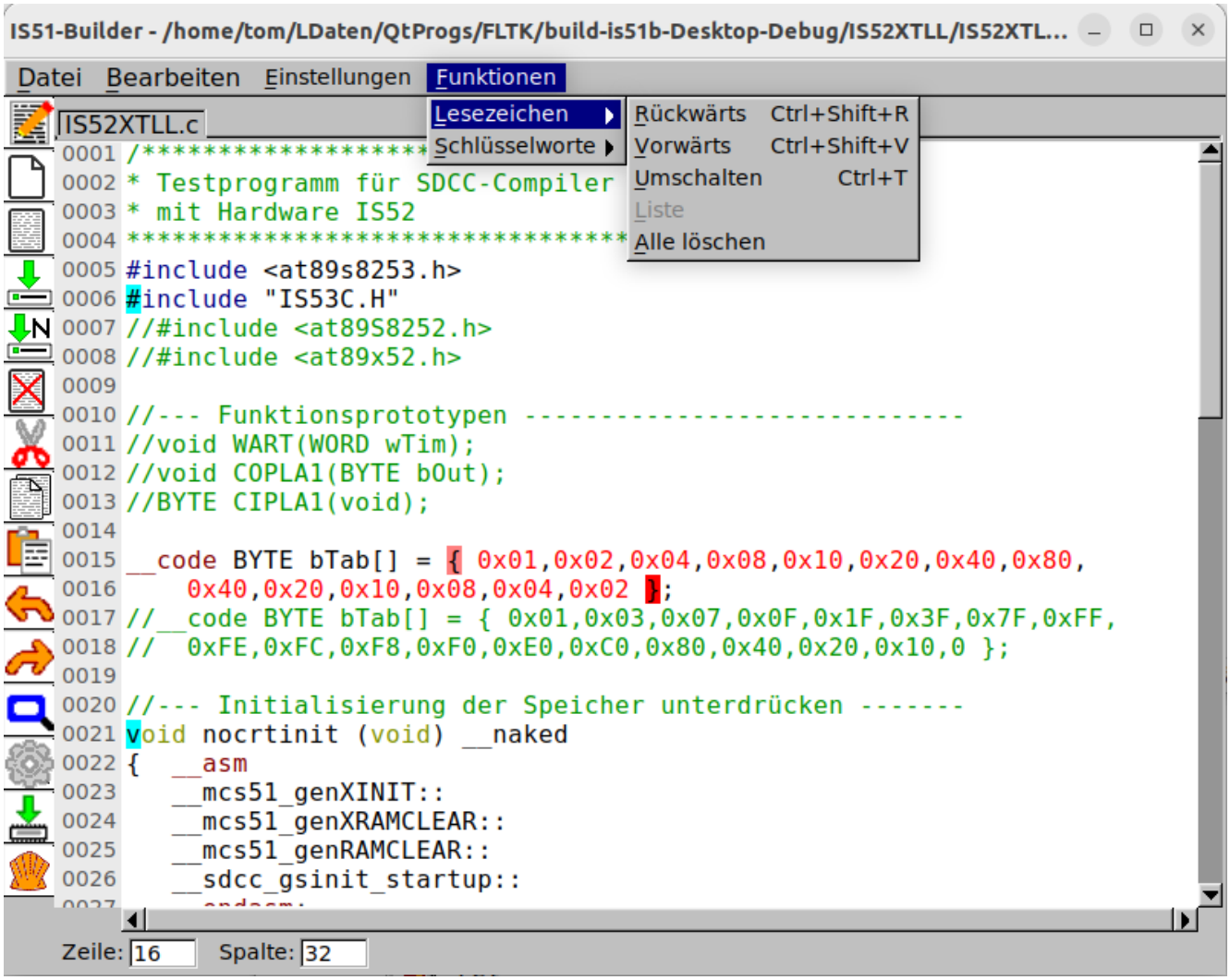
IS51-Builder

Notizen für Anwender



Stand: 25. März 2025
Tom Amann

Inhaltsverzeichnis

1 Einleitung

Der IS51-Builder ist eine Integrierte Entwicklungsumgebung (Integrated Development Environment – IDE) für die Mikrocontrollerfamilie MCS51 und speziell in Verbindung mit der Hardware der verschiedenen IS5xx-Platinen und dem SDC-Compiler. Es ist damit aber auch möglich Programme für andere Prozessoren und Mikrocontroller zu schreiben. Der Debugger wird aber nur mit MCS51 arbeiten Da man aus der IDE auch Programme starten kann, lassen sich natürlich auch Debugger anderer Prozessoren/Controller ausführen.

Das Grundkonzept der IDE ist möglichst einfach in der Bedienung zu sein und den Anwender bei seiner Arbeit möglichst wenig zu behindern. Bei vielen modernen IDE's wird man von gut gemeinten Hilfen förmlich erschlagen und verbringt einen guten Teil der Zeit damit die Folgen der Hilfestellungen wieder zu beseitigen. Meist kann man die Hilfe auch abschalten, aber dann hat man keine Hilfe mehr.

Ich versuche hier Hilfen zur Verfügung zu stellen, sie aber nur nach Aufforderung zu geben. Dadurch vermeide ich, von der Hilfe behindert zu werden.

Der Editor mit seinen Funktionen wird öffentlich sein, der Debugger nicht, da er sich auf Hardware bezieht die, ausser mir, niemand hat.

Das Programm ist in C++ und dem FLTK Framework unter und für Linux geschrieben und statisch gelinkt. Dadurch muss der Anwender kein eigenes Rahmenwerk dafür installieren. Das Programm einfach in ein Verzeichnis kopieren und dort starten.

Der Editor ist der aus den FLTK-Beispielen, erweitert um Multifile-, Lesezeichen-, Klammerfinde- und erweiterte Suchfunktionen.

Die Werkzeugleiste befindet sich am linken Bildrand, um Platz für mehr Zeilen im Editor zu schaffen.

Durch Rechtsklick im Editorfenster öffnet sich ein lokales Menü, in dem sich Hilfsfunktionen finden.

Öfter läuft beim Syntaxhighlight noch etwas schief, in dem Fall im Menü das highlighting deaktivieren und wieder aktivieren.

2 Programm ausführen

Das Programm wurde statisch gelinkt, dadurch ist alles was es benötigt integriert. Es muß nichts dafür installiert werden. Es kann einfach nur gestartet werden. In der Konsole mit dem Befehl: „./is51b“ oder durch anklicken des Symbol in einer grafischen Oberfläche.

Das Zahnradsymbol dient zum übersetzen (compilieren/assemblieren) der momentan aktiven Datei. Der erste Buchstabe des Dateiextend entscheidet über die Art der Übersetzung. Ist der Buchstabe ein „A“ oder „a“ wird die Datei assembliert, bei „C“ oder „c“ wird compiliert. Dazu werden der Compiler/Assembler des SDCC-Paketes verwendet. Es wird nur diese eine Datei übersetzt. Meldungen des Übersetzers werden in der Datei: „dateiname.log“ gespeichert. Fehlermeldungen landen in der Datei: „dateiname.err“. Dabei ist „dateiname“ der Name der aktiven Datei im Editor.

2.1 Einzeldateien

Im Modus für Einzeldateien können die nicht mehr gebrauchten Zwischendateien nicht automatisch gelöscht werden. Dafür ist es aber möglich sich ein einfaches makefile mit der benötigten Funktion zu erstellen.

Makefile:

Hinweis: zum löschen der Dateien "make clean CFL=Dateiname ohne Extend" aufrufen.

clean:

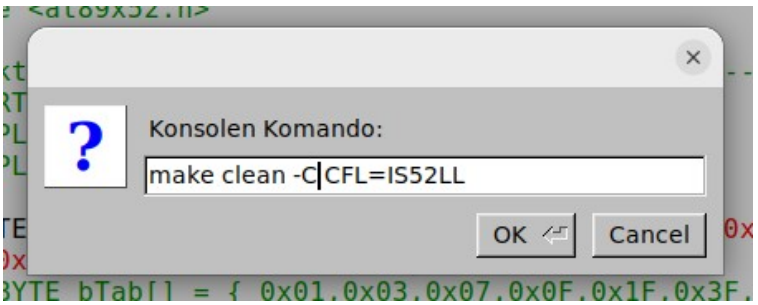
    $(RM) $(CFL).adb $(CFL).asm $(CFL).cdb $(CFL).ihx

    $(RM) $(CFL).lk $(CFL).lst $(CFL).map $(CFL).mem

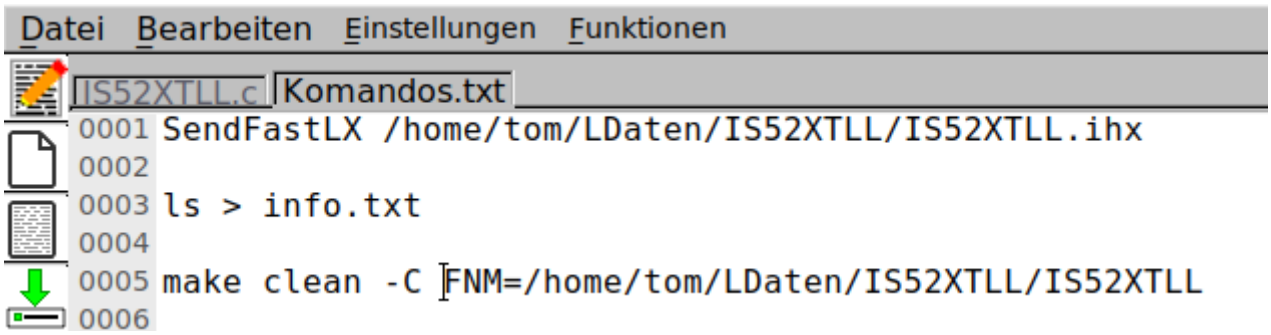
    $(RM) $(CFL).omf $(CFL).rel $(CFL).rst $(CFL).sym

    $(RM) $(CFL).err $(CFL).log

Aufgerufen kann „make clean“ über die Shell im Editor werden.



Um das(die) Kommando(s) nicht jedes mal neu eintippen zu müssen, kann man sie in eine eigene Textdatei schreiben und bei Bedarf aus dieser in das Eingabefenster kopieren.



2.2 Projekte mit mehreren Dateien

Um mehrere Dateien zum bilden eines Zielprogramms zusammen zu verarbeiten wird einmalig ein makefile für das Projekt erstellt und kann dann mehrfach benutzt werden. Das makefile wird vorteilhaft im Projektverzeichnis angelegt und wird dann mit „make“ aufgerufen. Braucht man mehrere makefiles mit unterschiedlichen Funktionen im gleichen Verzeichnis, können die makefiles unterschiedliche Namen haben und unter diesen Namen von make bedient werden. Aufgerufen werden sie dann unter der Angabe dieses Namens:

„make -f Dateiname“ – zum Beispiel: „make -f make1“

Innerhalb des makefiles kann nun auch ein Abschnitt „clean“ zum löschen nicht mehr benötigter Dateien sein. Dieser Abschnitt wird dann durch „make clean“ ausgeführt.

Um make beim Aufruf den Namen einer Datei mitzuteilen, kann dieser als Aufrufparameter übergeben werden:

make clean -C FNM=/home/tom/Ldaten/IS52XTLL/IS52XTLL

übergibt an make die Variable FNM (File NaMe), welche nun innerhalb von make benutzbar ist. Innerhalb wird diese Variable dann als „$(FNM)“ Verwendung finden und steht in diesem Fall für:

/home/tom/Ldaten/IS52XTLL/IS52XTLL

Im folgenden makefile wird die in $(FNM) übergebene C-Datei und die zugehörige Assemblerdatei „Startup.asd“ zum Hexfile mit dem übergebenen Namen und dem Extend „.hex“ verarbeitet.

Beispiel:

```
################################################################################
# GNU Makefile demonstrating combination of C and assembly source files
# File Name: makefile
# All targets in the makefile are processed sequentially by SDCC
# To create the file ‘file.hex’ using GNU make, just execute ‘make’
################################################################################
# The following lines defines additional directories to search for include files
#INCLUDES := -I”C:\Your Directory\Lab3\”

#Here are the sourcefiles used for the project
#CFX = weisnichtwas            hier können auch weitere Quelldateien eingefügt werden.
CFL = $(FNM)
AFL = Startup

#MODEL = --model-large
MODEL = --model-small

# The following line defines flags given to the SDCC C compiler
CFLAGS := -c --debug --verbose $(MODEL) $(INCLUDES)

# The following line defines flags given to the SDCC linker
# Non-specific: –code-loc 0x6000 –xram-loc 0xB000 –model-large
LFLAGS := --debug --verbose --code-loc 0x8100 --code-size 0x8000 --xram-loc 0xC000 --xram-size 0x4000 $(MODEL)

# The following line specifies the default target(s) to build
#all: file.hex

# The following line specifies the object files that are to be linked together
OBJECTS := $(CFL).rel $(AFL).rel
```

```
# The following lines define a rule that sends the object files through the linker to
# create file.ihx which then has to be processed by packihx to create file.hex
file.hex : $(OBJECTS)
	@echo "Linker:" >> $(CFL).log
	sdcc $(LFLAGS) $^ >> $(CFL).log 2>> $(CFL).err
	packihx $(CFL).ihx > $(CFL).hex

# The following rule sends each C file through the preprocessor and creates the asm file
# that is then assembled to create the rel file.
%.rel : %.c
	@echo "Compiler:" > $(CFL).log 2> $(CFL).err
	sdcc $(CFLAGS) $< >> $(CFL).log 2>> $(CFL).err

# The following rule sends each asm file (Not the asm files created by SDCC as an
# intermediate output of the compilation process.) through the assembler to create a rel
# file.
%.rel : %.asd
	@echo "Assembler:" >> $(CFL).log
	sdas8051 -plosgff $< >> $(CFL).log 2>> $(CFL).err

# The following rule will clean all the derived objects from your directory.  This will
# save you from accidentally tying 'rm *' if you are developing on a UNIX platform.
# Hinweis: zum löschen der Dateien "make clean" aufrufen.
clean:
	$(RM) $(AFL).lst $(AFL).rel $(AFL).rst $(AFL).sym
	$(RM) $(CFL).adb $(CFL).asm $(CFL).cdb $(CFL).ihx
	$(RM) $(CFL).lk  $(CFL).lst $(CFL).map $(CFL).mem
	$(RM) $(CFL).omf $(CFL).rel $(CFL).rst $(CFL).sym
	$(RM) $(CFL).err $(CFL).log
```

Fehlermeldungen werden in einer Datei „übergebener Name.err“, Standardmeldungen in „übergebener Name.log“ gespeichert. Ist die Fehlerdatei leer, so hat SDCC keinen Fehler erkannt.
Mit „clean“ werden alle dabei entstandenen Zwischendateien gelöscht. Übrig bleiben nur die Quelldateien und die Hexdatei. „make clean -C Dateiname-ohne-Extend“. Der Aufruf von make über

die IDE macht das automatisch, um den korrekten Inhalt des makefile muss man sich selbst kümmern.

2.3 Download zum Zielsystem

Unter Download wird das erzeugte Hexfile über USB zur IS5x übertragen. Da sonst niemand diese Hardware hat ist deren Beschreibung nicht wichtig.

Für abweichende Zielhardware kann das/die entsprechende(n) Kommando(s) in der Shell gegeben werden. Um nicht so viel tippen zu müssen kann man die erforderlichen Kommandos auch in der Kommandodatei ablegen und von dort in die Shell kopieren.

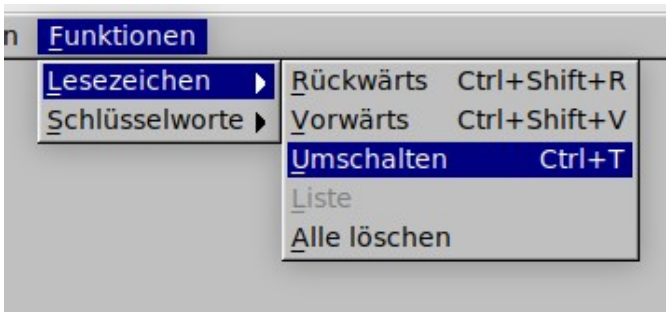
3 Kurze Übersicht

Von den üblichen Tastenkommandos für Editoren sind die meisten umgesetzt. Die zugehörigen Funktionen sind auch über das Menü erreichbar, dort finden sich auch die zugeordneten Tastenkürzel.

3.1 Klammerpaare

Wird eine Klammer mit der Maus oder der Tastatur markiert, wird sie und die zugehörige Klammer rot markiert. Nachdem irgend etwas am Text geändert ist, verschwinden auch die Markierungen. Es werden nur runde- „()“, geschweifte- „{}“ und eckige „[]“ Klammern berücksichtigt, da die spitzen Klammern auch ungepaart im Quelltext vorkommen können.

3.2 Lesezeichen



Im Text können an beliebigen Stellen Lesezeichen gesetzt werden. Durch Umschalten wird ein Lesezeichen gesetzt, oder falls bereits vorhanden, entfernt. Mit Rückwärts und Vorwärts geht es zum jeweils vorigen oder folgenden Lese-zeichen. Die Arbeitsweise ist dabei modulo, wenn ein Ende erreicht ist, wird am anderen Ende weiter gemacht.

Die Liste um ein Lesezeichen gezielt aufzusuchen ist noch nicht realisiert.

Die Funktion „Alle löschen“, löscht alle Lesezeichen unwiderruflich. Die Funktion hat kein Tastenkürzel, damit sie nicht versehentlich aufgerufen wird.

3.3 Die Werkzeugleiste

Datei - Neu

Menü: Datei – Neu (strg+n) Symbol: 

Erstellt eine neue Datei mit dem Namen „neu.txt“

Datei - Öffnen

Menü: Datei – Öffnen (strg+o) 

Öffnet eine bereits bestehende Datei

Datei - Speichern

Menü: Datei – Speichern (strg+s) 

Speichert eine bearbeitete Datei.

Datei – Speichern unter

Menü: Datei – Speichern unter. (strg+shift+s) 

Speichert die aktuelle Datei unter einem neuen Namen.

Datei - Schließen

Menü: Datei schließen (strg+w) 

Schließt die aktuelle Datei.

Bearbeiten - Ausschneiden

Menü: Bearbeiten Ausschneiden (strg+x) 

Markierten Text ausschneiden und in die Zwischenablage legen.

Bearbeiten - Kopieren

Menü: Bearbeiten Kopieren (strg+c) 

Markierten Bereich in die Zwischenablage legen.

Bearbeiten - Einfügen 

Menü: Bearbeiten Einfügen (strg+v)

Inhalt der Zwischenablage einfügen.

Bearbeiten – Rückgängig (undo)

Menü: Bearbeiten Rückgängig (strg+z)

Letzte Funktion zurücknehmen.

Bearbeiten – Wiederherstellen (redo)

Menü: Bearbeiten Wiederherstellen (strg+y)

Letztes Undo zurücknehmen.

Bearbeiten – suchen

Menü: Bearbeiten suchen (strg+f)

Text in Quelldatei finden/ersetzen.

Bearbeiten – Übersetzen (Compiler/Assembler)

Menü: Bearbeiten Übersetzen

Quelldatei assemblieren/compilieren.

Bearbeiten – Download

Menü: Bearbeiten Download

Bearbeiten – Shell

Menü: Bearbeiten Shell

Ausführen von Konsolenprogrammen.

3.4 Menübefehle

Im Menü erreicht man auch die Funktionen, welche sich nicht in der Werkzeugleiste finden. Diese Funktionen werden zum Teil auch über Tastenkürzel aufgerufen.

3.4.1 Datei

Neu (Ctrl+N)

Erstellt und öffnet im Verzeichnis von is51b eine neue und leere Textdatei mit dem Namen „neu.txt“. Ist dort bereits eine Datei dieses Namens vorhanden, wird diese geöffnet. Dadurch kann eine Rahmendatei erstellt und bei jeder neuen Datei benutzt werden. Wird die Datei woanders gebraucht, kann sie mit „Speichern unter“ im gewünschten Verzeichnis mit dem erforderlichen Namen gespeichert werden. Der Editor übernimmt den neuen Pfad und Namen für die Datei. Sind bereits Dateien geöffnet, wird der Pfad der zuletzt aktiven Datei für die Neue verwendet.

Öffnen (Ctrl+O)

Zeigt einen Dialog, in dem eine bereits vorhandene Datei wählbar ist. Um die Auswahl auf bestimmte Dateitypen zu beschränken, kann der Dateifilter eingestellt werden. Grundeinstellung ist hier Alle-Dateien (*.*). Als Verzeichnis wird das der momentan aktuellen Datei vorgeschlagen. Auch hier ist es möglich vordefinierte Rahmendateien zu nutzen.

Speichern (Ctrl+S)

Speichert die Datei an ihrem ursprünglichen Ort mit dem aktuellen Namen auf Datenträger.

Speichern unter (Ctrl+Shift+S)

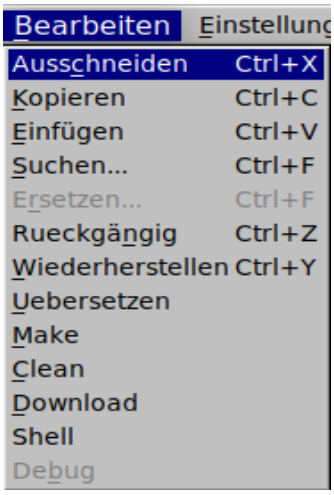
Öffnet einen Dialog, in dem ein neuer Ort und ein neuer Name wählbar ist. Hiermit lassen sich neu erstellte Dateien ihrem Ziel zuordnen.

Schließen (Ctrl+W)

Die aktuelle Datei wird geschlossen. Wurde sie seit dem letzten speicher geändert, Öffnet sich ein Dialog in dem gefragt wird, ob sie gespeichert werden soll.

Beenden (Ctrl+Q)

Beendet den IS51-Builder. Sind noch Dateien offen, die nach dem letzten speichern geändert wurden, erfolgt für jede dieser Dateien eine Abfrage ob sie gespeichert werden soll.



3.4.2 Bearbeiten

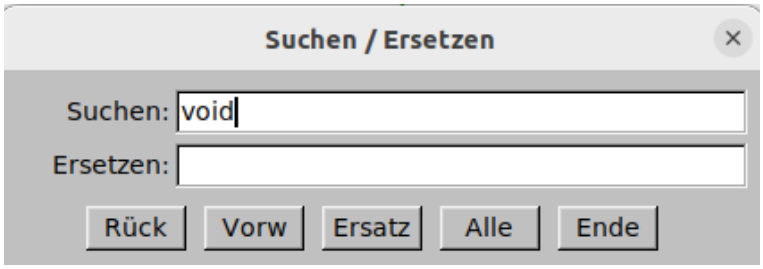
Ausschneiden (Ctrl+X)
Entfernt den markierten Bereich aus dem Text und legt ihn in die Zwischenablage.

Kopieren (Ctrl+C)

Kopiert den markierten Bereich aus dem Text und legt ihn in die Zwischenablage.

Einfügen (Ctrl+V)
Fügt den Inhalt der Zwischenablage an der aktuellen Cursorposition ein.

Suchen (Ctrl+F)



Sucht im aktuellen Text nach einem bestimmten Abschnitt, der im Eingabefeld „Suchen“ vorgegeben wird. Wird im Text ein Abschnitt / Wort markiert, z.B: durch Doppelklick, wird dieser markierte Text beim öffnen des Dialogs als Suchtext übernommen, sonst der zuletzt benutzte Suchtext. Beim ersten Aufruf des Dialoges ist der letzte Suchtext noch leer. Durch „Rück“ und „Vorw“ wird der jeweils vorherige oder folgende Treffer angewählt. „Alle“ ist noch nicht verfügbar (alle Vorkommen des Suchtextes markieren) und springt zum letzten Vorkommen des Suchtext.

Ist ein Ersatztext vorhanden, wird durch „Ersatz“ das aktuell gefundene Suchwort durch diesen ersetzt. „Alle“ ersetzt alle Suchworte von der aktuellen Cursorposition bis zum Ende des Textes. Um wirklich alle Vorkommen im Text zu ersetzen ist der Cursor auf den Dateianfang zu setzen. Die ursprüngliche Cursorposition kann durch ein Lesezeichen (strg+t) festgehalten werden, so kommt man schnell zur alten Position zurück.

Vor dem ersetzen aller Vorkommen empfiehlt es sich die Datei zu speichern, da man zwar alle Vorkommen gemeinsam ersetzen, aber dies nur einzeln rückgängig machen kann.

Durch „Ende“ wird der Dialog beendet, dabei bleibt der letzte Suchtext erhalten und beim erneuten Aufruf wieder als Suchtext vorgegeben (sofern kein Bereich im Quelltext markiert ist).

Rückgängig (Ctrl+Z)
Macht die letzte Aktion rückgängig. Erneutes aktivieren die Aktion davor und so weiter. Bis zum Startpunkt des Editors.

Wiederherstellen (Ctrl+Y)
Stellt die letzte rückgängig gemachte Aktion wieder her. Erneutes aktivieren die Aktion davor und so weiter. Bis zum Startpunkt von „Rückgängig“.

Übersetzen
Assemblieren/compilieren der aktuellen Datei der IDE. Die Art des übersetzens ist abhängig vom ersten Buchstaben des Dateiextend. Ist dieser ein „A“ oder „a“ wird assembliert (z.B: xxx.asm, xxx.asd, ...), ist es ein „C“ oder „c“ wird compiliert (z.B: xxx.c, xxx.c51, ...).

Wurde die aktuelle Datei seit dem letzten speichern geändert, erscheint ein Dialog in dem man speichern wählen kann.

Make
Ruft das „makefile“ auf.

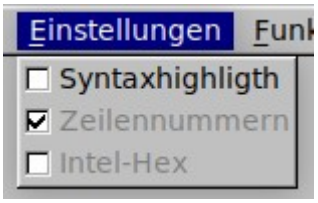
Wurde die aktuelle Datei seit dem letzten speichern geändert, erscheint ein Dialog in dem man speichern wählen kann.

Clean
Ruft „make clean“ auf, um die nicht mehr benötigten Zwischendateien zu löschen. Es ist vorteilhaft sich zwei makefiles, für C oder Assembler, zurechtzulegen. Für C-Programme ist die x.asm nur eine Zwischendatei, für Assembler ist es möglicherweise die Quelldatei und darf nicht gelöscht werden. Es ist vorteilhaft für Assemblerquelldateien den Extend “.asd“ (aSSEMBLER sdCC) zu vergeben, damit sie nicht versehentlich gelöscht werden. Der Builder erwartet ein Makefile „aclean.mak“ für Assembler- und „cclean.mak“ für C-Quelldateien.

Download
Ruft den Download der Intel-Hex Datei über USB auf.

Shell
Ermöglicht die Eingabe von Befehlen in die Konsole.

3.4.3 Einstellungen



Syntaxhighlight:

Aktivieren und deaktivieren des Syntaxhighlight im aktiven Editorfenster.

Zeilennummern

Wird noch nicht bedient. Die Zeilennummern werden immer angezeigt.

Intel-Hex

Wird noch nicht bedient. Intel-Hex Editor zum bearbeiten von Hex-Dateien.

3.4.4 Funktionen

Lesezeichen

Bearbeiten (umschalten, anspringen, löschen) der Lesezeichen.

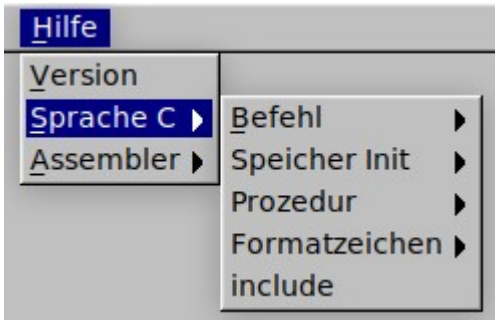
Zeile

Zu einer gegebenen Zeile springen.

Schlüsselworte

Wird noch nicht bedient. Hilfslisten mit oft benötigten Worten um sie daraus in den Quelltext kopieren zu können. Dies zeigt Mnemonics, C-Befehle, SFR, SFR-Bit, Registernamen und vieles mehr, um sie nicht in verschiedenen Dokumenten nachschlagen zu müssen.

3.4.5 Hilfe

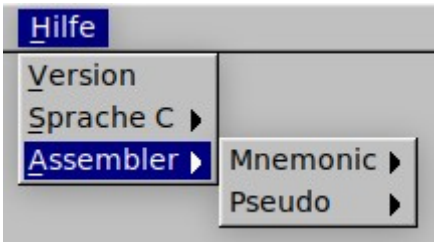


Version

Anzeige des vorliegenden Programmstatus.

Sprache C
Hilfestellung für die Programmiersprache C.

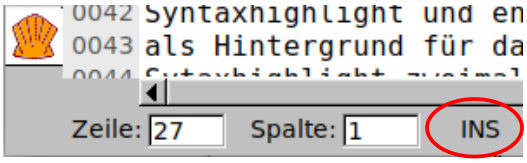
Assembler
Hilfestellung für die Programmiersprache Assembler.



3.5 Bedienung

Viele Funktionen gleichen denen anderer Editoren, einige weichen davon ab. Hier sind die abweichenden Funktionen der Bedienung des Editors im IS51-Builder beschrieben.

Durch die Einfügen-Taste der Tastatur wird der Einfügemodus umgeschaltet. Er wechselt zwischen „INS“ (INSert – Einfügen) und „OVR“ (OVeRwrite – Überschreiben). Die entsprechende Anzeige des Modus ist in der Statuszeile rechts neben der Spaltennummer zu finden.



3.5.1 Zeileneinrückung

Anstatt immer nur den linken Rand, oder immer die Position der vorherigen Zeile zu verwenden (was recht nervig sein kann), wird hier eine dreifache Wahl angeboten:

Enter:	Linker Rand
Ctrl+Enter:	Wie vorige Zeile
Alt+Enter:	Wie folgende Zeile

Ist keine vorige- (Textanfang), oder folgende- (Textende) Zeile vorhanden, wird für die neue Zeile der linke Rand gewählt. Hierbei wird, dem Konzept entsprechend, dem Anwender nichts aufgezwungen, sondern er hat die freie Wahl.

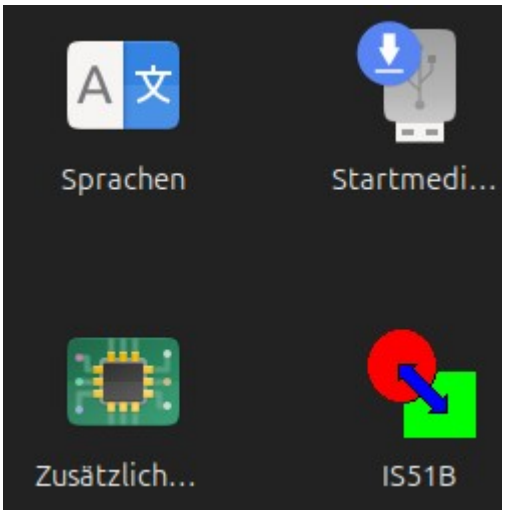
4 Ausführen

Hier finden sich einige Beispiele für den Umgang mit IS51-Builder.

4.1 Startsymbol einrichten

Das ICON des Programms und die Datei „is51b.desktop“ in den versteckten Ordner /home/user/.local/share/applications/ kopieren. Nun kann sie aus den Anwendungen gestartet oder an die Schnellstartleiste angeheftet werden.

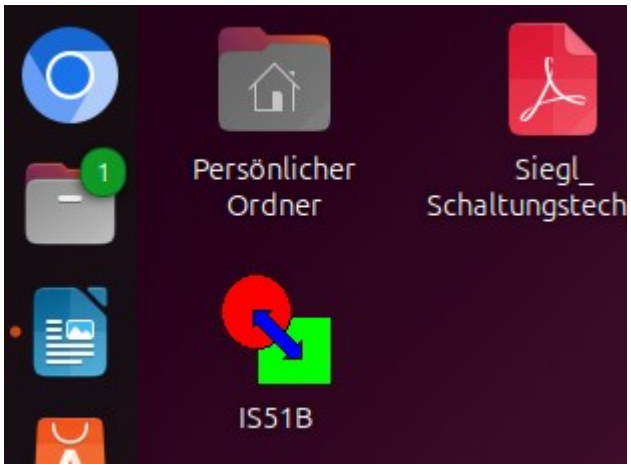
Beispiel – is51b.desktop:




```
[Desktop Entry]
Type=Application
Icon=/home/tom/.local/share/applications/IS51ICO.ico
Name=IS51-Builder
Comment=8051-IDE für SDCC
Exec=/home/tom/LDaten/QtProgs/FLTK/is51b/is51b/is51b
Path=/home/tom/LDaten/QtProgs/FLTK/is51b/
StartupNotify=true
Terminal=false
MimeType=x-directory/normal;inode/directory;
```

4.2 Desktop Startsymbol

Die Datei is51b.desktop in den Desktop-Ordner (Schreibtisch) kopieren, dann das Icon rechts anklicken und „Starten erlauben“ aktivieren. Dadurch wird das Programm über den Desktop als Symbol erreichbar.

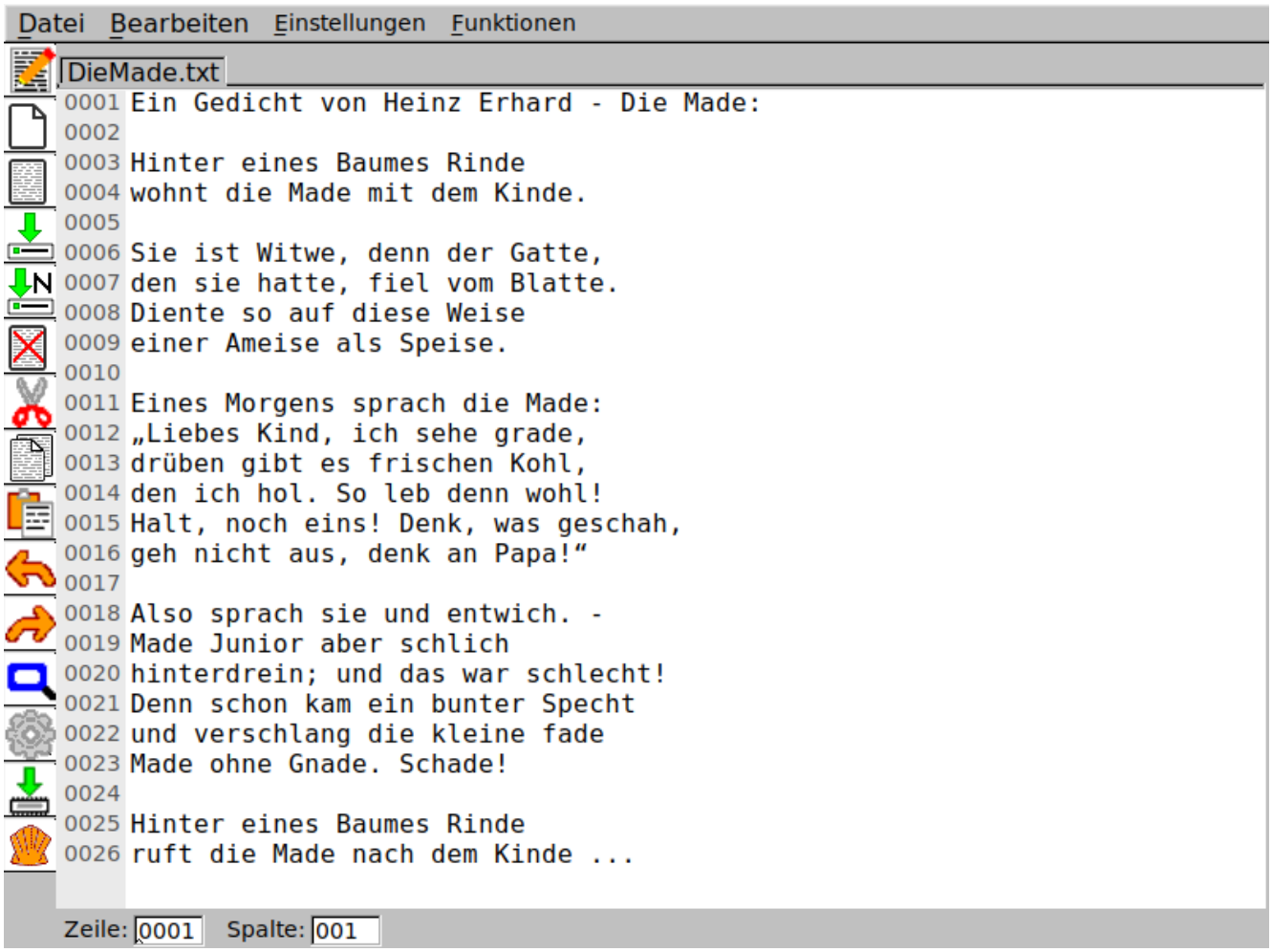


5 Beispiele

Hier finden sich einige Beispiele für den Umgang mit IS51-Builder.

5.1 Einfache Textdatei

Im normalen Textmodus verhält sich der IS51-Builder wie viele andere Editoren auch. Es können Lesezeichen an beliebigen Stellen gesetzt werden. Es werden immer Zeilennummern angezeigt. Die Anzeige der Cursorposition Spalte/Zeile ist unten links in der Statuszeile. Die Funktionen einfügen/überschreiben fehlen noch.



5.2 C-Programmdatei

Hier ein kleines Testprogramm in C für MCS51. Es liest, in einer Enlosschleife, die Stellung von 8 Schaltern an Port P1 ein und gibt diese an die LED‘s an Port P3 aus.

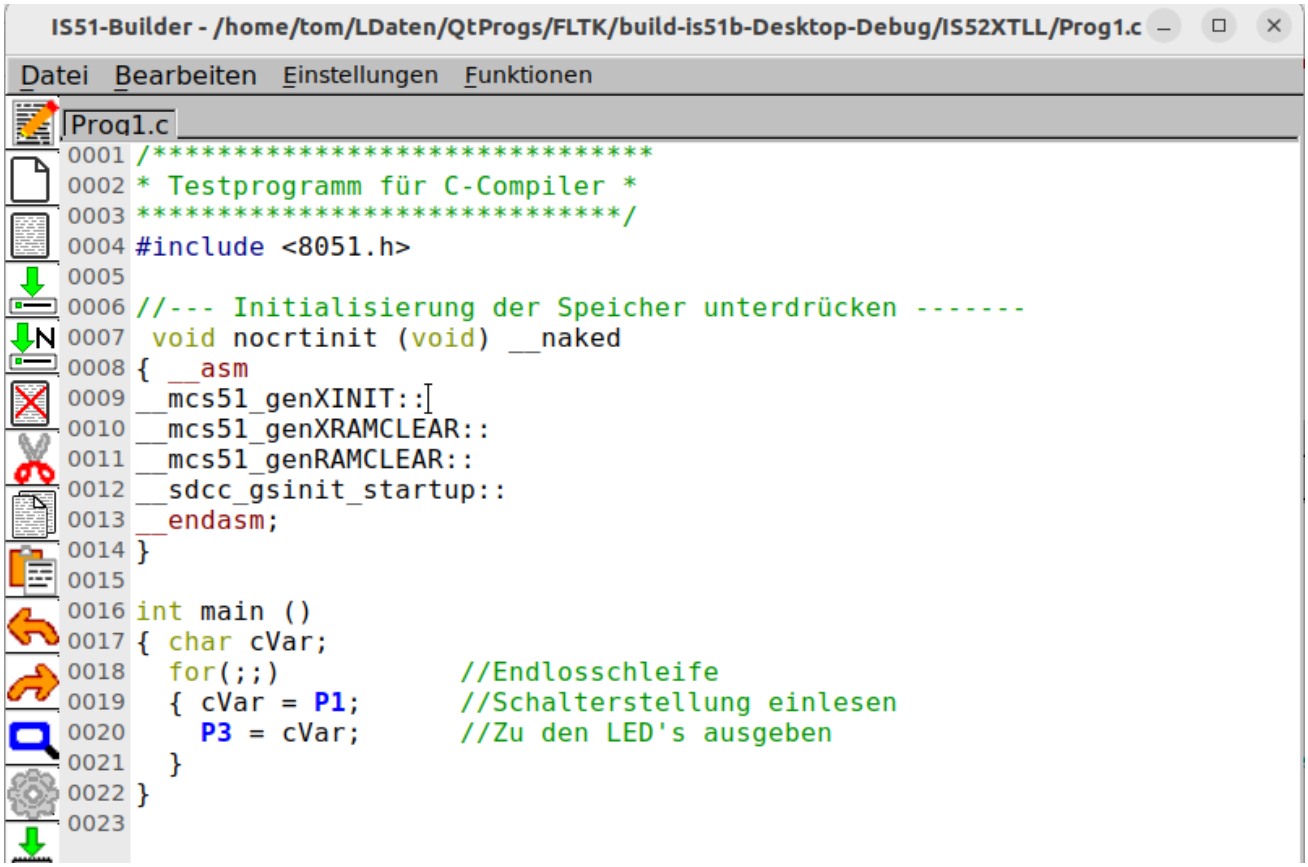
Programm Prog1.c zum kopieren:

```
/*****************************
* Testprogramm für C-Compiler *
*****************************/
#include <8051.h>

//--- Initialisierung der Speicher unterdrücken -------
 void nocrtinit (void) __naked
{ __asm
__mcs51_genXINIT::
__mcs51_genXRAMCLEAR::
__mcs51_genRAMCLEAR::
__sdcc_gsinit_startup::
__endasm;
}

int main ()
{ char cVar;
  for(;;)            //Endlosschleife
  { cVar = P1;       //Schalterstellung einlesen
    P3 = cVar;       //Zu den LED's ausgeben
  }
}
```

Programm Prog1.c im Editor:



Compilieren durch Klick auf das Zahnrad der Werkzeugleiste oder Menü: Bearbeiten – Uebersetzen. Dadurch wird eine ganze Reihe Zwischendateien erzeugt, die bei Nichtgebrauch mit Menü: Bearbeiten – Clean wieder entfernt werden können, sofern ein entsprechendes makefile vorhanden ist.

Daraus erzeugte Intel-Hex Datei:

:03000000020000FB

:03000600020003F2

:03000300020009EF

:060009008590B080FB228F

:00000001FF

Diese HexDatei kann nun zum Zielsystem übertragen und dort ausgeführt werden.

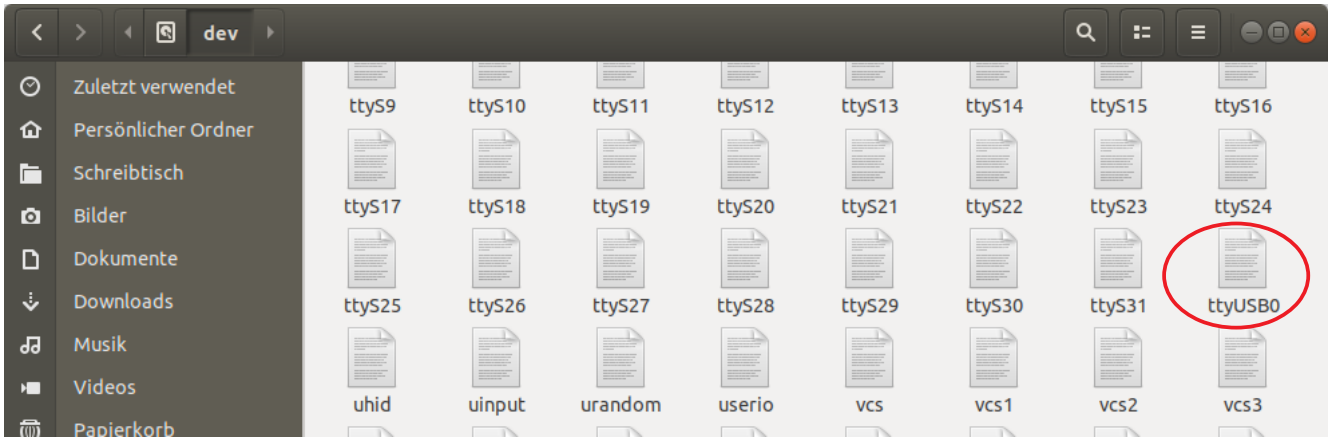
6 USB unter Linux

Unter Linux ist es nicht ganz einfach eine zugekaufte oder selbstgebaute Hardware einzubinden, wenn sie nicht automatisch vom System erkannt wird. Dies betrifft auch die vielen verschiedenen Programmieradapter für Mikrocontroller.

6.1 USB-RS232 Wandler

Die RS232-Schnittstelle ist eine genormte serielle Schnittstelle, welche mit unterschiedlichen Bezeichnungen verbreitet ist. Nach amerikanischer Norm heißt sie RS232, nach europäischer Norm V.24 und nach deutscher Norm DIN 66020. Da heute die meisten PC's keine RS232-Schnittstelle mehr haben, werden billige USB-RS232 Adapter angeboten. Die Hersteller dieser Adapter scheren sich oft nicht viel um die Norm, was die Sache etwas verkompliziert. Doch das ist im Netz vielfach beschrieben und wird hier nicht weiter ausgeführt. Hier geht es um das einbinden und benutzen dieser Adapter.

Beim einstecken eines funktionierenden Adapter an einen USB-Anschluß, wird er vom System erkannt und unter einem bestimmten Namen eingetragen. Diesen Namen herauszufinden ist wichtig, um die Schnittstelle ansprechen zu können. Dazu sieht man sich im Hauptverzeichnis der Festplatte den Inhalt des Ordners „dev" (devices - Geräte) genauer an.



Der hier verwendete Adapter wurde unter dem Namen ttyUSB0 eingetragen, wobei die Nummer 0 eine fortlaufende Nummer ist. Wird zusätzlich ein zweiter, gleicher Adapter angesteckt erhält er die nächst höhere Nummer, also ttyUSB1. Ein Teil meiner Adapter meldet sich mit ttyACMx (Abstract Control Model). Um sicherzustellen dass es auch wirklich der richtige Adapter ist, kann man ihn wieder abstecken und der Eintrag verschwindet.

ttyUSBx = USB-UART Konverter (Hardware), ttyACMx = USB-CDC Programm (Software).

Die Schnittstellen sind auch durch „dmesg“ in einem Terminal ermittelbar:

```
dmesg | grep tty
```

```
tom@tom-TM6595T:~/LDaten$ dmesg | grep tty
[    0.000000] console [tty0] enabled
[    0.708179] 0000:00:16.3: ttyS4 at I/O 0x30b0 (irq = 19, base_baud = 115200) is a 16550A
[27540.663961] cdc_acm 2-1.2:1.0: ttyACM0: USB ACM device
```

Damit ist nun bekannt unter welchem Namen der Adapter angesprochen wird.

Nun ist zu prüfen, ob man die Berechtigung besitzt auf das Gerät zugreifen zu dürfen. Dazu in der Konsole die Gruppenzugehörigkeit ermitteln mit "id":

```
tom@tom-TM6595T:~/LDaten/QtProgs$ id
uid=1000(tom) gid=1000(tom) Gruppen=1000(tom),4(adm),20(dialout),24(cdrom),
27(sudo),30(dip),46(plugdev),118(lpadmin),128(sambashare)
```

Um mit den seriellen Schnittstellen arbeiten zu dürfen muss man sich in der Gruppe „dialout“, und für die USB-Schnittstelle besser auch in der Gruppe „plugdev“ befinden. In meinem Fall ist das hier gegeben, in der Liste sind die Gruppen 20(dialout) und 46(plugdev) aufgeführt. Ist man kein Mitglied der entsprechenden Gruppe, muss man sich dieser anschließen:

Bestehenden Benutzer einer weiteren Gruppe hinzufügen:

Allgemein: sudo usermod -aG GRUPPENNAME BENUTZERNAME

Beispiel: sudo usermod -aG plugdev tom

Als Benutzername ist nicht tom, sondern der Eigene einzusetzen. Danach Linux abmelden und wieder anmelden damit die Änderung wirksam ist.

Nachdem nun der Zugriff auf die Hardware sichergestellt ist, wird jetzt die Software für das Downloaden eingerichtet. Dazu sind die Übertragungsparameter der Zielhardware richtig einzustellen, damit sich die Teilnehmer untereinander verstehen. Welche die richtigen Parameter der Zielhardware sind, ist deren Dokumentation zu entnehmen. Die Parameter für den PC und damit für den IS51-Builder werden mit dem „stty“ Kommando bestimmt:

```
stty 9600 -F /dev/ttyUSB0
```

Hier wird die Baudrate von ttyUSB0 auf 9600 Bit je Sekunde eingestellt. Näheres findet sich in der Dokumentation von „stty“. Sind die richtigen Einstellungen für die Zielhardware ermittelt, werden sie in eine der freien Einstellungen des IS51-Builder übernommen und sind dann auf Mausklick erreichbar.

Gleiches gilt für den eigentlichen Befehl zum Download. Er wird in eine der freien Einstellungen, z.B Senden, übernommen und ist dann auch per Mausklick erreichbar. Der Konsolen-Befehl nennt sich „cat“ und ist geeignet eine Datei über die serielle Schnittstelle zu senden. Im Beispiel wird die Intel-Hex Datei IS52XTLL.ihx aus dem Pfad „/home/tom/LDaten/“ zum Gerät „/dev/ttyUSB0“ gesendet.

```
cat /home/tom/LDaten/IS52XTLL.ihx > /dev/ttyUSB0
```

6.2 Libusb Geräte

Auch hier ist es wichtig das richtige Gerät zu ermitteln. Dazu das Gerät mit einem freien USB-Anschluss des PC verbinden und in der Konsole den Befehl „lsusb“ eingeben (der erste Buchstabe ist ein kleines L, kein großes i).

Ermittelt durch "lsusb":

```
tom@tom-TM6595T:~/LDaten/QtProgs$ lsusb
Bus 002 Device 003: ID 1c7a:0603 LighTuning Technology Inc.
->  Bus 002 Device 004: ID 16d0:0418 MCS
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
```

Mein Programmiergerät (ID 16d0:0418 MCS) ist hier das zweite Gerät: Bus 002 Device 004.

Nun die Rechte der Geräte prüfen. Hier prüfe ich die Geräte am Bus 002:

```
ls -l /dev/bus/usb/002
```

```
tom@tom-TM6595T:~/LDaten$ ls -l /dev/bus/usb/002
crw-rw-r-- 1 root root 189, 128 Mai 11 09:04 001
crw-rw-r-- 1 root root 189, 129 Mai 11 09:04 002
crw-rw-r-- 1 root root 189, 130 Mai 11 09:04 003
->  crw-rw---- 1 root root 189, 131 Mai 11 13:49 004
```

Mein Gerät (Nummer 004) hat Lese/Schreib-Rechte (rw) für den Besitzer und die Gruppe, gehört aber nur zu Gruppe root und erlaubt mir deshalb als Benutzer keinen Zugriff.

6.2.1 Zugriff vorübergehend einrichten

Zum testen des Gerätes und prüfen der Einstellungen können diese vorübergehend verändert werden. Die Änderungen sind sofort wirksam, aber spätestens beim nächsten Systemstart sind sie wieder zurückgestellt.

Da es sich um ein USB-Gerät handelt, ist es in der Gruppe „plugdev“ (Steckbare Geräte) gut aufgehoben. Ändern der Gruppe für das Gerät:

```
sudo chown root:plugdev /dev/bus/usb/002/004
```

Überprüfen der Änderung:

```
ls -l /dev/bus/usb/002
```

```
crw-rw-r-- 1 root root    189, 128 Mai 11 09:04 001
crw-rw-r-- 1 root root    189, 129 Mai 11 09:04 002
crw-rw-r-- 1 root root    189, 130 Mai 11 09:04 003
->  crw-rw---- 1 root plugdev 189, 132 Mai 11 14:15 004
```

Jetzt befindet sich das Gerät in Gruppe „plugdev“ und ich darf als Mitglied dieser Gruppe auch darauf zugreifen. Sollten dem Gerät die nötigen Schreib/Lese-Rechte fehlen, lassen diese sich mit „chmod“ einstellen. Das „+w“ steht für Schreibzugriff ermöglichen:

```
sudo chmod u+w /dev/bus/usb/002/004
```

Überprüfen der Änderung: ls -l /dev/bus/usb/002

6.2.2 Zugriff dauerhaft einrichten

Um den Zugriff dauerhaft, bei jedem Systemstart freizugeben ist anders vorzugehen als zuvor beschrieben. In einem zugänglichen Verzeichnis (z.B: der Projektordner) eine Textdatei mit dem Extend „.rules“ erstellen, in der die gewünschten Geräte mit ihrer VID, PID, Zugriffsmodus und Gruppe aufgelistet sind:

Im Beispiel hier werden vier Geräte eingerichtet. Ein selbstgebautes Programmiergerät, zwei USB-µC im Bootloadermodus und ein AVR ISP MKII-Programmiergerät:

Datei: usbproger.rules

```
# IS51 libusb-device
ATTRS{idVendor}=="16d0", ATTRS{idProduct}=="0418", MODE="0660", GROUP="dialout"

# AT89C5131 - Bootloader
ATTRS{idVendor}=="03eb", ATTR{idProduct}=="2ffd", MODE="0664", GROUP="plugdev"
```

```
# AT89C5122 - Bootloader
ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="2ffe", MODE="0664", GROUP="plugdev"

# AVR ISP MKII
ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="2104", MODE="0664", GROUP="plugdev"
```

Hinweis! -> Hexzahlen in Kleinschreibung, nicht 0x03EB sondern 0x03eb.

idVendor = Vendor-ID (hier 0x16d0, 0x03eb)

idProduct = Produkt-ID (hier 0x0418, 0x2ffd, 0x2ffe, 0x2104)

MODE = Zugriffsmodus (hier 0664: x6xx=owner-rw, xx6x=group-rw, xxx4=other-r)

GROUP = Gruppe für rw(read-write) Zugriff (Der Anwender muß sich darin befinden)

Abschließend ist die Datei, mit Admin-Rechten, in den Ordner "/etc/udev/rules.d" zu kopieren:

```
sudo cp usbproger.rules /etc/udev/rules.d
```

Die Änderung wird dann beim nächsten Neustart von Linux wirksam, oder sofort durch:

```
sudo /etc/init.d/udev stop
sudo udevadm control --reload-rules
sudo /etc/init.d/udev start
```

X Bekannte Fehler

Hier sind erkannte Fehler gelistet, die im derzeit oder dauerhaft nicht beseitigt werden.

X.1 Lesezeichen

Da beim setzen eines Lesezeichens die Hintergrundfarbe des Zeichens gespeichert wird, wird nach wechsel von Syntaxhighlight und entfernen der Markierung diese Farbe als Hintergrund für das Zeichen weiter verwendet - Sytaxhighlight zweimal umschalten behebt das Problem.

Y Infos

Zusätzliche Informationen zum Programm und dessen Umfeld.

y.1 Versionsnummer

Die Versionsnummer besteht aus drei, durch Punkt getrennte, Nummern. Darin bedeutet die erste Nummer die Version. Die zweite Nummer die Revision und die dritte Nummer den Fehlerstatus.

Version: 0.5.3 – 3. Fehlerbereinigung

| |------ 5. Änderung/Erweiterung

|--------- 0=Alphastatus (1=Beta, >1=Endstatus)

Der Fehlerstatus zählt die bereinigten Fehler der Version. Der Änderungsstatus die Änderungen oder Erweiterungen der Version. Und der Programmstatus den Zustand, wobei Nummern größer 1 auf entscheidende Änderungen/Erweiterungen hinweisen.

y.2 A

D