# Chapter 8

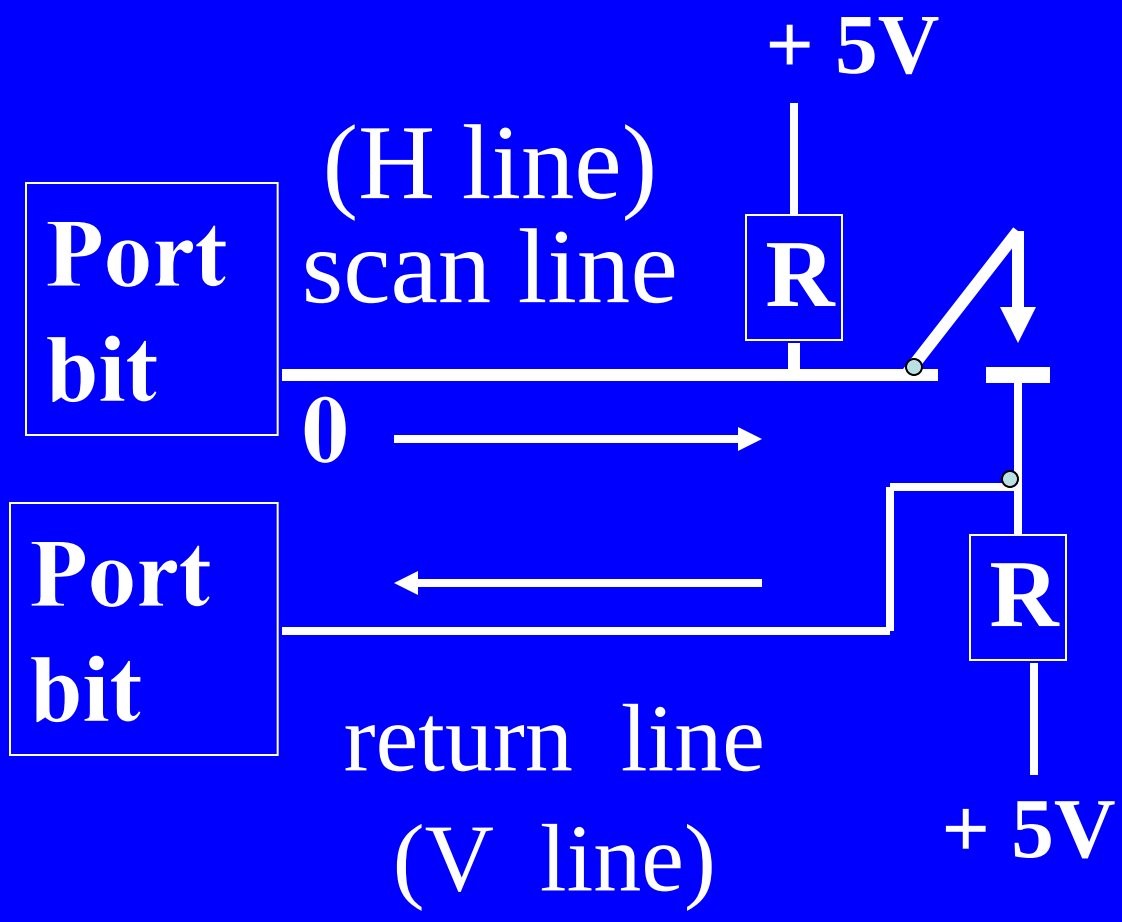
# Digital and Analog Interfacing Methods

# Lesson 1

# Key, keypad and keyboard

# Single Key Interface

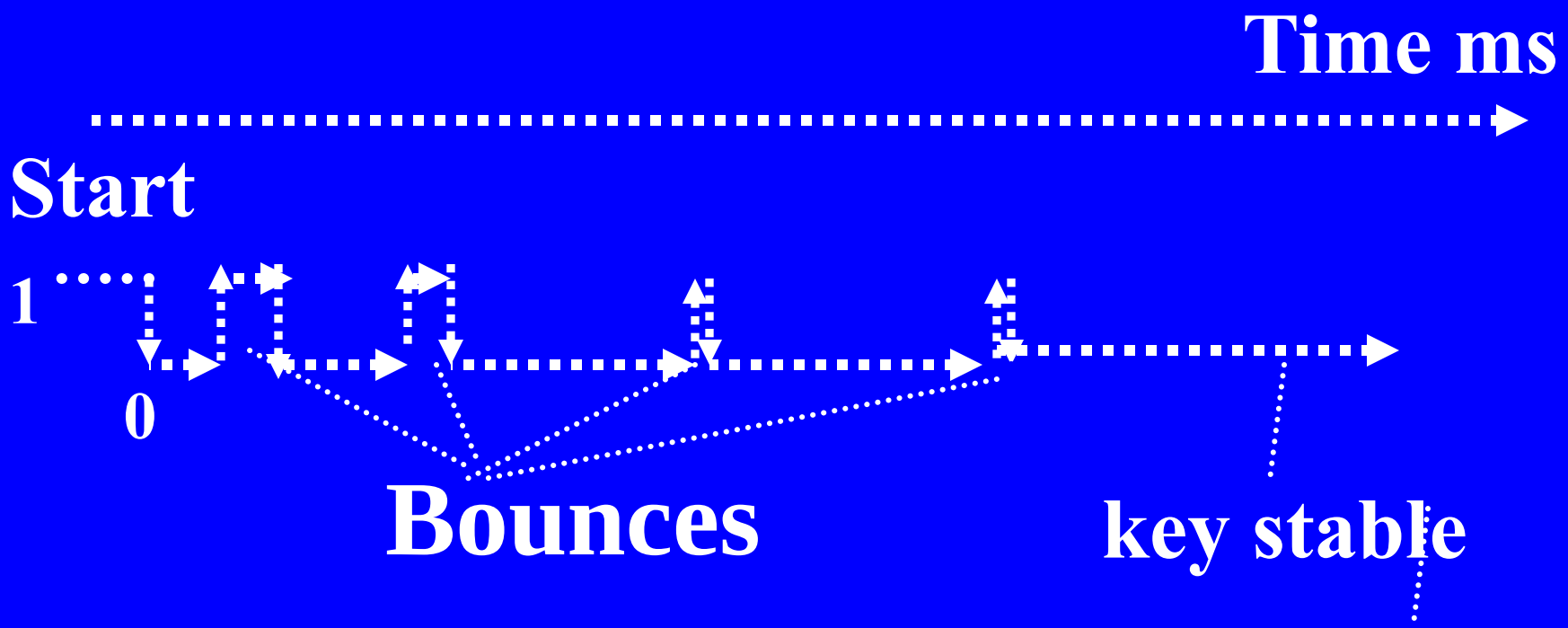
# Single key Interfacing to a Port



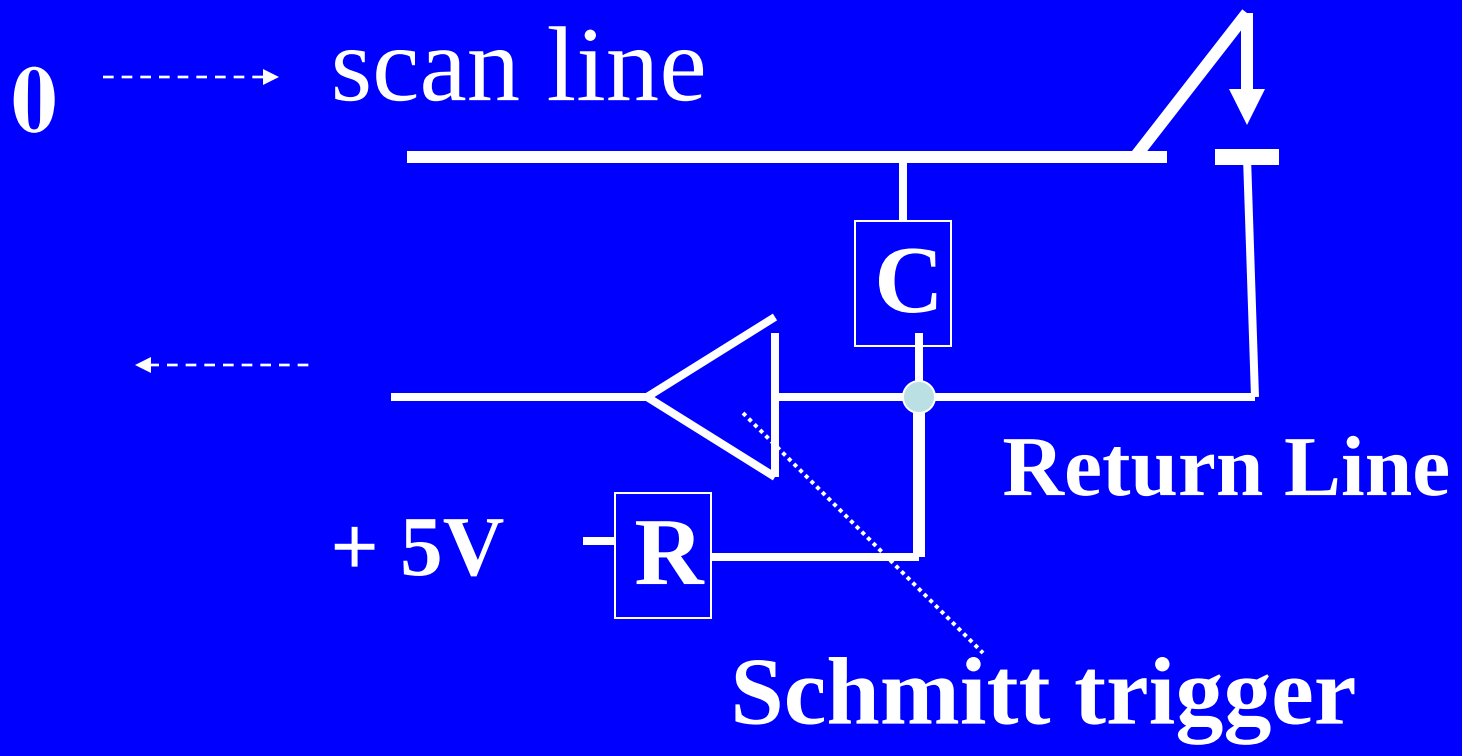
# Scan (H) and Return (V) Lines

- Scan start by sending Send_out_bit = 0 and
- If return line Recv_in_bit becomes 0, it means key (switch) pressed
- If return line Recv_in_bit remains 1, it means key (switch) not pressed

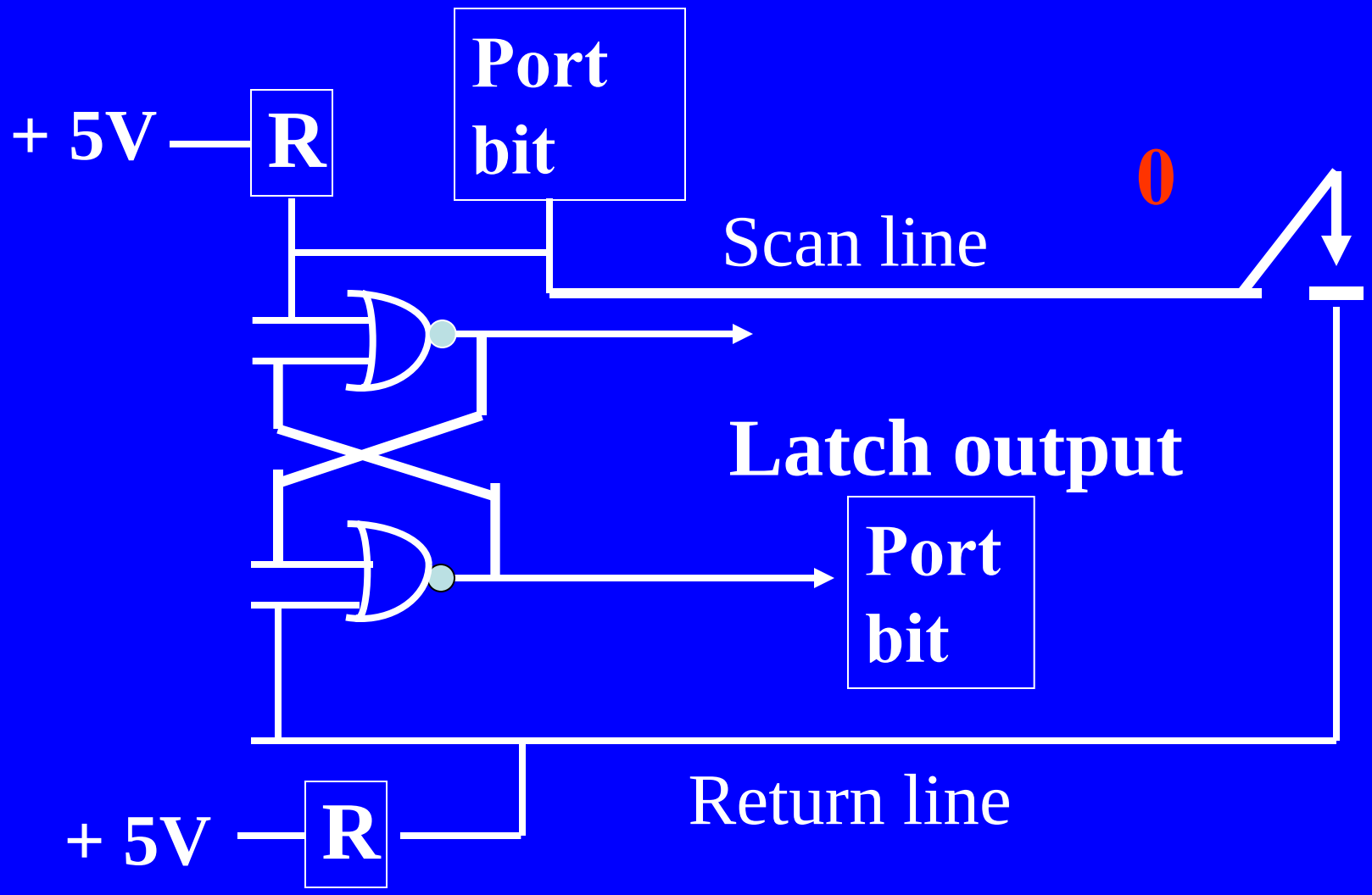
# Return line after scan line forced at 0



# Hardware debouncing



# Hardware debouncing



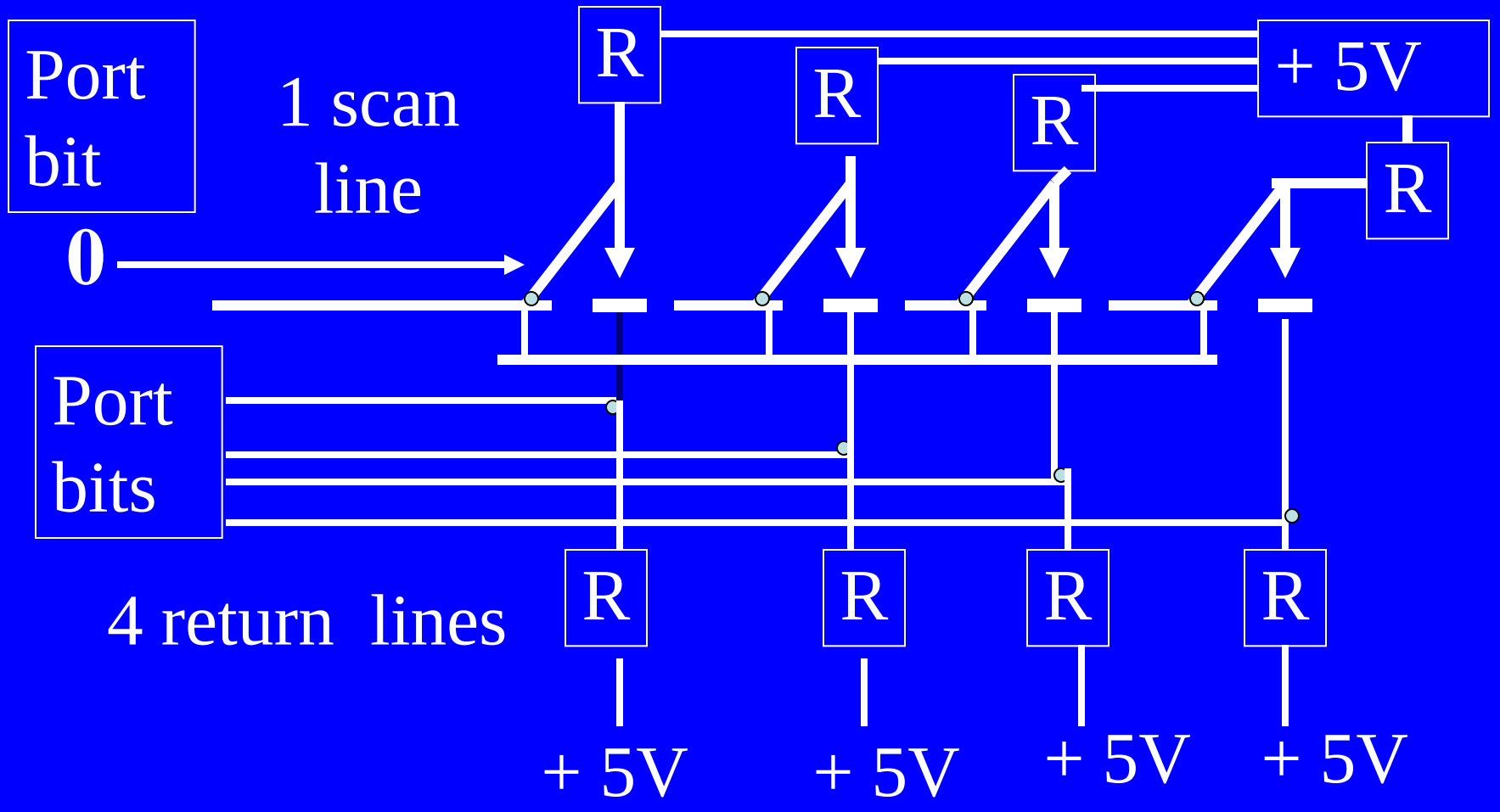
# Key debouncing routine

# Software Debouncing Algorithm

- Assume t > 5 ms required to settle the bounce and Assume 0 acceptable if no bounce for consecutive 5 ms
- Check 0 consecutively after period t > 5 ms from scan start
- If 5 times 0 is found then key accepted as pressed and Recv_in_bit accepted = 0
- Refer text for details of algorithm

# Interfacing an array of keys

# Key array to a Port

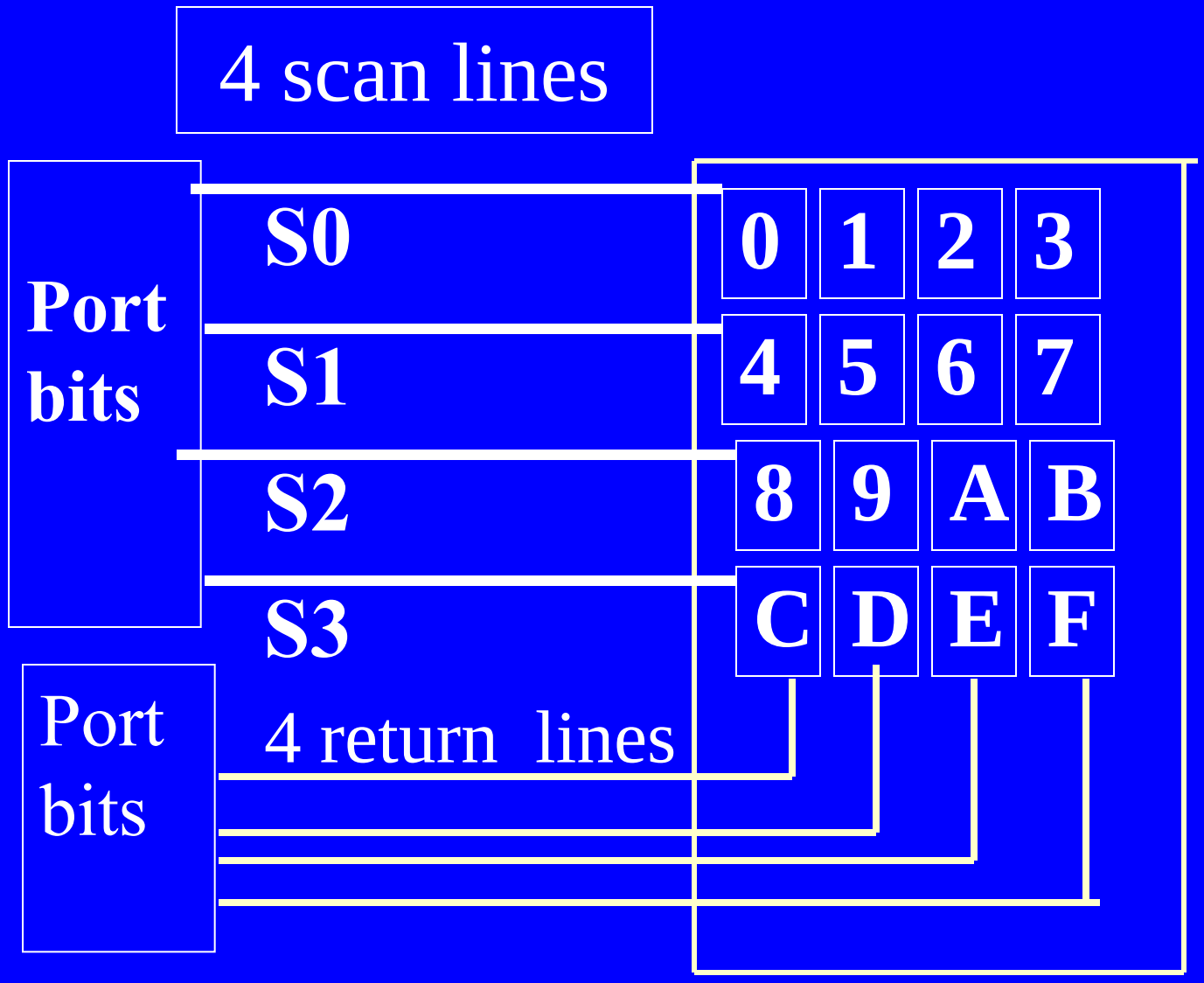


# 4-Keys Software Debouncing

- Assume t > 5 ms required to settle the bounce. Assume 0 acceptable if no bounce for consecutive 5 ms
- Check each return line for 0 consecutively after period t > 5 ms from scan start
- If 5 times 0 is found then key accepted as pressed and Recv_in_bit accepted = 0
- Refer text for details of algorithm

# Interface to a Keypad Interface

# Keypad of 16 keys



# 16-Keys Cyclic scan

- Four scan line are given outputs 1110, 1101,1011, 011 cyclically at intervals periods $4 \times \Delta T$ and the four return lines are read within the intervals of $\Delta T$
- Each line reading is accepted as the port inputs after an algorithm debounces each line and 0 is found for consecutive 5 ms after 5 ms from the scan start.

# Keyboard Interface

# **64-Keys plus** Shift and control two keys **Board**

- Assume 64 keys
- Shift and control two keys

# 64-Keys Cyclic scan

- 64 keys need 8 scan lines and 8 return lines
- Three bits used to generate 8 scan lines using decoder for encoded bits for scan
- Three encoded scan lines at the port generate as per a 3-bit counter output, generating sequences 000, 001,...,110, 111
- The 3 to 8 decoder output is 0 only at one of the 8 internal scan lines S0,S1, S2, ..,S7 .

# 64-Keys Cyclic scan

- The scan is at intervals of $\Delta T$ when counter is counting at the $\Delta T^{-1}$
- During the successive interval $\Delta T$ the return lines are internally debounced by hardware and encoded for sending the 3 port input bits.

# 64-Keys return lines

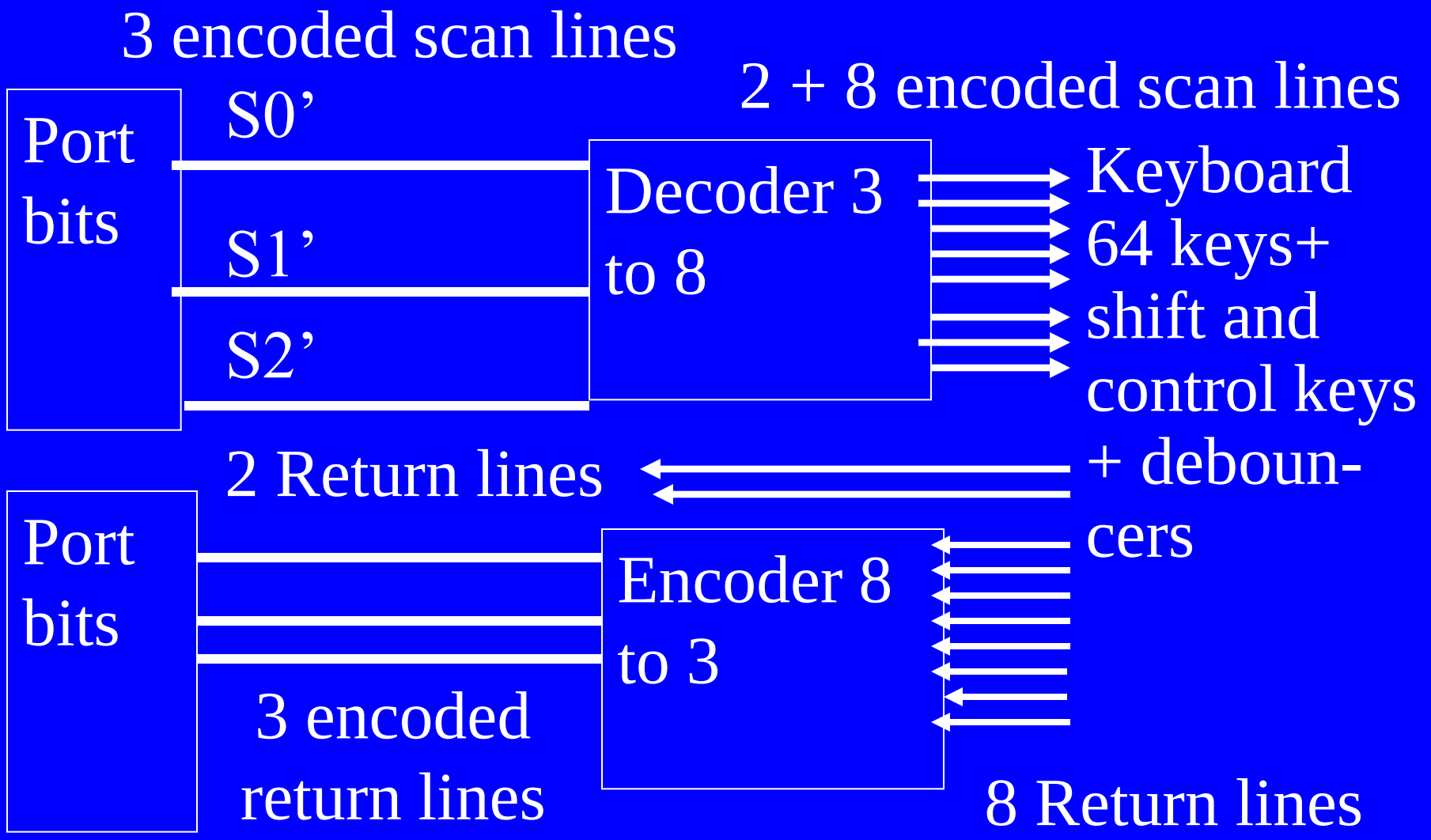
- 8 return lines
- Three bits receive after encoding 8 return lines using an encoder for return lines
- The 8 to 3 encoder input is 000, 001, .. 110, 111 as per which of the 8 internal return lines R0,R1, R2, .., R7 is at 0 after internal debouncing.

# Shift and control two keys Scan and **Return lines**

- 2 scan lines connect to Shift and control two keys.
- Two return lines connect from Shift and control two keys
- 1 scan line used with a decoder 1 to 2
- 1 return line used with an encoder 2 to 1

# Keyboard 64 keys + shift and control keys

## Internal circuit

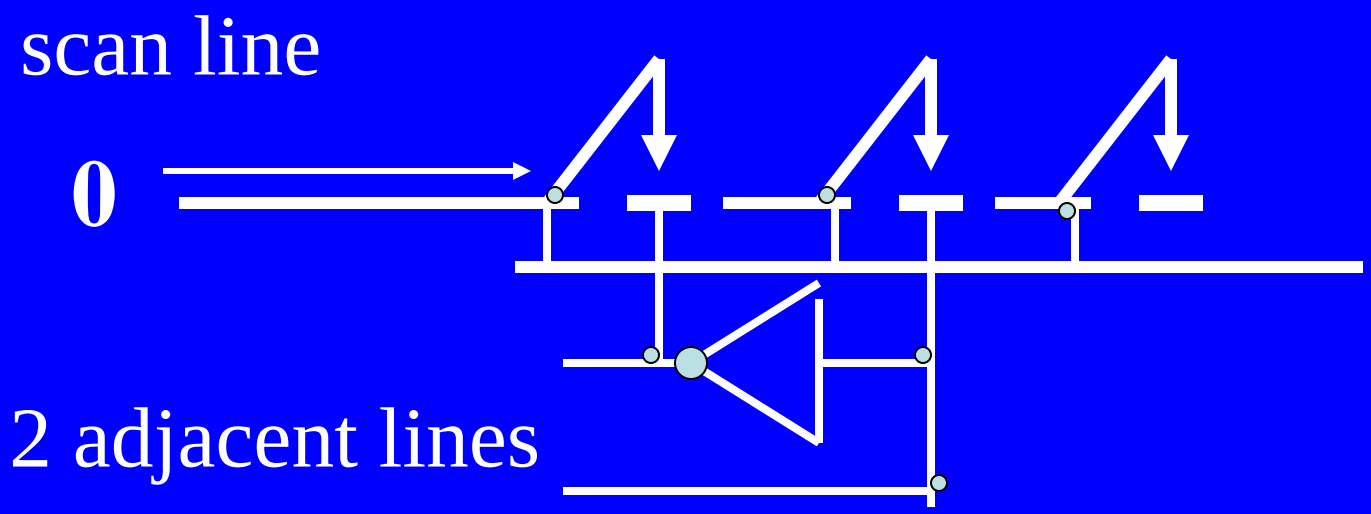


# N-Key rollover and Repeated Character

- Several keys read in quick successions in a FIFO
- A keyboard controller or software routine takes account of the rollover

- After one scan cycle, if a key is again found pressed, the code for it sent in the FIFO

# Two keys lockout - (only)recognize first pressed key




Accidental two adjacent key- press protection

# Keyboard Controller

• Instead of using MCU ports, a keyboard controller (KC) does the internal scan counting and scanning KC debounces and encodes the return lines, which connects as address bits to an internal ROM

# Keyboard Controller

- The ROM gives ASCII code for each pressed key or each grouping of two or three pressed keys
- KC then sends at UART serial port the ASCII code at baud ~ 1200 baud per second)

# 105 keys keyboard with shift-key and other keys

- Shift key and other control keys separately scanned.
- Return lines of these are also the input address bits to ROM
- ROM generates ASCII code for UART port for these keys also

# Keyboard Circuit

- Internal ROM- return lines address inputs and ASCII codes as data output
- Internal converter to return on a serial UART line at ~1200 baud

# Summary

# We learnt

- Each key has a scan input and return output
- Each key has a bounce
- Each key needs Hardware debouncing
- Alternative Software debouncing

# We learnt

- Keypad encoded scan inputs
- Encoded or not encoded return inputs
- Protection from two adjacent keys accidental contact

# We learnt

- Keyboard internal encoded scan inputs generator and internal encoded return outputs,that are input addresses to internal ROM
- ROM converts retru line inputs to ASCII codes
- Internal UART interface sends output to serial port