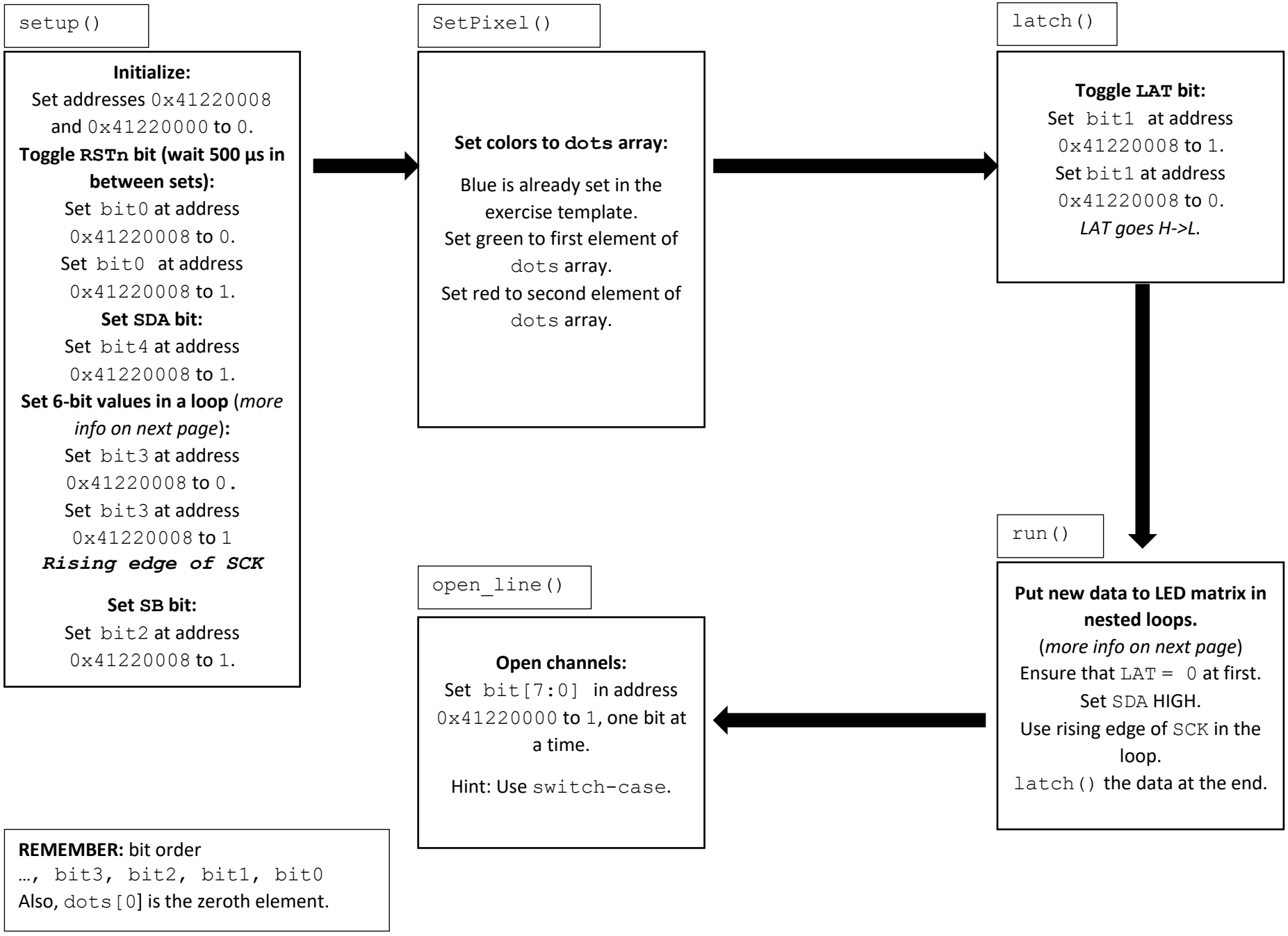
Tampere University

LED matrix block diagram

COMP.CE.100 Introduction to Embedded Systems

LED MATRIX BLOCK DIAGRAM



`setup()`:

- This function will make the necessary initializations to setup the LED matrix for further use.
- Address `0x41220008` is for control signals and address `0x41220000` is for channels. We need to first initialize them by setting them to zero.
- After initializing the addresses, the LED matrix should be reset. This is done by toggling the `RSTn` bit from LOW to HIGH because it is active when low. We can operate with the matrix after the screen has been reset and it is not ON anymore.
- `SDA` bit should be set to 1 to be able to send serial data.
- Next 6-bit values should be set in the register of the DM163 chip. All bits in the register should be set to `1` because the values in the 6-bit register will be multiplied with an 8-bit register. 24 (3*8, for R, G and B) "bytes" should be transferred, so in total 24*6 bits should be sent in a loop structure.
- When sending the data, use the rising clock edge.
- Finally, the `SB` bit should be set to 1 to transfer the data to the serial bank.
- For setting the bits, see for example *COMP.CE.100_exercise_guide.pdf, pages 16, 19-20*.

`SetPixel()`:

- In this function the color and location of the pixels in LED matrix is set in the `dots` array.
- The blue color is already set in the example. Green and red colors should be set (in this order) to the matrix as well.

`latch()`:

- We need to toggle the latch bit from HIGH to LOW to store the pixel data values to the register bank.
- Latch bit is `bit1` of control signals.

`run()`:

- Here we need to write the pixel data to the LED matrix driver.
- 3 nested loops are needed:
- Before loops, latch should be 0 (see `latch()`)
- ----the outermost loop should loop through pixels.
- --------The next loop should loop for colors (in order B, G, R) and
- ----------the innermost loop should loop through the actual color data.
- In the innermost loop `SDA` should be set to 1 if there is still data to be sent. `SDA` bit is the `bit4` of the control signal address.
- Else, `SDA` should be set to 0.
- Finally, `SCK` should be toggled from 0 to 1 and the dots array should be shifted left by 1. `SCK` is the `bit3` of the control signal address.

`open_line()`:

- We need to set the corresponding channel bit to 1 at the channel address to open the columns and show the pixel data on the matrix.
- To do this, a switch-case statement can be used. The cases should be channel numbers from 0 to 7 and the channel bits should be set to 1 accordingly: channel number is the bit position, so channel 0 is `bit0`, etc.
- The default case should close all channels, so channels should be set to 0.

INTERRUPT HANDLERS

BUTTON/SWITCH INTERRUPT

`ButtonHandler()`

Check `Status` variable for button presses and switch toggles:

Buttons are `bit[3:0]`.

Switches are `bit[5:4]`.

Hint: Use `if`-statement.

REMEMBER: bit order
`…, bit3, bit2, bit1, bit0`

TIMER INTERRUPTS

`TickHandler()`

Refresh the LED matrix screen:

The channels should be turned ON individually.

Check if channels are inside the screen.
Close all channels.
Call `run()` and `open_line()` on channel.
Increment channel.

`TickHandler1()`

Handle alien and shooting:

Call functions for alien movement and shooting here, according to your game logic.

Interrupt basics:

- Do not have to wait for button presses etc. in a loop (this is called *polling*).
- Instead, the normal program execution will be stopped and the execution time is given to interrupt handlers.
- After the interrupt, the normal program execution will continue from the point where it was before calling the interrupt.
- Some info is available in the *exercise guide COMP.CE.100_exercise_guide.pdf, page 11* and in the *Project work extra material PDF*.
- More information will be given in the course `EE.ELE.250 Microcontrollers (5 cr)`.

`TickHandler():`

- This function is the timer interrupt handler for the LED matrix screen. The refresh rate is 800 Hz.
- This makes it look like there are many LEDs on at the same time, although they are ON individually and just updated very fast.
- To update the matrix, it should be first checked that the channel is not greater than 7 (over the screen). If it is, zero it.
- Next, all channels should be closed with the default case of `open_line()`.
- After this, the current channel should be run and opened, and the channel should be incremented.

`TickHandler1():`

- This function is the timer interrupt for the gameplay, i.e. moving the alien, shooting with the gun etc. The refresh rate is 10 Hz by default.
- The refresh rate can be changed by calling `change_freq()` function inside `ButtonHandler()`. It can be assigned to `SW1`, for example.
- It would be good to implement the gameplay functions in `Pixel.c` or your own custom file and call them in this timer interrupt function.

`ButtonHandler():`

- This function is the button/switch interrupt for the gameplay controls, i.e. moving the alien, shooting with the gun etc. The frequency is 10 Hz by default.
- Use `Status` parameter to check which button caused the interrupt.
- `BTN0` is bit0, `BTN1` is bit1, `BTN2` is bit2, `BTN3` is bit3, `SW0` is bit4, `SW1` is bit5
- Bank information is not needed.
- Call ship movement functions accordingly.
- The frequency can be changed with `change_freq()` function. It can be assigned to `SW1`, for example.