Einführung in die Programmierung mit BASCOM

Inhalt

1 Einleitung 3

2 Installation und Vorbereitung 4

2.1 Wo bekommt man BASCOM? 4

2.2 Was ist bei der Installation zu beachten? 4

3 Die BASCOM Benutzeroberfläche 5

4 Konfiguration von BASCOM für die myAVR-Produkte 6

4.1 Konfiguration für das myAVR Board MK1 LPT 7

4.2 Konfiguration für den mySmartUSB MK2 8

4.3 Konfiguration für den mySmartUSB MK3 9

4.4 Konfiguration für den mySmartUSB light 10

5 Erstellen und Brennen eines AVR BASCOM-Programms 11

6 BASCOM-Befehlssatz 13

7 Programmierung in BASCOM 15

7.1 Variablen und deren Verwendung in BASCOM 15

7.2 Eingaben in BASCOM 17

7.3 Ausgaben in BASCOM 19

8 Manual Program 20

3. Auflage:Februar 2015

Promenadenring 8
02708 Löbau
Deutschland

www.myAVR.de
support@myavr.de

Tel: ++49 (0) 358 470 222
Fax: ++49 (0) 358 470 233

1 Einleitung

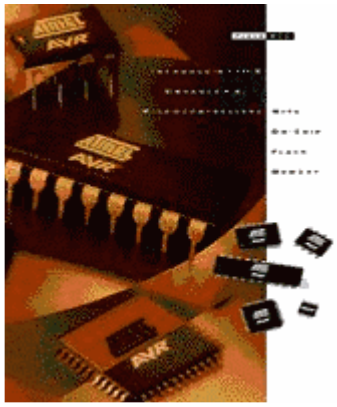
Dieser Abschnitt ist als Schnelleinstieg in die Programmierung von AVR-Mikrocontrollern mit BASCOM gedacht. Dabei werden Kenntnisse einer Programmiersprache wie BASIC, C/C++ oder PASCAL vorausgesetzt.
Erstellt wurde dieses Dokument mit der BASCOM Version 1.11.9.8.
BASCOM ist kein BASIC-Interpreter wie der Name vielleicht vermuten lässt, sondern ein echter Compiler. Genau wie bei Assembler und C ist das Endprodukt der Programmierung mit BASCOM-AVR auch ein HEX-File, die per SPI auf den Controller gebrannt wird.
Die Beliebtheit von BASCOM resultiert daraus, dass die Sprache stark an BASIC angelehnt ist und der Einstieg durch mächtige und verständliche Befehle recht leicht gemacht wird. Leider verbirgt diese Hochsprache etwas die interne Struktur und Funktionsweise des Mikrocontrollers und so manche spezielle Aufgabenstellung lässt sich mit BASCOM dann doch nur schwer lösen.

2 Installation und Vorbereitung

2.1 Wo bekommt man BASCOM?

BASCOM-AVR© ist ein Produkt der Firma MCS Electronics in Holland.

MCS Electronics
De Paal 112
Almere-Haven
HOLLAND
1351JJ
http://www.mcselec.com/



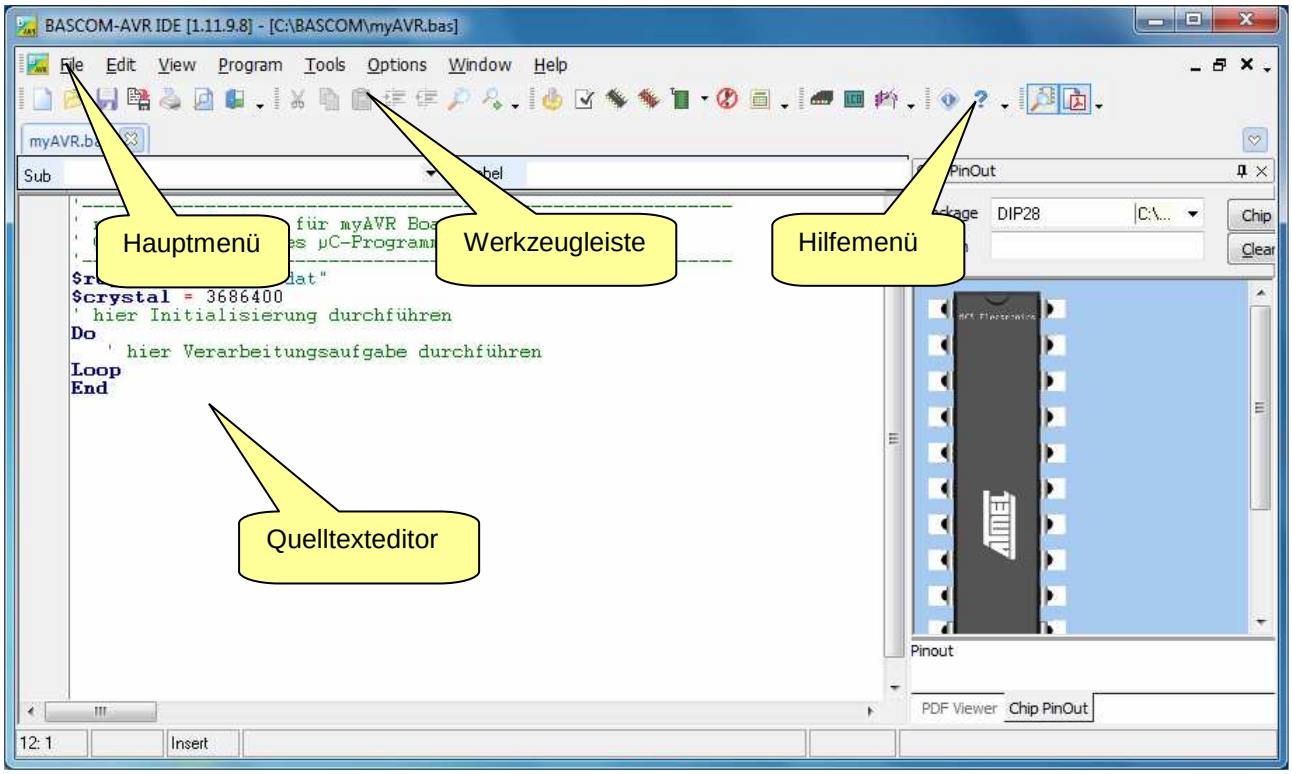
Die Software ist lauffähig unter Windows 95, 98, NT, 2000, XP, Vista und Win7. Der Preis für die Basisversion liegt bei 89,00 € (Stand 07/2010). Es gibt auch eine kostenlose Demo-Version, wobei der Quellcode hierbei auf 4 KB beschränkt ist.

2.2 Was ist bei der Installation zu beachten?

Die Installation wird mit *setup.exe* gestartet und ist unproblematisch. Bei der Demo ist zu beachten, dass das heruntergeladene ZIP-Archiv in einem temporären Verzeichnis gespeichert und entpackt werden muss.
Mit *setup.exe* wird BASCOM dann aus diesem Verzeichnis heraus installiert. Dieses Verzeichnis kann nach der Installation gelöscht werden.
Unter Windows NT, 2000, XP, Vista und Win7 ist darauf zu achten, dass der Installationsvorgang mit Administratorrechten durchgeführt werden muss. Die Deinstallation erfolgt über *Systemsteuerung / Software entfernen*.

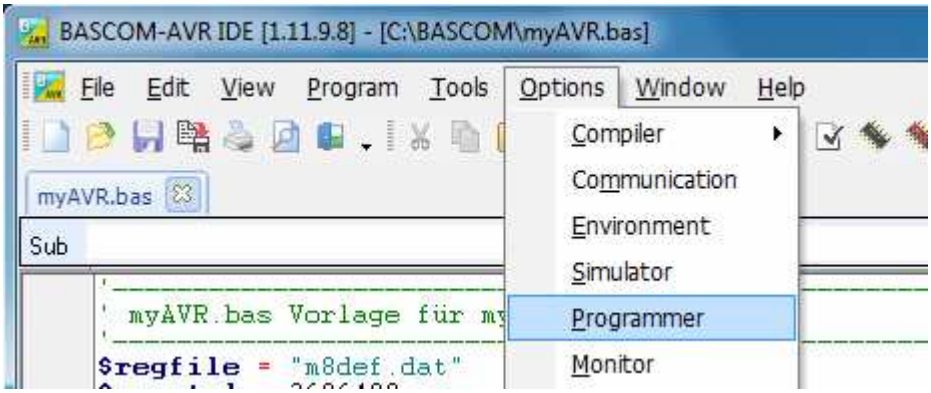
3 Die BASCOM Benutzeroberfläche

Die Entwicklungsumgebung (IDE, Integrated Development Environment) besitzt einen Quelltexteditor mit Syntaxhervorhebung, integriertem Compiler und Brennprogramm sowie umfangreiche Hilfefunktionen.



4 Konfiguration von BASCOM für die myAVR-Produkte

Bevor die Arbeit beginnen kann, sind noch ein paar Konfigurationsschritte nötig.
Damit die erstellten HEX-Dateien in den FLASH-Programmspeicher des Controllers gebrannt werden können, muss die Programmerkonfiguration eingestellt und das myAVR Board angeschlossen sein. Diese Einstellung erfolgt jeweils über die Menüfolge *Options/Programmer.*



In dem Dialogfenster zur Programmereinstellung (BASCOM-AVR Options) steht die Option „Auto Flash“ zur Verfügung. Diese Option erleichtert den Brennvorgang. Dabei werden die nötigen Schritte:

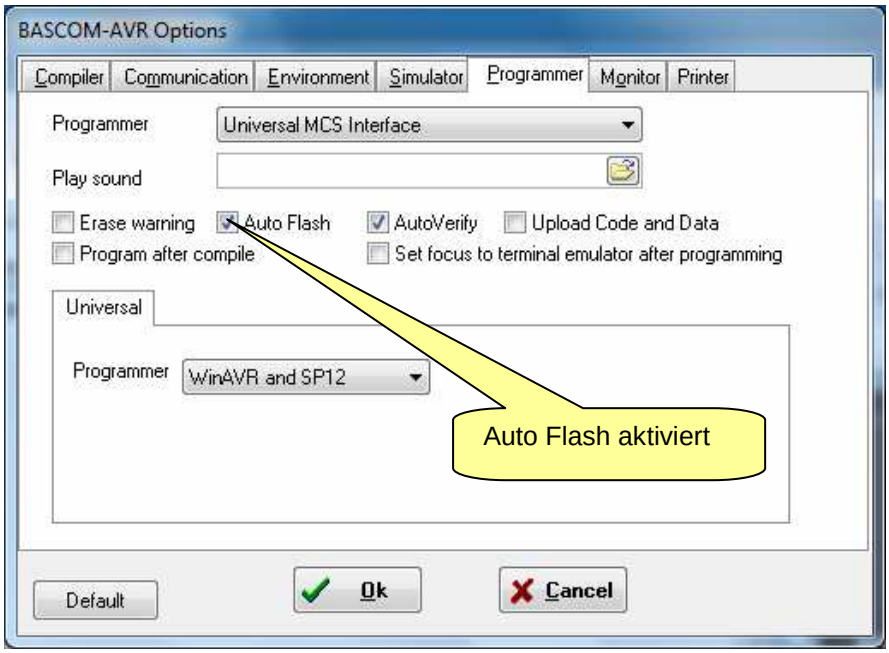
- Programmer und Controller checken,
- FLASH / EEPROM löschen und
- FLASH / EEPROM schreiben

in einem Arbeitsgang durchgeführt.
Für Fortgeschrittene stellt BASCOM noch „Manual Program“ bereit. Näheres hierzu, finden Sie im Abschnitt Manual Program.

Hinweis:
Nach den erfolgten Einstellungen kann es vorkommen, dass die weitere Arbeit mit BASCOM noch nicht möglich ist. In diesem Fall wird empfohlen, BASCOM zu beenden und erneut zu starten.

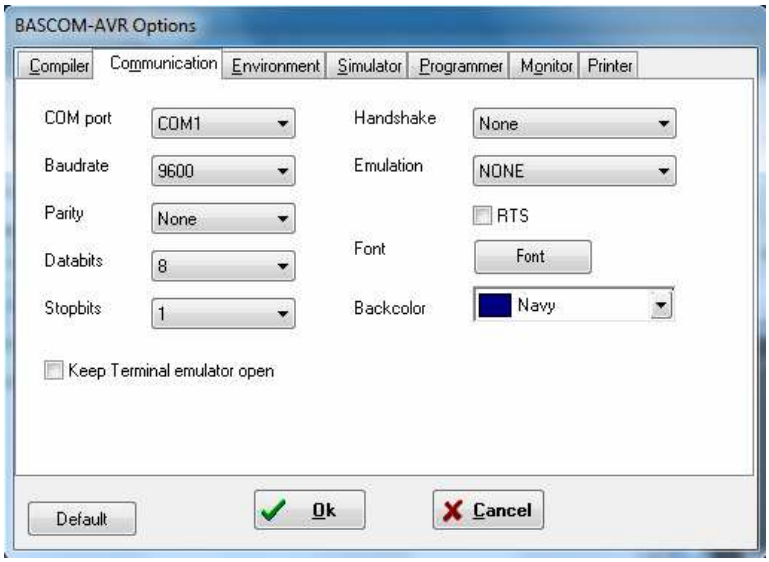
4.1 Konfiguration für das myAVR Board MK1 LPT

Das myAVR Board MK1 LPT verfügt über einen SP12-kompatiblen Parallelport-Programmer und eine serielle Schnittstelle.
Die folgende Abbildung zeigt die Einstellung für das myAVR Board MK1 LPT.



Programmer : Universal MCS Interface,
WinAVR and SP12

Über die serielle Schnittstelle kann mit dem myAVR Board MK1 LPT kommuniziert werden. BASCOM bietet dafür ein Terminal und ein DEBUGGER/SIMULATOR an. Dazu sind ein Null-Modemkabel und eine freie COM-Schnittstelle am PC notwendig. Die zu tätigenden Einstellungen in BASCOM sind die Folgenden:

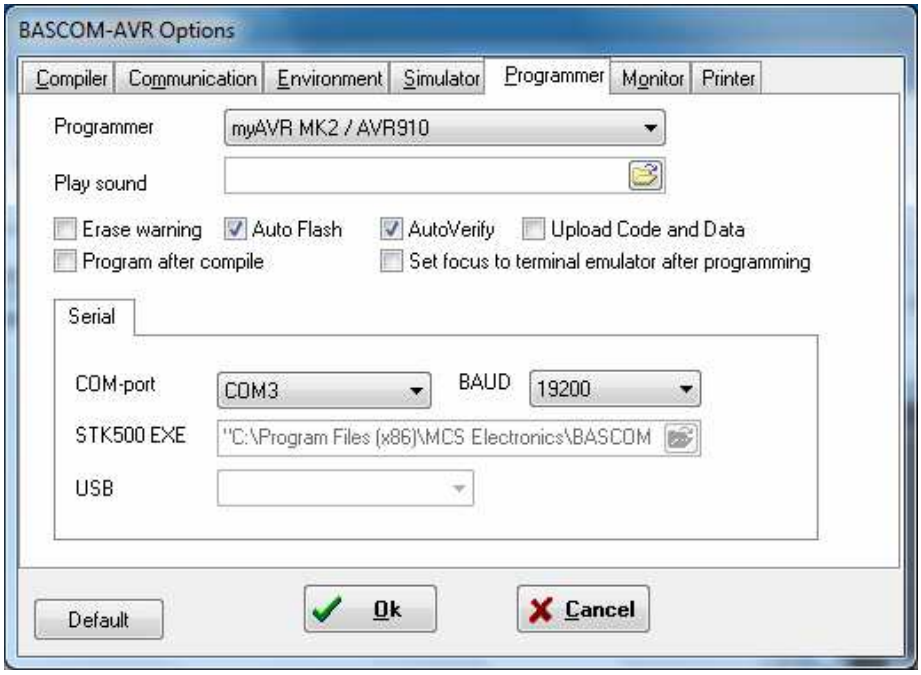


COM-Port	:	x (siehe Gerätemanager)	Handshake	:	None
Baudrate	:	9600	Emulation:	:	NONE
Parität	:	None			
Datenbits	:	8			
Stoppbits	:	1			

4.2 Konfiguration für den mySmartUSB MK2

Auch bei der USB-Version sind zuvor noch einige Einstellungen nötig, bevor man beginnen kann. Beachten Sie bitte den Hinweis unter Punkt 4.
Der mySmartUSB MK2 ist ein AVR911/910/109 kompatibler USB-Programmer. Damit die erstellten HEX-Dateien in den FLASH-Programmspeicher des Controllers gebrannt werden können, muss auch hier die Programmerkonfiguration eingestellt und der mySmartUSB MK2 angeschlossen sein.

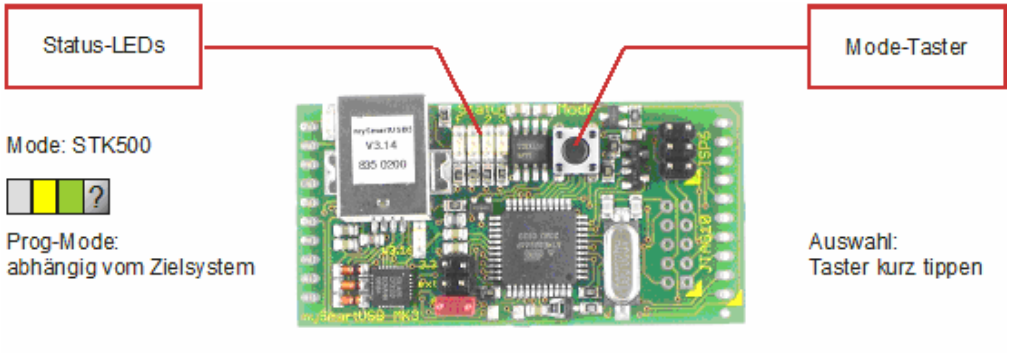
Die folgende Abbildung zeigt die korrekte Einstellung für den mySmartUSB MK2.



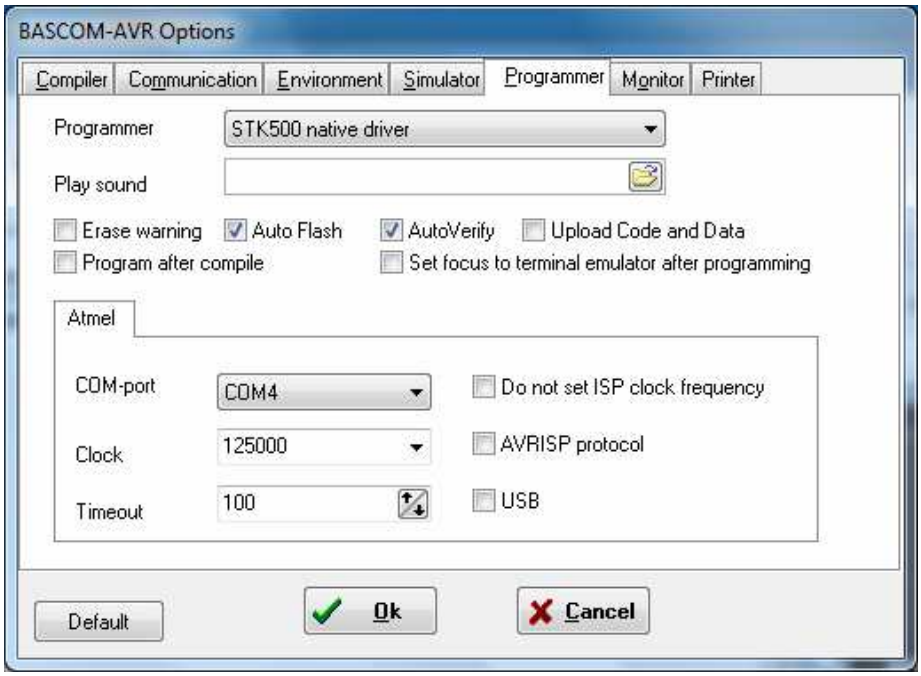
Programmer : myAVR MK2/AVR910
COM-Port : x (siehe Gerätemanager)

4.3 Konfiguration für den mySmartUSB MK3

Wie bei dem mySmartUSB MK2 sind auch hier einige Einstellungen vorzunehmen. Beachten Sie bitte den Hinweis unter Punkt 4.
Der mySmartUSB MK3 verfügt über verschiedene Programmiermodi, empfehlenswert ist z.B. der STK500-Mode. Diesen Modus kann man mit dem Mode-Taster einstellen (näheres in der Technischen Beschreibung zum mySmartUSB MK3).



Die folgende Abbildung zeigt die korrekte Einstellung für den mySmartUSB MK3

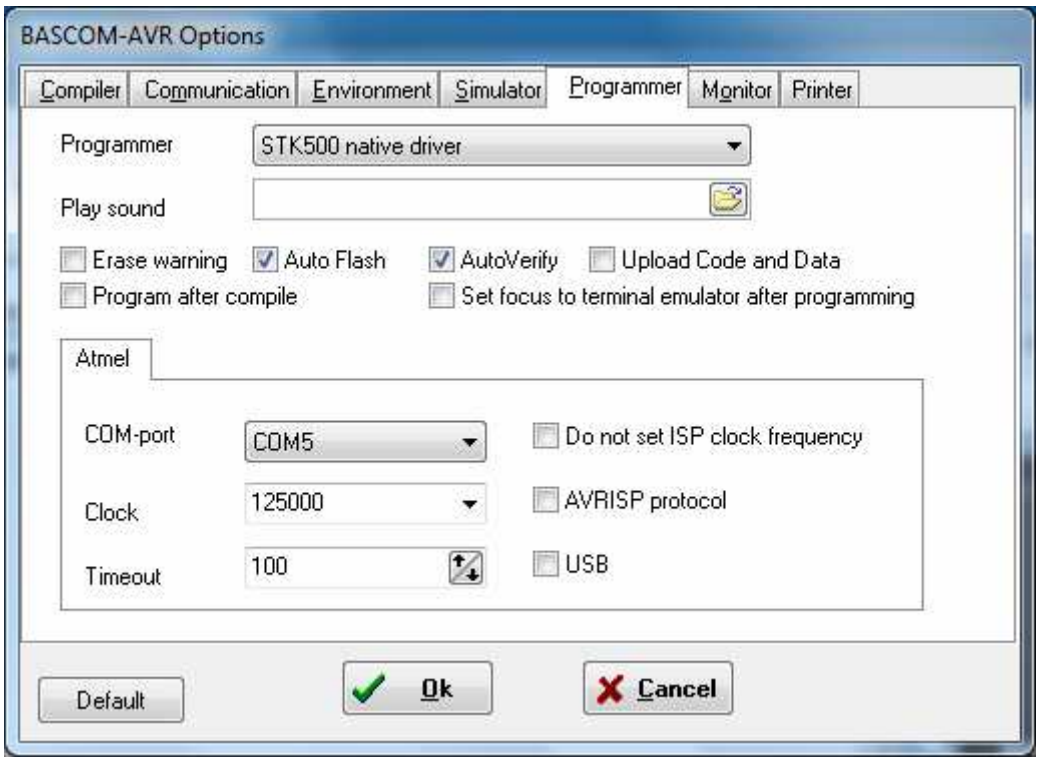


Programmer : STK500 native driver
COM-Port : x (siehe Gerätemanager)

4.4 Konfiguration für den mySmartUSB light

Der mySmartUSB light verfügt, wie auch der mySmartUSB MK3, über verschiedene Programmiermodi, empfehlenswert ist z.B. der STK500-Mode. Der Modus ist von der jeweiligen Firmware abhängig. Das Umstellen der Firmware ist möglich mittels der mySmartUSB light – SupportBox, einem kleinen Tool auf einer grafischen Oberfläche. Es steht unter www.myAVR.de zum Download bereit.
Beachten Sie bitte den Hinweis unter Punkt 4.

Die folgende Abbildung zeigt die korrekte Einstellung für den mySmartUSB light



Programmer : STK500 native driver
COM-Port : x (siehe Gerätemanager)

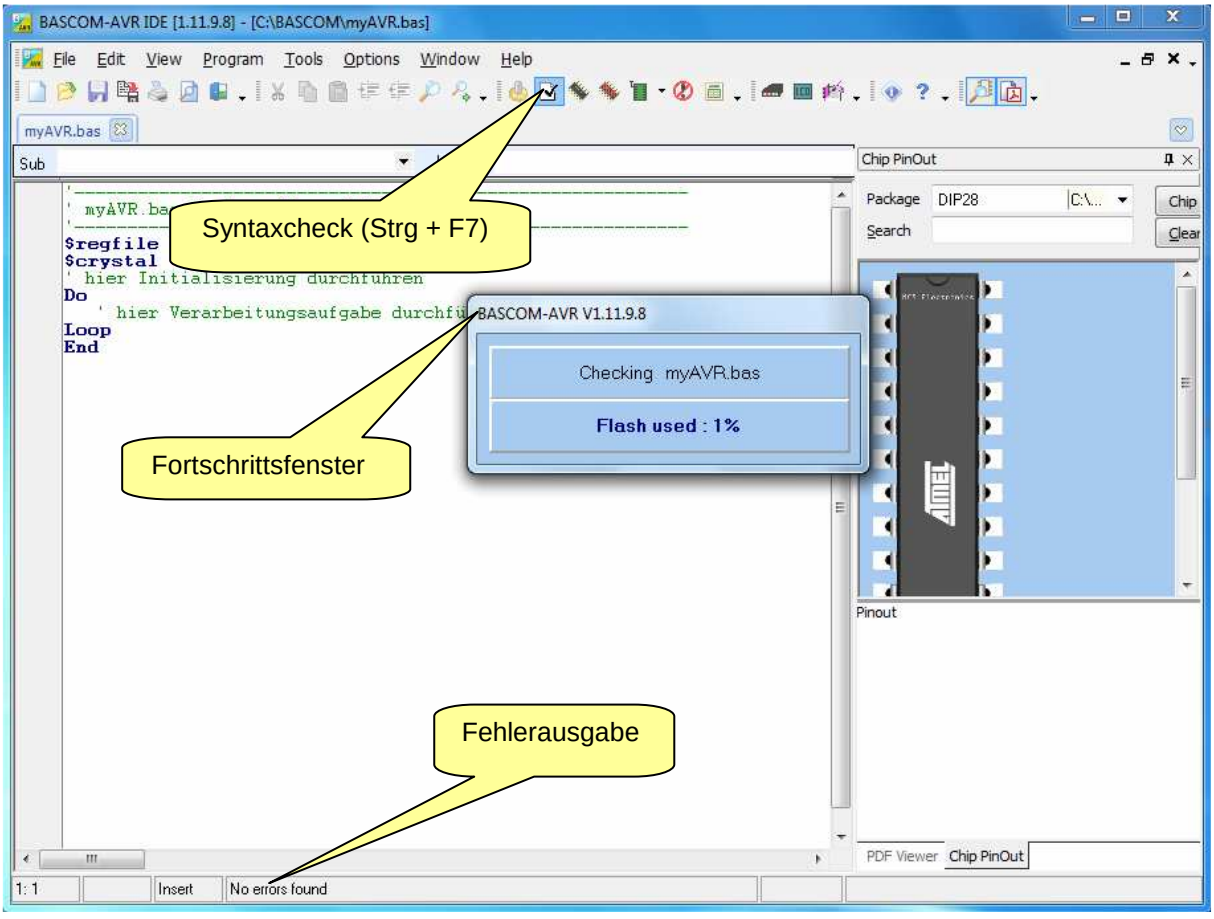
5 Erstellen und Brennen eines AVR BASCOM-Programms

Die Grundstruktur eines BASCOM-Programms ist der Struktur eines C-Programms sehr ähnlich. Die Definitionsdatei für AVR-Controller ist die Datei `*.def`. In C ist dies die Datei `io.h` und beim Assembler ist das die Datei `avr.h`.
Genau wie in C/C++ und in Assembler gliedert sich das Hauptprogramm (main) in die Initialisierungssequenz und eine Unendlichschleife, in der die eigentliche Funktion der Mikrocontrolleranwendung ausgeführt wird.

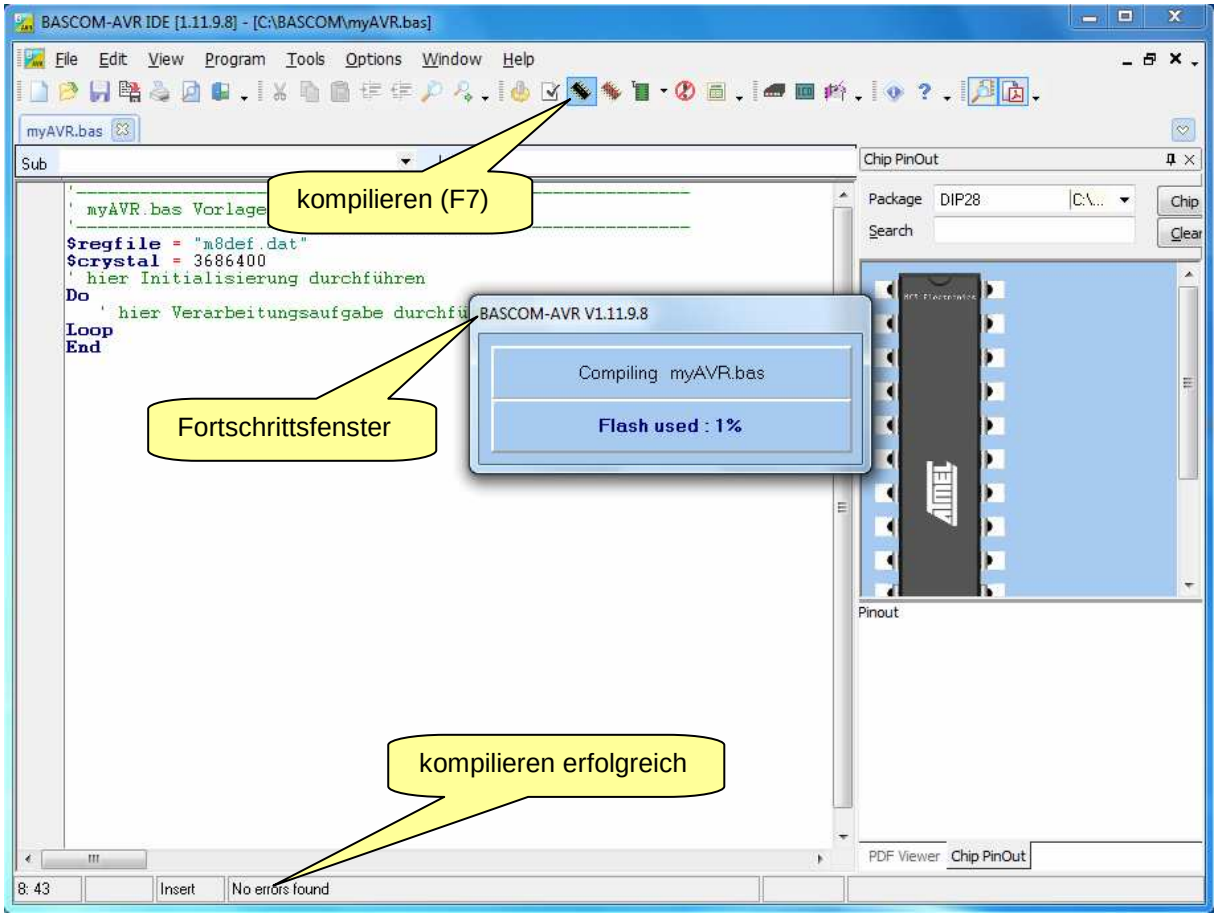
```
'-----------------------------------------------------------
' myAVR.bas Vorlage für myAVR Board
'-----------------------------------------------------------
$regfile = "m8def.dat"
$crystal = 3686400
' hier Initialisierung durchführen
Do
    ' hier Verarbeitungsaufgabe durchführen
Loop
End
```

Kompilieren und Brennen mit BASCOM
Anhand des Grundgerüsts werden im Folgenden der Syntaxcheck, Kompilieren und Linken und Abschließend das Brennen erläutert.

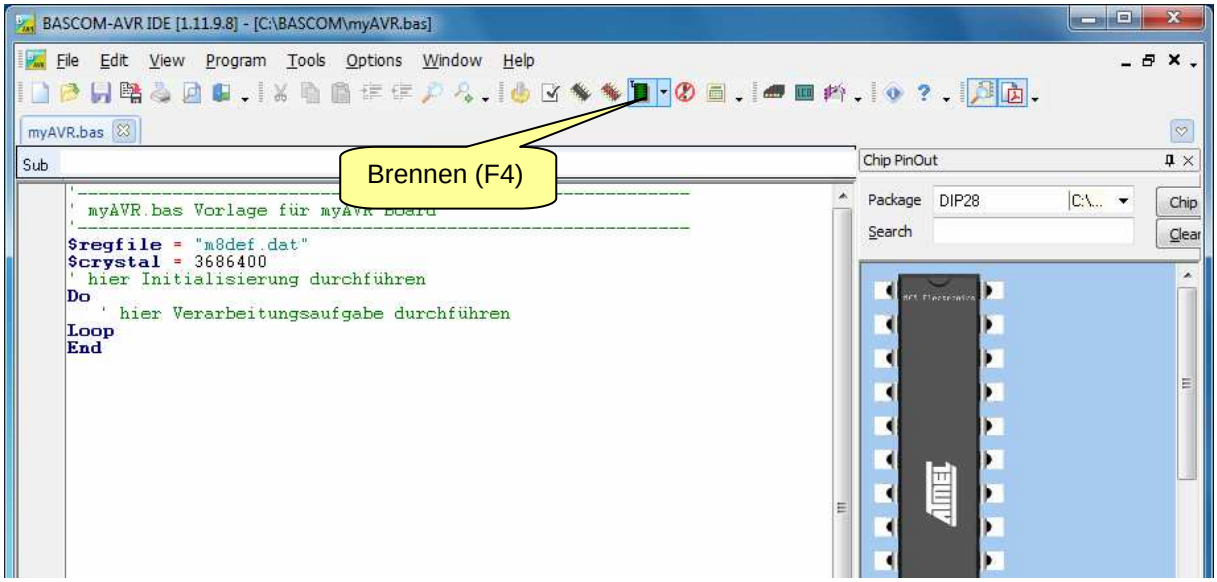
Die Korrektheit des eingegebenen Quellcodes kann mit einem Syntaxcheck (Strg + F7) überprüft werden.



Nach Prüfung der Syntax kann das Programm kompiliert werden



Wenn auch dieses ohne Fehler (No errors found) möglich war, kann nun gebrannt werden.



Beim Brennen öffnet sich kurz das Programmierfenster und verschwindet nach erfolgreichem Brennen wieder. Treten beim Brennen Fehler auf, so sind diese in dem Programmierfenster aufgelistet und erkennbar an dem Symbol (näheres zum Programmierfenster siehe im Abschnitt *Manual Program*)

6 BASCOM-Befehlssatz

Im Folgenden werden die verfügbaren Sprachkonstrukte aufgelistet. Für eine detaillierte Beschreibung kann die Hilfe in der Entwicklungsumgebung genutzt werden. Dazu stellt man einfach den Cursor auf das entsprechende Statement und betätigt die Taste F1.

Programmsteuerung:

Bedingungen: IF, THEN, ELSE, ELSEIF, END IF, SELECT, CASE.

Schleifen: DO, LOOP, WHILE, WEND, UNTIL, EXIT DO, EXIT WHILE, FOR, NEXT, TO, STEP, EXIT FOR.

Unterprogramme: ON .. GOTO/GOSUB.

Eingaben und Ausgaben:

UART: PRINT, INPUT, INKEY, PRINT, INPUTHEX, WAITKEY, INPUTBIN, PRINTBIN, OPEN, CLOSE.

LCD: LCD, UPPERLINE, LOWERLINE, DISPLAY ON/OFF, CURSOR ON/OFF/BLINK/NOBLINK, HOME, LOCATE, SHIFTLCD LEFT/RIGHT, SHIFTCURSOR LEFT/RIGHT, CLS, DEFLCDCHAR, SPC.

PORT/PIN: DDRx, PORTx.n, DEBOUNCE, SHIFTIN, SHIFTOUT, GETATKBD, SERIN, SEROUT.

TWI/I²C: I2CSTART, I2CSTOP, I2CWBYTE, I2CRBYTE, I2CSEND and I2CRECEIVE.

1WIRE: 1WWRITE, 1WREAD, 1WRESET, 1WIRECOUNT, 1WSEARCHFIRST, 1WSEARCHNEXT.

SPI: SPIINIT, SPIIN, SPIOUT, SPIMOVE.

Mathematische Funktionen:

AND, OR, XOR, INC, DEC, MOD, NOT, ABS, BCD, LOG, EXP, SQR, SIN, COS, TAN, ATN, ATN2, ASIN, ACOS, FIX, ROUND, MOD, SGN, POWER, RAD2DEG, DEG2RAD, LOG10, TANH, SINH, COSH.

Bitmanipulation

SET, RESET, ROTATE, SHIFT, BITWAIT, TOGGLE.

Stringmanipulation

STRING, SPACE, LEFT, RIGHT, MID, VAL, HEXVAL, LEN, STR, HEX, LTRIM, RTRIM, TRIM, LCASE, UCASE, FORMAT, FUSING, INSTR.

Variablen

DIM, BIT, BYTE, INTEGER, WORD, LONG, SINGLE, STRING, DEFBIT, DEFBYTE, DEFINT, DEFWORD.

Interruptprogrammierung

ON INT0/INT1/TIMER0/TIMER1/SERIAL, RETURN, ENABLE, DISABLE, COUNTERx, CAPTUREx, INTERRUPTS, CONFIG, START, LOAD.

Kommentare:

REM, ' .

Verschiedenes:
SWAP, END, STOP, CONST, DELAY, WAIT, WAITMS, GOTO, GOSUB, POWERDOWN, IDLE, DECLARE, CALL, SUB, END SUB, MAKEDEC, MAKEBCD, INP,OUT, ALIAS, DIM , ERASE, DATA, READ, RESTORE, INCR, DECR, PEEK, POKE, CPEEK, FUNCTION, READMAGCARD, BIN2GREY, GREY2BIN, CRC8, CRC16, CHECKSUM.

Compileranweisungen, Direktiven
$INCLUDE, $BAUD and $CRYSTAL, $SERIALINPUT, $SERIALOUTPUT, $RAMSIZE, $RAMSTART, $DEFAULT XRAM, $ASM-$END ASM, $LCD, $EXTERNAL, $LIB.

7 Programmierung in BASCOM

7.1 Variablen und deren Verwendung in BASCOM

Variablen werden mit dem Schlüsselwort `DIM` deklariert. Initialwerte sind danach als Wertzuweisung anzugeben. Diese werden im FLASH gespeichert und beim Programmstart automatisch in den SRAM übertragen. Bei Feldern (Arrays), wie zum Beispiel Zeichenketten, ist die Anzahl der Feldelemente als Faktor anzugeben.

Beispiel:

```
'ein Byte
Dim Mybyte As Byte
'ein 5 Byte
Dim Myarray As Byte * 5
Dim Myword As Word
Dim Mylong As Long
Dim Myint As Integer
Dim Mystring As String * 10
```

Die Verarbeitung der Daten erfolgt dann in der Hauptschleife (`mainloop`).

Wertzuweisung : `=`
Arithmetische Operationen : `+, -, *, /, ^`
Logische Operationen : `=, <, >, <=, >=, <>, NOT, AND,OR,XOR`
Bit-Operationen : `NOT, AND, OR, XOR, ROTATE, SHIFT`

Als Übung soll das Lauflichtbeispiel in BASCOM realisiert werden. Dabei wurde ein Bit in einem Register links verschoben (Rotation), kurz gewartet und das Register an einem PORT ausgegeben.

Benötigte BASCOM Befehle: `DIM, ROTATE, WAITMS`.

Nutzen Sie die Hilfe (F1), um die Befehle kennen zu lernen.

```
'-----------------------------------------------------------
' Titel         : Lauflicht für myAVR Board
'-----------------------------------------------------------
' Funktion      : Lauflicht
' Schaltung     : PD5-PD7 an LED`s
'-----------------------------------------------------------
' Prozessor     : ATmega8  3,6864 MHz
' Version       : 1.2  , 23.07.2010
' Autor         : Michael Gesell
'-----------------------------------------------------------
$regfile = "m8def.dat"          ' Prozessortyp ATmega8
$crystal = 3686400              ' Taktrate

Dim Mybyte as Byte              ' ein Byte als Variable
DDRD = &B11100000               ' PD 5-PD 7 auf Ausgang
PORTD     = &B00000000          ' alle LEDs off

Mybyte = 1                      ' Startwert &B00000001

Do                              ' Beginn Mainloop
     PORTD = Mybyte             ' Ausgabe
     Waitms 100                 ' Warte kurz
     Rotate Mybyte, Left        ' Bit laufen lassen (Rotation)
Loop                            ' Ende Mainloop
End                             ' Programmende
'-----------------------------------------------------------
```

7.2 Eingaben in BASCOM

Eingabeoperationen werden in BASCOM ebenfalls durch Wertzuweisungen oder auch durch Vergleichsoperationen in Bedingungen „direkt“ ausgeführt. Sie können die Ports und I/O-Register wie jede andere Variable handhaben.

Beispiele:

```
'eine Ausgabe an PORT B
PORTB = Mybyte
'eine Eingabe von PIN B
Mybyte = PINB
'eine Vergleichsoperation, bei der von PIN B.0 gelesen wird
If PINB.0 = 0 Then
'lesen eines I/O-Registers (Timer2)
Mybyte = TCNT2
```

Zu beachten sind wiederum die Steuerregister `DDR x`, in denen die Datenrichtung festgelegt wird. Eine besondere Form der Ausgabe ist der Befehl `config`. Dieser realisiert mehr oder weniger komplexe Initialisierungen, welche letztlich Ausgaben in den entsprechenden I/O-Registern sind.

```
'entspricht DDRB = &B00000000
config PORTB = Input
'entspricht DDRB = &B11111111
config PORTD = Output
```

In den folgenden beiden Beispielen wird das Programm “intelligenter Lichtschalter” in der Variante 1 ohne Verwendung von Variablen erstellt; in der Variante 2 unter Verwendung von Variablen.

```
' VARIANTE 1 ohne Variable
'-----------------------------------------------------------
' Titel         : intelligenter Lichtschalter für myAVR Board
'-----------------------------------------------------------
' Funktion      : LED bei Tastendruck ON
' Schaltung     : Taste an PD.5, rote LED an PD.6
' Prozessor     : ATmega8  3,6864 MHz
' Version:      : 1.1  , 20.07.2010
' Autor         : Michael Gesell
'-----------------------------------------------------------
$regfile = "m8def.dat"   ' Prozessortyp ATmega8
$crystal = 3686400       ' Taktrate
DDRD  = &B01000000       ' PD 6 auf Ausgang
PORTD = &B00100000

Do
     If PIND.5 = 0 Then  ' Eingabe PORT D.5!
          PORTD.6 = 1    ' dann PD 6 einschalten
     Else ' sonst
          PORTD.6 = 0    ' PD 6 anschalten
     End If
Loop
End
'-----------------------------------------------------------


' VARIANTE 2 mit Variable
'-----------------------------------------------------------
' Titel         : intelligenter Lichtschalter für myAVR Board
'-----------------------------------------------------------
' Funktion      : LED bei Tastendruck ON
' Schaltung     : Taste an PD.5, rote LED an PD.6
' Prozessor     : ATmega8  3,6864 MHz
' Version:      : 1.2  , 20.07.2010
' Autor         : Michael Gesell
'-----------------------------------------------------------
$regfile = "m8def.dat"        ' Prozessortyp ATmega8
$crystal = 3686400            ' Taktrate
DDRD  = &B01000000            ' PD 6 auf Ausgang
PORTD = &B00100000

Dim Mybyte as Byte            ' ein Byte als Variable

Do
     Mybyte = PIND            ' Eingabe von PORT D !
     If Mybyte.5 = 0 Then     ' Wenn PD 5 angeschalten
          PORTD.6 = 1         ' dann PD 6 einschalten
     Else                     ' sonst
          PORTD.6 = 0         ' PD 7 anschalten
     End If
Loop
End
'-----------------------------------------------------------
```

7.3 Ausgaben in BASCOM

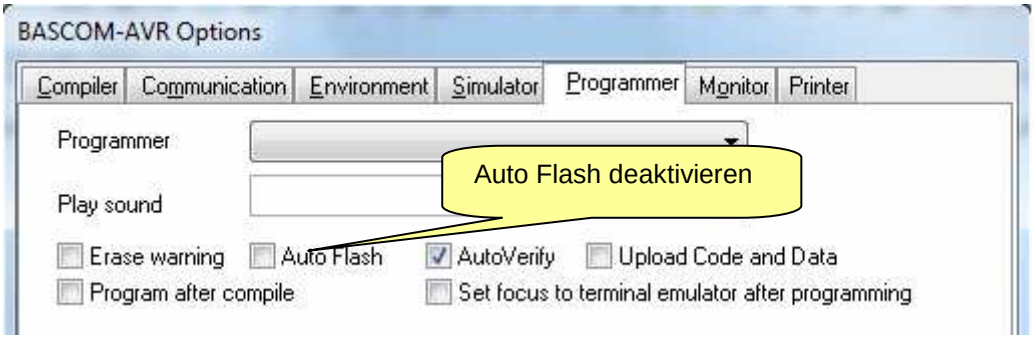
Digitale Ausgaben können in BASCOM genau wie in C als Wertzuweisung an den Port (`Port x = wert`) oder das gewünschte Bit (`Port x.n = 0|1`) erfolgen. Der Compiler setzt die Befehle entsprechend um. Zu beachten ist, dass trotzdem über die Steuerregister `DDR x` die Datenrichtung vorher festgelegt werden muss.

Beispiel:

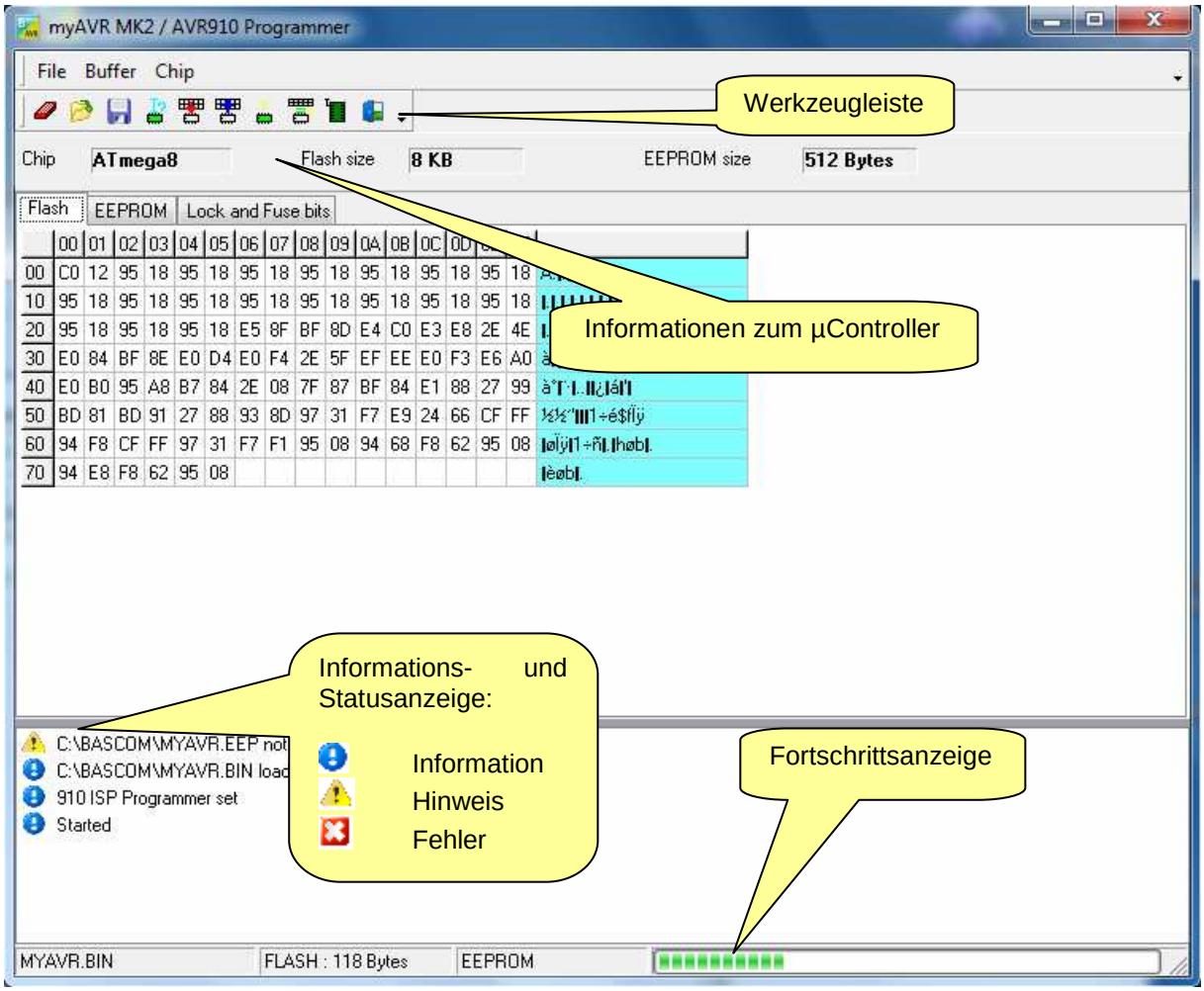
```
DDRB  = &B11111111       ' gesamter Port B auf Ausgang
PORTB = &B00000001       ' Port B.0 auf 1

Config PORTB = Output    ' gesamter Port B auf Ausgang
PORTB = &B00000001       ' Port B.0 auf 1

DDRB.0    = 1  ' Port B.0 auf Ausgang
PORTB.0   = 1  ' Port B.0 auf 1

'-----------------------------------------------------------
' Titel          : Beispiel "Hallo Welt" für myAVR Board
' Datei          : hallowelt.bas
'-----------------------------------------------------------
' Funktion       : LED einschalten
' Schaltung      : PB0 an LED
' Prozessor      : ATmega8  3,6864 MHz
' Version:       : 1.0  , 22.07.2010
' Autor          : Michael Gesell
'-----------------------------------------------------------
$regfile = "m8def.dat"            ' Prozessortyp ATmega8
$crystal = 3686400                ' Taktrate

DDRB = &B00000001                 ' Port B.0 auf Ausgang

Do                                ' Begin Mainloop
   PORTB.0 = 1                    ' PB 0 LED an
Loop                              ' Ende Mainloop
End                               ' Programmende
'-----------------------------------------------------------
```

8 Manual Program

Mit Manual Program kann man zusätzliche Funktionen nutzen, wie z.B. Flash und EEPROM auslesen, editieren, löschen,… und auch Fuse und Lockbits ändern. Dazu muss bei der Programmereinstellung die Option „Auto Flash“ deaktiviert werden.



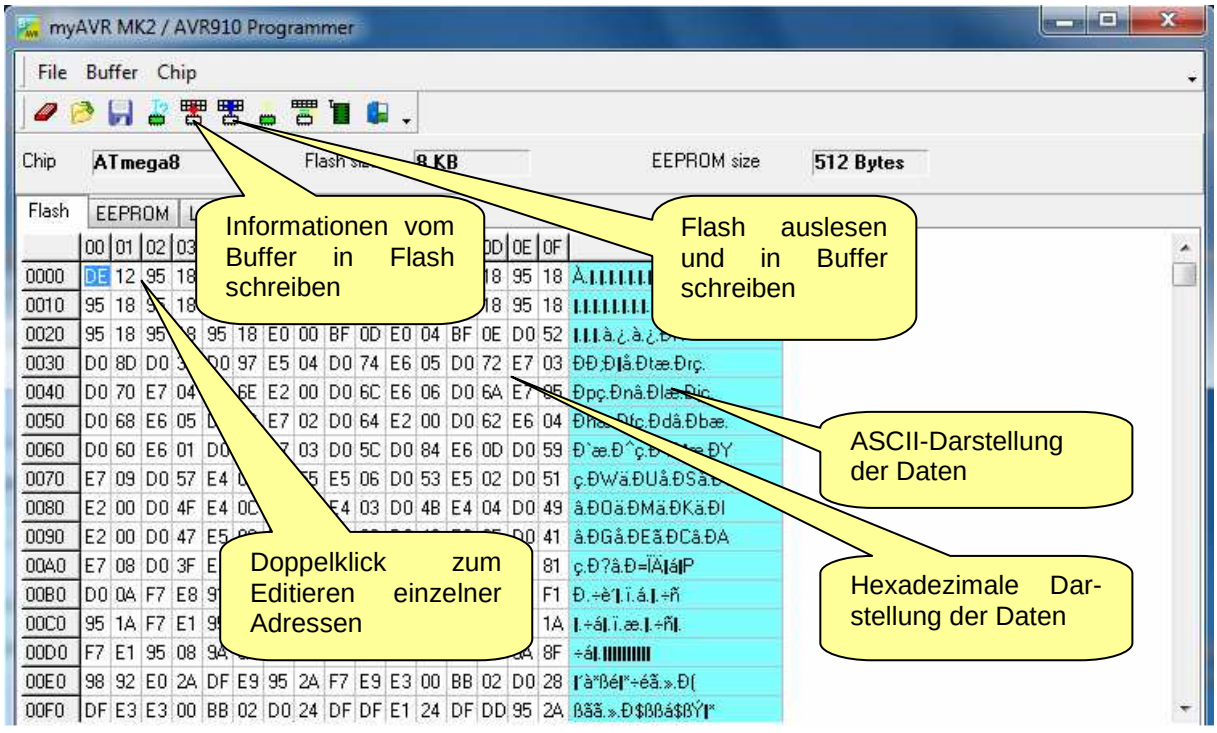
Programmierfenster



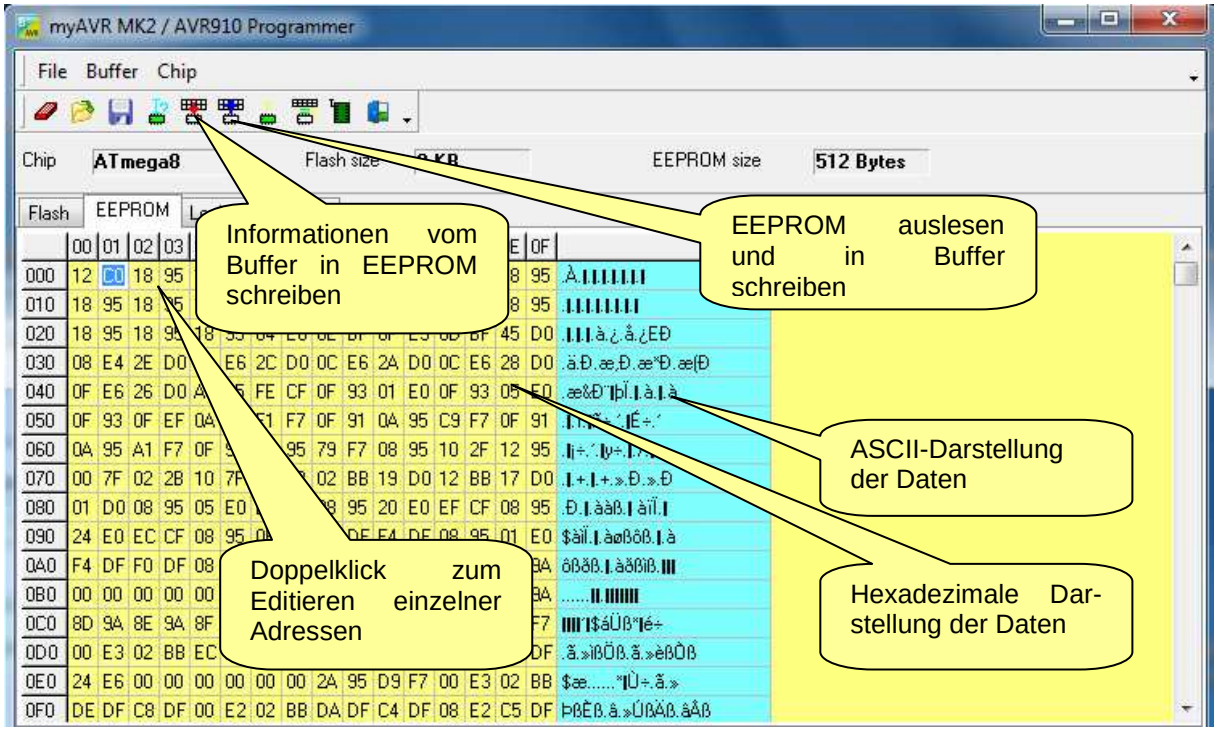
Werkzeugleiste

Icon	Funktion
	Buffer löschen
	Datei in Buffer laden
	Bufferinhalt in Datei speichern
	Controller-ID auslesen
	Bufferinhalt in Flash bzw. EEPROM schreiben (abhängig von der jeweiligen Auswahl, ob Flash oder EEPROM)
	Flash bzw. EEPROM auslesen und in Buffer schreiben (abhängig von der jeweiligen Auswahl, ob Flash oder EEPROM)
	Prüfen, ob der Flash bzw. EEPROM leer ist (abhängig von der jeweiligen Auswahl, ob Flash oder EEPROM)
	Bufferinhalt mit Flash bzw. EEPROM vergleichen (abhängig von der jeweiligen Auswahl, ob Flash oder EEPROM)
	Chip löschen und Bufferinhalt schreiben anschließend Rückkehr zum Hauptfenster
	Verbindung zum Programmer trennen anschließend Rückkehr zum Hauptfenster

Flash



EEPROM



Fuse und Lockbits

ACHTUNG! Ein Ändern der Fuse und Lockbits kann zur Folge haben, dass der µController nicht mehr programmierbar ist. Diese Einstellungen sind nur für User gedacht, welche wissen, was Sie da ändern.