SDIO

SD Karten Interface für den NDR Computer

Stand Juni 2017

Wichtiger Hinweis:

In dieser Anleitung wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage oder Lizenzrechte Dritter mitgeteilt. Sie sind ausschließlich für private Hobbyzwecke als auch Lehrzwecke bestimmt, eine gewerbliche Nutzung ist ausdrücklich untersagt.
Es wird keine Garantie noch eine juristische Verantwortung sowie irgendeine Haftung vom Autor übernommen.

Die Rechte an Firmennamen, Logos und Warenzeichen, die in dieser Anleitung genannt werden, liegen bei den Jeweiligen Inhabern.

Inhalt:

Funktionsbeschreibung der SDIO

Der NKC SD Controller

Klassische Mikrocomputer wie der NKC bekommen mit den Jahren alle das gleiche Problem: Verschleiß beim Massenspeicher. Alte Floppylaufwerke sind nur aufwändig zu reparieren, Festplattenlaufwerke gar nicht, und kompatibler Ersatz ist kaum aufzutreiben. Die IDE Schnittstelle ist nicht brauchbar spezifiziert und viele Open Hardware Controller laufen nur mit bestimmten Platten. CF und PATA sind 2017 beinahe schon ausgestorben und SATA ist nicht mit klassischer Technik adaptierbar. USB benötigt sehr komplexe Software. SD Cards hingegen gibt es schon lange und wird es noch lange geben, sie sind klein und billig, brauchen wenig Strom und können mit einfachen Schnittstellen und Treibern gut arbeiten.

Darum besteht der SD Controller auch nur aus langfristig leicht erhältlichen Bauteilen. Er stellt keine besonderen Anforderungen an den Bus und ist für verschiedene Systeme adaptierbar. Trotz des einfachen Aufbaus führt er den seriellen Datentransfer zur SD Card in Hardware durch, so dass faktisch die Busschnittstelle die Transferrate bestimmt. Durch den Aufbau als SPI Controller ist auch der Einsatz für andere Zwecke denkbar. Entsprechend der Tradition des NKC gibt es eine ausführliche Schaltungsbeschreibung.

Viele offene SD Card Projekte ignorieren Teile des SD Card Standards. Die Erfahrung aus der Praxis zeigt, dass diese Vorgehensweise leicht zu unzuverlässigem Betrieb oder Kompatibilitätsproblemen mit bestimmten Karten führt. Hier lohnt sich sorgfältige Arbeit, auch wenn der Standard nicht immer leichte Lektüre ist.

Leider ist nur die vereinfachte Fassung der SD Card Physical Layer Specification frei verfügbar. Ab Version 3 fehlen dort wichtige Abschnitte zu SPI, die nur in der kostenpflichtigen Version enthalten sind. Allerdings fand die letzte Aktualisierung von SPI in Version 2 statt und ist dort in der frei verfügbaren Version noch vollständig enthalten.

Die Datenübertragung von SD Cards

Zugunsten des einfacheren Aufbaus benutzt der SD Controller SPI. SD Cards verwenden den Mode 0, d.h. sie übernehmen die Daten mit der steigenden Flanke und ändern sie mit der fallenden Flanke des Takts. Tatsächlich ist Letzteres für SD Cards nicht ganz richtig, aber dazu später mehr.

Das Herzstück des SPI Controllers besteht aus zwei Schieberegistern: Der '165 realisiert MOSI, der '595 MISO.

Der '165 legt das Datenbit bereits auf den SPI-Bus, wenn das

Schieberegister geladen wurde. Für den Zweck von SPI müsste man ihn also nur 7 mal takten, um die weiteren niedrigwertigeren Bits auszugeben. Das serielle Eingabebit wird auf high gezogen, damit das Register nach Ablauf der Ausgabe 0xff enthält. Der 8. Takt sorgt dafür, das letzte Bit auf 1 zu setzen. Die Ausgabe muss mit der fallenden Flanke von SCLK erfolgen, darum wird mit dem invertierten Takt geschoben.

Der '595 soll das Eingabebit mit der steigenden Flanke von SCLK übernehmen, darum wird das Schieberegister direkt mit dem Takt gesteuert. Der invertierte Takt gibt das Schieberegister aus, denn laut Datenblatt kann man nicht gleichzeitig die Daten schieben und ausgeben. Alternativ könnte man die Ausgabe auch nur einmalig am Ende des Zyklus auslösen.

Die 8 Takte für die zwei Schieberegister werden durch den asynchronen Zähler (ripple counter) '393 erzeugt. Im Ruhezustand hält das Flipflop '74 den Zähler gelöscht. Wird es durch eine steigende Flanke gesetzt, beginnt der Zähler mit seiner Arbeit. Da dieser Vorgang nicht taktsynchron erfolgt, wird der erste Impuls angeschnitten. Um diesen Impuls zu unterdrücken, wird die erste Stufe des Zählers als Teiler benutzt, denn im geteilten Takt fällt der erste Impuls weg. Der Zähler läuft also nicht bis 8, sondern bis 16. Das Löschen des Zählers passiert, wenn er von 15 auf 16 schaltet. Die nötige Zeit zum Löschen ist kein Problem, weil genau dieser Impuls wie der erste angeschnittene Impuls durch die Teilung unterdrückt wird.

Es ist wichtig, keinen synchronen Zähler wie z.B. einen '163 zu verwenden, weil dieser Spikes auf den Ausgängen erzeugt.

Es gibt zwei Wege, die Datenübertragung zu starten:

- Durch die steigende Flanke von /WR beim Schreiben des Datenregisters, um das geschriebene Byte sofort zu übertragen.
- Durch die steigenden Flanke von /RD beim Lesen des Datenregisters. Sofern ein Bit im Statusregister gesetzt ist, wird so direkt die nächste Übertragung gestartet.

Der Verzicht auf extra Portzugriffe zum Start der Übertragung beschleunigt den Treibercode erheblich.

Während der Initialisierung der Karte darf der Takt nur maximal 400 kHz betragen. Darum kann man mit einem Bit im Statusregister zwischen einem entsprechend auf maximal 800 kHz geteilten (die SPI Übertragung teilt den Takt ja nochmals, darum nicht 400 kHz) oder dem ungeteilten Takt wählen.

Spannungsversorgung der SD Card

SD Cards arbeiten per default mit 3,3 V, die von einem LDO erzeugt und mit 47 uF (Physical Layer Specification E.1) und 100 nF (Physical Layer Specification E.2) geglättet werden.

Der einzige Weg, eine SD Card sicher neu zu initialisieren, ist die Unterbrechung der Spannungsversorgung. Dies wird durch einen schaltbaren LDO erreicht. Bei Tests zeigte sich, dass SD Cards auf Verletzungen des Protokolls gerne nicht durch einen Fehlercode, sondern mit Hängen reagieren.

Levelshifter für die SD Card

SD Cards haben keine 5 V toleranten Eingänge, d.h. es werden Levelshifter benötigt. Dazu dient ein mit 3,3 V Versorgungsspannung betriebener 74HC4050, dessen Eingänge unabhängig von der Versorgungsspannung bis zu 15 V tolerieren. Für MOSI gibt es bei allen Signalhöhen ausreichende Abstände. Leider kann der 74HC4050 nicht für MISO benutzt werden, weil V_OH der SD Card keinen ausreichenden Abstand zu V_IH des 74HC4050 hat:

SD Card (V_DD = 3,3 V, Physical Layer Specification 6.6.1)
V_OH = min 0,75 * V_DD = min 2,48 V
V_OL = max 0,125 * V_DD = max 0,41 V
V_IH = min 0,625 * V_DD = min 2,07 V
V_IL = max 0,25 * V_DD = max 0,83 V

74HC4050 (V_CC = 3,3 V)
V_OH = min 3,2 V
V_OL = max 0,1 V
V_IH = min 2,39 V (interpoliert)
V_IL = max 0,9 V (interpoliert)

74HCT595
V_IH = min 2,0 V
V_IL = max 0,8 V

74HCT165
V_OH = min 4,4 V
V_OL = max 0,1 V

In der Praxis hat sich gezeigt, dass die überprüften SD Cards alle deutlich mehr Spannung bei high lieferten. Dennoch wird im Sinne der Robustheit ein '125 als Levelshifter für MISO verwendet.

Auf den SD Card Anschlüssen 1,2,7,8,9 gibt es entsprechend der Physical Layer Specification Abschnitt 6 Pullup Widerstände, hier mit 47k.

Timing der SD Card

Der Takt von 10 MHz ist für den 74HC4050 akzeptabel, während ein CD4050 evtl. schon überfordert ist. 20 MHz funktionieren evtl. nicht mehr stabil. Leider macht das Datenblatt keine Aussage zum Betrieb bei 3,3V.

Der Abschnitt 6.6.6 Bus Timing (Default) der Physical Layer Specification bezieht sich auf das SD Card Protokoll und für SPI gibt es keine anderen Angaben, aber auch keinen Grund, von anderem Zeitverhalten auszugehen. Der Wert Output Delay time during Data Transfer Mode beträgt 14 ns, d.h. die Daten liegen nicht mit dem Flankenwechsel stabil an, sondern erst bis zu 14 ns später. Das wird im Diagrammen häufig falsch dargestellt.

Während der Initialisierungsphase kann diese Verzögerung noch deutlich höher sein!

Das Propagation Delay des 74HC4050 liegt zwischen 85 ns bei 2,0 V und 17 ns bei 4,5 V. Es ist zusammen mit der Verzögerung der SD Card also nicht zu vernachlässigen. Aus dem Grund wird das Schieberegister zum Empfang der Daten mit dem 3.3V Takt der SD Card betrieben, so dass sowohl Takt als auch Daten die gleiche Verzögerung erfahren.

Businterface

Es darf nicht passieren, dass die Treiber von CPU Karte und IO Karte gegeneinander arbeiten. Idealerweise wird ein Bustreiber erst aktiviert, wenn er auch ein Eingangssignal hat.

Die CPUZ80 Karte verzögert leider die Ansteuerung ihres Bustreibers unnötig. Um ein IC zu sparen, wurden statt einem AND zwei NAND Gatter eingesetzt, so dass die Richtung des Treibers unnötig spät umgeschaltet wird. Einzig das Propagation Delay der Bustreiber für die Steuersignale hilft, indem es /RD und /IORQ verzögert. Dennoch sehen Peripheriekarten schon ein /RD, während der Bustreiber der CPU-Karte noch kurz in der anderen Richtung arbeitet.

Es ist bei Peripheriekarten sehr beliebt, Adressen sowie /IORQ und /M1 mit einem 74HCT688 zu dekodieren und mit /RD die Richtung den Datenbustreibers zu steuern. Dies hat allerdings zur Folge, dass bei typischen Verzögerungen aller Bausteine nur 10 ns zwischen der Umschaltung des CPU-Bustreibers und des Karten-Bustreibers vergehen.

Ein 74LS245 braucht typischerweise 15 ns zum Ab- und 27 ns zum Einschalten, ein 74HCT245 aber nur 16 ns zum Einschalten. Eine gemischte Bestückung reduziert den ohnehin nicht ausreichenden Spielraum für Abweichungen vom typischen Verzögerungswert weiter. Typischerweise funktioniert es gerade noch so eben, aber Abweichungen innerhalb der Spezifikation können zu Störungen führen.

Aus diesem Grund wird das Signal zur Aktivierung des Bustreibers noch ein wenig verzögert. Die Schaltung sorgt dafür, dass die Deaktivierung weniger verzögert wird.

Das Schieberegister zur Wandlung des seriellen SPI-Datenstroms in parallele Bytes hat leider ein erhebliches Output Delay. Zusammen mit dem Propagation Delay des Dekoders für die Chip Select Signale ergeben sich

etwa 30 ns, in denen der Bustreiber auf der IO-Karte schon aktiv ist, aber noch kein Signal anliegt. Wie sich zeigte, neigt ein 74HCT245 in dieser Zeit zum Schwingen auf Leitungen, die im letzten Lesezyklus low waren und durch das Schieberegister high werden. Wenn das viele Bits betrifft, ergeben sich Nadelimpulse, die auf dem ganzen Controller zu finden sind und bei nicht sehr großzügiger sternförmig verlegter Masse sogar zu erheblichem ground bounce führen. Dieses Problem der kurzfristig undefinierten Leitungen wird mit Pullup-Widerständen auf dem internen Datenbus vermieden.

Die Alternative wäre eine weitere Verzögerung der Aktivierung des Treibers, aber es zeigte sich, dass das Propagation Delay einzelner Gatter teilweise überraschend vom typischen Wert im Datenblatt abweicht, so dass sich kein innerhalb der Spezifikation sicherer Aufbau ergibt. Weiterhin blieben dann die Eingangssignale der Bustreiber auf der CPU-Karte undefiniert, was dort evtl. das gleiche Problem verursachen könnte.

Ports

Der SD Controller belegt zwei Ports. Die Defaultadresse ist 0x20 (Jumper 1101111).

Schreiben

0:[7-0] Byte zur SD Card senden.

1:[7] SLOW (1 = max. 400 kHz Takt, 0 = volle Geschwindigkeit)
1:[6] ONLYRD (1 = Nur lesen, 0 = Lesen und Transfer starten)
1:[5-4] reserviert
1:[3] POWER (1 = SD Card eingeschaltet, 0 = ausgeschaltet)
1:[2] reserviert
1:[1] /CS1 (Card select SD Card 1)
1:[0] /CS0 (Card select SD Card 0)

Lesen

0:[7-0] Letztes empfangenes Byte lesen. Falls SINGLERD 0 ist, wird danach ein neuer Transfer von 0xff ausgelöst.

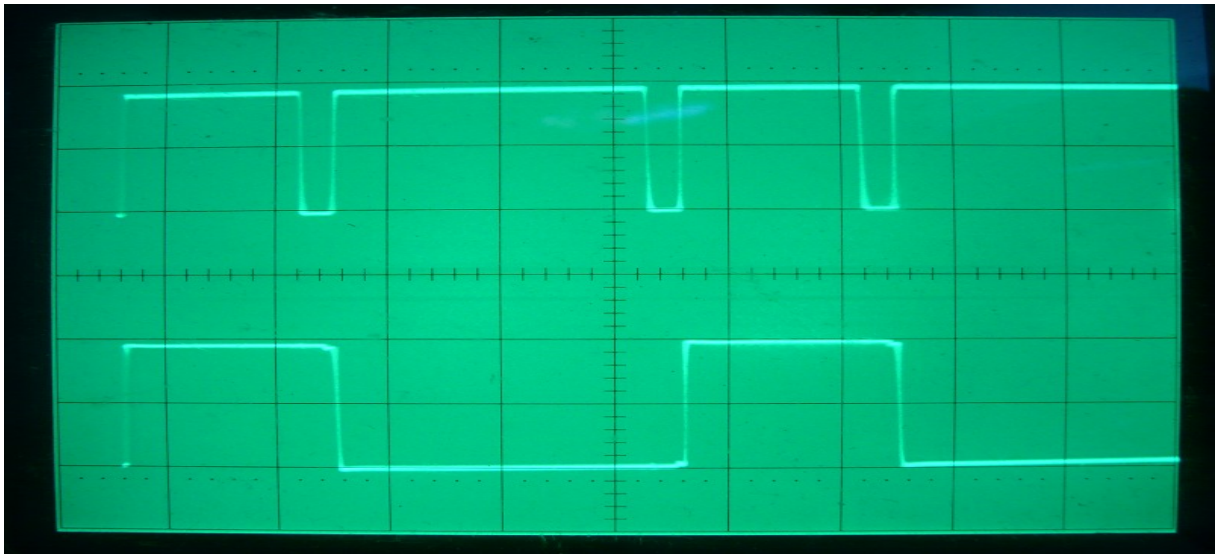
1:[7-1] reserviert
1:[0] BUSY (1 = Transfer läuft, 0 = Transfer fertig)

Testprogramme

Michael hat für den gesamten Aufbau diverse Testprogramme (siehe zip Datei) entwickelt. Diese sind hier kurz beschrieben:

sdcard0.z80 Quellcode
sdcard0.com ausführbare Datei

SD-Card Statusport ansprechen.
Das wiederholte Ansprechen des Statusports erzeugt ein periodisches Signal am Adresskomparator (IC 3 Pin 19) und von /WE1 (IC 15A Pin 5). Steuerport Bit 7 (IC8 Pin 19) wird getoggelt.



sdcard1.z80 Quellcode
sdcard1.com ausführbare Datei

SD-Card Takterzeugung durch /WE0 bei reduzierter Geschwindigkeit.

Das wiederholte Ansprechen des Datenports erzeugt ein periodisches Signal am Adresskomparator (IC 3 Pin 19) und von /WE0 (IC 15A Pin 4).

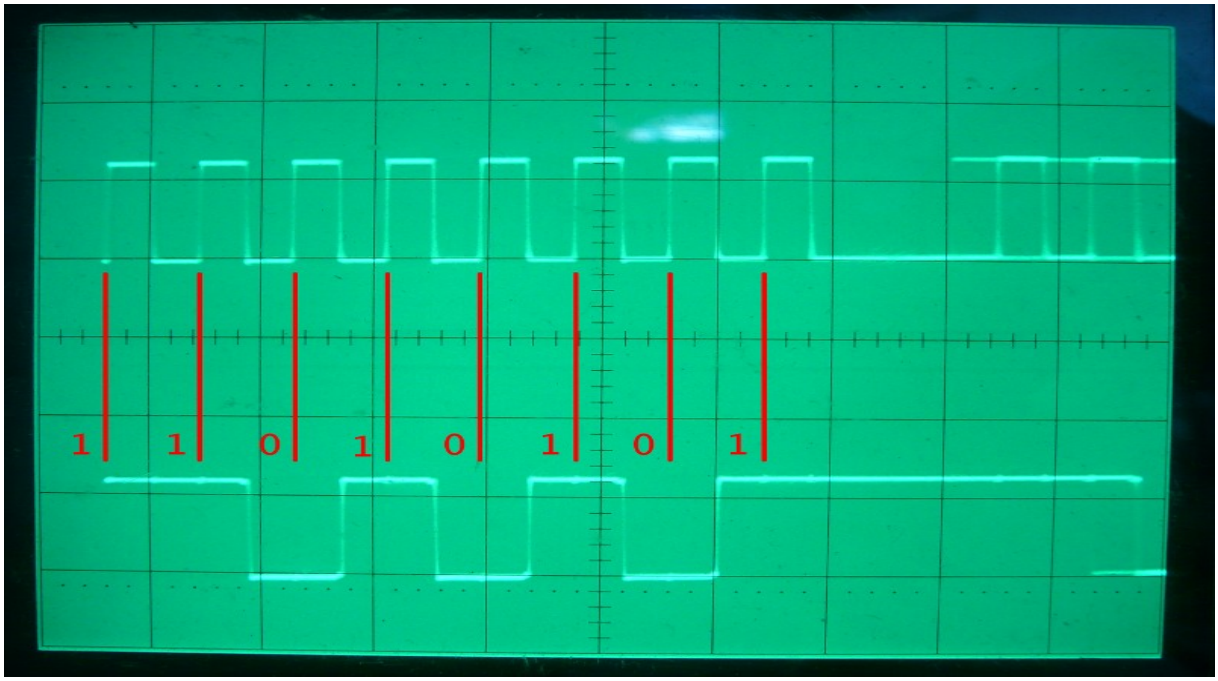
Steuerport Bit 7 (IC8 Pin 19) ist high, so dass der geteilte Takt am Zählereingang (IC17 Pin 1) ankommt. Die übrigen Bits des Steuerports sind high, so dass Spannung an der Karte anliegt, aber keine Karte selektiert wird. Immer, wenn in den Datenport (IC 6) geschrieben wird, sollte die steigende Flanke von /WE0 das Flipflop takten (IC 19A Pin 3), so dass es high wird und den Zähler (IC 17) für 16 Taktimpulse laufen laesst. Die halbierte Taktrate (312 kHz) wird ausgegeben, so dass 8 Takte an Pin 5 der SD-Card Sockel zu sehen sind, die jeweils doppelt so lang wie der Eingangstakt sind. Durch die Halbierung entfaellt der erste Taktimpuls, der in der Regel angeschnitten ist, weil die Busschnittstelle

durch einen anderen Takt synchronisiert wird.

Bei 625 kHz Oszillatortakt dauern 16 Takte 25,6 us oder 80 T States des Z80. Ein NOP dauer 4 T States = 1 us @ 4 MHz. 25 NOPs decken also die Zeit einer Uebertragung ab. Das jp sorgt danach für eine Pause.

Das Oszilloskop sollte mit der steigenden Flanke von SCLK (SD Card Pin 5) getriggert werden. Mit gut 2 us/div sieht man etwa einen Zyklus. Da die steigende Flanke nicht synchron zum Takt ist, verschiebt sich der Beginn des nächsten Zyklus auf dem Oszilloskop, so dass sich ein verwaschenes Bild ergibt. Ein Takt sollte etwa 3,2 us lang sein.

Dazu zeichnet man das Datensignal zur Karte auf (SD Card Pin 2). Das Muster 0d5h ist 11010101: Man kann das MSB am Anfang als solches erkennen und hat ansonsten alternierende Datenbits, die erlauben, die Beziehung zum Takt darzustellen. SD Karten benutzen SPI mit Polarität und Phase 0, d.h. der Takt beginnt mit einer low Phase, während der das Datenbit bereits anliegt und mit der steigenden Flanke übernommen wird.



Sdcard2.z80 Quellcode
Sdcard2.com ausführbare Datei

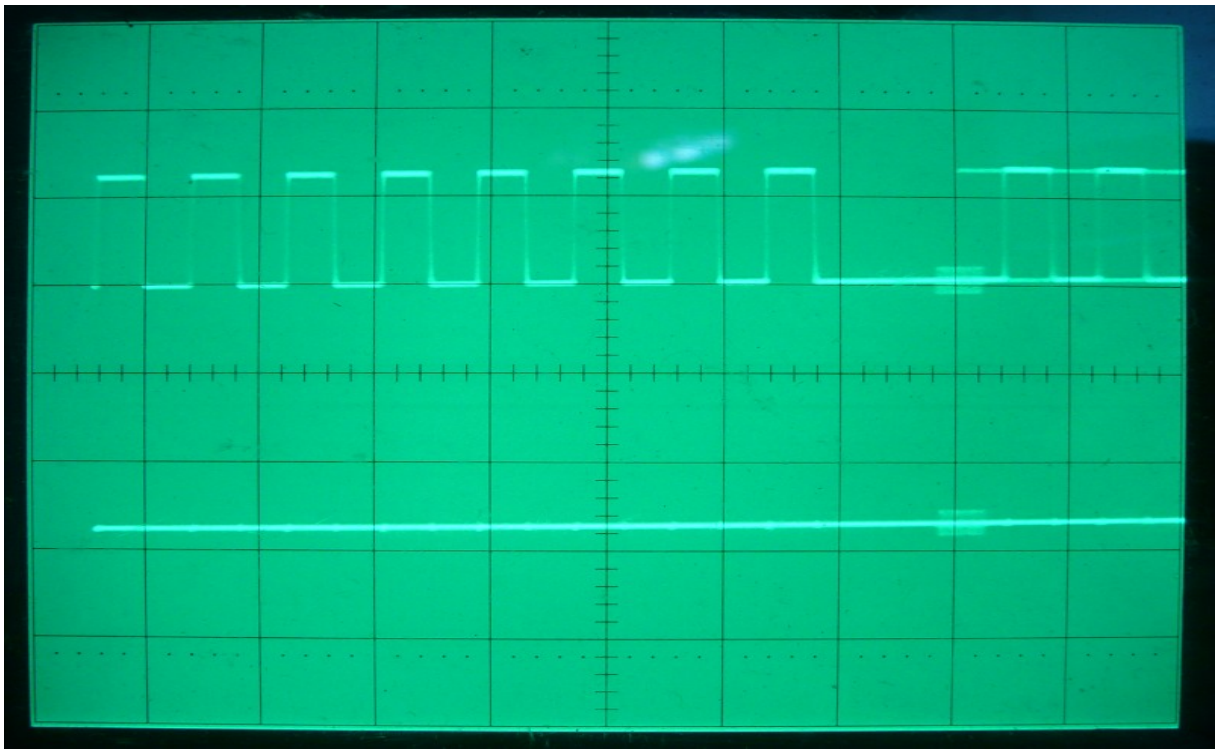
SD-Card Takterzeugung durch /RE0 bei reduzierter Geschwindigkeit.

Das wiederholte Ansprechen des Datenports erzeugt ein periodisches Signal am Adresskomparator (IC 3 Pin 19) und von /RE0 (IC 15B Pin 12). Das Löschen von Steuerport Bit 6 löst noch keinen Transfer aus, d.h. der erste Transfer muss noch extra gestartet werden.

Steuerport Bit 7 (IC8 Pin 19) ist high, so dass der geteilte Takt am Zählereingang (IC17 Pin 1) ankommt. Die übrigen Bits des Steuerports sind high, so dass Spannung an der Karte anliegt, aber keine Karte

selektiert wird. Immer, wenn vom Datenport (IC 6) gelesen wird, sollte die steigende Flanke von /RE0 das Flipflop takten (IC 19A Pin 3), so dass es high wird und den Zähler (IC 17) für 16 Taktimpulse laufen lässt. Die halbierte Taktrate (312 kHz) wird ausgegeben, so dass 8 Takte an Pin 5 der SD-Card Sockel zu sehen sind, die jeweils doppelt so lang wie der Eingangstakt sind. Durch die Halbierung entfällt der erste Taktimpuls, der in der Regel angeschnitten ist, weil die Busschnittstelle durch einen anderen Takt synchronisiert wird.
Bei 625 kHz Oszillatortakt dauern 16 Takte 25,6 us oder 80 T States des Z80. Ein NOP dauert 4 T States = 1 us @ 4 MHz. 25 NOPs decken also die Zeit einer Übertragung ab. Das jp sorgt danach für eine Pause.

Das Oszilloskop sollte mit der steigenden Flanke von SCLK (SD Card Pin 5) getriggert werden. Mit gut 2 us/div sieht man etwa einen Zyklus. Da die steigende Flanke nicht synchron zum Takt ist, verschiebt sich der Beginn des nächsten Zyklus auf dem Oszilloskop, so dass sich ein verwaschenes Bild ergibt. Ein Takt sollte etwa 3,2 us lang sein.
Das Datensignal zur Karte (SD Card Pin 2) zeigt keine Aktivitaet, weil das Schreiben über /RE0 0ffh überträgt.



sdcard3.z80 Quellcode
sdcard3.com ausführbare Datei

SD-Card Takterzeugung durch /WE0 bei voller Geschwindigkeit.

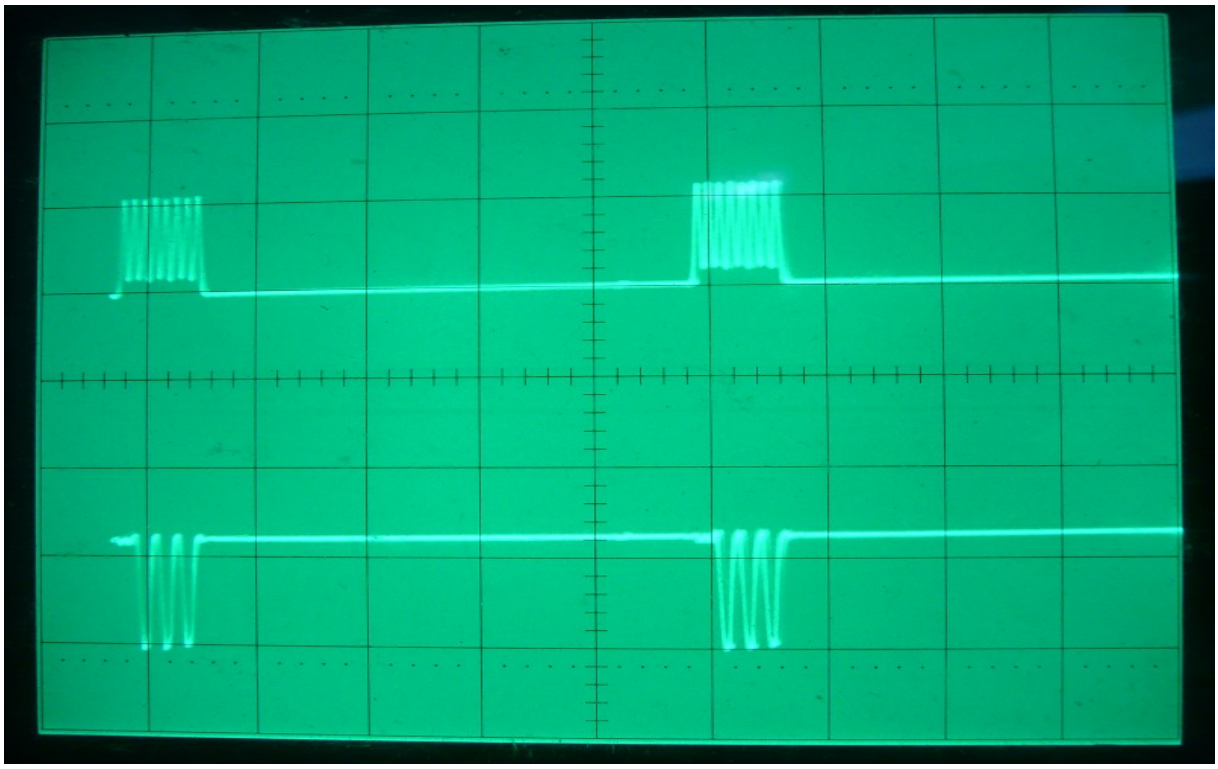
Das wiederholte Ansprechen des Datenports erzeugt ein periodisches Signal am Adresskomparator (IC 3 Pin 19) und von /WE0 (IC 15A Pin 4).

Steuerport Bit 7 (IC8 Pin 19) ist low, so dass der Takt mit voller Geschwindigkeit am Zählereingang (IC17 Pin 1) ankommt. Die übrigen

Bits des Steuerports sind high, so dass Spannung an der Karte anliegt, aber keine Karte selektiert wird. Immer, wenn in den Datenport (IC 6) geschrieben wird, sollte die steigende Flanke von /WE0 das Flipflop takten (IC 19A Pin 3), so dass es high wird und den Zähler (IC 17) für 16 Taktimpulse laufen lässt. Die halbierte Taktrate wird ausgegeben, so dass 8 Takte an Pin 5 der SD-Card Sockel zu sehen sind, die jeweils doppelt so lang wie der Eingangstakt sind.

Bei 20 MHz Oszillatortakt dauern 16 Takte 0,8 us oder 5 T States des Z80. Die Übertragung läuft also während des jp.

Das Oszilloskop sollte mit der steigenden Flanke von SCLK (SD Card Pin 5) getriggert werden. Mit 1 us/div sieht man zwei Transfers, die jeweils knapp 1 us dauern. Hier gibt das typische 10 MHz Oszilloskop erkennbar auf.



sdcard4.z80 Quellcode
sdcard4.com ausführbare Datei

SD-Card Takterzeugung durch /RE0 bei voller Geschwindigkeit.

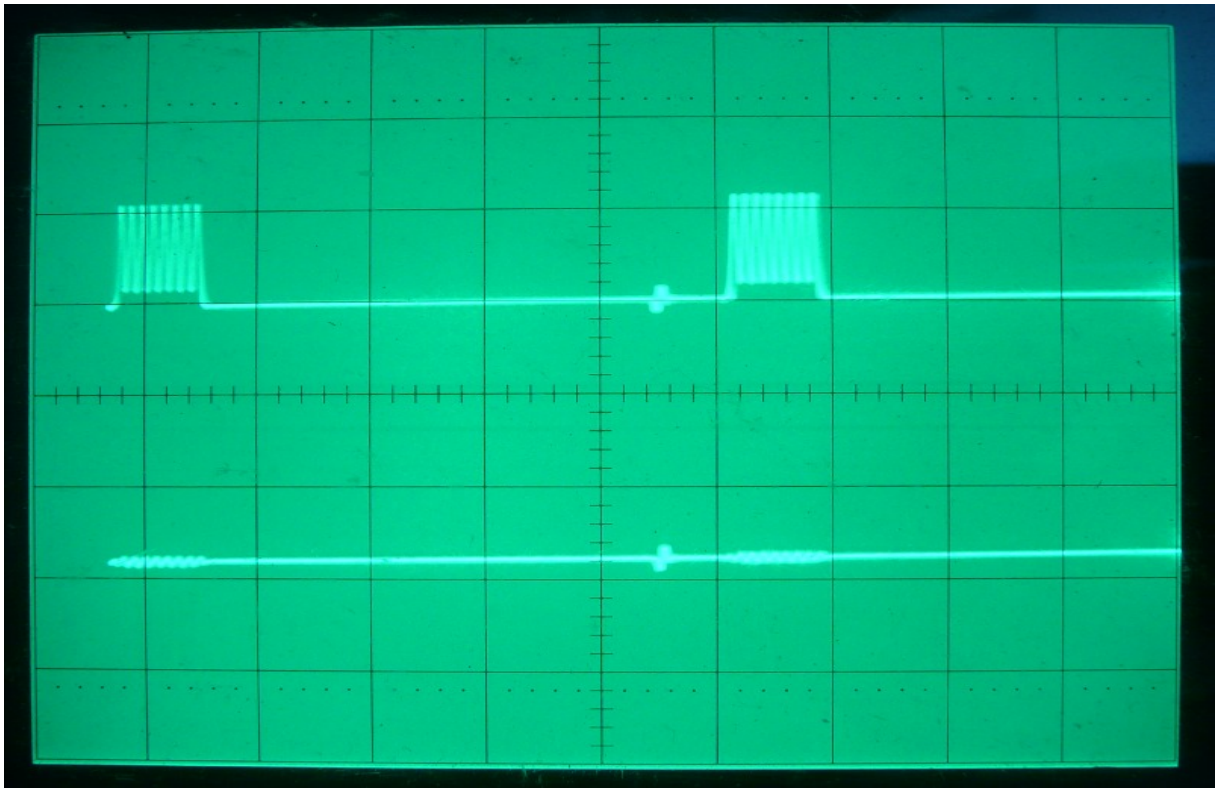
Das wiederholte Ansprechen des Datenports erzeugt ein periodisches Signal am Adresskomparator (IC 3 Pin 19) und von /RE0 (IC 15B Pin 12).

Steuerport Bit 7 (IC8 Pin 19) ist low, so dass der Takt mit voller Geschwindigkeit am Zählereingang (IC17 Pin 1) ankommt. Die übrigen Bits des Steuerports sind high, so dass Spannung an der Karte anliegt, aber keine Karte selektiert wird. Immer, wenn in den Datenport (IC 6) geschrieben wird, sollte die steigende Flanke von /WE0 das Flipflop

takten (IC 19A Pin 3), so dass es high wird und den Zaehler (IC 17) für 16 Taktimpulse laufen lässt. Die halbierte Taktrate wird ausgegeben, so dass 8 Takte an Pin 5 der SD-Card Sockel zu sehen sind, die jeweils doppelt so lang wie der Eingangstakt sind.

Bei 20 MHz Oszillatortakt dauern 16 Takte 0,8 us oder 5 T States des Z80. Die Übertragung läuft also während des jp.

Das Oszilloskop sollte mit der steigenden Flanke von SCLK (SD Card Pin 5) getriggert werden. Mit 1 us/div sieht man zwei Transfers, die jeweils knapp 1 us dauern. Hier gibt das typische 10 MHz Oszilloskop erkennbar auf.



sdcard5.z80 Quellcode
sdcard5.com ausführbare Datei

SD-Card Datenübertragung mit Fehlerprüfung.

Wenn die Takterzeugung samt Zähler funktioniert, können MOSI (IC 13B Pin 4) und MISO (IC 11A Pin 2) verbunden werden, so dass geschriebene Daten wieder gelesen werden können.

Das Programm gibt nur fehlerhafte Daten aus: Erst das empfangene Byte, dann das gesendete Byte und dann ein Leerzeichen als Trenner. So kann man es eine Weile laufen lassen: Gibt es nichts aus, ist alles in Ordnung.

Steuerport Bit 7: 1 = Takt <= 400 kHz
Steuerport Bit 6: 1 = Das Lesen der Daten loest keinen neuen Transfer aus

Steuerport Bit 3: 1 = 3,3V Versorgung fuer Karte und Levelshifter
Steuerport Bit 1: 1 = /CS1 inaktiv
Steuerport Bit 0: 1 = /CS0 inaktiv

SPI geht davon aus, dass die Daten schon während der low Phase des Takts anliegen, um mit der steigenden Flanke übernommen zu werden. Das Propagation Delay der Levelshifter wird ausgeglichen, indem der Empfangstakt ebenfalls damit verzögert wird (IC 11B).

sdcard6.z80 Quellcode
sdcard6.com ausführbare Datei

SD-Card Datenübertragung mit Fehlerprüfung.

Wenn die Takterzeugung samt Zähler funktioniert, können MOSI (IC 13B Pin 4) und MISO (IC 11A Pin 2) verbunden werden, so dass geschriebene Daten wieder gelesen werden können.

Das Programm gibt nur fehlerhafte Daten aus: Erst das empfangene Byte, dann das gesendete Byte und dann ein Leerzeichen als Trenner. So kann man es eine Weile laufen lassen: Gibt es nichts aus, ist alles in Ordnung.

Steuerport Bit 7: 0 = Takt 10 MHz
Steuerport Bit 6: 1 = Das Lesen der Daten löst keinen neuen Transfer aus
Steuerport Bit 3: 1 = 3,3V Versorgung für Karte und Levelshifter
Steuerport Bit 1: 1 = /CS1 inaktiv
Steuerport Bit 0: 1 = /CS0 inaktiv

SPI geht davon aus, dass die Daten schon während der low Phase des Takts anliegen, um mit der steigenden Flanke übernommen zu werden. Das Propagation Delay der Levelshifter wird ausgeglichen, indem der Empfangstakt ebenfalls damit verzögert wird (IC 11B).

sdcard7.z80 Quellcode
sdcard7.com ausführbare Datei

SD Card Initialisierung und Lesetest.
Das Programm ist eine Kopie der Routinen aus FLOMON, um den Test zu vereinfachen. Es gibt im Idealfall 5 Sternchen aus: Einmal für die erfolgreiche Initialisierung der Karte, die etwa eine Sekunde dauert, und viermal beim erfolgreichen Lesen des Bootsektors, was sehr schnell geht.

Stückliste

Part	Value	Device	Package	Library	Sheet
C1	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C2	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C3	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C4	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C5	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C6	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C7	47µ	C7.5/5	C7.5B5	capacitor-wima	1
C8	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C9	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C10	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C11	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C12	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C13	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C14	47µ	C7.5/5	C7.5B5	capacitor-wima	1
C15	10µ	C5/2.5	C5B2.5	capacitor-wima	1
C16	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C17	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C18	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C19	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C20	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C21	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
C24	0,1µ	C5/2.5	C5B2.5	capacitor-wima	1
IC1	74HCT32N	74HCT32N	DIL14	74xx-eu	1
IC2	74HCT245N	74HCT245N	DIL20	74xx-eu	1
IC3	74HCT688N	74HCT688N	DIL20	74xx-eu	1
IC5	74HCT595N	74HCT595N	DIL16	74xx-eu	1
IC6	74HCT165N	74HCT165N	DIL16	74xx-eu	1
IC8	74HCT374N	74HCT374N	DIL20	74xx-eu	1
IC9	74HCT00N	74HC00N	DIL14	74xx-eu	1
IC10	74HCT08N	74HCT08N	DIL14	74xx-eu	1
IC11	74HCT125N	74HCT125N	DIL14	74xx-eu	1
IC13	74HC4050	4050N	DIL16	40xx	1
IC15	74HCT139N	74HCT139N	DIL16	74xx-eu	1
IC16	74HCT393N	74HCT393N	DIL14	74xx-eu	1
IC17	74HCT393N	74HCT393N	DIL14	74xx-eu	1
IC19	74HCT74N	74HCT74N	DIL14	74xx-eu	1
JP1		PINHD-1X30	1X30	pinhead	1
JP2		JP4Q	JP4Q	jumper	1
JP3		JP7Q	JP7Q	jumper	1
Q1		MCP1826DDPAK	D2PAK	ic-package_HK	1
QG1	20MHz	QG5860	DIL14S	crystal	1
R1	47k	R-EU_0207/12	0207/12	resistor	1

R2	10k	R-EU_0207/10	0207/10	resistor	1
R3	47k	R-EU_0207/12	0207/12	resistor	1
R4	47k	R-EU_0207/12	0207/12	resistor	1
R5	10k	R-EU_0207/10	0207/10	resistor	1
R6	10k	R-EU_0207/10	0207/10	resistor	1
R7	47k	R-EU_0207/12	0207/12	resistor	1
R8	47k	R-EU_0207/12	0207/12	resistor	1
R9	47k	R-EU_0207/12	0207/12	resistor	1
RN1	10K	G08R	SIL9	resistor-sil	1
RN2	10k X 8/9	G08R	SIL9	resistor-sil	1
SV1		FE08-1	FE08	con-lsta	1
SV2		FE08-1	FE08	con-lsta	1
X1	FPS009-30003	FPS009-30003	FPS009-3003	con-yamaichi	1
X3	FPS009-30003	FPS009-30003	FPS009-3003	con-yamaichi	1

Erstellen einer SD Karte für den NKC

Dateien: mbrsdc.sys und sdc.sys

Device feststellen ist bei den jeweiligen Linuxsystemen unterschiedlich
Beim Raspberry funktioniert:
root@test:~# ls /dev/ | grep sd
sda
sda1

sda ist das Device sda1 ist die erste Partion

fdisk aufrufen

fdisk /dev/sda

d aktuelle Partion löschen
n neue primäre Partion erstellen
t Partions Typ CPM 52
a Partion aktivieren
w Partion schreiben
danach den Bootsektor und das CPM System schreiben
cat mbrsdc.sys >/dev/sda
cat sdc.sys >/dev/sda1

cpmtools (apt-get install cpmtools)
unter /etc/cpmtools/diskdefs folgende Einträge ergänzen:

```
diskdef sdcard
  seclen 512
  tracks 256
  sectrk 64
  blocksize 8192
  maxdir 256
  skew 0
  boottrk 1
  os 2.2
end
```

cpmls - list sorted directory with output similar to ls, DIR, P2DOS DIR and CP/M3 DIR[FULL]
cpmcp - copy files from and to CP/M file systems
cpmrm - erase files from CP/M file systems
cpmchmod - change file permissions
cpmchattr - change file attributes
mkfs.cpm - make a CP/M file system
fsck.cpm - check and repair a CP/M file system (only simple errors can be repaired so far). Some images of broken file systems are provided for testing.
fsed.cpm - view CP/M file system
manual pages for everything including the CP/M file system format

Beispiele:

mkfs.cpm -f sdcard /dev/sda1 Dateisystem erstellen
cpmls -f sdcard /dev/sda1 Direktory anzeigen
cpmcp -f sdcard /dev/sda1 file.xxx 0: Kopieren von files
cpmrm -f sdcard /dev/sda1 0:file.xxx Löschen von files

0:bezeichnet den USER, ein Überschreiben eines Files funktioniert nicht… vorher vorhandenes File löschen

fsed.cpm -f sdcard /dev/sda1 Speicherabbild ansehen