EUzebox – Retrospielekonsole selbst gebaut

Alles begann mit einer uralten NES Konsole, die ein tristes Dasein in einem Karton in der Garage fristete. Beim Aufräumen gelangte diese Konsole wieder ans Tageslicht und wurde von meinen beiden Kindern ungläubig bestaunt. Mit solchem altertümlichen Equipment wurde also in der Steinzeit gespielt. Nach einiger Bastelei hüpfte dann auch wirklich Mario wieder über den Bildschirm. Leider fehlten die passenden Gamepads und so kam nicht wirklich Freude auf. Beim Betrachten des über 20 Jahre alten Innenlebens aus diversen DIL-Schaltkreisen und Spielemodulen von der Größe einer Toastscheibe stellte ich mir die Frage, welchen Aufwand man denn wohl heute im Zeitalter von ARM & Co. treiben müsste, um zu einem ähnlichen Ergebnis zu kommen. Die Antwort war schnell gefunden. Einen ARM oder eine andere 32-Bit Plattform braucht man gar nicht, um die bekannten 8Bit-Spieletitel aus den Achtzigern wieder über den Bildschirm flimmern zu lassen.

Bild 1: Tetris Clone Megatris

Im Internet stieß ich auf das ambitionierte Opensource Projekt Uzebox [1] des Kanadiers Alec Bourque. Die Uzebox nutzt einen übertakteten AVR Mikrocontroller, um alle Funktionen einer Spielekonsole wie Grafikausgabe am TV, Soundeffekte, Abfrage der Gamepads und Gameplay zu realisieren. Die Rechenleistung des Mikrocontrollers reicht aus, um auf einem Fernsehgerät 240x224 Pixel mit 256 Farben auszugeben. Der Sound wird dabei über 4 unabhängige Wavetable-Kanäle erzeugt. Als Gamecontroller (Gamepad) kommen die bekannten SNES-Controller zum Einsatz, die man für wenige Euro via eBay erhalten kann.
Eine kleine, aber stetig wachsende Fangemeinde hat bereits viele Spiele auf diese Plattform portiert. So stehen mittlerweile über 60 Demos und Spiele zum kostenlosen Download bereit. Darunter Klassiker wie Pacman, Arkanoid oder Tetris **Bild 1**, aber auch bekannte Spiele aus der Zeit, als MSDOS und Windows laufen lernten, sind darunter. Grafik und Sound sind dabei erstaunlich gut und entprechen häufig im Detail den Originaltiteln. Die Spiele können entweder via ISP-Anschluss mit einem Programmieradapter direkt in den AVR gebrannt oder via Bootloader, auch Gameloader genannt,

von einer SD-Karte geladen werden.
Da es sich um ein Projekt aus Übersee handelt, erfolgt die Ausgabe des Videosignals im NTSC Format. Das wäre an sich kein Beinbruch, da die meisten Fernsehgeräte neben PAL auch NTSC unterstützen, leider wird für die Umwandlung der RGB Signale in FBAS aber der AD725 von Analog Devices verwendet, der zum Teil nur schwer erhältlich und nicht gerade preiswert ist. Eine Umstellung auf PAL empfiehlt sich nicht, da ja dann auch alle Spiele geändert und neu compiliert werden müssten. Zum Glück gibt es ja in Europa den SCART Anschluss, der nach wie vor auch in Zeiten von HDMI noch an vielen Geräten zu finden ist und dieses zusätzliche Bauteil überflüssig macht. So entstand die Idee einer (EU)zebox mit SCART- Ausgang **Bild 2**.

Bild 2: EUzebox Platine (links) und im Gehäuse (rechts)

In [2] wurde zwar bereits eine Uzebox mit SCART-Ausgang vorgestellt, die Schaltung ist aber sehr spartanisch und enthält einige Fussangeln, da hier ein SCART-Kabel einseitig mit der Platine verbunden wurde. Im Normalfall sind verschiedene Ein- und Ausgangssignale im SCART Kabel aber gekreuzt, so dass sich dieser Schaltplan nicht ohne Modifikationen mit einer handelsüblichen SCART-Buchse nutzen ließ.
Herzstück des Schaltplanes ist der Mikrocontroller ATMEGA644, der mit 28,6363 Mhz übertaktet wird. Es ist sehr wichtig, dass hier nicht der pinkompatible ATMEGA644V verwendet wird, da sich diese Type nicht so hoch übertakten läßt. Alternativ kann entweder ein Quarz in HC49 Bauform Q1 oder ein röhrenförmiger Miniaturquarz Q2 bestückt werden. Das RGB-Signal wird über 3 Spannungsteiler für Rot, Grün und Blau an PortC erzeugt. Die Widerstände des Spannungsteilers sollten möglichst 1% Typen sein, um Farbverfälschungen zu vermeiden. Der Sound wird über den Ausgang PortD.7 ausgegeben und direkt über R9 mit der SCART-Buchse verbunden. Über den ISP-An-

schluss K2 muss zumindest die Programmierung des Bootloaders erfolgen.
Die Stromversorgung kann aus einem unstabilisierten Steckernetzteil mit 9 – 12V AC oder DC erfolgen. Dabei wird die Eingangsspannung vom Festspannungsregler IC2 auf 5V stabilisiert. Für das SD-Karteninterface werden zusätzlich 3,3V benötigt, die über den Lowdrop-Regler IC1 bereitgestellt werden.
Aus diesem Grund ist die SD-Kartenfassung auch nicht direkt mit dem Mikrocontroller verbunden. Über die Spannungsteiler R1 bis R6 werden die SPI-Ausgänge des Mikrocontrollers an die Spannungspegel der SD-Karte angepasst.
Die Buchsen zum Anschluss der SNES-Controller X1 und X2 sind nur sehr schwer zu bekommen. Der Autor bietete auf seiner Homepage [3] deshalb einen Bausatz mit Platine und allen Teilen inklusive dieser Buchsen an. Man kann sich aber auch sehr einfach behelfen, indem man den ursprünglichen Stecker am SNES-Controller durch einen 9poligen SUB-D-Stecker ersetzt und auf der Gegenseite ebenfalls eine SUB-D-Buchse verwendet. Der Controller ist nach dieser Behandlung natürlich nicht mehr mit einer SNES-Konsole verwendbar.
Mit dem Optokoppler IC4 kann ein optionales MIDI-Interface realisiert werden. Dieses Interface ist eigentlich nur für Spieleentwickler interessant, die den Sound simultan auf einem Keyboard komponieren und über die Konsole ausgeben möchten. Otto Normalverbraucher benötigt diese Erweiterung nicht. Um dieses Interface zu nutzen, muss K4 mit einer handelsüblichen DIN-5 Buchse verbunden werden. Dazu werden die Pins 3 und 5 der DIN-Buchse an Pin 1 von K5 und der Pin 4 an Pin 2 von K5 angeschlossen.
Soll die Platine in ein Gehäuse eingebaut werden, kann über die ebenfalls optionale Stiftleiste K4 ein externer Resettaster und eine externe LED als Betriebsanzeige angeschlossen werden.

Inbetriebnahme:

Nach dem Aufbau der Schaltung sollten zunächst ohne Mikrocontroller und gesteckte SD-Karte die beiden Betriebsspannungen geprüft werden. Es ist empfehlenswert, zunächst nicht den Bootloader, sondern ein beliebiges Spiel oder den Controllertester [4] in den Mikrocontroller zu brennen. Nun kann man ein SNES-Gamepad anschließen und die Funktion auf dem Bildschirm prüfen.
Wurde dieser erste Test erfolgreich bestanden, kann der Bootloader programmiert werden. Hier sind unbedingt die Einstellungen der Fusebits entsprechend der Hinweise in [1] zu beachten **Bild 3**. Nun kann man eine SD-Karte am PC formatieren und die gewünschten Spiele auf die Karte kopieren. Es sind nicht alle SD-Karten geeignet. Insbesondere mit SDHC kann es Probleme geben. Wenn die SD-Karte und der Bootloader funktionieren, erscheint eine Liste der Spiele auf dem Bildschirm. Mit den Pfeiltasten am Gamepad kann durch diese Liste navigiert werden. Durch Druck auf den Startknopf wird das ausgewählte Spiel dann geladen und ausgeführt.

Wer mag, kann die Leiterplatte auch in ein passendes Gehäuse einbauen. Zu diesem Zweck bieten wir ein formschönes Aluminiumgehäuse an. An den Frontseiten ist das Gehäuse mit entsprechend vorgefrästen, schwarzen Frontplatten aus Hard-PVC ausgestattet **Bild 4**.

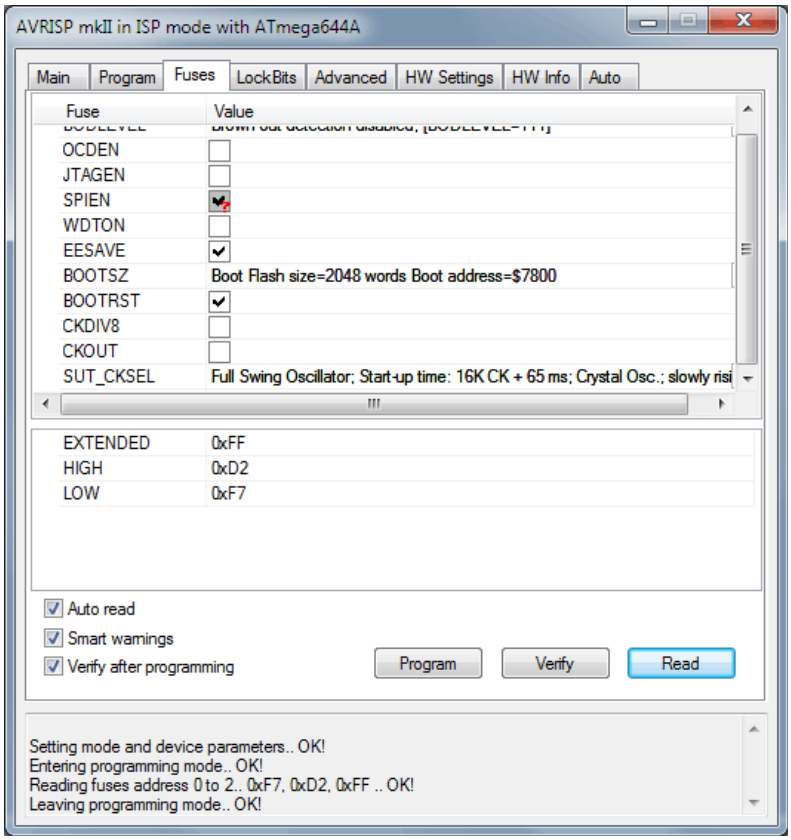

Bild 3: Fuse Einstellungen

Bild 4: EUzebox im Alugehäuse

Das eigene Spiel für die Uzebox:

Da die Uzebox ein OpenSource-Projekt ist, kann man auch selbst Spiele oder Anwendungen entwickeln. Dazu muss man nur grundlegende C-Kenntnisse aber kein Spezialwissen über ATMEL AVR Mikrocontoller besitzen. Alle hardwarenahen Funktionen sind in einem Kernel gekapselt. Der aktuelle Kernel 3.1 stellt daher alle Funktionen für die Grafik- und Soundausgabe sowie die Abfrage der Controller in einer API zur Verfügung. Der Entwickler muss sich „nur“ noch um die Erstellung der eigentlichen Grafik- und Sounddateien sowie die Programmierung des Gameplays kümmern. Dabei sind alle wichtigen Informationen zur API, Tools und Tutorials in einem Wiki gesammelt. Zusätzlich kann man sich auch Support über das angeschlossene Forum holen.
Möchte man selbst für die Uzebox Software entwickeln, sollte man sich zunächst die notwendige Entwicklungsumgebung auf dem PC installieren. Dazu installiert man den freien GNU-Compiler WINAVR und das ebenfalls kostenfreie AVR Studio (in dieser Reihenfolge). Da die Uzebox verschiedene Videomodes mit unterschiedlichen Eigenschaften unterstützt, sollte man sich vor der eigentlichen Spieleentwicklung mit den Vor- und Nachteilen der verschiedenen Videomodi beschäftigen und erst dann mit der Erstellung der Grafiken und Programmierung beginnen. In den meisten Fällen dürfte aber der Videomodus 3 für das eigene Spiel die richtige Wahl sein. Dieser Modus unterstützt Sprites und Scrolling und wird vom Kernel am besten unterstützt.
Für die eigentliche Entwicklung benötigt man übrigens keine funktionierende Uzebox. Es existiert ein sehr guter Emulator für den PC, auf dem man sehr einfach die übersetzten Programme testen kann. Mit diesem Emulator kann man sich natürlich auch einen Eindruck von schon vorhandenen Spielen auf dem PC verschaffen.
Um selbst erstellte Grafiken und (MIDI-)Sounds in C-Code zu konvertieren, kann man auf diverse Commandozeilen orienierte Tools zurückgreifen, die ebenfalls von der Projekt Homepage kostenlos heruntergeladen werden können.
Im Kernel ist auch eine Unterstützung der sonst nicht genutzten UART des Mikrocontrollers enthalten. Daraus ergeben sich vielfältige Möglichkeiten, die Uzebox-Plattform auch für andere Anwendungen zu nutzen. Beispielsweise ließe sich so einfach ein Funkmodul anbinden, um Einrichtungen in Haus und Hof fernzusteuern.

Meine Kinder waren dann von den einfachen Retrogames doch begeistert. Weniger ist ja bekanntlich oft mehr und schlechtes Gameplay oder eine dünne Spielidee lassen sich halt auch von aktueller 3D-Grafik nicht ausgleichen.

Quellen:

[1] Bourque, A., The Uzebox project,
http://www.uzebox.org

[2] F0lken, Uzebox-SCART
http://znoxx.wordpress.com/projects/uzebox-scart/

[3] Wendt, H., Homepage des Autors
http://www.hwhardsoft.de

[4] Zavan, F., nebososo.com
http://www.nebososo.com/uzebox/ctester.php