PS/2 Keyboard for MSX And Legacy computers

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.

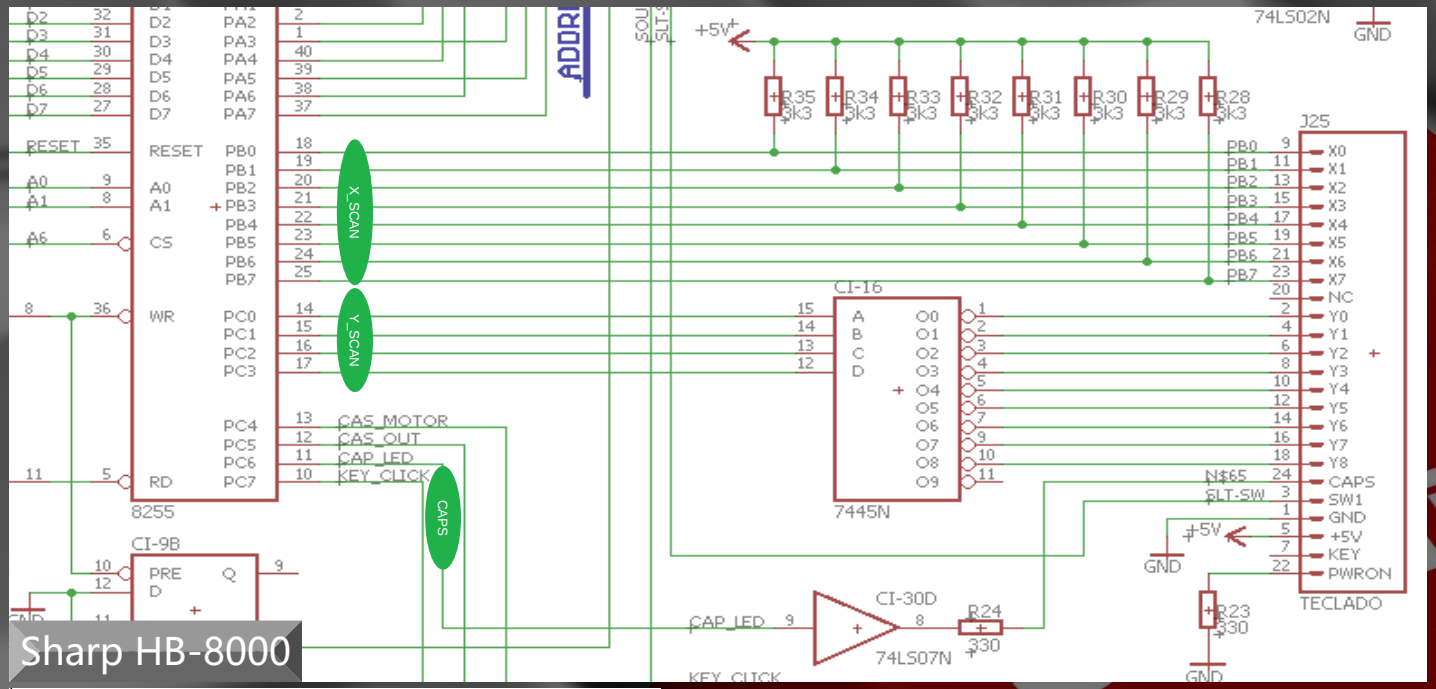
Show up a simple and low cost solution to implement.

Index

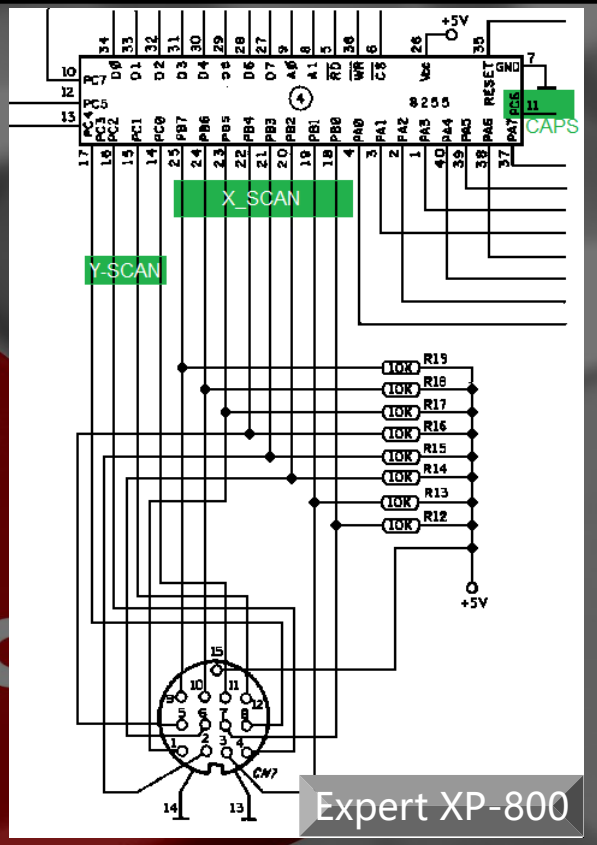
Now

- 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



Expert XP-800

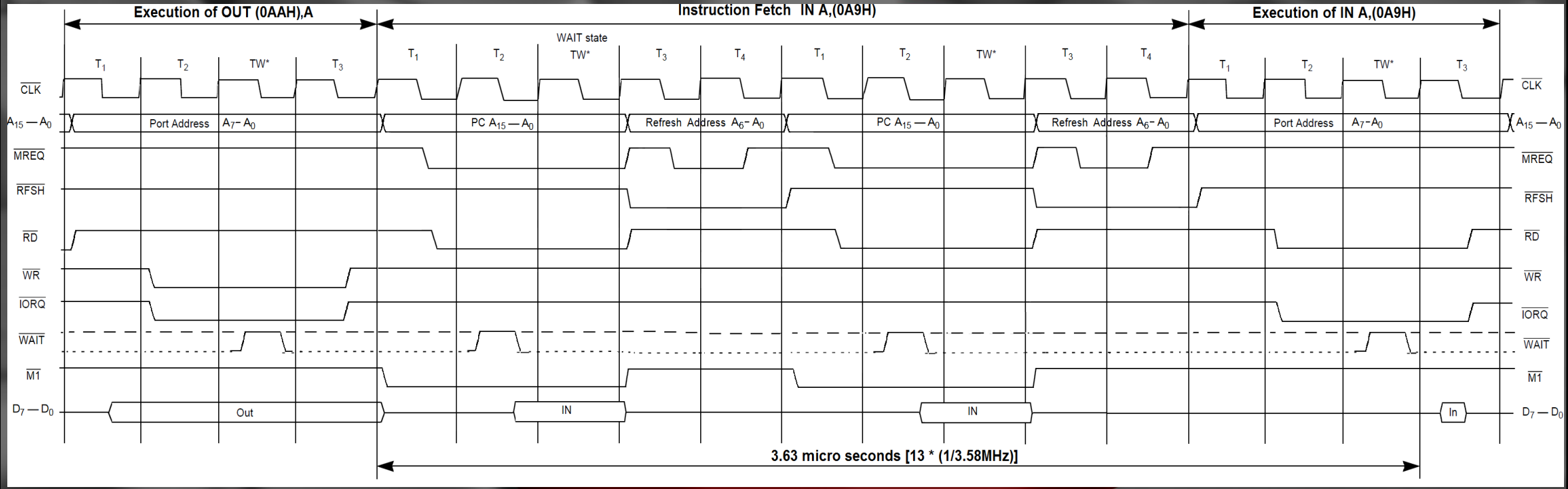
		0	1	2	3	4	5	6	7	8
		Y=0	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6	Y=7	Y=8
FE	X=0	) 0	* 8	^ ~	C c	K k	S s	SHIFT	F4	Space
FD	X=1	! 1	(9	] [	D d	L l	T t	CTRL	F5	CLS HOME
FB	X=2	@ 2	_ -	; ,	E e	M m	U u	GRAPH	ESC	INS
F7	X=3	# 3	+ =	: .	F f	N n	V v	CAPS	TAB	DEL
EF	X=4	$ 4	^ \	? /	G g	O o	W w	CODE/ KANA	STOP	LEFT
DF	X=5	% 5	` ´	> <	H h	P p	X x	F1	BackSpace	UP
BF	X=6	" 6	' ¨	A a	I i	Q q	Y y	F2	SLCT	DOWN
7F	X=7	& 7	Ç ç	B b	J j	R r	Z 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 conector 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.

How MSX works (2/2)



Detailed timming 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 aproximately 3,6µs;

- Obs.: In special circunstancies, 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.

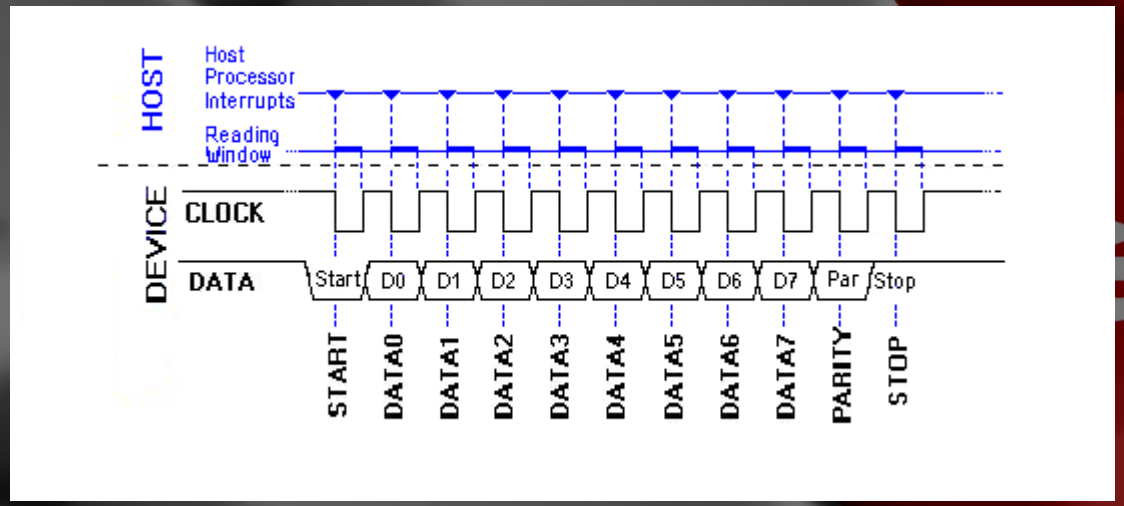
PS/2 Keyboard - Timmings

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

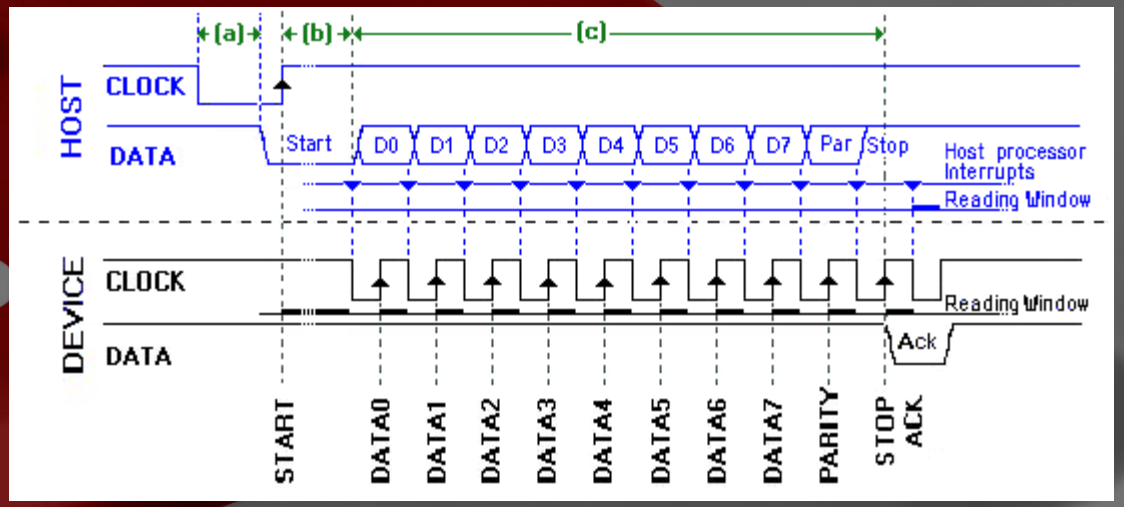
- The clock rate is in range from 10K to 16,7KHz;

Observe the following timming diagrams of:

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



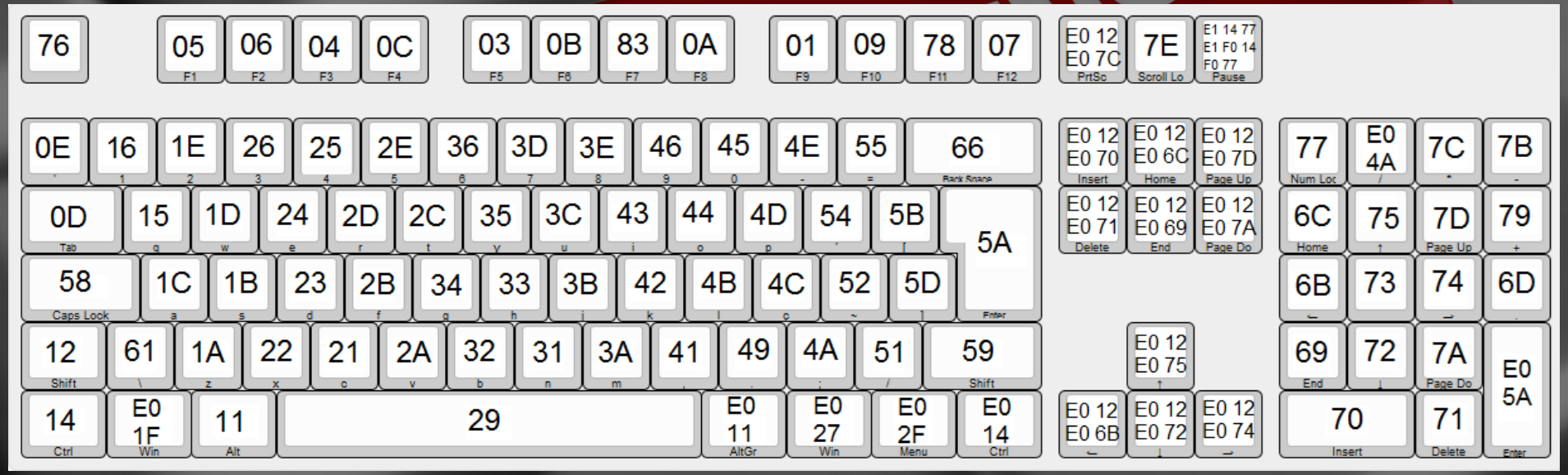
2) From host to keyboard:



- (a) Host forces the clock line to low (0) for at least 100µs;
- (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 by up to 2ms;

PS/2 Keyboard - Layout

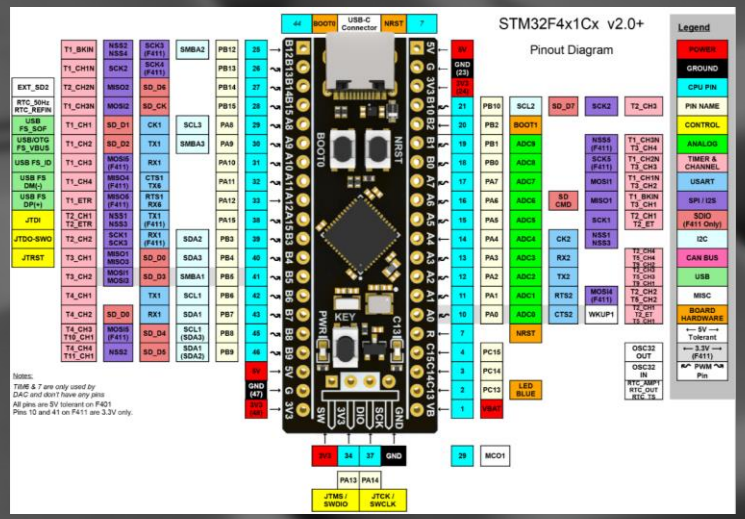
- There are many exclusive MSX keys, like Graph, Code, Select and Stop. That though is also valid for a PC keyboard;
- The scan codes do not have have any pairing (relationships) with keytops mark;
- The scan codes have different sizes: from 1 to 8 bytes;
- The PS/2 keyboard sends Type 2 codes (AT type) 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:

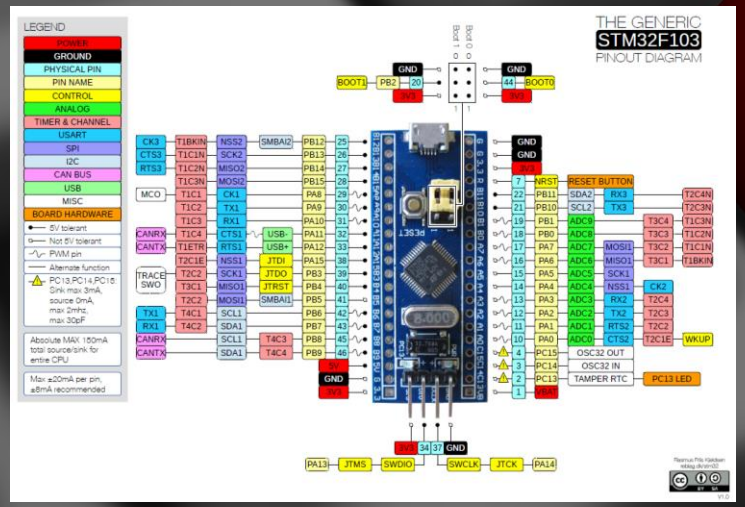


Now

- Enviroment characteristics and considerations
- About getting the right MCU
- Develop enviroment setup
- The electronics
- Firmware features
- Firmware Technical Datasheets

Picking the MCU (Micro Controller Unit)





Features met

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 resourses;
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.

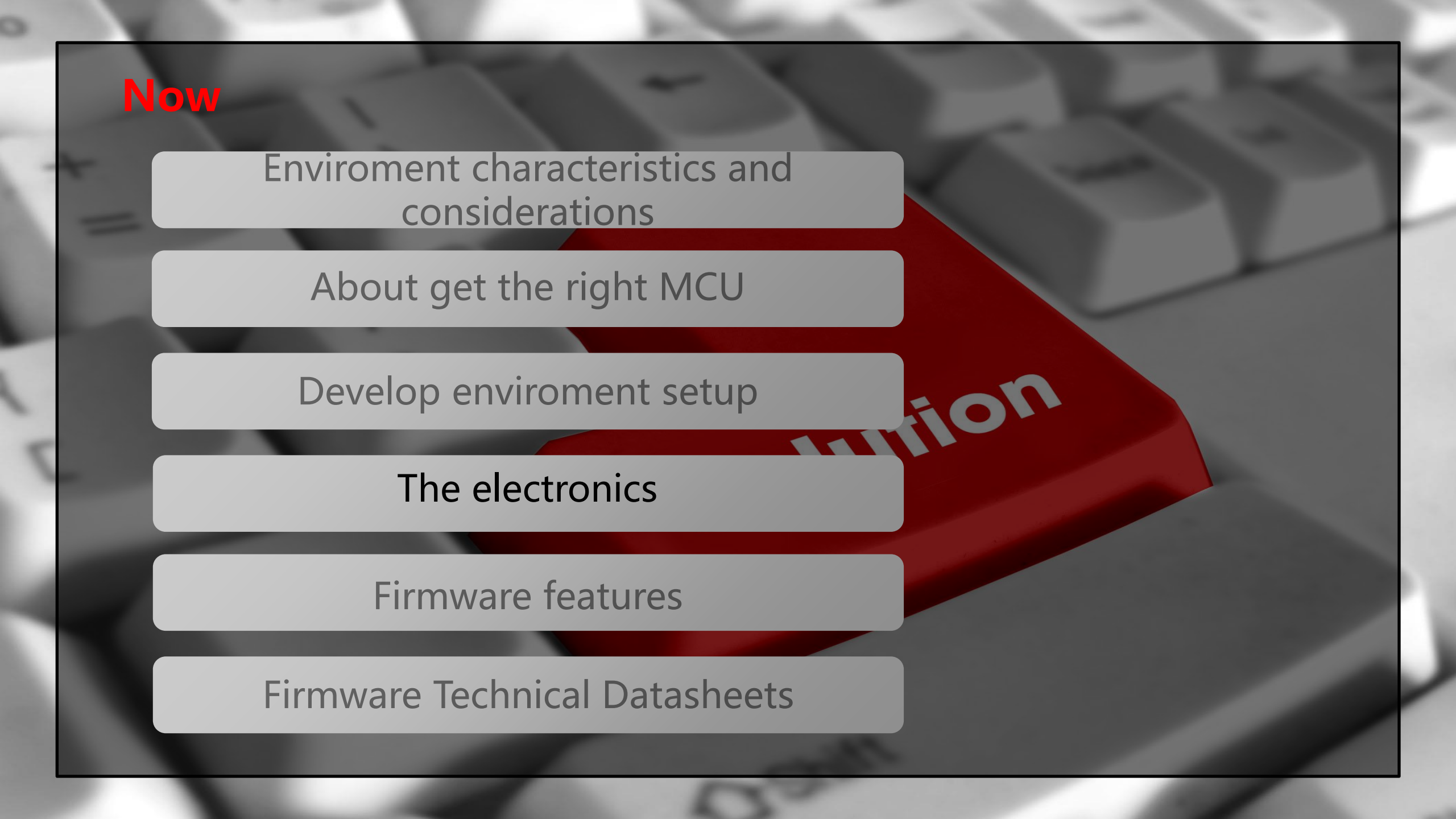
Now

- 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!

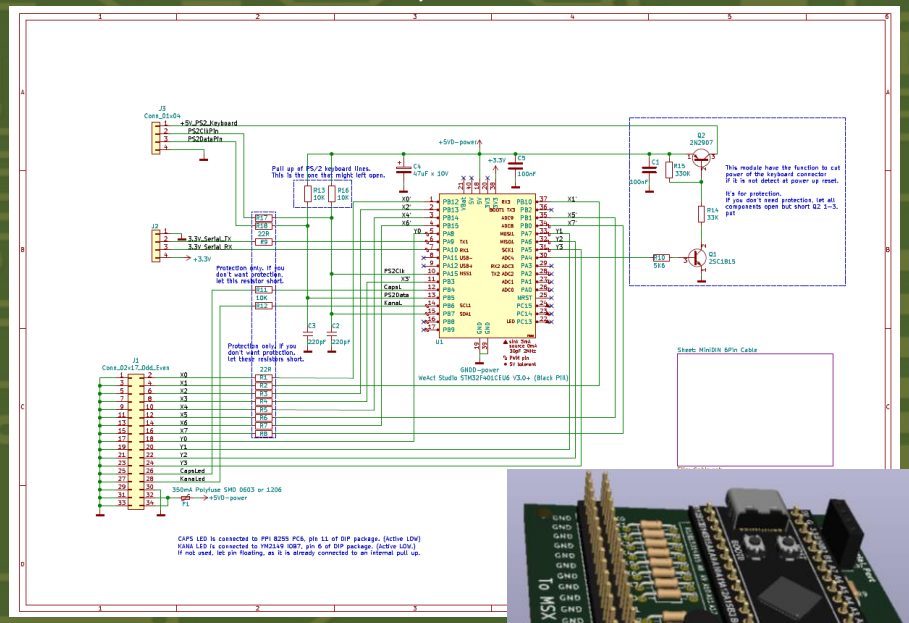

Now
Enviroment characteristics and considerations
About get the right MCU
Develop enviroment setup
The electronics
Firmware features
Firmware Technical Datasheets

The Electronics

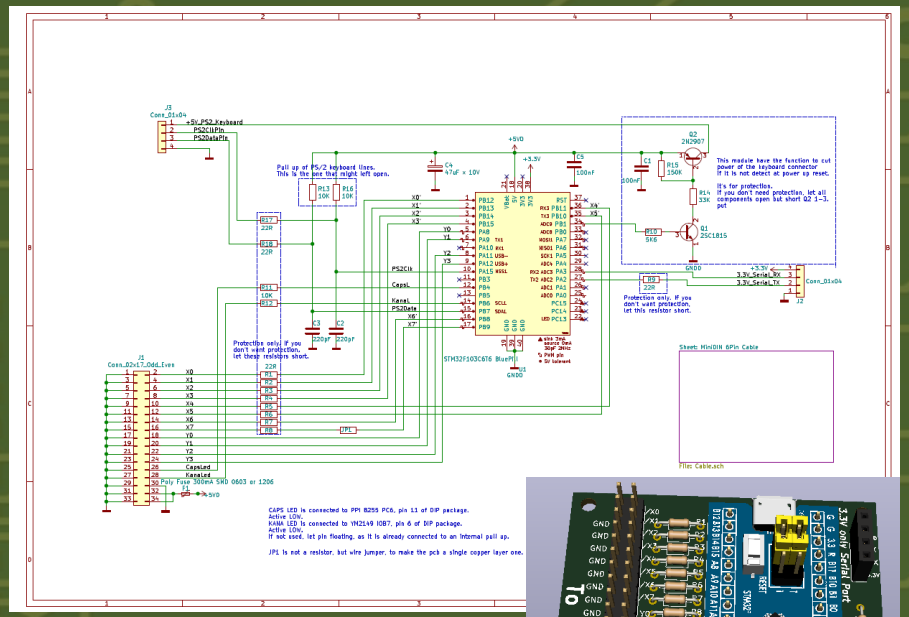
Modules usage comes with advantages:

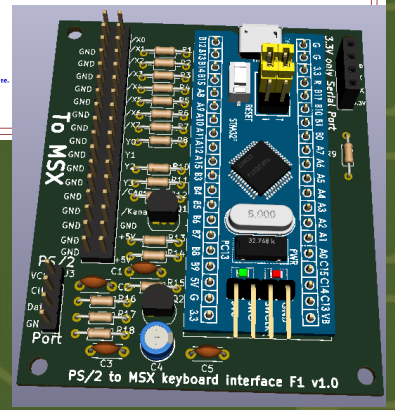
- ARM 3.3V power is resolved from MSX 5V;
- All support circuitry (reset, crystal, SWD, status led, etc) already up;
- 2.54mm DIP form feed, instead of SMD: Easier to match legacy.

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



Black Pill Module Version

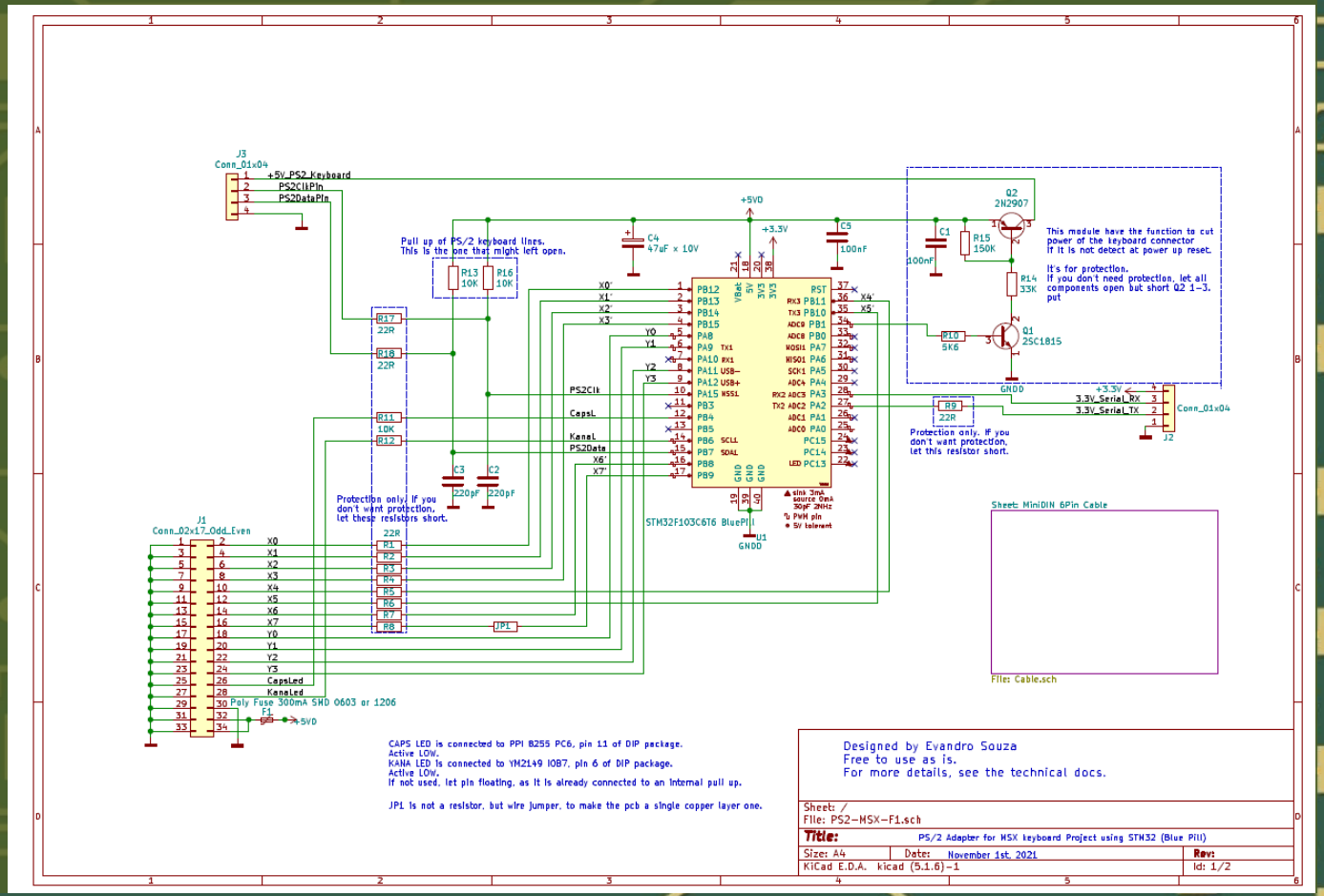




Blue Pill Module Version

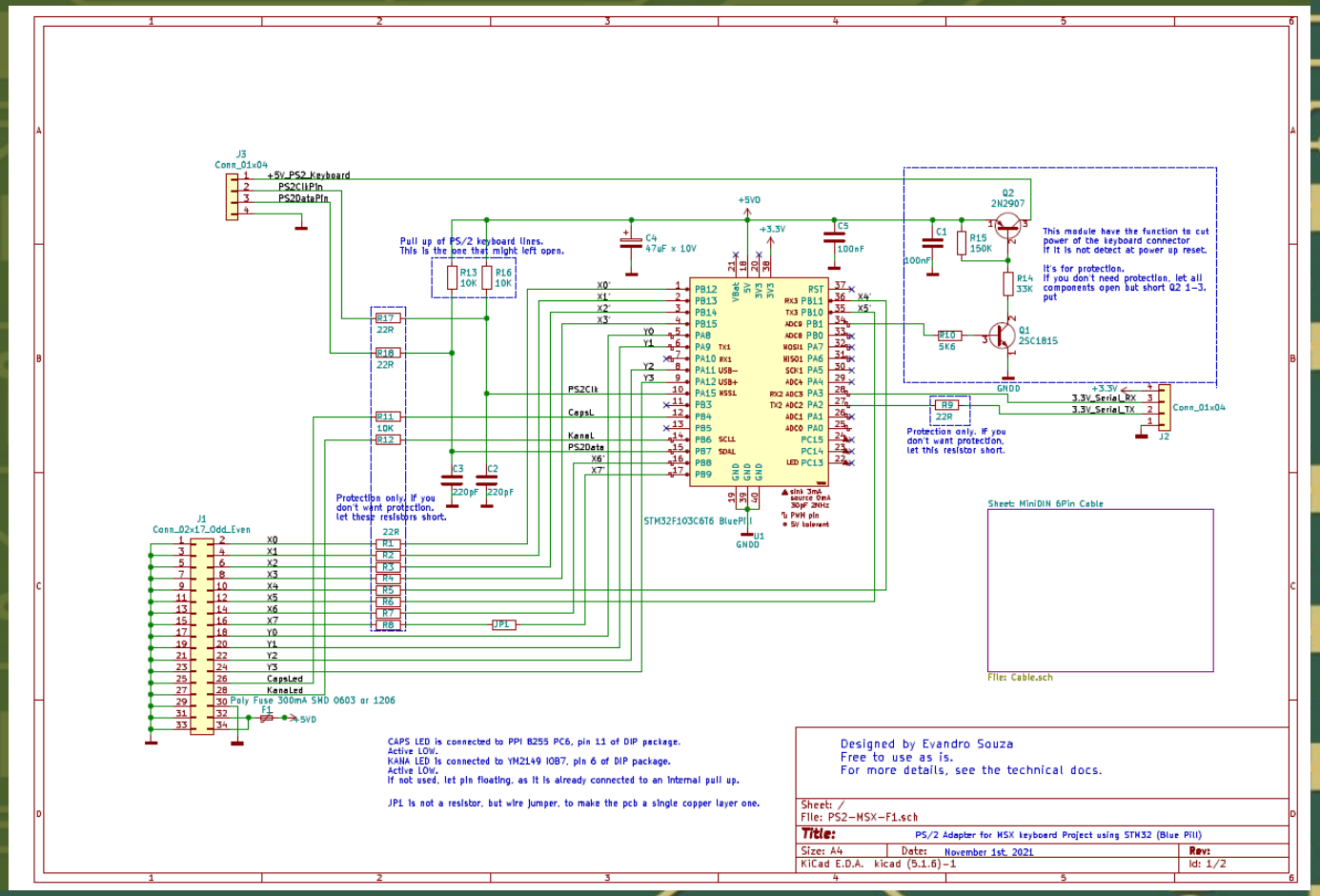
The electronics with Black Pill module

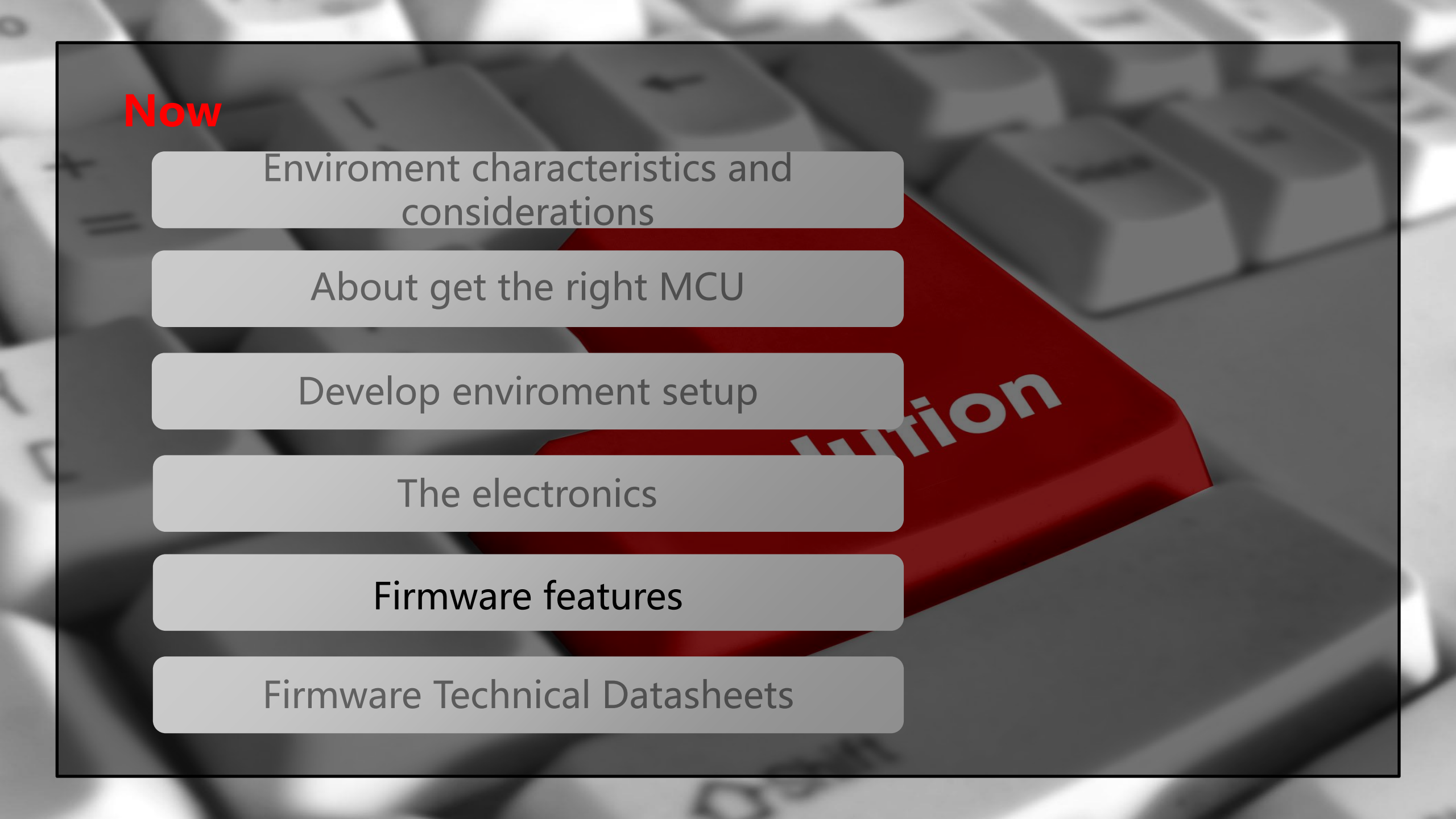
- 100% of external electronics has the function of protecting the PS/2 Keyboard, ARM module and MSX host;
- Powers down the PS/2 interface when keyboard is not detected while booting-up;
- C2 and C3 (the capacitors in parallel with PS2Clk and PS2Data) absorv switching spikes, minimizing false detections of PS/2 changes;
- The mesasured 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.

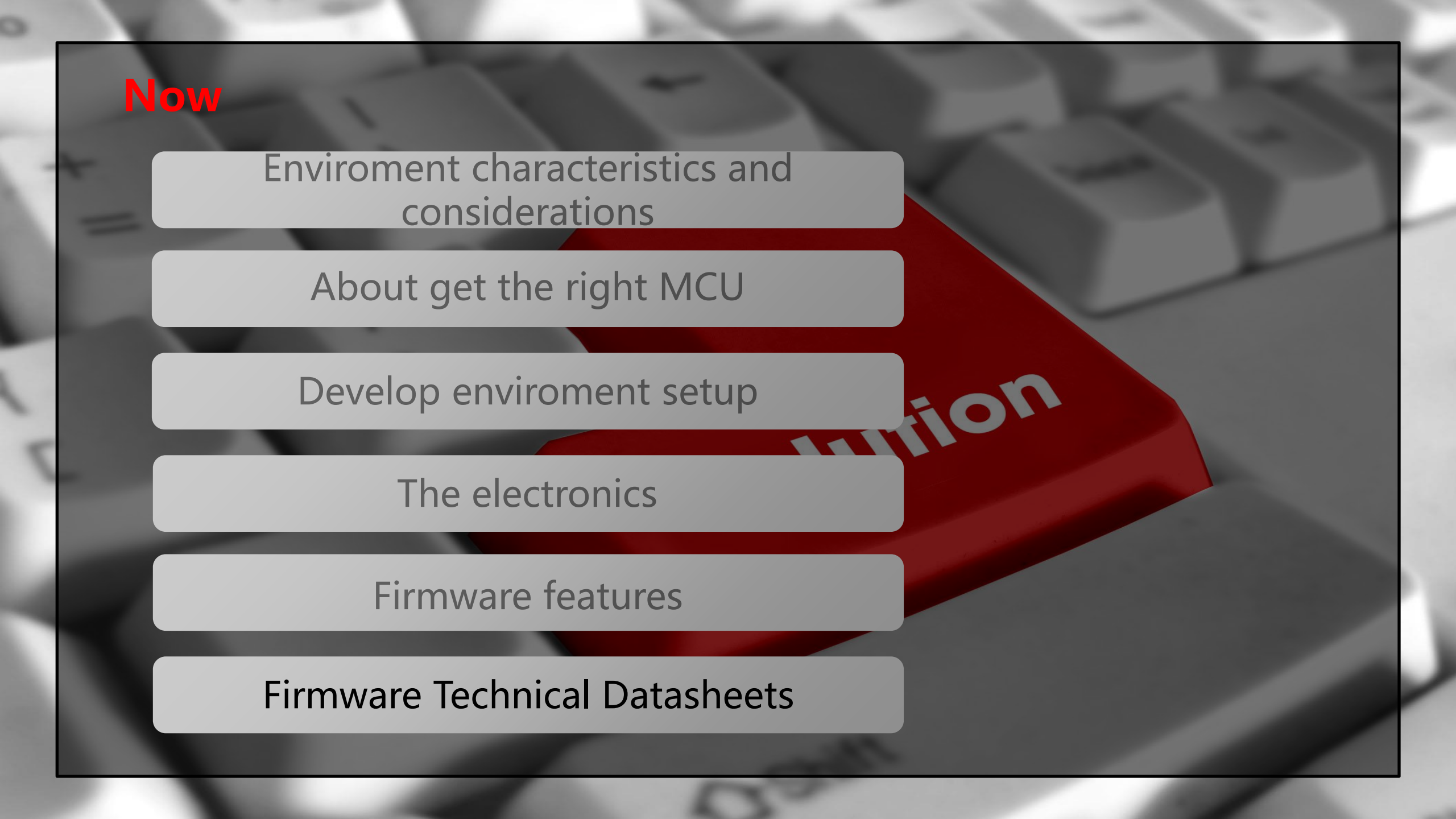
The Database (an Intel Hex text file) is updated via console, with a help of a tty program (eg TIO) , that will guide you through the steps => A 15 seconds operation.

The Converter controls NumLock as PC does, and CapsLock and Scroll Lock leds, this last one mapped from Katakana (or Cyrilic 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 initalized as 1.0s and repeat rate as 2CPS.

The MSX puts colunm (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 phylosofy used is to to 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 Appications) through Microsoft Excel.

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

- The complete source code with Doxygen documentation, open excel Database compiler, schematics, pcb and their gerber files are all availabe 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