

Sign In (<https://www.hackster.io/users/auth/arduino?>

[current_site=arduino&setup=true&redirect_to=%2Fprojecthub%2FRobSmithDev%2Farduino-amiga-floppy-disk-reader-writer-v2-2-239c97">current_site=arduino&setup=true&redirect_to=%2Fprojecthub%2FRobSmithDev%2Farduino-amiga-floppy-disk-reader-writer-v2-2-239c97\)](#)




Arduino Amiga Floppy Disk Reader/Writer (V2.2) © GPL3+ (<http://opensource.org/licenses/GPL-3.0>)

An Arduino powered floppy disk controller and reader/writer for making and writing disk images from and to old AmigaDOS floppy disks.

floppy disk (/projecthub/projects/tags/floppy+disk)

29,606 VIEWS 26 COMMENTS 22 RESPECTS

COMPONENTS AND SUPPLIES



Arduino UNO

(/projecthub/products/buy/41?s=BAhJlh1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

(/projecthub/products/buy/41?s=BAhJlh1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

×1

(/projecthub/products/buy/41?s=BAhJlh1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

Shopping cart icon

(/projecthub/products/buy/41?s=BAhJlh1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

https://create.arduino.cc/projecthub/RobSmithDev/arduino-amiga-floppy-disk-reader-writer-v2-2-239c97

2/45

4

1

?

s

=

B

A

h

Jl

h

l

l

M

z

c

l

M

i

x

Q

c

m

9

q

Z

W

N

O

B

j

o

G

R

U

Y

%

3


D

%
O
A
)



SparkFun Arduino Pro Mini 328
- 5V/16MHz
(/projecthub/products/buy/42746?
s=BAhJlh1lMzc1MixQcm9qZWNOBjoGRUY%3D%0A)

(/projecthub/products/buy/42746?
s=BAhJlh1lMzc1MixQcm9qZWNOBjoGRUY%3D%0A)

× 1 
((/proj (/projecthub/product
/ ecthu s/buy/42746?
p b/pro s=BAhJlh1lMzc1MixQc
r ducts m9qZWNOBjoGRUY%
o /buy/ 3D%0A)
j 42746
e ?
c s=BAh
t Jlh1lM
h zc1Mi
u xQcm
b 9qZW
/ NOBjo
p GRUY
r %3D%
o 0A)
d
u
c
t
s
/
b
u
y
/
4
2
7
4
6

?

S

=

B

A

h

Jl

h

l

l

M

z

c

l

M

i

x

Q

c

m

9

q

Z

W

N

O

B

j

o

G

R

U

Y

%

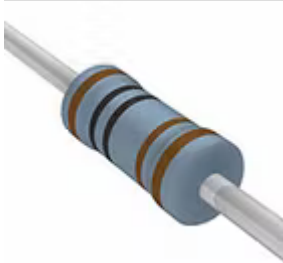
3

D

%


O

A
)



Resistor 1k ohm
(/projecthub/products/buy/37912?
s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

(/projecthub/products/buy/37912?
s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

× 1 
((/proj (/projecthub/product
/ ecthu s/buy/37912?
p b/pro s=BAhJlhI1Mzc1MixQc
r ducts m9qZWNOBjoGRUY%
o /buy/ 3D%0A)
j 37912
e ?
c s=BAh
t JlhI1M
h zc1Mi
u xQcm
b 9qZW
/ NOBjo
p GRUY
r %3D%
o 0A)
d
u
c
t
s
/
b
u
y
/
3
7
9
1
2
?
s


=
B
A
h
Jl
h
l
l
M
z
c
l
M
i
x
Q
c
m
9
q
z
W
N
O
B
j
o
G
R
U
Y
%
3
D
%
O
A
)



Breadboard (generic)

(/projecthub/products/buy/42749?s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

(/projecthub/products/buy/42749?s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

× 1 
((/projecthub/products/buy/42749?s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)
P b/products/buy/42749?s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)
r ducts
o /buy/42749?
j 42749
e ?
c s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)
t JlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)
h zc1MixQcm9qZWNOBjoGRUY%3D%0A)
u xQcm9qZWNOBjoGRUY%3D%0A)
b 9qZWNOBjoGRUY%3D%0A)
/ NOBjoGRUY%3D%0A)
P GRUY%3D%0A)
r %3D%0A)
o OA)
d
u
c
t
s
/
b
u
y
/
4
2
7
4
9
?
s
=
B

A
h
Jl
h
l
l
M
z
c
l
M
i
x
Q
c
m
9
q
z
W
N
O
B
j
o
G
R
U
Y
%
3
D
%
O
A
)



Jumper wires (generic)
(/projecthub/products/buy/42742?
s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

(/projecthub/products/buy/42742?
s=BAhJlhI1Mzc1MixQcm9qZWNOBjoGRUY%3D%0A)

×
(
/
1
P (/proj (/projecthub/product
r ecthu s/buy/42742?
o b/pro s=BAhJlhI1Mzc1MixQc
j ducts m9qZWNOBjoGRUY%
e /buy/ 3D%0A)
c 42742
t ?
h s=BAh
u JlhI1M
b zc1Mi
/ xQcm
P 9qZW
r NOBjo
o GRUY
d %3D%
u OA)
c
t
s
/
b
u
y
/
4
2
7
4
2
?
s
=
B
A
h

Jl
h
l
l
M
z
c
l
M
i
x
Q
c
m
9
q
z
W
N
O
B
j
o
G
R
U
Y
%
3
D
%
O
A
)

×

(



SparkFun FTDI Basic Breakout - 5V

(/projecthub/products/buy/68?
s=BAhJlh1Mzc1MixQcm9qZWNO
BjoGRUY%3D%0A)

(/projecthub/pro
ducts/buy/68?
s=BAhJlhI1Mzc1Mi
xQcm9qZWNOBj
oGRUY%3D%0A)

/ p
 r o
 j e
 c t
 h u
 b /
 p r
 o d
 u c
 t s
 / b
 u y
 / 6
 8 ?
 s =
 B B
 A h
 J l
 h l
 1 M

1



(/proj
ecthu
b/pro
ducts
/buy/
68?
s=BAh
Jlh1M
zc1Mi
xQcm
9qZW
NOBjo
GRUY
%3D%
OA)

(/projecthub/product
s/buy/68?
s=BAhJlhI1Mzc1MixQc
m9qZWNOBjoGRUY%
3D%0A)

Z
C
I
M
i
x
Q
c
m
9
q
Z
W
N
O
B
j
o
G
R
U
Y
%
3
D
%
O
A
)

APPS AND ONLINE SERVICES





Arduino IDE

(<https://www.arduino.cc/en/main/software>)

(<https://www.arduino.cc/en/main/software>)



(<https://www.arduino.cc/en/main/software>)

ABOUT THIS PROJECT



This project continues the work from my previous disk reader project at <https://create.arduino.cc/projecthub/projects/485582/> (<https://create.arduino.cc/projecthub/projects/485582/>)

For more information visit <http://amiga.robsmithdev.co.uk> (<http://amiga.robsmithdev.co.uk/>)

- **My Aim:** To create a simple, cheap and open source way to recover and rewrite data from and to Amiga DD floppy disks from within Windows 10.
- **My Solution:** An Arduino sketch + a Windows application (*which can be ported to other O/S*) that actually works!
- **Why:** To preserve data from these disks for the future. Also, a normal PC can't read/write Amiga disks due to the way they are written.

Writing Data - Attempt 1

So after successfully being able to read disks, I figured if you want to keep the original physical medium, you might want to write disks back again. I figured I'd work this out in reverse, starting with the software (i.e.: converting the ADF disk files into MFM data for the interface to write *somehow*).

So I started by adding classes to read an ADF disk, and encode all the sectors as one track. Knowing I could potentially test the data I created by feeding it back into the decoding part, I started work on this. While working on this I decided to try to find out

what was wrong with my Amiga. After all, I can't test any disks I create if I don't have anything *real* to test them on.

Taking my A500+ apart, I noticed it had suffered one of the most common problems, the clock battery had leaked (<http://members.iinet.net.au/~davem2/overclock/batt.html>) everywhere. So I desoldered this from the board and set about cleaning the board up. Whilst at it, I pulled the entire machine out and set about cleaning up 20 years of dust and grime. I even took the floppy drive apart to clean it.

Whilst cleaning it, I decided it was time to get rid of the yellowing, so I followed the information about RetrObrite (<http://www.retrObright.com/>) and tried it.

I then checked all of the joints on the main motherboard and found a loose connection by the power connector, a few touchups with the soldering iron and as good as new. I waited until I was happy with the RetrObrite process before reassembling the computer.

Meanwhile I continued working on the code for writing disks. I wanted to read the status of the write protect line, but no matter what I set it to it didn't seem to change voltage. So I pulled the drive apart and followed the traces from the little switches that detect the write protect status to a little IC. At this point I guessed that the output is probably only available when you actually want to write data.

After *a lot* of experimentation, I found that you needed to pull the `/WRITE_GATE` pin LOW before spinning up the drive to enable writing. At this point you could obtain the write protect status. I also noticed that while the `/WRITE_GATE` was low the drive didn't switch back off like it used to until that pin had returned to its default HIGH state.

The Amiga would write an entire track in one go. A track in memory is 11×512 bytes (5638 bytes), however, after MFM encoding and putting in correct AmigaDOS format, the track works out as 14848 bytes. Well, there's no way that can fit in the Arduino's 2k of memory, nor its 1k of EEPROM. I needed an alternative method.

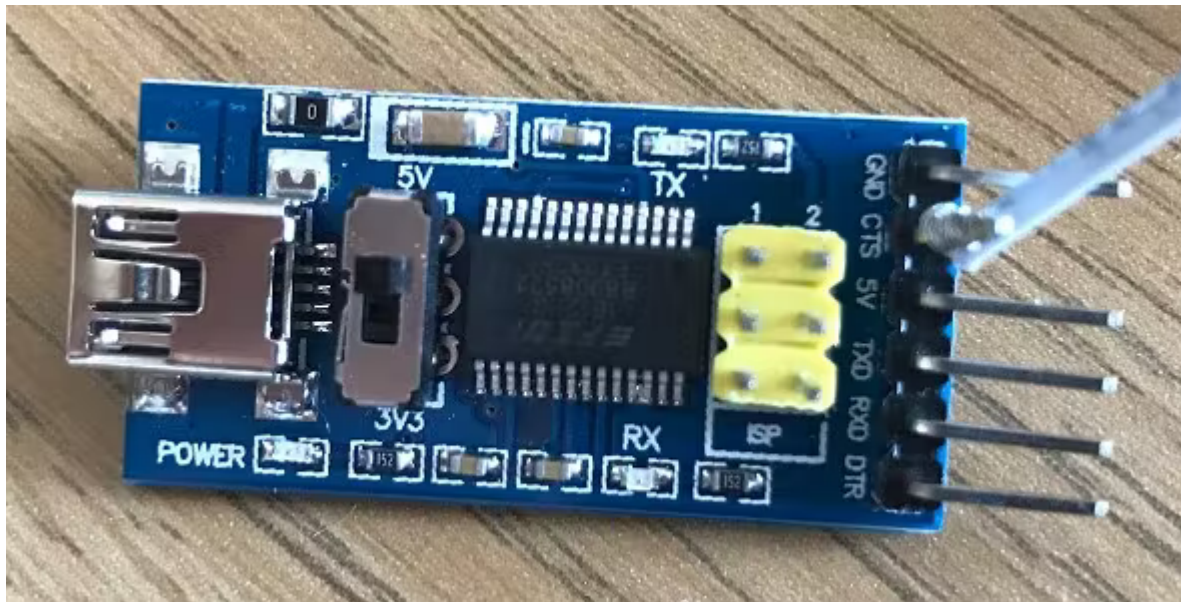
I decided I would try to send the data 1 byte at a time in a high priority thread and wait for a response byte from the Arduino before sending the next. I changed the baud rate to 2M to reduce the lag between characters. This meant that it took roughly 5.5 uSec to send each character, and 5.5 uSec to receive one back. The Arduino would need to write out 8 bits, at 500khz, so it would need a new byte every 16 uSec. So there should be time, assuming the code loop is tight enough and the operating system doesn't delay the sending and receiving too much.

This was a complete, utter failure. The entire read/write cycle took far too long, well beyond one revolution of the disk. The Arduino side was probably fast enough, but the OS wasn't responsive enough. Reading disks works because the OS (Windows in my case)

would buffer the data coming in, but writing, Windows would just send it all in one go, but because the rate I'm sending at is far faster than the Arduino needs it, data would be lost. This was why I decided on this two-way acknowledgement process.

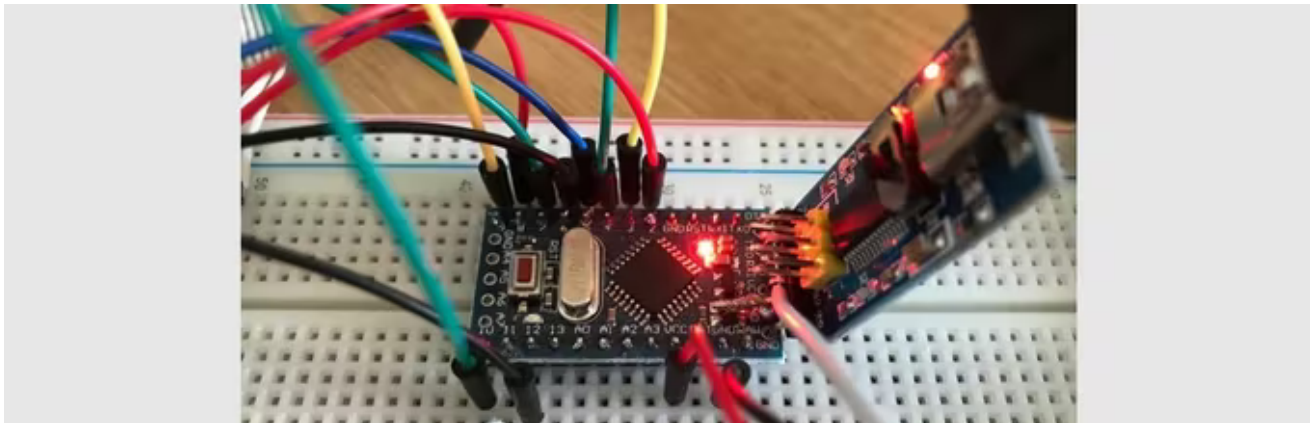
Writing Data - Attempt 2

Software flow control for this application was just not fast enough. I decided to investigate hardware flow control. I noticed on the FTDI breakout board there are CTS and DTR pin. These stand for *Clear To Send* (https://en.wikipedia.org/wiki/RS-232#RTS.2C_CTS.2C_and_RTR) and *Data Terminal Ready* (https://en.wikipedia.org/wiki/Data_Terminal_Ready). I noticed that while the breakout board was connected, the Arduino board connected the CTS to GND.



FTDI breakout board with CTS pin bent out and a wire attached.

I also didn't know which direction these pins were actually in, but after some experimentation, I found the CTS pin could be signaled from the Arduino and used by the PC to control the flow. Normally this is done using a circular buffer, but in my case I couldn't allow this, so I simply set it to '1' when I don't want data, and '0' while I do.



This now meant I could just ask the OS to bulk send the bytes as one chunk, and hope that it was all handled at the kernel level so it wouldn't get interrupted.

I had an inner loop that output each bit from the 8 bits but decided it was probably better timing wise to unravel it into 8 sets of commands instead.

This didn't work. If I allowed the code to run without actually running the disk writing part, then all bytes were received correctly, but with running the code, it didn't and bytes being received were being lost.

I suspected that changing the status of the CTX line didn't instantly stop the flow of data and the computer may still send a character or two. Possibly by the time I had signaled the CTX line, it was already in the process of sending the next character.

Writing Data - Attempt 3

I didn't want to have a serial interrupt as I didn't want any of the writing timings to be distorted. I realised that inbetween writing each bit to the floppy drive there would be a number of CPU cycles sitting in the next while loop. I decided to check between each bit write if another byte had been received since CTX went high and store it.

My theory was that when you raised CTX, the computer was probably already in the middle of transmitting the next byte and as you can't stop it mid-stream, then it would half after this one. This means I only need to check for one extra byte during the loop and use it if found instead of looking at the serial port again.

So this seemed to work, and the Arduino completed the write without losing any data from the computer. The only questions now were: has it actually written any data, and if so, is any of it valid?

At this point I had only encoded one track, so I decided to run the entire algorithm to encode all 80 tracks. Something strange was happening. The drive head wasn't moving at all. It still did when reading, but not when writing.

I found that in order to move the drive head back and forth you first had to raise the /WRITE GATE pin, I suspected this was required for changing the surface also. Once I added code to do this the drive head moved as expected. This did make sense and would prevent accidental writing of tracks while moving the head around.

So at this point I wrote a disk image out I had created previously, and then tried to read it back. Nothing could be detected! Either the data I had written was invalid, or the way I was writing it was wrong.

I decided to feed the encoded MFM sector data that I was creating into my sector decoding algorithm used by the reader to validate that what I was generating was correct and valid, and it was. Something was obviously wrong with how I was writing the data to the disk.

Writing Data - Attempt 4

As no data was being read correctly I decided try a few different approaches. I wasn't sure if the /WRITE DATA pin should be pulsed (and if so, by how long), toggled or just set to the raw data value. My current implementation pulsed the pin. I hadn't been able to find any information online about how the write pin was physically suppose to be manipulated when writing.

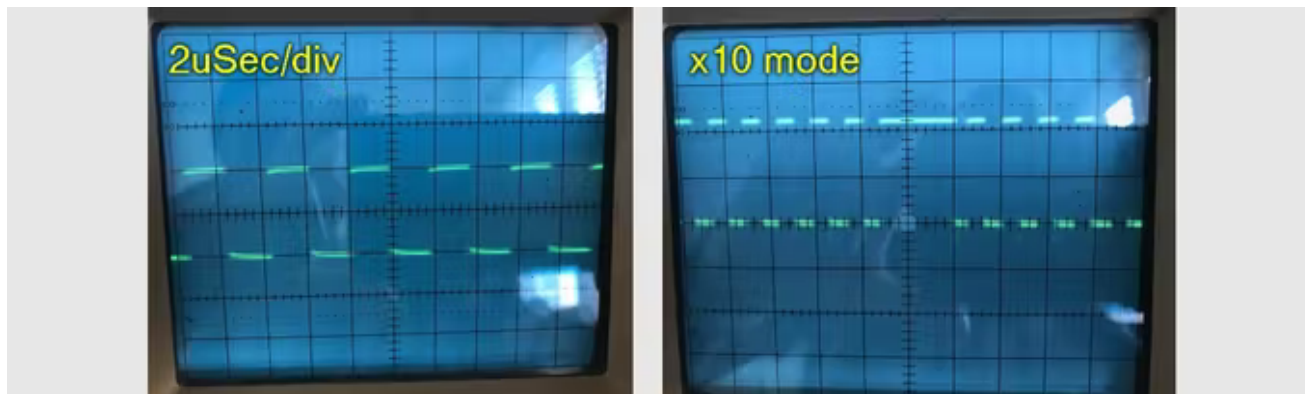
The read head would send us a pulse each time there is a flux reversal. I decided to change the implementation so that WRITE DATA was just set to the value of the bit. That didn't work either. So I changed the code to toggle the current state of the pin. Still no luck.

Clearly one of these approaches must have been the correct one. So I decided to get out the trusty oscilloscope again to have a look at what was going on. I decided to write the MFM pattern 0xAA to every byte on a track continuously. 0xAA in binary is B10101010, so this would give me a perfect square wave that I could monitor for the required frequency.

If it didn't see a perfect square wave at the desired frequency, then I knew there must be some kind of timing issue.

I hooked up the scope, but was surprised to see the timings *were* perfect. However, being an old scope I couldn't see more than a few pulses. The scope had this wonderful x10 "mag" mode. When pressed, it increased the timebase by 10, but more importantly allowed you to scroll through all of the data much like on a modern digital scope.

Something wasn't correct here. It looked like every 12 bits or so I ended up with a period of just "high".



Either the data I was sending was in some way invalid, or there was something causing a pause in the writing process every 12 bits or so. 12 being a strange number considering there are only 8 bits in a byte.

After thinking about this, I wondered if I was back with a flow control issue. The way I had designed the loop was to scoop up any stray extra bytes that were received after we had waited for one. But it wasn't intelligent enough to prevent the wait every other byte. I had two choices, move *something* into an interrupt, or patch the loop.

I decided to have a go at correcting the way the loop worked first. The issue was as a result of a delay caused by waiting for the next byte from the computer. If we lowered CTX and waited for a byte, by the time we raised CTX again, another byte was already on the way.

I change the loop so that when the second byte received had been used, the Arduino momentarily pulled CTS low and then high again to allow another character to be sent. This meant on the next loop we would have already received the next byte so no waiting was required.

Testing this produced a perfect square wave:



This meant all of the timing for writing a track was perfect, it was just down to the *actual* data that was being written. I decided to let this run for a few tracks and sides, and then read it back to see if it had written correctly. I was setting the `/WRITE_DATA` pin to the corresponding bit value from the data received.

When reading the data back it looked like nothing had been encoded, but then I skipped to the other side of the disk. Sure enough there was my pattern. I didn't know why it had only written to one side of the disk.

After some thinking I started to wonder if the `/WRITE GATE` pin didn't actually work the way I thought it did. It occurred to be that by pulling the pin low it may be enabling the erase head on the drive. If this was the case, then I should only do this when I was actually writing or I might end up with noise on the disk as it spins and erases.

I changed all of the code so that the `/WRITE GATE` was only used when first starting the drive, and later only literally during the write loop. That worked! I was now writing data to both sides of the disk!

So I tried again with a real ADF disk image and let it complete. I then used the reader portion to see if I could read it back. It worked! But for some reason it took quite some time to read this disk back. I wasn't getting any MFM errors but it was struggling to find all of the sectors.

There's were two possibilities for me to look at now: firstly, had the data actually written timely enough; and secondly, would the disk actually work in a real Amiga?

Too excited with the idea that I might have actually written a disk I booted up the *now working* A500+ and put the disk in. Moments later, the disk started booted and then displayed the famous checksum error message. So I was writing *something* valid, but it wasn't consistent.

I decided that unless I could read the data back at a much more accurate rate, writing a disk was pointless.

Reading Data (again)

I wanted to improve the reading quality as I wasn't happy with the current implementation. The current implementation didn't allow enough flexibility for the pulses to arrive at slightly odd times. I needed a new approach.

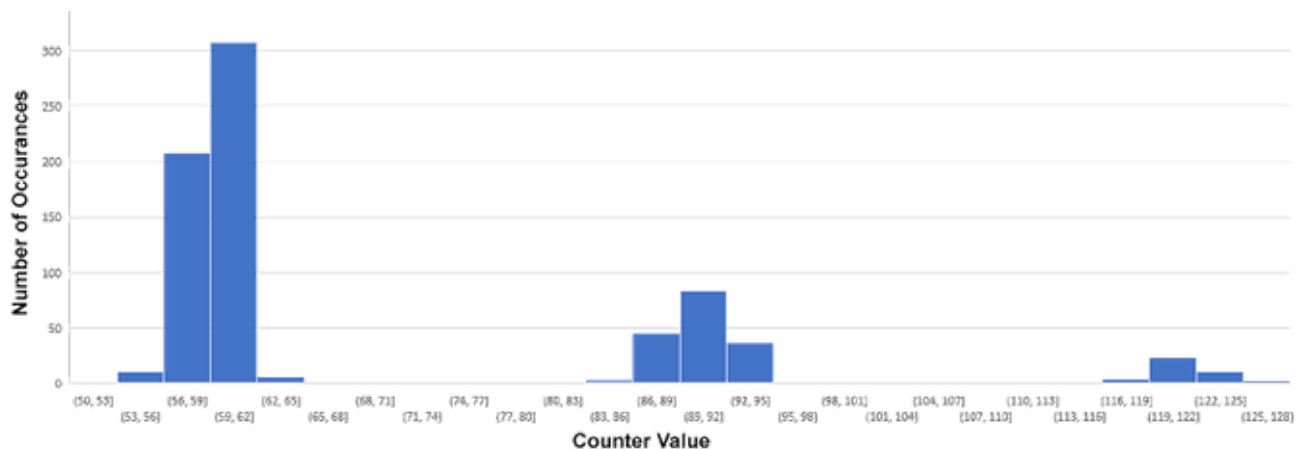
Firstly, I decided I was going to sync the reading to the /INDEX pulse. It's not required by the Amiga but may come in handy later on for me testing, writing and reading.

Several people in the comments to the first half of this project suggested that I should be recording the timing between pulses rather than the method that I had implemented. The only issue with this was getting this data to the PC fast enough. If I was to send a byte for each bit, then I could easily exceed the maximum 2M baud.

I decided that the best thing to do would be to try to make sense of the data a little. So I decided to let the counter I was using originally to free-run, right up to 255. I then put the code in a loop waiting for a pulse and that this point saw how much time had passed.

In an ideal situation, the lowest possible minimum value would be 32 (corresponding to 2 uSec). With MFM you could only ever have a maximum of three 0's in a row, so the maximum this value should reach was 128. This meant there were a maximum of 4 possible combinations in a row.

I sampled several disks to see where the majority of these frequencies lay, and the results can be seen below:



Looking at this, I find the majority of the points around a counter of 52, 89 and 120. However, these were somewhat specific to my drive and therefore not a good guideline. After some experimentation, I used the following formula: $value = (COUNTER - 16) / 32$. When clipped between 0 and 3 this gave me the output I required. Every 4 of these and I could write a byte out.

It occurred to me that because you couldn't have two '1's together in an MFM encoded bit stream I could safely assume anything for the first value was invalid and could be treated as another '01' sequence. The next part was to unpack this data once received by the PC and turn it back into MFM data. This was simple, since 00 couldn't happen, a 01 meant write '01', an 10 meant write '001' and a 11 meant write '0001'. I gave this a try and to my surprise my results were 100% successful. I tried with a few more disks too, 100%! I now had a very reliable disk reader.

With this new approach being a lot more tolerant on the data from the disk I no longer needed any phase analysis or as many retries. Most of my disks now read perfectly. Some required a few retries but got there in the end. The last part was to statistically analyse the data and see if it could be repaired, however, 99% of the time bad data coming in was completely unrecognizable and so was little help.

Writing Data - Attempt 5

Now that I could verify what I had written with high accuracy it meant testing the writer would be much easier.

I set about analysing the code to see what was going wrong. I wrote a 0x55 sequence to an entire track and then read it back in. From time to time a bit had shifted in the data coming back, meaning there was some kind of timing issue in writing.

It turned out that this was partly due to the way I was handling the serial port, and partly due to the use of the timer. I was waiting for the timer to reach the value 32, writing the bit, and then resetting it. I changed it so I didn't have to modify the timer counter value.

I would write the first bit when the counter reached 16, then the next when it reached 48 (16+32), and the next when it reached 80 (16+32+32) and so on. Timer2 being only 8-bit rolls over back to zero after the 8th bit, exactly when we needed it to. This meant that as long as we wrote the bit at the required timer value we would be at exactly 500kbps.

I also looked at how I was reading the data from the serial port. This was being read inbetween each bit, but this needed to be as short as possible too. After a little experimentation I achieved the shortest working block.

After modifying the Windows code to support verify, I was now ready to try again. This time I knew that if the disk verified properly, then it should work properly in the Amiga.

So I tried writing another disk out. With verify it took longer. With the new algorithm about 95% of the tracks passed verification on the first go, with only the remaining 5% having to be re-written once more. I was happy with this and popped the disk into the Amiga. It worked perfectly!

Writing Data - Attempt 6

After some feedback from some people who have been using this it was clear that even with verify on the drive wasn't always producing fully readable disks. The software could read them back perfectly, but the Amiga computers would report of a few checksum errors here and there.

I had another look at the code, wondered if it was a timing issue and looked to see if it could be made to be interrupt driven, but sadly with the small amount of time between each bit there simply isn't enough time with interrupts to achieve this with preserving the registers you modify etc.

I then looked back at the writing code. There is a small chance that after a full byte has written, the code could have looped back to start writing the next byte before the timer had overflowed back to 0, allowing the first bit to be written early.

I added a small loop to ensure this couldn't happen which hopefully will fix this for anyone having this issue.

Writing Data - Attempt 7

After getting a lot of reports of of checksum errors for written disks I started to investigate. I thought at first I was going to have to get down to looking at the MFM data from the disk but the problem was actually much simpler

Looking at XCopy Pro to see the checksum errors, it reported codes 4 and 6 meaning checksum errors in the sector headers and data areas. If it had just been the data area then I would have assumed that it was purely something to do with writing the last few bits of the track, but it wasn't.

I started looking at the writing code and the padding I had around each track, wondering if I was overwriting the start of a track now and then, so I massively reduced the post-track padding from 256 bytes to 8. To my surprise my verify then kicked out a tonne of errors.

This made me wonder if the actual issue is I'm not writing enough data. I set about adding a Track Erase command to the Arduino which would write the 0xAA pattern to the entire track and then write my track afterwards. To my surprise XCopy gave it a 100% thumbs up. So hopefully thats cleared that problem up.

Diagnostics

I have had lots of feedback from people who have successfully made this project, both fully working and not working. I decided I would build a diagnostics module into the code to help anyone who can't get theirs to work.

The diagnostics option consists of a few extra commands for the Arduino to process as well as a whole series of events that get ran through to ensure everything is wired correctly.

So What's Next?

The entire project free and open source under GNU General Public Licence V3 (<http://amiga.robsmithdev.co.uk/licence>). If we want to have any hope of preserving the Amiga, then we shouldn't be ripping each other off for the privilege. And besides, I want to give back to the best platform I ever worked on. I'm also hoping people will develop this and take it further and keep sharing.

The current writing solution isn't an option on the Arduino UNO unless you use a separate FTDI/serial breakout board, so my next tasks are to make it work on that (possibly using the 23K256 IC to buffer the track before writing it to the disk).

I still want to look at other formats. ADF files are good, but they only work for AmigaDOS formatted disks. There are lots of titles with custom copy protection and non-standard sector formats that simply cannot be supported by this format. I have received some very useful information on this but don't currently have many disks to test with.

According to Wikipedia, there's another disk file format, the FDI (https://en.wikipedia.org/wiki/Amiga_Disk_File#FDI) format. A universal format, that's well documented. The advantage of this format is it tries to store the track data as close to the original as possible so hopefully will fix the above issues!

I also came across the Software Preservation Society (<http://www.softpres.org/>), specifically CAPS (formally the *Classic Amiga Preservation Society*) and their IPF (https://en.wikipedia.org/wiki/Amiga_Disk_File#IPF) format. After a little bit of reading, I was very disappointed; it's all closed, and it felt like they were just using this format to sell their disk reading hardware.

So my focus will be on the FDI (https://en.wikipedia.org/wiki/Amiga_Disk_File#FDI) format. My only concern here is with data integrity. There won't be any checksums for me to check against to see if the read was valid, but I have a few ideas to resolve that!

CODE



Sketch and Windows Application Source

Arduino Sketch, and example Windows application source code

172 (https://github.com/RobSmithDev/ArduinoFloppyDiskReader/watchers)

42 (https://github.com/RobSmithDev/ArduinoFloppyDiskReader/forks)

RobSmithDev (https://github.com/RobSmithDev) / **ArduinoFloppyDiskReader**
(https://github.com/RobSmithDev/ArduinoFloppyDiskReader)

Arduino Amiga Floppy Disk Reader/Writer - Hardware and software interface for accessing Amiga disks (read/write ADF and SCP) on non-Amiga hardware — Read More
(https://github.com/RobSmithDev/ArduinoFloppyDiskReader#readme)
https://amiga.robsmithdev.co.uk (https://amiga.robsmithdev.co.uk)

Latest commit to the master branch 2 days ago (https://github.com/RobSmithDev/ArduinoFloppyDiskReader/zipball/master)

Download as zip (0.00 MB)

SCHEMATICS

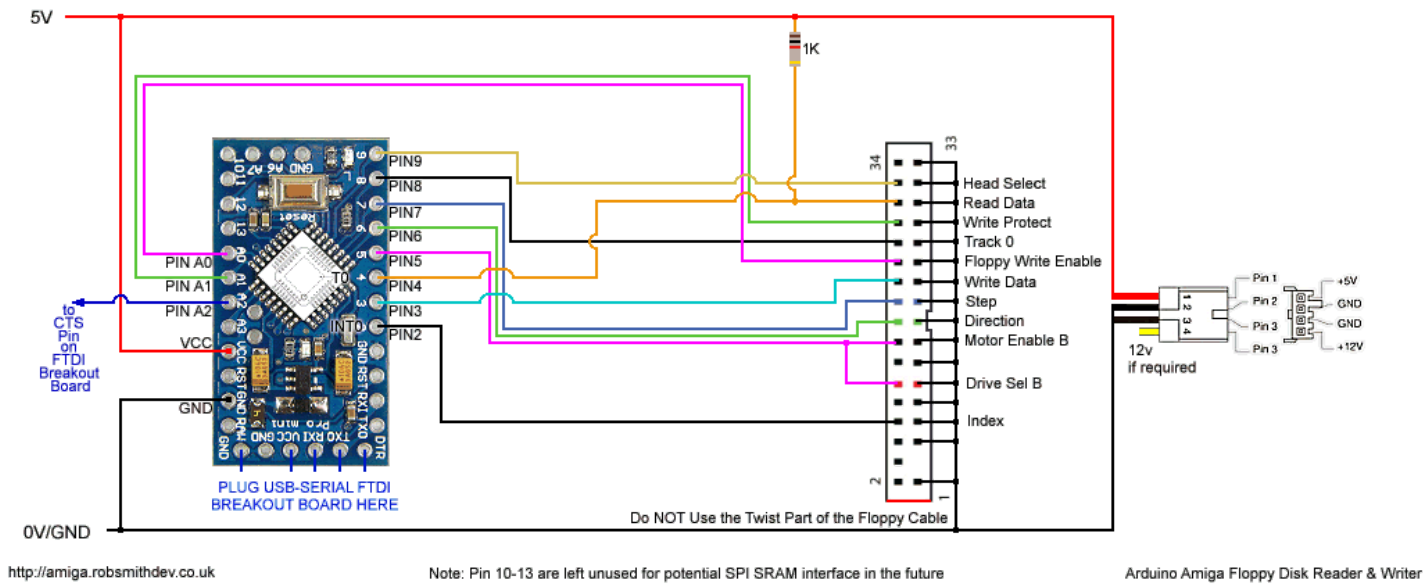


Circuit for Arduino

DOWNLOAD (HTTPS://HACKSTERIO.S3.AMAZONAWS.COM/UPLOADS/ATTACHMENTS/351218/CIRCUIT_FEZE)

r
o
M
i
n
i

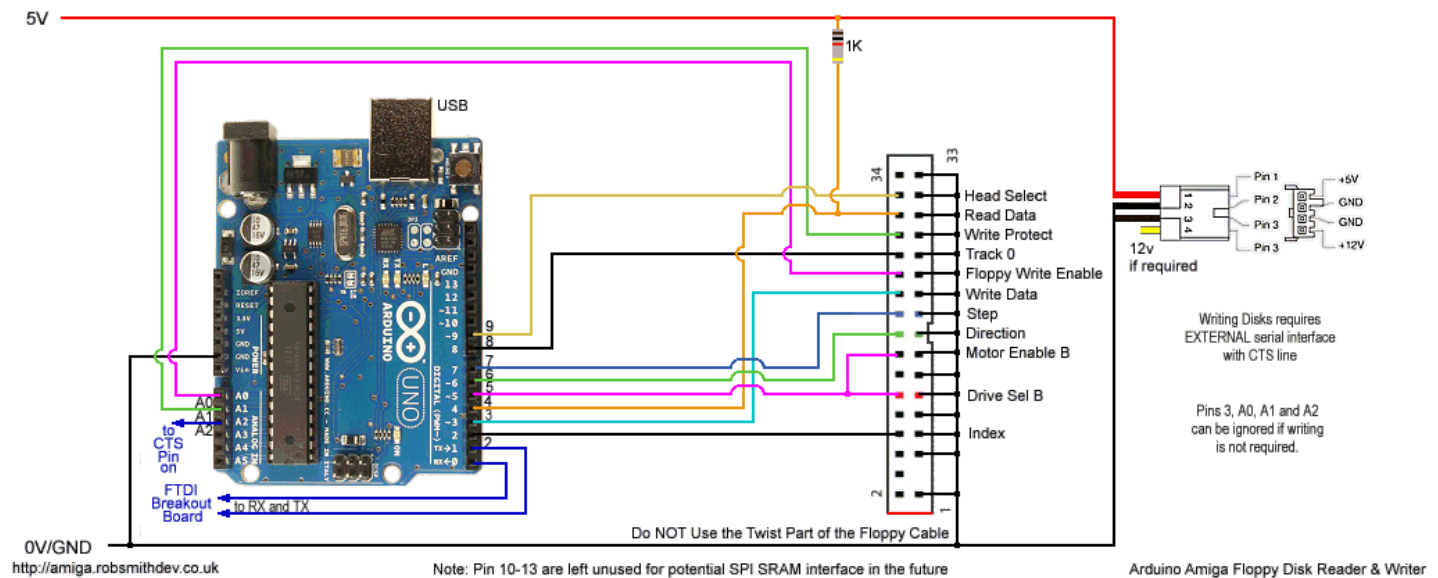
Circuit for Arduino Pro Mini



C ir c u it f o r A r d u i n o U DOWNLOAD (HTTPS://HACKSTERIO.S3.AMAZONAWS.COM/UPLOADS/ATTACHMENTS/351219/CIRCUIT2_O4

N
O

Circuit for Arduino UNO



COMMENTS

Please log in (/projecthub/users/sign_in?

id=53752&m=project&reason=comment&redirect_to=%2Fprojecthub%2FRobSmithDev%2Farduino-amiga-floppy-disk-reader-writer-v2-2-239c97%23comments) or sign up (/projecthub/users/sign_up?

id=53752&m=project&reason=comment&redirect_to=%2Fprojecthub%2FRobSmithDev%2Farduino-amiga-floppy-disk-reader-writer-v2-2-239c97%23comments&source=popup) to comment.



Daniel Bergman (/projecthub/daniel-bergman)

4 years ago

(/pr
ject
hub/
daniel-
berg
I've been following the initial project implementing disk reading. Today I was searching for that to print the instructions and try to build it later this week and found out about your progress in writing too.

My aim is to find an easier way to preserve different kinds of floppy formats, including Amiga and the easier msdos format, without having to fill my workplace with old computer gear. This could potentially be extended to be a cheap diy kryoflux or catweasel alternative.

Amazing work! Kudos to you! 😊
I'm gonna give it a go this weekend.



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

(/projecthub/Alelectronic)
Rob did an excellent job bij creating this easy (and cheap) Amiga disk reader for PC computers! I have built this Amiga mini Pro adapter twice today to make sure that I made no mistake because it still won't write disks. Reading disk works splendidly though. Maybe I used an old beta .ino version or my version AmigaReader is not up-to-date? Got still no clue why because my floppydrive(s, -I tested 5 different drives that all were okay-) worked perfectly on my other ADFcopy and Amiga Xcopy floppy tot SD card copier? Hope there will be a solution for this strange not writing problem soon.



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

(/projecthub/Alelectronic)
Rob, yesterday I started to examine the not writing problem with my cheap new Saleae clone 24MHz 8 channel Logic Analyzer and the fantastic PulseView sampling software that is free on the SigRok site. You do not need an oscilloscope at all to view the Motor_on, CTS, Index, Step, Write enable, Write data, Head select and Read data signals all simultaneous! You can view them all at the same time in a time diagram with these cheap (8 euro) Saleae (also working with their software!) compatible Logic Analyzers. The diagram showed that Writing to disk with verify on, that just stopped every time after a few seconds, had a briefly inactive Motor_on signal for about 375 ms. And all signals, except CTS and Motor_on, start with being a logic '1'. Only the Index signal shows 3 little logic '0' pulses (2 ms) just before the Motor_on signal starts going inactive for about the mentioned 375 ms. And just before the Motor_on pulse starts being a '1' pulse the Head Select signal becomes a logic '0' signal for about 1370 ms. And only after that pulse is '1' again the Write enable and Write data signal very briefly pulse to (2ms?) logic '0'. And at the same time CTS briefly pulses a logic '1' (also about 2 ms).

Almost all that time there are a lot of data pulses on the Read Data line. And after about 4550 ms and also 16 more Index '0' pulses (2 ms) the Motor_on signal is a logic '1' again. And Read data stops. And what it shows is that the STEP signal never gives any signal like it did when it successfully was Reading a disk. It always just stays being a logic '1'. I can send you the Wave diagram if you like?



RobSmithDev (/projecthub/RobSmithDev)

4 years ago

You'd have to follow the logic of the program to verify the various logic levels and validate them, but given your "non verified" disks are completely unreadable I still think the issue is with the CTS flow control. Without verify about 90% of the tracks will write correctly first time, all verify does is to read the track after each write, and if it doesn't match the original it rewrites it.

If writing fails completely then something in the interface is either returning an error, or there is an I/O timeout as a result of failed CTS hardware handshaking. The CTS line is used to make the PC "drip feed" the data to it as required rather than it all rushing in all at once as the Arduino doesn't have enough ram to store the entire track in memory.

The write process is as follows:

The PC sends:

Send "Enable Write" command

Send "Rewind to Track 0" command

Send "Switch to Upper Side" command

Then in a loop for each track:

1. Send "select track" command
2. Send "select surface" command
3. Send "write Track" command with data
4. If Verify send "Read track" command and compare. If fails goto 3
5. Go to next surface/track as required and goto 1

any of the above commands can report an error, and it would be easier to get the code compiling on the PC to see which reason it breaks out.

I am planning on an upgrade to the software with a "hardware diagnostics" option to help debug this issue, but so far out of all the people reporting to have built this I have only see this issue twice (and one turned out to be a faulty wiring).

The above, whilst thorough doesn't help me to diagnose what the issue is. The best thing you can do is to get the code downloaded and running in a debugger such as Visual Studio (which is what it was developed in) placing breakpoints throughout the code where errors are reported.



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

(/pr

ject
ADDF
Ale
tron

Thanks for your quick reply Rob. I tried to answer on the site but couldn't so I reply your e-mail.

You write that Writing to disk without Verify_on works 90% of the tracks, but sadly it NEVER does. Not with and not without Verify on. And I guess that you never used a logic analyzer to debug your project?

The only thing that works is the protect notch on the Disk check to preserve the original data on disk. And of course reading from disk. I will enclose that Write protect screenshot. Which worked with Writing Verify On and also Off. But sadly also immediately afterwards followed by a communication error.

Maybe I should use your Github sourcefile by making a working file but I do not work with Visual Studio. I only used your 1.96 MB file that probably is bugged. I also build Nick's ADFcopy that perfectly reads AND writes to Amiga disk (and never fails!). I will try to sample those communication signals too to analyze where your program differs and apparently fails.

I was so happy Nick's interface worked that I contributed him a small donation. Writing with Nick's interface only failed previously because the Shugart cable was with about 15 cm of length too long to work. It now works splendidly after I shorted that cable to about 3 cm. But because you said that shouldn't matter with your project because you used 30 cm long flatcable without any problem I kept my 15 cm long flatcable.

I do not quite understand your explanation and found your 7 write attempts far from helpful to say the least. In this case I think only Logic Analyzer Wave diagrams would shed some light in this matter. And that is also what Manufacturers use in their specification guides. I own 4 oscilloscopes of which 2 are High frequency Tektronics scopes but none will help in a case like this where we need to examine at least 8 signals in time.

Regards.

Albert.

**RobSmithDev (/projecthub/RobSmithDev)**

4 years ago

(/pr
Object
Hub/
Rob
Smit
hDe
v)
As I said you have an issue with your CTS line. Weather it's in the breakout board or your wiring, you have an issue there.

Rob
Smit
hDe
v)

**Albert van Bemmelen (/projecthub/Alectronic)**

4 years ago

(/pr
Object
Hub/
Alec
tron
ic)
The CTS line hardly is doing anything. It maybe only activates ones in a while for a very brief 2ms. So it is not programmed to do more. And also Write enable and Write data only popup during that same 2ms. And both are logic '1' the rest of the time as in Read mode.

By-the-way: You can check this in the Wavediagrams. Legenda D0 to D7:
Motor_on, CTS, Index, Step, Write enable, Write data, Head sel, Read data.
I will check the timing in the wave diagrams with those of the my working ADFcopy interface.

Thanks,
Albert.

**RobSmithDev (/projecthub/RobSmithDev)**

4 years ago

(/pr
Object
Hub/
Rob
Smit
hDe
v)
Please understand the CTS line is CRITICAL to this working, every byte that is received during writing a disk will be triggered using the CTS line. If it is not functioning correctly a disk will not be written. The size of the pulses from the Arduino for CTS will be less than 4uSec I would have thought, so you won't see them at 2ms resolution.

**Albert van Bemmelen (/projecthub/Alectronic)**

4 years ago

(/pr
Object
Hub/
Alec
tron
ic)
I understand that it must be something critical why the program keeps breaking with a communication error. I therefore also tested the interface on my quad-core PC with USB 3.0 interface but still the same problem. I however see the CTS

works perfectly with my Saleae Logic Analyzer Rob. There only is one CTS pulse visible of about 2 ms which becomes active '1' just after Head select returns to '1' high state during writing in the about 2378 millisecond. And at the same time Write enable and Write data turn low for about 2 ms. After that those 3 signals do not change anymore and Motor_on deactivates back to '1' after about 4750 ms. And the Writing with Verify On failed. None of those 3 pulses show up during reading of course. I don't think both my new FTDI FT232RL are the problem, and neither both also brandnew Arduino Pro Mini Boards. If I have time in the next few days I will check the communication protocol of all signals on my ADFcopy interface and I will record those correct timing signals too! Thanks!



jonni (/projecthub/jonni)

4 years ago

Writing works for me just fine and I've already built 10+ boards. And I'm even using the precompiled windows binaries for the reader client.

What version of Arduino IDE (and have you double checked that IDE has right board in settings) are you using to compile and upload the .ino file to Arduino Pro? And have you made sure that there are no other devices trying to use the same com port at the some time when running the client. Like if I forget to close Arduino IDE, then reader client fails with communication errors.

One suggestion would be to upload pics of your actual build somewhere and maybe we can see problematic connection somewhere (even I mixed couple of traces when building the 1st time). Or something silly like twisted floppy cable etc.



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

Thank you jonni for your appreciated confirming on the writing disks part. I already have checked all signal connections multiple times and used the same boards as Rob showed in his article (and also made it twice with all brandnew boards!) . Writing is still just useless. With Verify Off it only produces very quick useless disks from perfect .adf images. I've send Rob some photos of my device(s) and also a lot of screenshots of the signals checked with my logic analyzers. It clearly showed why it could not work. I also compared these with screenshots of the signal lines I took on Nick's ADFcopy device. They were completely different. Nick's project uses

a more expensive Teensy 3.2 board but also writes disks perfectly (have to use a very short FDD cable though because of the 3.3V levels that have to simulate the 5V FDD signal lines).

I yesterday bought the RS232 USB adapter boards on eBay you successfully had used in your PCB project. I hope that adapter will finally solve the writing useless Amiga disk problem with Verify Off, and NOT writing and breaking up with a communication error when Verify ON is selected.



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

(/pr

Today 14 March I received the long awaited previously by jonni mentioned CH340 FTDI RS232 to USB board. Which I of course immediately tested with the V2.0 version software. Still the same problem! Reading disks works perfectly. Writing disks breaks of with a communication error every single time.

Sorry but I think I give up on the writing disks part because it is just never going to work and it seems Fake?

Or the issue is solved in new version 2.1 which I haven't tried yet?



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

(/pr

I noticed Rob's update ArduinoFloppyReaderWin.exe v2.1 file and just tested it by pressing the Diagnostics button. Very impressive add-on that Rob managed to create!

Sadly Writing a disk failed again and ended in an error saying: DIAGNOSTICS FAILED: An unknown response was (2x was in sentence) received from the Arduino while executing the WriteTrack command. So the not known response likely still is a wrong timed programming error? I can't share the screenshot I took here why I described the Diagnostics error message that was given.



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

(/pr

oject

Today I also made Rob's Arduino UNO interface version to make sure it were not the Mini Pro Boards that caused the Floppydisk writing problem in both 2 and 2.1 versions. It didn't make any difference sadly. Same story same problem. So I wonder if others indeed were able to fix or escape the not writing disks issue, could it maybe still have something to do with the length of the Floppy shugart cable? Because I already tried all different FTDI232RL to USB boards and also another Mini Pro Board which didn't make any difference. According to Rob who used an original 30 cm long cable my 15 cm original cable should not be any problem. For now I stop my experiments. Maybe a new update version will fix this not writing disks issue.



Albert van Bemmelen (/projecthub/Alelectronic)

4 years ago

Very good news! After endless testing with 3 previous useless and all different FTDI RS232 to USB boards (including the one that Rob proposed and used in the article!) it finally works!! Conclusion: Only jonni's little eBay adapters, that I recently got, do work!! So if anyone has the same disk writing problems he/or she better only buys the right FTDI RS232RL adapter board. (Mine had the text: FTDI 1221-C GO030631 FT232RL). All others failed after weeks of trying! Don't know why but apparently only the ones ordered on eBay work (the not or too slow working ones came from Aliexpress)! Thanks jonni for the great tip. And indeed a Great job Rob! Sorry for you having to go through all the trouble to add the Diagnostic part into the program. Which by-the-way worked splendidly but previously only reported the errors created by the 3 useless RS232 UART USB boards. Question: I noticed the little housing you created which looks very compact. Does it have enough room for the Floppy drive, the Mini Pro Arduino module and the FTDI RS232 to USB interface? And can I please order one? (through PayPal or Ideal?) Thanks!



Mariusz Morawiec (/projecthub/mariuszmorawiec1)

3 years ago

I have the same troubles while floppy writing. Unfortunately the bath and LOT numbers of FTDI chip you gave are not helpful. Could You please give some more details about board you successfully used for floppy writing?

mor

awie

**Albert van Bemmelen (/projecthub/Alectronic)**

3 years ago

(/pr

Hi Mariusz, you used the same wrong FTDI chip adapter I had used that failed working.

You need this one from seller Deeplearnings:

[https://www.ebay.nl/itm/FT232RL-USB-To-Serial-Adapter-Module-USB-TO-RS232-Max232-Download-For-Arduino-dp-/253065338990?](https://www.ebay.nl/itm/FT232RL-USB-To-Serial-Adapter-Module-USB-TO-RS232-Max232-Download-For-Arduino-dp-/253065338990?hash=item3aebdea06e&_uhb=1)

[hash=item3aebdea06e&_uhb=1 \(https://www.ebay.nl/itm/FT232RL-USB-To-Serial-Adapter-Module-USB-TO-RS232-Max232-Download-For-Arduino-dp-/253065338990?hash=item3aebdea06e&_uhb=1\)](https://www.ebay.nl/itm/FT232RL-USB-To-Serial-Adapter-Module-USB-TO-RS232-Max232-Download-For-Arduino-dp-/253065338990?hash=item3aebdea06e&_uhb=1)

(;) Albert.

**Albert van Bemmelen (/projecthub/Alectronic)**

3 years ago

(/pr

Keep in mind Mariusz, that the FTDI board that Rob showed in his instructions probably also had problems with correctly writing Amiga disks. As you and I already found out the hard way after many tests with probably that same FTDI board. That board in tests only managed to successfully read Floppies. Only the eBay board from deeplearnings worked for me (and that the right eBay boards work was also previously mentioned by Jonni). All other boards must be fake somehow or just can't keep up with the speed that is needed for the correct timing for writing images to real Amiga Disks. Rob's great AFR project will definitely work when the right ftdi usb to serial adapter is used!

**Phil Mundy (/projecthub/ppmundy)**

4 years ago

(/pr

Nice work Rob. Today I started to build a PIC24 floppy reader to try and recover data from a bunch of commodore PET, BBC and Amstrad 1512 disks from my attic life archive pp 5 1/4 and 3.5" diskettes. So I've reached the point of streaming data events and index pps and we'll see where we get to.... I also have a few wd2797s but I get the feeling I'll be able to decode all this on the fly for SD and DD formats....

It's pretty impressive what you can do with a little MCU these days, just wish I could go back to the 80's with it 😊

**Wattyka (/projecthub/wattyka)**

4 years ago

(/pr

ject

hub/

watt

yka)

I can

confirm

that

disk

reading

works

perfectly

fine

with

the

common

chinese

CH340G

based

Arduino

Nano

clone.

You

just

have

to

solder

a

wire

to

the

unconnected

CTS

pin

9

of

the

CH340G

and

connect

it

to

A2

to

pass

the

diagnostics.

Although

I

used

I can confirm that disk reading works perfectly fine with the common chinese CH340G based Arduino Nano clone. You just have to solder a wire to the unconnected CTS pin 9 of the CH340G and connect it to A2 to pass the diagnostics.

Although I used a 5V 2A power supply, the voltage would seriously break down when the stepper motor turned on, causing all sorts of random failures. A beefy electrolytic cap close to the drive fixed that.

Writing fails with "An unknown response was received from the Arduino while executing the WriteTrack command.". The disk track is unreadable once I've tried to write though, so it did at least write something. It turns out that the "unknown response" 'X' is infact a known error code for "buffer underflow", meaning that for some reason the PC/CH340 wasn't able to provide the data in time although the CTS mechanism basically works.

Anyway: You've shown some extraordinary persistence in getting something to work that everyone thought was impossible to do with only 16 Mhz and 2K RAM. Great job!

**hugo_nl (/projecthub/hugo_nl)**

4 years ago

(/pr

ject

hub/

hug

o_nl

I

wonder

if

this

all

will

work,

including

write

support,

to

connect

an

original

Amiga

500

internal

disk-drive

to

Rob, thanks for all the effort in researching this and writing all this up!

I wonder if this all will work, including write support, to connect an original Amiga 500 internal disk-drive to the PC?

I read that, while working on v1, you connected the original drive to confirm something was not working correctly, so that implies it could. But then I also read something about /INDEX not being used on the Amiga, but I do see it connected in the schematics and I'm curious about its requirement. Thanks!

**RobSmithDev (/projecthub/RobSmithDev)**

4 years ago

(/pr

ject

hub/

Rob

Smith

Dev)

Hi

I don't think it's currently being used, some games I think used it for copy protection but not sure. I want to use it later on for sync while reading non dos

~~Risks~~~~Regards~~~~Rob~~ Smith

v)

**hugo_nl (/projecthub/hugo_nl)**

4 years ago

(/pr

ject

500,

an

hug

Anyw

\

Amiga

wondered

if

you

could

point

me

in

the

right

direction:

- MOTOR_ENABLE_B (PIN 5) that would make sense to be 16, MTR0D but I'm unsure.
- HEAD_SELECT (PIN 9) that I suspect is either 32, SIDEB or 2, CHNG.

I think I confidently paired the other ones up like this:

Internal PIN header => Arduino Reader/Writer)

8 INDEX => INDEX PIN 2

10 SELOB => DRIVE_SEL_B PIN 5

18 DIRB => DIRECTION PIN 6

20 STEPB => STEP PIN 7

22 DKWDB => WRITE_DATA PIN 3

24 DKWEB => WRITE_ENABLE PIN A0

26 TKO => TRACK_0 PIN 8

28 WPRO => WRITE_PROTECT PIN A1

30 DKRD => READ_DATA PIN 4

Thanks in advance!

(Edited to add) that I'm sorry for the dumb questions 😊 I'm a software person by trade, not a hardware guy. I have taken my Amiga 500 apart and noticing the drive has an identical connector as 'normal' disk drives. Searching a bit more suggests the pin-out is identical.

I pulled a floppy drive from an ancient server and am getting unstable, random results. I wonder if that is the drive (cleaning did not help), or the Arduino board (it's not a genuine UNO).

I managed to wire up the FTDI to TX/RX of a new Arduino Mega2560 and (after changing the code for the different pin-out), am getting much more stable results now. I never seen the progress bar reach 1%, and it's now over 50%. Looking good! (I'm using the command line tool by John Tsiombikas as I my host computer is Linux.)

Thanks again for your work!

(BTW, the Arduino Mega also has 8K of SRAM which could perhaps be utilised to cache an entire track.)



S_Tintillier (/projecthub/S_Tintillier)

4 years ago

(/pr
Great job!

As a former Amiga developer I'm surprised you had no problems with "precomp" when writing!

In fact each bit you create on a floppy is like a magnet and when you write "00" they repel each other, so you must write them a bit faster than when you write "01" or "10" (they attract each other).

By the way, one of most used copy protection was writing faster (and more than 11 sectors on a track). A normal disk drive can't do that, but thanks to the included PLL you could read those tracks.

I think I will give it a try, and get my A1000 back to life 😊



lamerjack (/projecthub/lamerjack)

a year ago

(/pr
Hi. Use my Amiga 500 as keyboard for pc.

So now replaced the A500 drive with a pc floppy and i use your wonderfull project to read my old amiga disks.

I have 2 question.

1. Everything works fine... except if i use hd floppyes (formatted a low density) these disks are not readable i have to close the "HD window" with tape to read it. (if i try to write the floppy without tape the verify fail)
2. i bought from aliexpress a usb 2 floppy adapter to connect a old floppy to a new computer.

The idea is connect both to floppy 34 pin interface. And then connect only one usb

cable at time.

Before to do it i checket the resistance of every pin of the adapter with a tester...

And i saw that some pin are connected together:

On your adapter pin 16 is connected with 12.

But on the usb adapter pin 16 is alone and 14 12 10 and 6

On the adapter PIN 30 (read data) is connected to pin 22 (write data)

On your project pin 30 is indipendent to pin 22.

Any suggestion to connect both?

I ordered another arduino and ftdi adapter.

I want to modify sketch to use same pin for read and write.

If it works connect with the usb to floppy adapter. and here put a 1k resistor between pin 16 and 12 instead connect togheter.

It make sense?



mlorenzati (/projecthub/mlorenzati)

a year ago

(/pr

ject

hub/

mlo

renz

ati)

I'm

able

to

com

ple

te

the

tes

t

and

w

ri

th

er

it

mo

st

ce

l

y

o

u

g

I built the circuit with just an arduino nano and connecting the CTS to the USB to serial

chip. I can share my PCB

It appears that the timing issues are more visible when trying to access and write track

zero

I'm able to complete the test and write with verify, but then If I read it most certainly

the track zero will have reading errors.

Any idea if there is some possible consideration for track 0? I understand the FDD is a

CAV type of disk

AUTHOR



(/projecthub/RobSmithDev)

RobSmithDev (/projecthub/RobSmithDev)

3 PROJECTS 17 FOLLOWERS

[FOLLOW \(/PROJECTHUB/USERS/SIGN_UP?ID=234087&M=USER&REASON=FOLLOW&RED\)](/PROJECTHUB/USERS/SIGN_UP?ID=234087&M=USER&REASON=FOLLOW&RED)

PUBLISHED ON

September 17, 2017

👍 RESPECT PROJECT (/PROJECTHUB/USERS/SIGN_UP?ID=53752&M=ARTICLE&REASON=RESPECT&RE...

✍ WRITE A COMMENT

🔗 Share

MEMBERS WHO RESPECT THIS PROJECT



(/projecthub/PSoC_Rocks)



(/projecthub/varunjha089)



(/projecthub/daniel-

bergman)



(/projecthub/dimovfx)



(/projecthub/mike-allen)



(/projecthub/Aelectronic)



(/projecthub/brooklyncheetham805)



(/projecthub/ghadeb)

and 14 others

SEE SIMILAR PROJECTS YOU MIGHT LIKE

SIMILAR PROJECTS YOU MIGHT LIKE

(/projecthub/robsmithdev/arduino-amiga-floppy-disk-reader-v1-485582?
ref=similar&ref_id=53752&offset=0)

Arduino Amiga Floppy Disk Reader (V1) (/projecthub/robsmithdev/arduino-amiga-floppy-disk-reader-v1-

Project tutorial by **Team RobSmithDev** (/projecthub/teams/robsmithdev)

24,879 VIEWS **11** COMMENTS **15** RESPECTS

(/projecthub/iotboys/control-led-by-clap-using-arduino-and-sound-sensor-e31809?ref=similar&ref_id=53752&offset=1)

Control LED By Clap Using Arduino and Sound Sensor (/projecthub/iotboys/control-led-by-clap-using-arduino-and-

by Team IoTBoys (/projecthub/teams/iotboys)

145,516 VIEWS **33** COMMENTS **85** RESPECTS

(/projecthub/aip06/arduino-bluetooth-robot-for-android-device-ec93b2?
ref=similar&ref_id=53752&offset=2)

Arduino Bluetooth Robot for Android Device (/projecthub/aip06/arduino-bluetooth-robot-for-android-device-

Project showcase by aip06 (/projecthub/aip06)

8,852 VIEWS **2** COMMENTS **24** RESPECTS

(/projecthub/jsirgado/magnet-levitation-with-arduino-eeeeee4?
ref=similar&ref_id=53752&offset=3)

Magnet Levitation with Arduino (/projecthub/jsirgado/magnet-levitation-with-arduino-eeeeee4?)

Project tutorial by **jsirgado** (/projecthub/jsirgado)

105,649 VIEWS **129** COMMENTS **273** RESPECTS

(/projecthub/onyx/buzzer-alarm-system-with-help-of-arduino-8be82d?ref=similar&ref_id=53752&offset=4)

Buzzer Alarm System With Help Of Arduino (/projecthub/onyx/buzzer-alarm-system-with-help-of-arduino-

by Team ONYX (/projecthub/teams/onyx)

89,669 VIEWS **12** COMMENTS **50** RESPECTS

(/projecthub/Jalal_Mansoori/simplest-way-for-voice-recognition-project-using-c-toarduino-105138?ref=similar&ref_id=53752&offset=5)

Simplest Way for Voice Recognition Project Using... (/projecthub/Jalal_Mansoori/simplest-way-for-voice-recognition-

Project tutorial by Jalal Mansoori (/projecthub/Jalal_Mansoori)

33 337 VIEWS **37** COMMENTS **70** RESPECTS

(<https://www.arduino.cc>)

Powered by
(<https://www.hackster.io>)