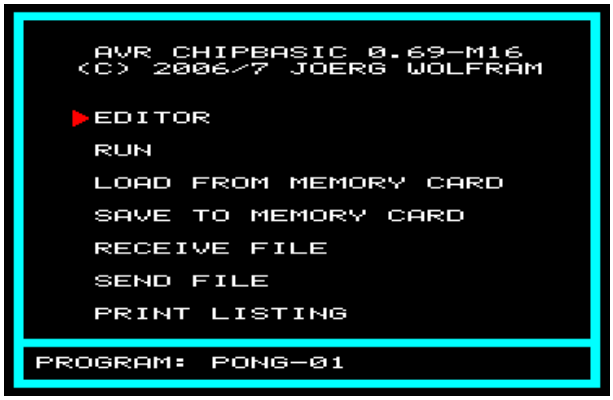
AVR-ChipBASIC: Ein BASIC-programmierbarer Einchip-Computer

V0.69 (c) 2006/2007 Jörg Wolfram



1 Lizenz

Das Programm unterliegt der GPL (GNU General Public Licence) Version 2 oder höher, jede Nutzung der Software/Informationen nonkonform zur GPL oder ausserhalb des Geltungsbereiches der GPL ist untersagt! Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGENDEINE GARANTIE, auch ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK.

2 Geschichte und Features

Nachdem ich mich nach langer 8051'er Zeit mit den mit der AVR-Assemblerprogrammierung beschäftigt hatte, war es einfach Zeit, die Grenzen ein bisschen auszuloten.
Herausgekommen ist dabei ein kleiner in einer Tiny-Basic sehr ähnlichen Sprache programmierbarer Computer mit umfangreichen Schnittstellen und Funktionen. Als Eingabegerät dient eine normale PS2-Computertastatur, als Ausgabegerät ein Fernsehgerät mit Scart-Eingang (Farbe) oder BAS-Eingang (Graustufen) oder auch verschiedene PAL-/NTSC-taugliche TFT-Displays.

- 20 Programmzeilen mit maximal 27 Zeichen, Fullscreen-Editor
- Basic-Programmiersprache mit abkürzbaren Schlüsselwörtern
- 30x23 Zeichen mit maximal 8 Farben, Pseudografik, Großschrift
- PAL/NTSC und Synchronsignale über Jumper einstellbar
- PS2-Tastatur zur Eingabe
- 1-Kanalige Audioausgabe (Noten, Rauschen) mit Hüllkurve
- serielle RS232-Schnittstelle mit 1200 Baud und Ladungspumpe
- parallele Druckerschnittstelle, auch als I/O und Analogeingänge nutzbar
- optionales Programm-EEPROM (24C64) für bis zu 16 Programme
- optionales Daten-EEPROM (24C64) für Datenlogger etc
- I2C-Anschluß für bis zu 8 LM75 Temperatursensoren
- Up/Download von Programmen über die serielle Schnittstelle

- Listingdruck über die Druckerschnittstelle
- automatischer Start des Programmes nach Einschalten über Jumper einstellbar
- Tastenkombinationen für Abbruch, Neustart und Screenshot

3 Hard- und Softwarekonzept

Kernstück des Ganzen ist ein ATmega16, die restliche Hardware besteht im Wesentlichen aus passiven Bauelementen und Steckverbindern. Optional kann ein 24C64-EEPROM als eine Art Disk angeschlossen werden, darauf lassen sich dann 16 Programme abspeichern. Timer 1 ist für das Videotiming zuständig. Kanal A arbeitet als PWM und erzeugt den Zeilen-Synchronimpuls. Im CSYNC-Mode wird während des vertikalen Synchronimpulses der PWM-Wert so hoch gesetzt, dass der Ausgang ständig auf LOW bleibt.
Kanal B von Timer 1 erzeugt den (Horizontal-) Videointerrupt. Dieser enthält eine Synchronisation auf den Timer, dadurch erzeugen auch EEPROM-Lesezugriffe mit 5 Takten CPU-Stop keine Bildstörungen. Timer 2 erzeugt PWM mit dem Audiosignal. Dieses wird entweder mit dem (Pseudo-)Zufallsgenerator oder per DDS mit Wavetable und Hüllkurve erzeugt.
Entgegen den meisten anderen Lösungen ist die PS2-Tastatur and den USART des ATMega angeschlossen. Eine Nutzung externer Interrupts verbietet sich schon wegen des Videotimings, man kann zwar das Clock-Signal während der Bilddarstellung auf LOW ziehen und damit die Tastatur zum Warten verdonnern, aber nicht jede Tastatur schafft es, ein Datenpaket während der Austastlücke zu senden und bei NTSC klappt das Ganze dann aber gar nicht mehr, weil dort die Austastlücke noch kürzer ist. Die zweite Möglichkeit wäre die SPI-Schnittstelle, aber die ist dann auch belegt. Und gerade während der Entwicklungsphase ist es lästig, wenn man ständig die Tastatur abziehen und anstecken muss. Die Nutzung des USART hat hingegen den grossen Vorteil, dass alles automatisch geht und man nur noch das fertige Zeichen abholen muss.
Wobei sich natürlich dann die Frage stellt „und was ist mit seriell?" Mit reduzierter Bitrate (1200Baud) lässt sich die serielle Schnittstelle im Horizontal-Interrupt mit zwei Portpins realisieren. Timerausgang von Timer 0 treibt eine Ladungspumpe, die ca. -4,7V für die serielle Schnittstelle bereitstellt, dadurch sind keine Spezialbausteine (MAX232 etc.) notwendig. Um Strom zu sparen, lässt sich diese Funktion auch per Software abschalten.
Die parallele Schnittstelle wird über Port A realisiert (+2 Steuerleitungen aus Port B). Wenn keine Druckerschnittstelle benötigt wird, können die 8 Pins als Ein-/ oder Ausgang konfiguriert oder auch als Analogeingang genutzt werden. Die meisten Funktionen werden über die **libmio** Bibliothek realisiert, die sich auch für eigene Projekte verwenden lässt. Allerdings steht dort noch die Dokumentation aus.

4 Systemvoraussetzung Host

Da fast alle I/O-Funktionen von der **libmio** bereitgestellt werden, sind zum Assemblieren avr_libmake (läuft nur unter Linux) und der AVRA-Assembler notwendig. Das Hex-File sollte sich auch unter anderen Betriebssystemen brennen lassen. Für die Funktionen der seriellen Schnittstelle sollte folgende eingestellt werden
8 Bit — 1200 Baud — no Parity — 2 Stopp-Bits

5 Tastatur

Die vorliegende Version benötigt eine deutsche Tastatur, funktioniert aber auch mit der englischen, wobei man die deutsche Tastenbelegung kennen sollte ;-) Bei Bedarf kann ich auch das Programm an andere Tastaturbelegungen anpassen. „Das Beste aus allen Systemen" ist in den Tastenkombinationen wiederzufinden:

- CTRL+ALT+DEL startet komplett neu, dabei werden alle Variablen gelöscht
- CTRL+C unterbricht das Programm zum nächstmöglichen Zeitpunkt
- CTRL+P sendet einen Screenshot an die serielle Schnittstelle

Für den Screenshot wird das Programm **screenshot.pl** im libmio/tools-Verzeichnis benötigt. Es benötigt ImageMagick und wird mit „./screenshot.pl dateiname.endung" aufgerufen. Danach kann dann der Screenshot vom Controllerboard ausgelöst werden. Wer die Funktion auf andere Betriebssysteme übertragen will, es ist recht einfach:

- alle Zeichen des Bildwiederholspeichers werden nacheinander mit gesetztem Bit 7 gesendet
- Am Schluss der Übertragung wird ein LF (0x0a) gesendet

6 Der Editor

Die Programme werden mit einem einfachen Fullscreen-Editor geschrieben. Ganz oben steht der Programmname und ganz oben links ein farbiges Quadrat. Sobald der Text verändert wird, ändert sich auch die Farbe dieses Quadrats von grün nach gelb. Die Position des Cursors wird durch ein grünblaues blinkendes Quadrat dargestellt. Unten befindet sich die Statuszeile, in der z.B. Fehler angezeigt werden.

- Standardmäßig wird eingefügt, Zeichen unter/rechts vom Cursor wandern nach rechts
- Die Cursor-Tasten bewegen den Cursor im Textfeld, bei Bedarf wird der Text gescrollt
- Mit der Taste **POS1** wird der Cursor auf das erste Zeichen der aktuellen Zeile gesetzt.
- Mit der Taste **ENDE** wird der Cursor auf das letzte Zeichen der aktuellen Zeile gesetzt.
- Mit der Taste **DEL** wird das Zeichen unter dem Cursor gelöscht, von rechts rücken Zeichen nach.
- Mit der **Backspace**-Taste wird das Zeichen links neben dem Cursor gelöscht, von rechts rücken Zeichen nach.
- Mit der **ENTER** Taste wird an den Anfang der nächsten Zeile gesprungen
- Mit der Tastenkombination **ALT+INS** wird an der aktuellen Cursorposition eine Zeile eingefügt. Alle Zeilen ab der Cursorposition wandern nach unten, die bisherige letzte Zeile geht dabei verloren.
- Mit der Tastenkombination **ALT+DEL** wird die Zeile der aktuellen Cursorposition gelöscht. Alle Zeilen unterhalb der Cursorposition wandern nach oben, unten wird eine Leerzeile angefügt.
- Mit der Taste **F2** kann das aktuelle Programmm im internen EEPROM gespeichert werden.
- Mit der Taste **F3** kann der letzte Programmstand aus dem internen EEPROM geladen werden.
- Mit der Taste **F8** wird das Programm im Editor gelöscht.
- Mit der Taste **F9** wird der Programmname geändert. Dabei funktionieren nur die Buchstaben- und Zifferntasten, sowie die Pfeiltasten nach links und rechts. Mit „ENTER” gelangt man wieder zurück.
- Mit der Taste **F10** wird das aktuelle Programm gestartet.

7 Die Dateiauswahl

Da zu den Programmen immer der Name mit abgespeichert wird, gibt es eine einfache Dateiauswahl. Mit den Cursortasten wird das zu ladende / zu speichernde Programm ausgewählt, **ENTER** startet den Transfer. Mit **ESC** gelangt man wieder zurück zum Hauptmenü.

8 Dateitransfer

Ab Version 0.69 wird anstelle von X-Modem normale Textübertragung verwendet. Unter Windows kann man weiterhin das Hyperterminal verwenden (Text senden/aufzeichnen) verwenden, unter Linux minicom oder das Program **chiptrans.pl** im Examples-Ordner.

8.1 Senden zum AVR

1. **RECEIVE FILE** am AVR auswählen
2. Transfer am PC mit **./chiptrans.pl w Dateiname** starten
3. am AVR wird das Programm, am PC der Status angezeigt

8.2 Empfangen vom AVR

1. Transfer am PC mit **./chiptrans.pl r Dateiname** starten
2. **SEND FILE** am AVR auswählen
3. am PC wird der Status angezeigt

9 Changelog

17.12.2006 Erste öffentliche Version (0.61)
12.1.2007 Bugfixes und Erweiterungen (0.69)

- Überarbeiteter Editor mit größerem Zeilenabstand und Scrolling
- Bugfixes beim Zugriff auf Daten-EEPROM
- Linien in Pseudografik und einfache „Sprites” mit Kollisionserkennung
- das Array liegt jetzt parallel zu den Variablen, @(0) entspricht der Variable A
- Umstellung der Dateiübertragung von X-Modem auf normalen Text-Transfer