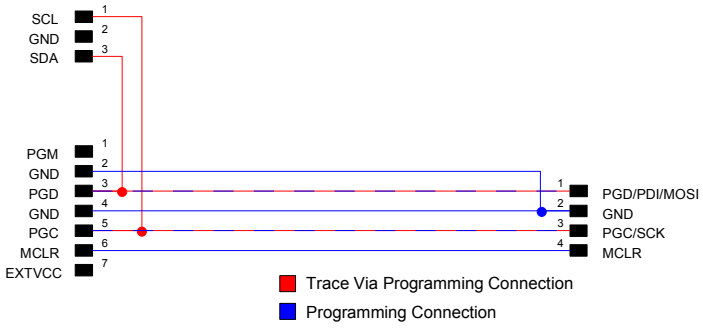

Illustration 8: Connection for tracing via programming pins

Ganz allgemein kann man sagen, dass die Trace-Funktion für jede erdenkliche MCU zu gebrauchen ist. Sogar für MCUs, die von Amadeus nicht unterstützt werden. Alles was dazu nötig ist, sind zwei Leitungen (für PIC und dsPIC können die Programmierleitungen verwendet werden, bei AVR kann man dieses auch tun, oder wenn man die Programmierpins für andere Zwecke braucht [Serieller Port!] auf andere Pins ausweichen). Es gibt eine Takt- (Clock) und eine Datenleitung (Data). Diese müssen mit SCL/SDA vom Programmer verbunden werden. Zusätzlich müssen beide Leitungen über einen Pull-Up Widerstand nach VCC gezogen werden. Verwendet man die Programmierpins zum Tracen, so liegen SCL/SDA

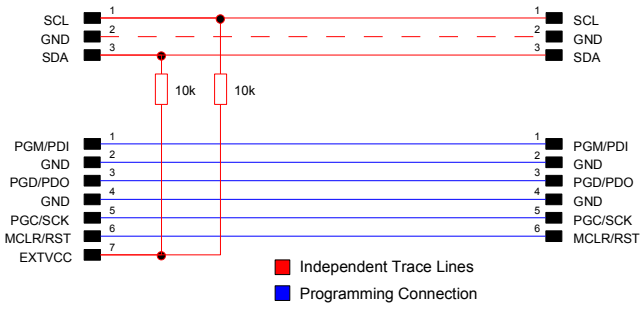

Illustration 9: Connection for individual program and trace lines

*parallel zu PGC/PGD (Abbildung 8). In diesem Fall kann man die internen Weak-Pull-Up Widerstände von Amadeus mit **Enable Pullups** aktivieren. Verwendet man hingegen andere Pins, liegen sie nicht mehr parallel zu PGC/PGD (Abbildung 9) und benötigen je einen 10k Widerstand gegen EXTVCC vom Programmiergerät. **Enable Pullups** muss deaktiviert sein, damit PGC/PGD hochohmig werden und die Funktionen an den Programmierpins nicht stören.*

After the MCU is connected properly, you'll have to add some lines of code to your project. Even if there are some predefined trace macros/functions (for some MCUs in assembly language), here is all the necessary information you need to design your own trace functions or port them into any other language like C(++).

When tracing is activated, both lines (SCL and SDA) need to be pulled up to VCC. When you intend to use the programming pins for tracing, just activate the internal pull-up resistors with the **Enable Pullups** check box. Otherwise it is necessary, that both lines are pulled up by external resistors to EXTVCC. EXTVCC will be activated automatically, when tracing is activated.

Your program can send traces in the following manner. After reset it is important, that both pins are in high impedance mode. Before you are allowed to send data, you'll have to check whether the clock line is high. When the clock line is low, you are **not allowed** to send any data on the trace bus (**very important!**). In the future a low on the data line might be important, too.

Now, that you are allowed to send data (clock high), the following rules must be kept (Illustration 10): Amadeus shifts data in, on the rising edge of SCL, most significant bit first. You are allowed to transmit several bytes as a chunk. After the last bit is sent, both pins must return into high impedance state.

A trace message must have this form: The first byte indicates the length of the message. The length byte itself does not count. The length byte value must be in the range of 1 to 31. To reduce traffic and loss of performance, there is a special definition for simple tracing. When the first byte (the lenght byte) is in the range of 128 to 254 (255 is not allowed), it is interpreted as lenght=1 with the trace number x-128 (=>0-126).

How fast can you send data? As a rule, less is better. The average shouldn't be higher than 110kBytes/s. Short messages (<32 byte) can be sent with up to 200kBytes/s, when there is a break afterwards, so that the average is below 110kBytes/s (9µs per byte). A single byte can be sent with up to 100ns/bit.

If you plan to send trace data out of interrupt service routines, too, you will have to disable all interrupts that might interrupt a trace function, before calling the trace function. If an ISR does not send trace data of its own, it can be enabled.

With **Suppress Trace** you can force the traget MCU to skip all trace functions. In this case the programmer pulls SCL low. Because all trace functions have to test for SCL high, the trace function calls are skipped and the program will run a little bit faster.

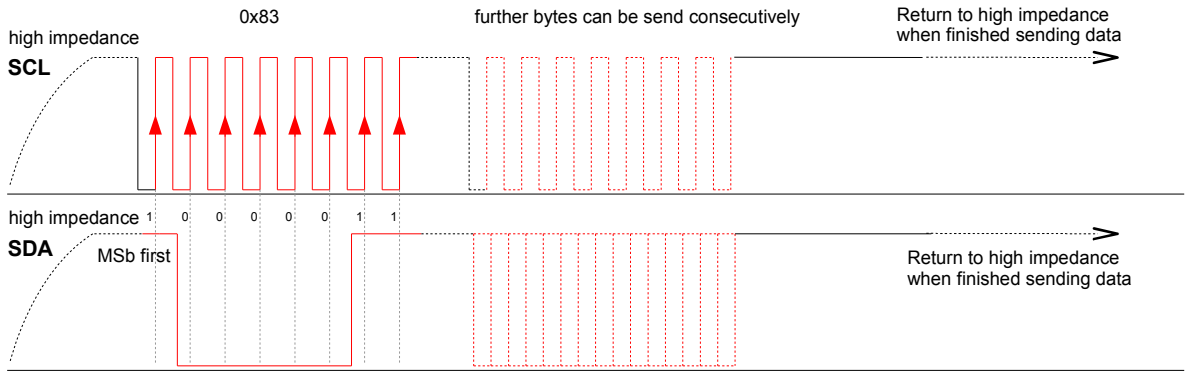

Illustration 10: Trace Data Transmission Schema

Nachdem Du die MCU mit dem Programmer richtig verbunden hast, mußt Du noch ein paar Programmzeilen zu Deinem Programm hinzufügen. Für den einen oder anderen Mikrokontroller habe ich zwar vorgefertigte Include-Dateien beigelegt, wer allerdings mit anderen Mikrokontrollern arbeitet oder aber mit anderen Sprachen wie C(++), der kommt nicht umhin sich eigene Makros und Funktionen zu schreiben.

Wenn Tracing aktiv ist, müssen beide Leitungen (SCL und SDA) auf VCC gezogen werden. Wer die Programmierpins zum Tracen verwendet, braucht nur die internen Pull-Up Widerstände mit einem Häkchen neben **Enable Pullups** *zu aktivieren. Andernfalls müssen beide Leitungen mit einem 10k Widerstand mit EXTVCC verbunden werden. EXTVCC wird automatisch aktiviert, sobald* **Trace** *aktiviert wird.*

Dein Programm kann nun wie folgt Daten senden: Es ist wichtig, dass die Trace-Leitungen nach dem Reset hochohmig (Default Input) sind. Bevor Du nun Daten senden darfst, mußt Du erst überprüfen, ob die Clock-Leitung high ist. Ist SCL low, dürfen ***auf gar keinen Fall*** *Daten über den Tracebus gesendet werden (****sehr wichtig****). In Zukunft kann auch ein Low auf der Datenleitung wichtig sein.*

Wenn aber beide Leitungen high sind, können Daten gesendet werden. Amadeus übernimmt Daten bei der positiven Flanke von SCL, wobei das höhstwertigste Bit zuwerst übertragen wird (Abbildung 10). Du kannst mehrere Bytes direkt nacheinander senden. Nachdem das letzte Bit übertragen wurde, müssen beide Leitungen wieder hochohmig geschaltet werden.

Eine Trace-Nachricht muss folgende Form haben: Das erste Byte enthält die Länge der Nachricht. Das Längenbyte selber zählt nicht mit. Die Länge muss zwischen 1 und 31 liegen. Um das Datenaufkommen zu reduzieren und den Mikrokontroller nicht unnötig zu belasten, gibt es eine Besonderheit. Liegt das erste Byte im Bereich von 128 bis 254 (255 ist nicht erlaubt) wird die Länge=1 angenommen und der Tracewert als x-128 (also als 0-126) interpretiert.

Wie schnell dürfen Daten gesendet werden? Allgemein gilt, je weniger und lansammer, desto besser. Im Durchschnitt darf man nicht mehr als 110kBytes/Sekunde (9µs pro Byte) übertragen. Kurzzeitig sind auch 200kBytes/Sekunde erlaubt. Ein einzelnes Byte darf mit bis zu 100ns/Bit übertragen werden.

Wenn Du beabsichtigst Trace Daten sowohl aus dem Hauptprogramm als auch aus einer oder mehrer Interrupt Service Routinen zu senden, so gilt, dass man alle Interrupts abschalten (disablen) muss, welche selber Trace-Daten senden wollen. ISRs ohne Trace Aufrufe dürfen erlaubt (enabled) bleiben.

Mit **Suppress Trace** *kann man vorrübergehen das Senden von Trace Daten unterdrücken. Hierzu zieht der Programmer die SCL Leitung auf low. Da alle Trace Routinen auf SCL high Prüfen müssen, werden so alle Aufrufe zum Senden von Trace Daten unterbunden. Die Programmgeschwindigkeit der Ziel-MCU wird folglich weniger gebremmst.*

3.2.5 Formating Trace-Messages/Formatieren von Trace-Nachrichten

Normally trace messages are displayed in the form:

Trace 2: 00, 55, AA ...

But it is possible to format such hex-strings. Therefore you have to create an ASCII file with the extension .tdf or .txt.

The structure of the file is very simple. There is a fix syntax for a format string. Every line that does not match this syntax is considered to be a comment.

```
Trace Definition file:
L=21:C=01:S=xx:M=The memory is: \x \x \x \x \P03-\x \P02\x \P01\x \P00\x
L=01:C=02:S=03:M=\R
write comments wherever you like...
L=01:C=02:S=xx:M=Text \R
L=01:C=01:S=xx:M=More Text \R
```

Text 1: Trace Definition File

First sign in a row must be a capital 'L' directly followed by an equals sign '='. Then the minimum length of the trace-message as two-digit hex number i.e. '0A' for at least 10 bytes. Next there is a colon followed by a capital 'C'+'=' followed by the trace-code as two-digit hex-code (first byte of message) followed by ':S='. If you want a second byte to match, just add the second byte of the trace-message as two-digit hex. If you want to check only the first byte, write “S=xx” (small x). After the following “:M=” you can define the subsitution string to the end of the line. Except the backslash '\' every character is allowed.

All bytes that follow the code byte (or subcode byte if not xx) are the data. Add the following escape sequences for special functions. By default low byte first and word lengh equals one byte is used.

See Text 2 on page 19 for an example.

Command	*Description*
\r	Add the rest of the trace message as hex dump. *Fügt den Rest der Trace-Nachricht als Hex Dump an.*
\R	Add the rest of the trace message as ASCII string. *Fügt den Rest der Trace-Nachricht als ASCII-String an.*
\1, \2, \3 or \4	Determines the size of the next displayed number (hex and dec) *Bestimmt, welche Größe, in Byte, die Zahlen haben, die man mit \X, \D und \S ausgeben kann.*
\l or \L	Defines that bytes are sent low byte first *Bestimmt, das die Daten Low-Byte-Zuerst gesendet werden.*
\h or \H	Defines that bytes are sent high byte first *Bestimmt, das die Daten High-Byte-Zuerst gesendet werden.*
\x or \X	Display next data as hex (1-4 byte!!!) *Zeigt die nächste Zahl als Hex an (1-4Bytes!!!)*
\d or \D	Display next data as decimal (1-4 byte!!!) *Zeigt die nächste Zahl dezimal an (1-4Bytes!!!)*
\s or \S	Display next data as signed decimal (1-4 byte!!!) *Zeigt die nächste Zahl als vorzeichenhafte Dezimalzahl an (1-4Bytes!!!)*
\a or \A	Display next data as character *Zeigt das nächste Byte als Zeichen an*
\f or \F	Display next 4 bytes as float (this requires, that the float (32 bit) is sent MSB first) *Zeigt die nächsten 4 Bytes als float an (das setzt voraus, dass ein float (32 Bit) MSB übertragen wurde)*
\phh or \Phh	Allows to resort the data. When your data are 01, 02, 03, 04 you could write "\P03\x \P00\x" to first output the fourth value (zero based) and then output the first value as hex. The position is zero-based and has to be written as two-digit-hex. *Erlaubt das hin- und herspringen in der Trace-Nachricht. Wenn also die Werte 01, 02, 03, 04 übertragen wurden, dann kann man mit "\P03\x \P00\x" zuerst die vierte Stelle (wir zählen von NULL) und dann die erste Stelle ausgeben.*
\\	Backslash

Table 1: Trace Format

Normalerweise werden Trace Nachrichten als Hex-Dump ausgegeben.

Trace 2: 00, 55, AA ...

Es ist aber möglich diese unleserlichen Zahlenketten als Text zu Formatieren. Hierzu mußt Du eine Textdatei mit der Endung .tdf oder .txt erzeugen. Siehe Text 1 auf Seite 17.

Jede Trace-Zeile wird nun der Reihe nach von oben nach unten mit den Formatierungsvorschriften verglichen. Stimmen die Bedingungen, wird die Formatierung angewendet.

Die Zeilen, die mit groß 'L' beginnen, sind Formatierungsvorschriften. Bis zum 'M=' dürfen keine Leerzeichen eingefügt werden. Alle anderen Zeilen werden als Kommentare ignoriert.

Hinter 'L=' wird die Mindestlänge der Trace-Nachricht in Bytes angegeben. Für Nachrichten mit weniger Bytes kommt diese Formatierung nicht in Frage. Die Länge muss als zweistellige Hex-Zahl angegeben werden.

Als nächstes wird der Trace-Code (das erste Byte im Hex-String) auf Übereinstimmung verglichen. 'C=' + Trace-Code in zweistelliger Hex-Form.

Bei Bedarf kann man noch einen Trace-Subcode 'S=' definieren (zwei stellig hex! Das zweite Byte in der Hex-Nachricht). Wenn man das nicht will, muss man 'S=xx' schreiben (xx klein!).

Nachdem nun, entsprechend Reihenfolge, Länge, Trace-Code und gegebenenfalls noch Trace-Subcode, eine Formatierung ausgewählt wurde, folgt nach 'M=' der Ersetzungstext. Abgesehen von '\' (Backslash) ist jedes Zeichen erlaubt.

Siehe Tabelle 1 für eine Beschreibung der Sonderkommandos.

Voreinstellungsmäßig werden Zahlen mit der Länge ein Byte ausgegeben. Auch wird davon ausgegangen, dass das niederwertigste Byte zuerst übertragen wird.

```
L=02:C=5A:S=00:M=LED Off
L=02:C=5A:S=01:M=LED On

Without Formating / Ohne Formatierung       0x5A(hex)=90(dec)
16:53:22:78 Trace 90 - 00
16:53:22:78 Trace 2 - 02
16:53:23:37 Trace 90 - 01
16:53:23:37 Trace 2 - 02
16:53:23:95 Trace 90 - 00
16:53:23:95 Trace 2 - 02

With Formating / Mit Formatierung
16:53:24:65 LED On
16:53:24:65 Trace 2 - 02
16:53:25:39 LED Off
16:53:25:39 Trace 2 - 02
16:53:25:81 LED On
16:53:25:81 Trace 2 - 02
16:53:26:35 LED Off
16:53:26:35 Trace 2 - 02
16:53:26:90 LED On
```

Text 2: Trace Formarting Example

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.

For the ATtiny2313 use the ATmega programming interface. See the next chapter.

Der Programmer arbeitet bei diesen Typen immer im Hochvolt-Modus.

Für den ATtiny2313 bitte das Programmierinterface vom ATmega benutzen. Siehe nächste Kapitel.

3.3.1 How To Connect/Programmierschnittstelle

For electrical details see chapter “Electrical Characteristics/Elektrische Eigenschaften“ on page 47.

Für elektrische Details bitte im Kapittel “Electrical Characteristics/Elektrische Eigenschaften“ auf Seite 47 nachschlagen.

6 lines are required for programming. *6 Leitungen werden benötigt.*

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit. *Beim Programmieren in der Schaltung **nicht anschließen**.*
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming. *Spannungsversorgung für Standalone Programmiergeräte. **Nie mit einer externen Spannung verbinden**.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	PB5	Reset (about 12V during programming) ***Reset kann wärend der Programmierung auf 12V gehen.***
PGM/PDI	PB0	Serial Data Input/*Serieller Dateneingang*
PGD/PDO	PB1	Serial Instruction Input/*Serieller Befehlseingang*
PGC/SCK	PB2	Serial Data Output/*Serieller Datenausgang*
SCL	PB3	Serial Clock Input/*Takteingang*

Table 2: ATtiny programming adapter definition

3.3.2 Programming/Programmieren

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

Die Grundlagen findest Du im Kapitel “Basics/Grundlagen“ auf Seite 7.

3.3.3 Additionlly Features/Zusätzliches

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).

Es ist möglich das Kalibrierungsbyte entweder ans Ende des Flash oder am Anfang/Ende es EEPROMS zu positionieren (nur ATtiny12).

3.4 ATmega/ATtiny except ATtiny11/12 / außer ATtiny 11/12

See chapter “Supported Devices/Unterstützte Mikrocontroller“ on page 4 for a list of supported devices. On the contrary to other MCUs, ATmegas need a valid clock to be programmed. The internal 1MHz oscillator is enabled by default. See documentation of your MCU.

Eine Liste aller unterstützter MCUs findest Du im Kapitel “Supported Devices/Unterstützte Mikrocontroller“ auf Seite 4.

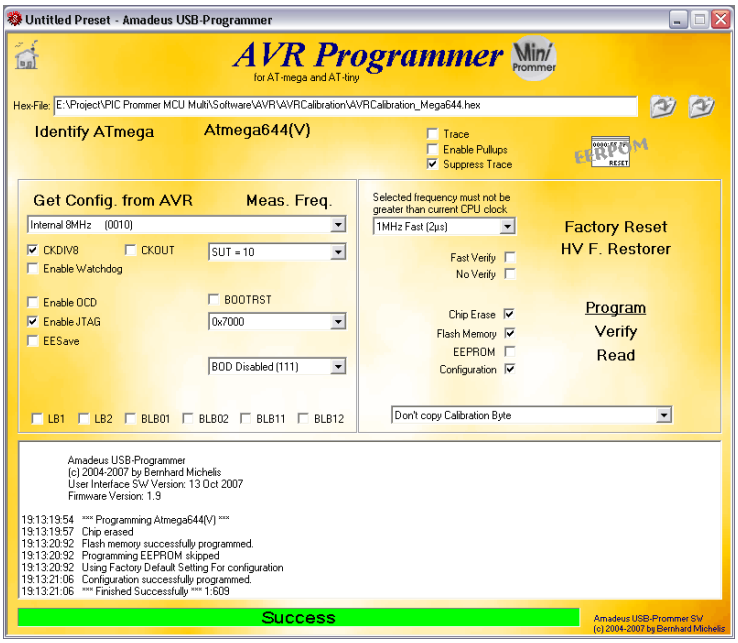

Illustration 11: AVR Programming Interface

3.4.1 How To Connect/Programmierschnittstelle

For electrical details see chapter “Electrical Characteristics/Elektrische Eigenschaften“ on page 47.

5 lines are required for programming.

Für elektrische Details bitte im Kapitel “Electrical Characteristics/Elektrische Eigenschaften“ auf Seite 47 nachschlagen.

5 Leitungen werden benötigt.

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit. *Beim Programmieren in der Schaltung **nicht anschließen**.*
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming. *Spannungsversorgung für Standalone Programmiergeräte. **Nie mit einer externen Spannung verbinden**.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	RESET	Only low voltage programming. *Nur Niedervolt Programmierung.*
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”) *MOSI: Dateneingang*
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”) *MISO: Datenausgang*
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”) *SCK: Takt*

Table 3: ATmega programming adapter definition

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

Beachte, dass verschiedene MCUs unterschiedliche Pins zur Programmierung verwenden.

3.4.2 Programming/Programmierung

Only additional information on programming are given in the paragraph. Have a look at the chapter “Basics/Grundlagen” on page 7 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.

Die Grundlagen findest Du im Kapitel “Basics/Grundlagen“ auf Seite 7.

Vor dem Programmieren muss die richtige Programmiergeschwindigkeit gewählt werden.

Option	*Description*
10000kHz Normal (2ms)	Select this option if you clock your MCU below 1MHz and internal oscillator does not offer internal 8MHz clock. *Wähle diese Einstellung, wenn die MCU mit weniger als 1MHz getaktet ist, und der interne Oszillator keine 8MHz bietet.* PT64: 20.6s

Option	*Description*
10000kHz Fast (2ms)	Select this option if you clock your MCU below 1MHz and the internal oscillator offers 8MHz. *Wähle diese Einstellung, wenn die MCU mit weniger als 1MHz getaktet ist, und der interne Oszillator 8MHz bietet.* PT64: 5.9s
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. *Wähle diese Einstellung, wenn die MCU mit wenigstens 1MHz getaktet ist, und der interne Oszillator keine 8MHz bietet.* PT64: 19.8s
1MHz Fast (2µs)	Select this, when MCU is clocked at 1MHz or faster. (Default) *Wähle diese Einstellung, wenn die MCU mit wenigstens 1MHz getaktet ist, und der interne Oszillator 8MHz bietet.* PT64: 4.3s
6,7MHz (300ns)	Select, when MCU is running above 6.7MHz. E.g. the internal oscillator@8MHz or a crystal, or ... *Wähle diese Einstellung, wenn die MCU mit wenigstens 6,7MHz getaktet ist.* 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. *Wähle diese Einstellung, wenn die MCU mit 10-12MHz oder mehr als 15MHz getaktet ist.* PT64: 3.7s

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.

If any Lock Fuses are set, don't use fast Mode. If the MCU clock is below 1MHz click Factory Reset first.

***PT64** gibt die Programmierzeit für das 64k Flash des ATmega64 an, wenn es komplett mit 0x00 beschrieben und verglichen wird.*

Wichtig: Wenn Lock Fuses gesetzt sind, muss man auf den Fast-Mode verzichtet. Und bei einem MCU Takt von weniger als 1MHz vorher einen Factory Reset durchführen.

- During development it is recommended to select ***Fast Verify***.
- 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.
- *Währen der Entwicklung sollte immer Fast Verify aktiv sein.*
- *Wenn das EEPROM nicht gelöscht werden muss, sollte die **EESave** Fuse gesetzt werden.*

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

Wenn das Programm den Flashspeicher nicht ausfüllt, werden leere Bereiche übersprungen.

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, or 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 a parallel programmer is able to reanimate the MCU. (See chapter 3.4.6)

Warnung

Einige MCUs haben eine RSTDISBL Fuse. Amadeus bietet zwar keine Möglichkeit diese expliziet zu setzen, aber wenn ein falsches Timing (zu schnell) gewählt wird, kann es doch passieren. Also bei solchen MCUs unbeding auf die richtige Programmiergeschwindigkeit achten, da ansonsten die MCU nicht mehr über SPI angesprochen werden kann. Hier hilft dann nur noch ein paralleler Hochvolt-Programmer. (Siehe auch Kapitel 3.4.6)

3.4.3 Factory Reset / Reanimate Function / Wiederbedebung

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).

If Tracing is used, you must disconnect the Trace-connection before performing the **Factory Reset**, because while the reset is performed, a clock signal is generated on **SCK**. This might interfere with programming clock line when connected as in Illustration 8 on p. 14.

*Ein Fehler, der immer wieder gemacht wird, ist, dass die falschen CKSRC Fuse Bits (Oszillator Auswahl) gesetzt werden. Hier schaft der Factory-Reset Abhilfe. Im Falle des Falles den **SCL Pin** des Programmers mit **XTAL1** der MCU verbinden und Factory-Reset anklicken. Danach wieder trennen.*

*Wenn die Trace-Funktion von Amadeus verwendet wird, sollte die Traceverbindung vor einem **Factory Reset** getrennt werden, da während des Resets auf **SCL** ein Taktsignal erzeugt wird, das mit dem Programmiertakt (**SCK**) zusammenfällt, wenn die Zielschaltung wie in Abbildung 8 auf S. 14 Verbunden ist. Wenn der **SCL** Pin, wie oben beschreiben mit **XTAL1** verbunden wird, sollte die Trace Verbindung auch entfernt werden.*

3.4.4 Oscillator Calibration Byte / Kalibriungsbytes

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.

Amadeus kann die Kalibriungsbytes bei Bedarf ans Ende des Flashspeichers oder an den Anfang/das Ende des EEPROMs kopieren.

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.

Eine neu Kalibrierung kann der Programmer nicht durchführen, aber wer seinen Chip genau kalibriert hat, sollte diesen Wert ans Ende des Flashspeicher legen (Highbyte) und beim Programmieren ***Preserve Calibration Byte at end of Flash*** *wählen.*

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.

Hinweis: *Im Falle von* ***Preserve Calibration Byte*** *nie das Flash löschen ohne es neu zu Programmieren. Beim* ***Factory Reset*** *geht das Byte ebenfalls verloren.*

3.4.5 EEPROM Programming / EEPROM Programmierung

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.

Für die ATmega Familie handhabt Amadeus die EEPROM Programmierung auf eine besondere Art. Wenn EESave aktiv ist oder Erase Chip nicht markiert ist, werden nur die in der Hex-Datei definierten Bytes programmiert. Die anderen Zellen bleiben unberührt.

Das kann für die Ablage des Kalibrierungsbytes im EEPROM von Bedeutung sein.

3.4.6 Reanimation / Wiederbelebung

Under certain conditions, it might happen, that it is no longer possible to talk to the ATmega, if the fuses got corrupted. The only way to reanimate such a MCU is to reset the fuse bits in high-voltage parallel programming mode. Therefore a special parallel interface is needed. See *ATmega parallel interface* on the download website. Using this additional interface, you can reset the fuses, so that the serial programmer can talk to the MCU again. Connect the MCU with the parallel interface to Amadeus hardware and click the **HV F. Restorer** button.

There is no feedback. But if everything is correct, the MCU can be programmed again in a serial fashion.

Wenn die Fuse-Bits des ATmega durcheinander geraten sein sollten, lässt sich die MCU unter Umständen nicht mehr seriel programmieren. Mit Hilfe eines parallelen Adapters (siehe **"ATmega parallel interface"** *auf der Download Website) ist es aber möglich, die Fuse-Bits so zurückzusetzen, dass die MCU wieder seriell ansprechbar wird. Hierzu muss die MCU über den Paralleladapter an die Amadeus Hardware angeschlossen werden und danach die* **HV-F.Restorer** *Schaltfläche betätigt werden.*

Es gibt keine Rückmeldung, aber wenn alles korrekt funktioniert hat, lässt sich die MCU nun wieder seriell programmieren.

3.4.7 Measure MCU Speed / MCU Geschwindigkeitsmessung

It's possible to measure the current MCU clock speed using the **Meas. Freq.** button. Therefore before clicking the **Meas. Freq.** button, a special software needs to be programmed into the AVR. You can find hex files for some AVRs in the Amadeus software folder AVR/AVRCalibration. If the correct hex file for you MCU is not present, you can adapt the given AVR-Studio project in the same folder. You only need to change the include filename and change the definitions (.equ) for the used pins.

Depending of the current AVR clock frequency the test takes form 1 second to 4 minutes, so please be patient. The accurateness is quite exceptable.

Es ist möglich die Geschwindigkeit des angeschlossenen AVR zu testen. Hierzu muss ein spezielles Programm in den AVR programmiert werden. Für einige AVRs findest du die entsprechende hex-Datei im Amadeus Software-Ordner unter AVR/AVRCalibration. Ist deine MCU nicht dabei, kannst du die hex-Datei selber erzeugen. Hierzu musst das AVR-Studio Projekt im gleichen Ordner öffnen und ein paar Anpassungen vornehmen, d.h. die include-Datei anpassen und die Portdefinitionen (.equ) anpassen. MISO ist der Pin, der wähend der Programmierung als Dateneingang arbeitet und SCK der Takteingang.

*Ist die hex-Datei programmiert, kannst du mit einem Klick auf **Meas. Freq.** den aktuellen Takt der MCU messen. Je nach Frequenz kann das 1 Sekunde bis 4 Minuten (bei einem Takt im kHz-Bereich) dauern.*

Tipp: Wenn die MCU-Software nicht stimmt, wird beim Messen 0.000kHz angezeigt.

3.4.8 Mini Prommer Window

See chapter 3.7.5 on page 35 for the informaiton.

Siehe Kapitel 3.7.5 auf Seite 35 für weitere Informationen.

3.4.9 Tracing/Tracen

Tip: In the Amadeus download area, there you can find an AVR project for AVR-Studio/GCC that demonstrates the tracing abilities.

Tipp: Im Amadeus Download-Bereich gibt es ein AVR GCC-Projekt, das die Möglichkeiten der Trace-Funktionen von Amadeus demonstriert.

3.5 PIC10F20x

The PIC10F20x are the smallest MCUs available today. They are offered with a SOT23-2 package. And have 3-4 I/O pins. Flash memory up to 512 instruction and about 1MIPS.

It is not possible to auto-detect the connected type. Therefore you have to choose it manually from a list. Take care that you select the correct type, otherwise the MCU might get corrupted.

There are **BackUp** types in the list. It is recommended to select the BackUp type first, read out the memory and store it in a Hex-File. Then switch back to the normal type. Don't use the BackUp type for programming, unless it is stated in this documentation.

Amadeus can program the program memory and the configuration bytes, as well as the factory calibration value for special purposes. ID-Location programming is not supported. There is no EEPROM.

Die PIC10F20x sind die derzeit wohl kleinsten Mikrokontroller. Sie werden im SOT23-2 Gehäuse geliefert. Sie verfügen über 3-4 I/O Pins. Der Flashspeicher reicht, je nach Typ, für bis zu 512 Befehle und sie arbeiten mit ca. 1MIPS.

Eine automatische Typenerkennung ist nicht möglich. Daher bitte den korrekten Type aus der eingeblendeten Liste auswählen. ***Die Auswahl eines falschen Typen kann den Chip unbrauchbar machen.***

In der Liste gibt es auch so genannte ***BackUp*** *Typen. Es wird empfohlen, diesen bei der ersten Inbetriebnahme des Mikrocontrollers auszuwählen, den Speicher auszulesen (Read) und als Hex-File wegzuspeichern. Danach den normalen Type auswählen. Nie mit dem BackUp Type programmieren, außer die Dokumentation weißt explizit darauf hin.*

Amadeus kann den Programmspeicher und die Konfigurationsdaten, sowie den Oscilatorkalibrierungswert schreiben. Die ID-Location wird nicht unterstützt. Ein EEPROM existiert nicht.

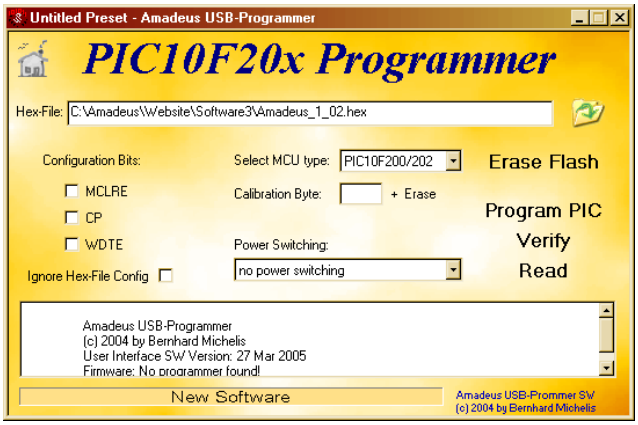

Illustration 12PIC10 User-Interface

3.5.1 How To Connect/Programmierschnittstelle

For electrical details see chapter “Electrical Characteristics/Elektrische Eigenschaften“ on page 47.

Für elektrische Details bitte im Kapitel “Electrical Characteristics/Elektrische Eigenschaften“ auf Seite 47 nachschlagen.

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

4 Leitungen (+Spannungsfreischaltung) werden für ICSP benötigt.

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit. *Beim Programmieren in der Schaltung **nicht anschließen**.*
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming. *Spannungsversorgung für Standalone Programmiergeräte. **Nie mit einer externen Spannung verbinden**.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	MCLR	Reset (about 12V during programming) *Reset kann wärend der Programmierung auf 12V gehen.*
PGD/PDO	ICSPDAT	Serial Data (GP0)
PGC/SCK	ICSPCLK	Serial Clock (GP1)

Table 5: PIC12F/16F ICSP adapter definition

3.5.2 Programming/Programmierung

Only additional information on programming are given in the paragraph. Have a look at the chapter “Basics/Grundlagen” on page 7 for the basic programming procedure.

After every programming the backup OSCCAL value is copied to the reset vector (last program memory location).

The **Erase Flash** button has a special function, when a **Calibration Byte** is given. In this case the factory default calibration value is replaced by this new value.

When, after replacing the factory calibration value, the MCU can be reprogrammed properly, but it does not run any more, you should try the following:

Die Grundlagen findest Du im Kapittel “Basics/Grundlagen“ auf Seite 7.

Nach jedem Programmiervorgang wird das Hersteller-Kalibrierungsbyte ans Ende des Programspeichers kopiert (Reset Vector).

*Die **Flash Erase** Schaltfläche hat eine besondere Bedeutung, wenn das **Calibration Byte** angegeben wird. In diesem Fall wird der Herstellerwert für OSCCAL durch den neuen ersetzt.*

Sollte es aus irgendeinem Grund an dieser Stelle dazu führen, dass sich der Mikrocontroller zwar mit dem Programmer weiterhin programmieren lässt, aber die Programme nachher nicht starten, müssen die Herstellerwerte wieder hergestellt werden.

3.5.3 Restoring Factory Calibration/Wiederherstellung der Herstellerwerte

In case of problems select the normal MCU type, enter the value 815 into the **Calibration Byte** field and click **Erase Flash**.

Next switch to BackUp MCU type and load the BackUp Hex-File you should have created at the beginning. And program it. Afterwards switch back to normal MCU type.

Im Fall von Problemen, den richtigen normalen Mikrocontroller auswählen, in das Feld **Calibration Byte** den Wert 815 eintragen und auf **Erase Flash** klicken.

Danach den BackUp Mikrokontrollertype auswählen, das BackUp Hex-File laden, das du zu Beginn anlegen solltest. Programmieren, und danach unbedingt wieder zum normalen Mikrokontrollertyp zurückschalten.

3.6 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.

Es gibt zwei Abgleichwerte, die immer wieder hergestellt werden müssen. Nämlich die Oszillatorkalibrierung (letzte Wort im Flashspeicher) und die BG-Kalibrierung (im ID-Wort).

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 6: 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.

*Wenn **Ignore Hex-File Configuration** markiert ist, werden die angezeigten Konfigurationsdaten für die Programmierung/Vergleich verwendet.*

3.6.1 How To Connect/Programmierschnittstelle

For electrical details see chapter “Electrical Characteristics/Elektrische Eigenschaften“ on page 47.

Für elektrische Details bitte im Kapitel “Electrical Characteristics/Elektrische Eigenschaften“ auf Seite 47 nachschlagen.

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

4 Leitungen (+Spannungsfreischaltung) werden für ICSP benötigt. ☐

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit. *Beim Programmieren in der Schaltung **nicht anschließen**.*
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming. *Spannungsversorgung für Standalone Programmiergeräte. **Nie mit einer externen Spannung verbinden**.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	MCLR	Reset (about 12V during programming) *Reset kann wärend der Programmierung auf 12V gehen.*
PGD/PDO	ICSPDAT	Serial Data
PGC/SCK	ICSPCLK	Serial Clock

Table 7: 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 10.

Laut Datenblatt muss die Betriebsspannung geschaltet werden, wenn der Reset-Pin deaktiviert wurde.

Versuche haben aber gezeigt, dass es offensichtlich auch ohne geht. Bitte selber ausprobieren.

3.6.2 Programming/Programmierung

Only additional information on programming are given in the paragraph. Have a look at the chapter “Basics/Grundlagen” on page 7 for the basic programming procedure.

Die Grundlagen findest Du im Kapitel “Basics/Grundlagen“ auf Seite 7.

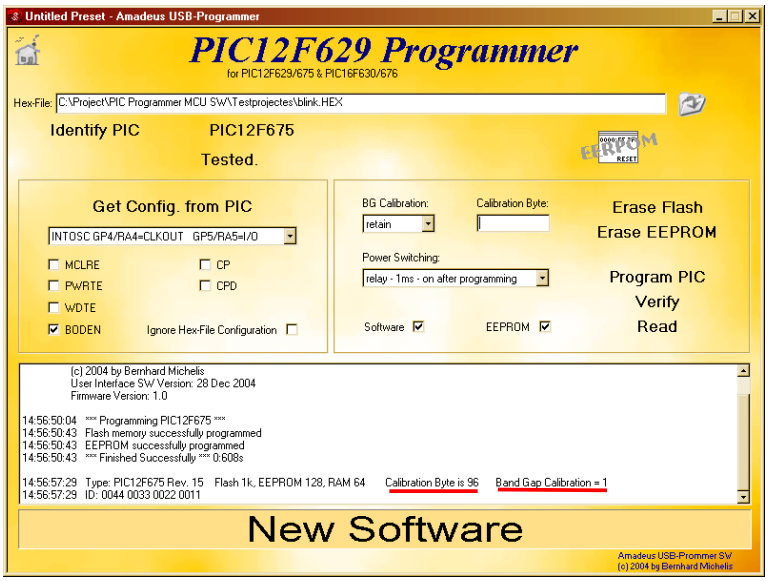

Illustration 13: PIC12F629 Programming Interface

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.

*Beim ersten erkennen des PICs mit Identify PIC sollte man sich das **Oscillator Calibration Byte** (0-255) und den **BG Calibration** Wert (0-3) aufschreiben, falls mal etwas schief geht.*

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.

*Wenn **Software** abgehakt ist, werden Flashspeicher, ID-Location und Konfigurationsdaten programmiert. Wenn **EEPROM** abgehakt ist, wird das EEPROM programmiert.*

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

***Erase Flash** löscht den Flash Speicher, die ID-Location und die Konfigurationsdaten. **Erase EEPROM** das EEPROM.*

3.6.3 Calibration Recovery/Kalibrationswiederherstellung

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.

*Wenn aus irgend einem Grund die Kalibrierung wieder hergestellt werden muss, muss der Oszillator Kalibrierungswert in **Calibration-Byte** geschrieben werden und der entsprechende Eintrag unter **BG-Calibration** ausgewählt werden. Danach **Erase Flash** anklicken. Wenn **BG-Calibration** **retain** anzeigt, werden keine Kalibrierungsdaten geschrieben.*

3.7 PIC18Fxxx(x)

Do not use this programming interface for PIC18FxxJxx types. J-types are 3V MCUs, not 5V!

A wide range of PIC18 MCUs is supported. They contain up to 128kBytes Flash memory, up to 4kBytes of RAM, up to 1024Byte EEPROM, an 8Byte ID-Location and 14 configuration bytes.

Some MCUs offer an internal clock. Externally, up to 40/48MHz (10/12MIPS) are possible or in HS-PLL mode up to 10/12MHz (10/12MIPS!!!, too). USB chips can run at 48MHz.

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

PIC18FJ sind 3V Typen und dürfen nicht mit diesem Interface (5V) programmiert werden.

Eine breite Palette von PIC18 Typen wird unterstützt. Siehe Kapitel "Supported Devices/Unterstützte Mikrocontroller" auf Seite 4.

Es gibt keine Kalibrierungswerte, die gesichert werden müssen.

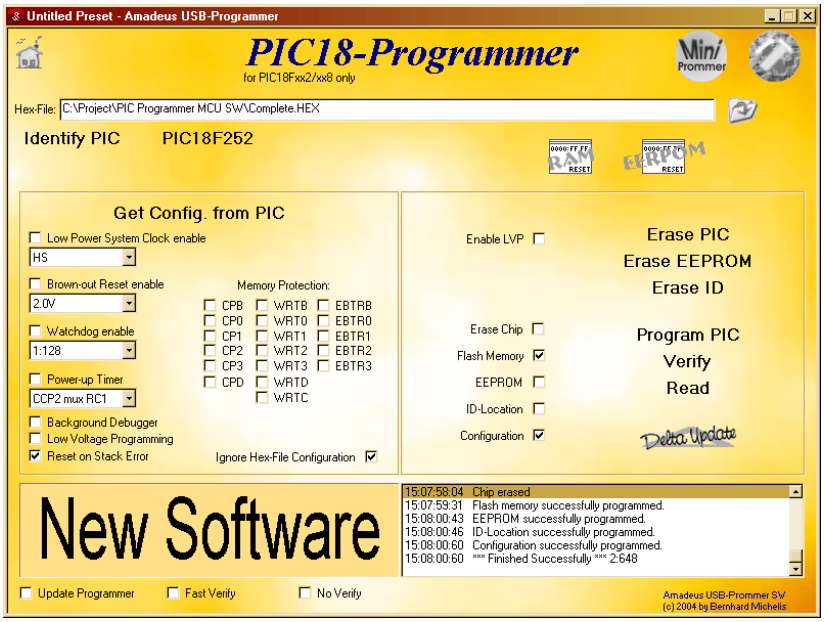

Illustration 14: PIC18F Programming Interface

3.7.1 How To Connect/Programmierschnittstelle

For electrical details see chapter "Electrical Characteristics/Elektrische Eigenschaften" on page 47.

Für elektrische Details bitte im Kapittel "Electrical Characteristics/Elektrische Eigenschaften" auf Seite 47 nachschlagen.

4/5 lines are required for programming. *4/5 Leitungen werden benötigt.*

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit. *Beim Programmieren in der Schaltung **nicht anschließen**.*
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming. *Spannungsversorgung für Standalone Programmiergeräte. **Nie mit einer externen Spannung verbinden**.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	MCLR	Reset (about 12V during programming). LVP must not me set. *Reset kann wärend der Programmierung auf 12V gehen. LVP darf nicht gesetzt sein.*
PGD/PDO	PGD	Serial Data
PGC/SCK	PDC	Serial Clock
PGM/PDI	PGM	Only required for Low Voltage Programming. Enable LVP must be checked. *Nur erforderlich für Low Voltage Programmierung. LVP muss gesetzt sein.*

Table 8: PIC18 ICSP adapter definition

3.7.2 Programming/Programmierung

The PIC18F 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.

Die PIC18F Familie bietet viele Updatemöglichkeiten. Um die Programmierung zu beschleunigen, immer Fast Verify setzen. Unterschiedliche Programmfragmente sollten bei 0x0000, 0x2000, 0x4000, 0x6000 (nur xxx und xx20 Typen) beginnen. Darüber hinaus sollte man Erase Chip deselektieren und alle Programmierschritte entfernen, die nicht benötigt werden.

- Always select **Fast Verify**.
- The xxx and xx20 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
Programming and verifying 64kBytes Flash (full)	4.7s
Programming and verifying 128kBytes Flash (full)	7s
EEPROM programming and verifying 256Bytes (full)	1s

Table 9: PIC18Fxx2/xx8 programming durations

3.7.3 Delta Update

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 from last programming. If you restart Amadeus, the data history is all 0xFF.

*Für superschnelle Updates gibt es die **Delta Update** Funktion. Hier kann man kleine Programmänderungen in 200ms aktuallisieren. Das Funktioniert aber nur, wenn sich nicht alle nachfolgenden Instruktionen verschieben. Deshalb sollte man den verschiedenen Programmteilen (ISR, Main-Loop, Funktionen für dieses und jenes) unterschiedliche, fixe Adressen zuweisen und zwischen ihnen etwas Platz für Erweiterungen lassen.*

Bevor Du mit Delta Update arbeiteten kannst, muss die MCU einmal normal programmiert werden. Diese gilt auch, wenn Amadeus neu gestartet wurde.

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

*Wenn der Programmer den PIC nicht finden kann, sollte man prüfen, ob **Enable LVP** richtig gesetzt ist. Für High-Voltage Programmierung LVP löschen.*

3.7.4 PIC18 below 4.5V/ PIC18 an Betriebsspannungen unter 4,5V

When the supply voltage of the target PIC18 is below 4.5V, it is no longer possible to *bulk-erase* the chip. To reprogram the MCU anyway, you have to activate **Low Voltage Erase** and deactivate **Chip Erase**. Then the memory is erased row by row, what takes a few seconds. Keep in mind, that the Code-Protection can only be remove by a bulk-erase at at least 4.5V!

*Wenn die Betriebsspannung des Zielmikrokontrollers unter 4,5V liegt, kann kein BulkErase (Löschen in einem Rutsch) mehr durchgeführt werden. Um trotzdem weiterhin Programmieren zu können, ist es notwendig den Speicher Reihenweise (row-by-row) zu löschen. Hierzu **LowVoltageErase** aktivieren und **ChipErase** deselektieren (gilt auch für **Erase PIC** und **Erase EEPROM**). Der Löschvorgang dauert nun allerdings einige Sekunden. Bitte unbedingt beachten, dass ein aktivierter Schreib-/Leseschutz (Code Protection) nur mit einem BulkErase entfernt werden kann!*

3.7.5 Mini Prommer Window

Have a look at the *Mini Prommer* Window. It stays always on top of your IDE, so you can quickly start the Flash update with one click.

Das Mini Prommer Fenster ist ein Fenster, das immer im Vordergrund bleibt. So kann man mit nur einem Klick sein Update durchführen.

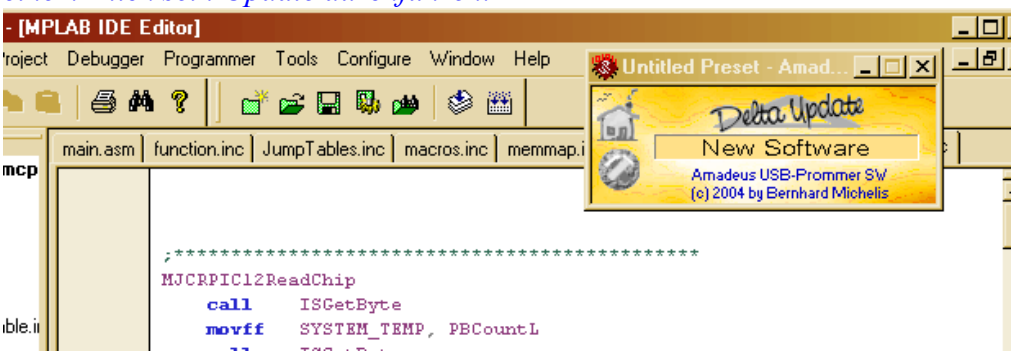

Illustration 15: MP-Lab with Mini-Prommer window in front of it

3.7.6 PIC18F6x10, PIC18F6x90, PIC18F8x10, PIC18F8x90

According to the programming specification, this chip need a 12V power-supply which can supply up to 100mA. Amadeus' internal 12V generation can supply 3mA max. However, during development it seemed that the 100mA in the programming specification might be incorrect. A PIC18F8490 was successfully programmed using the internal 12V supply.

Furthermore this type of MCU has a slightly different flash concept. Always select **Erase Chip**, and do *not* activate **Low-Voltage-Erase**, when the supply voltage is below 4.5V. Don't use **Delta Update**. The bulk erase should work down to 2,75V. See documentation for further detail.

Laut Programmierspezifikation benötigen diese Chips eine 12V Spannungsquelle, die bis zu 100mA Strom liefern kann. Amadeus' interne 12V Spannungserzeugung kann aber nur max. 3mA liefern. Wie auch immer, während der Entwicklung war es problemlos möglich einen PIC18F8490 ganz normal mit der internen 12V Spannung zu programmieren.

Desweiteren gelten einige Besonderheiten. Aktiviere immer Erase Chip, aktiviere nie Low-VoltageProgramming, auch nicht, wenn die Betriebsspannung unter 4,5V liegt. Verzichte auf Delta-Update. Das Chip Erase sollte runter bis 2,75V funktionnieren. Siehe Programmier-Spezification für weitere Details.

3.8 PIC18FJ

PIC18FxxJxx are 3V MCUs and are completely different from PIC18F-types from the programmers point of view. Thus the J-types have their own programming algorithems. Amadeus uses the UPAL interface to support PIC18FJ. For the pin layout for a programming adaptor (5V-3V-levelshifter) and user instructions refer to chapter 3.10 PIC24FJ/PIC24HJ/dsPIC33 on page 38, with one exception. Configuration settings **must** be included in the hex file, because currently there is no way to edit configuration setting from Amadeus' user interface.

Please consider, that the configuration data are stored in the last 8 bytes of flash memory in a hex-file for PIC18FJ. This has the effect, that hex files for PIC18FJ with 64kByte flash memory will not run on 128kByte flash memory devices, because the last 8 bytes of a 64kB hex file are lying somewhere in the middle of the 128kBytes memory and thus are not interpreted to be configuration settings. In such cases you should work on that hex file in MP-Lab.

It's not possible to verify PIC18F97J60 devices, when code protection is enabled.

PIC18FxxJxx und PIC18F (ohne 'J') dürfen nicht verwechselt werden. Aus Sicht des Programmiergerätes werden sie vollkommen unterschiedlich gehandhabt. Für die J-Typen ist eine 5V-3V-Pegelanpassung erforderlich. Bezüglich Anschlussbelegung des Programmier-adapters und Bedienung sind die J-Typen mit den PIC24/dsPIC33 identisch (Anleitung in Kapitel 3.10 „PIC24FJ/PIC24HJ/dsPIC33“ auf Seite 38 folgen). Eine Ausnahme gibt es aber. Die Konfigurationseinstellungen (Configuration-Bytes) ***müssen zwingend im Hex-File enthalten sein****, da zur Zeit keine Möglcihkeit besteht, mit Amadeus Konfigurationseinstellungen für PIC18FJ zu editieren. Am besten gleich im Projekt definieren und mit MPLab speichern.*

Bitte beachten, dass bei PIC18FJ Mikrocontrollern die Konfigurationseinstellungen in den letzten 8 Bytes des Flash Speichers abgelegt werden. Will man beispielsweise eine hex-Datei eines 64kByte Bausteins in ein 128kByte Baustein programmieren, werden die Konfigurationsdaten irgendwo in der Mitte des Flashspeichers abgelegt und somit nicht als Konfigurationsdaten interpretiert. In solchen Fällen muss vorher in MP-Lab die hex-Datei überarbeitet werden.

Bei der PIC18F97J60 Familie ist kein Vergleichen möglich, wenn Code Protection aktiviert ist.

3.9 dsPIC30

dsPIC30 are 16bit DSP-MCUs and can work up to 120MHz (30MIPS). There are no calibration bytes. IDLocation programming is not supported.

dsPIC30 sind 16Bit MCUs mit DSP-Funktionalität, die mit bis zu 120MHz (30MIPS) getaktet werden können. Es gibt keine Kalibrierungswerte, die gesichert werden müssen. IDLocation Programmierung ist nicht notwendig und wird nicht unterstützt.

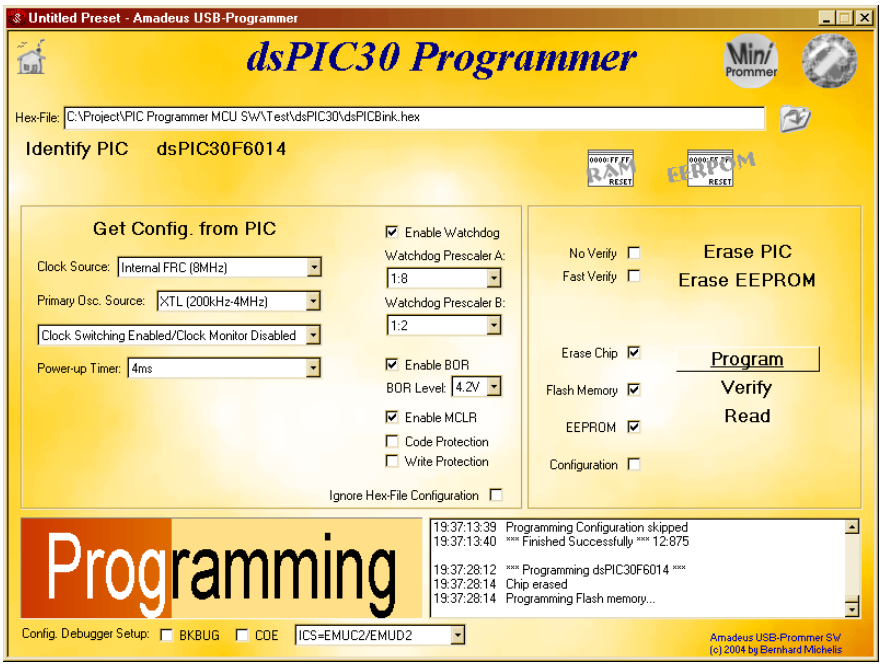

Illustration 16: dsPIC30 UI

3.9.1 How To Connect/Programmierschnittstelle

For electrical details see chapter “Electrical Characteristics/Elektrische Eigenschaften“ on page 47.

Für elektrische Details bitte im Kapittel “Electrical Characteristics/Elektrische Eigenschaften“ auf Seite 47 nachschlagen.

4 lines are required for programming. *4 Leitungen werden benötigt.*

Amadeus side	*MCU side*	*comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit. *Beim Programmieren in der Schaltung **nicht anschließen**.*
EXTVCC !!!	VCC	**For stand allone programming only!!!** Do never connect EXTVCC for in-circuit programming. *Spannungsversorgung für Standalone Programmiergeräte. **Nie mit einer externen Spannung verbinden**.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	MCLR	Reset (about 12V during programming). *Reset kann wärend der Programmierung auf 12V gehen.*
PGD/PDO	PGD	Serial Data
PGC/SCK	PDC	Serial Clock

Table 10: dsPIC30 ICSP adapter definition

3.9.2 Programming/Programmierung

See chapter "Basics/Grundlagen" on page 7 for a general description. When the supply voltage of the dsPIC is below 4.5V the check mark next to **Erase Chip** has to be removed. Then the bulkerase is replaced by a rowerase. This is also true for the Erase PIC function. Keep in mind, that codeprotection can only be removed by bulk-erasing the MCU at at lease 4.5V supply voltage!

When EraseChip is selected, the programm memory and the EEPROM will be erased, even if it is deselected.

The configuration will never be erased. Not even with a chiperase (bulkerase). It can only be reprogrammed.

*Siehe Kapitel "Basics/Grundlagen" auf Seite 7 für eine allgemeine Beschreibung. Wenn die Betriebsspannung unter 4,5V liegt, muss **EraseChip** (BulkErase) deaktiviert werden. Jetzt wird der Speicher reihenweise gelöscht (row-erase). Das dauert etwas länger, funktioniert aber auch unterhalb von 4,5V. Die Einstellung gilt auch für die Erase PIC und Erase EEPROM Funktion. Bitte beachten, dass zum Löschen des Schreib-/Leseschutzes (CodeProtection) ein BulkErase erfoderlich ist, der nur bei mindestens 4,5V möglich ist.*

Beachte, dass wenn EraseChip ausgewählt ist, sowohl der Flash-Speicher als auch das EEPROM gelöscht werden, auch wenn es nicht auswählt ist.

Die Konfigurationeinstellungen werden nie gelöscht. Sie können nur überschrieben werden.

3.9.3 Further Features/Weitere Fähigkeiten

Amadeus supports the **RAM-ReadReset** for dsPIC30. See the chapters "Debugging Features/ Fehlersuch Möglichkeiten" on page 11 and **Delta Update** chapter "Delta Update" on page 34. Furthermore see the "Mini Prommer Window" on page 35.

*Amadeus unterstützt das lesen des RAMs via **RAMReadReset**. Siehe Kapitel "Debugging Features/Fehlersuch Möglichkeiten" auf Seite 11. Darüber hinaus ist auch ein **DeltaUpdate** möglich. Siehe "Delta Update" auf Seite 34. Auch ein "Mini Prommer Window" (Seite 35) steht zur Verfügung.*

3.10 PIC24FJ/PIC24HJ/dsPIC33

For Amadeus PIC24FJ/HJ and dsPIC33 are identical. Therefore they are described here. As user interface you have to choose the UPAL interface. For more details about UPAL see chapter 4 "UPAL" on page 42.

Picure 17 shows the userinterface for PIC24/dsPIC33. The function Show Configuration reads out the connected PIC and shows its configuration settings as clear text.

PIC24FJ/HJ und dsPIC33 sind aus Programmersicht identisch und werden hier gemeinsam behandelt. Es sei angemerkt, dass es verschiedene Typen gibt. Derzeit werden nur PIC24HJ und dsPIC33FJ entsprechend der Auflistung unterstützt.

Als Benutzeroberfläche wird die UPAL Oberfläche verwendet. Siehe hierzu in jedem Fall auch das Kapitel 4 „UPAL" auf Seite 42.

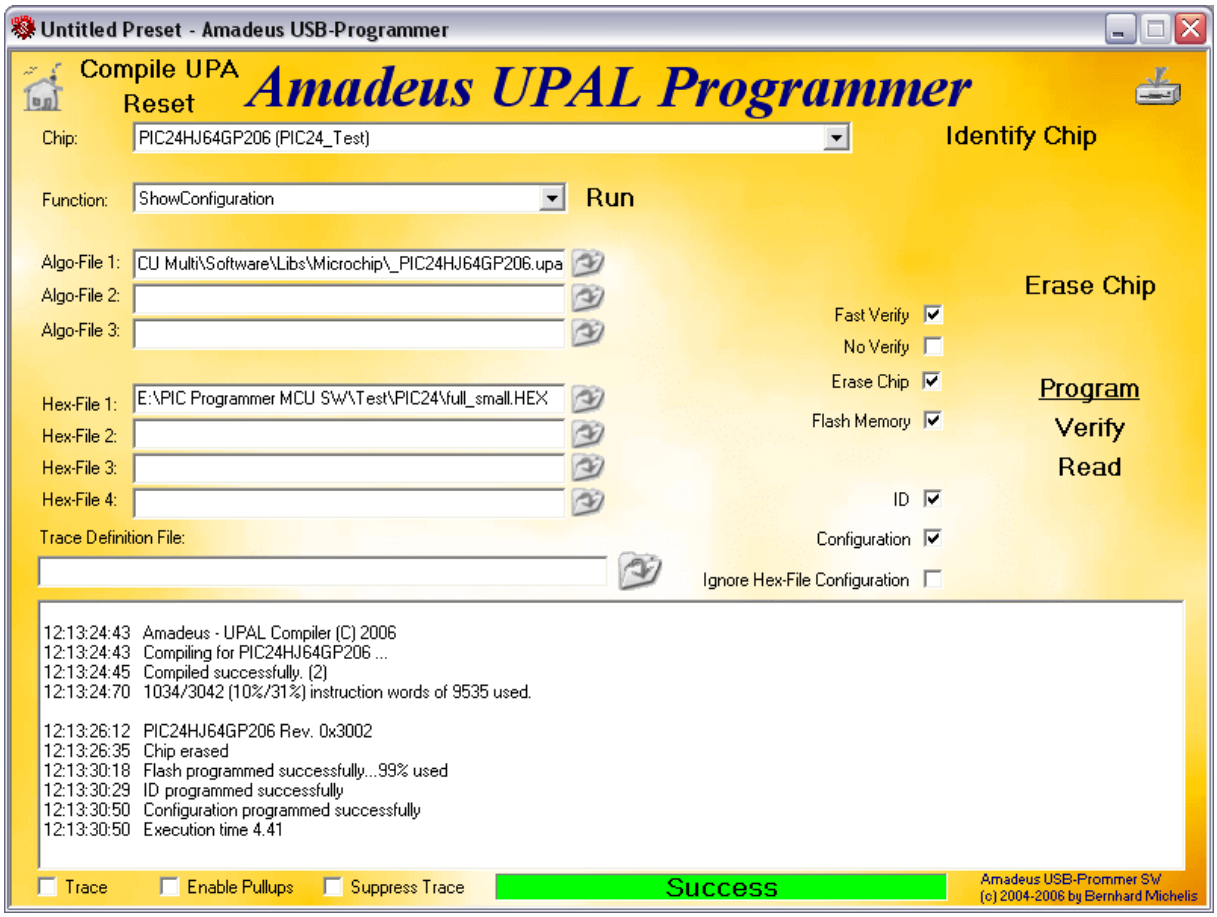

Illustration 17: Programming PIC24HJ64GP206

In Bild 17 sehen wir die Benutzeroberfläche für PIC24/dsPIC33. Die Funktion Show Configuration ließt den angeschlossenen PIC24/dsPIC33 aus und zeigt dessen aktuelle Konfiguration in Klartext an.

3.10.1 How To Connect/Programmierschnittstelle

To connect Amadeus to a PIC24/dsPIC33 you need to consider, that Amadeus is working 5V while the PIC24/dsPIC33 are 3V types. Therefore it might be mandatory to use a 5V/3V levelshifter as describt in chapter 5.1.2 “5V/3V Level-Shifter/Pegel-Wandler“ or something similar.

For electrical details see chapter “Electrical Characteristics/Elektrische Eigenschaften“ on page 47.

Um Amadeus mit einem PIC24/dsPIC33 zu verbinden, ist zu beachten, dass Amadeus mit ca. 5V Pegel arbeitet, während PIC24/dsPIC33 nur mit ca. 3V betrieben werden. Somit muss unbedingt der in Kapitel 5.1.2 „5V/3V Level-Shifter/Pegel-Wandler“ gezeigte Pegel-Wandler oder ein Verbindung mit vergleichbaren Eigenschaften zur Verbindung verwendet werden.

Für elektrische Details bitte im Kapitel “Electrical Characteristics/Elektrische Eigenschaften“ auf Seite 47 nachschlagen.

5/4 lines are required for ICSP.

5/4 Leitungen werden für ICSP benötigt.

Amadeus side	*MCU side*	*Comment*
n.c.	VCC	**For in-circuit programming**, the MCU gets its power from the target circuit. *Beim Programmieren in der Schaltung **nicht anschließen**.*
EXTVCC !!!	n.c.	You can use EXTVCC to generate a reference voltage for the 5V/3V level-shifter. **After first usage in UPAL interface, EXTVCC stays active.** *Der EXTVCC Ausgang liefert eine Spannung von 5V, **die nach der ersten Programmeraktivität dauerhaft im UPAL Modus anliegt.** Diese Spannung kann als Versogungsspannung für die Referenzspannungs-erzeugung für den 5V/3V Pegelwandler verwendet werden.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	MCLR	Reset
PGD/PDO	PGD	Serial Data
PGC/SCK	PDC	Serial Clock

Table 11: PIC24/dsPIC303ICSP adapter definition. ***Please consider the 5V/3V problem (Chapter 5.1.2)!***

3.11 PIC32

For basic information about PIC32 programming please read chapter 4 “UPAL“ and the chapter about PIC24/dsPIC33 programming. In the following only PIC32 specific information is provided. PIC32 programming uses the **A-UPA** user interface of Amadeus.

Dieses Kapitel beschreibt nur die PIC32 spezifischen Besonderheiten. Wer noch nie mit der Amadeus-UPA Benutzeroberfläche gearbeitet hat sollte sich unbedingt das Kapitel 4 „UPAL“ und das Kapitel über PIC24/dsPIC33 durchlesen, um die Grundbedienung zu erlernen.

3.11.1 Setup/Einstellungen und Vorbereitungen

The programming algorithem for PIC32 only supports basic programming functions as well as a flash cycle counter if desired.

Der PIC32 Programmieralgorithmus unterstützt die Basisfunktionen Programmieren, Vergleichen und Chip auslesen. Darüber hinaus ist ein Programmierzyklenzähler enthalten, der allerdings expliziet eingestellt werden muß.

3.11.2 How To Connect/Die Programmierschnittstelle

There are two ways to program the PIC32. ICSP (2-wire) and JTAG (4-wire). For connection details and about 5V-to 3.3V adaption and ICSP (programming via PGCx/PGDx) see chapter 3.10.1. For the connection for JTAG (TMS, TDI, TDO, TCK) see the diagram below. As the programming pins are 5V tolerant, I personally connected it 1-to-1 without any levelshifter, which works fine for me. **Be careful not to select a wrong programming algorithm, as it might put 12V on the Reset line.** But if yo want a save way, you might want to use any 5V-to-3.3V levelshifter or the one presented in chapter 5.1.2.

Es gibt zwei Wege den PIC32 zu programmieren. Zum einen über die ICSP (2-Draht) Schnittstelle oder über JTAG (4-Draht). Die Anschlußbelegung für ICSP (PGCx/PGDx) ist in Kapitel 3.10.1 zu sehen. Der Anschluß über JTAG (TMS, TDI, TDO, TCK) ist weiter unten beschreiben. Ein Pegelwandler/Begrenzer wie in Kapitel 5.1.2 dargestellt kann zur Sicherheit benutzt werden. Da die Programmierpins allerdings 5V tolerant sind, verwende ich eine einfache 1-zu-1 Verbindung ohne Pegelwandler. ***Es sollte allerdings bedacht werden, daß bei Auswahl eines falschen Programmieralgorithmuses 12V auf die Reset-Leitung gelangen können!***

Amadeus side	*MCU side*	*Comment*
EXTVCC !!!	**not connected** ***nicht angeschlossen***	You can use EXTVCC to generate a reference voltage for the 5V/3V level-shifter. **After first usage in UPAL interface, EXTVCC stays active.** *Der EXTVCC Ausgang liefert eine Spannung von ca. 5V,* ***die nach der ersten Programmeraktivität dauerhaft im UPAL Modus anliegt.*** *Diese Spannung kann als Versogungsspannung für die Referenzspannungs-erzeugung für den 5V/3V Pegelwandler verwendet werden.*
GND	GND	Connecting one GND line might be enough. Connect both, if you are using longer programming cables. *Masseverbindung.*
MCLR/RST	MCLR	Reset
PGD/PDO	TDI	
PGC/SCK	TCK	
PGM/PDI	TMS	
SDA	TDO	This pin belongs to **N2**! *Dieser Pin ist an Stecker* ***N2*** *zu finden!*

Table 12: PIC32 JTAG adapter definition.

3.11.3 Program Verify and Read/Programmieren Vergleichen und Lesen

Choose the hex-file to be programmed to **Hex-File 1**. Because Amadeus doesn't offer editing configuration settings, these must be included in the hex-file. So please use MPLab to include configuration data in your hex-file.

Das zu programmierende hex-File muß unter ***Hex-File 1*** *ausgewählt werden. Da Amadeus keinen Konfigurationseditor für PIC32 enthält, müssen die Konfigurationsdaten mit Hilfe von MPLab mit in die Hex-Datei geschrieben werden. Dieses sollte ohnehin der Normalfall sein.*

Whenever you programm the PIC32, the complete chip will be erased in advance then only areas other than 0xFF will be programmed to speed up programming. After programming the program will verified. Therefore 3 versions exist. If you choose **No Verify** (check box) the program gets verified by calculating and comparing a checksum. This is very fast and should be okay for normal use. The second variant is to uncheck **No Verify** but check **Fast Verify**. Now only programmed areas (not completely 0xFF) will be read back and fully verified. The last version reads back the complete memory from the chip and compares it to the programming hex-file. My suggestion is to use variant 2 to be 99.999 % sure or variant 1 to be 99.9 % sure that everything is okay. Variant 3 is only recommanded for final programming when you have finished your development.

*Wird mit **Program** die Programmierung gestartet, wird der Chip zuerst gelöscht. Danach wird der Chip blockweise programmiert, wobei nur Blöcke geschrieben werden, welche vom Leerzustand abweichende Daten enthalten. Anschließend wird der Speicherinhalt mit der zu programmierenden hex-Datei verglichen. Hierbei exestieren drei Varianten. Ist **No Verify** ausgewählt, wird der Chip Speicher nur anhand eines Checksummenvergleichs geprüft. Dieses geht extrem schnell, erfaßt den ganzen Speicher und ist relativ sicher, was die Fehlererkennung angeht. Ist **No Verify** nicht ausgewählt, werden Daten aus dem Chip zurückgelesen und byteweise mit den hex-File Daten verglichen. Dabei gibt es zwei Varianten. Ist **Fast Verify** ausgewählt, werden nur die porgrammierten Bereiche gelesen und verglichen. Ist **Fast Verify** nicht ausgewählt wird der komplette Chip ausgelesen und verglichen. Die 2. Variante sollte in der Regel eine fast 100 %-ige Sicherheit geben und ist relativ schnell. Die Variante 1 sollte in der Regel eine 99,9 %-ige Sicherheit geben und ist besonders schnell. Die Variante 3 sollte 100 % sicher sein, braucht aber je nach Chip sehr lange. Ich würde sie nur zum finalen Programmieren verwenden, wenn die Entwicklung abgeschlossen ist oder wenn der Verdacht besteht, daß der Flashspeicher defekt sein könnte.*

If code protection is enabled, only **Identify Chip** and **Erase Chip** are functional. That means that if you want to reprogram a code protected chip, you need to manually erase it in advance.

*Ist der Chip gegen Auslesen geschützt (Code Protection), funktionieren nur **Identify Chip** und **Erase Chip**. Das bedeutet, daß vor dem Umprogrammiere von geschützten ICs im vorhinein extra **Erase Chip** verwendet werden muß, um den Chip wieder ansprechen zu können.*

3.11.4 Flash-Cycle-Counter/Programmierzykluszähler

To use the *flash-cycle-counter* you need to create an empty .hex file. E.g. create a new text file and rename it to mycounter.hex or whatever you like. Select the mycounter.hex file as **Hex-File 3**. Then choose **ResetFlashCounter** from **Function** drop-down menu and click **Run**. To activate/deactivate the counter function open the PIC32.upa file (CTRL + left mouse button if you in advance set up Windows to open .upa files in your favorite text editor) and set the counter value to 1 for counting and to 0 for not counting.

*Um den oben erwähnten Programmierzyklenzähler zu verwenden wird eine hex-Datei als Zählerdatei verwendet. Die Datei muß als Hex-File 3 ausgewählt werden. Um die Datei anzulegen, kann einfach eine neue txt-Datei angelegt werden, welche anschließend in MeinZähler.hex (der Dateiname ist frei wählbar) umbenannt wird. Zur Initialisierung der Zählerdatei unter **Functions** die **ResetFlashCounter** Funktion auswählen und **Run** anklicken. Um das Zyklenzälen zu aktivieren oder zu deaktivieren muß in der Algorithmusdatei der Wert `counter` editiert werden. Hierzu mit STRG drücken und mit der linken Maustaste auf das Ordnersymbol neben **Algo-File 1** klicken (funktionniert nur, wenn unter Windows Dateien mit der Endung **upa** mit einem Texteditor verknüpft sind!). `counter=1` aktiviert den Zähler und `counter=0` deaktiviert den Zähler.*

4 UPAL

UPA is designed to handle the programming of microcontrollers and memory ICs that use synchronous serial interfaces to get programmed, like e.g. PIC and AVR microcontrollers as well as I²C ICs.

What is the difference between the build in programming algorithms and the Universal Programming Algorithm module? To be a little bit more precise, it is not a universal

algorithm, but a Univeral Programming Algorithm Language. That means, it is possible for everybody to extend available algorithems or write new ones.

So called upa files are located next to the Amadeus executable file or in it's subdirectories.

UPA ist dafür ausgelegt Mikrokontroller und Speicher-ICs mit seriellem, synchronem Interface zu programmieren, wie es zum Beispiel bei PIC und AVR Controllern der Fall ist. Ebenso für I²C Bus ICs.

Was ist der Unterschied zwischen den eingebauten Programmieralgorithmen und Universal-Programmier-Algorithmen? Nun ja, genau genommen ist UPAL eine universelle Programmiersprache zum programmieren von Programmieralgorithmen. Die Algorithmen sind in Klartext geschrieben und können von jedermann editiert und erweitert werden. Auch ist es natürlich möglich, sich eigene Algorithmen für neue Controller zu schreiben.

So genannte UPA Dateien sind neben der Amadeus.exe oder deren Unterordnern gespeichert.

How to use predefined upa files to program a microcontroller? Just go to the UPA user interface module and select a chip from the Chip drop down box. Amadeus then automatically sets up the programmer for the selected target chip. The corresponding algorithm file is set accordingly. When you want to display the algorithm file's content just click the folder symbol next to it while holding down the CTRL key (it is important, that you once set a text editor for the .upa extension. You can do so by double-clicking on a upa file and then say, *"use always the selected program to open this document type."*

Wie verwende ich vorgefertigte UPA Dateien um einen Mikrocontroller zu programmieren? Hierzu einfach das UPA Modul in der Amadeus Software auswählen. Dann aus der Chip-Auswahl Box den Controller oder das Speicher-IC auswählen und los geht's.

*Die entsprechende Algorithmusdatei wird automatisch ausgewählt und geladen. Die Benutzeroberfläche passt sich eventuell noch an. Eine Hex-Datei (standardmäßig **Hex-File 1**) auswählen und den Programmierknopf drücken. Hält man beim auswählen der Dateien die CTRL-Taste gedrückt, wird die vorhandene Datei geöffnet (Voraussetzung: Unter Windows ist der Dateityp [*.upa, *.lib] mit einem Text-Editor verknüpft.), anstatt das eine Dateiauswahl eingeblendet wird.*

It's obvious that it is a little bit difficult to handle all kinds of chips with only one user interface. Therefore Amadeus predefines a set of functions, usually used during programming, like **Erase**, **Program**, **Read**, **Verify** as well as a possibility to set up steps of these algorithms in more detail.

Es ist offensichtlich, dass es nicht ganz so einfach ist eine Vielzahl von Mikrocontrollern oder Speicherbausteinen mit nur einer Benutzeroberfläche zu programmieren. Um es trotzdem leicht zu machen, definiert Amadeus eine Reihe von Standardfunktionen wie zum Beispiel Erase, Program, Read, Verify mit der Möglichkeit diesen Optionen zuzuordnen.

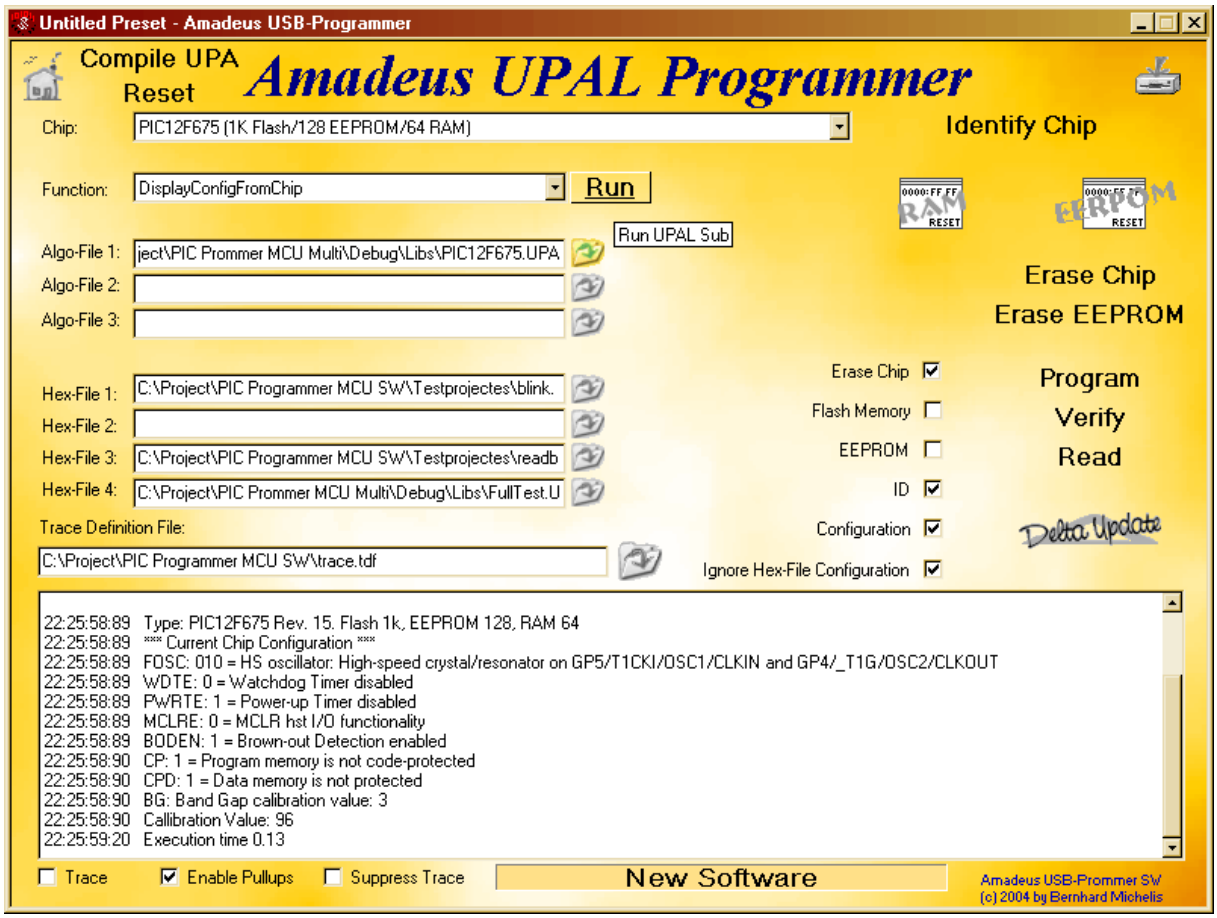

Illustration 18: UPA user interface

Select a hex file by clicking on the folder symbol next to the hex file path field and click on **Program** and you are done.

*Wähle die Hex-Datei mit einem Klick auf das Ordnersymbol und klicke danach auf die Schaltfläche **Program**.*

As you see, there is no great difference to the other user interfaces of Amadeus. Okay, there is one. The configuration editor is not there, but .upa files might contain a configuration information function, that displays configuration information as text in clear as you can see in Illustration 18.

Wie man sieht, ist der Unterschied zu den speziellen Benutzeroberflächen von Amadeus, mal abgesehen vom Konfigurationseditor, nicht viel anders. Um trotzdem nicht ganz auf Konfigurationen verzichten zu müssen, können upa-Dateien eine Funktion zum Anzeigen von Konfigurationsdaten im Klartext bieten. Siehe hierzu Bild 18.

If configuration data are already included inside the hex file, everything stays the same as with the other Amadeus user interfaces. If you want to define the configuration bits manually, you can open the **Algorithm-File** in a text editor (Illustration 19). Therefore you have to set up window, that a double click on a .upa file open the right text editor (see above). If this is done once, hold down CTRL-key and click on the folder symbol to open the editor.

Sofern die Konfigurationsdaten bereits in der Hex-Datei enthalten sind, können diese genutzt werden. Um Konfigurationsdaten allerdings manuell zu konfigurieren, muss die

Algorithmusdatei (.upa) geöffnet werden. Sofern für upa-Dateien bereits ein Editor eingestellt ist, die Strg-Taste drücken und mit der Maus auf das Ordner Symbol klicken.

Everything after a semicolon is a remark and tells you what is going on. For the given example (PIC12F675) only the yellow highlighted lines are important.

Just copy the given constant strings behind the `_config` variable concatenating them by an & sign and *save* the .upa file again. When programming the next time with the hook next to **Ignore Config. form Hex** the upa-files configuration is used.

Alles was nach einem Semikolon ';' folgt, ist ein Kommentar und dient Dir als Information. Im Beispiel sind zum Beispiel nur die gelb hervorgehobenen Zeilen interessant.

Bei PIC-Mikrocontrollern werden zum Beispiel gerne so genannten Konfigurationsvariablen (hier: _config) entsprechende Konstanten zugewiesenen, die im Beispiel mit einem Und-Symbol '&' verknüpft werden müssen.

*Da wir die Konfigurationsdateien direkt in der Algorithmus Datei geändert haben, müssen wir den Algorithmus erneut kompilieren. Handelt es sich um die Hauptalgorithmusdatei (Algo-File 1), geschieht dieses automatisch, sobald wir sie speichern. Handelt es sich um eine weitere Datei, müssen wir die **Compile UPA** Schaltfläche anklicken um unsere Änderungen zu übernehmen.*

To verify your configuration, use the above mentioned **DisplayConfigFromChip** function. Then you can be sure, that configuration is as it should be.

In the given example you can also assign a new value to **`_callibrationValue`** to change the calibration value and to **`_BGCallibration`** to change or reset the Band Gap calibration.

I think this is an easy and safe way to handle configuration of an unlimited amount of chips types.

```
; *** configuration word ***

_config = _HS_OSC & _MCLRE_OFF & _WDT_OFF

; copy the desired flags behind the '=' and separat them using a '&' (binary AND) sign
; use the constants defined in microchips *.inc files
; _CPD_ON,       _CPD_OFF,
; _CP_ON,        _CP_OFF,
; _BODEN_ON,     _BODEN_OFF,
; _MCLRE_ON,     _MCLRE_OFF,
; _PWRTE_OFF,    _PWRTE_ON,
; _WDT_ON,       _WDT_OFF,
; _LP_OSC, _XT_OSC, _HS_OSC, _EC_OSC, _INTRC_OSC_NOCLKOUT, _INTRC_OSC_CLKOUT,
;   _EXTRC_OSC_NOCLKOUT, _EXTRC_OSC_CLKOUT

; *** set callibration value ***

_callibrationValue = -1          ; set to -1 to preserve in chip callibration value
_BGCallibrationValue = -1        ; set to -1 to preserve BG callibration from chip
```

Illustration 19: Head of UPA source code

What else should you know? Well, there is a **Function** drop-down field, containing all available functions in the upa file. This way you can start every function and not only the predefined one's. Lock out for functions containing the word **Info**. They might contain more detailed information about the implementation of the algorithm.

Why there are three Algo-Files and four Hex-Files?

The Algo-Files get loaded one after another. Normally one file is enough, but it is possible to overwrite some function or variables with other values, or other function content. So functions (subs) and constants in Algo-File 2 overwrite functions of Algo-File 1 and Algo-File 3 overwrites functions and constants of AlgoFiles 2 and 1. So you can also write the above mentioned configuration (Illustration 19) to a separate file and select it as Algo-File 2. This way you can have several predefined configurations, you just need to choose from.

Or you can add your own new functions (see separate Amadeus-UPAL Documentation on how to write your own programming-algorithms) in a separate file based on Algo-File 1. Whenever there might be an update for the algorithm, you have your extensions in a separate file.

Normally only Hex-File 1 is used, but the algorithm might use Hex-File 2 as EEPROM data source and Hex-File 3 as Configuration source, or you might want to user Hex-File 4 as Image and Sound source for your own applications. So you can merge more the one hex file in upa (merging needs to be programmed (your own additions) or offered by the algorithm).

That's it.

The buttons **Compile** and **Reset** are not needed during normal operation. They are intended to be for algorithm developers. **Reset** resets the Amadeus' hardware MCU and **Compile** re-triggers the automatic compilation of the upa file. In normal use, Amadeus updates itself as soon as a new Algorithm file is selected or as soon as a selected algorithm-file is saved.

*Was sollte man sonst noch über die UPA Benutzeroberfläche wissen? Es gibt ein **Function** Auswahlmenü, dass alle zur Verfügung stehenden Funktionen anzeigt. Mit **Run** können diese ausgeführt werden.*

*Warum kann man bis zu drei Algorithmen auswählen? Normalerweise reicht es eine Datei auszuwählen. Es ist aber möglich weitere Dateien anzugeben, die dann z.B. Funktionen oder Konstanten der ersten Algorithmusdateien überschreiben. So ist es z.B. möglich, seine eigenen Konfigurationsdaten in einer separaten Datei abzulegen und bei Bedarf die entsprechende Konfiguration zu laden. Nach dem Laden muss mit der **Compile UPA** Schaltfläche der Algorithmus aktualisiert werden.*

Warum gibt es mehrere Hex-Dateien? Die meisten Programmieralgorithmen werden mit Hex-File 1 auskommen, aber es ist durchaus möglich das ein Algorithmus vor dem Programmieren mehrere dieser Dateien kombiniert.

*Die Schaltflächen **Compile UPA** und **Reset** werden normalerweise nicht verwendet. Sie dienen in erster Linie den Algorithmusentwicklern. **Reset** setzt die Amadeus Hardware zurück, wenn diese aufgrund eines Algorithmusfehlers nicht mehr reagiert. Die **Compile UPA** Schaltfläche startet den Kompilierungsvorgang einer upa Datei. Im Normalfall geschieht dies jedoch automatisch, sobald Amadeus erkennt, dass sich eine Datei geändert hat oder neu gespeichert wurde.*

When **Auto Program** is checked, Amadeus watches the first hex-file. Whenever it changes, Amadeus **clicks** on the Program button for you.

*Wenn **Auto Program** abgeharkt ist, überwacht Amadeus die erste hex Datei. Sobald sie sich ändert wird automatisch die Programmierung (**Program** Schaltfläche) gestartet.*

Clicking on the MiniPrommer button will open the Mini Prommer window. See 3.7.5 for more details.

Das Mini-Prommer-Fenster kann über die entsprechende Schaltfläche gestartet werden. Siehe Abschnitt 3.7.5 für weitere Details.

When running serial number or flash-cycle-counter are needed, you can add them yourself. For further details have a look at the **save** function in the UPAL documentation.

*Wenn laufende Seriennummern oder Flashzykluszähler benötigt werden, musst du diese selber hinzufügen. Wie es geht wird in der UPAL Dokumentation im Zusammenhang mit der **save** Funktion beschrieben.*

5 Amadeus Programmer Hardware

5.1 Electrical Characteristics/Elektrische Eigenschaften

5.1.1 Serial Programming Interface

At the moment, the complete hardware is designed for 5V programming. Every output can drive about 20mA. The 12V can drive about 2-3mA.

Derzeit arbeitet die Hardware nur mit 5V. Die Ausgänge können ca. 20mA Treiben. Die 12V dürfen nicht belastet werden (max. 2-3mA).

*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, lots of PIC18 and dsPIC30 @2.5V with the upper mentioned method without any resistor. Worked so far.

*Wenn Du beabsichtigst Mikrokontroller, die mit weniger als 5V arbeiten, zu programmieren, kannst Du folgendes versuchen (**auf eigene Gefahr**). Nahzu alle Mikrokontroller sind mit sogenannten ESD-Schutzdioden ausgestattet. Diese ziehen Überpannungen an den Pins auf ca. 0,6V über VCC herunter. Je nach Belastung der Betriebsspannung (VCC) kann sich diese über die Schutzdioden auch erhöhen. Mit einem kleinen Widerstand im Programmierkabel oder einer Spannungsbegrenzung mit Dioden (4-5 Dioden in Reihe für ca. 2,5V-3V) sollte es also gehen (**auf eigenes Risiko!!!**).*

Ich habe diese Methode mit einem ATmega64 bei 3V und verschiedenen PIC18 und dsPIC30 bei 2,5V getestet. Hat bei mir auch ohne Schutzwiderstand oder -diode funktioniert.

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

Die folgenden Tabellen zeigen die Pinbelegung des Programmers.

Pin on Programmer Hardware	***Description***
Pin 1: SCL	High-impedance during stand-by. Can drive up to 20mA during programming. *Hochohmig im Standby. Kann wärend der Programmierung bis zu 20mA treiben.*
Pin 2: GND	Ground *Masse*
Pin 3: SDA	High-impedance during stand-by. Can drive up to 20mA during programming. *Hochohmig im Standby. Kann wärend der Programmierung bis zu 20mA treiben.*

Table 13: N2

Pin on Programmer Hardware	*Description*
Pin 1: PGM/PDI	High-impedance during stand-by. Can drive up to 20mA during programming. *Hochohmig im Standby. Kann wärend der Programmierung bis zu 20mA treiben.*
Pin 2: GND	Ground *Masse*
Pin 3: PGD/PDO	High-impedance during stand-by. Can drive up to 20mA during programming. *Hochohmig im Standby. Kann wärend der Programmierung bis zu 20mA treiben.*
Pin 4: GND	Ground *Masse*
Pin 5: PGC/SCK	High-impedance during stand-by. Can drive up to 20mA during programming. *Hochohmig im Standby. Kann wärend der Programmierung bis zu 20mA treiben.*
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 2-3mA form this line. Otherwise the 12V breaks down. *Für hochohmig, siehe Datenblatt zu 74LS05.* *Wenn ein Pull-Up Widerstand für Reset benötigt wird, sollte er im Bereich 100-200kΩ liegen.*
Pin 7: EXTVCC	Switched power-supply for standalone programming. Do never connect to any other power supply. Do not draw more than 100mA. *Spannungsversorgung für Standalone Programmer. Nie mit mehr als 100mA belasten.*

Table 14: N3

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

Table 15: N4 and N5

5.1.2 5V/3V Level-Shifter/Pegel-Wandler

Not all 3V MCUs are 5V tolerant, but Amadeus is working with 5V, thus you need a special interface for not 5V tolerant MCUs. The adaptor in illustration 20 will solve this problem as it makes sure, the 5V coming from Amadeus are reduced to an adjustable level.

Da insbesondere bei 3V Mikrocontrollern das Anlegen von 5V Pegeln schadhaft sein kann, bietet sich folgender Adapter (Abbildung 20) zur Lösung an. ***Die Leitung MCLR fehlt in der Abbildung, sie wird aber benötigt und sollte genau so aufgebaut sein, wie die PGM/PDI Leitung.***

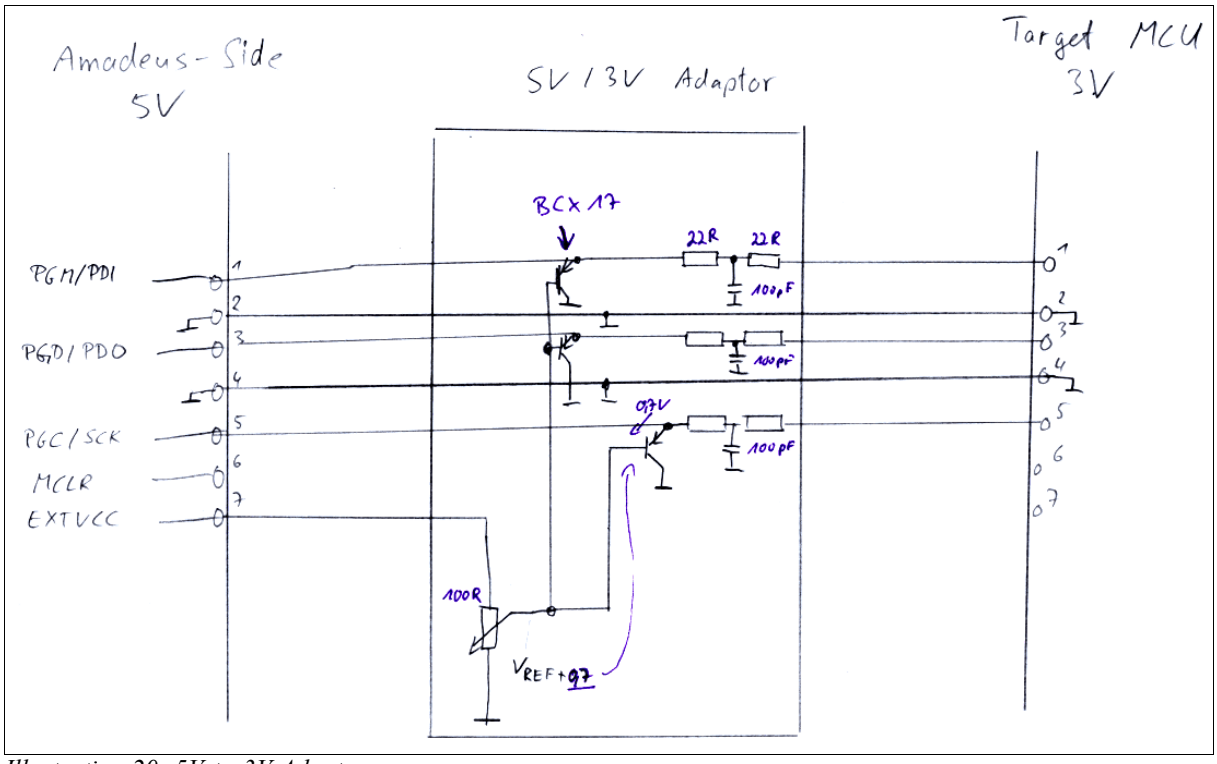

Illustration 20: 5V-to-3V Adaptor

Here is a short description, how the adaptor works: The 7 connections on the left side are Amadeus' N3 connector. On the right side you find the target programming adaptor. The pin assignment stays the same, but in every conductor we add a filter **(in the drawing the MCLR conductor is missing, connect it the same was as PGM/PDI)**. In the drawing they are 22Ω and 100nF. As transistor we need a type with high gain (e.g. BCX17). Using the potentiometer we adjust the voltage so, that adding the transistor's base-emitter-voltage, we get the desired maximum voltage. E.g. when you run your target at 3.3V it is recommended to set up the voltage limit to about 3.4V (0.1V above the target voltage). Have a look at the MCU's specification what voltage is allowed. After the first programming action the EXTVCC pin continuously delivers about 5V depending on your USB port. So be aware that the reference voltage may change in our case, when we connect Amadeus to another USB port!

To set up the potentiometer correctly, go to the UPAL userinterface and select **Chiptype** "Amadeus [Hardwaretest]". Now you can activate the output by selecting the **Function** xxxx_ON and clicking **Run**. Then measure with a multimeter and adjust the voltage with the potentiometer.

Microchip names this active clamping[1] (DS41285A page 24). Have a look in this document. Maybe you will alter something. When you want to design your very own voltage limiting, keep in mind that Amadeus transmits data at 10Mbit (5MHz).

Anbei eine kurze Beschreibung: Die 7 Anschlüsse auf der linken Seite stellen den Stecker N3 auf dem Amadeus-USB Programmer dar. Auf der rechten Seite sehen wir unsere Zielschaltung. Die Steckerbelegung ändert sich nicht. Jedoch kommt in alle Signalleitungen, außer den Masseverbindungen, ein Filter, bestehend aus einem 100pF Kondensator und ein oder zwei Widerständen. In der Schaltung sind zwei Widerstände à 22Ohm gezeigt. Alternativ

1 Microchip DS41285A page 24 (http://ww1.microchip.com/downloads/en/DeviceDoc/41285A.pdf)

(bei mir in Betrieb) nur ein 33Ohm Widerstand zwischen Transistor und Kondensator. Als Transistor wird ein PNP Transistor mit möglichst großer Verstärkung benötigt. BCX17 oder vergleichbare sollten geeignet sein. Die Basen der Transistoren liegen auf einer mit dem Potentiometer fest eingestellten Referenzspannung. Speziell im UPA Modus liegen nach der ersten Aktivität dauerhaft 4,5V-5V (abhängig von der USB Spannung des Verwendeten Computers; also aufpassen, wenn man den Port wechselt!) an EXTVCC an, so dass wir hier unsere Spannung für das Potentiometer her bekommen. Im UPA Modus gibt es einen ***Chiptype*** *namens „Amadeus [Hardwaretest]". Unter* ***Function*** *tauchen nun verschiedene Testfunktionen auf. Einfach alle Leitungen auf ON schalten (auswählen und auf Run klicken). Dann mit dem Voltmeter die Spannung auf der Ziel MCU Seite messen und so lange am Potentiometer drehen, biss die gewünschte Spannung erreicht ist. Empfehlenswert sollte minimal (0,1V) oberhalb der Betriebsspannung des Zielmikrocontrollers sein (ins Datenblatt schauen was zulässig ist; bei PIC24/dsPIC33 z.B. max. 0,3V). Da wir einen einfachen Spannungsteiler (Poti) verwenden, ist unsere Referenzspannung nur so gut wie unsere Versorgungsspannung, dass heißt, dass wir bei einem Portwechsel die Spannung unter Umständen neu einstellen müssen. Auch sollten am gleichen USB-Port keine Geräte betrieben werden, die sehr viel Strom benötigen.*

Microchip bezeichnit diese Technik der Pegelanpassung als Active Clamp[1] (DS41285A Seite 24). Nach Microchip kann man unter Umständen auch direkt die Spannung der Zielschaltung verwenden, wobei allerdings zu bedenken ist, dass die Basis-Emitter-Spannung wohl größer als 0,3V ausfallen wird. Ebenfalls könnte man noch einen Widerstand zwischen den Amadeus Ausgängen (5V Seite) und Transistor einfügen.

Für alle, die ihre eigene Schaltung für die Spannungsanpassung entwerfen möchten sei erwähnt, dass die maximale Datenrate beim Programmieren 10Mbit (5MHz) erreicht.

Wenn jemand noch gute Vorschläge hat, wie man die Anpassung realisieren kann (incl. Parametrierung der Bauteile), einfach an mich wenden (E-Mail siehe Website).

1 Microchip DS41285A page 24 (http://ww1.microchip.com/downloads/en/DeviceDoc/41285A.pdf)

5.2 Schematics

If **12 volt** are not used D1-D7, C21-C27, R9-R12, IC3, R16, R17, Q2 and D9 don't need to be assembled.

If the **relay** is not needed D8, C15, REL1, N4 and N5 don't need to be assembled.

If you don't need an **active indication** you can omit LED1 and R14.

Improvement: Connect C2 between the other side of R1 and GND and C3 between the other side of R2 and GND. Also see comment to L1 in chapter 5.3 “Partlist“ and page 52.

*Wenn keine **12V** benötigt werden, brauchen D1-D7, C21-C27, R9-R12, IC3, R16, R17, Q2 und D9 nicht bestückt werden.*

*Wenn das **Relais** nicht benötigt wird, brauchen D8, C15, REL1, N4 und N5 nicht bestückt werden.*

*Wenn keine **Aktivitätsanzeige** benötigt wird, brauchen LED1 und R14 nicht bestückt werden.*

***Verbesserung:** Verbinde C2 mit der anderen Seite von R1 und GND und C3 mit der anderen Seite von R2 und GND. Siehe auch den Kommentar über L1 im Kapitel 5.3 “Partlist“ und Seite 52.*

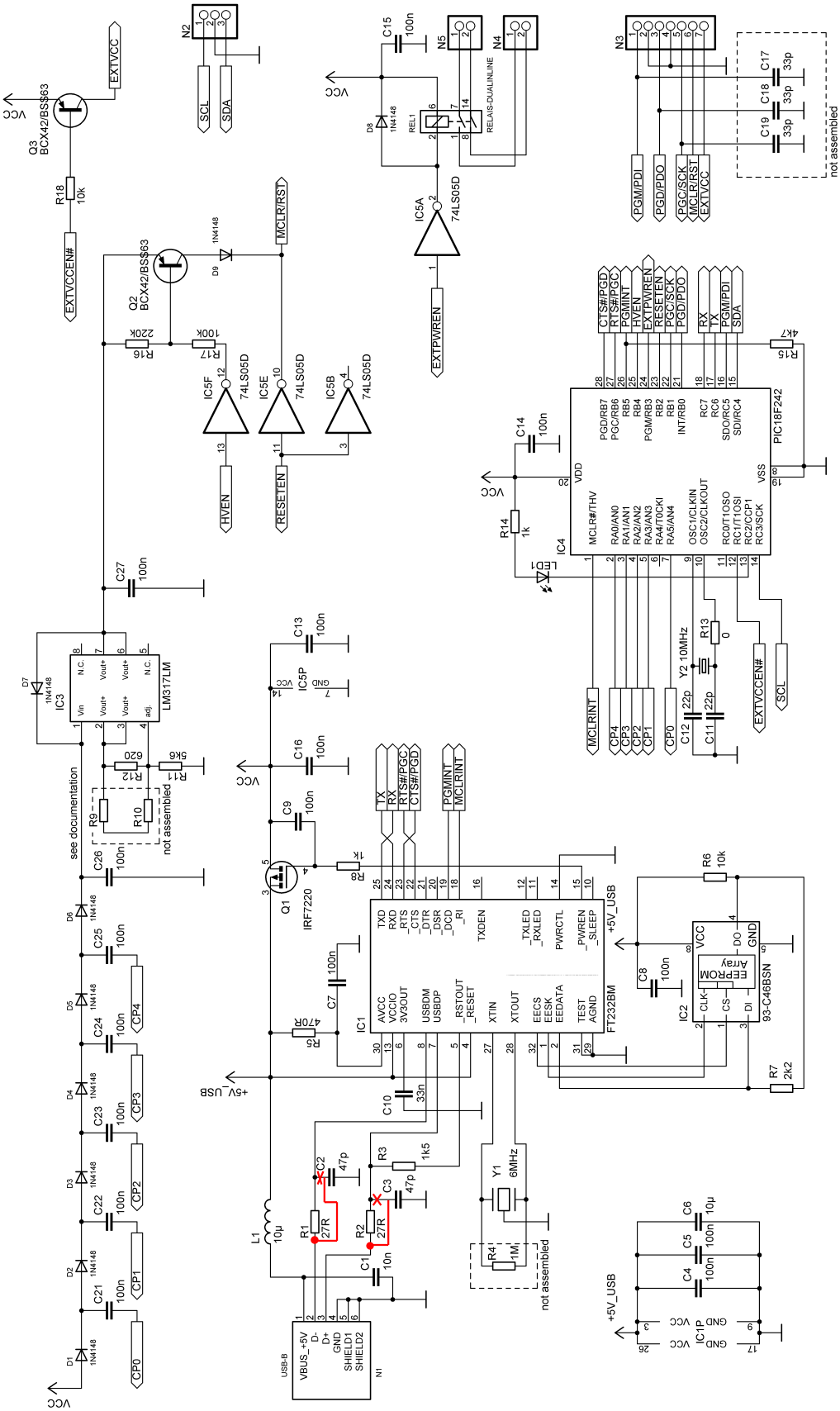

Illustration 21: Schamatics

5.3 Partlist

Bezugsquelle für Deutschalnd: Derzeit (Dezember 2004) sind alle Teile bei www.reichelt.de zu bekommen.

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	*Nicht bestücken*
C18	Not Assembled	C0805	*Nicht bestücken*
C19	Not Assembled	C0805	*Nicht bestücken*
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 or FT232BL	LQFP32	FTDI FT232BM or the newer lead free version FT232BL *Der FT232BL ist die Bleifrei-Variante des FT232BM*
IC2	93-C46BSN	SO-08	93C56; 93C66
IC3	LM317LM	SO-08	LM317LM

Position	*Value*	*Package*	*Comment/Reichelt ordercode*
IC4	(PIC18F242 or) **PIC18F252**	SO-28W	*PIC18F242 or **PIC18F252 (reqired for A-UPAL*)*** *Important: If you are recycling an old PIC, Low Voltage Programming must be enabled.* ***PIC18F252** ist in jedem Fall zu empfehlen. Derzeit sollte aber auch noch ein PIC18F242 reichen. Wenn A-UPAL* verwendet wird, PIC18F252 wählen!* * Amadeus Universal Programming Algorithm Language
IC5	74LS05D	SO-14	Only use LS type *Nur LS Typen verwenden*
L1	10µ	L0805	*z.B. JCI 2012 10µ (Reichelt)* !!! Using this component works quite good for me, but as I was told the maximal allowed current is to low. So if you run into trouble, use a coil/ ferrite that allows a higher current. *!!! Die oben genannte Induktivität funktionert bei mir zwar bisher ohne Probleme, aber ihr maximal zulässiger Strom ist zu gering. Sollte es zu Problemen kommen, so ist diese Spule durch eine andere Spule oder Ferrit auszutauschen.*
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) *Mit den Widerständen werden die 12V eingestellt.*
R11	5k6	R1206	
R12	620R	R1206	You can assembled other resistors (R9+R10||R12) to adjust the accuracy of VPP (12V) *Mit den Widerständen werden die 12V eingestellt.*
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) *Mit den Widerständen werden die 12V eingestellt.*
REL1	RELAIS-DUALINLINE	DIL-14-RELAIS	MEDER electronic “DIP 7221-L 5V” www.meder.com
Y1	6MHz	RESONATOR	CSTCC 6,00

Position	*Value*	*Package*	*Comment/Reichelt ordercode*
Y2	10MHz	Crystal HC49U-/S or -HC18	10-HC18

Table 16: Partlist

5.4 Assembly/Bestückungsplan

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

Correction for illustration 22: Pin1 mark of N2, N3, N4 and N5 is on the wrong end of the connector.

See chapter 5.2 "Schematics" for other corrections.

Korrektur für Abbildung 22: ***Pin 1 ist bei N2, N3, N4 und N5 am falschen Ende eingezeichnet. Pin 1 befindet sich bei allen vier Steckern am unteren Ende.***

Siehe auch Kapitel 5.2 "Schematics" für weitere Korrekturen.

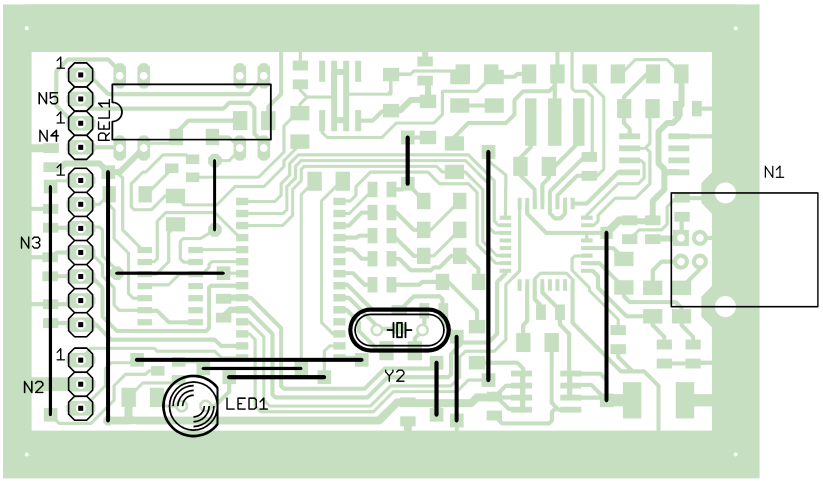

Illustration 22: PCB top view

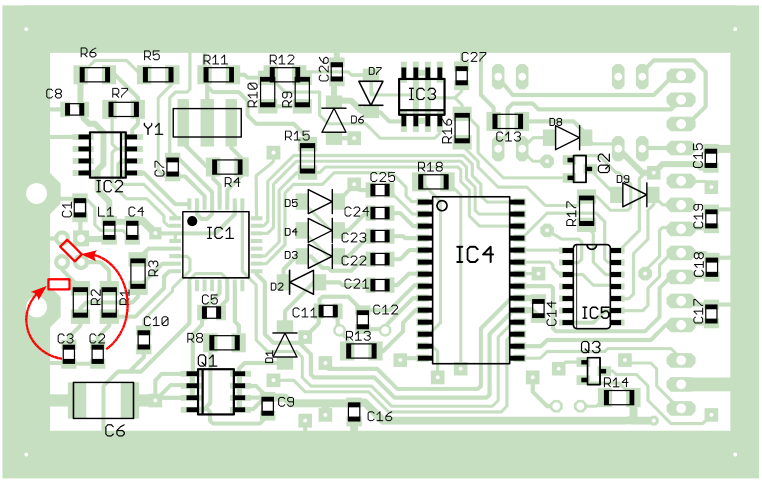

Illustration 23: PCB bottom view

5.5 Layout

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

Das Layout ist noch einmal als separates File mit dem Namen “Amadeus_PCB.gif” mit höherer Auflösung im Downloadbereich vorhanden. Nach dem Drucken muss das Layout 80mmx50mm groß sein.

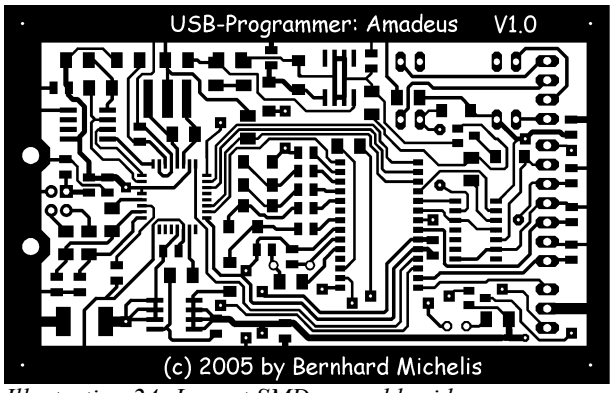

Illustration 24: Layout SMD assembly side

6 FAQ

Q: Will you add further PIC16 devices?
A: No.

Q: I cannot find the FT232BM anymore.
A: You can take the FT232BL. It's the lead free version of the FT232BM

F: Werden weitere PIC16 Typen implementiert?
A: Nein.

F: Ich bekomme den FT232BM nicht mehr zu kaufen. Kann ich auch einen FT232BL verwenden?
A: Ja, die FT232BL Version ist die bleifrei Variante des FT232BM.

Stichwortverzeichnis

History

Date	*Comment*
29.12.04	First version
02.01.05	German documentation added
20.01.05	Chapter AVR “Reanimation / Wiederbelebung“ added
17.02.05	Added hint to component list for L1
27.03.05	Chapter “Assembly/Bestückungsplan“: Correction of N1, N2, N3 and N4 Pin1 PIC10 MCUs added PIC18F4550 added
22.04.05	Addition PIC18 and PIC18 LowVoltageProgramming added dsPIC30 added
12.05.05	Chapter about Tracing functionality added Additional PIC18 added Component list: PIC18F252 recommended, now
18.05.05	Hardware improvement: See C2, C3 in chapter 5.2 “Schematics“ and 5.5 “Layout“.
16.06.05	Typlist updated. Chapter 3.7.6 about “PIC18F6x10, PIC18F6x90, PIC18F8x10, PIC18F8x90“ added FAQ added
26.07.06	A-UPAL (Amadeus – Universal Programming Algorithm Language) added Bug-Fix for dsPIC30: Saving configurations to hex-files Bug-Fix for dsPIC30F5011, dsPIC30F5013: Bulk erase should work now.
24.06.07	PIC24/dsPIC33 description added. Chapter 5.1.2 “5V/3V Level-Shifter/Pegel-Wandler” added.
17.07.07	PIC18FJ added UPAL “Auto Program” feature added
02.10.07	Tracing: Float format added Chapter 5.2 “Schematics“: Comments added
14.10.07	Chapter 3.4.3 "Factory Reset" edited New functionality for AVR interface added: Frequency Measurement and Mini-Prommer-Window. Chapter 3.4.7 "Measure MCU Speed / MCU Geschwindigkeitsmessung" added Chapter 3.4.8 "Mini Prommer Window" added Chapter 3.4.9 "Tracing/Tracen" added
23.10.08	Kapitel zum PIC32 hinzugefügt
06.11.08	Mini-Prommer-Window for UPA-UI addes PIC32 completely changed