Ein hoffentlich einfaches ToDo

Wickenhäuser µC51-Compiler zusammen mit dem Leistungsstarken Texteditor UltraEdit/UEStudio

Diese kleine Anleitung soll den Benutzern vom µC51 C-Compiler von der Firma Wickenhäuser Elektrotechnik (http://www.wickenhaeuser.de) ermöglichen den Universellen-Editor (UltraEdit bzw. UEStudio) (http://www.ultraedit.com) zu verwenden.

Da ich leider keine Hilfe oder fertige Anleitung im Bezug auf Einbindung des µC51 zusammen mit dem UEStudio gefunden habe, musste ich mich selbst daran setzten.

Damit andere Benutzer des µC51 in Zukunft das ganze etwas leichter haben, tippe ich nun ein kleines ToDo in der Hoffnung das es euch hilft.

Ich selbst habe den µC51 nur für 3 Lehrgänge der Firma Elektor benutzt, da ich normalerweise nur den ATMega benutze hatte ich bis dahin noch keine Erfahrungen mit dem 8051.

Aber wenn man eine IDE wie die vom „Embarcadero RAD-Studio 2010“ oder die Funktionen des „UltraEdit's/UEStudio“ gewohnt ist, ist der JFE doch sehr knapp an Zusatzfunktionen ausgestattet. Und man sehnt sich nach mehr Komfort ☺.

Ich selbst benutze das UEStudio wegen der Projektverwaltung, das macht das ganze doch sehr angenehm. Durch die CompilerConfig (Projekt-Vorlage) kann man dann ganz schnell seine Standard-Projekte auf Knopfdruck anlegen. Auch die optische Klammerführung, Code-Vervollständigungen und -Vorlagen sind doch sehr angenehm ☺.

Leider ist es mir nicht gelungen innerhalb der investierten Zeit (ca. 3-4h) das interne Make-Tool und Compiler-Config des UEStudio an den Wickenhäuser µC51 anzupassen. Ich habe wegen der Projektvorlage einfach mal die SDCC-Compiler-Vorlage als Grundlage genutzt, damit ich den Komfort der Projekt-Vorlage nutzen kann.

Vielleicht kann ja Jemand von euch das ganze entsprechend noch anpassen, gerne übersende ich ihm dann die Word-Datei zur weiteren Bearbeitung.

Eine Anpassung der beiden doch sehr guten Tools lohnt sich meines Erachtens für die Benutzer des µC51.

Aber nun zu der Praxis ☺

Einfaches ToDo vom Wickenhäuser µC51-Compiler zusammen mit dem Leistungsstarken Texteditor UltraEdit/UEStudio

von Thorsten Deck

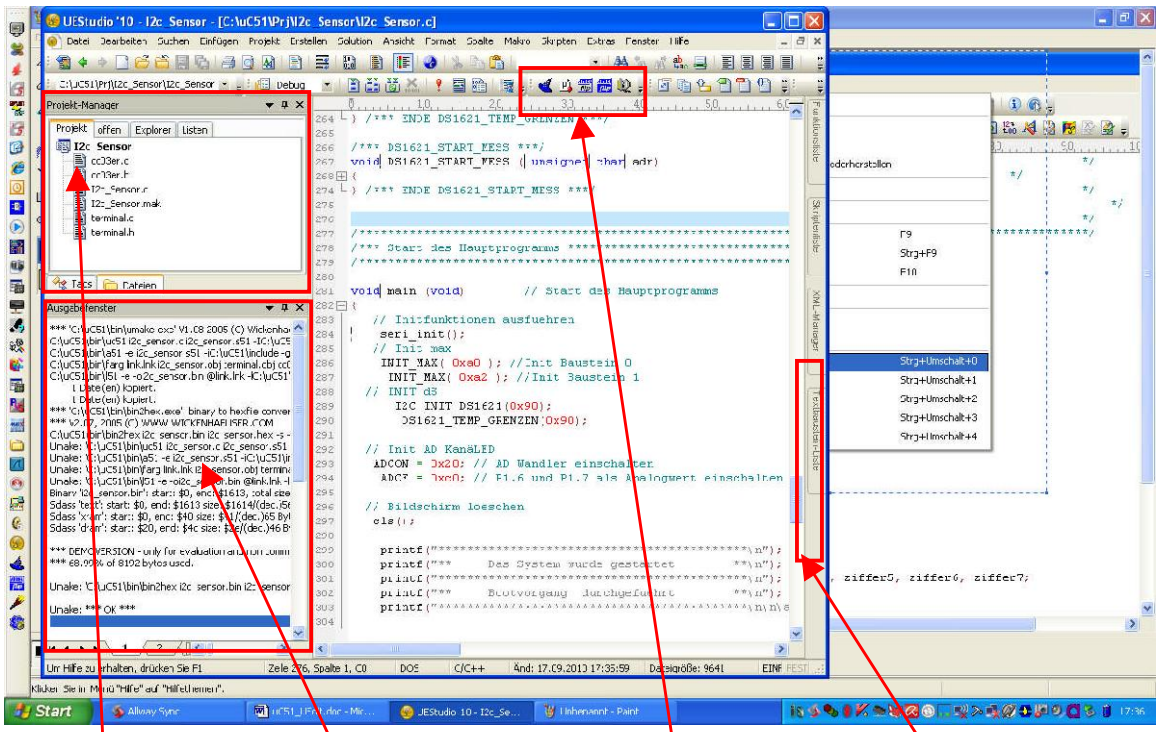
Die folgenden Programme sollten installiert sein:

UEStudio / UltraEdit:
UEStudio (mit der Projektverwaltung) oder der UltraEdit bei mir befindet sich das UEStudio im Pfad: „C:\Programme\IDM Computer Solutions\UEStudio“

µC51-Compiler:
Der Wickenhäuser µC51 sollte sich in einem Pfad OHNE Leerzeichen befinden, bei meiner Installation ist es „C:\uC51“

Atmel-Flip (wir gehen hier von RS232 aus):
Wenn man den AT89C51CC03 oder einen anderen Atmel 8051 als Zielsystem hat sollte sich das Upload-Tool von Atmel in dem Ordner „C:\Programme\Atmel\FLIP 2.4.6.\bin\flip.exe“ befinden. Von diesem Pfad und von dem Zielsystem gehe ich in diesem ToDo aus.

So soll es später aussehen (oder besser ☺)

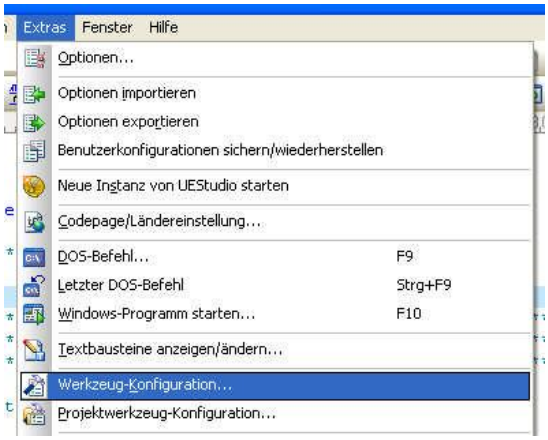


Projektverwaltung beim UEStudio

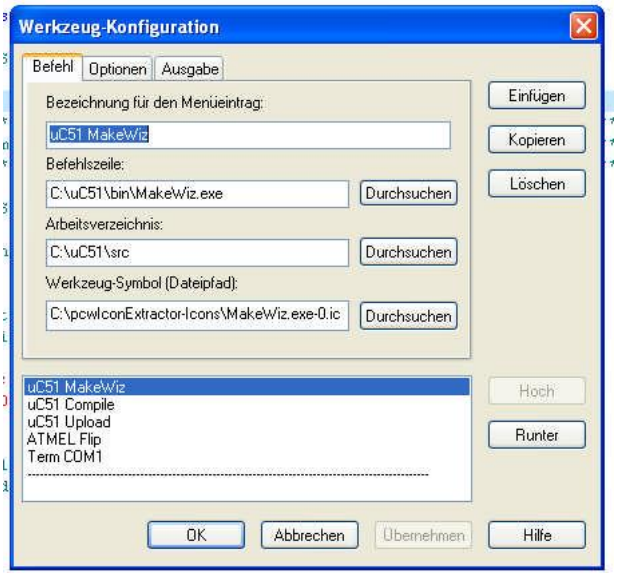
Text und Fehlerausgabe vom µC51-Umake und dem Flip-Upload über die DL.bat.
Bei Fehler einfach anklicken Und UEStudio/UltraEdit springt in die Zeile ☺

µC51-Funktionen
- µC51-Wizard
- uMake
- DL.bat usw.

Textbausteine
Code-Templates
usw.



Als erstes solltet ihr im UEStudio/UltraEdit unter Extras -> Werkzeug-Konfiguration reingehen und die neuen Funktionen gem. Den folgenden Bildern erstellen.



Die Icons könnt ihr euch aus den Dateien ziehen, dafür gibt es im Internet massenweise kleine Tools. (siehe pcwIconExtractor)

Wichtig ist die Befehlszeile, am besten über durchsuchen die MakeWiz.exe suchen.

Bild 1 von 2 MakeWiz

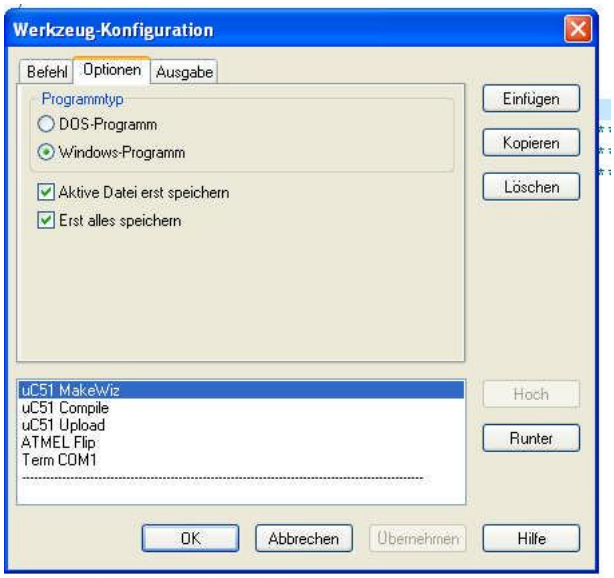
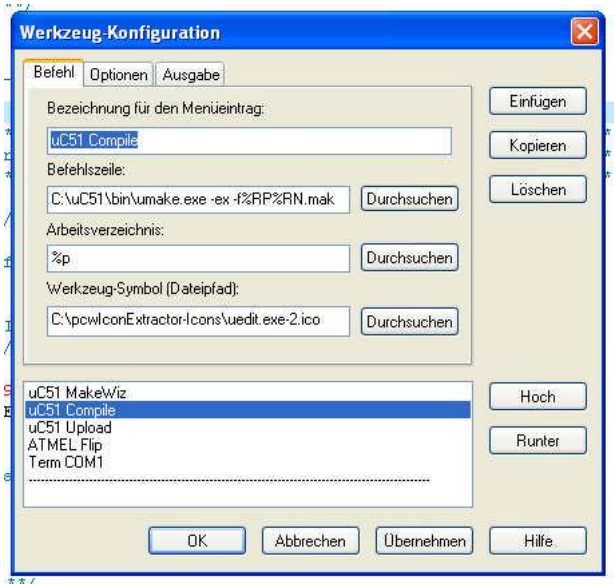


Bild 2 von 2 MakeWiz



Nun die Compiler-Anweisungen über das Make vom µC51.

Wichtig: Die Pfadangaben %RP%RN.mak (beim UEStudio) Hier heisst die Zieldatei (<Projektname>.HEX) genauso wie: <Projektname> bzw. Die Haupt-C-Datei eben auch <Projektname>.c
Damit ist ein automatisches anpassen der µC51-MakeDatei für die Projektvorlage gegeben. Auch der Upload über die Batch-Datei DL.bat (kommt später) ist dann auch gegeben.

Bild 1 von 3 uMake.exe

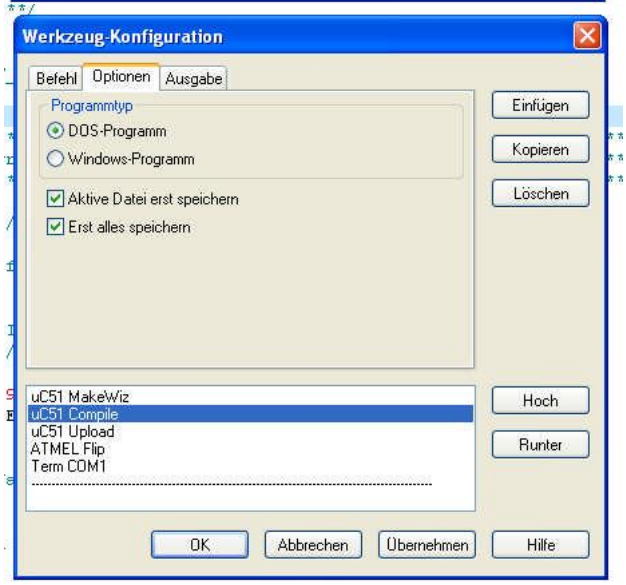


Bild 2 von 3 uMake.exe

Bild 3 von 3 uMake.exe

Einfaches ToDo vom Wickenhäuser µC51-Compiler zusammen mit dem Leistungsstarken Texteditor UltraEdit/UEStudio

von Thorsten Deck

Für die Upload-Batch (oder Übertragung zum Zielsystem) müsst ihr eine kleine Batch-Datei anlegen. Diese basiert auf dem Beispiel im Wickenhäuser „µC51 Handbuch“. Ich habe das ganze nur entsprechend den Vorgaben angepasst (siehe Hilfe vom Atmel-FLIP.

Ich habe diese Parameter für einen AT89C51CC03 den ich über die RS232 COM1 mit 115200 laden möchte angepasst.

Auch möchte ich das die folgenden Funktionen automatisch duchgeführt werden:

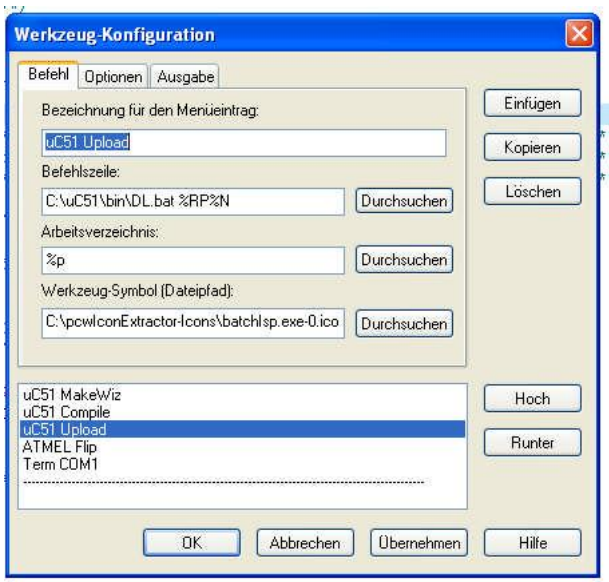
1. onfail abort = Bei einem Fehler bricht der Upload ab
2. erase f = Lösche den Speicher
3. blankcheck = Überprüfe ob der Speicher erfolgreich gelöscht wurde.
4. loadbuffer „%1.hex“= Lade die HEX-Datei die als Parameter ohne Endung übergeben wurde.
5. verify = überprüfe ob es korrekt geladen wurde
6. start reset 0 = Starte das Programm mit einem Reset

Ihr müsst diese Parameter entsprechend eurer Umgebung und Zielsystem anpassen. Speichert den folgenden Text (blau/rot) am besten in die Text Datei „DL.bat“ im Verzeichnis (c:\uC51\bin\) ab.
Wichtig! Die (rot) markierte Zeile MUSS eine einzige Zeile sein!

```
@ECHO OFF
ECHO Angepasst von Thorsten Deck für den Elektor Lehrgang
ECHO mit dem 8051 Atmel cc03er
ECHO Downloadroutine für den AT89C51CC03
ECHO ----------------------------------------------------
batchIsp.exe -device AT89C51CC03 -hardware RS232 -port COM1 -baudrate
115200  -operation onfail abort erase f blankcheck loadbuffer "%1.hex" program
verify start reset 0


@if ERRORLEVEL 1 goto :error
REM *** Flashom used as Terminal, COM and Baudrate unchanged
REM flashmon

@goto :exit
:error
ECHO *** ERROR! ***
ECHO *** ERROR! ***
ECHO *** ERROR! ***
:exit
exit
```



Nun die Upload-Funktion, damit dies nicht immer über die Windows-Oberfläche des FLIP gemacht werden muss.

Dadurch könnt ihr mit einfachen drücken der Tasten STRG+SHIFT+<Und Werkzeugnummer> direkt die Übertragung zum Zielsystem machen. Der Name der HEX-Datei wird hier automatisch übergeben. Natürlich muss der CC03 sich vorher im Bootloader-Mode befinden ☺

Bild 1 von 3 Übertragungsfunktion

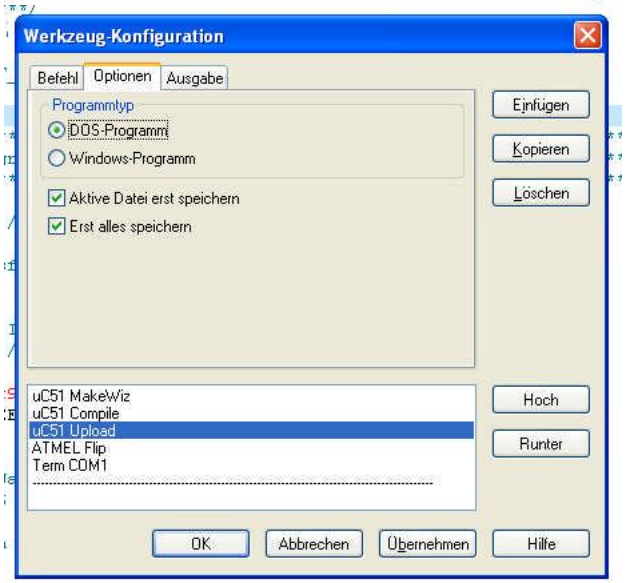
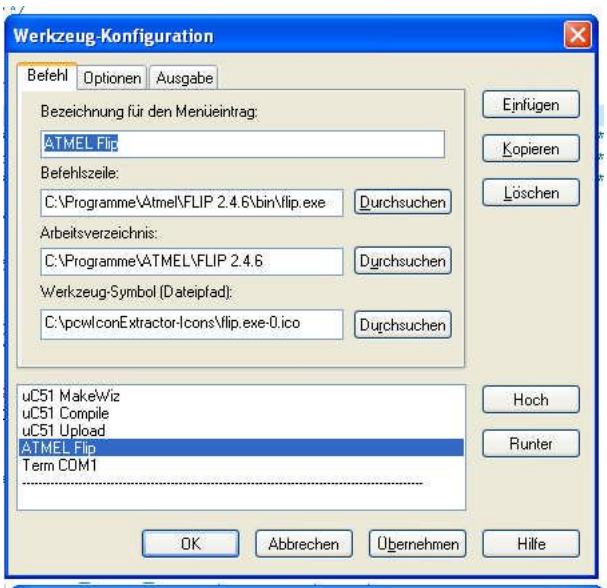


Bild 2 von 3 Übertragungsfunktion

Bild 3 von 3 Übertragungsfunktion



Falls ihr doch mal die Windowsoberfläche von FLIP benutzen möchtet, solltet ihr auch diese hier einbinden ☺

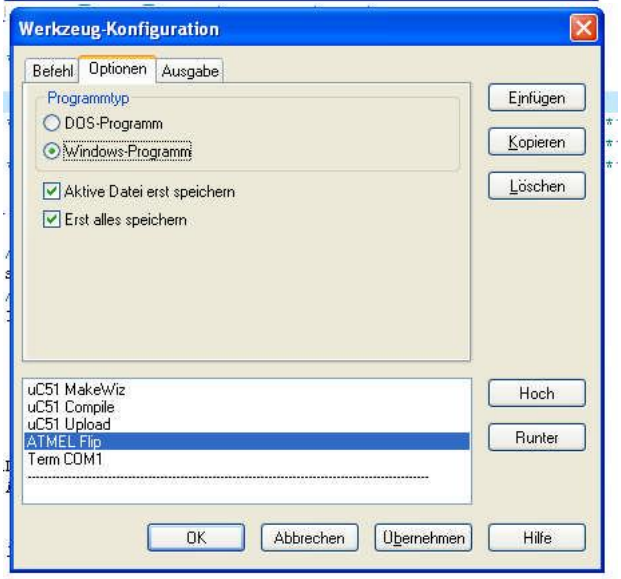


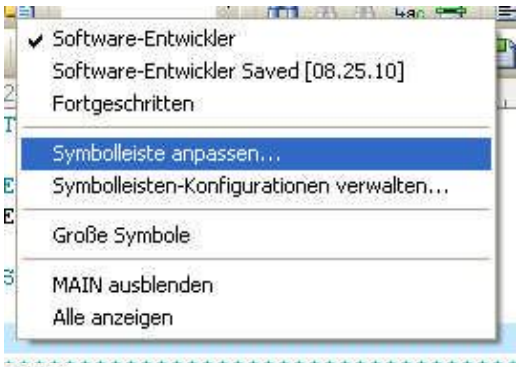
Bild 1 von 3 Windows-Flip

Bild 2 von 3 Windows-Flip

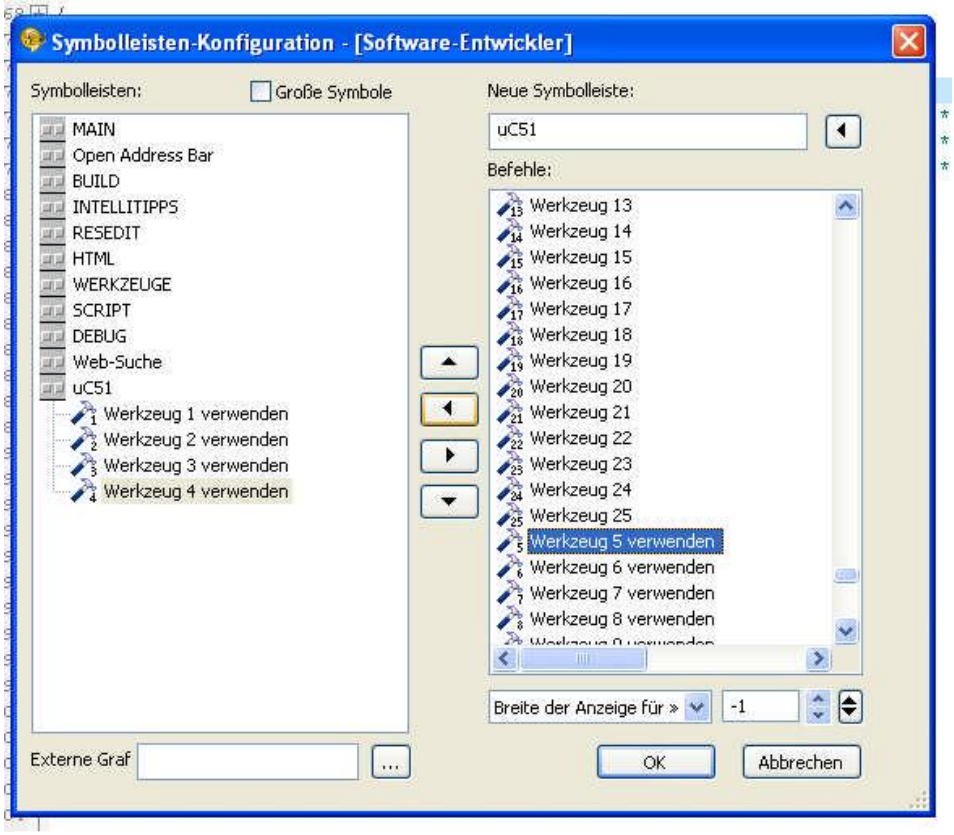
Bild 3 von 3 Windows-Flip

Natürlich wollen wir schöne Symbole in der Leiste haben ☺

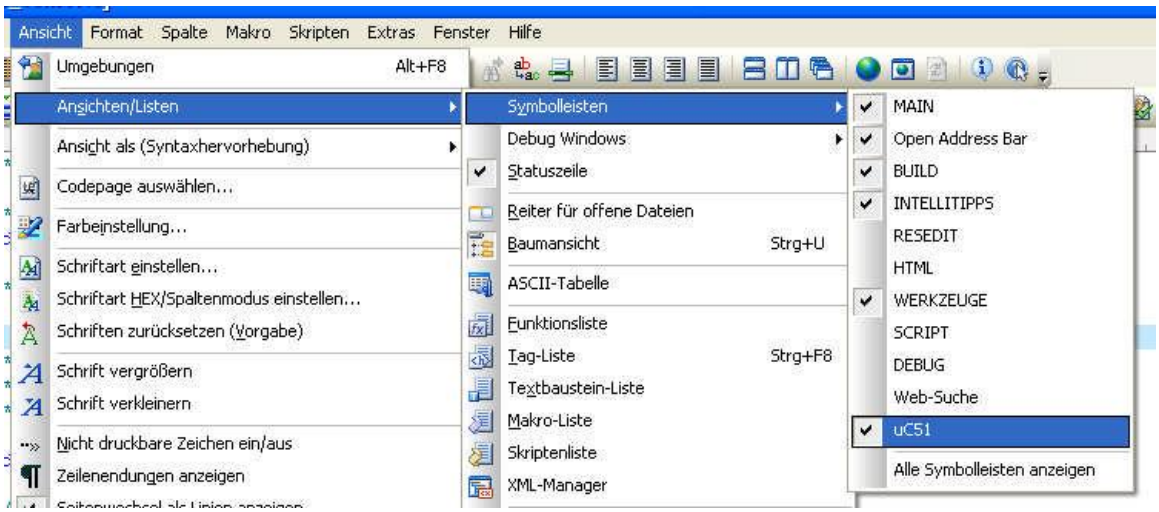
Im Internet nach der Datei „pcwIconExtractor“ suchen, ein einfaches Tool, was die ICON’s aus Dateien und DLL’s herausfiltert und in einem Verzeichnis ablegt. Diese könnt dann im Werkzeug-Editor (unter UEStudio/UltraEdit >> Extras) einstellen und es sieht dann hübsch aus.



Um eine neue Symbolleiste der 4 neuen Werkzeuge zu erstellen einfach im UEStudio / UltraEdit im Menü Ansicht -> Symbolleiste anpassen auswählen.

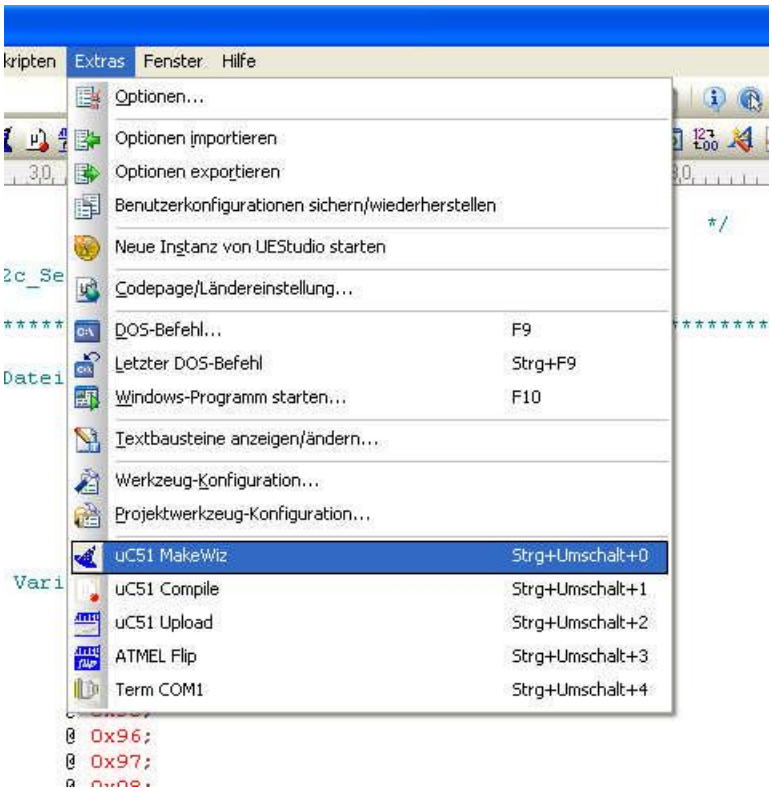


Dann eine neue Symbolleiste erstellen z.B. „uC51“ und die neuen Werkzeuge dort einfügen und mit „OK“ bestätigen.



Dann unter Ansicht >> Ansichten/Listen >> Symbolleisten >> uC51 aktivieren.

Dann könnte die Symbolleiste in etwa so aussehen, wenn die richtigen Symbole hinterlegt wurden.



Natürlich könnt ihr auch diese Befehle mit den tasten „STRG+UMSCHALT+<Werkzeugnummer>“ oder über das Menü Extras auswählen.

Das bleibt euch überlassen.

☺

Einfaches ToDo vom Wickenhäuser µC51-Compiler zusammen mit dem Leistungsstarken Texteditor UltraEdit/UEStudio

von Thorsten Deck

Für die Möglichkeit sein „Standard-Leer-Projekt“ in der Projektverwaltung auf Knopfdruck zu erstellen, wird das UEStudio benötigt.

Das UEStudio ist etwas teurer als der UltraEdit, aber alleine durch die Projektverwaltung lohnt sich die paar Euro unterschied.

Den folgenden Text (blau) komplett in eine neue Datei schreiben, in dem UEStudio-Pfad:
„C:\Programme\IDM Computer Solutions\UEStudio\projects\uC51_8051“

Die Textdatei sollte einen sinnvollen Namen haben, z.B.: Standard_8051_leer.wiz

Diese Dateien könnt ihr nach Lust und Laune entsprechend mit den liebsten Grund-Funktionen nach euern Geschmack bestücken.

Denkt aber an die Anpassung der MakeDatei, entweder gleich richtig in der Vorlage oder dann doch über den MakeWiz von Wickenhäuser.

Die MakeDatei (File1) ist hier an meine Lehrgangsumgebung und Zielsystem angepasst. Natürlich könnt ihr noch weitere C-Dateien hierüber erzeugen lassen, am besten mal den MakeWiz von Wickenhäuser benutzen und sich die Datei im Texteditor anschauen (hat bei mir auch geholfen).

```
################################################################################
####
#                                                                              #
#           UEStudio's User Defined Project Template                           #
#                                                                              #
# The following template is to be modified by the user.  It describes          #
# individual preferences for new project creation.                             #
#                                                                              #
# The first and most important section is [UDT] which contains a "Compiler"    #
# variable and a list of seed files to be inserted in the newly created       #
# project.  The Compiler variable specifies the name of the compiler's         #
# configuration directory, and the compiler configuration file.                #
# For example:                                                                 #
# Compiler = Microsoft Visual C++ Compiler\Win32 Console Application           #
#                                                                              #
# The list of project files may be any combination of source, header,          #
# resource, and data files.  An example follows:                               #
#  File0=MyApp.h                                                               #
#  File1=MyApp.c, Source Files                                                 #
#  File2=MyApp.ico                                                             #
# and so forth....                                                             #
#                                                                              #
# All subsequent section(s) of this template describe the contents of one      #
# or more of the project files defined above.  An optional group name may      #
# be specified after the file name.  In the above example, the group           #
# "Source Files" will automatically be created and MyApp.c will be included    #
# in the group folder.                                                         #
```

```
#                                                                     #
# For binary file contents such as Resource, Icon, or any other data file,   #
# where it is inconvenient to describe the contents in a script, the        #
# following format may be used:                                       #
# [File2]                                                           #
# INCLUDEBINARY=c:\my_data_files\ico_data.raw                               #
# This will create [File2], in this case "MyApp.ico", with the content       #
# ico_data.raw                                                        #
#                                                                     #
# Alternately, binary file contents may be defined in this template using    #
# the ":HEX" directive.  This specifies that all data is represented as      #
# hexadecimal values.  For example:                                         #
# [File3:HEX]                                                       #
# 0065 66 F0, 0FF0, 0F0F0F0F                                                #
# 12, FF, 20; 0F 40                                                   #
#                                                                     #
# where all numbers above are interpreted based on data length, as follows:  #
#  1-2 = bytes, 3-4 = words, 5-8 = dwords, >8 = an array of bytes.          #
#                                                                     #
# Predefined variables may be used to specify the names of files as well as  #
# the content of the files, e.g.:                                           #
# USER (user name), COMPUTER (computer name), TIME, DATE, PRJPATH (project   #
# path), PRJNAME (project name), PRJDIR (project directory)                 #
# Example: File0=$(PRJNAME).cpp                                             #
#                                                                     #
#######################################################################################
####

[UDT]

Compiler = uC51\Application

File0 = $(PRJNAME).c
File1 = $(PRJNAME).mak


[File0]
/*********************************************************************************************/
/*                                                                   */
/*    Programm:                    $(PRJNAME)                                    */
/*                                                                   */
/*                                                                   */
/*    Autoren:          $(USER).                                    */
/*                                                                   */
/*    Zielsystem:       AT89C51CC03 (Atmel)                         */
/*    IDE/C-Compiler:   uC/51 (Wickenhäuser)                                */
/*                                                                   */
/*    Letzte Änderung: $(DATE).                                            */
/*                                                                   */
/*    Datei:            $(PRJNAME).c                                        */
/*                                                                   */
/*********************************************************************************************/

/*** Einbinden von Include-Dateien ***/
```

```
#include <stdio.h>
#include <at89c51cc03.h>


// Defines der Adressen und Variablen
unsigned char bit BNC          @ 0x91;
unsigned char bit LED_TAST1    @ 0x92;
unsigned char bit LED_TAST2    @ 0x93;
unsigned char bit LED_TAST3    @ 0x94;
unsigned char bit LED_TAST4    @ 0x95;
unsigned char bit SUMMER       @ 0x96;
unsigned char bit POTI1        @ 0x97;
unsigned char bit POTI2        @ 0x98;

int    x;
float  Wert;


void cls (void)
{
  // Bildschirm loeschen
   printf("\x1b\x48\x1b\x4a");

}



/*****************************************************************************************/
/*** Start des Hauptprogramms ************************************************************/
/*****************************************************************************************/

void main (void)                        // Start des Hauptprogramms
{
   // Initfunktionen ausfuehren
    seri_init();

  // Bildschirm loeschen
   cls();

    printf("*******************************************\n");
    printf("**     Das System wurde gestartet        **\n");
    printf("*******************************************\n");
    printf("**     Bootvorgang  durchgefuehrt        **\n");
    printf("*******************************************\n\n\a");

    while(1)   // Beginn Hauptschleife
       {
                                        // Ab hier der LOOP



       } // Ende Hauptschleife

}  // Ende VOID MAIN
```

```
/*****************************************************************************************/
/*** Ende des Hauptprogramms, d.h. Ende des gesamten Programms!
*****************************/
/*****************************************************************************************/

[File1]
##################################################
# Project: $(PRJPATH)$(PRJNAME).mak
# (generated with WIZ for UESTUDIO)
# uC/51 ANSI C Compiler - www.Wickenhaeuser.de
##################################################

L51FLAGS = -r$0,$0
SIOTYPE = K
MODEL = small
B2HFLAGS = -s
A51FLAGS = -g
C51FLAGS = -dCPU_NSEC=1085

$(PRJNAME).obj: $(PRJNAME).c
$(PRJNAME).bin: $(PRJNAME).obj
$(PRJNAME).hex: $(PRJNAME).bin
```

Hier könnt ihr so viele Vorlagen erstellen wie ihr wollt und auf Knopfdruck habt ihr dieses Projekt erstellt.

Die Compiler-Vorlage ist natürlich eine RIESEN-Baustelle (ich bin mir aber sicher das geht irgendwie über das interne MAKE vom UEStudio. Ich habe es aber nicht auf die schnelle geschafft (vielleicht probiert es noch Jemand anderes ☺)

Für die Compiler-Vorlage habe ich mir einfach die Config des SDCC kopiert.
Leider habe ich den uC51 nicht dazu bekommen, das ich ihn komplett über diese Config steuere, deswegen der Weg über die Werkzeuge.

Die Compiler-Config benutze ich nur um schnell ein Projekt gem. meiner Vorlage zu erstellen. Man kann sich natürlich so viele eigene Vorlagen basteln wie man möchte ☺.

Den folgenden Text (blau) bitte in dem Pfad „C:\Programme\IDM Computer Solutions\UEStudio\configs\uC51“ in den Dateinamen „Application“ ohne Dateiendung als Textdatei speichern. (wenn auch dort das UEStudio installiert ist ☺)

```
# ----------- uC51 configuration ------------
# --- general -------------------------------------
# $P  - project name
# $Pp - path to project directory
# $Pn - project name
# --- compile -------------------------------------
```

```
# $I  - input full name
# $Ip - input path
# $In - input name
# $Ie - input extension
# $O  - output file
# $Op - path to output file
# $On - output filename (without path)
# $Oe - output extension
# $R  - release/debug setting for compiler
# --- build ----------------------------------------
# $T  - target full name
# $Tp - target path
# $Tn - target name
# $O  - output file
# $Op - path to output file
# $On - output filename (without path)
# $Oe - output extension
# $R  - release/debug setting for linker

[Settings]

Target =
Category&01 = DEFAULT RUN CONFIGURATION
Working Directory =.
Command Line Arguments =
Category&02 = COMPILER OPTIONS
Processor = Intel MCS51 (8051-family)|Dallas DS80C390|Dallas
DS80C400|Freescale/Motorola HC08|Zilog Z80|GameBoy Z80|Atmel AVR|Microchip PIC 14-
bit (p16f84-family)|Microchip PIC 16-bit (p18f452-family)|Toshiba TLCS-900H|Phillips XA51
Compiler Options =
Category&02 = LINKER OPTIONS
Output Format = Intel HEX|Motorola S19
Linker Options =

[SettingsInfo]
Target = Provides a space for you to specify an output file and location of the program that
the linker creates.
Working Directory = Provides a space for you to specify the directory in which executing
occurs. If you do not specify a directory, executing occurs in the directory where the
executable is located.
Command Line Arguments = Provides a space for you to specify command-line arguments
you want to pass to the program at startup.
Processor = This option defines the processor which is being used.
Output Format = Use this option to specify the output format.
Compiler Options = Provides a space for you to specify additional compiler options.
Linker Options = Provides a space for you to specify additional linker options.

[SettingsReps]

Processor = Intel MCS51 (8051-family)=mcs51|Dallas DS80C390=ds390|Dallas
DS80C400=-ds400|Freescale/Motorola HC08=hc80|Zilog Z80=z80|GameBoy
Z80=gbz80|Atmel AVR=avr|Microchip PIC 14-bit (p16f84-family)=pic14|Microchip PIC 16-bit
(p18f452-family)=pic16|Toshiba TLCS-900H=tlcs900h|Phillips XA51=xa51
Output Format = Intel HEX=ihx|Motorola S1/S9 HEX=s19
```

```
[Environment]
INCLUDE=C:\uC51\include
LIB=C:\uC51\lib

[Variables]

COPT = $(Compiler Options)
LOPT = $(Linker Options) -m$(Processor) --out-fmt-$(Output Format)

[General]

TargetExt = .hex
ReleaseOut = Release
DebugOut = Debug
UseFullPaths = 0
UseDosNames = 0
Excludes = $(Excluded Files)
RemoveDot = 0
ConvertBS = 0
ShowCmdLine = 1
GrabOut = 1
GrabErr = 1
MakeTool=C:\uC51\bin\MakeWiz.exe
.C51 = .C

[MakeCommands]

makef=Show Makefile

[InsertFiles]

[FileGroups]

FGR = .rel;
FGL = .lib;

[Build]

Out = $T
Depends = $FGR
DebugFlag = --debug
#Cmd0 = SDCC --use-stdout $R $(LOPT) $FGR
#Cmd1 = REN *.$(Output Format) $T
Cmd0 = C:\uC51\bin\umake.exe -ex -f$Pp\$Pn.mak
[.C]
Out = $In.rel
Depends = $I
Cmd0 = C:\uC51\bin\umake.exe -ex -f$Pp\$Pn.mak
#Cmd0 = SDCC --use-stdout $R $(COPT) -c -o $O $I
IncFiles = 1
CaseSensitive = 1
IncDirs = .;$(INCLUDE);
IncKeyWords = #include;
```

```
Comments = /*.*/.//.eol.

[Show Makefile]
Title=Show makefile
Cmd0=uestudio $(UESMAKEFILE)
ShowWindow=1
DisplayConsole=0
```

Tipp:
Schaut euch auf noch die Textblock-Funktionen an, hier kann man sich jegliche Codevorlage direkt zur Verfügung stellen, evtl. markierter Text wird dabei mit in den Code-Block eingebunden. Sehr hilfreich!

Noch ein paar Worte / Hinweise zum Schluss:

Da ich normalerweise nicht mit diesem Compiler Arbeite erschlagt mich nicht, wenn ich hier irgendwas falsch gemacht habe. Dieses ganze ToDo soll den Benutzern vom Wickenhäuser-µC51-Compiler einfach nur etwas bei der Einbindung helfen.

Natürlich übernehme ich auch keine Haftung für evtl. Schäden die durch die Benutzung des ToDo's entstehen. Es gibt keine Verpflichtung auf die Richtigkeit der Links und/oder Inhalte.

Den Wickenhäuser-µC51-Compiler gibt es als Demo-Version (8kb-Codegröße begrenzt) und das UEStudio gibt als 30-Tage-Testversion. Probiert es erstmal aus ☺

Wenn ihr noch Tipps für die Compiler-Vorlage habt und/oder Fehler feststellt einfach eine eMail an mich: tdeck@pvs-deck.de

Ansonsten wünsche ich euch viel Spass mit dem Arbeiten am µC51 in der UltraEdit/UEStudio-Umgebung.

Perfektes Wissen im Bereich der Mikrocontrollern:
Noch ein großes „Lob" an Elektor und die beiden Referenten Prof. Dr.-Ing. Bernd vom Berg und Dipl.-Ing. Peter Groppe von der Technischen Fachhochschule (TFH) Georg Agricola zu Bochum.

Die Lehrgänge „3-tägiges Seminar C-Programmierung für Mikrocontroller" und „3-tägiger Workshop Serielle (Geräte-)Bussysteme" am Atmel AT89C51CC03 durchgeführt haben.

Beide Lehrgänge sind wirklich sehr hilfreich bei der Arbeit mit Mikrocontrollern.

Viel Spass und Erfolg wünscht euch

Thorsten Deck