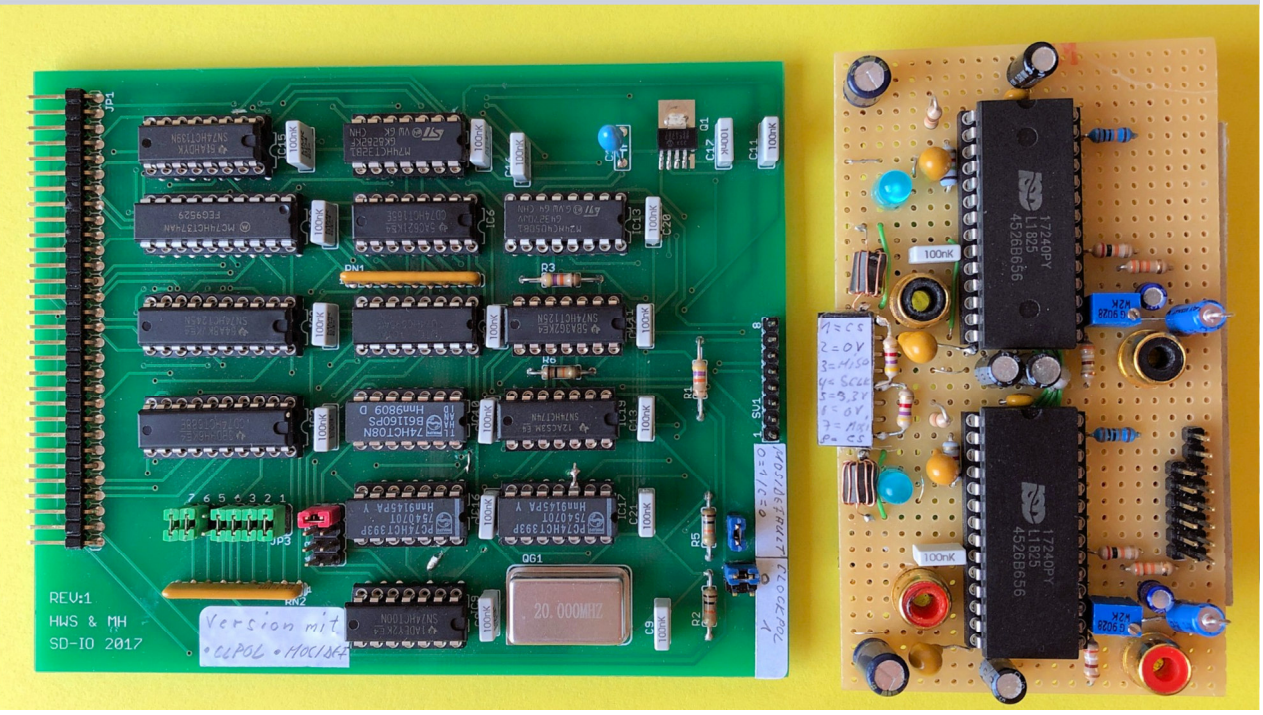
Spezifikation

SDIO als SPI-Controller
Am Beispiel des Soundrecorders ISD17240



Version 1.0

Idee:

Sascha Neuschl
Pirolweg 21
48167 Münster
Email: scn69@gmx.de

Dokumentenhistorie

Version	Autor(en)	Änderung	Datum
1.0	Neuschl, Sascha	Erste Version	17.08.2020

Inhaltsverzeichnis

1 Vorwort

1.1 Idee

Ich hatte eine SDIO-Karte, die ich im Standard aber nicht mit SD-Cards für mein 68008-System mit JADOS einsetzen konnte. Nun sollte die mal was tun – als **universeller SPI-Controller** …

Und das Testopfer wurde der Soundrecorder-Baustein ISD17240. Idee war, einen Player analog zum NKC-CD-Player zu bauen. Aber leider kann man die Daten im FIFO-RAM des Bausteins nicht auslesen und schreiben. Sonst hätte man mit einer Festplatte oder SD-Card zusammen einen netten Player haben können.

Also nur die SPI-Steuerung des Bausteins mit aufgenommenen und intern fest gespeicherten Sounds …

1.2 Ansatz

Der Baustein ISD17240 soll via SPI mit der SDIO-Karte gesteuert werden. Deshalb an dieser Stelle einmal die Auffrischung der **SPI-Basics** (Quelle: Wikipedia):

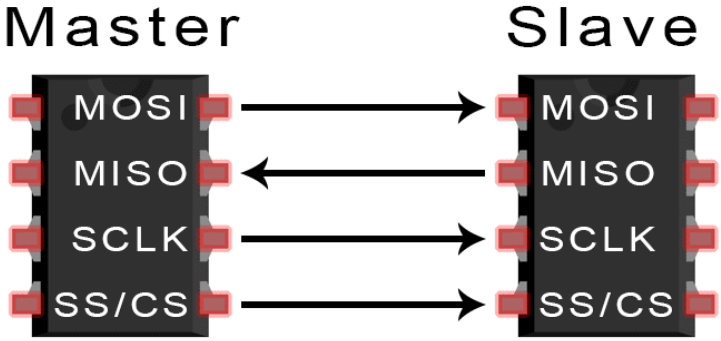
Eigenschaften

- Die drei gemeinsamen Leitungen, an denen jeder Teilnehmer angeschlossen ist, sind:

    1. **SCLK** (englisch *Serial Clock*) auch SCK, wird vom Master zur Synchronisation ausgegeben
    2. **MOSI** oder SIMO (englisch *Master Output, Slave Input*)
    3. **MISO** oder SOMI (englisch *Master Input, Slave Output*)

- Die Datenleitungen werden manchmal auch SDO (englisch *Serial Data Out*) und SDI (englisch *Serial Data In*) genannt, wobei die Benennung meistens aus der Sicht des jeweiligen Busteilnehmers erfolgt, so dass hier die Leitungen über Kreuz verbunden werden müssen. Statt SDI mit SDI und SDO mit SDO zu verbinden, müssen jeweils SDI mit SDO der Gegenstelle verbunden werden.

- Eine oder mehrere mit logisch-*0* aktive **Chip-Select-Leitungen**, welche alle vom Master gesteuert werden und von denen je eine Leitung pro Slave vorgesehen ist. Diese Leitungen werden je nach Anwendung unterschiedlich mit Bezeichnungen wie SS, CS, STE oder CE für *Slave Select*, *Chip Select*, *Slave Transmit Enable* oder *Chip Enabler* bezeichnet, oft noch in Kombination mit einer Indexnummer zur Unterscheidung.



- voll duplexfähig

- **Viele Einstellmöglichkeiten, wie**

  - mit welcher **Taktflanke** ausgegeben oder eingelesen wird
  - **Wortlänge**
  - Übertragung**: MSB oder LSB zuerst**
  - Unterschiedliche **Taktfrequenzen** bis in den MHz-Bereich sind zulässig.

  Viele Einstellungsmöglichkeiten sind unter anderem deshalb erforderlich, weil die Spezifikation für den SPI-Bus in vielen Eigenschaften nicht festgelegt ist, wodurch verschiedene, zueinander inkompatible Geräte existieren. Häufig ist beispielsweise für jeden angeschlossenen Schaltkreis eine eigene Konfiguration des steuernden Mikrocontrollers (Master des SPI-Bus) erforderlich.

Protokollablauf und Einstellmöglichkeiten

- An den Bus können so viele Teilnehmer angeschlossen werden, wie Slave-Select-Leitungen vorhanden sind, zuzüglich des genau einen Masters, der seinerseits das Clock-Signal an SCK erzeugt. Der Master legt mit der Leitung „Slave Select“ fest, mit welchem Slave er kommunizieren will. Wird sie gegen Masse gezogen, ist der jeweilige Slave aktiv und „lauscht“ an MOSI, bzw. legt er seine Daten im Takt von SCK an MISO. Es wird ein Wort vom Master zum Slave und ein anderes Wort vom Slave zum Master transportiert.
- Ein Protokoll für die Datenübertragung wurde von Motorola zwar nicht festgelegt, doch haben sich in der Praxis vier verschiedene „Modi“ durchgesetzt. Diese werden durch die Parameter "Clock Polarity" (CPOL) und "Clock Phase" (CPHA) festgelegt. Bei CPOL=0 ist der Clock Idle Low, bei CPOL=1 ist der Clock Idle High. CPHA gibt nun an, bei der wievielten Flanke die Daten übernommen werden sollen. Bei CPHA=0 werden sie bei der ersten Flanke übernommen, nachdem SS auf Low gezogen wurde, bei CPHA=1 bei der zweiten. Somit werden die Daten bei CPOL=0 und CPHA=0 mit der ersten Flanke übernommen, die nur eine steigende Flanke sein kann. Bei CPHA=1 wäre es die zweite, also eine fallende Flanke. Bei CPOL=1 ist es folglich genau andersherum, bei CPHA=0 fallende Flanke und bei CPHA=1 steigende Flanke.
- Der Slave legt bei CPHA=0 seine Daten schon beim Runterziehen von SS an MISO an, damit der Master sie beim ersten Flankenwechsel übernehmen kann. Bei CPHA=1 werden die Daten vom Slave erst beim ersten Flankenwechsel an MISO gelegt, damit sie beim zweiten Flankenwechsel vom Master übernommen werden können. Der Master hingegen legt seine Daten immer zum gleichen Zeitpunkt an, meist kurz nach der fallenden Flanke von SCK.
- Mit jeder Taktperiode wird ein Bit übertragen. Beim üblichen Bytetransfer sind also acht Taktperioden für eine vollständige Übertragung nötig. Es können auch mehrere Worte hintereinander übertragen werden, wobei in der Spezifikation nicht festgelegt ist, ob zwischen jedem Wort das SS-Signal kurz wieder auf High gezogen werden muss. Eine Übertragung ist beendet, wenn das Slave-Select-Signal endgültig auf High gesetzt wird.
- **Es werden 4 SPI-Modes unterschieden**

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Timings der 4 SPI-Modes (Quelle: introduction-to-spi-interface.pdf)

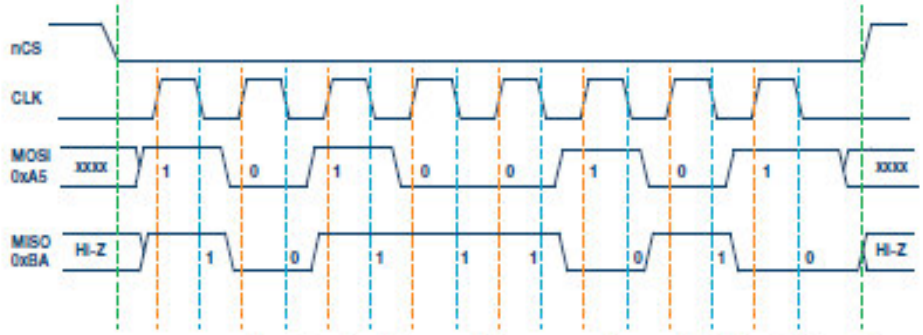


Figure 2. SPI Mode 0, CPOL = 0, CPHA = 0: CLK idle state = low, data sampled on rising edge and shifted on falling edge.

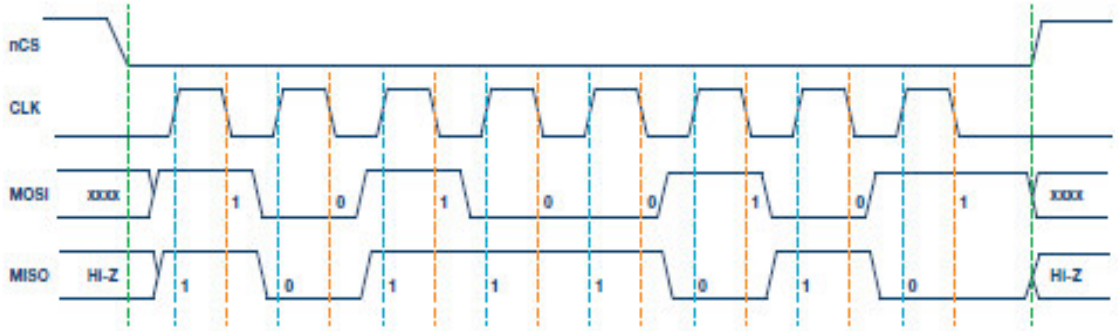


Figure 3. SPI Mode 1, CPOL = 0, CPHA = 1: CLK idle state = low, data sampled on the falling edge and shifted on the rising edge.

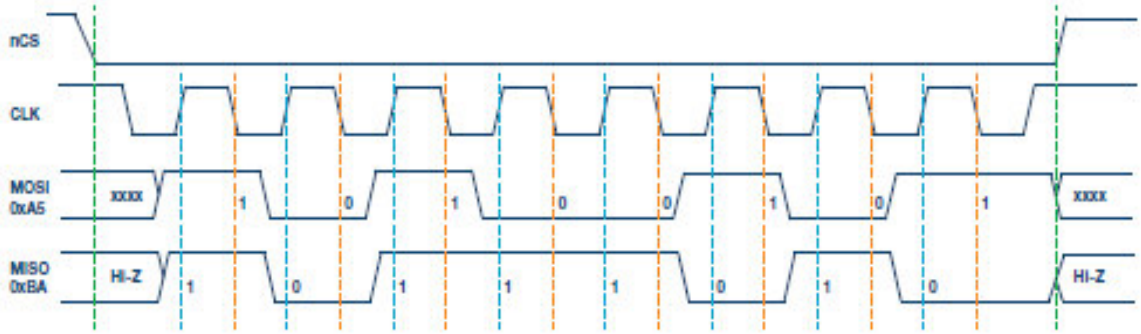


Figure 4. SPI Mode 2, CPOL = 1, CPHA = 1: CLK idle state = high, data sampled on the falling edge and shifted on the rising edge.

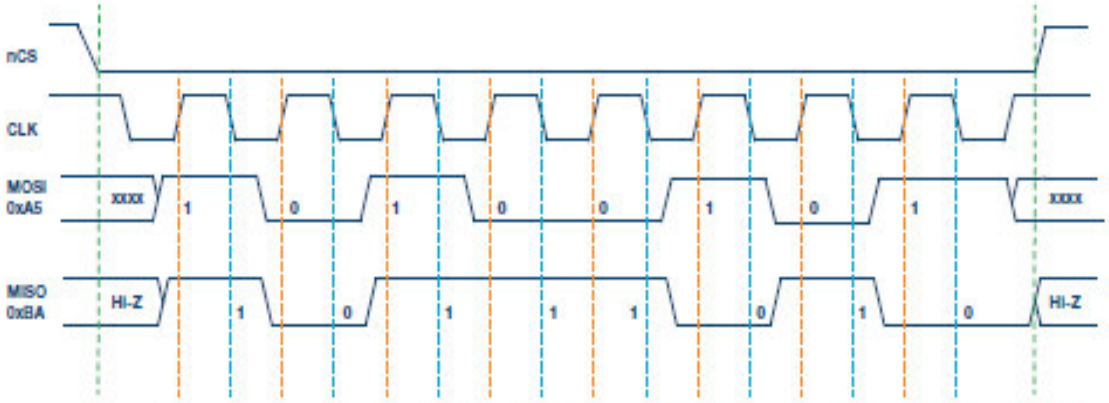


Figure 5. SPI Mode 3, CPOL = 1, CPHA = 0: CLK idle state = high, data sampled on the rising edge and shifted on the falling edge.

1.3 Aktueller Stand

Ich habe 2 Modifikationen an der SDIO-Karte vorgenommen, um sie universeller für SPI einsetzen zu können:

- Einstellung der Taktpolarität (positiv / negativ)
- Einstellung des IDLE-Pegels an MOSI (0 / 1)

Des Weiteren habe ich eine Platine mit 2 Bausteinen ISD17240 erstellt, die sich an direkt an SV1 der SDIO-Karte anschließen lässt, sowie ein Programm in 6800x-Assembler, das die SDIO zusammen mit dieser Platine betreibt.

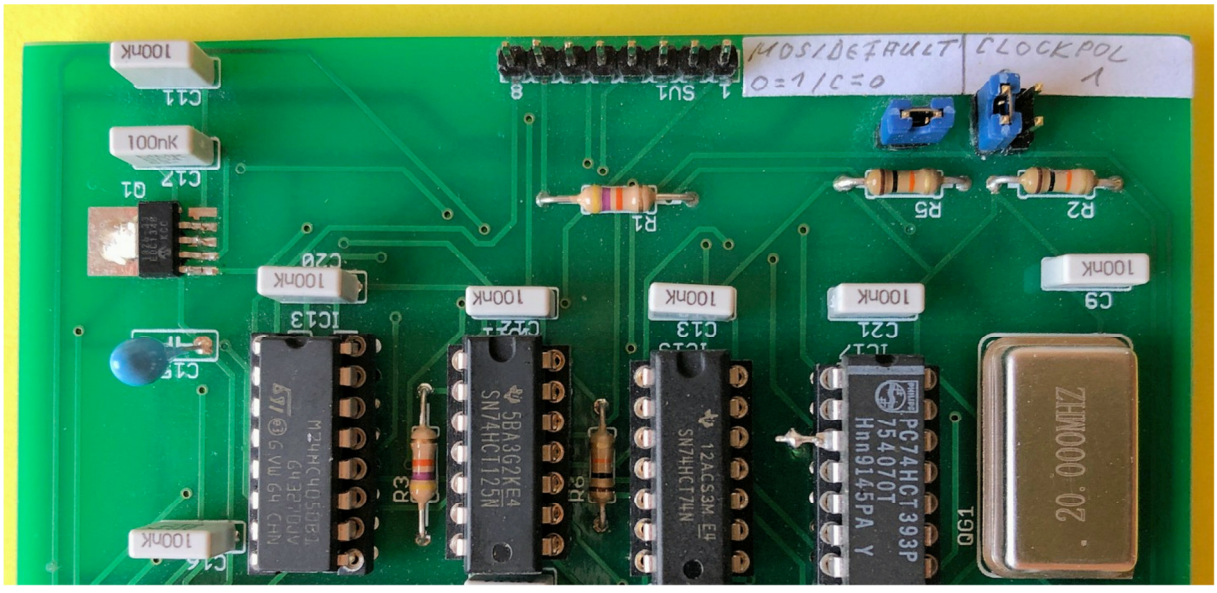
2 Beschreibung des Konzepts

Alles was man braucht, ist eine SDIO, an der 2 Modifikationen vorgenommen werden müssen, und ein **per SPI anzusteuernder Baustein – im vorliegenden Fall die Platine mit 2 ISD17240.**

Modifikationen an der SDIO-Karte

- Einstellung der Taktpolarität (positiv / negativ):
  - IC17 / 74HCT393 - Pin 3 wird herausgebogen, sodass er nicht mehr in der IC-Fassung sitzt.
  - Dieses Taktsignal wird auf einen **neuen Jumper A1** geführt:
    - In Stellung „links“ wird es - wie vorher - an IC 09 / 74HCT00 - Pins 1,2 geführt
    - In Stellung „rechts“ wird es auf ein **neues Single-Inverter-IC SN74LVC1G04** geführt, bevor es an IC 09 / 74HCT00 - Pins 1,2 geführt wird

- Einstellung des IDLE-Pegels an MOSI (0 / 1):
  - Der Pin 10 von IC 06 / 74HCT165 wird auf der Karte über den Widerstand R5 / 10 KΩ auf logisch 1 / +5 Volt gezogen.
  - Über einen **weiteren, neuen Jumper A2** kann Pin 10 von IC 06 / 74HCT165 nun auf **logisch 0 / 0 Volt** gezogen werden.



Folgender, per SPI zu steuernder Baustein wird eingesetzt

Funktion	Baustein	Beschreibung	Datenblatt
Soundrecorder	ISD17240	Soundrecorder mit FIFO-RAM	ISD1700.pdf, ISD1700_Design_Guide.pdf, Design_Considerations_for_ISD1700_Family.pdf

3 Einstellung der modifizierten SDIO-Karte für den Betrieb mit dem Baustein ISD17240

3.1 SPI-Anforderungen des ISD17240

The ISD1700 series operates via the SPI serial interface with the following protocol.
Data transfer protocol requires that the microcontroller's SPI shift registers are clocked out on the falling edge of the SCLK. The SPI protocol of the ISD1700 device is as follows:

1. A SPI transaction is initiated on the falling edge of the $\overline{SS}$ pin.
2. $\overline{SS}$ must be held Low during the entire data transfer process.
3. Data is clocked into the device through the MOSI pin on the rising edge of the SCLK signal and clocked out of the MISO pin on the falling edge of the SCLK signal, with LSB first.
4. The opcodes contain command, data and address bytes, depending upon the command type.
5. While control and address data are shifted into the MOSI pin, the status register and current row address are simultaneously shifted out of the MISO pin.
6. The SPI transaction is completed by raising the $\overline{SS}$ to High.
7. After completing an operational SPI command, an active Low interrupt is generated. It will stay Low until it is reset by the CLR_INT command.

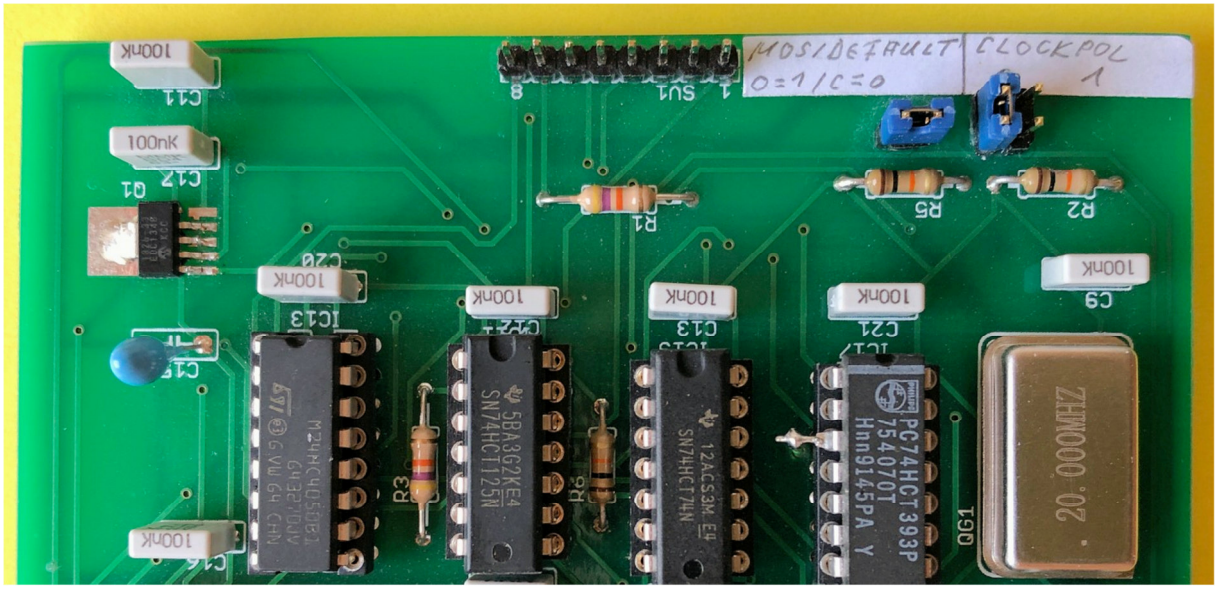
Ergebnis

- **SPI-Mode 0**
- mit der Nebenbedingung, dass **MOSI den IDLE-Wert logisch „0“** besitzen muss (siehe 2. Modifikation der SDIO-Karte)
- Außerdem muss bei **MOSI zuerst das LSB ausgegeben** werden und bei **MISO das LSB zuerst entgegen genommen** werden. Das Drehen der Bytes wird in einem Unterprogramm des Testprogramms erledigt.

3.2 Einstellung der SDIO-Karte

Die SDIO-Karte wird an den beiden neuen Jumpern folgendermaßen eingestellt:

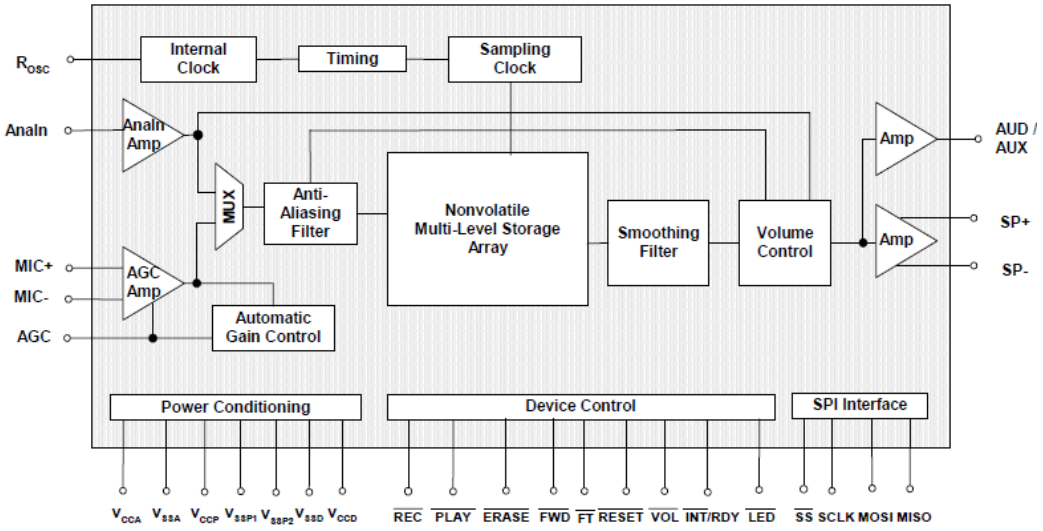
- Einstellung der Taktpolarität (positiv / negativ) – Jumper A1 auf „links“
- Einstellung des IDLE-Pegels an MOSI (0 / 1) – Jumper A2 gesteckt



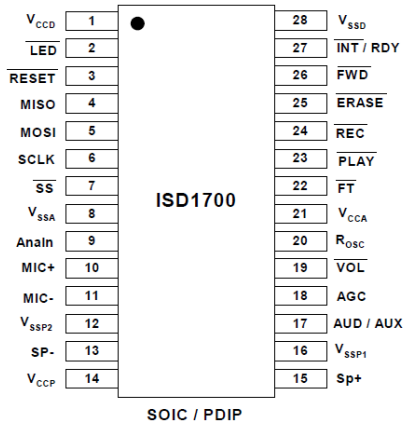
3.3 Der Baustein ISD17240

Es handelt sich bei dem Baustein um einen Soundrecorder, der mit Tasten - stand alone - oder SPI bedient werden kann (Quelle: ISD1700_Design_Guide.pdf).

Blockdiagramm



Pin-Belegung



SOIC / PDIP

PIN NAME	PDIP / SOIC	TSOP	FUNCTIONS [3]
V_{CCD}	1	22	**Digital Power Supply**: It is important to have a separate path for each power signal including V_{CCD}, V_{CCA} and V_{CCP} to minimize the noise coupling. Decoupling capacitors should be as close to the device as possible.
$\overline{LED}$	2	23	**LED**: With an LED connected, this output turns an LED on during recording and blinks LED during playback, forward and erase operations.
$\overline{RESET}$	3	24	**RESET**: When Low, the device enters into a known state and initializes all pointers to the default state. This pin has an internal pull-up resistor [1].
MISO	4	25	**Master In Slave Out:** Data is shifted out on the falling edge of SCLK. When the SPI is inactive ($\overline{SS}$ = high), it's tri-state.
MOSI	5	26	**Master Out Slave In:** Data input of the SPI interface when the device is configured as slave. Data is latched into the device on the rising edge of SCLK. This pin has an internal pull-up resistor [1].
SCLK	6	27	**Serial Clock:** Clock of the SPI interface. It is usually generated by the master device (typically microcontroller) and is used to synchronize the data transfer in and out of the device through the MOSI and MISO lines, respectively. This pin has an internal pull-up resistor [1].
$\overline{SS}$	7	28	**Slave Select:** This input, when low, selects the device as slave device and enables the SPI interface. This pin has an internal pull-up resistor [1].
V_{SSA}	8	1	**Analog Ground**: It is important to have a separate path for each ground signal including V_{SSA}, V_{SSD}, V_{SSP1} and V_{SSP2} to minimize the noise coupling.
AnaIn	9	2	**AnaIn:** Auxiliary analog input to the device for recording or feed-through. An AC-coupling capacitor (typical 0.1uF) is necessary and the amplitude of the input signal must not exceed 1.0 Vpp. Depending upon the D3 of APC register, AnaIn signal can be directly recorded into the memory, mixed with the Mic signal then recorded into the memory or buffered to the speaker and AUD/AUX outputs via feed-through path.
MIC+	10	3	**MIC+:** Non-inverting input of the differential microphone signal. The input signal should be AC-coupled to this pin via a series capacitor. The capacitor value, together with an internal 10 KΩ resistance on this pin, determines the low-frequency cutoff for the pass band filter. The Mic analog path is also controlled by D4 of APC register.
MIC-	11	4	**MIC-**: Inverting input of the differential microphone signal. The input signal should be AC-coupled to the MIC+ pin. It provides input noise-cancellation, or common-mode rejection, when the microphone is connected differentially to the device. The Mic analog path is also controlled by D4 of APC register.
V_{SSP2}	12	5	**Ground for Negative PWM Speaker Driver**: It is important to have a separate path for each ground signal including V_{SSA}, V_{SSD}, V_{SSP1} and V_{SSP2} to minimize the noise coupling.

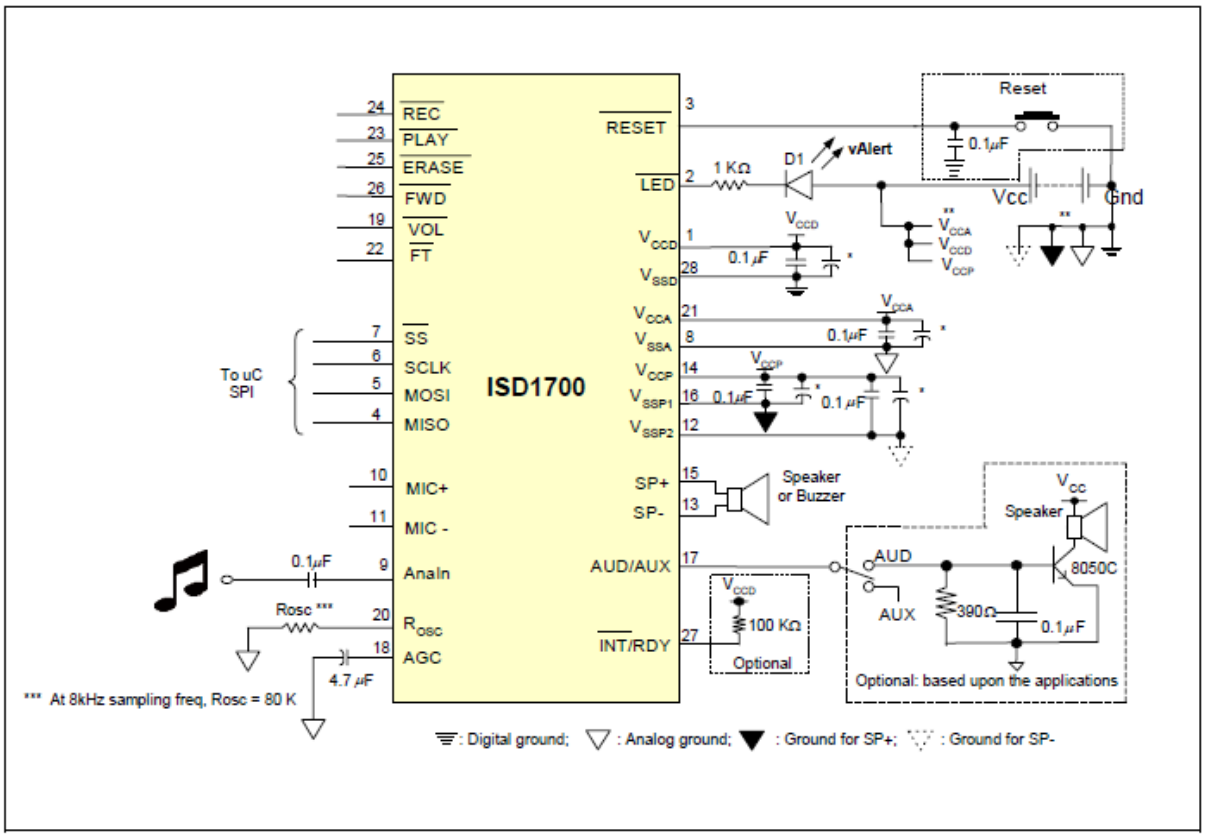
PIN NAME	PDIP / SOIC	TSOP	FUNCTIONS [3]
SP-	13	6	**SP-**: The negative Class D PWM provides a differential output with SP+ pin to directly drive an 8 Ω speaker or typical buzzer. During power down or not used, this pin is tri-stated. This output can be controlled by D8 of APC register. The factory default is set at on state.
V_{CCP}	14	7	**Power Supply for PWM Speaker Driver**: It is important to have a separate path for each power signal including V_{CCD}, V_{CCA} and V_{CCP} to minimize the noise coupling. Decoupling capacitors to V_{SSP1} and V_{SSP2} should be as close to the device as possible. The V_{CCP} supply and V_{SSP} ground pins have large transient currents and need low impedance returns to the system supply and ground, respectively.
SP+	15	8	**SP+:** The positive Class D PWM provides a differential output with the SP- pin to directly drive an 8 Ω speaker or typical buzzer. During power down or not used, this pin is tri-stated. This output can be controlled by D8 of APC register. The factory default is set at on state.
V_{SSP1}	16	9	**Ground for Positive PWM Speaker Driver**: It is important to have a separate path for each ground signal including V_{SSA}, V_{SSD}, V_{SSP1} and V_{SSP2} to minimize the noise coupling.
AUD / AUX	17	10	**Auxiliary Output**: Depending upon the D7 of APC register, this output is either an AUD or AUX output. AUD is a single-ended current output, whereas AUX is a single-ended voltage output. They can be used to drive an external amplifier. The factory default is set to AUD. This output can be powered down by D9 of APC register. The factory default is set to On state. For AUD output, there is a ramp up at beginning and ramp down at the end to reduce the pop.
AGC	18	11	**Automatic Gain Control (AGC)**: The AGC adjusts the gain of the preamplifier dynamically to compensate for the wide range of microphone input levels. The AGC allows the full range of signals to be recorded with minimal distortion. The AGC is designed to operate with a nominal capacitor of 4.7 µF connected to this pin. Connecting this pin to ground (V_{SSA}) provides maximum gain to the preamplifier circuitry. Conversely, connecting this pin to the power supply (V_{CCA}) provides minimum gain to the preamplifier circuitry.
$\overline{VOL}$	19	12	**Volume:** This control has 8 levels of volume adjustment. Each Low going pulse decreases the volume by one level. Repeated pulses decrease volume level from current setting to minimum then increase back to maximum, and continue this pattern. During power-up or $\overline{RESET}$, a default setting is loaded from non-volatile configuration. The factory default is set to maximum. This output can also be controlled by <D2:D0> of APC register. This pin has an internal pull-up device [1] and an internal debounce (T_{Deb}) [2] for start and end allowing the use of a push button switch.
R_{OSC}	20	13	**Oscillator Resistor**: A resistor connected from R_{OSC} pin to ground determines the sample frequency of the device, which sets the duration. Please refer to the Duration Section for details.

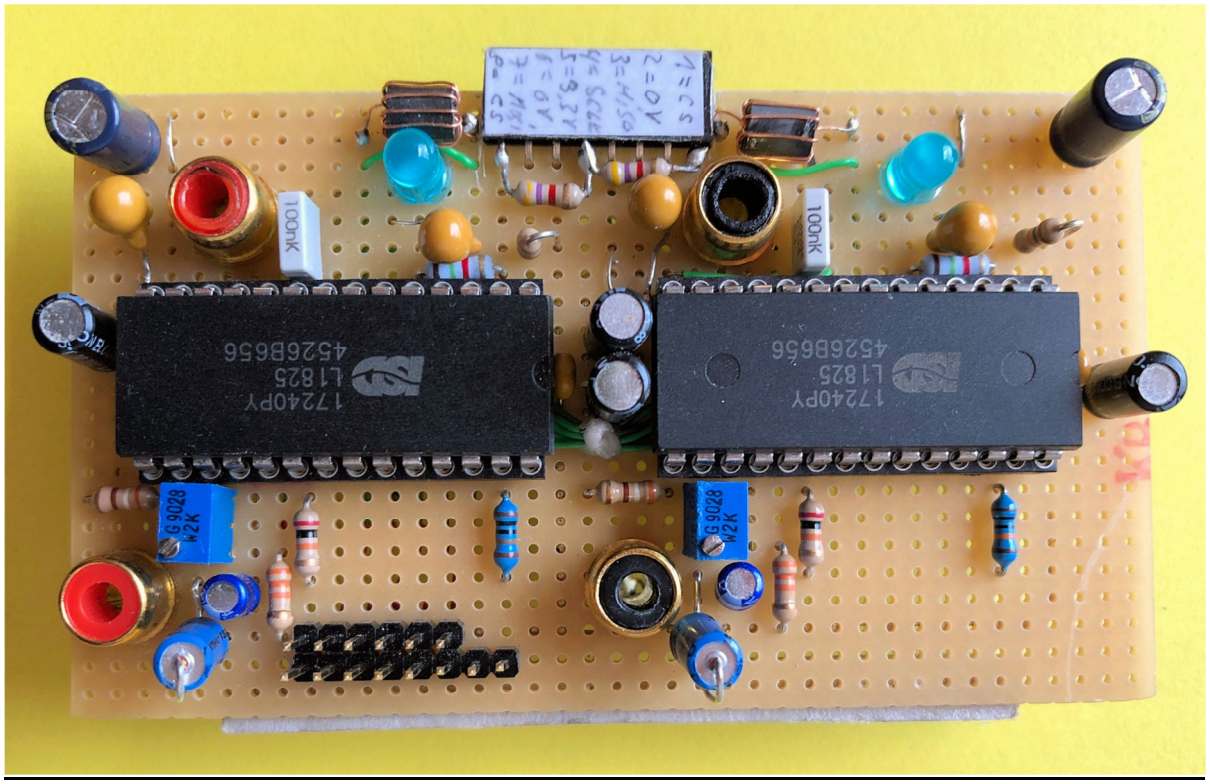
PIN NAME	PDIP / SOIC	TSOP	FUNCTIONS [3]
V_{CCA}	21	14	**Analog Power Supply**. It is important to have a separate path for each power signal including V_{CCD}, V_{CCA} and V_{CCP} to minimize the noise coupling. Decoupling capacitors to V_{SSA} should be as close to the device as possible.
$\overline{FT}$	22	15	**Feed-through:** In Standalone mode, when FT is engaged low, the AnaIn feed-through path is activated. As a result, the AnaIn signal is transmitted directly from AnaIn to both Speaker and AUD/AUX outputs with Volume Control. However, SPI overrides this input, while in SPI mode, and feed-through path is controlled by a D6 of APC register. This pin has an internal pull-up device [1] and an internal debounce (T_{Deb}) [2] for start and end allowing the use of a push button switch.
$\overline{PLAY}$	23	16	**Playback:** Pulsing $\overline{PLAY}$ to Low once initiates a playback operation. Playback stops automatically when it reaches the end of the message. Pulsing it to Low again during playback stops the operation. Holding $\overline{PLAY}$ Low constantly functions as a sequential playback operation loop. This looping continues until $\overline{PLAY}$ returns to High. This pin has an internal pull-up device [1] and an internal debounce (T_{Deb}) [2] for start and end allowing the use of a push button switch.
$\overline{REC}$	24	17	**Record:** The device starts recording whenever $\overline{REC}$ switches from High to Low and stays at Low. Recording stops when the signal returns to High. This pin has an internal pull-up device [1] and an internal debounce (T_{Deb}) [2] for start allowing the use of a push button switch.
$\overline{ERASE}$	25	18	**Erase:** When active, it starts an erase operation. Erase operation will take place only when the playback pointer is positioned at either the first or last message. Pulsing this pin to Low enables erase operation and deletes the current message. Holding this pin Low for more than 3 sec. initiates a global erase operation, and will delete all the messages. This pin has an internal pull-up device [1] and an internal debounce (T_{Deb}) [2] for start and end allowing the use of a push button switch.
$\overline{FWD}$	26	19	**Forward:** When triggered, it advances to the next message from the current location, when the device is in power down status. During playback cycle, pulsing this pin Low stops the current playback operation and advances to the next message, and then re-starts the playback operation of the new message. This pin has an internal pull-up device [1] and an internal debounce (T_{Deb}) [2] for start and end allowing the use of a push button switch.
RDY/$\overline{INT}$	27	20	An open drain output. ***Ready (Standalone mode):*** This pin stays Low during record, play, erase and forward operations and stays High in power down state ***Interrupt (SPI mode):*** After completing the SPI command, an active low interrupt is generated. Once the interrupt is cleared, it returns to High.

PIN NAME	PDIP / SOIC	TSOP	FUNCTIONS [3]
V_{SSD}	28	21	**Digital Ground**: It is important to have a separate path for each ground signal including V_{SSA}, V_{SSD}, V_{SSP1} and V_{SSP2} to minimize the noise coupling.

Note: [1] 600 kΩ
[2] TDeb = Refer to AC Timing
[3] For any unused pins, left floated.

Schaltplan und Testplatine





SPI-Befehlssatz

Instructions [1]	Com-mand Byte [2]	Data Byte1	Data Byte2 or Start Address Byte1 [3]	Data Byte3 or Start Address Byte2 [3]	End Address Bytes 1/2/3 [3]	Description
PU	0x01	0x00				
STOP	0x02	0x00				Stop the current operation
RESET	0x03	0x00				Reset the device
CLR_INT	0x04	0x00				Clear interrupt and EOM bit
RD_STATUS	0x05	0x00	0x00			Returns status bits & current row counter in first 1st 2 bytes and operating status in 3rd byte
RD_PLAY_PTR	0x06	0x00	0x00	0x00		Returns status bits & current row counter in 1st 2 bytes and playback pointer in 3rd & 4th bytes
PD	0x07	0x00				Power down the device
RD_REC_PTR	0x08	0x00	0x00	0x00		Returns status bits & current row counter in 1st 2 bytes and Record pointer in 3rd & 4th bytes
DEVID	0x09	0x00	0x00			Read the device ID register.
PLAY	0x40	0x00				Play from current location without LED action until EOM or STOP command received
REC	0x41	0x00				Record from current location without LED action until end of memory or STOP command received
ERASE	0x42	0x00				Erase current message to EOM location
G_ERASE	0x43	0x00				Erase all messages (not include Sound Effects)

Instructions [1]	Com-mand Byte [2]	Data Byte1	Data Byte2 or Start Address Byte1 [3]	Data Byte3 or Start Address Byte2 [3]	End Address Bytes 1/2/3 [3]	Description
RD_APC	0x44	0x00	0x00	0x00		Returns status bits & current row counter in first 1st 2 bytes and the contents of APC register in 3rd & 4th bytes.
WR_APC1	0x45	<D7:D0>	<xxxx D11:D8>			Write the data <D11:D0> into the APC register with volume setting from $\overline{\text{VOL}}$ pin
WR_APC2	0x65	<D7:D0>	<xxxx D11:D8>			Write the data <D11:D0> into the APC register with volume setting from bits <D2:D0>
WR_NVCFG	0x46	0x00				Write the contents of APC to NVCFG
LD_NVCFG	0x47	0x00				Load contents of NVCFG to APC Register
FWD	0x48	0x00				Forward playback pointer to start address of next message. Forward will be ignored during operating, except Play
CHK_MEM	0x49	0x00				Check circular memory
EXTCLK	0x4A	0x00				Enable/disable external clock mode
SET_PLAY	0x80	0x00	<S7:S0>	<xxxx S10:S8>	<xxxx xxxx xxxxxE10:E0>	Play from start address <S10:S0> to end address <E10:E0> or stop at EOM, depending on the D11 of APC
SET_REC	0x81	0x00	<S7:S0>	<xxxx S10:S8>	<xxxx xxxx xxxxxE10:E0>	Record from start address <S10-S0> to end address <E10:E0>
SET_ERASE	0x82	0x00	<S7:S0>	<xxxx S10:S8>	<xxxx xxxx xxxxxE10:E0>	Erase from start address <S10:S0> to end address <E10:E0>

Note: [1] Set initial SPI condition as listed in Section 10.2 before any SPI command is sent.

[2] Bit C4 (LED) must be set to 1 if LED indication is required. During the active state of LED output, no new command will be accepted.

[3] For "xxx…", recommend to use "000…"

3.4 Testprogramm

Das Testprogramm **„NKCDS10.ASM“** besitzt folgende Funktionen:

- **Einstellung der SDIO-Karte:**
  - Datenregister : `spidata equ $FFFFFF20*CPU`
  - Statusregister: `spistat equ $FFFFFF21*CPU`
    - Einstellung: `spiinit equ %11001011`

      `400 kHz, ReadOnly, Power on, kein Geraet ausgewaehlt`
- **Globale SPI-Einstellung:**
  - Variable „spidev“ legt das SPI-Gerät fest
  - Variable „twistdat“ legt fest, ob ein zu übertragener (MOSI) und zu empfangender (MISO) Datenpuffer byteweise gedreht werden soll (MSB zu LSB)
- **Unterprogramme:**
  - „Befehl“:
    - Ausführung eines in A0 übergebenen Befehls
    - Statusprüfung auf ERROR
  - „Status“:
    - Modus „init“ :
      - Wartet eine Zeit, Löscht INT und prüft auf RDY für nächsten Befehl
    - Modus „standard“:
      - Wartet auf INT, Löscht INT und prüft auf RDY für nächsten Befehl
- **Hauptprogramm:**
  - Initialisierung SDIO-Karte
  - SPI-Gerät auswählen
  - Festlegung, dass Datenbytes gedreht werden
  - Power UP – Schalten in SPI-Modus
  - Audiopfad-Konfiguration
  - Play eines vorher per Taste aufgenommenen Sounds
  - Power Down

```
;************************************************************************
;*                              NKC-Digisound                           *
;************************************************************************
;* Steuerung des Soundrecorder-ICs ISD17240 via SPI - Bus               *
;* Als SPI - Controller wird die NKC-SDIO-Karte verwendet.              *
;* Basisadresse ist $FFFFFF20                                           *
;************************************************************************
;* Testprogramm   Version 1.0   15.08.2020  by sEn                      *
;*                                                                      *
;* Funktionen:                                                          *
;*      - Initialisierung SDIO-SPI-Controller                           *
;*      - Initialisierung Baustein ISD17240                             *
;*      - Ausführung von Befehlen an Baustein ISD17240                  *
;*      - Statusueberwachung Baustein ISD17240                          *
;************************************************************************


 start:
 bra digsound
 rts


;*** Konstanten SDIO-SPI-Controller

 cpu            equ 1                   ;68008

 spidata        equ $FFFFFF20*CPU       ;SPI-Datenregister
 spistat        equ $FFFFFF21*CPU       ;SPI-Statusregister

 spiinit        equ %11001011           ;400 kHz, ReadOnly, Power on, kein
                                        ;Geraet ausgewaehlt
 spidev0        equ %11001010           ;Geraet 0 ausgewaehlt
 spidev1        equ %11001001           ;Geraet 1 ausgewaehlt

 ;*** Konstanten SPI-Commands fuer ISD 17240

 pu:                    dc.b    $01,$00,$FF
 stop:                  dc.b    $12,$00,$FF
 reset:                 dc.b    $03,$00,$FF
 clrint:                dc.b    $04,$00,$FF
 rdstatus:              dc.b    $15,$00,$00,$FF
 rdplaypt:              dc.b    $06,$00,$00,$00,$FF
```

```
 pd:                    dc.b    $07,$00,$FF
 rdrecpt:               dc.b    $08,$00,$00,$00,$FF
 devid:                 dc.b    $09,$00,$00,$FF
 play:                  dc.b    $50,$00,$FF
 rec:                   dc.b    $51,$00,$FF
 erase:                 dc.b    $52,$00,$FF
 gerase:                dc.b    $53,$00,$FF
 rdapc:                 dc.b    $44,$00,$00,$00,$FF
 wrapc1:                dc.b    $45,$0B,$0D,$FF         ;eigene Konfiguration
 wmrapc2:               dc.b    $65,$0B,$0D,$FF         ;eigene Konfiguration
 wrnvcfg:               dc.b    $46,$00,$FF
 ldnvcfg:               dc.b    $47,$00,$FF
 fwd:                   dc.b    $48,$00,$FF
 chkmem:                dc.b    $49,$00,$FF
 extclk:                dc.b    $4A,$00,$FF
 setplay:               dc.b    $80,$00,$00,$00,$00,$00,$00,$FF
 setrec:                dc.b    $81,$00,$00,$00,$00,$00,$00,$FF
 seterase:              dc.b    $82,$00,$00,$00,$00,$00,$00,$FF

 ;*** Variablen

  spidev:               df.b  1, $00            ;SPI-Geraet
  twistdat:             df.b  1, $00            ;MOSI/MISO-Bytes drehen
  miso:                 df.b 10, $FF            ;MISO-Puffer
  datch:                df.b 10, $FF            ;MISO-Puffer gedreht

 ds 0


;*** Unterprogramme

;Allgemein

 twdat:                         ;Alle Bytes eines Datenpuffers werden gedreht -
                                ;MSB zu LSB. A0 ist Eingagnspuffer.
  lea datch(pc), A1             ;Zielpuffer der gedrehten Daten
  combyte:
   move.b (A0)+, d0             ;1 Byte der Originaldaten
   cmp.b #$FF, d0               ;Endekennung eines Befehls
   beq.s twdatend
   clr.l d2
   move.l #8, d3
```

```
  chbyte:                       ;Sortiert Byte aus D0 mit MSB zu LSB in D1
   subq #1, d3
   btst d3, d0                  ;Abfrage Bitwert in D0 ...
   beq.s null
   bset d2, d1                  ;Wenn 1, dann in D1 setzen
   bra.s chbytend
   null:
   bclr d2, d1                  ;Wenn 0, dann in D4 löschen
   chbytend:
   addq #1, d2
   cmp.b #8, d2
   bne.s chbyte
   move.b d1, (A1)+
   bra.s combyte
  twdatend:
  move.b d0, (A1)               ;Endkennung des Befehls
  lea datch(pc), A1             ;Puffer neu laden fuer Verwendung in Befehl
 rts

;SDIO-SPI-Controller

 sdiobusy:                      ;Prueft bysy-Status des SDIO-SPI-Controllers
  btst #0, spistat
  bne.s sdiobusy
 rts

;ISD 1742-Baustein

status:                         ;Prueft, ob Befehl fertig verarbeitet und ob
                                ;bereit fuer naechsten Befehl
                                ;D0=0 - monitor INT, CLR INT, check RDY
                                ;D0=1 - delay, CLR INT, check RDY
 cmp.b #1, d0                   ;Check auf init oder standard
 beq.s puinit
 lea rdstatus, A0               ;RDSTATUS-Befehl
 bsr befehl                     ;und ausführen
 move.b 0(A1), d0               ;1. MISO-Byte
  btst.b #4, d0                 ;INT-Bit
  beq.s status                  ;Wenn 1, dann aktuelle Operation fertig
  bra.s initclr
 puinit:
   move #1, d0                  ;Warten auf power up
```

```
  moveq #!delay, d7
  trap #1
initclr:
 lea clrint, A0                ;CLRINT-Befehl
 bsr befehl                    ;und ausführen
chkrdy:
 lea rdstatus, A0              ;RDSTATUS-Befehl
 bsr befehl                    ;und ausführen
 move.b 2(A1), d0              ;3. MISO-Byte
 btst #0, d0                   ;RDY-Bit
 beq.s chkrdy                  ;Wenn1, dann ready fuer naechsten Befehl
rts

befehl:                        ;Verarbeitet SPI-Befehle fuer ISD17240
                               ;Der Befehl steht in A0, das SPI-Geraet in D0.
                               ;D4=1, dan werden Daten gedreht MSB zu LSB.
                               ;Wenn Fehler, dann erneute Ausfuehrung
 movea A0, A1                  ;Wenn nicht gedreht wird, Befehl direkt
 cmp.b #1, twistdat            ;Bytes für MOSI drehen?
 bne.s spidevs
 bsr twdat                     ;Der Befehl aus A0 steht gedreht in A1
 spidevs:
  cmp.b #1, spidev             ;Test auf SPI Geraet 1
  bne.s dev0
  move.b #spidev1, spistat     ;SPI-Geraet 1 mit CS1=0 auswählen
  bra process
  dev0:
   move.b #spidev0, spistat    ;SPI-Geraet 1 mit CS0 = 0
 process:
 lea miso(pc), A2              ;MISO-Puffer laden
 byte:
  move.b (A1)+, d0             ;Byte aus Puffer holen und SPI-Transfer - MOSI
  cmp.b #$FF, d0               ;bei $FF ist der Befehl zuende!
  beq.s byteend
  move.b d0, spidata           ;MOSI
  bsr sdiobusy                 ;Warten bis SDIO-SPI-Controller ready ...
  move.b spidata, (A2)+        ;und MISO
  bra byte                     ;bis alle Bytes gesendet sind
 byteend:
  move.b d0, (A2)              ;MISO-Puffer-Abschluss
  move.b spiinit, spistat      ;SPI-Geraete deaktivieren, Transaktion
                               ;abschliessen
```

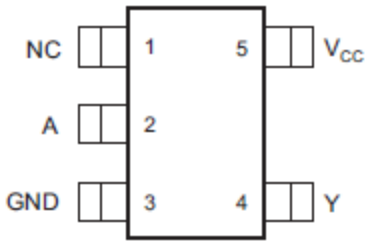
```
  lea miso(pc), A1              ;Wenn nicht gedreht wird, MISO direkt
  cmp.b #1, twistdat            ;Bytes für MISO drehen?
  bne.s chkstat
  lea miso(pc), A0
  bsr twdat                     ;Die Daten aus A0 stehen gedreht in A1
  chkstat:                      ;Statuspruefung
  move.b 0(A1), d0              ;1. MISO-Byte
  btst.b #0, d0                 ;Test auf CMDERR-Bit des Statusregisters
  bne.s befehl                  ;Wenn 1, dann Befehlt erneut senden
 rts


;*** Hauptprogramm

 digsound:
  clr.l d0
  clr.l d1
  clr.l d2
  clr.l d3


 ;* Initialisierung SPI-Controller

  move.b #spiinit, spistat              ;Einstellung SPI-Controller:
                                        ;400 KHz,onlyrd=1,power=1,CS0=CS1=1
  move.b #$00, spidata                  ;Shiftregister mit $00 laden fuer
                                        ;MOSI default = 0 - Pullup-Widerstand
                                        ;R5 auf SDIO auf 0V gejumpert!
  move #1, d0                           ;Warten auf init
  moveq #!delay, d7
  trap #1

 ;* SPI-Graet auswaehlen

  ;clr.b spidev                         ;SPI-Geraet 0
  move.b #1, spidev                     ;SPI-Geraet 1

 ;* Festlegung, ob Datenbytes gedreht werden sollen - MSB zu LSB

   ;clr.b twistdat                      ;n i c h t  drehen
   move.b #1, twistdat                  ; drehen
```

```
 ;* Initialisierung des Bausteins ISD17240

  lea pu, A0                          ;Befehl "power up" - switch auf SPI
  bsr befehl                          ;Befehl ausfuehren

  move.b #1, d0                       ;Status - Init
  bsr status                          ;Statuspruefung

;* APC-Konfiguration - Audio-Einstellungen:

  lea wmrapc2, A0                     ;APC-Schreibbefehl mit meiner Konfig.
  bsr befehl
  move #1, d0                         ;Status - Init
  bsr status                          ;Statuspruefung

;* Ausführung von Aktionen (record, play, stop, erase, setrecord, setplay ...)
  lea play, A0                        ;PLAY-Befehl
  bsr befehl                          ;Befehl ausfuehrenb

  clr.b d0                            ;Status - Standard
  bsr status

  lea pd, A0                          ;Befehl Power Down
  bsr befehl

 rts

end.
```

4 Anhang

4.1 Datenblätter TTL-Bausteine:

4.1.1 SN74LVC1G04



Pin Functions

PIN						DESCRIPTION
NAME	DBV, DCK, DRL	DSF, DRY	YZP	YZV	DPW	
NC	1	1, 5	A1, B2	–	1	No connect
A	2	2	B1	A1	2	Input
GND	3	3	C1	B1	3	Ground
Y	4	4	C2	B2	4	Output
V_{CC}	5	6	A2	A2	5	Power terminal

4.2 Verweis auf Datenblätter komplexer Bausteine und Spezifikationen / Quellennachweis

Baustein/Objekt	Funktion	Datenblatt / Spezifikation
SPI-Grundlagen	SPI-Bus	introduction-to-spi-interface.pdf
IDS17240	Soundrecorder IC	ISD1700.pdf, ISD1700_Design_Guide.pdf, Design_Considerations_for_ISD1700_Family.pdf