Virtuelle Machine WIN XP mit KEIL und FLIP

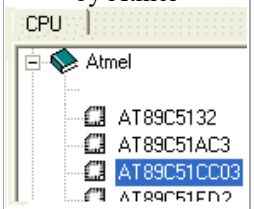
Ausstattung der Virtuellen Maschine:

Platte C:(IDE)	Betriebssystem
CDROM D:	Addons (readonly)
Platte E:(IDE)	Anwendungen und Daten
RAM 1GByte	Ist vollkommen ausreichend!!
Seriell COMx	Serielle Schnittstelle als Verbindung zum Controller *Benutzt einen vorhandenen USB-Serial-Adapter*
Netzwerkkarte	Verbindung zum Internet(default: AUS) *Benutzt einen WiFi-Adapter des Hosts*

Falls die Hardware-Erkennung anspringt! → 2 x auf ABBRECHEN klicken

---

Der "8051" von Intel ist eine ab 1980 erschienene 8-Bit-Microcontroller-Familie die weltweit sehr stark verbreitet ist.
Auch wenn heute die 32 bzw 64 Bit Prozessoren den Markt scheinbar beherschen, so werden auch heute noch sehr viele 8-Bit Controller verbaut.
Fast jedes elektronische Haushaltsgerät enthält einen Chip aus der 8051-Familie!

"Unser" Chip: **AT89C51CC03** by Atmel 	8051 based CMOS controller with • PCA, Dual DPTR, WDT, • 10 Bit ADC, • Full CAN, • 40 MHz High-Speed Architecture, • X2 function, 32+2 I/O lines, • 3 Timers/Counters, • 14 Interrupts/4 priority levels, • 64K FLASH, 2K EEPROM, 256 Bytes on-chip RAM, additional 1K XRAM

Verdrängt werden die "51"-er erst heutzutage durch WiFi-fähige Controller im IOT-Bereich.

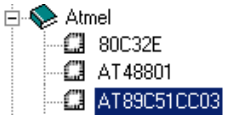
KEIL™	Die Münchner Firma KEIL*(mittlerweile übernommen von ARM)* hat sehr brauchbare Software-Tools zur Programm-Entwicklung und Simulation von für Microcontroller entwickelt!
FLIP	Ist das "Standard-Tool" zum "Brennen" eines Programms in einen Controller. Computer → Download/Flashing → µController
HTERM	Ist das beste Werkzeug zur Kommunikation mit einem µController über die Serielle Schnittstelle HTerm ←→ Serial ←→ µController
PuTTy	Ist das Standard-Werkzeug zur Kommunikation mit einem (entfernten) µController Putty ←→ Serial ←→ µController Putty "versteht" auch farbige Ausgaben via ANSI-Codes durch den Microcontroller

Programmiert werden die Controller in

Maschinensprache	Hochsprache
Assembler *.a51 → hex	C-Compiler *.c → Binary
hardwarenah	hardwareunabhängig
unschlagbar schnell	Unschlagbar bei Übertragung auf andere Hardware
Blink-Prog = 20 Byte	Blink-Prog = 20 **K** Byte

89c51cc03 Assembler-Programmierung mit KEIL µVision:

- Start µVision: 

- Neuen Ordner anlegen: **myProjekt_ASM**
- Projekt->Neu im Ordner **myProjekt_ASM**
- Projektdatei anlegen: **myProjekt_ASM.uvproj**
- Select CPU: 

- Copy 8051 Start-UP Code: **Ja**
- Target Options 
OUTPUT **[x] Create HexFile**
Target -> X-tal [MHz] **12** *statt 33*
- StartUP.A51 oeffnen und **alles (ca 190 Zeilen!!) löschen!**
- **Neuer** Content für STARTUP.A51: *siehe unten*
- Rebuild all Targets 
- Close uVision ALT-F4
- Im Projekt Ordner ALLE DATEIEN löschen ausser ***.uvopt *.uvproj *.A51 *.HEX**
- Upload HEX File to 89C51CC03 mit **FLIP**

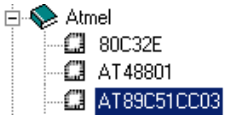
QUELLCODE: **StartUP.A51**

```
01: /* COMMANDOS FUER COMMAND Fenster:
02: MODE COM1 57600, 0, 8, 1   // COM1: 57600 bps, no parity, 8 data , 1 stop bit
03: ASSIGN COM1 SOUT     // SerialINput SerialOUTput erfolgt ueber echte Hardware COM1:
04: // S0TIME = 0              // ignore timing of simulated Serial Interface
05: */
06:
07: $NOMOD51                              ; Die Defaults fuer Standard "51" vergessen und statt dessen die
08: $include(at89c51cc03.inc)              ; speziellen Festlegungen fuer unsere CPU einbinden
09:
10: Init:       CALL    Serial_Begin_57600
11:             CALL    StartMsg
12: HabeFertig: JMP     $
13:
14: HalloWelt:  db 'Hallo Welt, hier ist mein uC Programm',0Dh,0Ah,0
15:
16: StartMsg:   MOV     DPTR,#HalloWelt
17: hwloop:     CLR     A                  ; Akku loeschen
18:             MOVC    A,@A+DPTR          ; Zeichen auf das der DataPointer zeigt laden
19:             JZ      hwdone             ; Wenn Zeichen==ZERO, dann letztes Zeichen erreicht
20:             MOV     SBUF,A             ; Sendebuffer mit Zeichen aus Akku laden
21:             JNB     TI,$               ; Warten bis Zeichen gesendet
22:             CLR     TI                 ; Transmit-Interupt Flag loeschen
23:             INC     DPTR               ; Datapointer erhoehen
24:             SJMP    hwloop             ; naechster Buchstabe
25:
26: hwdone:     RET                        ; Fertig mit dem Unterprogramm!, also Zurueck zur Aufrufstelle
27:
28:
29: Serial_Begin_57600:
30:             MOV     CKCON,#1
31:             MOV     SCON,#50h          ; Serielle Schnittstelle mit Timer2
32:             MOV     RCAP2H,#0FFh       ; High-Wert Timer fuer 57600 ;#0FFh fuer 9600
33:             MOV     RCAP2L,#0F3h       ; Low -Wert Timer fuer 57600 ;#0B2h fuer 9600
34:             SETB    RCLK               ; aktiviere Receive
35:             SETB    TCLK               ; aktiviere Transmit
36:             SETB    TR2                ; Timer 2 Aktivieren
37:             SETB    REN                ; Receiver ENable
38:             CLR     RI                 ; Receive Interupt Flag loeschen
39:             SETB    EA                 ; Erlaube External Access
40:             CLR     ES                 ; Dis-able Serial Interrupts !! -> use Polling!!
41:             RET
42: end
```

C-Programmierung mit Keil µVision

- Start µVision

- Project-> New Project
- Lege Projektordner an: **myProjekt_Cv01**
- Wechsle in diesen Ordner
- Projectname: **HalloWelt_Cv01**.uvproj
- uC-Controller Auswahl:



- Copy Standard-Startup-File to Projektspace: **NO**
- File New
- File Save as: **main.c**
- Select **Source Group 1** im Projektspace unter Target 1
- Rechts-Klick: **Add Files to Source Group 1**
- Project -> Options for Target1 -> Output -> **[x] Create HEX-File**
- Project -> Options for Target1 -> Target -> X-tal [MHz] **12** *statt 33*
- Edit main.c z.B.: *siehe Beispiele unten*

- Save all
- Rebuild all 
- Close uVision
- Im Projekt Ordner alles löschen ausser: ***.uvopt *.uvproj *.h *.c *.HEX**
- Download *.HEX zum 89C51CC03 mit: **FLIP**
- Kommunikate via Serial-Terminal-Client: **Putty**

Will man die lästigen L16-Warnungen bezüglich nicht eingebundener Module ausschalten,

```
*** WARNING L16: UNCALLED SEGMENT, IGNORED FOR OVERLAY PROCESS
```

so kann man dies einstellen über:

```
Project -> Options for Target1 -> BL51 Misc -> Disable Linker Warnings: 16
```

QUELLCODE: **Main.c**

```
01: /*
02: ===================================================================
03: Projekt:    uV_HalloWelt_01
04: Programm:   Main.c
05: Aufgabe:    Zeigt die CSharp-nahe Programmierung von AT89C51CC03
06: Sprache:    Keil-C fuer Mikrocontroller
07: Hardware:   uC Board der RBS ULM mit AT89C51CC03
08: Version:    2021-03-13
09: Autor:      R. Hoermann
10: ===================================================================
11: */
12:
13: #include <at89c51cc03.h>           // Spezialitaeten der CPU
14:
15: void _putchar(char c)                   // Redeclaration von putchar um ein Zeichen auszugeben!!
16: {
17:   SBUF = c;
18:   while(!TI);
19:   TI   = 0;
20: }
21:
22: void uart_putc(char c)                  // Ein Zeichen seriell ausgeben
23: {  _putchar(c); }
24:
25:
26: void Console_Init_57600(void)
27: {
28:   CKCON0 = 0x01;                        // x2 Bit 24 MHz statt 12 MHz
29:   SCON   = 0x50;                        // 8 Bit , No Parity, 1 Stop
30:   RCAP2H=0xFF; RCAP2L=0xF3;             // Reload-Werte fuer 57600 Bit/sec
31:   RCLK=1;  TCLK=1;  TR2=1;  REN=1;  RI=0;  EA=1;
32:   ES=0;                                 // Polling statt Interrupts !!
33: }
34:
35: void Console_WriteCn(char c,int n)      // Ein Zeichen  n mal ausgeben
36: {
37:   int i;
38:   for (i=0;i<n;i++){uart_putc(c);}
39: }
40:
41: void Console_Beep(void)                 //  Piepston ausgeben
42: {
43:   uart_putc(7);
44: }
45:
46: void Console_NewLine(void)              // Wagenruecklauf und Zeilenvorschub ausgeben
47: {
48:   uart_putc(13); uart_putc(10);
49: }
50:
51: void Console_Write(const char *s)       // Einen String seriell ausgeben
52: {
53:   while (*s) uart_putc(*s++);
54: }
```

```
55:
56: void Console_WriteLine(const char *s)   // Einen String seriell ausgeben + New Line
57: {
58:   while (*s) uart_putc(*s++);
59:   Console_NewLine();
60: }
61:
62: void main()
63: {
64:   Console_Init_57600();
65:   Console_WriteCn('=',40);
66:   Console_NewLine();
67:   Console_Write("Hallo ");
68:   Console_WriteLine("Welt");
69:   Console_WriteCn('=',40);
70:   Console_WriteLine("");
71:   Console_Beep();
72:   while(1){}
73: }
```