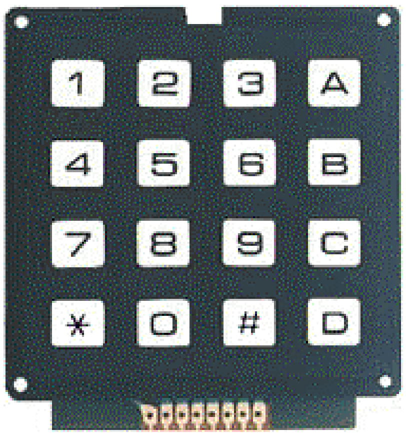
Course on BASCOM AVR - (11)

Theoretic/Practical course on BASCOM AVR Programming.
Author: DAMINO Salvatore.

K E Y B O A R D S (2).



In this example we'll examine a real **Scanning** program for a matrix keyboard, in a complete mode. We'll describe all the necessary solutions that avoids the typical problem of keys electric rebounds, through a time based debouncing.

The program doesn't generate the required timings by using the **Microcontroller** internal **Timer**. In fact this modality certainly should be the best solution but, as it employ also an **Interrupt** management, we decided to describe these features in a dedicated following chapter, with details.

The solution that doesn't use **Interrupt** has a disadvantage: the generated timings are not really accurate and they change according with program flow. However this bad features are not so important, especially in a small program as this one is. On the other side the program source can be immediately read and understood.

A deep attention has been dedicated to program **Flow Chart** development. In order to increase readibility the flow includes either a general structure that summarizes all operations performed by program and some explosions of the most important components parts.

By following with care all the steps of the management, you can learn and understand the reason why of all the choices performed in the complete program, and their consequences in the program source development.

For example in the endless loop of the main program there is a delay equal to debouncing time. Whenever the program is modified and its executes other or different operations, it must be properly re-calibrated.

Example.018. Keyboard Management. Debouncing Modality.

Added Definitions:
None

Added Declarations:
None

Added Instructions:
None

Added Operators:
None

Example Program.018 of **BASCOM AVR** course.

It manages **All Keys** of a **4x4** matrix keyboard.

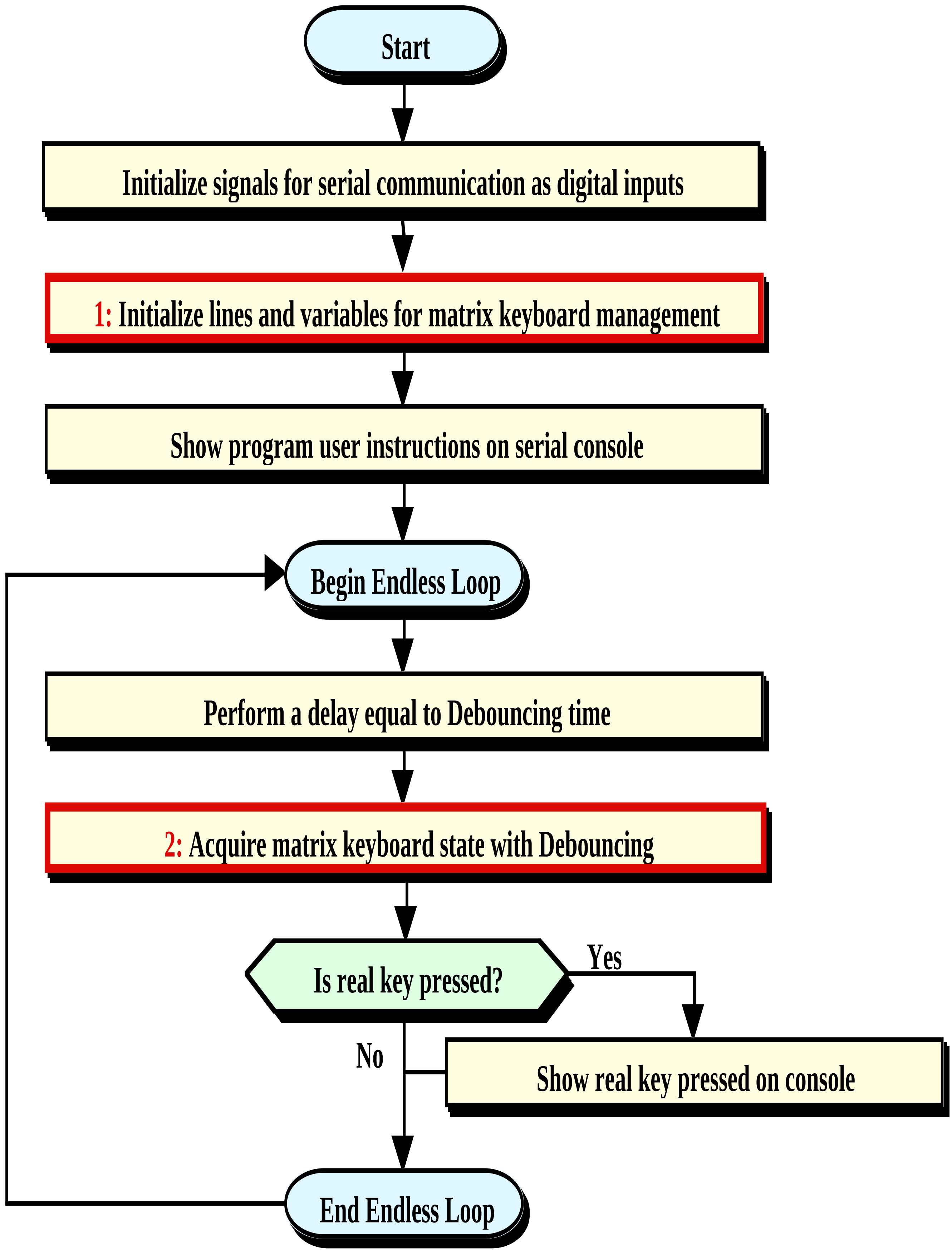
The program continuesly acquires the state of **16** keys connected to matrix keyboard available on **GMM TST3** and it transmits them on serial line. The acquisition is performed with **Debouncing**, without **Autorepeat** and without **Times Controls**.

The visualization of pressed keys is performed on a serial console provided of monitor and it must communicate with a fixed physical protocol at **19.200 Baud**, **8 Bit x chr**, **1 Stop bit**, **No Parity**.

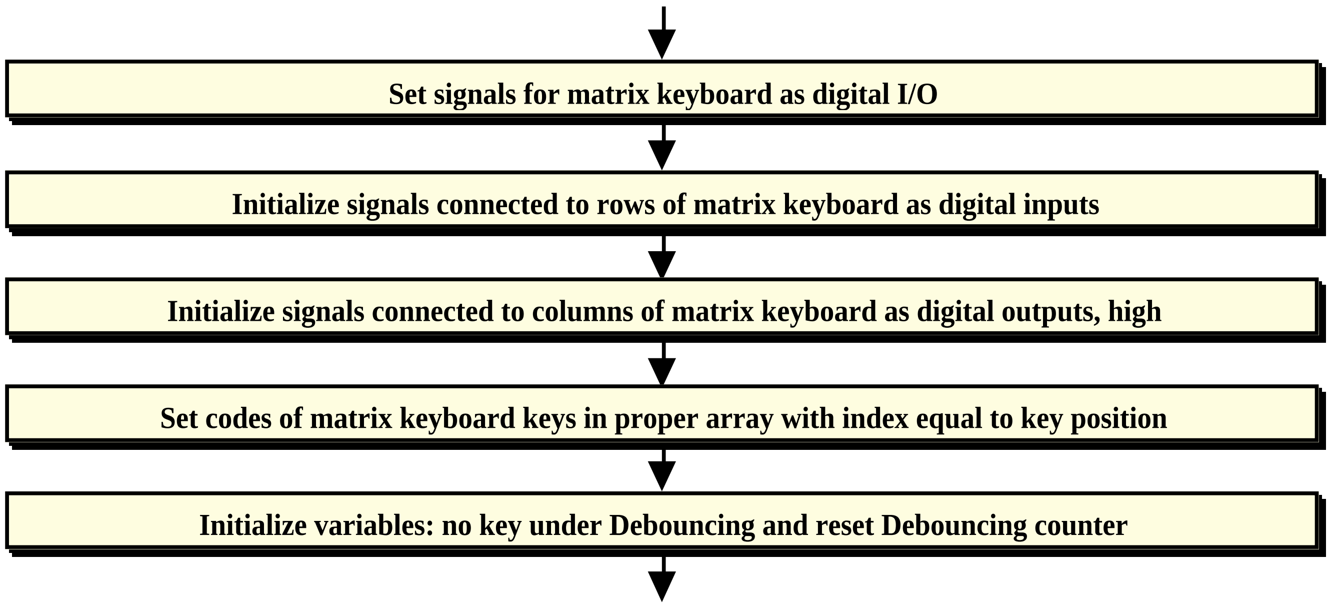
This console can be another system capable to support a serial **RS 232** communication. In order to simplify the use it can be used a **PC** provided of one **COMx** line, that execute a terminal emulation program as **HYPERTERMINAL** or the homonym modality provided by **BASCOM AVR** (see **IDE** Configuration).

The program works only when the **GMM AM08** is mounted on **Z1** socket of **GMM TST3**!!

Inside the program the row and column terms refers to electric diagram of matrix keyboard, not to its physical format!!

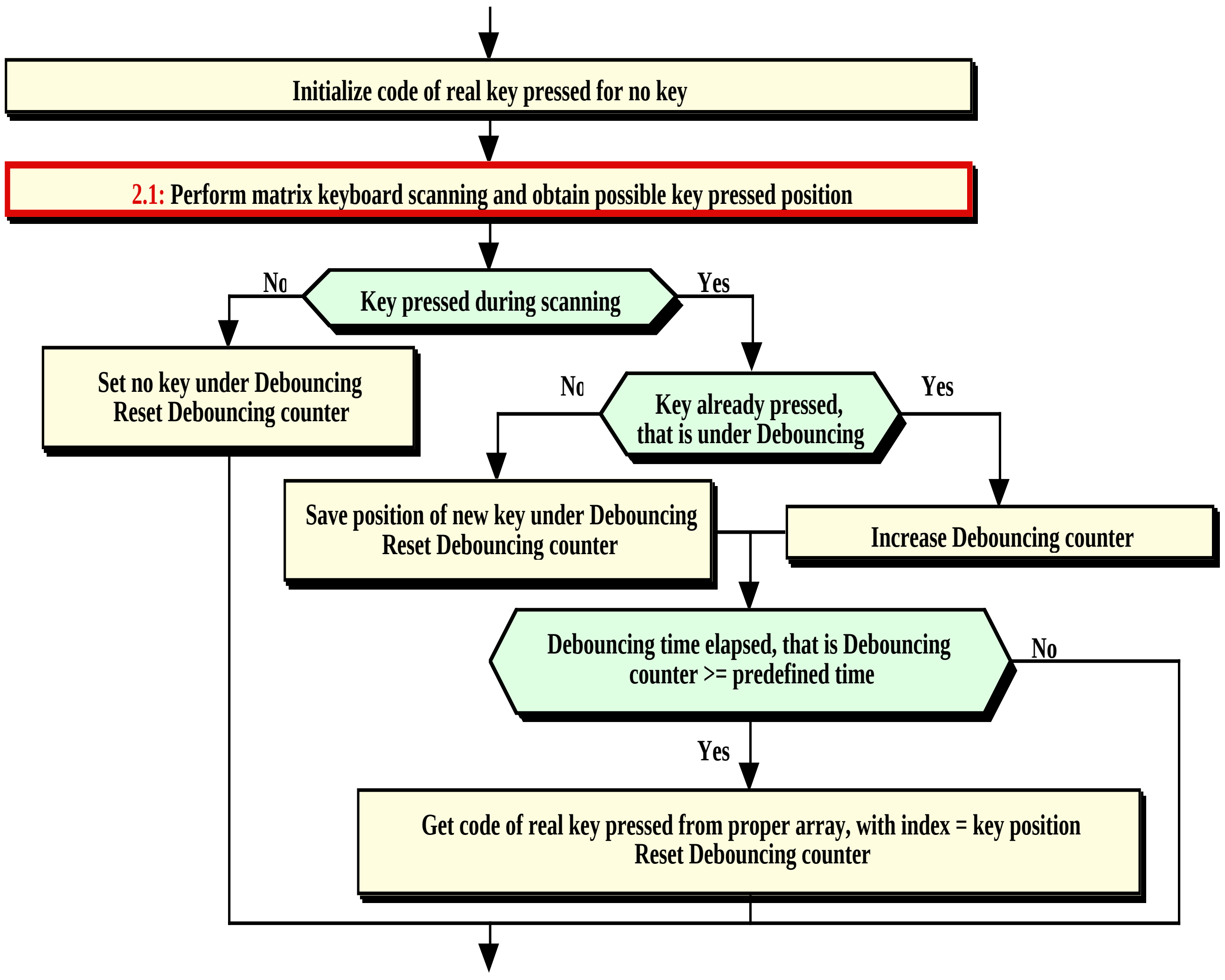

Start
Initialize signals for serial communication as digital inputs
1: Initialize lines and variables for matrix keyboard management
Show program user instructions on serial console
Begin Endless Loop
Perform a delay equal to Debouncing time
2: Acquire matrix keyboard state with Debouncing
Is real key pressed?
Yes
Show real key pressed on console
No
End Endless Loop

Flow Chart Diagram of the Program.



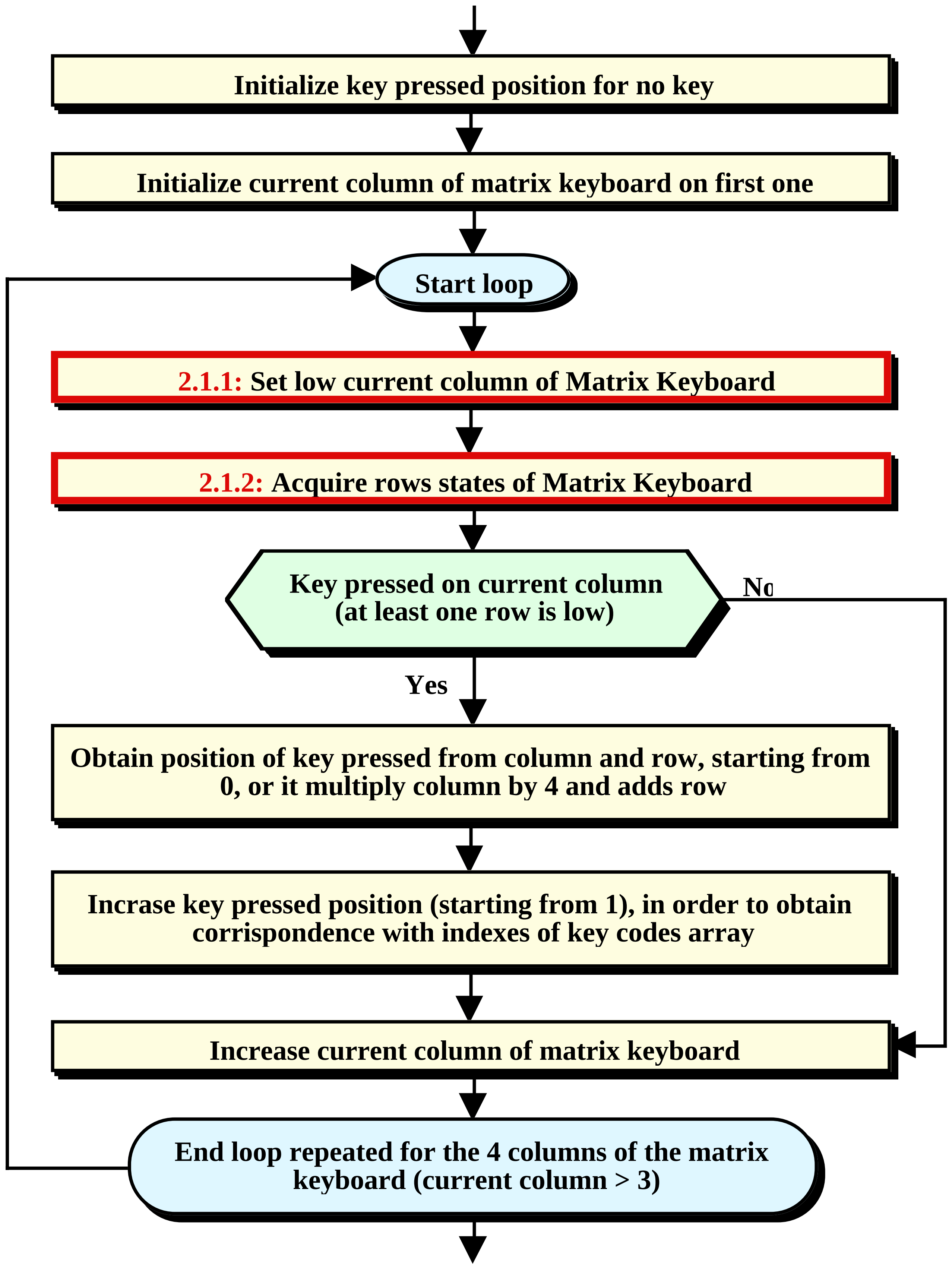
Set signals for matrix keyboard as digital I/O
Initialize signals connected to rows of matrix keyboard as digital inputs
Initialize signals connected to columns of matrix keyboard as digital outputs, high
Set codes of matrix keyboard keys in proper array with index equal to key position
Initialize variables: no key under Debouncing and reset Debouncing counter

1: Initialize Lines and Variables for Matrix Keyboard Managemen.

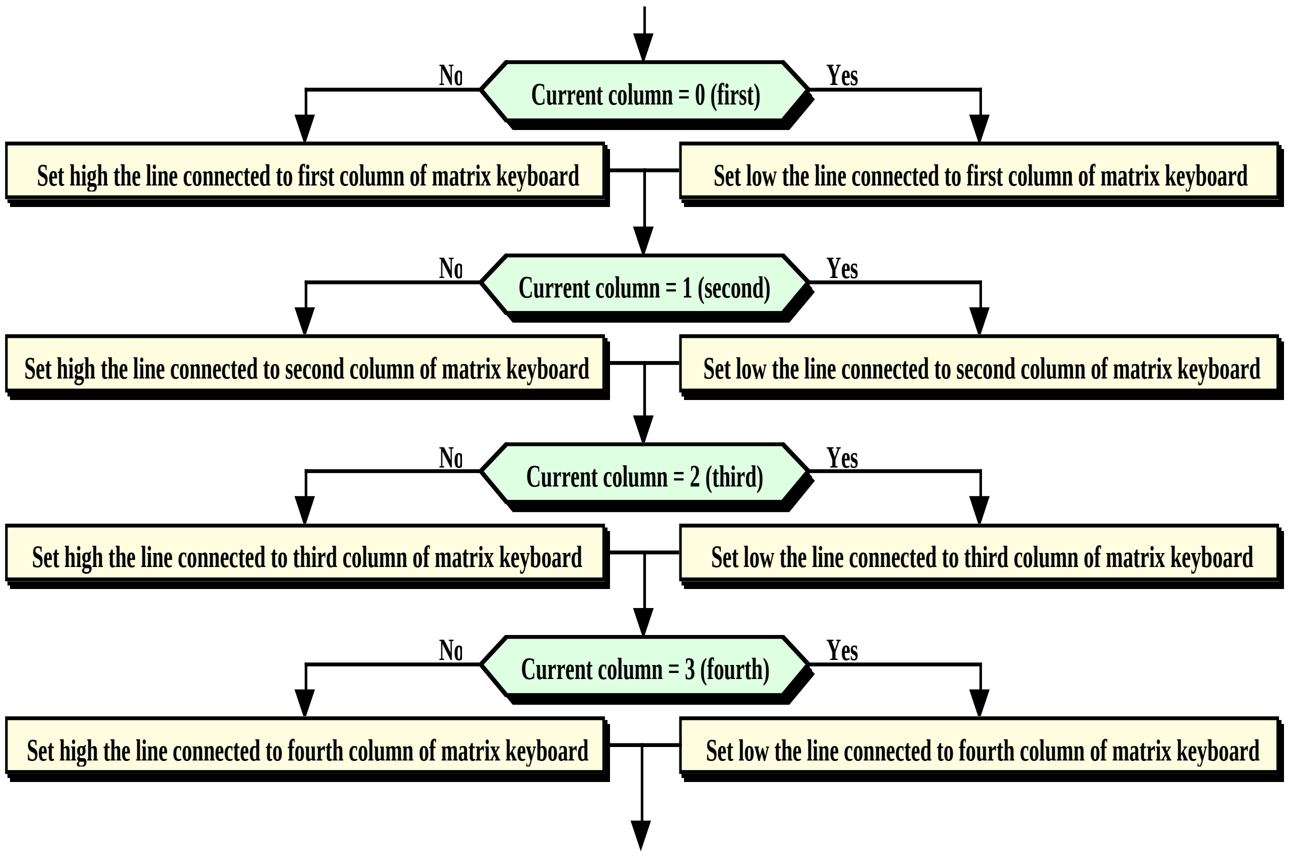


Initialize code of real key pressed for no key
2.1: Perform matrix keyboard scanning and obtain possible key pressed position
Key pressed during scanning
No
Yes
Set no key under Debouncing
Reset Debouncing counter
Key already pressed, that is under Debouncing
No
Yes
Save position of new key under Debouncing
Reset Debouncing counter
Increase Debouncing counter
Debouncing time elapsed, that is Debouncing counter >= predefined time
No
Yes
Get code of real key pressed from proper array, with index = key position
Reset Debouncing counter

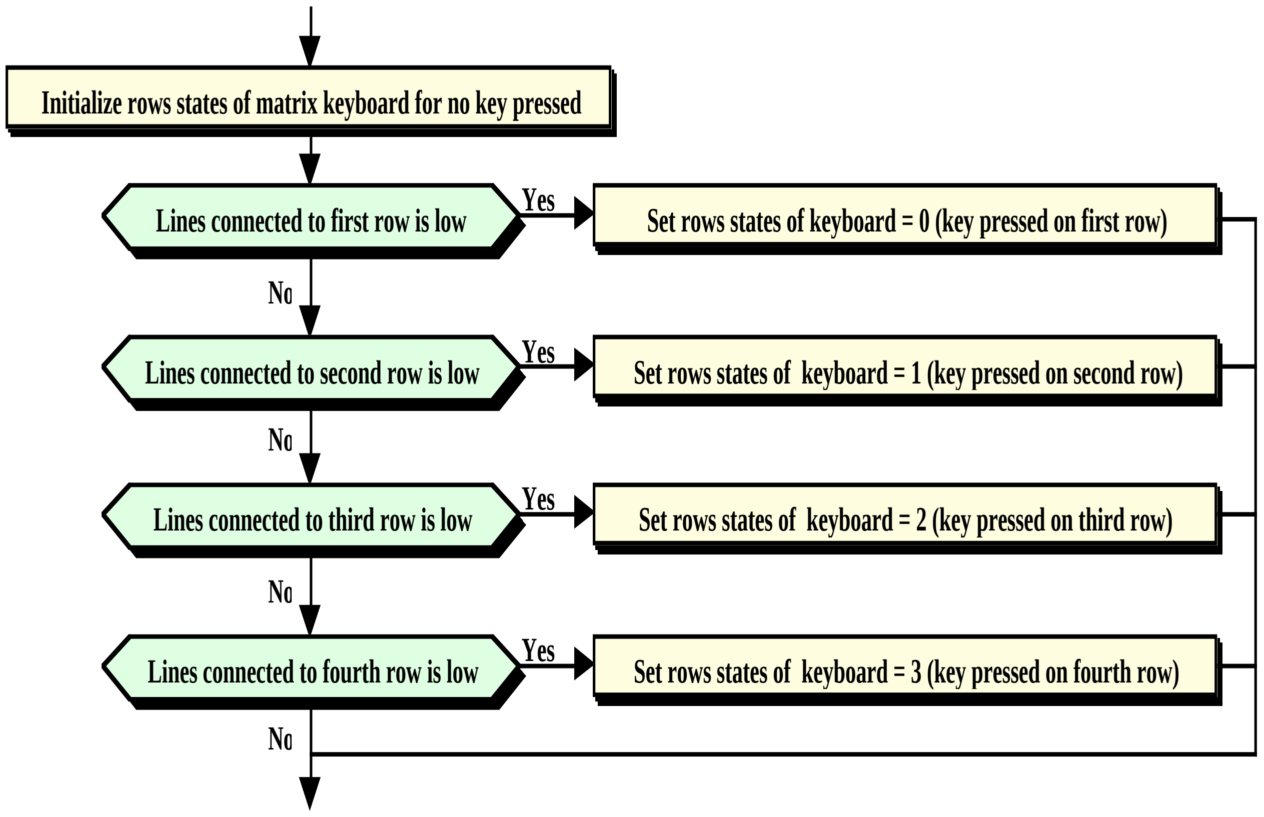
2: Acquire Matrix Keyboard State With Debouncing.


Initialize key pressed position for no key
Initialize current column of matrix keyboard on first one
Start loop
2.1.1: Set low current column of Matrix Keyboard
2.1.2: Acquire rows states of Matrix Keyboard
Key pressed on current column (at least one row is low)
No
Yes
Obtain position of key pressed from column and row, starting from 0, or it multiply column by 4 and adds row
Incrase key pressed position (starting from 1), in order to obtain corrispondence with indexes of key codes array
Increase current column of matrix keyboard
End loop repeated for the 4 columns of the matrix keyboard (current column > 3)

2.1: Perform Matrix Keyboard Scanning and Obtain Possible Key Pressed Position.


Current column = 0 (first)
No
Yes
Set high the line connected to first column of matrix keyboard
Set low the line connected to first column of matrix keyboard
Current column = 1 (second)
No
Yes
Set high the line connected to second column of matrix keyboard
Set low the line connected to second column of matrix keyboard
Current column = 2 (third)
No
Yes
Set high the line connected to third column of matrix keyboard
Set low the line connected to third column of matrix keyboard
Current column = 3 (fourth)
No
Yes
Set high the line connected to fourth column of matrix keyboard
Set low the line connected to fourth column of matrix keyboard

2.1.1: Set low Current Column of Matrix Keyboard .


Initialize rows states of matrix keyboard for no key pressed
Lines connected to first row is low
Yes
Set rows states of keyboard = 0 (key pressed on first row)
No
Lines connected to second row is low
Yes
Set rows states of keyboard = 1 (key pressed on second row)
No
Lines connected to third row is low
Yes
Set rows states of keyboard = 2 (key pressed on third row)
No
Lines connected to fourth row is low
Yes
Set rows states of keyboard = 3 (key pressed on fourth row)
No

2.1.2: Acquire Rows State of Matrix Keyboard.