Keyboard driver presentation

KEYBOARD DRIVER PRESENTATION

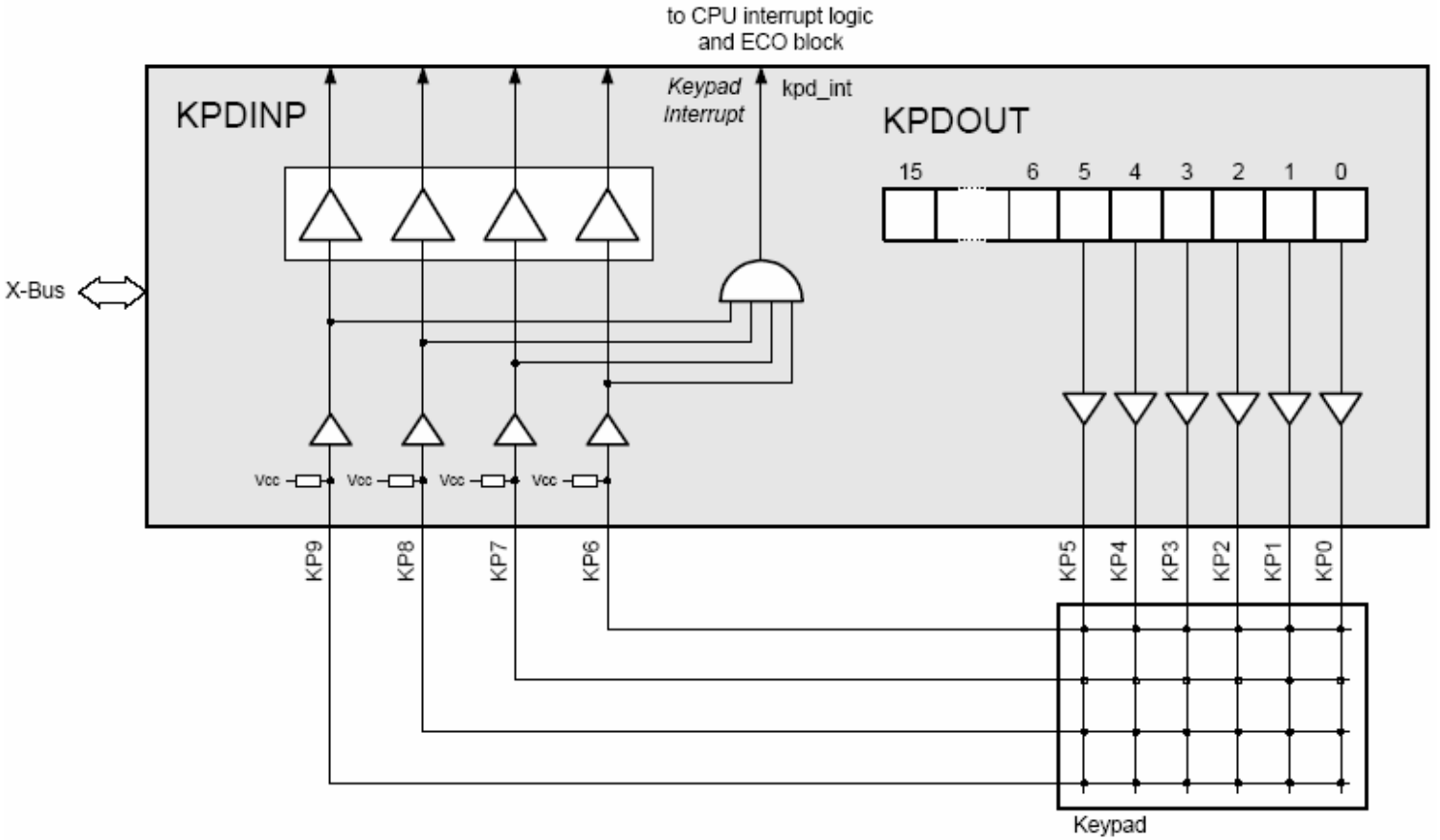
NEONSEVEN

Purpose of the keyboard driver

- Detect key press and release, perform key mapping and send the information (SDL Signal) to the MMI
- Handle open/close flip and send proper indication to the MMI (SDL Signal) (Only on BP30 platform)
- Handle hook key press and release
- Provide key press simulation API to other SW modules

EGoldRadio Keypad interface

- The EGoldRadio keypad interface is a 10-bit port with 4 input and 6 output pins plus two 16-bit interface registers (KPDINP and KPDOUT)



EGoldRadio Keypad interface

- The levels at the input pins KP(9:6) are loaded into KPDINP with a read access
- The levels of the output pins KP(5:0) are loaded into KPDOUT with a write access.
- Pressing a key directly generates an interrupt to the MCU, releasing this key generates an additional interrupt

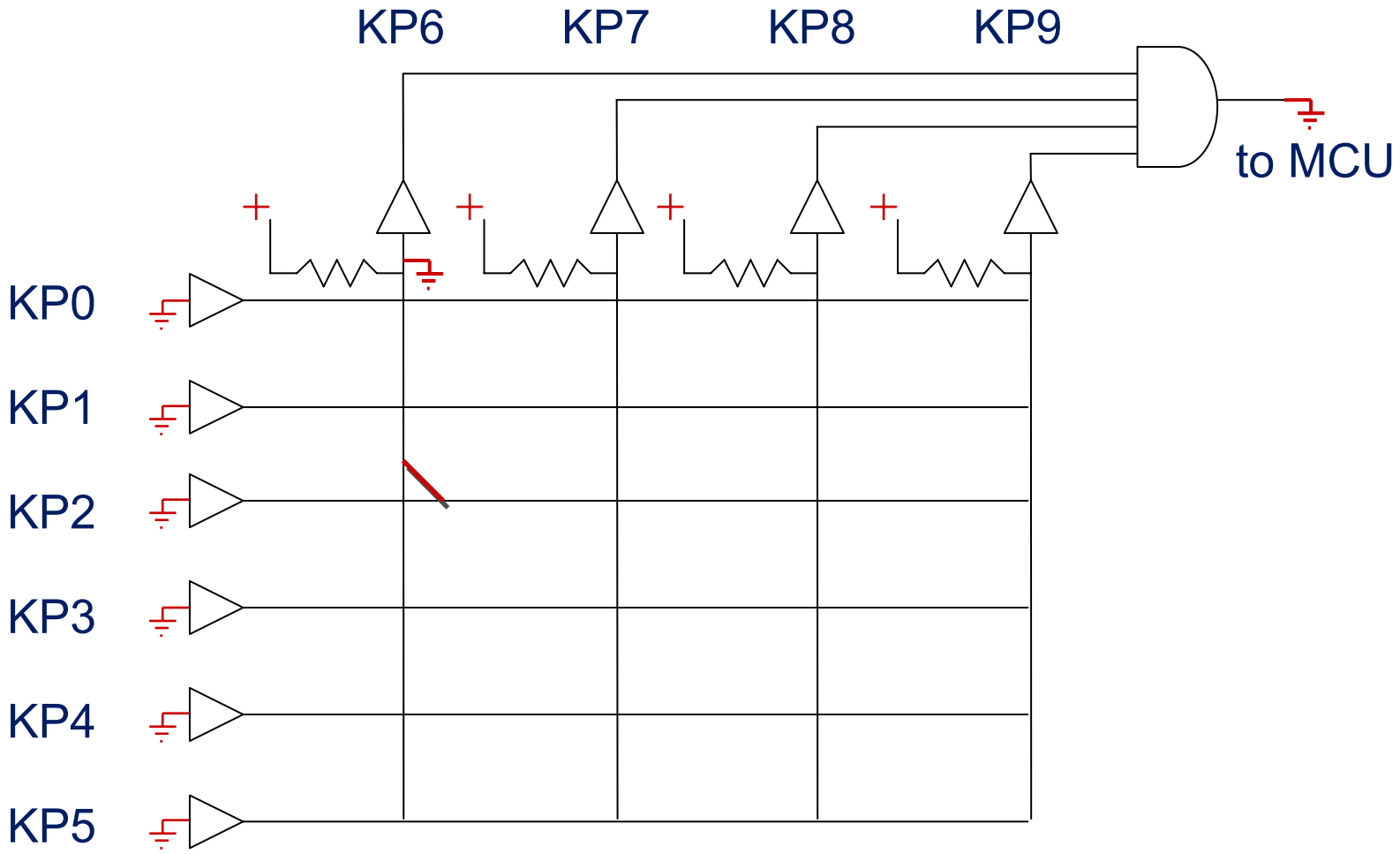
NEONSEVEN

Press/release detection

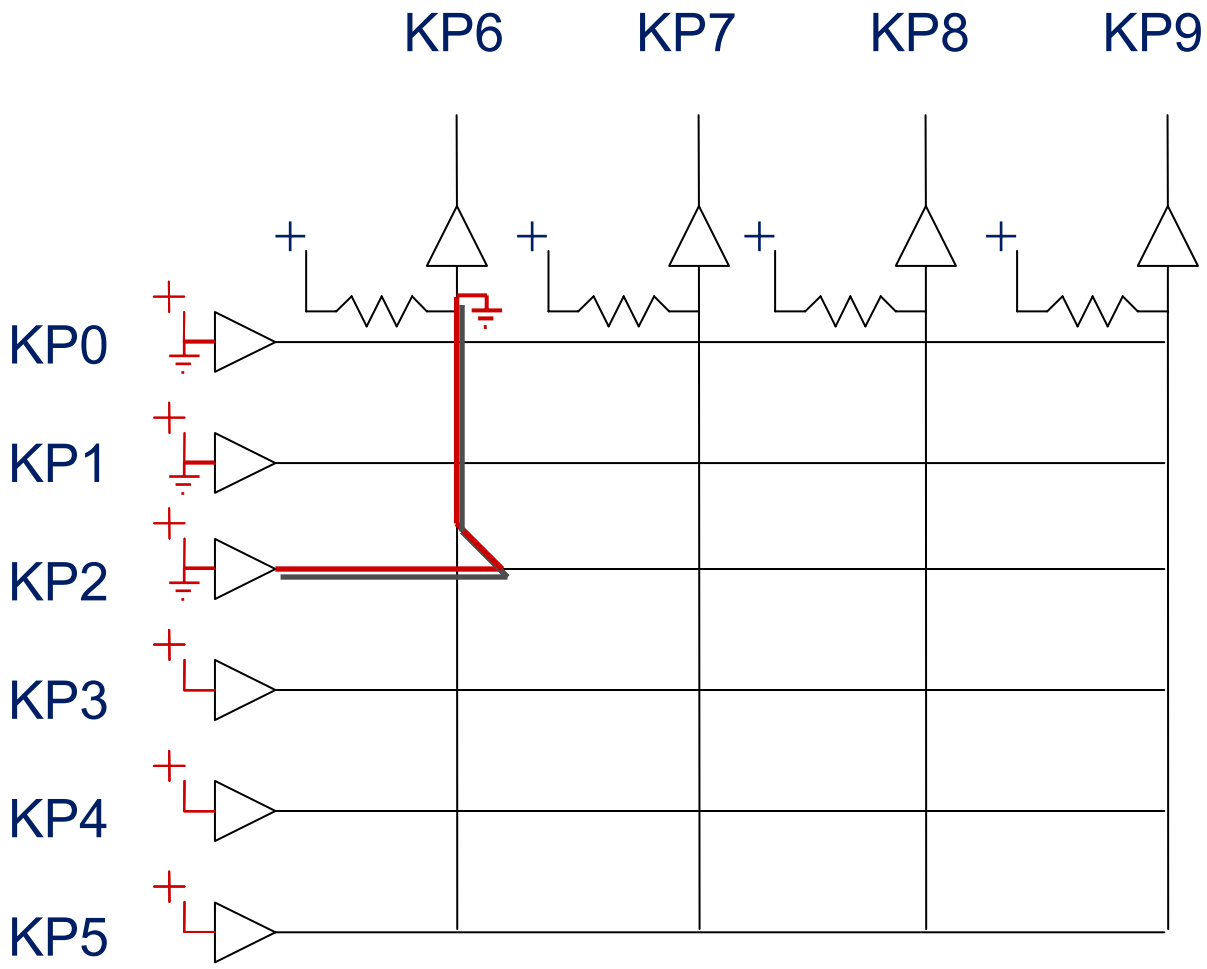
- In order to check the key status, the keyboard driver loads KPDOUT with 0 to start monitoring the keypad
  - If no input port pin (KP9...6) is connected to any output port pin (KP5...0), the input pins are pulled high internally and a high level appears on the keypad interrupt input of the controller.
  - If a key is pressed, both the corresponding keypad row and column are connected to each other and the corresponding KPDINP register bit is set to LOW

NEONSEVEN

Press/release detection



Keyboard scan



Keyboard driver architecture (1)

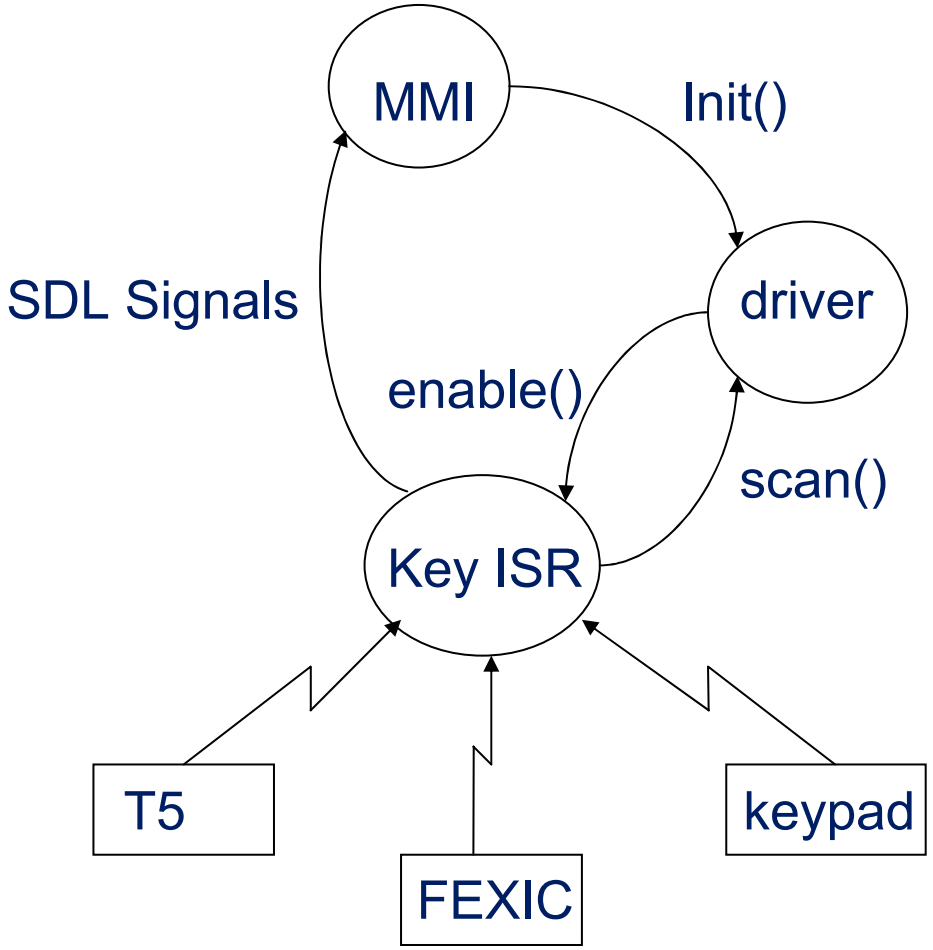
- Main control part: is implemented as a finite state machine, controlling key press/release actions and triggered by 3 LISR, each handling a specific interrupt:

1. KPDIC fires on each key press and release
2. T5INT generates a delay used for debouncing
3. FEX7IC
   - BP30: fires when opening/closing the flip on BP30
   - Goldfinch: fires when ON-KEY is pressed or RTC alarm event occurs

- Communication to the MMI is handled by a HISR (triggered by one of the LISRs) and is performed via SDL Signals.

NEONSEVEN

Keyboard driver architecture (2)



NEONSEVEN

Keyboard matrix

- The keyboard matrix contains 24 elements, identified by an index running from 0 to 23. Each element in the keyboard matrix can be associated to a key, named PBn (n=1,2,…) in the schematics.

	KEYIN0	KEYIN1	KEYIN2	KEYIN3
KEYOUT5	PB3	PB4	KOOK	PB9
KEYOUT4	PB15	PB5	PB2	RIGHT
KEYOUT3	PB11	PB1	PB14	UP
KEYOUT2	PB8	PB6	PB7	OK
KEYOUT1	PB17	PB18	PB13	LEFT
KEYOUT0	PB16	PB10	PB12	DOWN

NEONSEVEN

Key mapping

- The keyboard hardware driver maps the physical keys to their logical interpretation according to a platform dependent array

INDEX	KEY	FUNCTION
0	PB16	*
1	PB17	7
2	PB8	4
3	PB11	1
4	PB15	SND
5	PB3	LSO
6	PB10	0
7	PB18	8
8	PB6	5
9	PB1	2
10	PB5	C
11	PB4	RSO

INDEX	KEY	FUNCTION
12	PB12	#
13	PB13	9
14	PB7	6
15	PB14	3
16	PB2	DEL
17		HOOK
18		DOWN
19		LEFT
20		OK
21		UP
22		RIGHT
23	PB9	FREE

NEONSEVEN

Debouncing

- After each key interrupt (either press or release) a debounce timer is started
- On each timeout the keyboard matrix is scanned again and the result has to be constant within the debounce period in order to validate the key action
- The same debouncing strategy is also applied to flip open/close detection

Interrupt service routines

- *Keypad_HighISR*
  - Handles communication with the MMI via SDL signals
  - Keypad_HISR sends a startup signal after keypad initialization and then it notifies the MMI of each key press and each key release
  - The HISR is necessary in order to release the LISRs from heavy and time-consuming activities such as dynamic memory allocation and signal processing

NEONSEVEN

Interrupt service routines

- *KEY_keypad_lowisr*:

  Handles keyboard interrupts caused by either a key press or a key release;

- *KEY_timer5_lowisr*
  - Handles the debouncing procedure;

Interrupt service routines

- *KEY_flip_lowisr*: (*Only for BP30*)
  - Triggered by the flip interrupt, which fires each time the flip is either opened or closed;
- *KEY_on_off_lowisr. (Only for Goldfinch)*
  - Handles keyboard interrupts caused by ON-KEY press or RTC Alarm event;

NEONSEVEN

Multiple keys detection

- The keyboard driver supports both single and multiple keys press/release detection

  – *Single mode*: each time a key is pressed the driver informs the MMI via an SDL Signal containing the pressed key code; such key press message is always followed by a key release message containing the fixed release code 0x22.

  – *Multi mode*: for each key press the driver notifies the MMI via an SDL Signal with the pressed key code; for each key release the driver sends an SDL Signal with the fixed code 0x22 plus the released key code

NEONSEVEN

Keyboard driver interface

- Driver to MMI: asynchronous (SDL Signals)

1. KBDT_CONTROL: key press/release notification, including key code
   - Parameters:
     - Param1: length (always 1)
     - Param2: key code for a key press, 0x22 for a key release
     - Param3: key code for a key release in multi mode, always 0 otherwise

Keyboard driver interface

1. KBDT_STARTUP_STATUS_IND: driver start up notification, including “ON-KEY” status
   - Parameters:
     - Param1: active flag, true if the “ON-KEY” is pressed on start up
     - Param2: length (always 1)
     - Param3: key code
     - Param4: always 0

NEONSEVEN

Keyboard driver interface

- KEY_init: register and variable initialization
  - Is called on power up when Keypad_HighISR process is created.
  - KEY_init performs the initialization of the keypad registers, interrupt registers and global variables

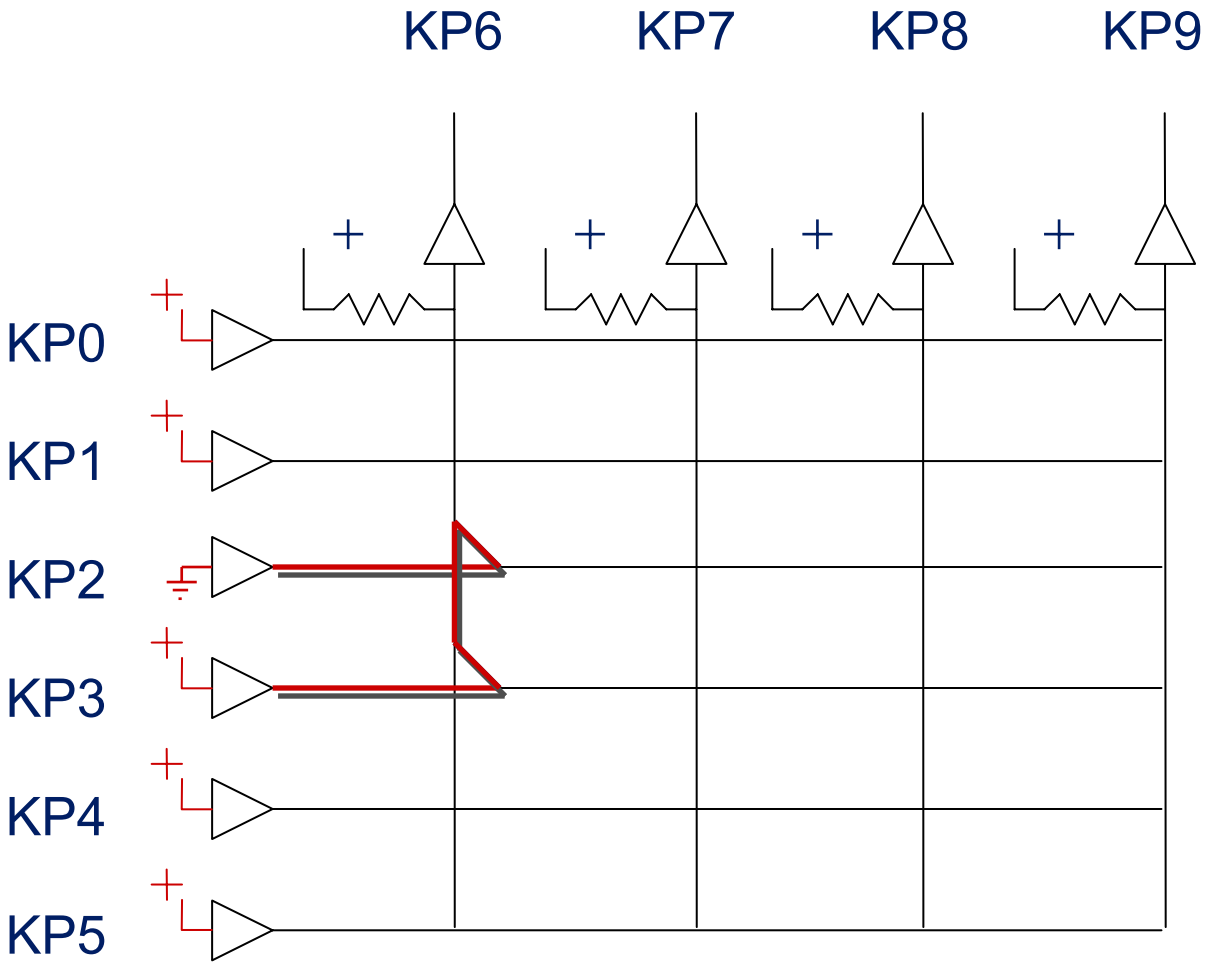
Keyboard driver interface

- KEY_confirm_sdl_mail: confirm the keyboard SDL signal reception (used to avoid MMI queue overflow)
  - Decrements key_mail_confirm_counter, which is used by MMI to confirm the reception of keyboard SDL signal.
  - If key_mail_confirm_counter reaches its maximum value then any further key press and release actions are ignored

Keyboard driver interface

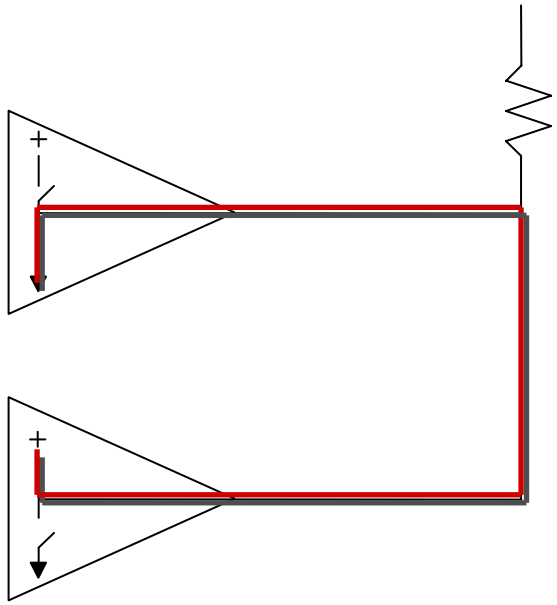
- KEY_get_key_status: returns the status of the specified key
- KEY_test_get_key_matrix: returns the updated key matrix status
- KEY_ext_key_press_simulation: simulation of a key press action
  - Sets a simulation flag, stores the required key value then forces a KPDIC interrupt, which triggers an SDL Signal to the MMI

Short circuit in multiple key press (1)



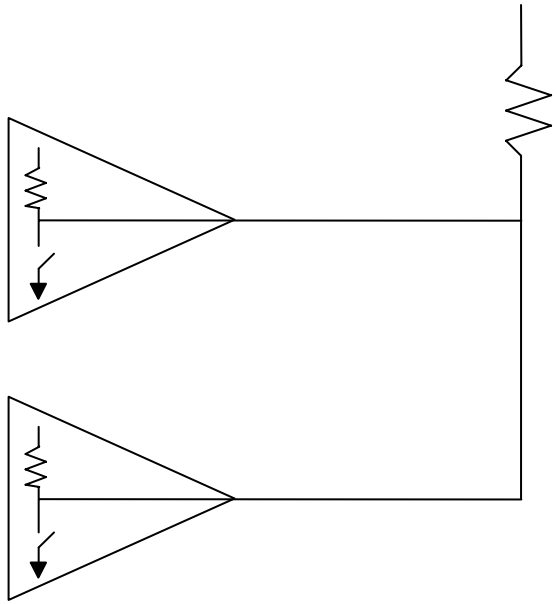
Short circuit in multiple key press (2)

- Push-pull configuration: in the standard scanning procedure if 2 or more keys on the same row are simultaneously pressed then a short circuit occurs when the corresponding KPDOUT bit is reset during the keyboard matrix scan



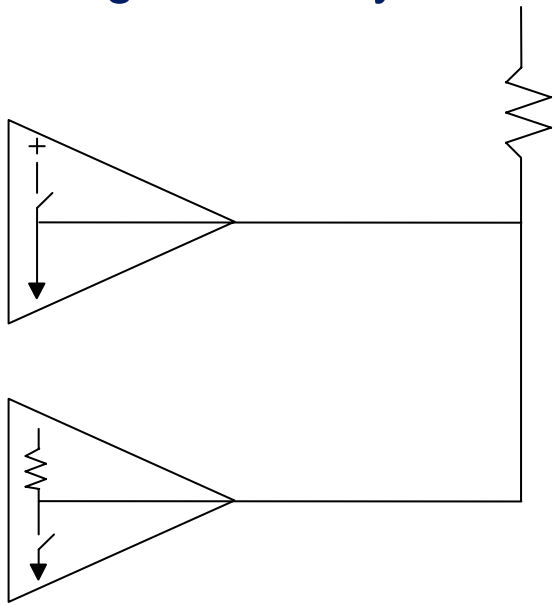
Improved scan algorithm

- In order to avoid this short circuit KP0…KP5 are configured as general purposes GPIOs with pull-up resistors in open-drain configuration



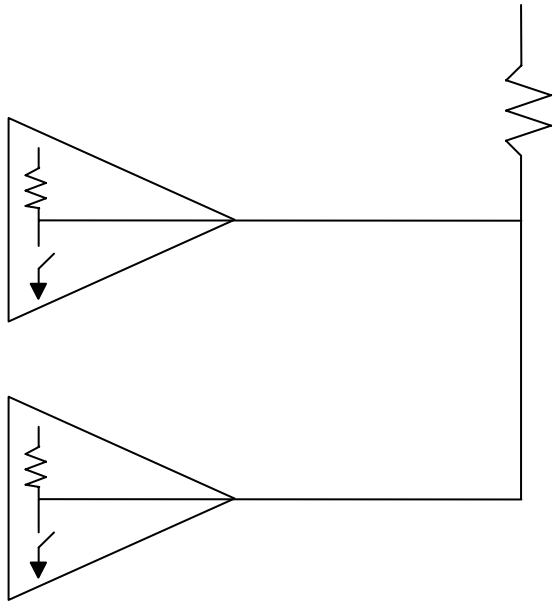
New scanning procedure (1)

1. KPn is configured as GPIO push-pull output pin and is reset; KBDINP is then read bit per bit detecting the status of all the keys in column n. As the other output pins are in open drain with pull-up configuration and are set, no short circuit is possible even though other keys are simultaneously pressed



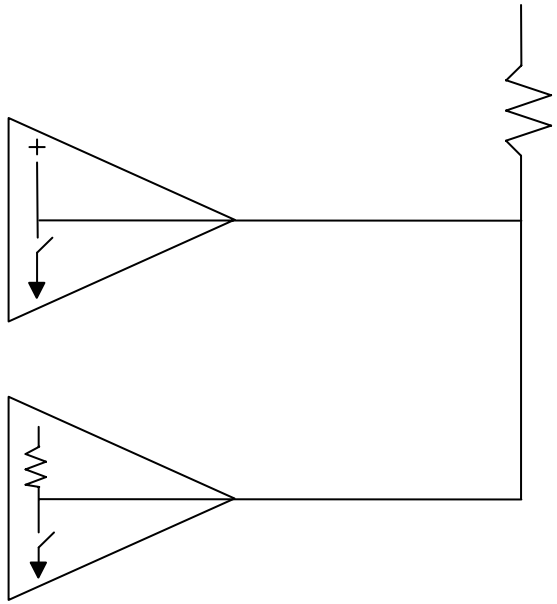
New scanning procedure (2)

2. KPn is then configured as GPIO open drain with pull-up and is set, avoiding any short circuit



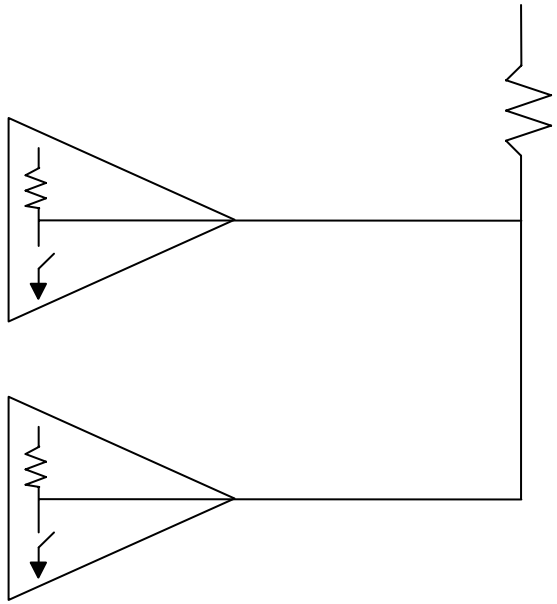
New scanning procedure (3)

3. Now in this safe situation KPn is configured as GPIO push-pull, to speed up the signal transition from 0 to 1 (see figure below)



New scanning procedure (4)

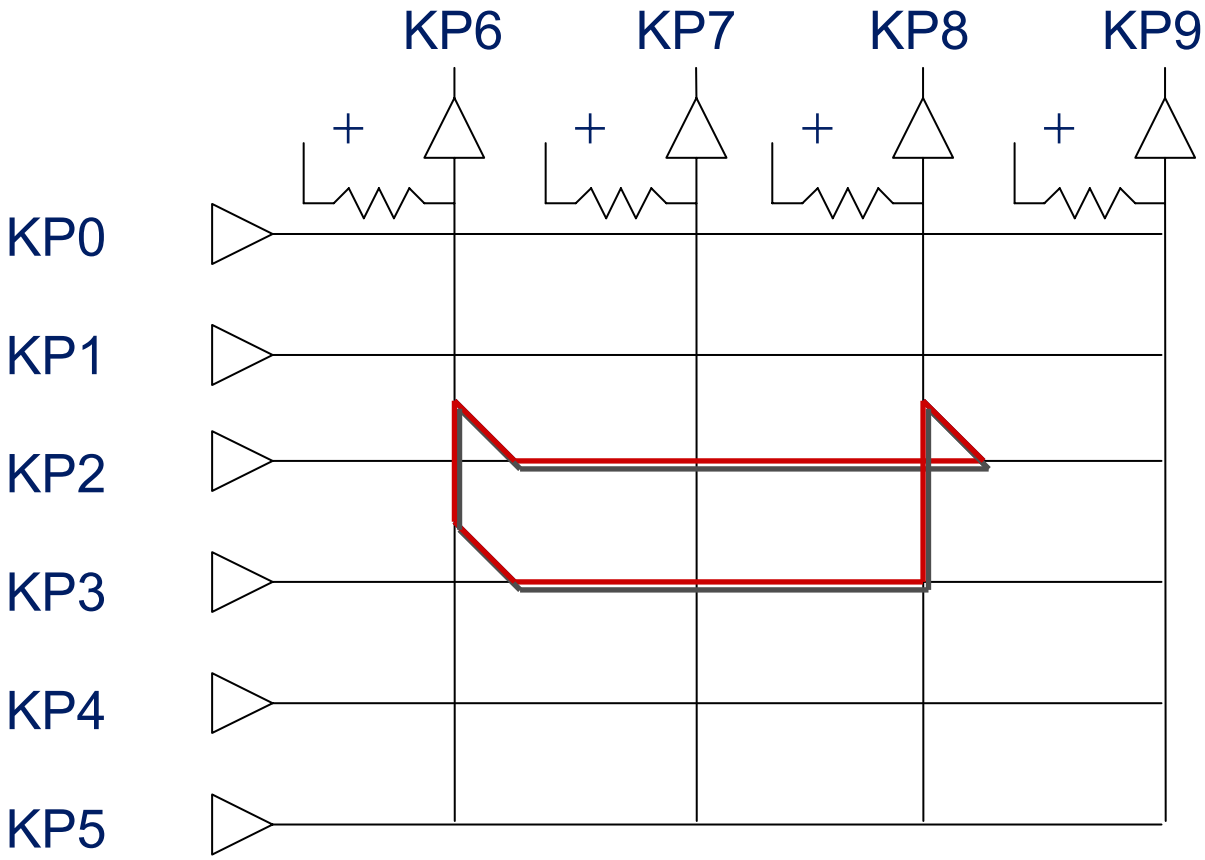
4. Then KPn is configured again as GPIO open drain with pull-up, to be ready for scanning the next column (see previous page).



NEONSEVEN

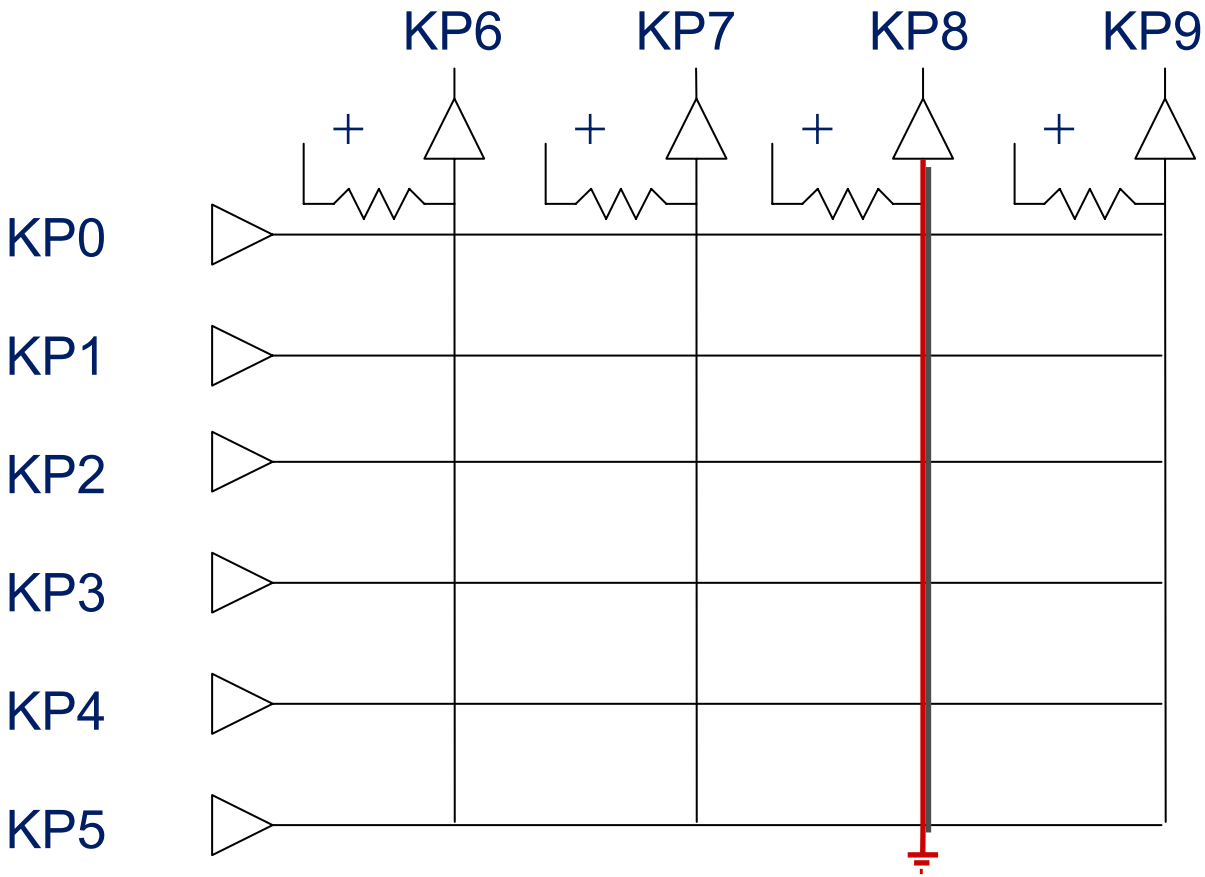
Multiple key limitations

- If 3 key on 3 corners of a rectangle are pressed then also the key on the fourth corner is detected as pressed



Hook key detection

- The hook key press/release detection needs a special handling because it pulls down the KEYIN2 line



Flip detection (Only for BP30)

- The flip is connected to an external interrupt, which triggers the debouncing algorithm followed by an indication to the MMI via KBDT_CONTROL SDL Signal.

1. On Globe6: FEX7IC only rising edge is detected, therefore flip open is handled by LISR and flip close by polling

Long Keypress detections

- Long keypress detection is handled by APOXI, it is not handled by Key driver.
- When a key is pressed for long time more interrupt are generated, but no indication is sent to MMI because the keyboard matrix is unchanged. (Key driver detects the same key)
- As soon as the Key is released a new interrupt is generated and the Key_release message is sent to MMI.

NEONSEVEN

Keyboard driver configuration

- Multiple keys detection: supported if KEY_MULTI is defined and enabled if key_multikey_mode_flag is set.
- MMI queue protection: supported if KEY_MAIL_CONFIRM is defined, sets a limit to the number of SDL Signals that the keyboard driver sends to the MMI without receiving any confirmation.
- Early initialization: anticipates the keyboard driver start up, supported if KEY_EARLY_INIT is defined.
- New scan algorithm: supported if KEY_NEW_SCAN is defined, avoids short circuits in multi key mode.
- Key flip: supported if KEY_FLIP is defined, notifies the MMI each time the flip is either opened or closed.