Sparrow

Handbuch

25. März 2024

Inhaltsverzeichnis

1 Hardware

1.1 Bestückungsvarianten

1.1.1 CPU

Die Platine wurde für die Hitachi 63C09 CPU entwickelt bei Verwendung eines Quarzes von 14,7456 MHz. Die CPU läuft dabei bei einem Viertel der Quarzfrequenz, also mit 3,6864 MHz. Diese Frequenz wurde gewählt, damit das serielle Interface in Form des DUART 68681 mit der von ihm benötigten Frequenz von 3,6864 MHz betrieben werden kann. Die langsamere Version 63B09 (2 MHz) oder die schnellste Version 68B09 (2 MHz) des ansonsten pinkompatiblen 6809 von Motorola können daher nicht verwendet werden, da dann der 68681 untertaktet werden müsste. Eine solche Untertaktung widerspricht der Spezifikation, wurde nicht getestet und wird daher nicht unterstützt.

Der 63C09 kann je nach Datenblatt mit 3,0 oder 3,5 MHz betrieben werden, wird also nach dem Datenblatt leicht übertaktet betrieben. Einige klassische, japanische Computer haben die 63C09 allerdings ab Werk mit der NTSC-Farbträgerfrequenz von 3,58 MHz betrieben und es gibt einige Berichte, dass die CPU bis etwa 5 MHz ohne negative Auswirkungen betrieben werden kann.

Bei der Beschaffung ist darauf zu achten, dass es zwei unterschiedliche Varianten HD63C09 und HD63C09E der CPU gibt, wobei eine ein „E“ in der Bezeichnung angehängt hat. Diese unterscheiden sich unter anderem in der Anschlussbelegung, so dass im Sparrow nur eine HD63C09 ohne „E“ am Ende verwendet werden kann.

1.1.2 RAM

Die Hauptplatine kann mit 32 KB, 128 KB und 512 KB SRAM-Chips bestückt werden. Abhängig von der gewählten Bestückung ist auf der Unterseite der Platine an JP3 entweder für 32 KB Chips oder für solche mit 128 oder 512 KB Chips zu konfigurieren. Wenn die Software es unterstützt, kann man also später zwischen einem 128 KB oder einem 512 KB großem Chip hin- und her wechseln, ohne dass erneute Lötarbeiten notwendig werden.

Das Auswahlfeld besteht aus drei Lötflächen, wobei mit etwas Lötzinn eine Verbindung zwischen der mittleren und einer der beiden äußeren Lötflächen hergestellt werden muss. Dabei ist darauf zu achten, dass nicht alle drei Lötflächen leitend verbunden werden.

Für den Sparrow sollte normalerweise ein 128 KB großer RAM-Chip verwendet werden. Diese Chips werden ausschließlich mit Zugriffszeiten von weit unter 200 Nanosekunden hergestellt, daher ist die tatsächliche Zugriffszeit für den Sparrow unerheblich.

Wenigstens die folgenden RAM-Chips können verwendet werden:

Hersteller	Typbezeichnung	Speicherkapazität
Hitachi	HM62256	32 KB
Hynix	HY62CT08081E	32 KB
Hitachi	HM628128A	128 KB
Samsung	KM681000ALP	128 KB
Alliance Memory Inc.	AS6C1008	128 KB
BSI	BS62LV1027P	128 KB
Hitachi	HM628512	512 KB
Samsung	K6X4800C1F	512 KB
Alliance Memory Inc.	AS6C4008	512 KB

1.1.3 ROM

Die Platine kann für EPROMs oder EEPROMs konfiguriert werden.

EPROMs haben den Vorteil, dass ihr Inhalt nicht versehentlich überschrieben werden kann und der Rechner daher immer normal booten wird. Bei Änderungen muss ihr Inhalt allerdings erst mithilfe von UV-Licht gelöscht werden, bevor er in einem besonderem Programmiergerät, einem sog. „EPROM-Brenner“, erneut beschrieben werden kann.

EEPROMs dagegen können auch eingebaut innerhalb des Computers neu beschrieben werden, wenn der Jumper JP6 gesetzt ist. Dieser Jumper ist auf der Platine mit „Write enable“ beschriftet. Zusätzlich ist ein Schreibschutz in Software möglich, so dass JP6 auch dauerhaft gesetzt bleiben kann. Wenn allerdings bei der Aktualisierung ein Fehler auftritt oder ein fehlerhaftes Update eingespielt wird, kann der Rechner nicht mehr starten und der Chip muss ebenfalls in einem Programmiergerät mit einem geeigneten Inhalt beschrieben werden.

EPROMs sind meist günstiger zu beschaffen als EEPROMs, dafür aber auch umständlicher in der Handhabung. Einen preiswerten Mittelweg stellt der ebenfalls verwendbare Chip W27E257 des Herstellers Winbond Electronics Corp. dar. Es handelt sich dabei um einen elektrisch löschbaren Chip, also eigentlich ein EEPROM. Mit den klassischen EPROMs hat er dagegen nicht nur die Anschlussbelegung gemeinsam, sondern ebenfalls kann er nur in einem Programmiergerät, nicht aber im Computer eingebaut beschrieben werden. Da er ohne Quarz-Fenster daher kommt, ist er günstiger herzustellen und man spart sich sowohl das Bedecken des Quarz-Fensters nach dem Programmieren als auch das Entfernen der Abdeckung vor dem Löschen des Chips.

Auf der Unterseite muss die Platine an zwei Lötjumpern JP1 und JP2 konfiguriert werden, je nachdem, ob die Typbezeichnung des verwendeten Speicherchips mit „27“ oder mit „28“

anfängt – wenigstens gilt diese Regel für die empfohlenen 32 KB großen Typen.

Die Auswahlfelder bestehen jeweils aus drei Lötflächen, wobei mit etwas Lötzinn eine Verbindung zwischen der mittleren und einer der beiden äußeren Lötflächen hergestellt werden muss. Dabei ist darauf zu achten, dass nicht alle drei Lötflächen leitend verbunden werden.

Es können ROMs mit Speichergrößen zwischen 8 und 32 KB verbaut werden. Normalerweise sollte ein 32 KB großer Chip bestückt werden. Hierfür kann ein 27C256, W27E257 oder notfalls auch 27256 verwendet werden; der Jumper JP6 und der Widerstand R2 können dann entfallen. Wird der Jumper dennoch bestückt, darf er nicht gesteckt werden. Wer sich den Luxus eines im Computer aktualisierbaren 28C256 EEPROMs gönnen möchte, muss JP6 und R2 bestücken und den Jumper JP6 setzen, um das Beschreiben des Chips zuzulassen.

Bei Verwendung der 8 KB großen 2764/27C64 und der 16 KB großen 27128/27C128 kann der Jumper JP6 entfallen. Wird er dennoch bestückt, darf er nicht gesteckt werden. Der Widerstand R2 ist bei diesen Typen erforderlich und die Lötbrücke JP1 muss entgegen der Beschriftung auf die Position „28256“ gesetzt werden.

Wer mit 8 KB Speicher auskommt, kann auch das günstige EEPROM AT28C64 oder X28C64 verwenden, die im eingebauten Zustand beschrieben werden können, wenn Jumper JP6 gesetzt ist. Der Widerstand R2 ist hierfür ebenfalls zu bestücken. Der Lötjumper JP1 ist an der Position „28256“ zu setzen. Der Lötjumper JP2 darf an keiner(!) Position gesetzt werden, hier müssen also alle drei Lötflächen frei bleiben.

Bei der Zugriffszeit ist abhängig von der verwendeten Technologie des GALs an Position U4 und des achtfach NAND-Gatters an U8 die Obergrenze bei etwa 250 Nanosekunden anzusetzen. Dieser Wert wird von den meisten 32 KB großen Typen in CMOS-Technologie deutlich unterschritten.

Wenigstens die folgenden ROM-Chips können verwendet werden:

Kapazität	Hersteller	Typ	Löschen	Beschreiben	JP1	JP2
32 KB	Generic	27C256	UV-Licht	Programmiergerät	27	27
32 KB	Winbond	W27E257	Elektrisch	Programmiergerät	27	27
32 KB	Atmel	AT28C256	Im System	Im System	28	28
32 KB	Xicor	X28C256	Im System	Im System	28	28
16 KB	Generic	27C128	UV-Licht	Programmiergerät	28	27
8 KB	Generic	27C64	UV-Licht	Programmiergerät	28	27
8 KB	Atmel	AT28C64	Im System	Im System	28	–
8 KB	Xicor	X28C64	Im System	Im System	28	–

1.1.4 DUART 68681

Der ursprünglich als MC68681 von Motorola hergestellte Chip ist auch als TMP68681 von Toshiba, als SCC68681 von Philips bzw. NXP und als XR68C681 von Exar Corporation hergestellt worden.

Die maximale Baudrate, die Chips aller Hersteller unterstützen, sind 38400 Baud. Philips/NXP und Exar stechen hier heraus, da deren Chips SCC68681 bzw. XR68C681 Daten mit zu 115200 Baud übertragen können. Daher sollte für die Hauptplatine des Sparrow09 ausschließlich diese beiden Typen verwendet werden. Die langsameren Typen anderer Hersteller können dagegen ohne weiteres für das PS/2-Tastatur- und Maus-Interface verwendet werden, wo die höhere Geschwindigkeit nicht erforderlich ist.

1.1.5 MAX232

Leider werden von diesem Chip sehr häufig unbrauchbare Plagiate angeboten, die sich bei der ersten Benutzung durch latch up unter Wärmeentwicklung verabschieden. Hier sollten die Chips ausschließlich aus bewährten, etablierten Quellen bezogen werden.

Anstelle des klassischen MAX232 von Maxim kann auch der MAX232A verwendet werden, der sich auch schon mit kleineren Kondensatoren zufrieden geben würde, nämlich schon mit 100 nF statt den bislang erforderlichen 1 µF. Erweitert um die Angaben zur Gehäuseform und des zulässigen Temperaturbereichs lauten die vollständigen Typenbezeichnungen dann MAX232CPE, MAX232EPE, MAX232ACPE oder MAX232AEPE.

1.2 Bestückung der Platine

Diese Anleitung gilt für die Platinenversion „Issue 4“. Die Versionsnummer ist auf der Platine bei I/O Slot 1 aufgedruckt.

Beim Aufbau soll die „InteractiveHtmlBom“ verwendet werden. Diese Datei liegt im Sparrow-Projektordner unter hardware/sparrow-motherboard-v4/ibom.html. Mit ihrer Hilfe kann u.a. direkt die Position der Bauteile bestimmt werden, indem einfach mit der Maus über den entsprechenden Eintrag in der Teileliste gefahren wird. Rechts neben der Teileliste wird die Platine dargestellt, auf der die gewählten Bauteile grafisch hervor gehoben werden.

Es wird empfohlen, die Bauteile in der folgenden Reihenfolge zu bestücken:

1. SMD-Chip CH340B
2. 4 Stück M3×8 Distanzbolzen, befestigt mit M3×6 Linsenkopfschrauben

3. SD-Slot
4. Widerstand R1, 68 Ohm
5. Widerstand R2 und R3, beide 10k
6. Diode BAT42, auf Polarität achten!
7. Quarz 14,7456 MHz
8. IC-Sockel, dabei auf die Ausrichtung achten
9. Widerstandsnetzwerke 4x10k
10. Widerstandsnetzwerk 4x47k
11. Kerkos 18pF
12. Kerkos 1 µF
13. Kerkos 100 nF
14. 3,3V Spannungsregler U7 MCP1700-3302E
15. LED D2
16. Reset-Taster SW1
17. I/O Slots J3-6
18. I/O-Slot Erweiterung 90 Grad J1
19. Stiftleiste JP7 zur Stromversorgung
20. Stiftleiste J8 für /RESET
21. Stiftleiste mit Jumper JP6 (write enable)
22. DIP-Schalter SW2
23. Elkos 10 µF C11, C14, auf Polarität achten!
24. Elko 47 µF C1, auf Polarität achten!
25. 5V DC Buchse J2
26. USB-Buchse J9
27. Sub-D-Buchse J11
28. Lötbrücken JP3 auf der Unterseite der Platine, abhängig vom verwendeten RAM-Chip
29. Lötbrücken JP1 und JP2 auf der Unterseite der Platine, abhängig vom verwendeten ROM-Chip

1.3 Tests bevor(!) Chips eingesteckt werden

1. Am Sockel des MAX232 U6 Pin 13 und 14 mit einer Drahtbrücke verbinden. Beim Terminalprogramm lokales Echo einstellen. Die Platine braucht keine Stromversorgung, die Baudrate und andere Einstellungen sind egal. Es muss jetzt jedes Zeichen, das eingetippt wird, zwei mal am Bildschirm erscheinen: einmal als lokales Echo und einmal als Zeichen, das über das Kabel hin- und durch die Drahtbrücke wieder zurück geschickt wurde.
2. Alle DIP-Schalter auf „OFF“ stellen. Ohne Stromversorgung den Widerstand zwischen +5V und GND messen, dieser muss unmessbar hoch sein. Das Schalten eines Schalters auf „ON“ senkt den Widerstand auf 10 Kiloohm.
3. Bei eingesteckter Versorgung an GND und +5V messen. Der gemessene Wert muss +5 Volt ± 0,25 Volt betragen.
4. Bei eingesteckter Versorgung die Spannung an C3 direkt am SD-Karten-Slot messen.

Der gemessene Wert muss +3,3 Volt ± 0,2 Volt betragen.

5. Bei ausgesteckter Versorgung den MAX232 Chip einsetzen und Versorgung einstecken. An Pin 2 müssen ca. +9,8 Volt anliegen und an Pin 6 ca. -9,7 Volt.
6. Am Sockel des DUART U5 Pin 10 und 11 mit einer Drahtbrücke überbrücken. Echotest wie gehabt durchführen, jedes Zeichen muss nun wieder zwei mal erscheinen.
7. Versorgungsspannung ausstecken und Chips einsetzen. Achtung: die Orientierungskerben der großen Chips zeigen alle zum Platinenrand, bis auf die HD63C09P CPU, dort zeigt er zur Platinenmitte! Auch bei allen anderen Chips unbedingt auf die richtige Ausrichtung achten!
8. Jumper JP7 neben der DC-Buchse für die gewünschte Quelle der Stromversorgung einstecken.
9. Mit einem Oszilloskop kann bei angelegter Versorgungsspannung die Reset- Erzeugung und der Takt überprüft werden. An der zweipoligen Stiftleiste J8 oberhalb der CPU kann nach Druck auf die Reset-Taste verfolgt werden, wie binnen etwa einer halben Sekunde die Spannung von 0 Volt wieder auf 5 Volt ansteigt. Beim Betätigen der Reset-Taste muss die Spannung auf 0 Volt absinken. An Pin 34 der CPU liegt der Takt von einem Viertel der Quarz- Frequenz an, das sind 3,6864 MHz bei Verwendung eines 14,7456 MHz Quarzes.
10. Wenn Versorgungsspannung anliegt und die Platine mit einem USB-B-Kabel an den PC angeschlossen wurde, muss am PC eine neue virtuelle serielle Schnittstelle sichtbar werden. Mit `sudo dmesg | grep usb` kann man unter Linux dazu die letzten Systemmeldungen anzeigen lassen. Typischerweise wird ein neues Gerät /dev/ttyUSB0 erzeugt.

1.4 USB Interface konfigurieren (CH340B)

Das im Sparrow verwendete USB-Seriell-Interface CH340B stammt von der Firma Nanjing Qinheng Microelectronics Co., Ltd. / WinChipHead (WCH).

Es enthält einen kleinen nichtflüchtigen Speicher, der mittels eines durch den Hersteller gelieferten Windows-Programms konfiguriert werden kann. Damit kann der Sparrow fortan direkt als solcher erkannt werden, wenn er an einem PC oder Mac angeschlossen wird. Diese Konfiguration ist nur einmalig nach dem Bestücken der Hauptplatine des Sparrows erforderlich.

Die benötigte Software kann direkt vom Hersteller unter folgender URL bezogen werden: `https://www.wch-ic.com/downloads/CH34xSerCfg_ZIP.html`. Zusätzlich ist sie im git repository im Ordner software/Configure-CH340B-USB-interface/ abgelegt.

Der Ablauf der Konfiguration ist wie folgt:

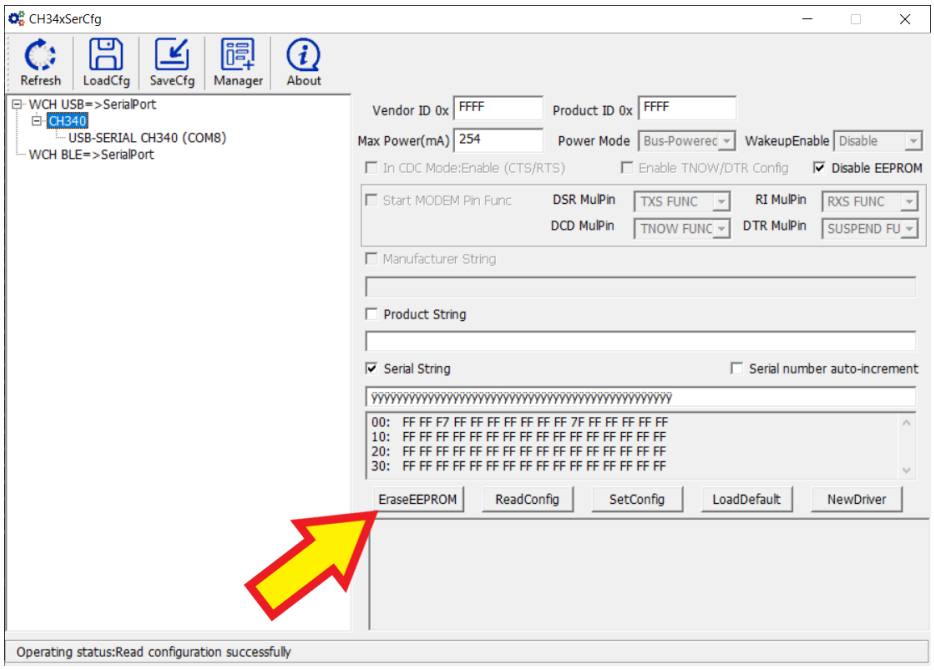
1. Sparrow einschalten

2. Sparrow über USB-Kabel ohne zwischengeschalteten USB-Hub mit PC verbinden

3. CH34xSerCfg.exe starten und die Frage „Möchten Sie zulassen, dass durch diese App Änderungen an Ihrem Gerät vorgenommen werden?“ mit „Ja“ beantworten

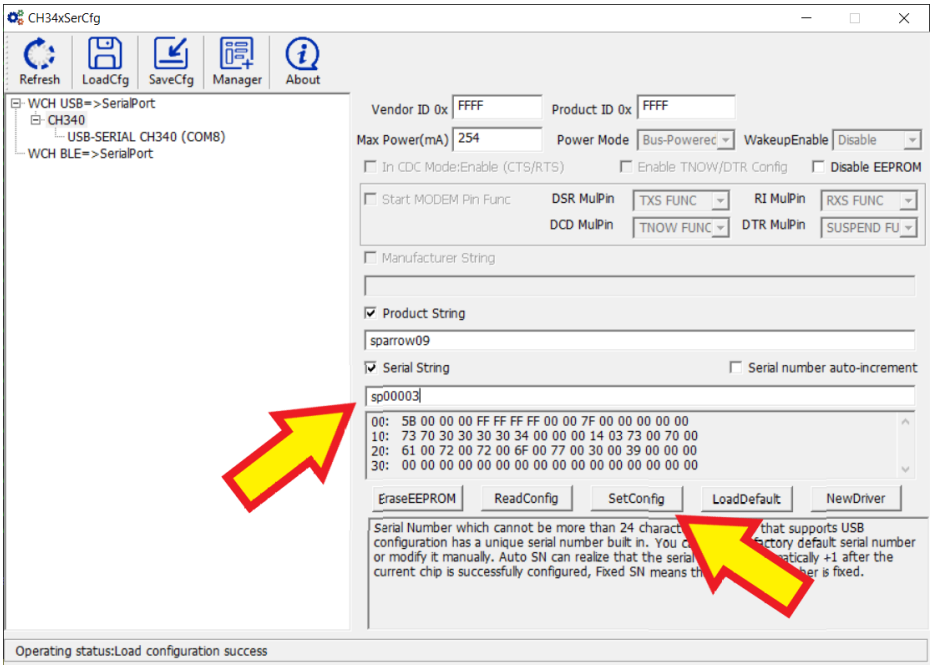
4. Im USB-Baum links im Programm muss nun ein **USB SERIAL CH340** mit zugewiesener COM-Port-Nummer auftauchen

5. Auf den Butten „Erase EEPROM“ klicken

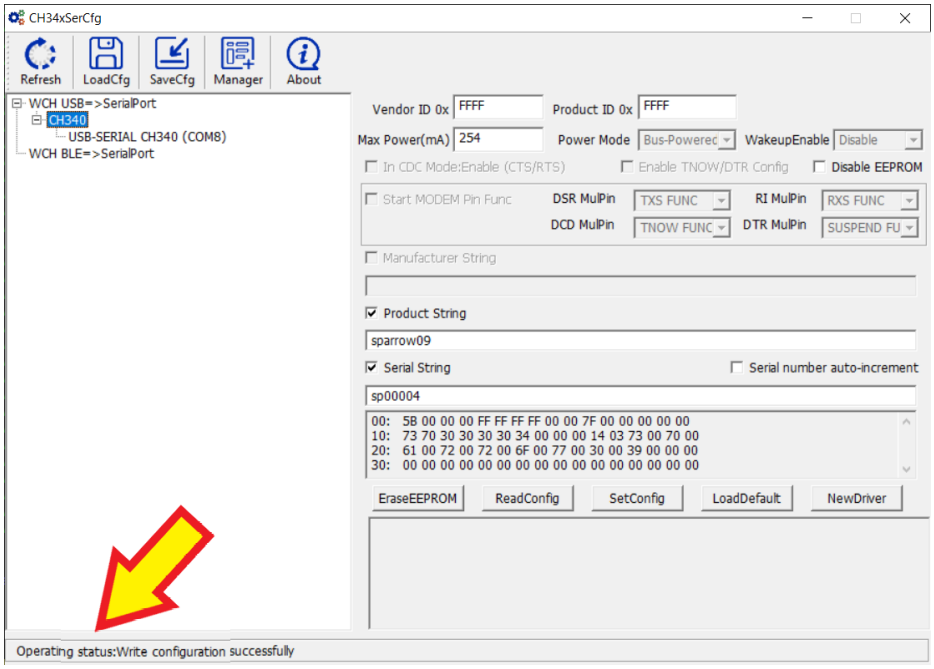


6. Oben links auf „LoadCfg“ klicken und die Datei **sparrow.cfg** durch Doppelklick laden

7. Rechts muss jetzt bei „Product string“ das Häkchen gesetzt sein und **sparrow09** eingetragen sein, dann die Seriennummer darunter anpassen. Die Gesamtzahl der Ziffern sollte unverändert bleiben.



8. Auf den Button „SetConfig“ klicken, wenig später muss unten links die Status-Meldung „Operation status: Write configuration successfully“ erscheinen, damit wurde der Vorgang erfolgreich abgeschlossen und das Programm kann beendet werden.



1.5 Spannungsversorgung

Der Sparrow kann entweder über ein USB-Kabel oder mittels externem Netzteil mit Strom versorgt werden. Die Auswahl wird mit dem Jumper JP7 neben der DC-Buchse getroffen.

Die Versorgung über USB ist eher als Hack, denn als Feature zu betrachten, da der tatsächliche Strombedarf größer sein kann, als der, der dem USB-Controller gemeldet wurde. Noch dazu wird direkt nach dem Einstecken des Sparrows die volle Leistung gezogen, ohne die Freigabe des USB-Controllers abzuwarten. Es kann daher durchaus vorkommen, dass der USB-Controller diesem wilden Treiben schnell ein Ende setzt oder die tatsächlich gelieferte Spannung die gewünschten 5 Volt unterschreitet. Der Sparrow kann unter Umständen auch mit Unterspannung funktionieren, aber wenn seltsame Effekte beim Betrieb auftreten, sollte man besser auf ein hochwertiges Steckernetzteil ausweichen.

Ein externes Netzteil wird empfohlen, wenn der Sparrow noch zusätzlich mit einigen Erweiterungskarten bestückt wird. Das Netzteil muss 5 Volt liefern. Die Stromstärke hängt vom Ausbauzustand ab, ohne Erweiterungen reichen 500 mA vollkommen aus, so dass man mit einem 2 A Netzteil für nahezu alle künftigen Ausbauzustände gewappnet sein sollte.

Der Stecker des Netzteils muss ein Hohlstecker mit Außendurchmesser 5,5 mm und Innendurchmesser 2,1 mm sein, Plus ist innen und Minus ist außen. Die Polarität darf auf keinen

Fall vertauscht werden und die Spannung niemals überschritten werden, da auf dem Sparrow keinerlei Schutzmaßnahmen gegen Verpolung oder Überspannung verbaut sind. Bei der Auswahl eines Netzteils ist also besondere Sorgfalt vonnöten!

1.6 GAL-Gleichungen

Im Anhang auf Seite 23 ist der Schaltplan der Hauptplatine des Sparrows enthalten. Die Funktionsweise des dort verwendeten programmierbaren Logikbausteins, ein GAL ATF22V10, ist naturgemäß nicht direkt ersichtlich. Die darin enthaltenen Terme sollen daher hier beschrieben werden.

Die Inhalte des GALs werden in einer Quelldatei mit der Dateierweiterung .PLD verfasst und durch den GAL-Assembler [1] [2] [3] in eine JEDEC-Datei mit der Dateierweiterung .JED übersetzt. Diese JEDEC-Datei wird dann von der Software eines GAL-Programmiergerätes verstanden und der GAL damit programmiert. Ein GAL kann etwa einhundertmal umprogrammiert werden.

Als Programmiergeräte kommen beispielsweise die GALEP-Geräte der Firma Conitec Datensysteme GmbH in Betracht oder von der chinesischen Firma XGecu die Geräte TL866IIPlus, T48 oder T56; ebenso von der Firma Hi-Lo Systems u.a. die Geräte ALL-03, ALL03A, ALL-07 und ALL-11.

Als bedingt geeignet hat sich Stand November 2023 der Batronix BX48 Batego II gezeigt, da dieser zu diesem Zeitpunkt nur GALs des Herstellers Lattice, nicht aber solche von Atmel programmieren kann. Leider sind ausgerechnet nur letztere noch als Neuware erhältlich, die von Lattice aber eben nicht.

GALs können in verschiedenen Modi betrieben werden. Hier wird der GAL in der einfachsten Variante „Kombinatorische Ausgänge“ verwendet, bei der alle Ausgänge ständig aktiviert sind und direkt von den anliegenden Eingängen abhängen. Es werden keine Flip-Flops verwendet.

Die Zuordnung der Eingangsvariablen zu den Ausgangsvariablen erfolgt über eine Wahrheitstabelle. In dieser Wahrheitstabelle können Signale direkt oder mit im Quelltext vorangestelltem Schrägstrich invertiert verwendet werden. Auch die Ausgänge können invertiert werden. Der Ausgangswert definiert sich durch mindestens einen Term, in dem Eingangsvariablen miteinander UND-verknüpft werden. Pro Ausgangsvariable können mehrere solcher Terme miteinander ODER-verknüpft werden, so dass Ausdrücke wie dieser entstehen: *Das Ausgangsbit Y soll gesetzt sein, wenn A UND B gesetzt sind ODER aber wenn A UND C UND /D (=D invertiert) gesetzt sind.* In der Syntax des GALasm würde der Produktterm dann wie folgt definiert:

[1] https://github.com/daveho/GALasm
[2] https://github.com/nils-eilers/GALasm
[3] https://github.com/simon-frankau/galette

```
Y = A * B
  + A * C * /D
```

Die folgenden Signale sind Eingänge: $E, \overline{FEFFXX}, \overline{DUART}, R/\overline{W}, A4, A8, A13, A14, A15, ROMON, B0, B1, B2$. Aus diesen erzeugt das GAL die Ausgänge $\overline{OE}, \overline{WE}, IOEN, \overline{ROMCS}, \overline{RAMCS}, SDCK, RA16, RA17$ und $RA18$.

```
/WE     = /RW * E
```

WE steht für write enable, low active. Der Schreibzugriff auf Speicher (WE=0) kann nur aktiviert werden, wenn gültige Daten und Adressen anliegen (E=1) und die CPU schreiben möchte (RW=0, read/write). Das invertierte RW-Signal der CPU und das Taktsignal E werden NAND-verknüpft.

```
/OE     =  RW * E
```

OE steht für output enable, low active. Speicherchips ist es nur erlaubt, Daten auf den Datenbus auszugeben (OE=0), wenn die Read/Write-Leitung der CPU high ist, die CPU also lesen möchte, und E=1 anzeigt, dass gültige Adressen am Adressbus anliegen. Das /RW-Signal der CPU und das Taktsignal E werden NAND-verknüpft.

```
SDCK    =  /DUART * A4
```

SDCK steht für SD card CLock. Die 16 Register des DUART liegen gespielt in den 32 Bytes der I/O-Karte vor. Bei Zugriffen auf die oberen 16 Bytes des Adressbereichs erfolgt ein normaler Zugriff auf die Register, gleichzeitig wird aber auch ein Puls am Takt-Signal der SD-Karte für die Dauer des Zugriffs erzeugt. Das dient dazu, die Zugriffe auf den SPI-Bus in Software etwas zu beschleunigen, da dadurch der Puls an der Takt-Leitung nicht mehr in Software erzeugt werden muss, sondern in Hardware erfolgen kann. Das invertierte Chip-Select Signal des DUART wird mit der Adressleitung A4 UND-verknüpft.

```
IOEN   = E * /FEFFXX * /A8
```

IOEN steht für Input Output ENable. Wenn das Signal high ist, erfolgt im Adressbereich $FE00-$FEFF ein Zugriff auf angeschlossene Peripherie. Der Zugriff erfolgt nur, wenn gültige Adressen anliegen (E=1). E wird mit der invertierten Adressleitung A8 und dem invertierten Signal FEFFXX UND-verknüpft.

```
/ROMCS  = E * /FEFFXX * A8
        + E * A15 * ROMON * /IOEN
```

Im Adressbereich von $FF00 bis $FFFF ist das ROM immer eingeblendet, unabhängig von den Signalen ROMON oder der mit B0-B2 gewählten RAM-Bank. Dieser Fall wird durch die erste Zeile erzeugt, außerdem müssen gültige Adressen anliegen (E=1). Die zweite Zeile blendet das ROM im Bereich $8000-$FFFF abzüglich des I/O-Bereichs von $FE00-$FEFF ein, wenn das ROMON-Signal high ist und gültige Adressen anliegen (E=1). E ist dann high; A15 high zeigt den Adressbereich $8000-$FFFF an und I/O nicht aktiv (IOEN=0) maskiert den ROM-Bereich innerhalb des I/O-Bereichs.

```
/RAMCS  = E * /A15
        + E * FEFFXX * /RW
        + E * FEFFXX *  RW * /ROMON
```

Das RAM wird unter drei verschiedenen Umständen angesprochen:

`/RAMCS = E * /A15` bedeutet, dass das RAM immer frei gegeben wird, wenn gültige Adressen ausgegeben werden (E=1) und diese die unteren 32K auswählen (A15=0).

`E * FEFFXX * /RW`: das RAM wird selektiert, wenn gültige Adressen ausgegeben werden (E=1), weder der I/O-Bereich, noch der Vektorenbereich adressiert wird (FEFFXX=1) und in den Speicher geschrieben wird (RW=0).

`E * FEFFXX * RW * /ROMON`: das RAM wird selektiert, wenn gültige Adressen ausgegeben werden (E=1), weder der I/O-Bereich, noch der Vektorenbereich adressiert wird (FEFFXX=1), gelesen werden soll (RW=1) und das ROM ausgeblendet ist (ROMON=0).

```
RA16    = /A15 * /A14 * /A13
        +  B0
RA17    = /A15 * /A14 * /A13
        +  B1
RA18    = /A15 * /A14 * /A13
        +  B2
```

RA16 bis RA18 sind Adressleitungen, die 64KB große Speicherbereiche innerhalb des RAM-Chips auswählen. Die Adressleitungen A0-A15 werden von der CPU geliefert, die höchstwertigen Adressleitungen RA16-RA18 werden durch das GAL erzeugt. Mit drei Adressleitungen sind $2^3 = 8$ Bänke auswählbar, damit können bis zu 8 x 64 KB = 512 KB RAM adressiert werden. Die Auswahl der aktiven RAM-Bank erfolgt durch die drei Banking-Signale B0-B2. Im Adressbereich $2000-$FFFF werden die Banking-Signale B0-B2 als RA16-RA18 weiter

gereicht, dies erfolgt durch die Zeilen `+ B0`, `+ B1` und `+ B2`. Wenn die unteren 8 KB des 64KB-Adressbereichs der CPU adressiert werden, sind A13, A14 und A15 low. In diesem Fall sind RA16 bis RA18 immer high. Als Beispiel für RA16: RA16 ist high, wenn A13 und A14 und A15 low sind. Wenn dies nicht der Fall ist, ist RA16 high, wenn B0 high ist. Wenn beides nicht der Fall ist, ist RA16 low.

2 Style Guide für Assembler-Quelltexte

Vorab sei erwähnt, dass dieser Style-Guide für Assembler kein Gesetzestext sein möchte, an den sich strikt zu halten sei. Er ist lediglich als Hilfestellung gedacht, um die Erstellung lesbarer und somit wartbarer Quelltexte zu erleichtern. Anwendung soll er finden bei der Sparrow09 Firmware und zugehörigen Hilfsprogrammen, die im Sparrow-Repository zentral verwaltet werden, für eigene Anwenderprogramme kann der jeweilige Programmierer seiner Phantasie bei der Gestaltung eigener Programme natürlich freien Lauf lassen.

Mnemonics, Pseudobefehle und Operanden werden klein geschrieben, Macros dagegen groß um sie direkt als solche kenntlich zu machen. Für Labels, die aus mehreren Worten zusammen gesetzt werden, darf gerne der Wortanfang groß geschrieben werden (`Wait_For_Byte`), kürzere Einwortlabels sollten komplett klein geschrieben werden (`.loop`).

Die Ausrichtung von Mnemonics, Operanden und Kommentaren orientiert sich an den Tab-Positionen. Als Tab-Größe wird acht verwendet, wobei der Quelltext keine harten Tabs enthalten soll, sondern Tabs sollen zu Leerzeichen expandiert werden. Mnemonics und Pseudobefehle starten ab Spalte 9, Operanden ab Spalte 17 und Kommentare ab Spalte 33. Kommentare, die einer Befehlszeile angehängt werden, sollen durch Semikolon und Leerzeichen abgetrennt sein, wogegen ganzzeilige Kommentare gerne mit dem Stern eingeleitet werden dürfen.

Labels, die von anderen Programmteilen aufgerufen werden, sind globale Labels. Für andere Labels, die nur hilfsweise innerhalb eines Unterprogramms oder Moduls verwendet werden, sollen lokale Labels mit vorangestelltem Punkt verwendet werden. Hierzu wird mit `MODULE foo` der Beginn des Bereichs abgesteckt, in dem lokale Variablen gelten, bis `ENDMOD` das Ende markiert. Auf diese Weise können einfache Namen wie beispielsweise `.loop` immer wieder verwendet werden.

Der BS9 Assembler unterstützt zwar auch die Verwendung namenloser lokaler Labels, die mit Plus- und Minus oder einer Verkettung dieser Zeichen benannt werden können, aber aus Gründen der Lesbarkeit und Fehlerträchtigkeit sollte auf diese Art der Namensgebung verzichtet werden. Ebenso sollten an globale Labels keine Zahlen angehängt werden.

Um Quelltexte an das Standardformat anzugleichen, kann der Quelltextformatierer form9 verwendet werden, der mit dem Assembler BS9 von https://github.com/Edilbert/BS9 herunter geladen werden kann.

```
*********
Mon_Colon
*********

        lda     #':'
        CALL    PUTC
```

```
        rts


*************************************************
* Test if a byte is available from the serial port *
* ---------------------------------------------- *
* Output: B: zero if no data available            *
*************************************************
module  Ser_Available
        ldb     %0000001        ; RxRDYA
        tst     BIOS_Channel_In
        bne     .chB
        andb    DUART_SRA
        stb     SOFF_B,S        ; return flag in stacked B
        rts
.chB    andb    DUART_SRB
        stb     SOFF_B,S        ; return flag in stacked B
        rts
endmod
```

3 Memory Map

3.1 Memory Map von RAM, ROM und I/O

Der Adressbereich der CPU umfasst insgesamt 64 Kilobytes, die sich über drei verschiedene Bereiche erstrecken: RAM, ROM und den Ein-/Ausgabebereich.

Der Ein-/Ausgabebereich, auch kurz *I/O-Bereich* genannt, umfasst 256 Bytes von $FE00-$FEFF und wird insgesamt 8 Erweiterungskarten (I/O-Slots 1-8) mit je 32 Bytes zur Verfügung gestellt. An Slot 1-4 können auf der Hauptplatine Erweiterungskarten eingesteckt werden, Slot 5 bis 7 können über eine Buserweiterung angesteckt werden und Slot 8 wird bereits intern auf der Hauptplatine durch den DUART-Chip verwendet. Der I/O-Bereich ist unabhängig von der aktuell eingestellten Banking-Konfiguration immer sichtbar.

An den oberen 32 KB von $8000-$FFFF ist nach dem Systemstart bzw. nach Reset das ROM eingeblendet, abzüglich des I/O-Bereichs, der 256 Bytes des 32 KB großen ROMs verdeckt. Mittels des ROMON-Signals kann das ROM ausgeblendet werden, dann wird die obere Hälfte der aktuell gewählten 64 KB großen RAM-Bank selektiert. Auch wenn das ROM eingeblendet ist, greifen Schreibzugriffe immer auf das darunter liegende RAM zu.

Die oberste Speicherseite des ROMs von $FF00-$FFFF ist immer eingeblendet, auch wenn der Rest des ROMs durch NOROM ausgeblendet wird, damit die Interrupt-Vektoren der CPU immer zur Verfügung stehen.

Abhängig vom verwendeten Speicherbaustein werden bis zu 512 KB RAM verteilt auf 8 Bänke von bis zu 64 KB Größe zur Verfügung gestellt. Die untersten 8 KB von $0000-$1FFF adressieren dabei immer die höchste Bank 7, die auch nach einem Reset vorausgewählt ist. Dieser Bereich wird auch als *common RAM* bezeichnet. Der darüber liegende Bereich erstreckt sich entweder bis $7FFF, wenn das ROM eingeblendet ist, oder bis an den I/O-Bereich, also bis Adresse $FDFF, wenn das ROM ausgeblendet ist.

3.2 Nicht nutzbarer Speicher

Die obersten 256 Bytes einer 64 KB umfassenden RAM-Seite werden immer durch die ROM-Seite der Interrupt-Vektoren überdeckt und stehen daher dem System nicht zur Verfügung. Die darunter liegenden 256 Bytes werden durch den I/O-Bereich verdeckt und stehen daher dem System ebenfalls nicht zur Verfügung. Die untersten 8 KB RAM adressieren immer RAM Bank 7, daher werden die untersten 8 KB der Bänke 0 bis 6 durch das 8 KB große *common RAM* der Bank 7 verdeckt und stehen daher ebenfalls nicht zur Verfügung. Insgesamt können also $8 \times (256 + 256) + 7 \times 8192$ Bytes = 60 KB von 512 KB nicht verwendet werden, so dass maximal 452 KB RAM effektiv zur Verfügung stehen. Bei Verwendung eines 128 KB großen

Speicherbausteins stehen dem System tatsächlich 128 KB - 2 × (256 Bytes+ 256 Bytes) - 8 KB = 119,5 KB RAM zur Verfügung.

Vom 32 KB großen ROM wird die vorletzte 256 Bytes große Seite durch den I/O-Bereich verdeckt, so dass dieser Speicherbereich dem System nicht zur Verfügung steht. Hier können Prüfsummen, Versionsnummern oder mehr oder weniger geheime Nachrichten abgelegt werden, die beim Umgang mit ROM-Abbildern verwendet werden können.

3.3 Adressen der I/O-Slots

Der Computer kann durch das Einstecken von Erweiterungskarten erweitert werden. Hierzu stehen auf dem Motherboard vier Steckanschlüsse „I/O SLOT 1-4“ zur Verfügung und an der linken Seite ist ein *slot expander* vorgesehen, in dem weitere Erweiterungskarten eingesteckt werden können. Jeder Erweiterungskarte ist dabei ein fester 32 Bytes großer Adressbereich zugewiesen, der sich aus der Position ergibt, an der die Karte eingesteckt wird. Die Basisadresse errechnet sich dabei wie folgt:

Basisadresse = $FE00 + (Slotnummer-1) × 32

Der auf der Hauptplatine verbaute DUART 68681 wird am Adressbereich des Slot 8 angesprochen:

DUART = $FE00 + (8-1) × 32 =
DUART = $FE00 + 7 × 32 =
DUART = $FE00 + 224
DUART = $FEE0

Position	Basisadresse Hex	Basisadresse Dezimal
Slot 1	$FE00	65024
Slot 2	$FE20	65056
Slot 3	$FE40	65088
Slot 4	$FE60	65120
Slot 5	$FE80	65152
Slot 6	$FEA0	65184
Slot 7	$FEC0	65216
Slot 8	$FEE0	65248

3.4 Ein- / Ausgabeadressen

3.4.1 DUART_RDIP

DUART_RDIP = DUART+$0D = $FEED dient zum Lesen der Eingabeleitungen IP0 bis IP5

Bit	Name	Signal	Beschreibung
0	IP0	SDCD	0: SD-Karte eingelegt, 1: keine Karte eingelegt
1	IP1	CTSI	CTS serieller Port B (RS232)
2	IP2	SDDO	Datenbit von der SD-Karte
3	IP3	SDWP	0: Schreiben auf SD-Karte möglich, 1: Karte schreibgeschützt
4	IP4	SW1	0: DIP Schalter 1 ist ON, 1: DIP Schalter 1 ist OFF
4	IP5	SW2	0: DIP Schalter 2 ist ON, 1: DIP Schalter 2 ist OFF
6	-	-	nicht verwendet, normalerweise 1
7	-	-	nicht verwendet, normalerweise 1

3.4.2 DUART_SETOP / DUART_RESOP

Die Ausgabebits des DUART sind nach dem Einschalten oder nach Reset alle auf eins gesetzt, an den Ausgängen liegen also ca. fünf Volt an. Dieses Verhalten ist insbesondere für das Banking mittels ROMON und Bankwahl-Leitungen B0, B1 und B2 praktisch.

DUART_SETOP = DUART+$0E = $FEEE write: Set Output Port Bits Command

DUART_RESOP = DUART+$0F = $FEEF write: Reset Output Port Bits Command

Bit	Name	Signal	Beschreibung
0	OP0	LED	Leuchtdiode, in Hardware erneut invertiert
1	OP1	RTSO	RTS serieller Port B (RS-232)
2	OP2	B0	Wahl der RAM-Bank
3	OP3	B1	Wahl der RAM-Bank
4	OP4	B2	Wahl der RAM-Bank, CS2 bei 128 KB RAM Chips
5	OP5	ROMON	0: durchgehend RAM, 1: ROM in oberer Speicherhälfte
6	OP6	SDDI	Datenbit zur SD-Karte
7	OP7	SDCS	Enable-Leitung der SD-Karte

Die digitale Ausgabe erfolgt über zwei verschiedene Register, je nachdem ob ein Bit gesetzt oder rückgesetzt werden soll. Schreibzugriffe auf DUART_SETOP (*set output port*) werden dazu verwendet, Bits zu setzen, wohingegen Schreiben auf DUART_RESOP (*reset output port*) dazu verwendet wird, Bits zurück zu setzen. Das Schreiben eines Bits, das logisch eins

ist, bewirkt das Setzen bzw. Rücksetzen des Ausgangs, Bits auf logisch Null bewirken keine Änderung.

Im Gegensatz zum Leseregister werden die ausgegebenen Bits invertiert. Wenn also mittels DUART_SETOP ein Bit gesetzt wird, liegt am Ausgang des DUART-Bausteins tatsächlich logisch Null an.

Wird als Beispiel der Wert 2 (binär 0000 0010) in das Register DUART_RESOP geschrieben, so werden am Ausgang OP1 etwa 5 Volt, also logisch eins anliegen.

Eine Ausnahme stellt die LED dar, da diese hardwaremäßig so verschaltet ist, dass das invertierte Ausgangssignal erneut invertiert wird. Die LED leuchtet also nach Schreiben eines gesetzten Bits auf DUART_SETOP und erlischt nach Schreiben auf DUART_RESOP.

3.4.3 SPI_CLK_SETOP / SPI_CLK_RESOP / SPI_CLK_RDIP

```
SPI_CLK_SETOP = DUART_SETOP+16 ;pulse clk line,then set bit
SPI_CLK_RESOP = DUART_RESOP+16 ;pulse clk line,then reset bit
SPI_CLK_RDIP  = DUART_RDIP+16  ;read input ports,pulse clk line
```

SD-Karten können über verschiedene Schnittstellen angesprochen werden. Beim Sparrow wird dazu der SPI-Bus verwendet, das ist ein serieller Bus mit einer Taktleitung, bei der bei jedem Takt ein Bit vom Sender zum Empfänger und gleichzeitig ein Bit in die Gegenrichtung übertragen wird. Zum Austausch mit der CPU, die normalerweise 8 Bit parallel verarbeitet, muss daher eine Wandlung seriell/parallel vorgenommen werden. Diese erfolgt beim Sparrow prinzipiell in Software über das sogenannte Bit-Banging-Verfahren, bei dem softwaregesteuert die Bits manuell heraus- und herein geschoben werden. Das ist natürlich deutlich langsamer, als wenn eine spezielle Hardware diese Wandlung vornehmen würde. Um das Bit-Banging wenigstens etwas zu beschleunigen, wurde beim Sparrow daher schaltungstechnisch in die Trickkiste gegriffen um wenigstens das Taktsignal in Hardware zu erzeugen: dazu wird auf den DUART über andere Adressen zugegriffen. Der GAL-Chip auf der Hauptplatine erkennt, dass der DUART über die anderen Adressen angesprochen wurde und erzeugt in diesem Fall zusätzlich einen Takt-Impuls für die SD-Karte.

Der DUART 68681 verfügt über 16 Register und ist auf der Hauptplatine fest an den Adressen von Slot 8 angeschlossen. Jeder Slot umfasst aber 32 Adressen, 16 blieben also ungenutzt. An diesen 16 Adressen wird der DUART nun noch einmal eingeblendet (*gespiegelt*), aber Zugriffe über diese Adressen erzeugen gleichzeitig einen Taktpuls für die SD-Karte.

Normalerweile erfolgt die digitale Eingabe über DUART_RDIP und die Ausgabe über die Adressen DUART_SETOP und DUART_RESOP. Diese Adressen an der gespiegelten, um 16 Adressen höheren Lage, sind in common.as9 als SPI_CLK_SETOP, SPI_CLK_RESOP

und SPI_CLK_RDIP definiert. Der Zugriff auf den DUART über diese Adressen erfolgt wie gewohnt, nur mit dem Unterschied, dass zusätzlich der GAL einen Taktimpuls erzeugt, wenn über diese höheren Adressen zugegriffen wird.

3.4.4 AV-Karte

Die Audio/Video-Karte (kurz: AV-Karte) ist mit einem Yamaha V9958 zur Bilderzeugung und einem Yamaha OPL2 Sound Chip YM3812 bestückt.

Sie ist mit 128 KB Video RAM ausgestattet und erlaubt Grafikauflösungen von bis zu 512×212 Pixeln bei maximal 19268 Farben und bis zu 32 Sprites. Auch Text-Modes von 80×24 oder 32×24 Zeichen sind möglich. Der VDP kann unter anderem selbstständig Bereiche kopieren, füllen und Linien zeichen, der Grafikbereich kann vertikal wie horizontal gescrollt werden. Der V9958 stellt eine Weiterentwicklung des V9938 dar und wurde unter anderem in MSX2+ Computern verbaut.

Der Soundchip YM3812, auch bekannt als OPL2, kann bis zu 9 Stimmen gleichzeitig erzeugen und wurde im PC in Adlib- und Soundblaster-Karten verwendet.

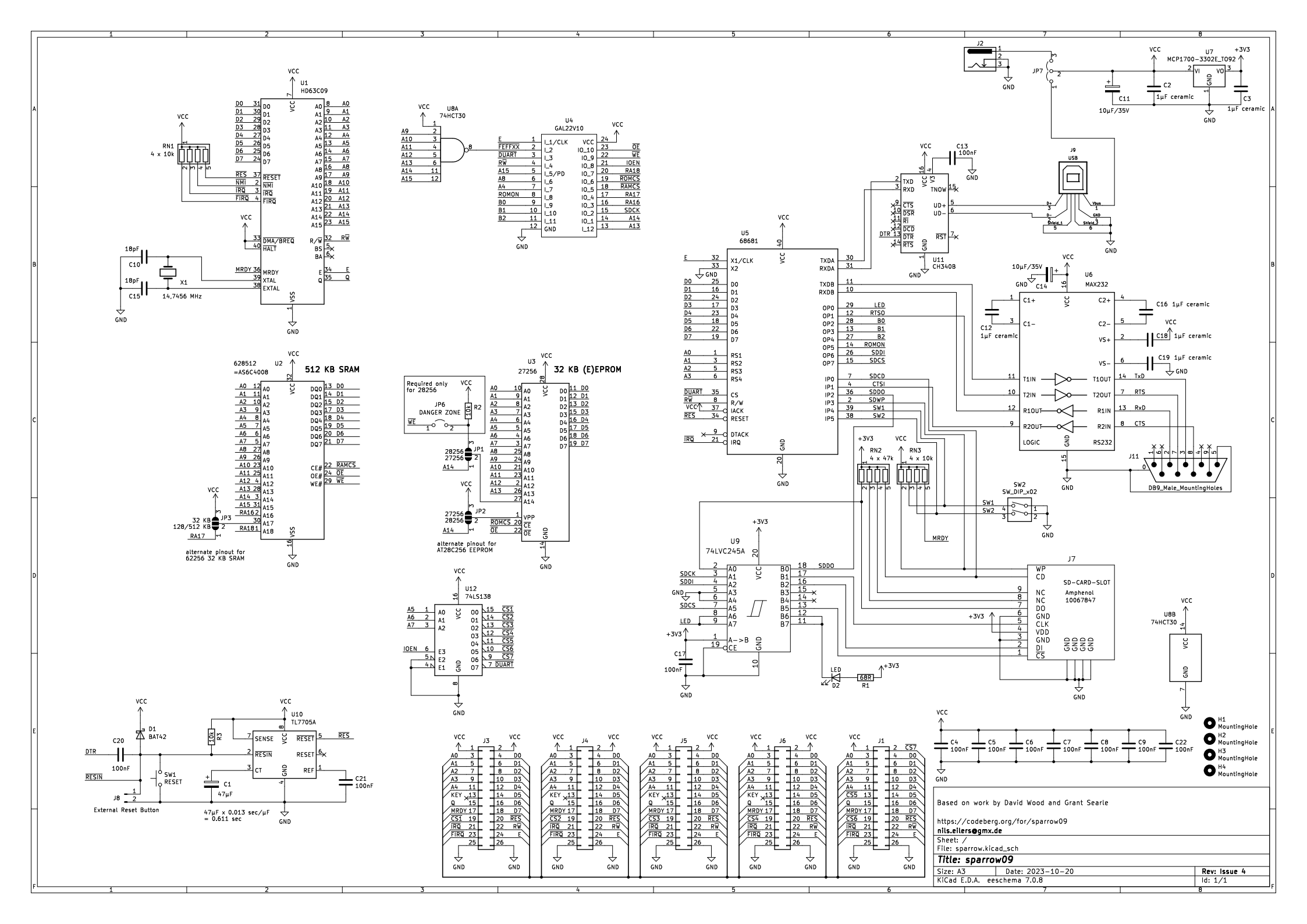
Als Standard sollte die AV-Karte in Slot 3 verwendet werden, so dass sich folgende I/O-Adressen ergeben:

Adresse	Offset	Name	Beschreibung
$FE40	0	IO_VDPRAM	VDP Daten
$FE41	1	IO_VDPSTATUS	Lesen: VDP Status
$FE41	1	IO_VDPSTATUS	Schreiben: VDP Befehle
$FE42	2	IO_VDPPALETTE	VDP Palette
$FE43	3	IO_VDPINDIRECT	VDP Indirect
$FE44	4	OPL2_REGSEL	Schreiben: OPL2 Registerauswahl
$FE44	4	OPL2_STATUS	Lesen: OPL2 Status
$FE45	5	OPL2_DATA	OPL2 Daten

Die anderen Adressen von $FE46 bis $FE5F mit den Offset-Adressen 6 bis 31 sollten nicht verwendet werden. Manche enthalten Spiegelungen der genannten Adressen, einige andere werden nicht verwendet.

4 Anhänge

4.1 Schaltpläne



Based on work by David Wood and Grant Searle

https://codeberg.org/for/sparrow09

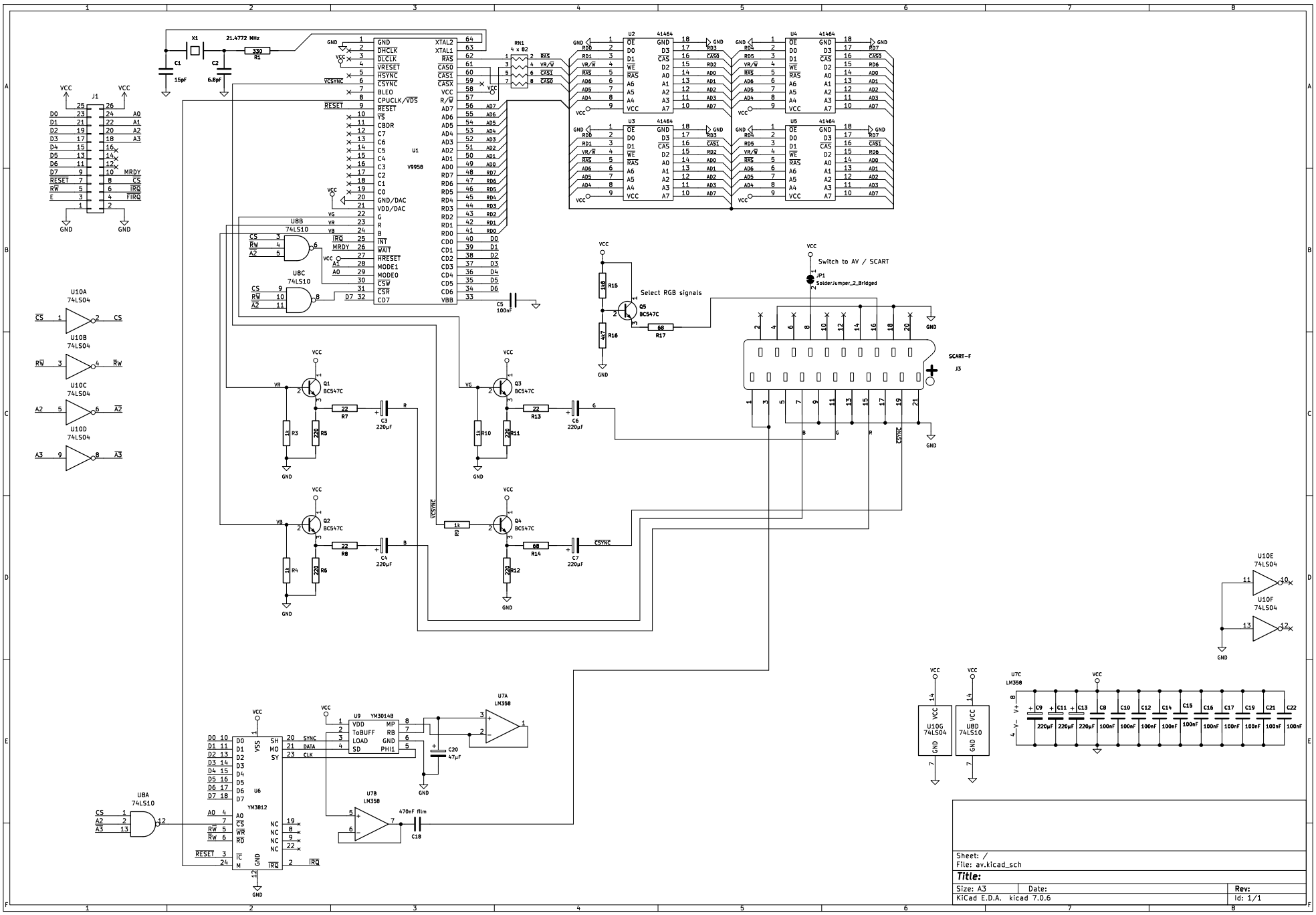
nils.eilers@gmx.de

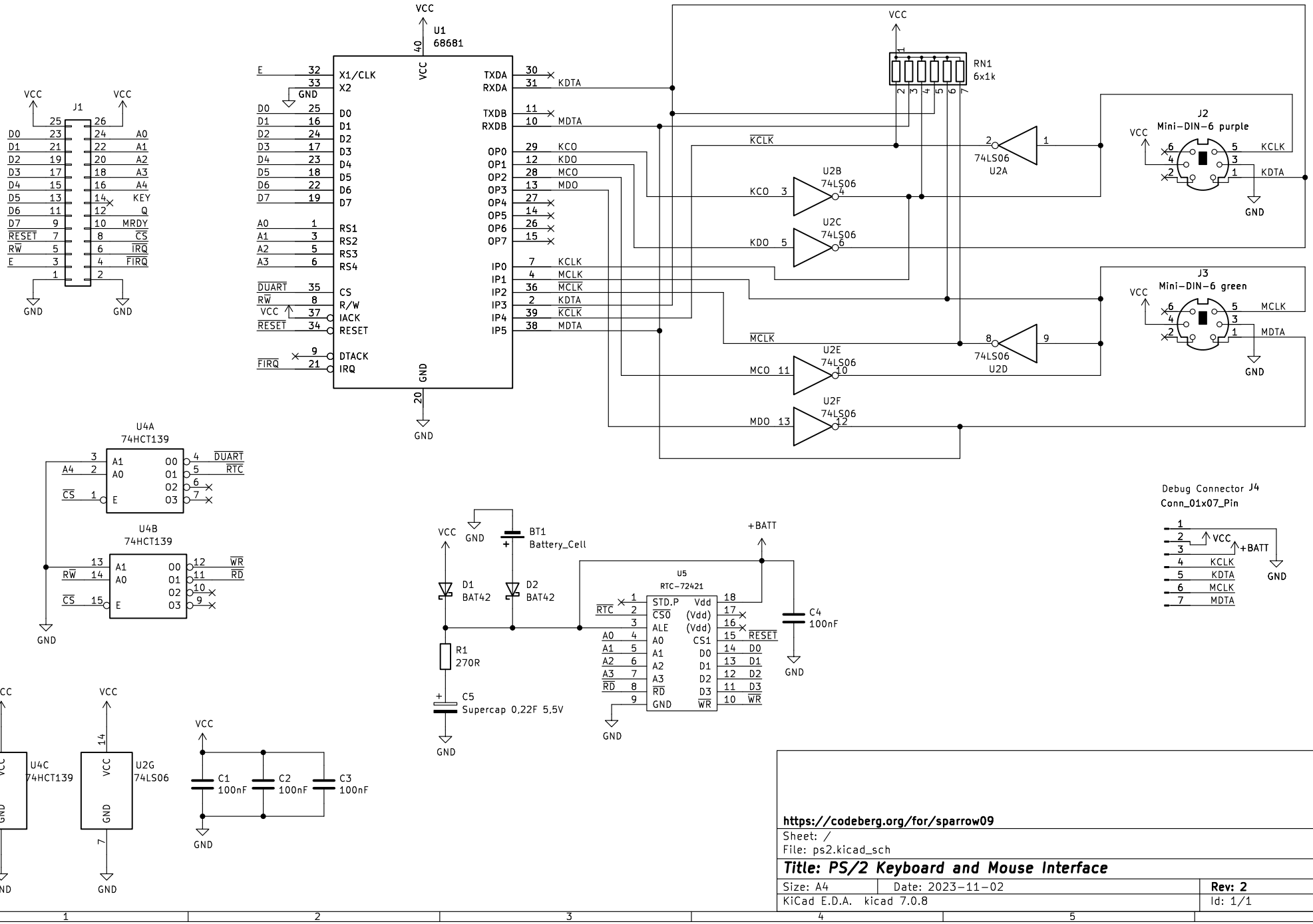
Sheet: /

File: sparrow.kicad_sch

Title: sparrow09

Size: A3	Date: 2023-10-20	**Rev: Issue 4**
KiCad E.D.A. eeschema 7.0.8		Id: 1/1


X1
21.4772 MHz
C1
15pF
C2
6.8pF
R1
330
J1
U1
V9958
RN1
4 x 82
U2
U3
U4
U5
41464
UBB
74LS10
UBC
74LS10
C5
100nF
Select RGB signals
Switch to AV / SCART
JP1
SolderJumper_2_Bridged
SCART-F
J3
Q1
BC547C
Q2
BC547C
Q3
BC547C
Q4
BC547C
Q5
BC547C
R15
148
R16
417
R17
68
R3
1k
R4
1k
R5
220
R6
220
R7
22
R8
22
R10
1k
R11
220
R12
220
R13
22
R14
68
C3
220µF
C4
220µF
C6
220µF
C7
220µF
U10A
74LS04
U10B
74LS04
U10C
74LS04
U10D
74LS04
U10E
74LS04
U10F
74LS04
U10G
74LS04
U8A
74LS10
U8B
74LS10
U9
YM3014B
U6
YM3812
U7A
LM358
U7B
LM358
U7C
LM358
C20
47µF
C18
470nF film
C8
100nF
C9
220µF
C10
100nF
C11
220µF
C12
100nF
C13
220µF
C14
100nF
C15
100nF
C16
100nF
C17
100nF
C19
100nF
C21
100nF
C22
100nF
VCC
GND
Sheet: /
File: av.kicad_sch
Title:
Size: A3
Date:
Rev:
KiCad E.D.A. kicad 7.0.6
Id: 1/1

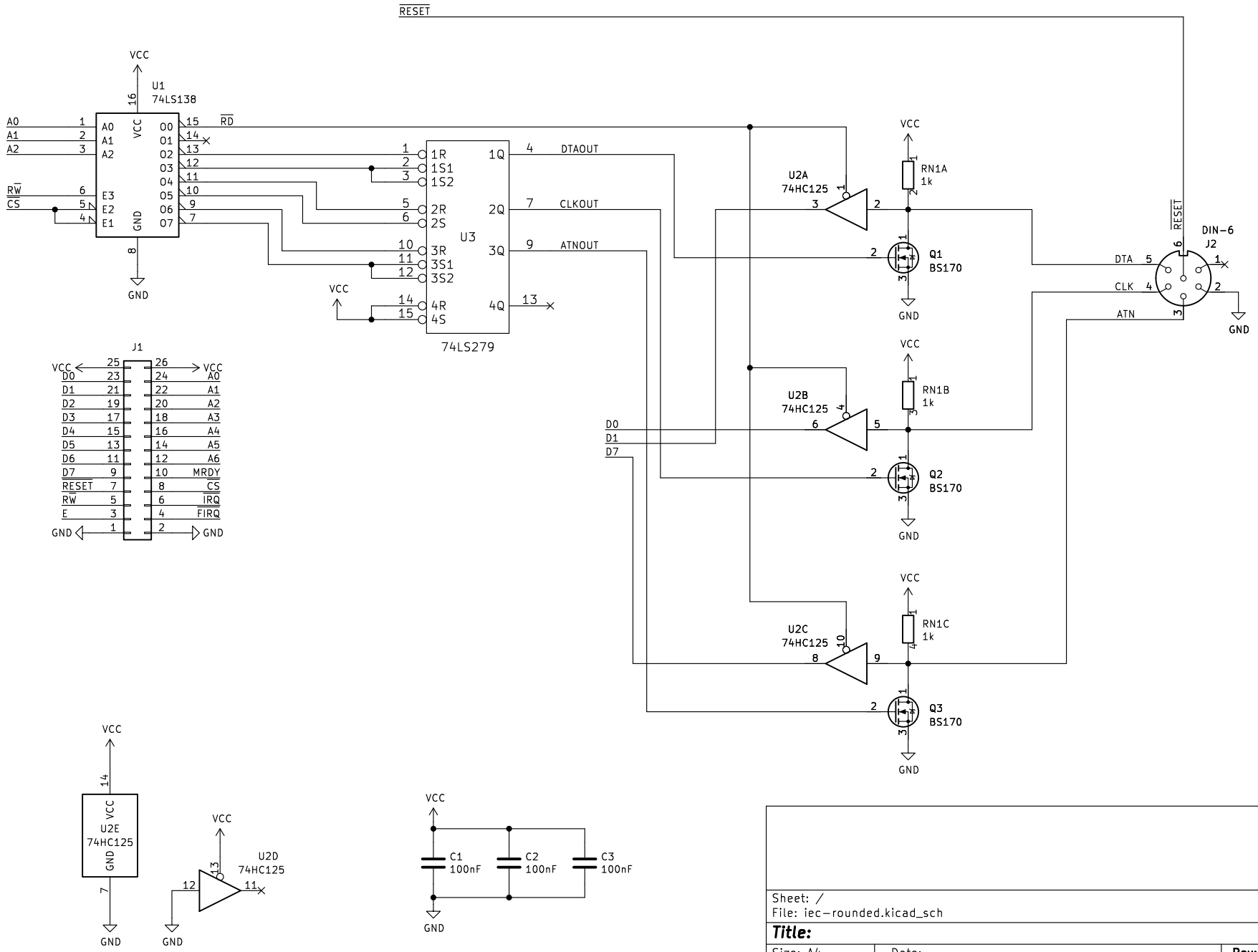


https://codeberg.org/for/sparrow09

Sheet: /
File: ps2.kicad_sch

Title: PS/2 Keyboard and Mouse Interface

Size: A4	Date: 2023-11-02	Rev: 2
KiCad E.D.A. kicad 7.0.8		Id: 1/1



Sheet: /		
File: iec-rounded.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 7.0.1		Id: 1/1