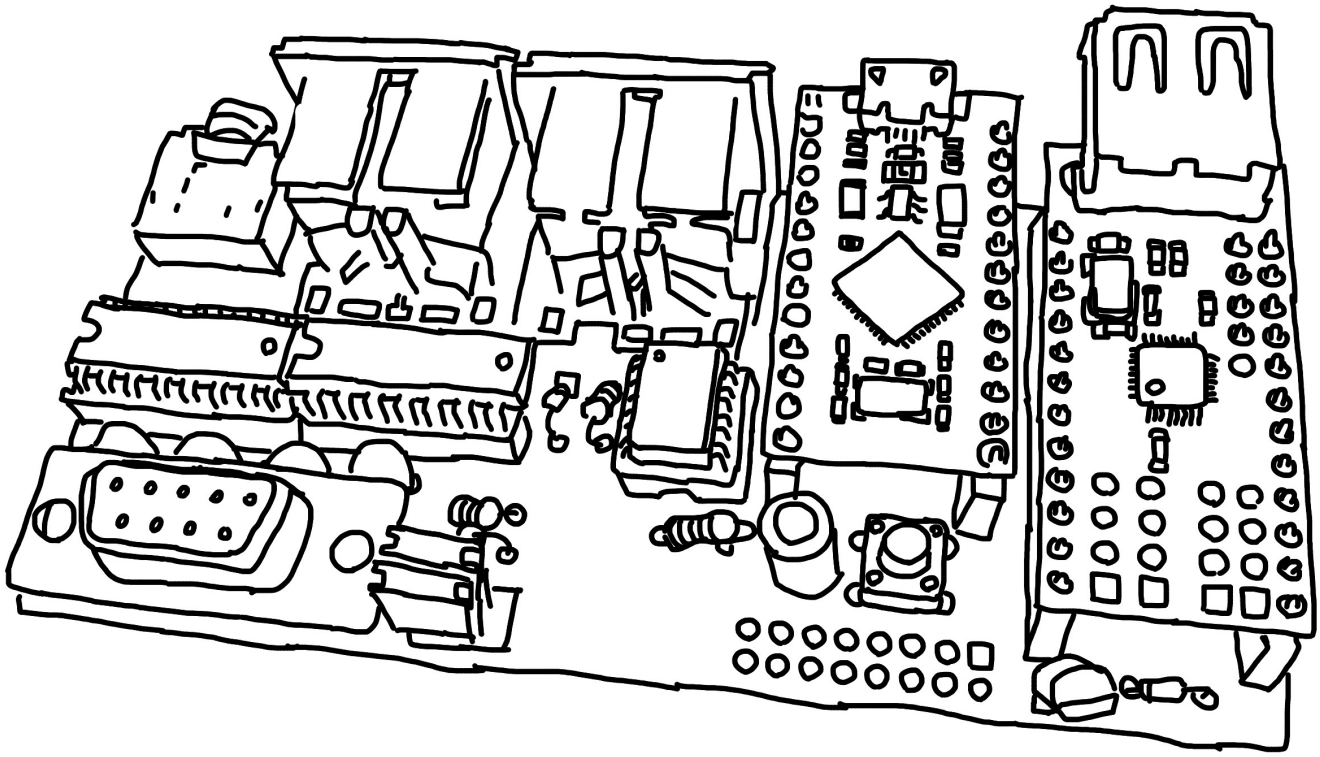
cyberboy666 & underscores.shop present
a midi to rs232 bridge circuit

transcribe

for midi control of old video
mixers & switchers



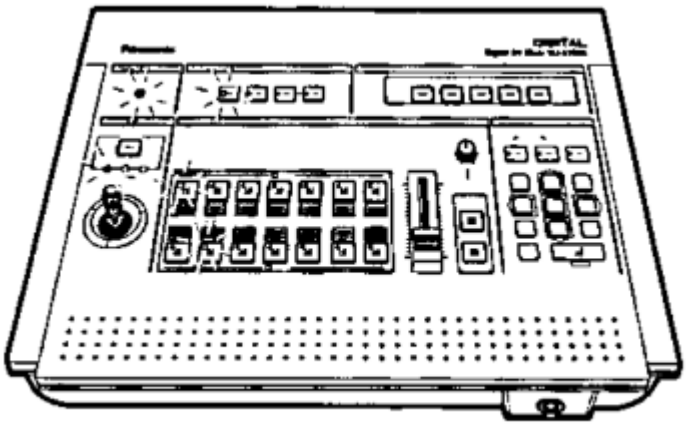
instruction manual and build guide
V0_5_1

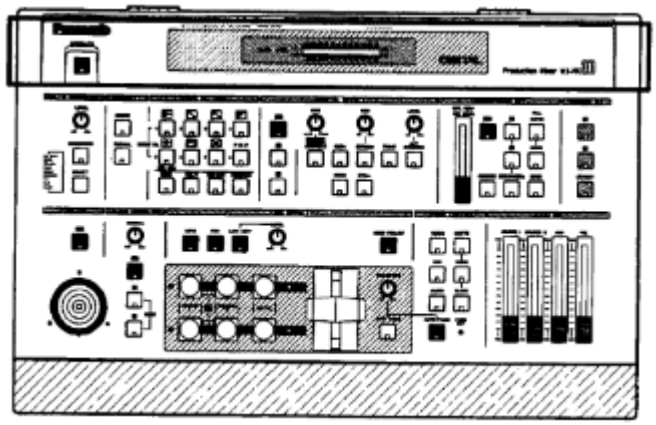
View this project online at underscores.shop/_transcribe_

INTRODUCTION

the *_transcribe_* circuit is a **midi** to **rs232 serial** utility module aiming to give better integrated performance capability to older/obsolete video gear.

some old video mixers and switchers have their own serial command specifications for externally controlling them. the arduino micro-controller in *_transcribe_* runs open source firmware that maps incoming generic midi messages to these specific serial commands.





- So far *_transcribe_* is tested and supported with:

- Panasonic ave55
- Panasonic mx30/50 (partial)
- Extron video switchers
- Kramer video switchers

More devices will be added as people start experimenting with the gear they have around – let me know if you are interested in helping add support for something!

FIRMWARE MAPPING

although no coding background is required for this project, you will need to be comfortable following install instructions and hacking at a few lines of arduino code to customize and set up your own **midi to rs232** mapping

There are guides to both installing the required software and in-depth explanations to how the code that does the mapping works on the projects github page: *github.com/cyberboy666/_transcribe_*

An example of a default mapping for the panasonic ave55 can be seen below

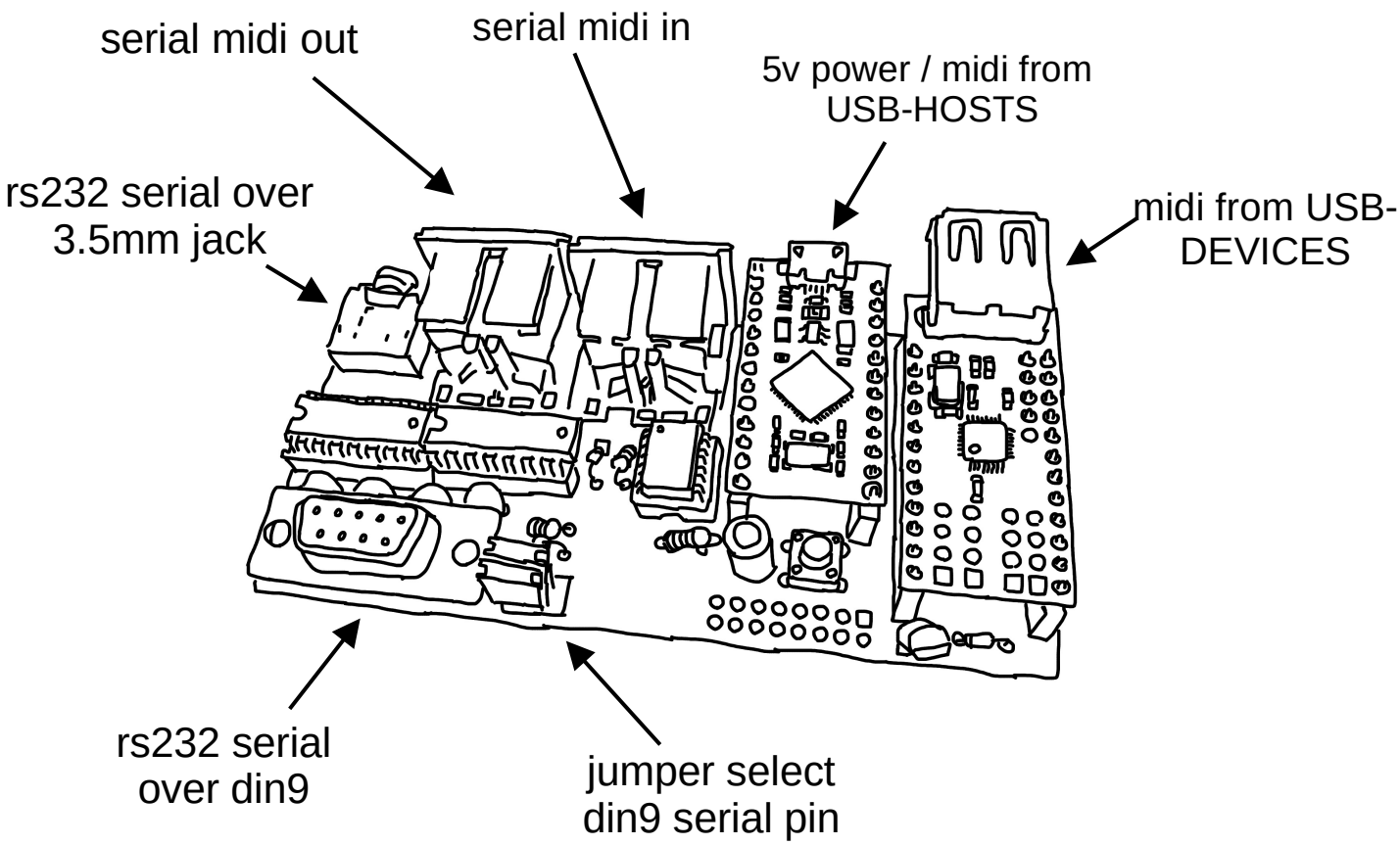
```
void setAVE55nanokontrolMap(){

  if(midiCommand == MIDICOMMAND::NOTEON){
    // put note commands here
  }
  else if(midiCommand == MIDICOMMAND::CC){
    // slider row:
    if(midiParam1 == 0){setCmd(A55_A_B_MIX_LEVEL, midiParam2);}
    else if(midiParam1 == 1){setCmd(A55_THRESHOLD_LUM_KEY, midiParam2);}
    else if(midiParam1 == 2){setCmd(A55_CENTER_WIPE_X, midiParam2);}
    else if(midiParam1 == 3){setCmd(A55_CENTER_WIPE_Y, midiParam2);}
    else if(midiParam1 == 4){setCmd(A55_SCENE_GRABER_X, midiParam2);}
    else if(midiParam1 == 5){setCmd(A55_SCENE_GRABER_Y, midiParam2);}
    else if(midiParam1 == 6){setCmd(A55_COLOR_CORECT_X, midiParam2);}
    else if(midiParam1 == 7){setCmd(A55_COLOR_CORECT_Y, midiParam2);}
```

Don’t worry if all this doesn’t make sense initially – most of the heavy lifting being done under the hood, you just need to say on which midi messages should trigger which serial commands.

However if you do know some **c++** theres room in the code to add some esoteric and fun *mapping functions* such as setting random wipe positions and types on every note etc..

OPERATING INSTRUCTIONS



There are a few different options for inputting midi into _transcribe_ and outputting serial, depending what you are connecting to it:

- input USB-midi from a USB-HOST such as a computer or rpi
- input serial-midi from older midi devices (din5)
- input USB-midi from a USB-DEVICE such as a korg nanokontroller2 or otherwise
- output RS232-serial commands over 3.5mm jack or dsub9 socket to control panasonic video-mixers and other devices
- all these interfaces are bi-directional and so can be modified with firmware to do other useful things (eg like a USB-DEVICE midi -> SERIAL midi converter)

OPERATING INSTRUCTIONS CONT

power

If you are not using micro-usb input on the uC to send midi you can power the circuit here by connecting usb cable to a wall adapter. Alternatively power can be supplied by the 5v line on a euro-power-header

midi input

The small tact button beneath the uC needs to be pressed to connect the usb-midi-device (eg nanokontroller) after it has been plugged in to the USB-A jack on the host-shield.

Serial midi and midi from usb-hosts will connect automatically

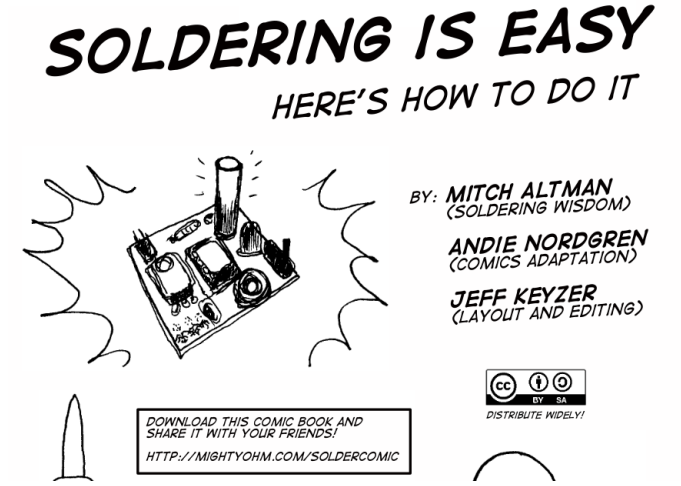
rs-232 output

RS-232 serial is most commonly sent over dsub9 cables with transmit messages on *pin3* and receive messages on *pin2* – however some devices such as Kramer switchers expect this reversed, with transmit messages on *pin2* – for this reason _*transcribe*_ has pin_jumpers (**j9**, **j10**) the left position is default.

*transcribe* also can the send rs-232 over 3.5mm trs stereo cables – where *tip* is transmit, *ring* is receive and *sleeve* is ground.

BUILD INSTRUCTIONS

It is highly recommended to use the interactive BOM to help with placement on this build – type *kutt.it/zLP9yE* into a browser or find the links from the github page



note on soldering: remember to heat pad first (2-3seconds), then add solder, then continue to heat (1-2seconds)

Checkout the web-comic *soldering is easy* for more soldering advice

interface choice / double footprint

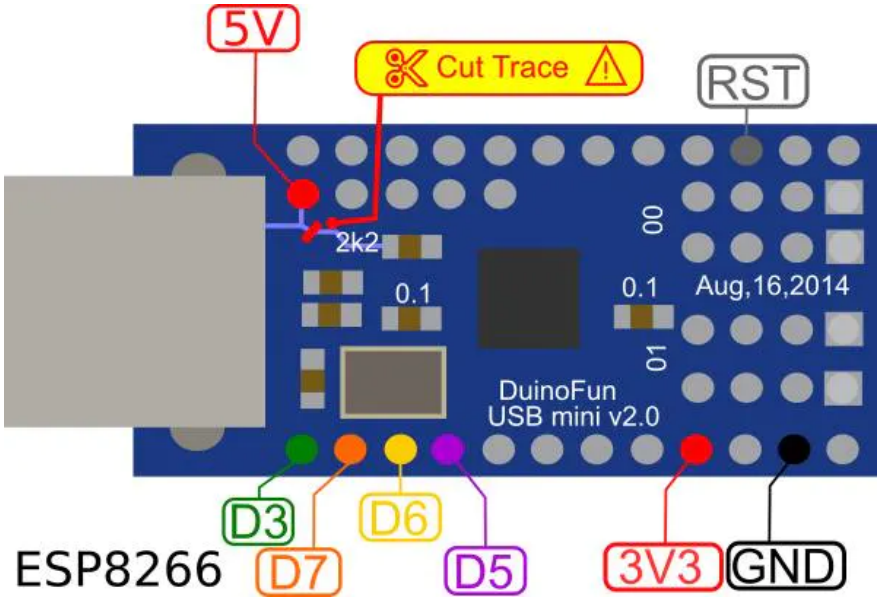
serial-midi is most commonly sent over **din5** (older hardware) *J5 & J7* or **trs 3.5mm jack** (newer hardware) *J4 & J6* - footprints for both are overlaid on the pcb so you can choose which one you would like to have

Usb-host shield

A trace needs to be cut on the usb-host-shield to allow it to be powered by 5v – *see the diagram below* – this is best done with a craft knife – take care not to cut any other traces accidentally.

All pins on the outer vertical lines need to be soldered to the board.

In addition to this the single topmost inner pin (labeled 5v in diagram) needs to be soldered to pcb also



BUILD INSTRUCTIONS CONT.

- Start by soldering the smallest parts first: resistors, diodes, capacitors and regulators - take note of the direction on the diodes - black bar on component matching black bar on footprint

- Next lets do the ic's/sockets - make sure the direction is correct! place in and fold two corner pins to hold in place, then solder all pins. you can place the ic in now too

- Now lets do the *micro-controller* and *usb-host-shield* - if you want to be able to remove them from the board you will need to solder header sockets to the board first – otherwise can directly solder the header pins

- for the *usb-host-shield* right_row I would do a 1x2 header horizontally at the top to catch that 5v pin and then 1x11 row vertically for the rest – the left_row can just be a single 1x12 vertical header

- Finally place the interface parts (eg jacks and sockets) be generous with the solder here -> this is to strengthen the mechanical connections as well as making electrical ones

- **Leave j8 header unpopulated** – this just exposes the bootloader pins so firmware can be reset in the rare case that the pro-micro gets bricked – also leave j1 header unpopulated unless you want to power from a euro-power-header

CREDITS AND MORE INFO

This circuit is distributed through UNDERSCORES – open video hardware label – visit underscores.shop for more info

The pcb was designed using KICAD , this booklet was created in LibreOffice Draw

Everything from gerbers, cad files, panels and documentation is freely available online and distributed under CC-BY-SA / open-source licenses – help us contribute to the commons !

Ask any questions or start discussions related to this project on the *scanlines.xyz* forum – an online community space dedicated to diy av / electronic media art

You can contact me directly at *tim (at) cyberboy666 (dot) com*
Please get in touch if you are interested in hosting a workshop !

Thanks to Gilbert Sinnott for helping with initial experiments. to Bastien Lavaud for circuit advice, always. To Ben Caldwell for project advice. To everyone who has or will contribute ♥♥♥