NASCOM Keyboard Adaptor

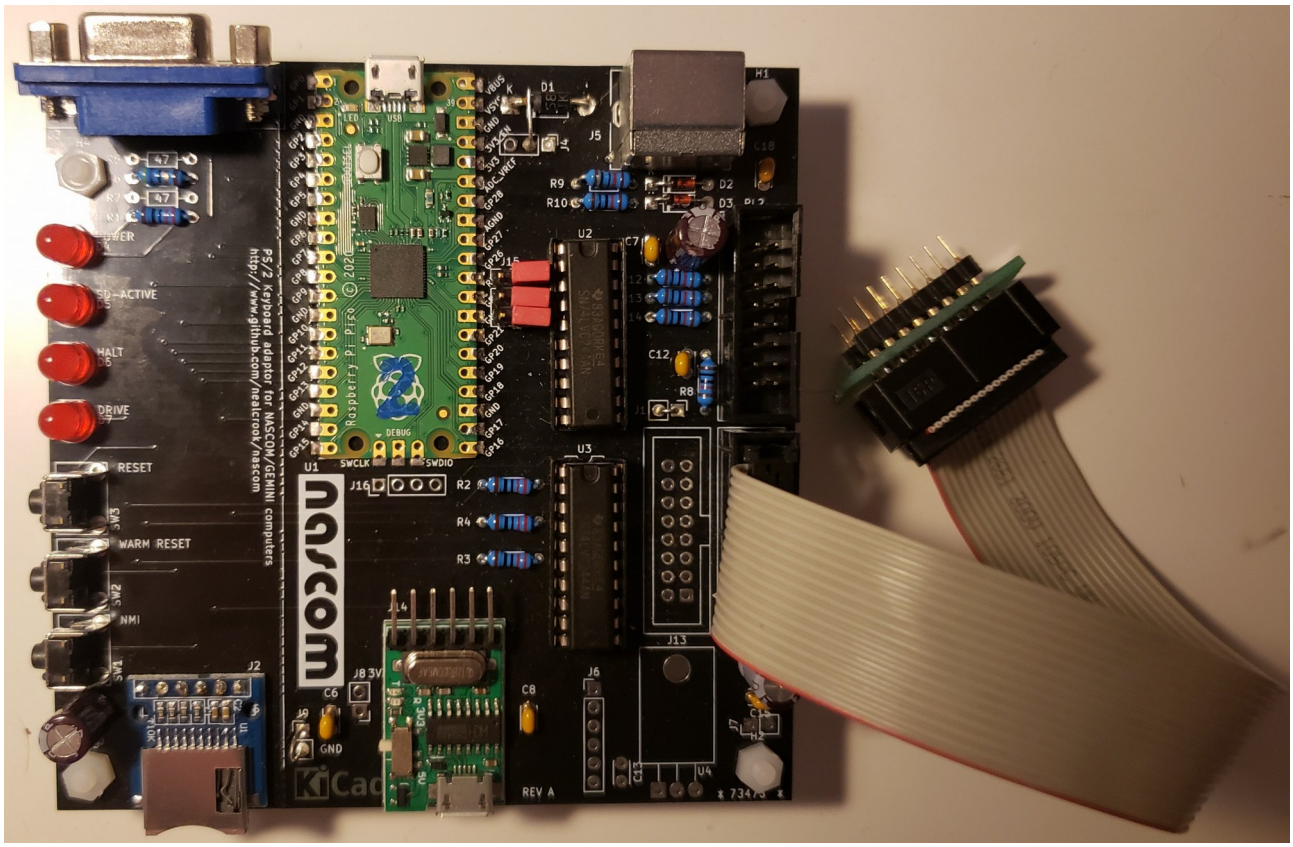

Table of Contents

Introduction

The nascom_kbd interface board measures 100mm x 100mm. It allows a PS/2 keyboard to be connected to the keyboard interface of an unmodified NASCOM 1 or NASCOM 2. It is powered from the NASCOM and appears, to software, identical to the standard Licon keyboard which is scanned under software control.

Some of the keyboard key assignments are changed, to get maximum similarity to the NASCOM layout. In addition:

- There are 2 sets of cursor keys: the “standard” PS/2 keys and repurposed keys to the left and right of the space bar to match the NASCOM 2 keyboard
- On a NASCOM 1, all the extra NASCOM 2 keys are available (cursor, GRAPH etc.)
- The PS/2 “Sys Req” key acts as a reset key for the NASCOM
- The PS/2 “Scroll Lock” key acts as an NMI key for the NASCOM 2 (with a wire-add to the NASCOM 1 it should also work on that machine)
- On a NASCOM 1, the board provides a power-on reset

The nascom_kbd interface board includes additional connectors for future experimentation or added functionality. The following are on my “future projects” list:

- A parallel keyboard suitable for connection to Gemini processor/video cards or the MAP80 Systems VFC
- A serial keyboard suitable for connection to a Gemini processor card.
- nascom_sdcard serial interface functionality
- complete Nascom 2 emulation.

This document covers usage, theory of operation, construction, programming and installation. Usage is at the front; the other sections are relegated to the back of the document because you may only care about them once.

Additional documentation (schematics, assembly drawings, photos, software, kicad PCB database) can be found at https://github.com/nealcrook/nascom/nascom_kbd/.

Errors and omissions

You may find some information here wrong or confusing. You may look on my github repository and not be able to find something that you are looking for: or it may seem wrong or out-of-date. Please report any such instances to the author.

Usage

This document (currently) assumes that you will only fit the components required for the NASCOM keyboard interface.

The three jumpers J15 select the operating mode as follows:

- All jumpers absent: NASCOM Keyboard
- All jumpers fitted: PS/2 reporting/debug mode
- All other combinations: reserved

Before first use, program the Raspberry Pi Pico (see Programming on page 3). Use a 16-pin ribbon cable terminated at each end with a female IDC. Connect a PS/2 keyboard to the nascom_kbd.

Connect PL3 on the nascom_kbd to PL3 on a NASCOM 2 or SK1 on a NASCOM 1 (see NASCOM 1 Adaptor on page 6).

- Ensure that the ribbon cable maintains the same Pin 1 polarity at each end.
- Be careful to orient the cable correctly for Pin 1 at each end.

Usage (debug)

In case of problems, use this debug mode to test the nascom_kbd with the PS/2 keyboard.

- Fit 3 jumpers to J15
- Connect a PS/2 keyboard to the nascom_kbd
- Connect a USB cable from the Raspberry Pi Pico on the nascom_kbd to a PC
- Run a terminal emulator on the PC at 115200bd

For example, start minicom using the following command:

```
$ minicom -b 115200 -o -D /dev/ttyACM0
```

Now, each press and each release of a key on the PS/2 keyboard should result in the keycodes being reported in the terminal. For example, pressing the Q key should generate a code of 15, which should repeat if the key is held down, and releasing the Q key should generate the two codes F0 15.

Programming

To program/reprogram the Raspberry Pi Pico (should work under Linux/MAC OS/Windows):

- Connect a USB cable from the Raspberry Pi Pico on the nascom_kbd to a PC. The “Power” LED should illuminate on the Raspberry Pi Pico.
- Press and hold the white BOOTSEL switch on the Raspberry Pi Pico. Press and release the RESET switch on the nascom_kbd. Release the BOOTSEL switch
- After about 1 second, the Raspberry Pi Pico should appear on your PC as a mass storage device. Open it with a file manager: it should contain files “INDEX.HTM” and “INFO_UF2.TXT”. Drag-and-drop the new image (nascom_multiboard.uf2) to the mass storage device. After a couple of seconds the Raspberry Pi Pico should reboot, unmounting itself in the process. Programming is now complete.

Keyboard Markup

The photo below shows the keyboard markup. This can be achieved by various methods. I used Edding 780 Extra-Fine paint markers: a black marker to obscure the original marking, followed by a white marker to provide the new marking.

Theory of operation

The Raspberry Pi Pico has 2 processors so I took a very simple approach to the software. The NASCOM controls the keyboard by using a "reset" and "clock" signal to select 9 columns of the matrix in turn. For each column, it reads back the state of the associated rows of the keyboard. In my implementation, processor 1 polls the PS/2 keyboard. It decodes the key press/key release codes and maintains a keyboard map which represents the state of the keyboard matrix. Processor 2 polls the "reset" and "clock" signals from the NASCOM and tracks what column is selected. Each time a column is selected processor 2 reads the associated entry from the keyboard map and drives it on the output to the NASCOM.

SN74LVC244 parts are used to perform level translation between 5V TTL signals on the NASCOM and the 3V3-tolorant pins of the Raspberry Pi Pico. It's annoying to have to use level shifters but I was not convinced that an Arduino could respond fast enough to the real-time polling signals from the NASCOM.

The Schottky diode is arranged so that the nascom_kbd can be powered either by the attached NASCOM or via the USB connector of the Raspberry Pi Pico. The diode ensures that one or other source will power the board but that neither source will try to power the other source.

Additonal components shown on the schematic are intended to allow the board to provide various other bits of functionality (some of which are mutually exclusive due to pin sharing).

Construction

The nascom_kbd has been designed to support multiple assembly options. This section describes the minimal assembly required for interfacing a PS/2 keyboard to a NASCOM.

In the parts-list (below), red text is part number from www.cpc.co.uk.

Reference	Description
	PCB
R8, R9, R10, R12, R13, R14	10K resistor
R1	220R resistor
D1	1N5817 Schotty Diode SC11443
D2, D3	3V3 Zener diode
J1	wire link
C6, C7, C8, C12, C18	0.1uF decoupling capacitor
C9, C16	10uF 16V electrolytic capacitor
U1	Raspberry Pi Pico (the orginal version; no wifi/bluetooth)
J4	wire link from pin 2 to K (bar side) of D1
D1	5mm red LED SC08044
U2, U3	SN74LVT244 (in 2, 20-pin DIL sockets if required)
J15	2x3 6-pin 0.1” pitch header pins
PL3	16-way 2x8 male pin IDC connector, polarised CN17032
SW3	momentary action push switch (Aliexpress)
H1, H2, H3, H4	Nylon standoff (optional)
J5	PS/2 female PCB-mount socket (Aliexpress/ebay)

You may also need (N2) a ribbon cable terminated at each end with a 16-way connector (CN21933) .

Construction is simplest if you fit low-profile components first.

Fit the components in the order that they are listed in the table above. Use a polarised open-frame socket for PL3 and fit it to match the polarity markings on the PCB.

J4 pin 2 must be wired to the K (bar) end of the Schottly diode, D1. This corrects a wiring error. This connection is marked up on the schematics with a dashed line and a note.

Solder U1 flush to the board; solder the castellations. Solder two diagonal corners first and check alignment before proceeding. You can use some header pins to align the module on the pcb.

NASCOM 1 Adaptor

The NASCOM 1 keyboard uses a 16-pin DIL socket. The NASCOM 2 uses a 16-pin IDC connector. The pin numbering of these connectors follows different conventions but physically the same pins are adjacent. Therefore, a very simple adaptor can be used to convert between IDC and DIL, allowing an IDC-terminated ribbon cable to be connected neatly to NASCOM 1 – the photo on page 1 shows such an adaptor; three are provided on my NASCOM 1 Multi-Expansion Board.

Change History

24Oct2022	Neal Crook, foofoobedoo@gmail.com	1st Edit.