AVR-ChipBasic2 - Hardware-Erweiterungen

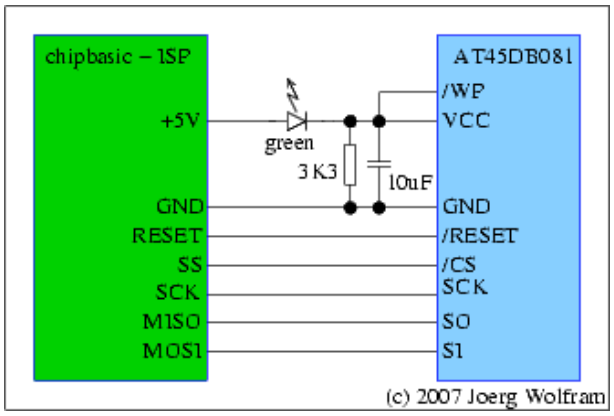
V1.50 (c) 2006-2015 Jörg Wolfram

1 Das Dataflash-Modul

Das erste Zusatzmodul ist ein ATMEL DataFlash und wird auf den (normalerweise freien) ISP-Steckverbinder gesteckt. Auf dem Modul können wie auf einer Diskette oder einem USB-Stick (als Analogie zum PC) Dateien gespeichert werden. Entweder BASIC- oder Binärprogramme, aber auch USER-Dateien, z.B. für Datenlogger. Das Ganze funktioniert mit dem AT45DB041B (512 kBytes) und dem AT45DB081B (1MByte). Der Unterschied zwischen beiden Bausteinen liegt nur in der Anzahl der verfügbaren Pages (2048/4096). Die D-Typen sollten auch funktionieren, wurden aber nicht getestet.

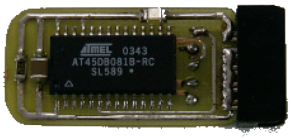
1.1 Schaltbild

Da der Dataflash 2,7-3,3 Volt Versorgungsspannung braucht, wird diese einfach durch eine in Reihe geschaltete grüne oder gelbe LED mit ca. 1,8V Durchlassspannung erzeugt. Ein Widerstand mit 3,3kOhm bildet die Grundlast für die LED, parallel dazu liegt ein Abblock-Kondensator. Dieser sollte nicht zu klein sein, damit es beim Schreiben in den Flash keine Probleme wegen des differentiellen Widerstands der LED gibt.

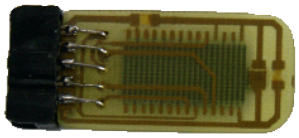


1.2 Leiterplattenlayout

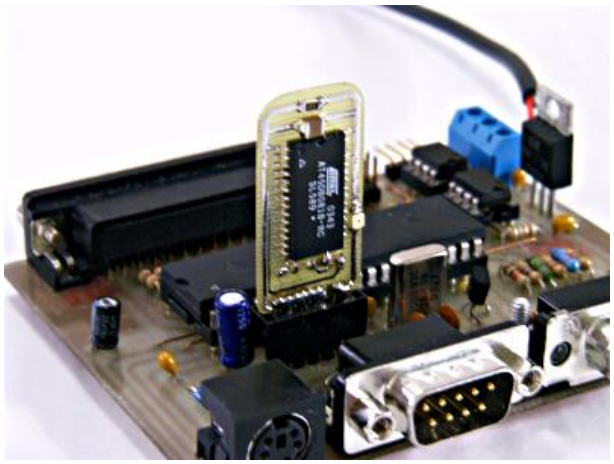
Die Leiterplatte besteht aus 0,4mm Epoxy, damit sie zwischen die zwei Reihen eines zehnpoligen Buchsenteils für die ISP-Schnittstelle passt. Die Kontakte auf der Leiterseite werden direkt aufgelötet, die auf der anderen Seite befindlichen Kontakte müssen mittels kurzen Drähten mit den Anschlüssen auf der Leiterplatte verbunden werden. Dadurch steht die Leiterplatte zwar nach oben, lässt sich aber recht gut stecken und auch wieder entfernen.
Die Leiterplattenversion 2 benutzt „Leiterplatten-Steckverbinder", wird waagerecht eingebaut und liegt dann über dem Mikrocontroller. Aufgebaut habe ich sie aber nicht, da ich Version 1 bevorzuge.



Über dem Dataflash im SOIC-28 Gehäuse ist die SMD-LED zu sehen, links ein keramischer 10uF Kondensator und ein 3,3KOhm Widerstand als Grundlast für die LED.



Auf der gegenüberliegenden Seite befinden sich nur 5 Drahtbrücken für den Steckverbinder.



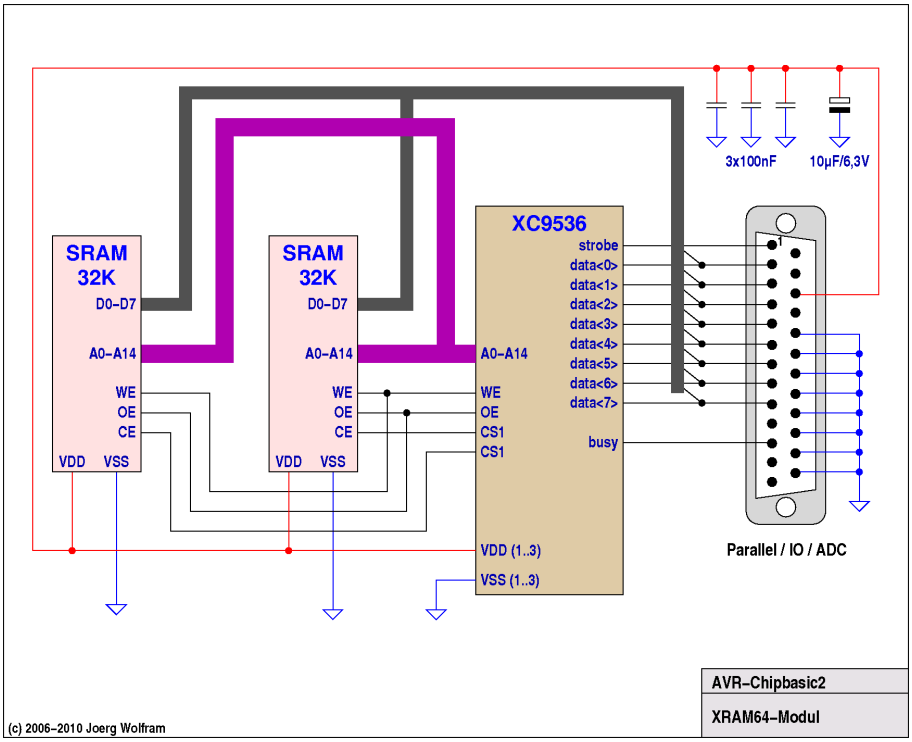
BASIC-Computer mit gestecktem Dataflash-Modul.

2 Die 64K Speichererweiterung XRAM64

Das XRAM64 Modul stellt 64 KBytes zusätzlichen Speicher (XMEM) zur Verfügung. Angeschlossen wird es am Parallelport, dieser ist darurch für andere Anwendungen nicht nutzbar. Binärprogramme können sowohl direkt auf den externen Speicher zurückgreifen als auch die entsprechenden Treiberroutinen benutzen, für BASIC Programme muss auf Programmplatz 8 ein Treiber mit der notwendigen Funktionalität geladen sein.

2.1 Schaltbild

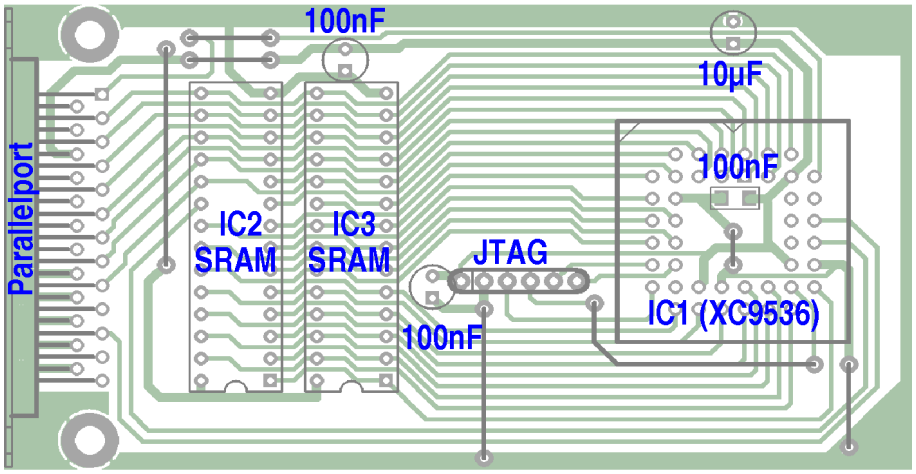
Die 64K Speichererweiterung besteht im Wesentlichen aus zwei SRAM mit 32 Kbyte und einem CPLD und wird an den Parallelport angesteckt. Dabei müssen aber die +5V an Pin 14 der parallelen Schnittstelle liegen, bei der Originalversion muss dies nachgerüstet werden.



Beim CPLD funktioniert auch das relativ langsame XC9536-15-PC44, bei den SRAM sollte die Zugriffszeit unter 50ns liegen, ansonsten müssen in die Treiberroutinen evtl. Waitstates eingebaut werden. Die genauen Werte der Blockkondensatoren sind unkritisch.

2.2 Leiterplattenlayout

Das Leiterplattenlayout ist wieder einseitig, dafür müssen allerdings 7 Drahtbrücken bestückt werden. Natürlich wäre auch ein zweiseitiges Layout möglich, aber einlagige Leiterplatten lassen sich mit weniger Aufwand selbst herstellen. Über den JTAG Stecker kann das CPLD programmiert werden.

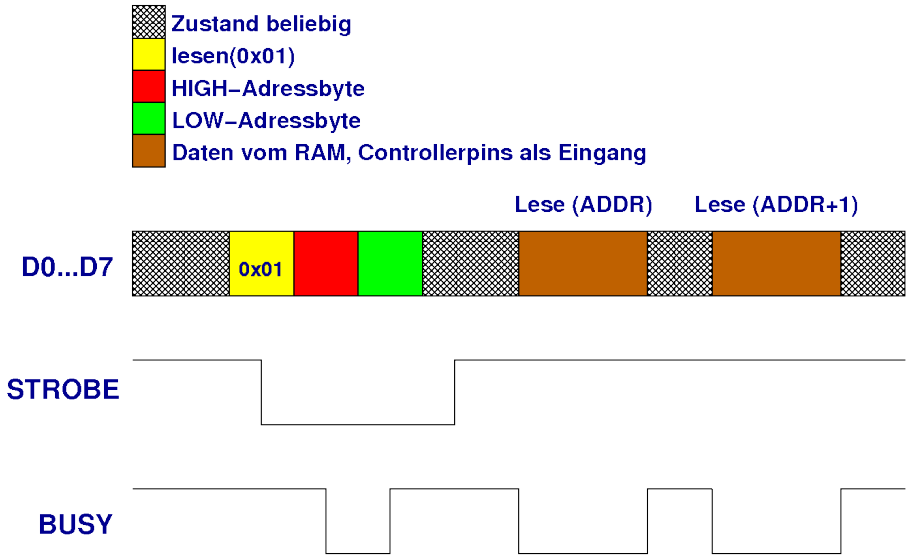


2.3 Ansteuerung

Für die Realisierung stand nur der Parallelport zur Verfügung, dadurch müssen die Adressleitungen zwischengespeichert werden. Die ließe sich auch durch zwei Register realisieren. Eine Besonderheit ist aber der schnelle inkrementale Zugriff, der für Lese- und Schreibzugriffe effektiv nur jeweils 3 Prozessortakte benötigt. Somit ist zum Beispiel eine Videoauflösung von 320x240 bei 16 gleichzeitig darstellbaren Farben möglich. Um das Modul mit eigenen Routinen im Rahmen eines Treibers oder eines Binärprogrammes ansteuern zu können, sind nachfolgend die verschiedenen Zyklen beschrieben.

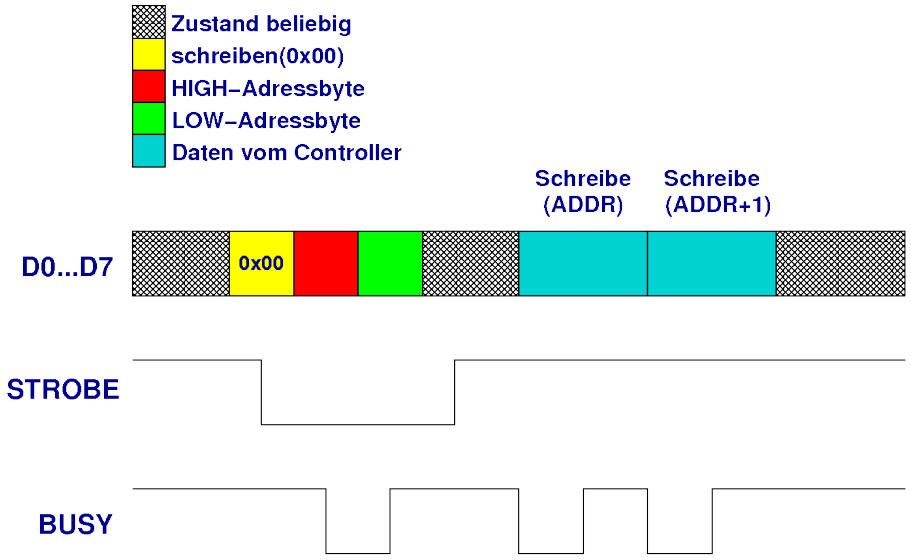
- Mit der fallenden Flanke von **STROBE** wird das Kommando von den Datenleitungen eingelesen. Derzeit wird nur Bit 0 benutzt, um zwischen Lesen (1) und Schreiben (0) zu wählen.
- Mit der fallenden Flanke von **BUSY** werden bei **STROBE=LOW** die oberen 8 Adressbits von den Datenleitungen eingelesen und in einem Register zwischengespeichert.
- Mit der steigenden Flanke von **BUSY** werden bei **STROBE=LOW** die oberen 8 Adressbits vom Register und die unteren 8 Adressbits von den Datenleitungen in den Adresszähler geschrieben.
- Mit der steigenden Flanke von **BUSY** wird bei **STROBE=HIGH** der Adresszähler um 1 erhöht.
- Die Signale **OE**, **WE**, **CS1** und **CS2** werden nur aktiv, wenn **BUSY=LOW** und **STROBE=HIGH** sind.

Timing für das Lesen vom XRAM



Beim Lesezyklus muss beachtet werden, dass das Einlesen der Portpins in den AVR eigentlich schon einen Takt früher erfolgt. Ein Readzyklus ist damit zwangsläufig mindestens einen Taktzyklus länger, aber in dieser Zeit kann z.B. der aus Registern gebildete Adresszähler im ATMega inkrementiert werden.

Timing für das Schreiben ins XRAM



In den XRAM64 Treibern findet man auch Codesequenzen zum Lesen und Beschreiben, die sich auch für eigene Ansteuerungen verwenden lassen.