

PS/2 Keyboard for MSX And Legacy computers

Solution

Based on:

- . ARM Cortex M3 Blue Pill stm32f103c6t6**
- . ARM Cortex M4 Black Pill stm32f401ccu6**

Evandro Souza <evandro.r.souza@gmail.com>

Introduction

The main objective was to learn to use the ARM Core M3 and M4 Open Develop Enviroment (free)

To do so, I choose a critic mission challenge, with implementation features not available in various smart solutions, as an example, the update translation database tables are not possible without full firmware recompilation and upload to hardware.

Show up a simple and low cost solution to implement.

Index

Enviroment characteristics and
considerations

About get the right MCU

Develop enviroment setup

The electronics

Firmware features

Firmware Technical Datasheets



Now

Enviroment characteristics and
considerations

About get the right MCU

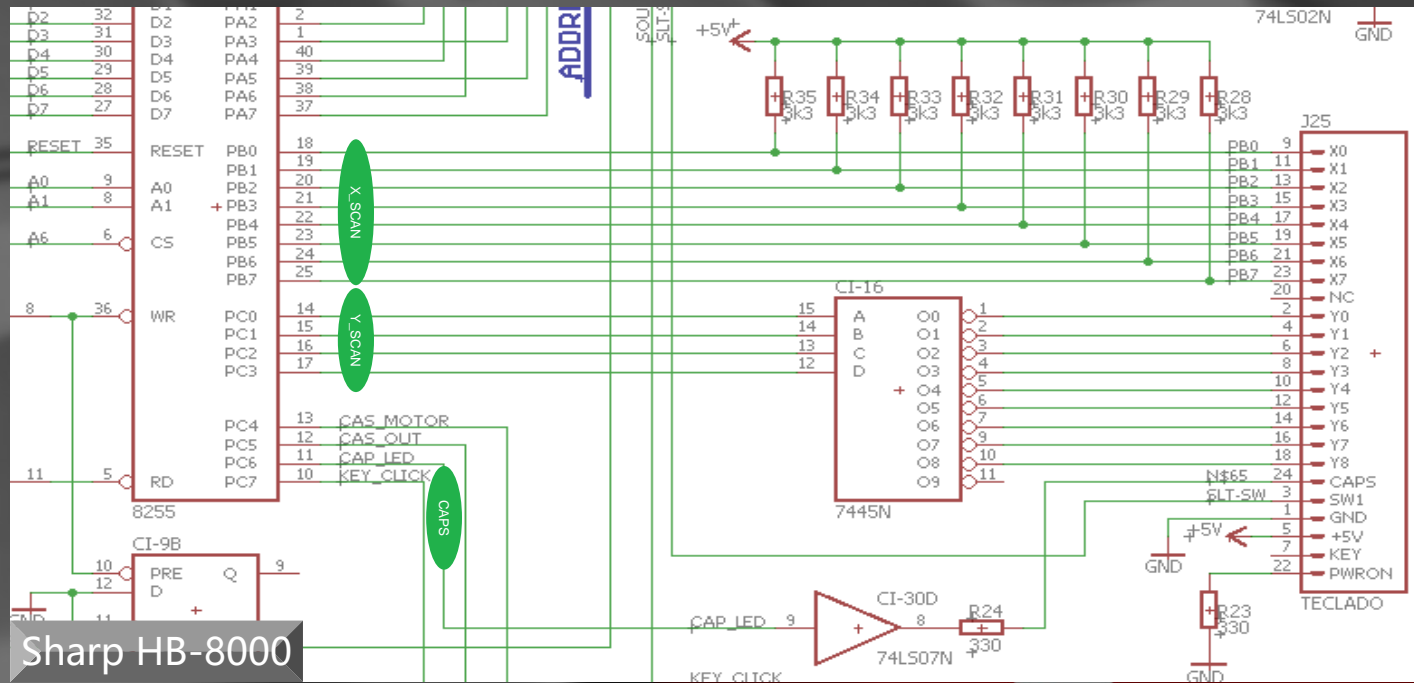
Develop enviroment setup

The electronics

Firmware features

Firmware Technical Datasheets

How MSX works (1/2)



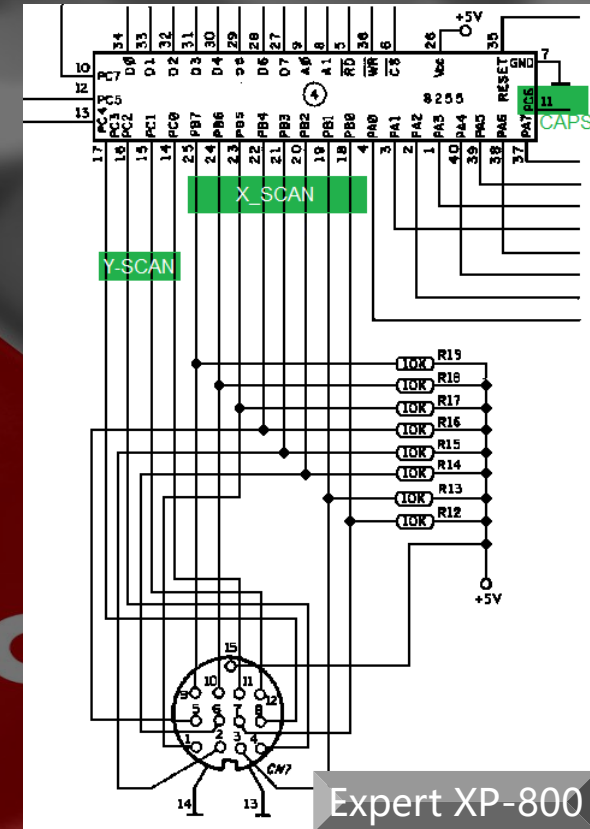
Sharp HB-8000

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|-----|-----|-----|-----|-----|-----|-----|------------|-----------|----------|
| | | Y=0 | Y=1 | Y=2 | Y=3 | Y=4 | Y=5 | Y=6 | Y=7 | Y=8 |
| FE | X=0 |) | * | ^ | C | K | S | SHIFT | F4 | Space |
| FD | X=1 | ! | (| [| D | L | T | CTRL | F5 | CLS HOME |
| FB | X=2 | @ | - | : | E | M | U | GRAPH | ESC | INS |
| F7 | X=3 | # | + | : | F | N | V | CAPS | TAB | DEL |
| EF | X=4 | \$ | ^ | ? | G | O | W | CODE/ KANA | STOP | LEFT |
| DF | X=5 | % | . | > | H | P | X | F1 | BackSpace | UP |
| BF | X=6 | " | ' | A | I | Q | Y | F2 | SLCT | DOWN |
| 7F | X=7 | & | ç | B | J | R | Z | F3 | RETURN | RIGHT |

The MSX keyboard is a 8 lines (X) by up to 15 columns (Y) matrix, which can achieve a theoretical maximum of 120 keys – The HB8000 has 9 columns, as seen on the example (J25 is the keyboard connector and, on left, its key mapping table). A highend japanese MSX uses a 11 columns keyboard;

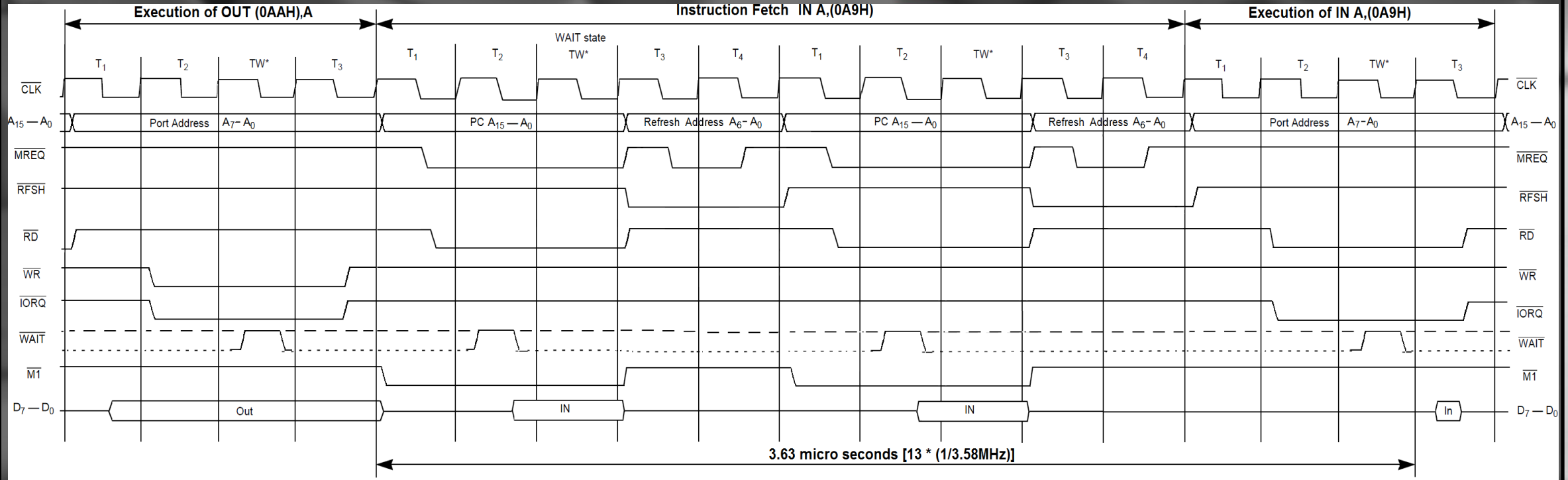
Firstly, the scan is done by writing on Y3:0 (PC3:0, address 0AAH), which goes to a decoder (up to 4x16 with open-collector outputs, and reading from X7:0 (through PPI 8255 PB7:0, port 0A9H);

The Caps Lock and Katakana states ("0" active) are sampled directly from MSX hardware and echoed on PS/2 keyboard LEDs.



Expert XP-800

How MSX works (2/2)



Detailed timing diagram of a MSX Keyboard read (OUT followed by an IN)

The available amount of time after the Y update (execution of OUT (0AAH), A) is approximately 3,6μs;

- Obs.: In special circumstances, the keyboard reading (IN A, 0A9H) is not done just after OUT (eg: games), so, the converter must update X as soon as the PS/2 Keyboard events occur, according to respective Y of the moment. In other words, the converter must keep updating X even if the MSX is not doing the keyboard scan.

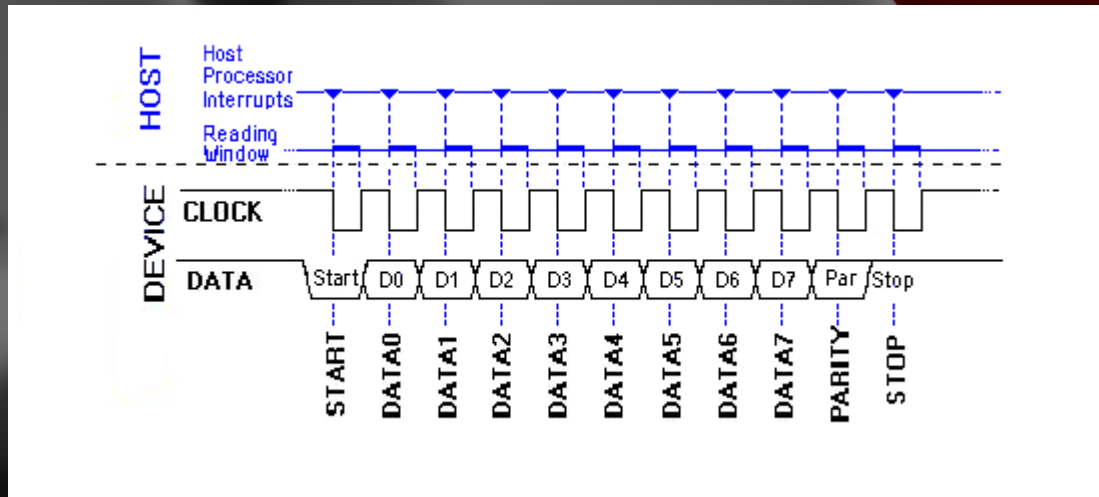
PS/2 Keyboard - Timmings

The PS/2 keyboard communicates to host computer through a *half duplex* synchronous serial line with 2 wires, in a unbalanced (gnd is the return path) communication mode. Two lines are used: Data and Clock;

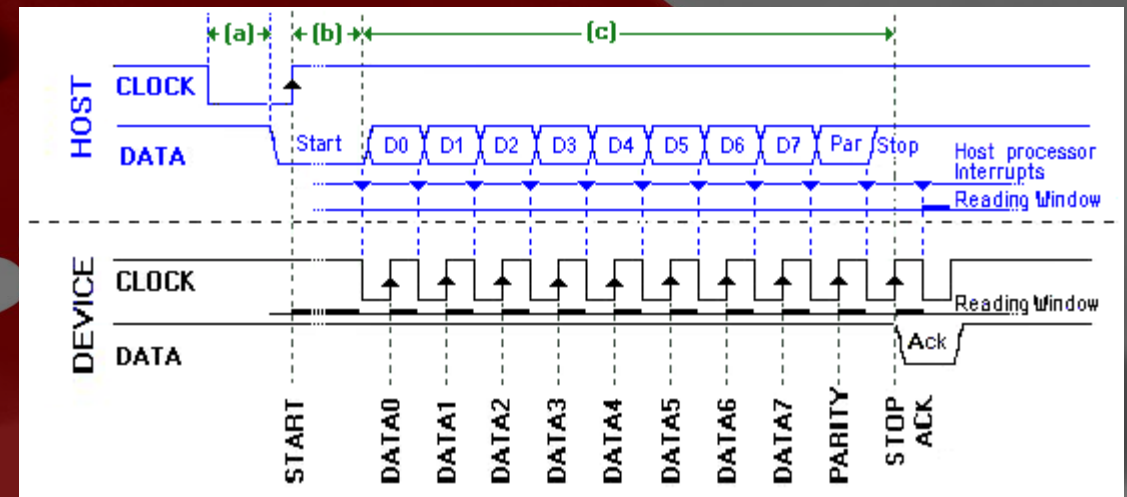
- The clock rate is in range from 10K to 16,7KHz;

Observe the following timing diagrams of:

1) From PS/2 Keyboard to host (PC or converter):

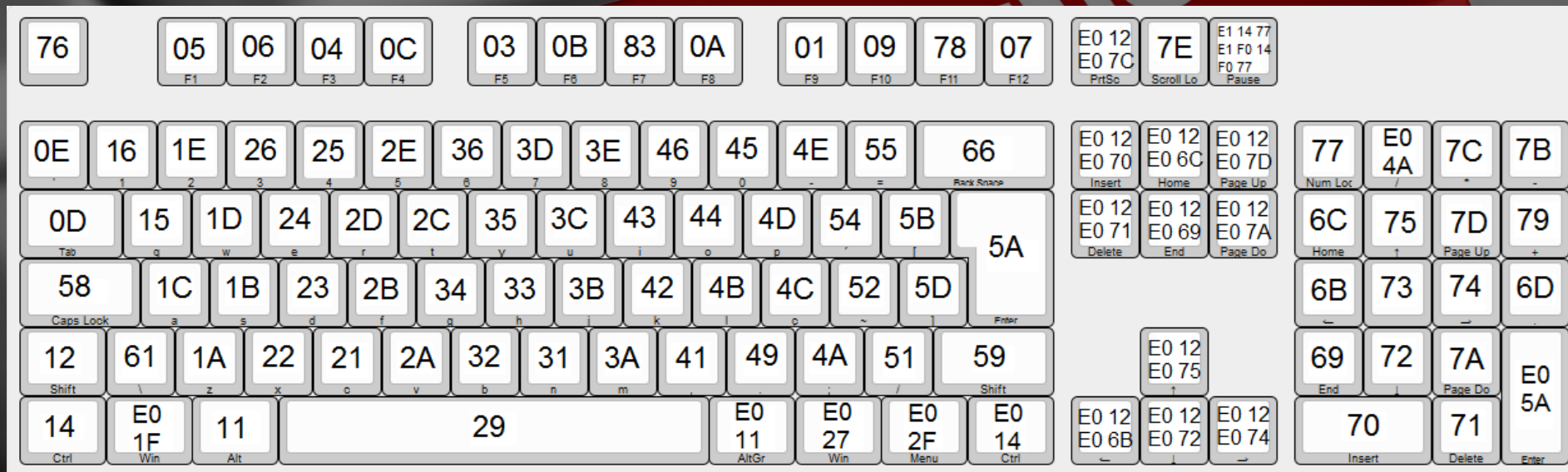


2) From host to keyboard:



- (a) Host forces the clock line to low (0) for at least 100 μ s;
- (b) Is the time that keyboard requires to start to send clock to receive the frame, which can take up to 15ms;
- (c) Is the time to conclude the frame reception, after the keyboard has started the clock sending, and must be up to 2ms;

- There are many exclusive MSX keys, like Graph, Code, Select and Stop. That though is also valid for a PC keyboard;
- The scan codes do not have any pairing (relationships) with keytops mark;
- The scan codes have different sizes: from 1 to 8 bytes;
- The PS/2 keyboard sends Type 2 codes (AT type) to the host and the auto-repeat is only on the last one pressed;
- The scan codes are not dependent of physical layout – For example: The codes 0x51 (key “/” on the left side of right Shift) and 0x6D (Key “.” of numeric keypad) are exclusive of ABNT2 (Id=275) keyboards;
- Here is the example of the base keyboard to todo this development (ABNT2 keyboard) with make scan codes:



The background of the slide is a grayscale image of a computer keyboard. A single red key is highlighted, positioned diagonally across the center-right of the frame. The word "Alt" is partially visible on the red key. The overall image is slightly blurred, focusing attention on the text overlay.

Now

Environment characteristics and considerations

About getting the right MCU

Develop environment setup

The electronics

Firmware features

Firmware Technical Datasheets

STM32F4x1Cx v2.0+

Pinout Diagram

The diagram shows the STM32F4x1Cx v2.0+ microcontroller with its pinout details. The pins are numbered 1 to 48. The functions are listed for each pin, with color-coded boxes indicating the pin type. The legend on the right defines the colors: POWER (red), GROUND (black), CPU Pin (blue), Pin Name (green), CONTROL (yellow), ANALOG (purple), TIMER & CHANNEL (orange), USART (pink), SPI/DS (light blue), SDO (dark blue), I2C (light green), CAN BUS (light orange), USB (light purple), MISOC (light pink), BOARD BAREWARE (light yellow), and 3.3V Tolerant (light blue). The notes at the bottom left specify that pins 1 and 4 are only used by CAN and don't have any pins, and all pins are 5V tolerant on PA01. Pins 10 and 41 on PA11 are 3.3V only.

| Pin | Function | Pin | Function | Pin | Function | Pin | Function |
|-----|----------|-----|----------|-----|----------|-----|----------|
| 1 | T1_BKIN | 25 | PA15 | 29 | GPIO | 41 | MCOD1 |
| 2 | NB5 | 26 | PA16 | 30 | GPIO | 42 | GPIO |
| 3 | CH1N | 27 | PA17 | 31 | GPIO | 43 | GPIO |
| 4 | CH2N | 28 | PA18 | 32 | GPIO | 44 | GPIO |
| 5 | CH3N | 29 | PA19 | 33 | GPIO | 45 | GPIO |
| 6 | CH4N | 30 | PA20 | 34 | GPIO | 46 | GPIO |
| 7 | CH5N | 31 | PA21 | 35 | GPIO | 47 | GPIO |
| 8 | CH6N | 32 | PA22 | 36 | GPIO | 48 | GPIO |
| 9 | CH7N | 33 | PA23 | 37 | GPIO | | |
| 10 | CH8N | 34 | PA24 | 38 | GPIO | | |
| 11 | CH9N | 35 | PA25 | 39 | GPIO | | |
| 12 | CH10N | 36 | PA26 | 40 | GPIO | | |
| 13 | CH11N | 37 | PA27 | 41 | MCOD1 | | |
| 14 | CH12N | 38 | PA28 | 42 | GPIO | | |
| 15 | CH13N | 39 | PA29 | 43 | GPIO | | |
| 16 | CH14N | 40 | PA30 | 44 | GPIO | | |
| 17 | CH15N | 41 | PA31 | 45 | GPIO | | |
| 18 | CH16N | 42 | PA32 | 46 | GPIO | | |
| 19 | CH17N | 43 | PA33 | 47 | GPIO | | |
| 20 | CH18N | 44 | PA34 | 48 | GPIO | | |
| 21 | CH19N | 45 | PA35 | | | | |
| 22 | CH20N | 46 | PA36 | | | | |
| 23 | CH21N | 47 | PA37 | | | | |
| 24 | CH22N | 48 | PA38 | | | | |
| | | | PA39 | | | | |
| | | | PA40 | | | | |
| | | | PA41 | | | | |
| | | | PA42 | | | | |
| | | | PA43 | | | | |
| | | | PA44 | | | | |
| | | | PA45 | | | | |
| | | | PA46 | | | | |
| | | | PA47 | | | | |
| | | | PA48 | | | | |
| | | | PA49 | | | | |
| | | | PA50 | | | | |
| | | | PA51 | | | | |
| | | | PA52 | | | | |
| | | | PA53 | | | | |
| | | | PA54 | | | | |
| | | | PA55 | | | | |
| | | | PA56 | | | | |
| | | | PA57 | | | | |
| | | | PA58 | | | | |
| | | | PA59 | | | | |
| | | | PA60 | | | | |
| | | | PA61 | | | | |
| | | | PA62 | | | | |
| | | | PA63 | | | | |
| | | | PA64 | | | | |
| | | | PA65 | | | | |
| | | | PA66 | | | | |
| | | | PA67 | | | | |
| | | | PA68 | | | | |
| | | | PA69 | | | | |
| | | | PA70 | | | | |
| | | | PA71 | | | | |
| | | | PA72 | | | | |
| | | | PA73 | | | | |
| | | | PA74 | | | | |
| | | | PA75 | | | | |
| | | | PA76 | | | | |
| | | | PA77 | | | | |
| | | | PA78 | | | | |
| | | | PA79 | | | | |
| | | | PA80 | | | | |
| | | | PA81 | | | | |
| | | | PA82 | | | | |
| | | | PA83 | | | | |
| | | | PA84 | | | | |
| | | | PA85 | | | | |
| | | | PA86 | | | | |
| | | | PA87 | | | | |
| | | | PA88 | | | | |
| | | | PA89 | | | | |
| | | | PA90 | | | | |
| | | | PA91 | | | | |
| | | | PA92 | | | | |
| | | | PA93 | | | | |
| | | | PA94 | | | | |
| | | | PA95 | | | | |



1. Low power consumption; ✓
2. At least 16 pins must be 5V tolerant; ✓
3. Output pins must support be configured as OD (Open Drain); ✓
4. OD pins must allow to get their state read at any time; ✓
5. At least 5 pins must be enabled as external interrupt; ✓
6. The external interrupt pins connected to Y_SCAN must not share interrupt resources; ✓
7. Its non volatile memory must be grather than 2560 bytes and must be writeable by software; ✓
8. Plenty documentation available; ✓
9. Open develop enviroment; ✓
10. Low cost; ✓
11. Easy to get. ✓

The background of the slide is a grayscale image of a computer keyboard. A single key, located in the lower-middle section, is highlighted in a vibrant red color. This red key features the word "Alt" in a white, sans-serif font. The other keys are in shades of gray, creating a strong contrast with the red key. The overall composition is clean and modern, with a focus on the highlighted key.

Now

Enviroment characteristics and considerations

About get the right MCU

Develop enviroment setup

The electronics

Firmware features

Firmware Technical Datasheets

Develop Enviroment Setup

AMD x64

Linux Ubuntu 22.04

ARM GCC Tools

LibOpenCM3

OpenOCD

Visual Studio Code

ARM aarch64

Debian Linux 11

ARM GCC Tools

LibOpenCM3

Black Magic Probe

Visual Studio Code

The first steps was done with the orientation of this remarkable book:

Warren Gay, Beginning STM32 Developing with FreeRTOS, libopencm3 and GCC, which was introduced me by Ismael Lopes da Silva, site <https://www.embarcados.com.br/serie/programacao-com-a-placa-blue-pill/>, to whom I am very grateful to his excellent level of didactic and evaluation article!



Now

Enviroment characteristics and
considerations

About get the right MCU

Develop enviroment setup

The electronics

Firmware features

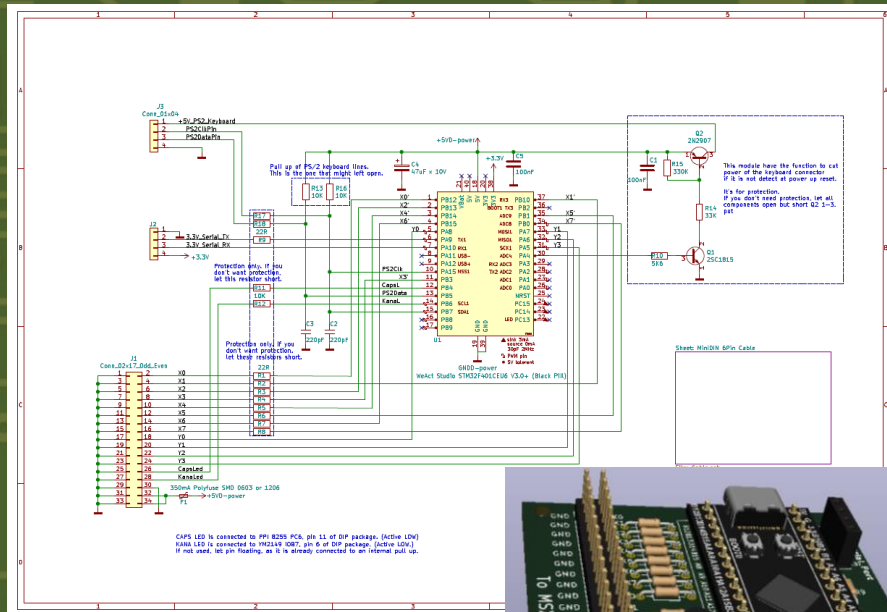
Firmware Technical Datasheets

The Electronics

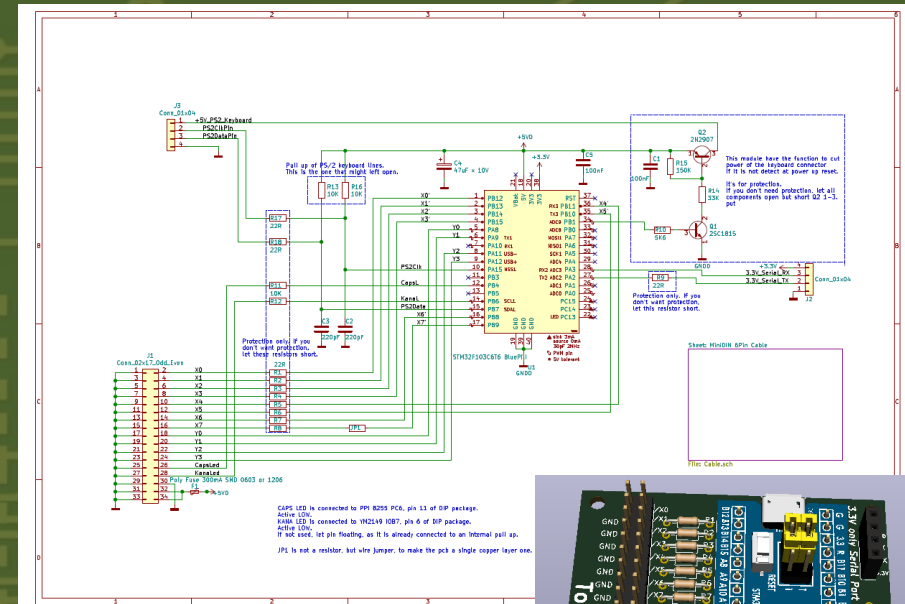
Modules usage comes with advantages:

- ARM 3.3V power is resolved from MSX 5V;
- All support circuitry (reset, crystal, SWD, status led, etc) already up;
- 2.54mm DIP form feed, instead of SMD: Easier to match legacy.

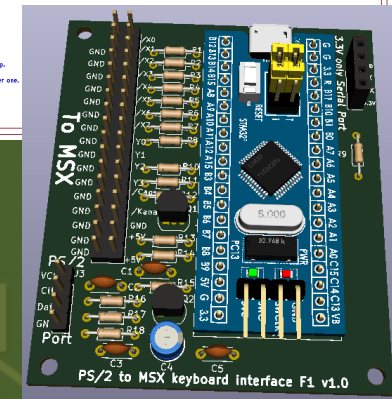
- The EDA (Electronic Design Assitant) used to do the design is KiCad;
- Both Eeschema and Pcbnew.



Black Pill Module
Version

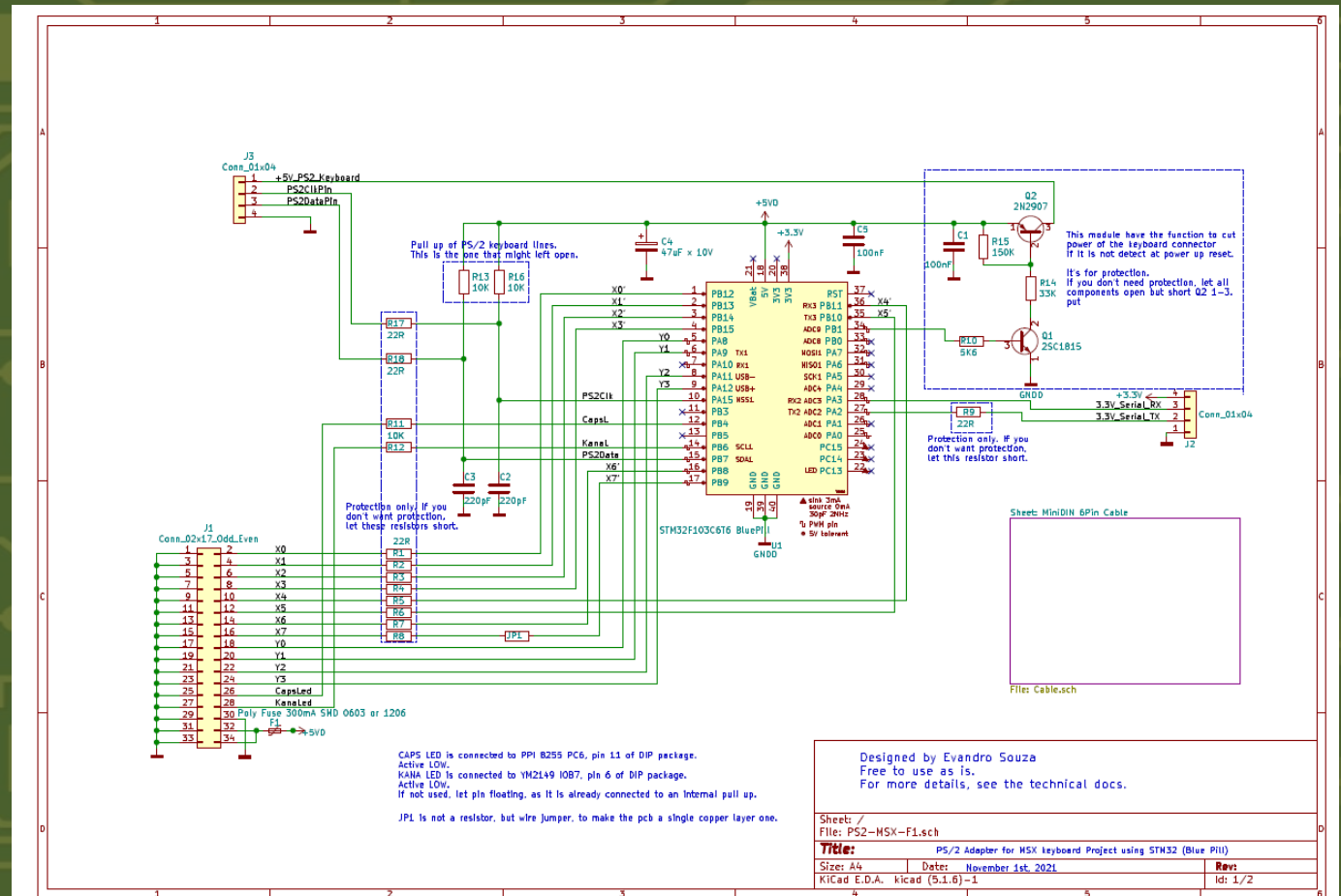


Blue Pill Module
Version



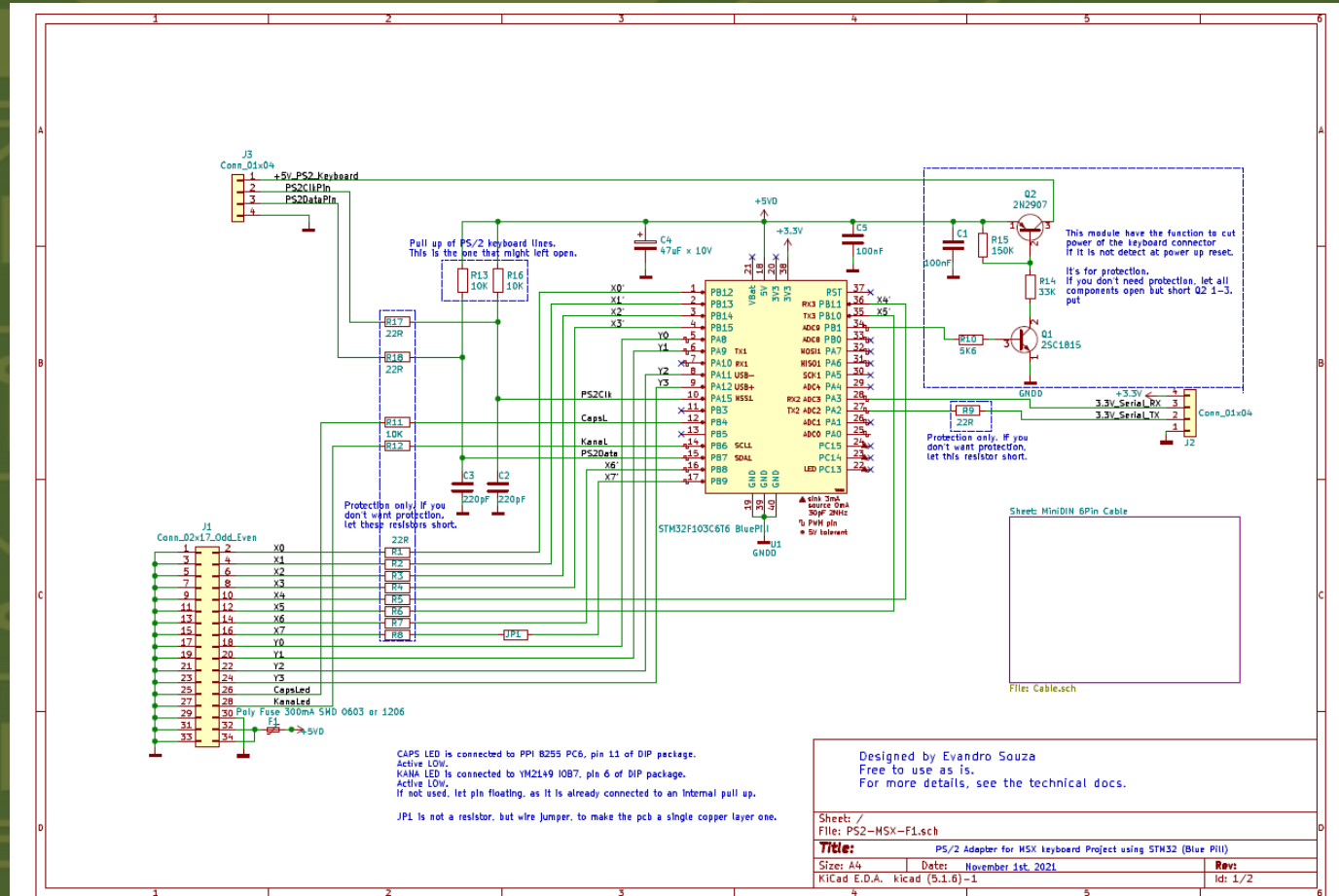
The electronics with Black Pill module

- 100% of external electronics has the function of protecting the PS/2 Keyboard, ARM module and MSX host;
- Powers down the PS/2 interface when keyboard is not detected while booting-up;
- C2 and C3 (the capacitors in parallel with PS2Clk and PS2Data) absorb switching spikes, minimizing false detections of PS/2 changes;
- The measured consumption with black pill module was 28mA @ 5V;
- Dimension of single layer PCB: 65 x 55mm;
- BlackPill has USB capabilities has USB and USART Port as 5V tolerant, mitigating damages to connect to 5V data.



The electronics with Blue Pill module

- Blue Pill has the same basic specifications as Black Pill, but does not have USB implemented. This reduced the code size, so I choose a cheaper MCU: stm32f103c6t6 (32K Flash 10K RAM).
- Although BluePill has USB capabilities, there is no sufficient resources (5V tolerant pins) to feasible USB. Even to UART I have to use 3.3V pins (no 5V tolerant ones).



Now

Enviroment characteristics and
considerations

About get the right MCU

Develop enviroment setup

The electronics

Firmware features

Firmware Technical Datasheets

Firmware features

The Database (PS/2 to MSX translation instructions) may be changed at any time: You have only to get a new Database, connect a tty terminal capable to send ASCII files and connect this terminal to the PS/2 to MSX Converter console, through an USB or a serial port. To do so, just unplug the keyboard and turn on the Converter.

The Database (an Intel Hex text file) is updated via console, with a help of a tty program (eg TIO) , that will guide you through the steps => A 15 seconds operation.

The Converter controls NumLock as PC does, and CapsLock and Scroll Lock leds, this last one mapped from Katakana (or Cyrillic or Korean) indicator, are sampled directly from MSX hardware. It echoes real time state changes.

Due to the easy Database upload task, the user is empowered to rebuild new one and upload anytime how this converter will act.

The valid PS/2 events for the Converter are only make and break. As auto repeat has no sense for Converter firmware, it has been kept as low as possible: Autorepeat start delay is initialized as 1.0s and repeat rate as 2CPS.

The MSX puts column (Y_Scan) and the converter answers with line pointed (X_Scan) by Y_Scan. If there are no Y_Scan changes, this Converter will update X_Scan PS/2 Keyboard events related to current value Y_Scan.

Now

Enviroment characteristics and
considerations

About get the right MCU

Develop enviroment setup

The electronics

Firmware features

Firmware Technical Datasheets

Firmware Technical Data



The philosophy used is to do a *Bare metal* programming, it means no use of any Operating System or RTOS.

The used languages are C++ and C, to make all firmware and TIO. The one used to create/manage Database was VBA (Visual Basic for Applications) through Microsoft Excel.

Task allocation here is used and implemented by hardware interruptions, triggering: MSX Y-Scan, PS/2 keyboard, USART, USB, DMA, 30Hz System timer & 1 μ s resolution Timer.

- The complete source code with Doxygen documentation, open excel Database compiler, schematics, pcb and their gerber files are all available at github repository: <https://github.com/evandrosouza-developer/ps2tomsxUSB>;
- Other designs of this ecosystem: MSX Keyboard Emulator <https://github.com/evandrosouza-developer/Tester-ps2-msx> and TIO (Tiny Terminal application) <https://github.com/evandrosouza-developer/tio-v2.5a>.

THANKS

Solution