TinyBasic fuer STM8S103F3P6

Ein ultraminimalistischer Basic-Interpreter

Initialprojekt und Ausgangsdatei von T. Suzuki, STM8S Port und Erweiterungen von R. Seelig

Im Rahmen eines kleinen privaten "Wettbewerbs" ging es darum, wie billig ein billigster BASIC-Computer ausfallen könnte. Vordergründig war dieses durch die damals extrem preiswerten (billigen) Teile aus China initiiert worden.

Da bei solchen Dingen der Spieltrieb und der Ehrgeiz geweckt wird hatte ich mich an das Werk gemacht und das hier war das Resultat.

Den preiswertesten Microcontroller den ich gefunden (und sogar verfügbar) hatte war im Jahr 2016 ein STM8S103F3P6.

Stand 2023 wäre ein STM32F030f4 bei sehr deutlich besseren Leistungsdaten preiswerter. Hier jedoch jetzt die finale Version für einen STM8S103. Da der Speicherplatz des Mikrocontrollers nun fast komplet ausgenutzt ist, wird es (höchstwahrscheinlich) keine weitere Arbeit und Erweiterung an der Version für den STM8S103 geben.

Serielle Schnittstelle / UART

Um ein Basic-Programm eingeben zu können bedarf es einer Eingabemöglichkeit. Dieses erfolgt über ein serielles Terminalprogramm und hierfür bedarf es wiederum einer seriellen Schnittstelle am PC. Da diese mittlerweile "ausgestorben" sind muß ein USB2UART Brücke die serielle Funktion übernehmen, die sich an einem PC dann als virtueller serieller Port anmeldet. Hierfür wird beim Basicinterpreter ein

CH340G

Chip verwendet. Allerdings ist jede andere USB2UART Brücke (bspw. von FTDI) verwendbar. Der CH340G Chip ist sehr preiswert und leistet unter Linux wie Windows super gute Dienste. Das Schnittstellenprotokoll des Interpreters ist:

19200 Baud, 1 Startbit, 8 Datenbits, 1 Stopbit

Basic Firmware

Die Firmware ist in der gepackten ZIP-Datei enthalten. Es benötigt ein Linux um dieses zu übersetzen. Derhierfuer verwendete Compiler fuer die Firmware ist SDCC in der Version 4.1.0 oder neuer. Er kann gedownloaded werden unter:

```
https://sourceforge.net/projects/sdcc/files/sdcc-linux-x86
```

Erstellen Sie ein Verzeichnis Ihrer Wahl, bpsw.:

```
cd ~
mkdir stm8projects
```

und entpacken Sie das Firmwarepackage in dieses Verzeichnis.

Es entsteht folgende Verzeichnissstruktur:

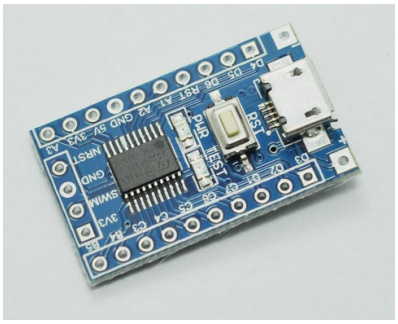
```
basict ---------- Makefile
                  sendbas19200
                  compilesendbas
                  tbasic.c
                  tbasic.ihx
                  adctest.bas
                  array.bas
                  ascii.bas
                  blink.bas
                  fakul.bas
                  lauflicht.bas

include --------- stm8s.h
                  stm8_init.h
                  stm8_gpio.h
                  uart_tbasic.h

src ------------- uart_tbasic.c
                  stm8_init.c
                  clean

tools/bin ------- ch340reset
                  cide
                  picocom
                  st8readihx
                  stm8_bootflash2
                  stm8flash
                  stm8lock
                  stm8lock.bin
                  stm8unlock
                  stm8unlock.bin
                  stm8unlock105

cide
cleandir
makefile.mk
```



Soll für den Interpreter ein preiswertes „Minimum Development Board“ aus China verwendet werden, sind diese bei der Auslieferung leider "gelockt", so dass dieser Lock aufgehoben werden muss. Das geschieht jedoch relativ einfach in Verbindung mit dem ST-Link v2.

Verbinden sie die 4 Leitungen des Boards mit dem ST-Link. Die Leitungen sind bezeichnet mit

SWIM, RST, GND, 3.3V

Wechseln Sie in das Verzeichnis ./stm8projects/tools/bin und rufen dort das Programm stm8unlock auf:

./stm8unlock

Der Lock wird hiermit aufgehoben.

Firmware flashen

Zum Flashen des Controllers benötigt man einen ST-LINK v2. Ein Chinaclone-Stick ist hier von Vorteil, weil dieser außer einem STM32 auch einen STM8 flashen kann.

Hier sind die Anschlüsse SWIM, RST, GND und 5V mit dem Mikroconroller zu verbinden. Im Ordner ./stm8/basict kann die Firmware erzeugt werden.

Ein Aufruf von

make

übersetzt die Firmware (ein installierter SDCC 4.1.0 oder höher wird vorausgesetzt). Wurde der Compiler- und Linkervorgang fehlerfrei durchgeführt, werden im Verzeichnis mehrere Dateien erzeugt, u.a. auch eine < tbasic.ihx >. Dies ist die Datei, die im Intel-Hexformat das Binary des Interpreters beinhaltet. Bevor die Firmware geflasht wird, empfielt es sich, mit einem Texteditor die Datei <Makefile> dahingehend zu kontrollieren, ob der zu benutzende Programmer korrekt eingestellt ist. Ein STLINKv2 Programmer wird hier eingetragen mit

FLASHERPROG = 0

Ist der Programmer korrekt eingetragen, kann mit

make flash

der Controller mit der Firmware programmiert werden, der Basicinterpreter ist dann einsatzbereit. Rufen sie in diesem Verzeichnis mit

../tools/picocom -b 19200 /dev/ttyUSB0

ein Terminalprogramm auf (es kann natürlich jedes andere Terminalprogramm benutzt werden, bspw. < putty > vom Desktop aus. Der Interpreter wird sich melden mit:

```
STM8S103F3P6 TBasic

OK
>
```

Firmware mittels Bootloader flashen

Da die Codegröße des Interpreters 7730 Bytes beträgt und damit sehr knapp unterhalb der max. Größe liegt wenn ein Bootloader den Flashvorgang übernehmen soll, kann alternativ die Firmware über den Bootloader geflasht werden (die max. Codegröße bei der Verwendung eines Bootloaders beträgt 7744 Bytes). Der Vorteil bei der Verwendung eines Bootloaders besteht darin, ein eventuelles Update oder vllt. eine komplett andere Firmware zu installieren, ohne einen STLINK anschließen zu müssen. Hierzu ist jedoch – jedoch nur einmalig – ebenfalls ein STLINKv2 von Nöten um den Bootloader zu installieren.

Schließen sie an den entsprechenden Pins des Mikrocontrollers den STLINKv2 an und wechseln sie in das Verzeichnis < ./bootloaderv2 >

Dort befindet sich das Binary und das Sourceprogramm des Bootloaders. Die Sourcedatei kann nur übersetzt werden, wenn ein Assembler namens < naken_asm > installiert ist (ein Übersetzen ist jedoch nicht notwendig, da wie oben bereits geschrieben das Binary vorliegt).

Flashen Sie den Bootloader mittlerweile

make flash103

Fortan können Programme über die serielle USB2UART Brücke übertragen werden. Ist der Bootloader installiert, können sie den STLINKv2 wieder entfernen. Der Bootloader verbleibt solange im Controller, bis ein anderes Programm mittels STLINKv2 übertragen wird.

Um die Firmware des Basicinterpreters zu flashen wechseln sie wieder in das Verzeichnis < ./basict > und ändern mit einem Texteditor an den Einträgen ab:

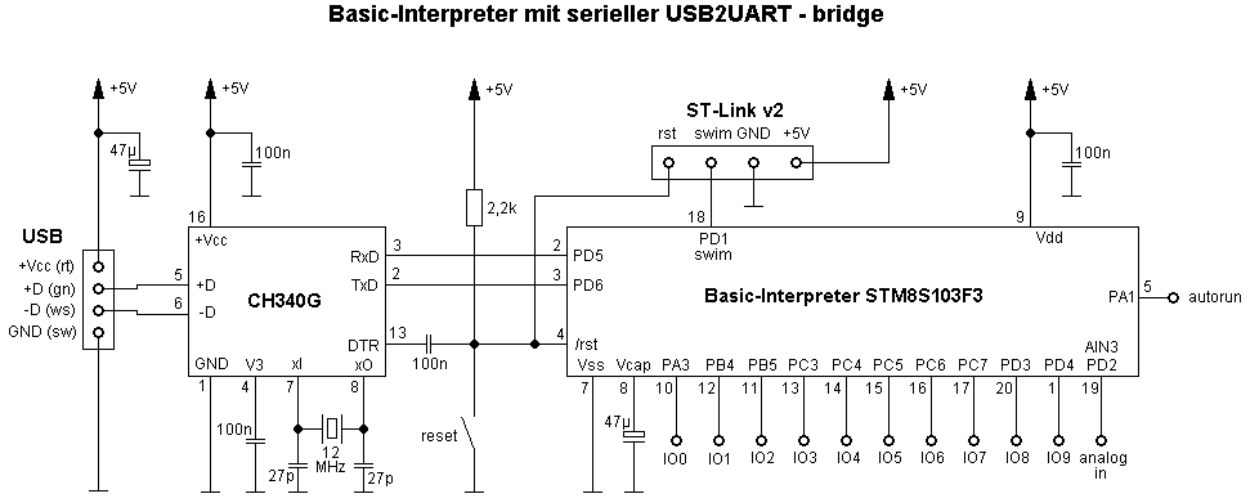
CH340RESET = 1
FLASHERPROG = 2

Bei angeschlossener serieller USB2UART Brücke wird der Controller nun über diese programmiert. Ein

make flash

wird dann auch hier die Firmware in den Controller übertragen.

Schaltplan



serielles Terminalprogramm

Um ein Programm eingeben zu können wird ein serielles Terminalprogramm benötigt. Dies kann zum einen das im Paket enthaltene Programm < picocom > oder ein anderes (bspw. < putty >) sein.

Picocom erwartet beim Start 2 Programmargumente:

picocom -b baudrate port

Bei Verwendung eines CH340G ist der Port < /dev/ttyUSB0 > (wenn der CH340G der einzige an den Computer angeschlossene virtuelle Comport ist). Wird eine USB2UART Brücke bassierend auf einem FTDI Chip verwendet ist dort der Port < /dev/ttyACM0 >

Aufruf bei Verwendung mit CH340G:

picocom -b 19200 /dev/ttyUSB0

Ein laufendes Picocom-Programm kann mit der Tastenkombination ALT – x beendet werden.

Der Basicinterpreter

Der Interpreter ist aufgrund der minimalen Hardware extrem "abgespeckt". Jeder Programmzeile muss eine Zeilennummer vorangestellt werden.

Ein laufendes Programm kann mit der ESC-Taste abgebrochen werden.

Variablename bestehen nur aus einem einzelnen Buchstaben. Variablebuchstaben sind gültig von a..z

Zusätzlich zu den Variablen existiert ein einzelnes Array, welches maximal 32 Werte aufnehmen kann:

Array

Das verfügbare Array wird durch das @ - Zeichen gekennzeichnet:

```
@(index)= Wert
```

```
10 print "Geben Sie 4 Zahlen ein"
20 for i= 1 to 4
30 print "Zahl ",i,": "
40 input z
50 @(i)= z
60 next i
70 print "Die Zahlen waren:"
80 for i= 1 to 4
90 z= @(i)
100 print "Zahl ",i,": ",z
110 next i
```

Programmbefehle

Folgende Programmbefehle sind implementiert:

- ABS - ADC - BYTEOUT - DELAY - FOR / NEXT - FUNC - GOSUB / RETURN – IN - IF - INPUT – OUT - PRINT - RND

ABS

Gibt den Absolutwert einer Zahl zurück (bei einer negativen Zahl wird das Vorzeichen entfernt)

```
10 B=-10
20 B=ABS(B)
30 PRINT B

Es wird 10 ausgegeben
```

ADC

Einen analogen Wert mit 10-Bit Auflösung einlesen. Der Analogeingang des Interpreters ist AIN3 / PD2. Beim STM8S103F3 ist das die Pinnummer 19

```
10 PRINT "ADC-Test"
20 PRINT "-----------"
30 PRINT
40 I=ADC()
50 PRINT " ADC-Value: ",I,"    ",
60 FUNC 1,13
70 DELAY (250)
80 GOTO 40
```

BYTEOUT

Ein gesamtes Byte wird auf I0 bis I07 ausgegeben.

```
Einen binären Zähler anzeigen

10 FOR I=0 TO 255
20 BYTEOUT (I)
30 DELAY (200)
40 NEXT I
50 GOTO 10
```

DELAY

Verzögert die weitere Programmausführung um die angegebene Anzahl Millisekunden.

```
Blinkprogramm, dessen Geschwindigkeit ueber ein Potentiometer das an den
ADC-Eingang angeschlossen ist, eingestellt werden kann

10 T=ADC()+50
20 OUT(0)=1
30 DELAY (T)
40 OUT(0)=0
50 DELAY (T)
60 GOTO 10
```

FOR / NEXT

Programmschleife

```
Führt Anweisungen wiederholt aus:

10 for z= 1 to 10
20 print z," * ",z," = ",z*z
30 next z
```

FUNC

führt einen "Systemaufruf" durch. Aufgrund der begrenzten Resourcen ist derzeit nur ein einziger Systemaufruf vorhanden.

FUNC Nummer, Argument

Systemfunktion

1 : gebe Asciizeichen aus
Argument: auszugebendes Ascii-Zeichen

```
10 for a= 65 to 90
20 func 1,a
30 next a
```

GOSUB / RETURN

Ein Unterprogramm aufrufen

```
10 out(0)= 0
20 gosub 200
30 out(0)= 1
40 gosub 200
50 goto 10
200 rem ############
210 rem delay
220 rem ############
230 delay(500)
240 return
```

IF

Bedingte Programmverzweigung

Der IF - Befehl ist in seiner Funktion eingeschränkt, er kennt nicht wie bei BASIC üblich ein THEN und ein ELSE, er kann lediglich bei erfüllter Bedingung zu einer anderen Programmzeile verweisen.

Befehlsbedingungen sind größer, kleiner, gleich (es gibt kein ungleich)

```
10 Z=10
20 PRINT "Zahl eingeben",
30 INPUT X
40 IF X>=10 GOTO 70
50 PRINT "Zahl ist kleiner als ",Z
60 STOP
70 PRINT "Zahl ist groesser-gleich ",Z
80 STOP
```

IN

Einen Portpin einlesen

```
10 i= in(3)
20 if i=0 goto 10
30 print "Pin ist 1"
40 goto 10
```

INPUT

Eine Variable über die serielle Schnittstelle einlesen

```
10 print "Zahl eingeben ",
20 input x
30 print "Die Zahl war: ",x
```

OUT

Einem Ausgangsportpin einen Wert zuweisen

out (pinnummer) = 0 setzt den Portpin auf logisch 0,
= 1 setzt den Portpin auf logisch 1

Das Pinmapping der Portpins

0 .... PA3
1 .... PB4
2 .... PB5
3 .... PC3
4 .... PC4
5 .... PC5
6 .... PC6
7 .... PC7
8 .... PD3
9 .... PD4

```
10 out(0)= 0
20 gosub 200
30 out(0)= 1
40 gosub 200
50 goto 10
200 rem ############
210 rem delay
220 rem ############
230 delay(500)
240 return
```

PRINT

Gibt Texte, Variableninhalte oder numerische Zahlen aus

```
10 print "Text"
20 print "Variable", a
```

Wird dem Print-Befehl ein Komma nachgestellt, erfolgt kein Zeilenvorschub:

```
30 print "Zahl: ",
40 print 4*3
```

RND

Erzeugt eine zufaellige Zahl im Bereich von 1 bis zum angegebenen Wert (inklusive)

```
10 PRINT "Enter druecken um zu wuerfeln"
20 PRINT "1 und Enter fuer Ende",
30 INPUT X
40 PRINT
50 IF X=1 STOP
60 Z=RND(6)
70 PRINT "gewuerfelt: ",Z
80 PRINT
90 GOTO 10
```

Befehle in der Kommandozeile

- LIST - LOAD - NEW - SAVE - SIZE

LIST: listet das aktuell im Arbeitsspeicher vorhandene Programm auf

LOAD: lädt ein Arbeitsspeicher aus dem EEPROM in den Arbeitsspeicher

NEW: löscht ein Programm aus dem Arbeitsspeicher

SAVE: speichert das im Arbeitsspeicher vorhandene Programm ins EEProm

SIZE: gibt den freien verfügbaren Speicher (Tokenanzahl) zurück
print size()

sendbas19200

Sie können Ihr Programm über ein Terminalprogramm in den Basicinterpreter eingeben. Diese ist jedoch recht unkomfortabel, zudem haben sie keine Möglichkeit mehrere Programme zu speichern. Eine Lösung hierfür ist < sendbas19200 >.

Sie können mit einem Texteditor Ihrer Wahl ein gültiges Programm erstellen und dieses mittels sendbas19200 an den Interpreter schicken. Speichern Sie hierzu Ihr Basicprogramm im Ordner

./stm8projects/basict

Hierfür öffnen Sie ein Terminalfenster und ein serielles Terminalprogramm Ihrer Wahl, bspw. < picocom > oder < putty > um den Basicinterpreter zu starten. Führen sie dort sicherheitshalber den Befehl „NEW“ aus, damit sichergestellt ist, dass kein Programm im Arbeitsspeicher vorhanden ist.

Wechseln Sie im Terminalfenster in den Ordner ./stm8/basict und geben dort ein:

./sendbas19200 meinbasicprogramm.bas /dev/ttyUSB0

Ihr Programm wird an den Basicinterpreter übertragen.

Autoload Funktion

Für den Basicinterpreter gibt es eine „Autoload-Funktion“. Hiermit ist es möglich, ein im Interpreter gespeichertes Programm mit Anlegen der Betriebsspannung an den Mikrocontroller dieses Basicprogramm zu starten, ohne dass der Start über ein serielles Terminalprogramm erfolgen muß. Hierfür gibt es der Mikrocontroller-Portpin PA1 (Anschlußpin Nr. 5 beim STM8S103F3) vorgesehen. Liegt an diesem Pin bei

anlegen der Betriebsspannung GND-Potential an, wird das Basicprogramm sofort gestartet. Liegt dort kein GND-Potential an wird der interaktive Eingabemodus (über ein serielles Terminalprogramm) aktiviert.

Anhang1: Basic Beispielprogramme

ADC-Test

Liest den ADC des Mikrocontroller aus und zeigt dessen 10-Bit Wert an. Dieser Wert wird in einen Spannungswert umgerechnet (hier wird angenommen, dass der Controller mit +5V betrieben wird).

```
10 print "ADC-Test"
20 print "-----------------"
30 i= adc()
40 print "  ADC: ",I,"   ",
50 l= i/4
60 k= (l*5) / 255
70 m= ((l*50)/255)
80 m= m-(k*10)
90 print k,".",m,"V     ",
100 func 1,13
110 byteout (I/4)
120 delay (250)
130 goto 30
```

Knightrider - Lauflich

```
10 print "Knighrider-Lauflicht"
20 b= 1
30 byteout (b)
40 gosub 130
50 b= b*2
60 if b= 128 goto 80
70 goto 30
80 byteout (b)
90 gosub 130
100 b= b / 2
110 if b= 1 goto 20
120 goto 80
130 rem ############
140 rem -- SPEED --
150 rem ############
160 d= in(8)
170 if d= 1 goto 200
180 delay (50)
190 return
200 delay (500)
210 return
```

ASCII-Ausgabe

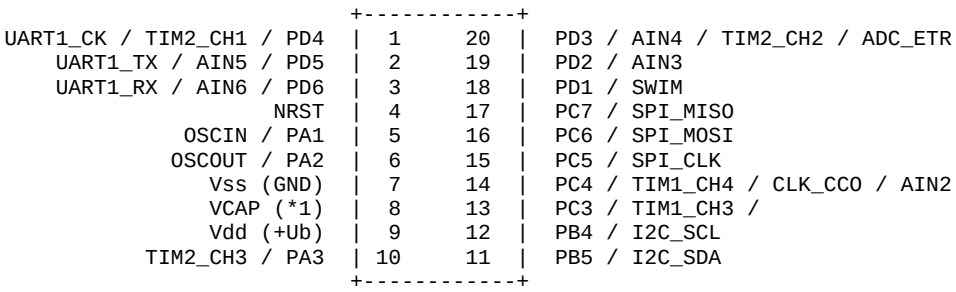
```
300 print "Die Ascii Tabelle"
302 print
305 b= 0
310 for a= 32 to 127
315 if a> 99 goto 320
317 func 1,32
320 print a,": ",
325 func 1,a
330 print "  ",
340 b= b+1
350 if b< 10 goto 380
360 func 1,10
370 func 1,13
375 b= 0
380 next a
390 print
400 stop
```

Fakultätsberechnung

```
10 rem ##################
20 rem Fakultaet
30 rem ##################
40 input "Eingabe Fakultaet : " x
50 f=1
60 for z=1 to x
70 f= f*z
80 next z
90 print "Fakultaet ",x," = ",f
100 stop
```

Anhang2: Pinbelegungen

Pinbelegung Mikrocontroler STM8S103F3P6



*1 : Ist mit min. 10µF gegen GND zu verschalten

Pinbelegung des STM8S103F3P6 Minimal-Board (China) mit alternativen Pinfunktionen
Hinweis: VCAP ist NICHT auf dem Board aufgelegt !!!