Tampere University

Alien shooter exercise on an LED matrix

Otto Eerola

COMP.CE.100 Introduction to Embedded Systems

Exercise main goals

- Learn the basics of embedded C-programming and how the LED matrix works
- Learn how to read information from datasheets or from other documentation
  - How to utilize this information in code
- Learn how to use the Xilinx SDK and its features (debugger, software loading to board etc..)

Tampere University

General info

- Updated exercise files will be added on Moodle.
- Check MapV2.pdf to find TC219 (if you don’t know where it is)
  - You will get access to rights to TC219 (need to read NDA and sign it somehow)
- Can also do remotely at home
  - Loan board and install Xilinx SDK
- Find yourself an **exercise pair/pairs** (or do solo)
  - Three is max group size
  - Use Moodle group tool to form groups!
- Deadline is on 3.12.2022!
  - Source codes must be returned by end of that day
    - Show to assistant or return project to Moodle

Tampere University

Guidance times

- Guidance times available sometime after exam week and last until deadline
  - Local guidance at TC219 and remotely at Zoom and Mattermost
    - Invite link to Mattermost: https://mattermost.tut.fi/signup_user_complete/?id=yf3x7e9xo3f4jy5um9pyr9a35y
    - Student channel: https://mattermost.tut.fi/studentsetstaff/channels/compce100-introduction-to-embedded-systems-students
- If there is any change on the times, we will give out a notification.

Tampere University

Borrowing PYNQ

- You need to borrow PYNQ with LED matrix to complete this project work, if doing remotely
  - You need your own Micro-USB cable
  - Borrowing times available on Moodle

Tampere University

Installing the software

- Download Xilinx SDK 2019.1
  - Quick install guide and link will become available on Moodle
  - ***If installer asks to install drivers, just do it***
- Windows or Linux is required with ~40 GB of free space
  - Size is large but SW can be used in other courses, like Digital Design

Tampere University

Exercise pass requirements, 5 p

- LED matrix is working
  - LED matrix LEDs can be set active any amount and anywhere with any color.

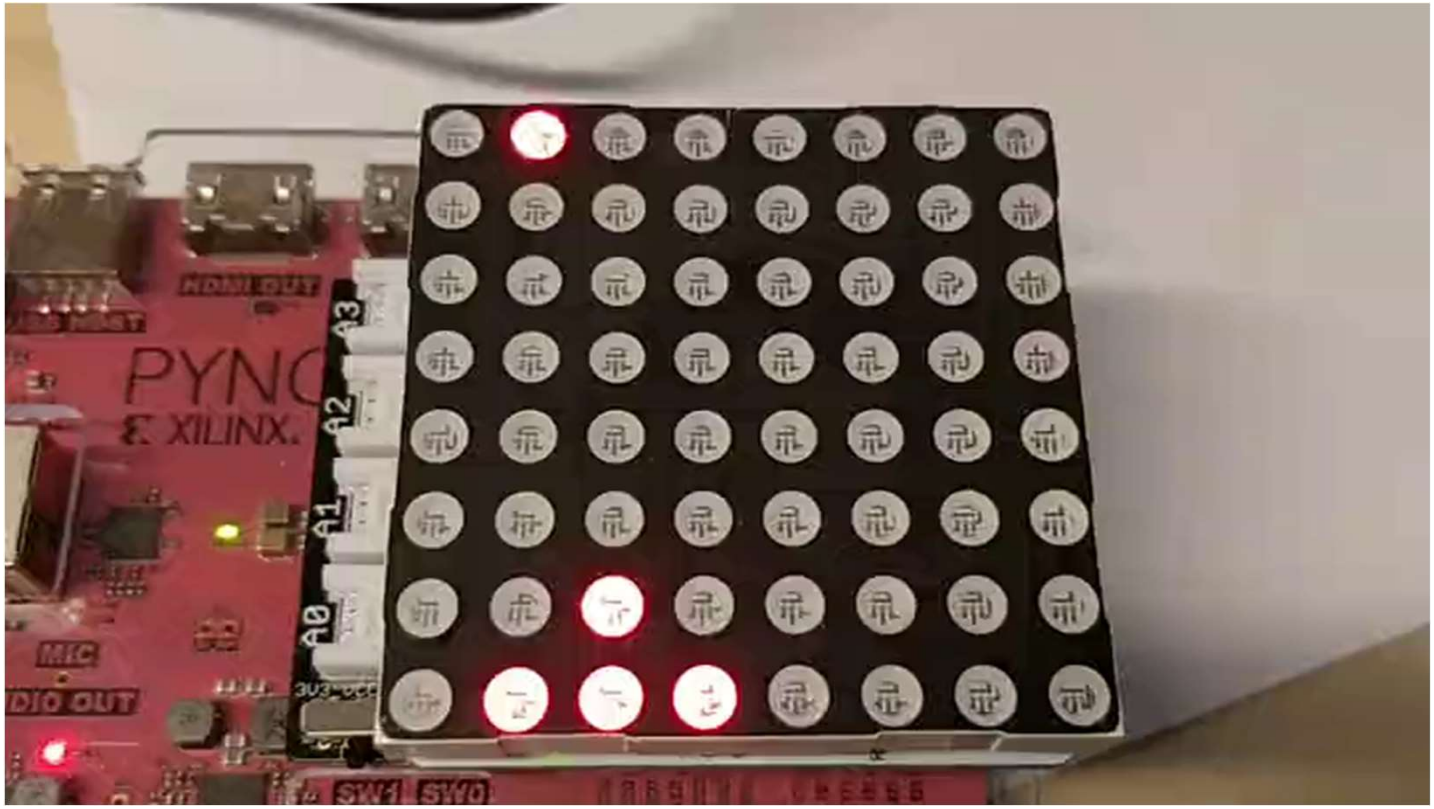
Tampere University

Exercise pass requirements, 10 p

- Code LED matrix to functional state and use that code to code the game
- ”ship” can be moved using buttons
- Alien is moving
- Ship can fire moving bullets
- Hitting alien increases score or/and missing increases miss score
- Game has ending
- Score needs to been shown somewhere
- Game can be restarted
- Coding style is free of choice

Tampere University

Example game screen



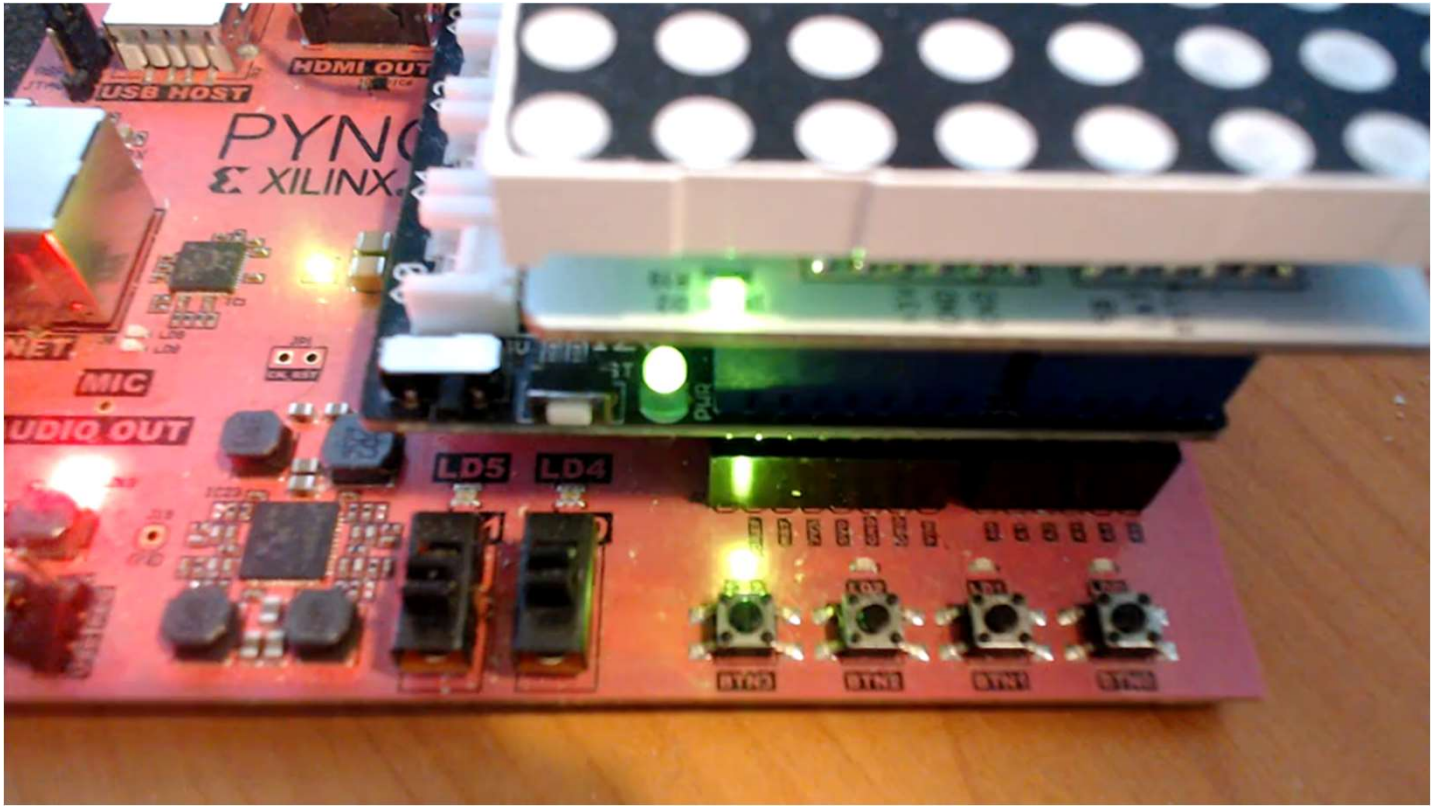
Tampere University

”extra”, 2 p

- Make button-LEDs (normal LEDs near buttons: LD0-LD3) go left->right->left unlimited times
  - LED shifting from left to right and back must be visible with human eyes!
- This needs to be written with assembler
  - Write code into given ASM file (blinker.S)
- This 2 points can be summed to both 10 and 5 points
  - Meaning total points would be 12 or 7

Tampere University

Button-LED blinker example



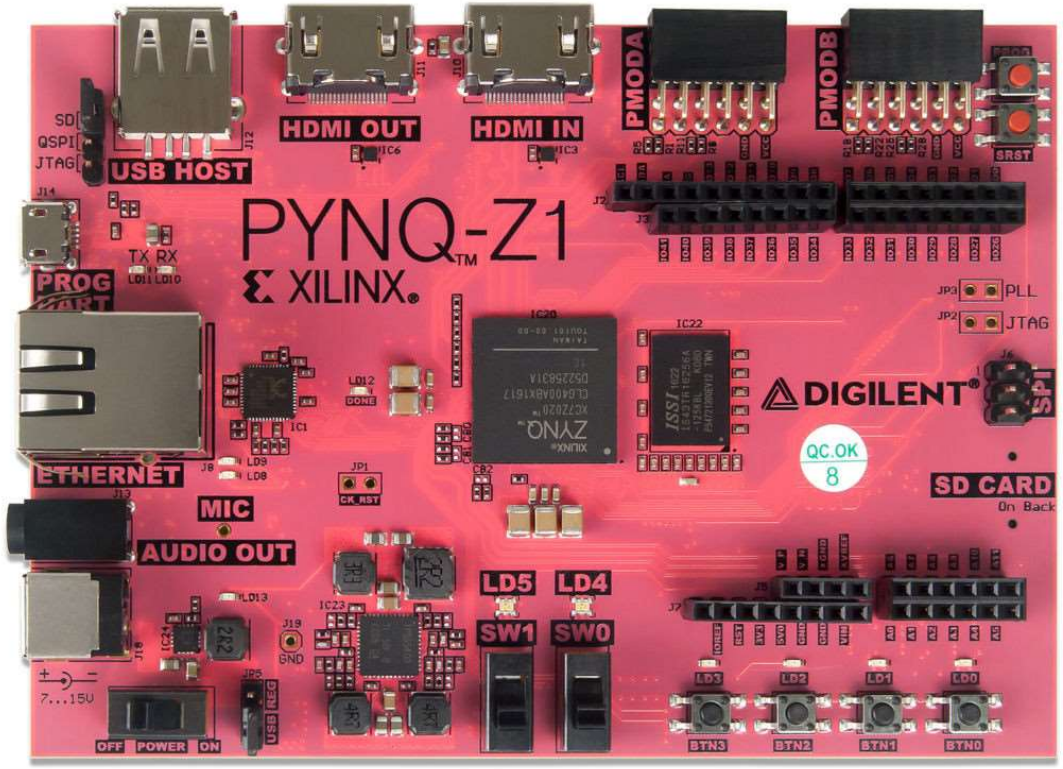
Tampere University

Points summary

- 5 points to get LED matrix working
- 5 points to get game working
- 2 points to get button-LED shifter working using assembler

Tampere University

PYNQ-Z1 Development board

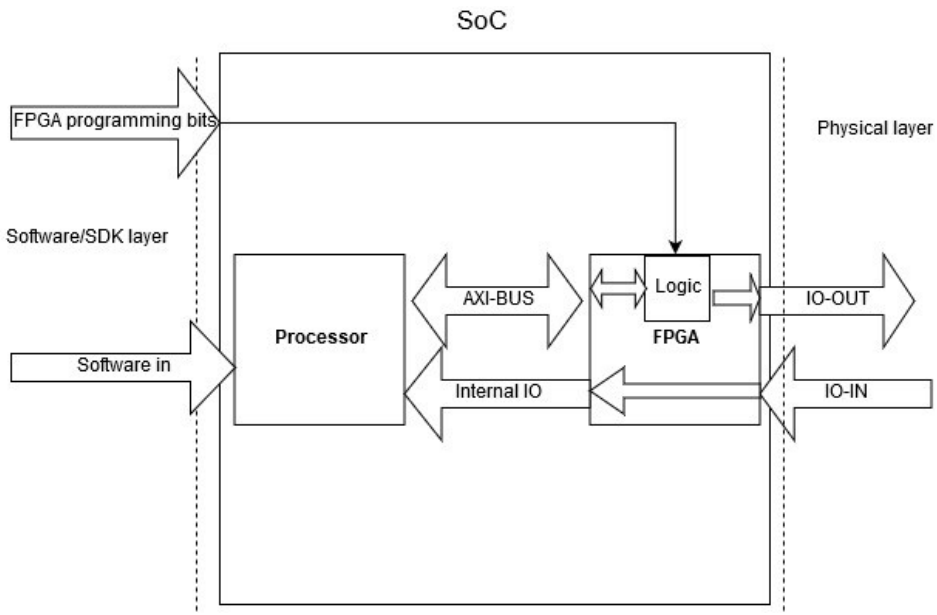


Features

- Zynq-7000 SoC chip
  - ARM Cortex-A9 Processor (Dual core)
  - Artix-7 family FPGA
- Arduino Compatibility: can use Arduino shields (like LED matrix in this exercise)
  - Does not support Arduino codes
- Supports OS and **baremetal programming**.

Tampere University

Simplified block diagram for PYNQ



- FPGA is only used to make interconnection between physical I/O's and processor
- In this course, we only program the processor.

Tampere University

Channel

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
C7	C6	C5	C4	C3	C2	C1	C0

Memory Address=0x41220000

Control_signals

Bit4	Bit3	Bit2	Bit1	Bit0
SDA	SCK	SB	LAT	RSTn

Memory Address=0x41220008

Inputs

Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SW1	SW0	BTN3	BTN2	BTN1	BTN0

Memory Address=0xE000A068

Leds

Bit3	Bit2	Bit1	Bit0
LD3	LD2	LD1	LD0

Memory Address=0x41200000

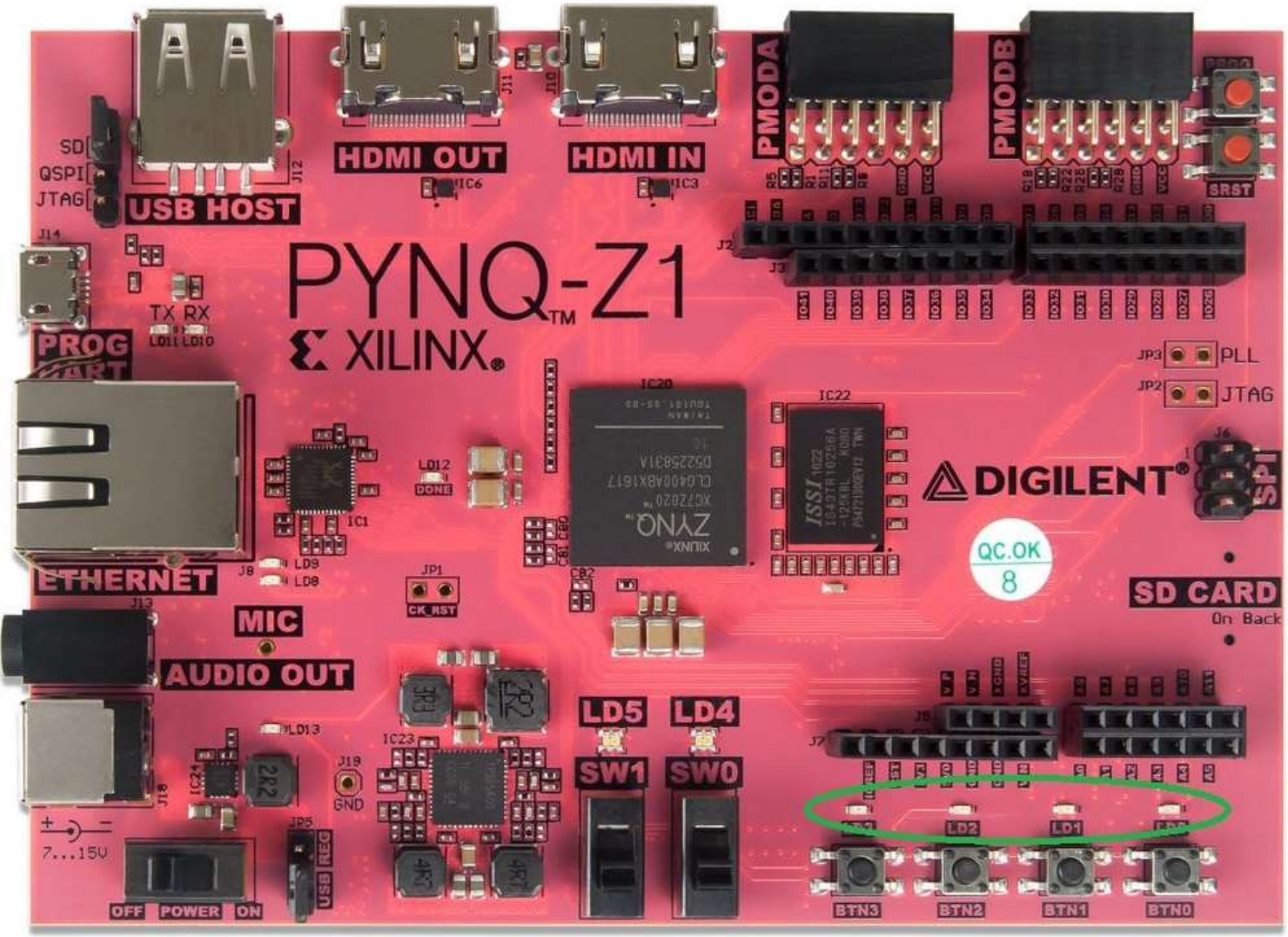
RGB leds

Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LD5.R	LD5.G	LD5.B	LD4.R	LD4.G	LD4.B

Memory Address=0x41240000

NOTE that reset is active when low (n suffix)!

Tampere University

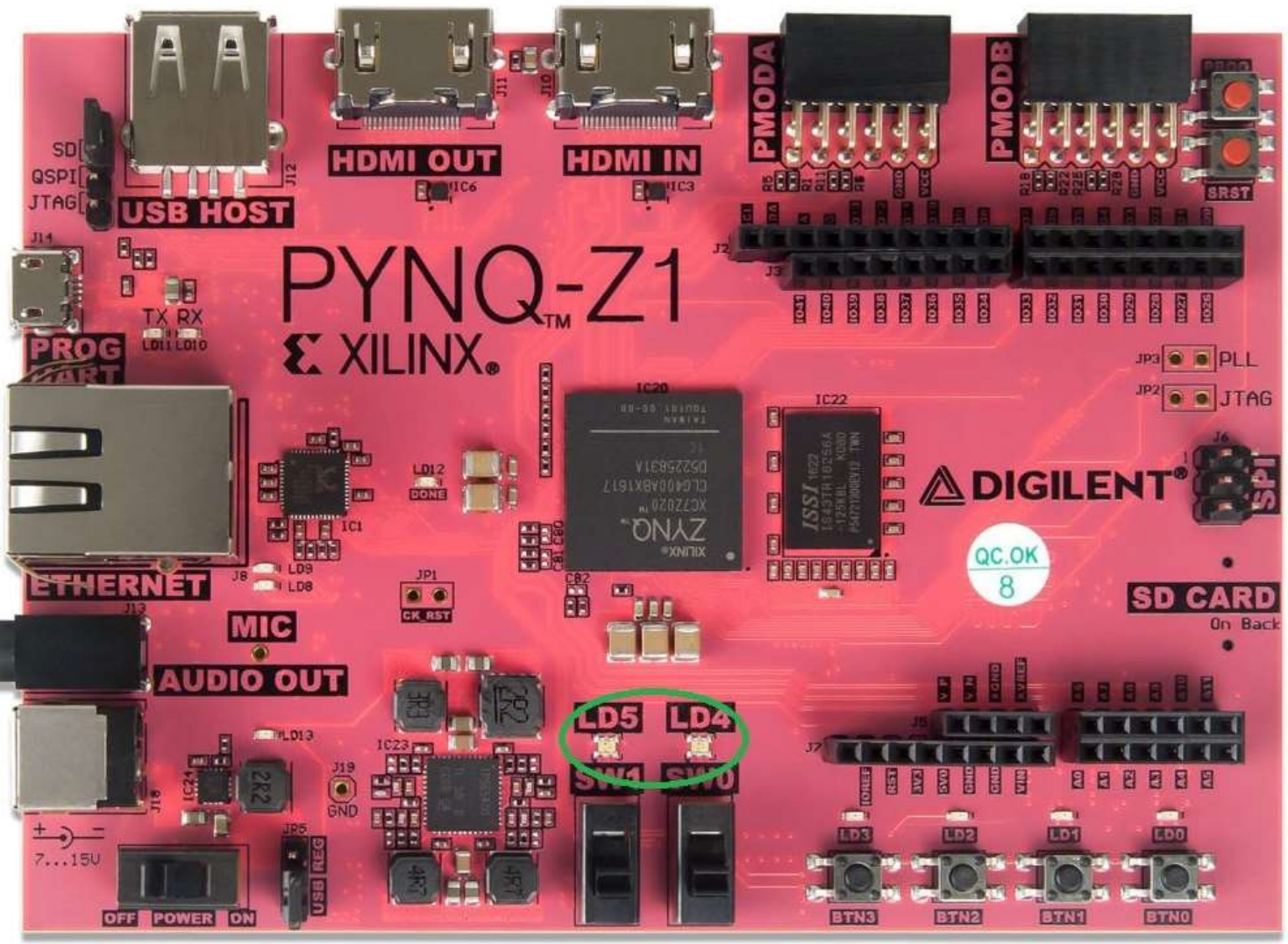


Leds

Memory

address:0x41200000

Tampere University

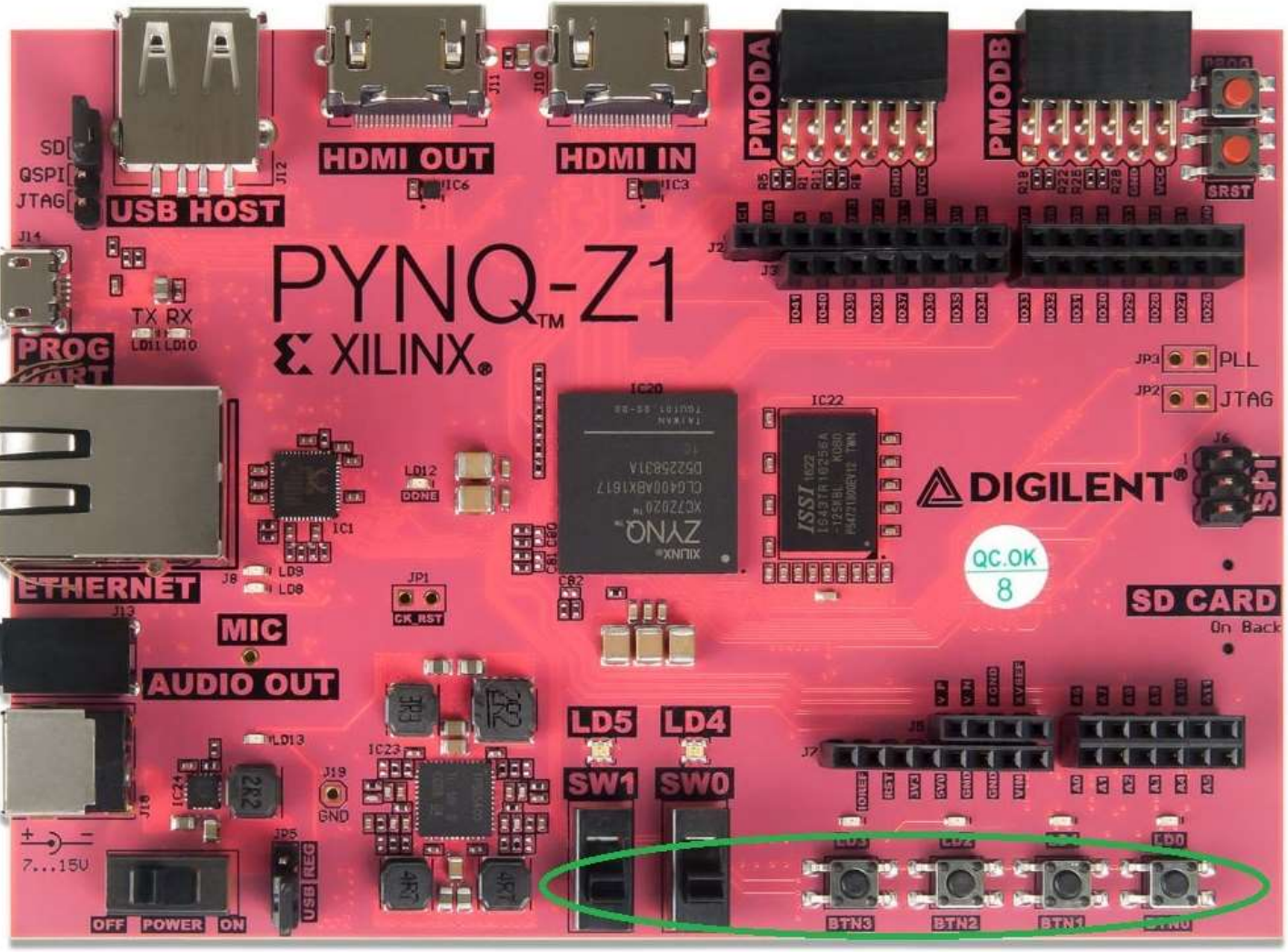


RGB leds

Memory

Address:0x41240000

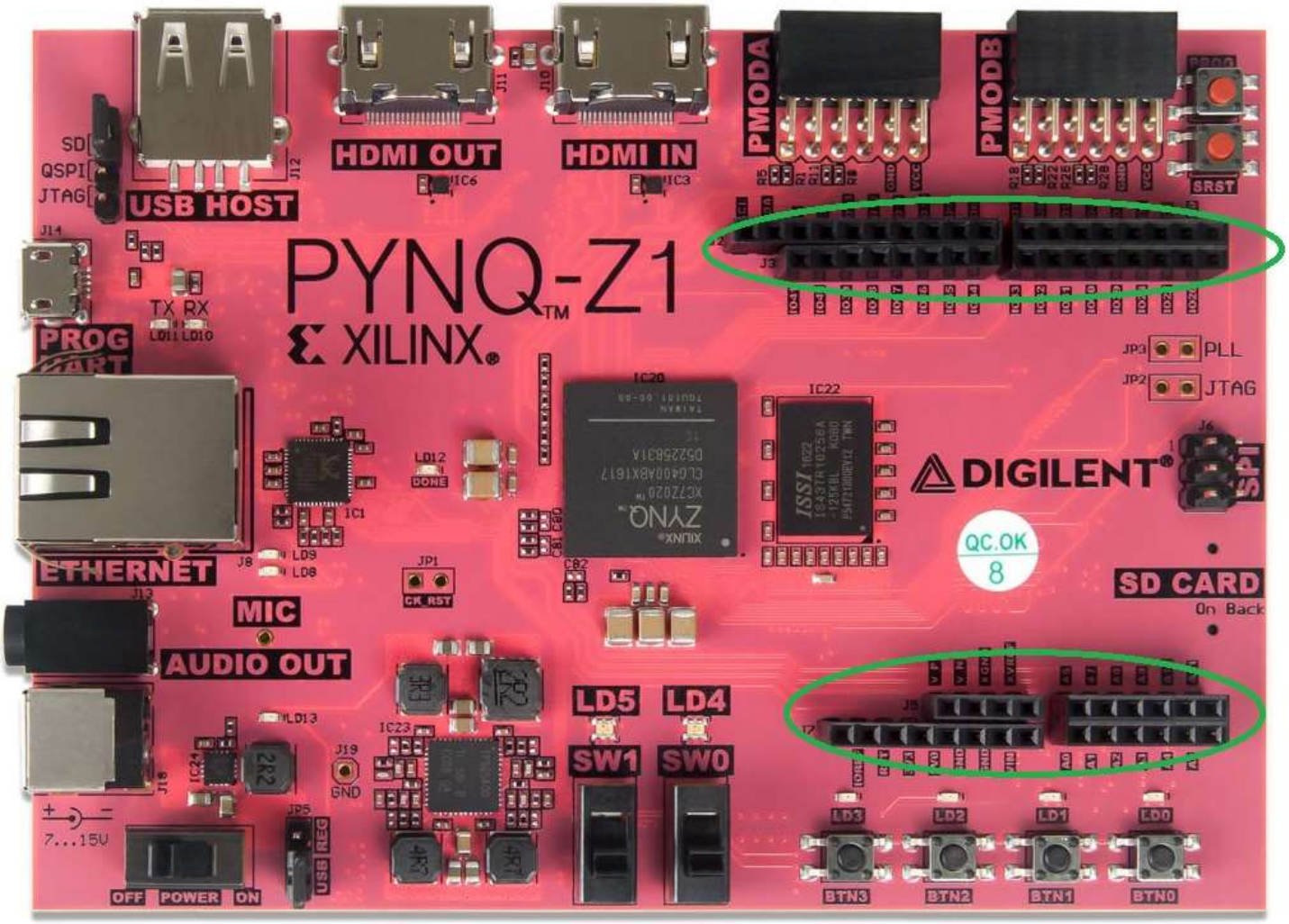
Tampere University



Inputs

Memory

address:0xE000A068



Channel
Memory
address:0x41220000

Control_signals
Memory
address:0x41220008

Tampere University

How to use memory addresses in C?

Example: setting LED **LD0** on

uint8_t **led = 0x41200000;*

**led |= 0x01;*

Example: read input

uint8_t **btns = 0xE000A068;*

uint8_t *a = 0;*

*a = *btns;*

Tampere University

How to use memory addresses in C?

- Led on:

**(uint8_t *) 0x41200000 |= 0x01;*

- Read button state to variable:

uint8_t btn = ***(uint8_t *)** 0xE000A068;

How to use memory addresses in C?

- One good way could be also to define address with some name
- Example, define in global:

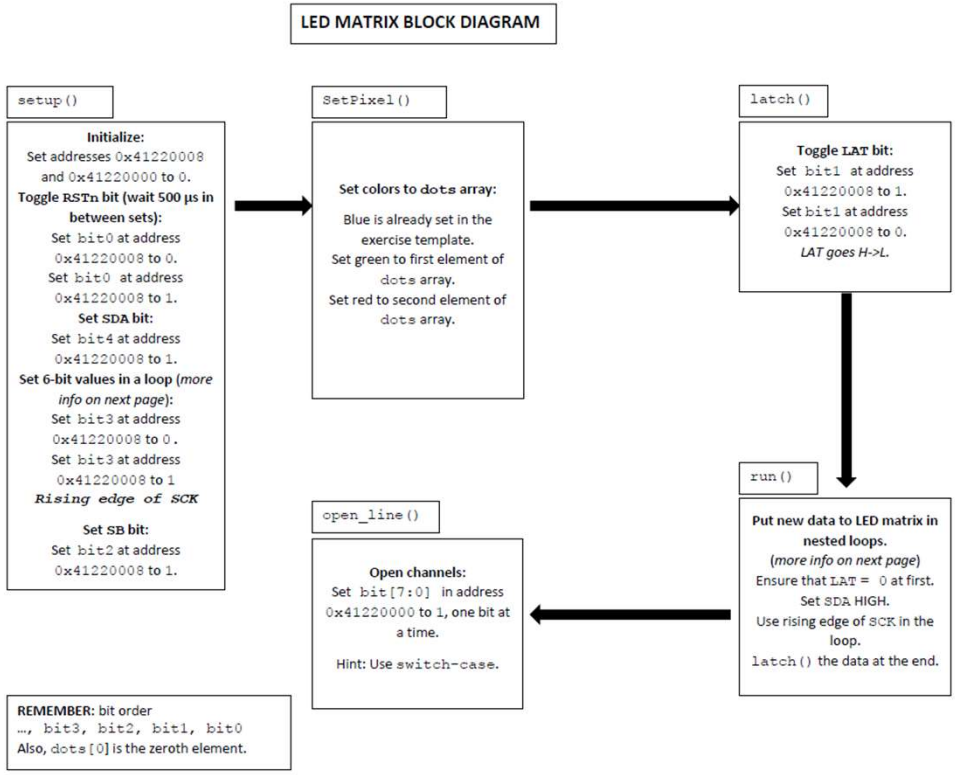
#define led ***(uint8_t *)** 0x41200000

- Now you can use led to modify data at that address, following will turn LED **LD3** on:

led |= 0x08;

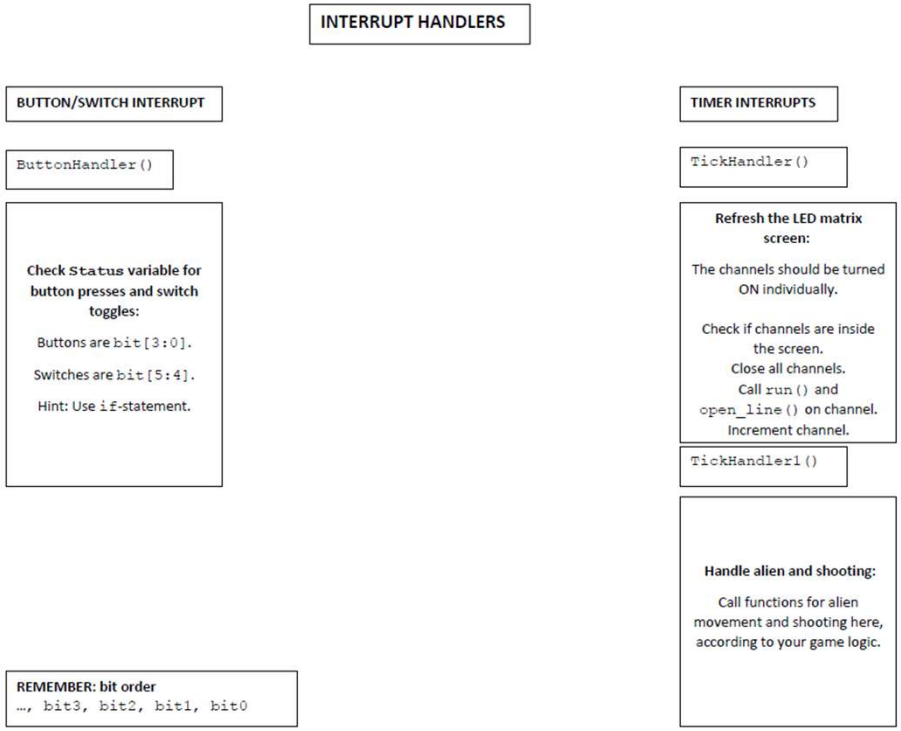
Tampere University

Block diagram of LED matrix functions



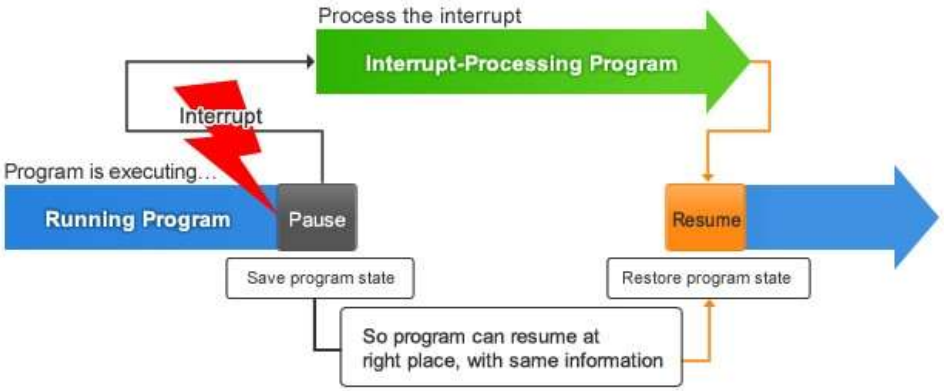
Tampere University

Interrupt handlers



Tampere University

Interrupts



Source: *https://www.renesas.com/eu/en/support/technical-resources/engineer-school/mcu-programming-peripherals-04-interrupts.html*

- Interrupts cause an interrupt in the main program and jump to run the interrupt subroutine.
- In exercise, there are two interrupt methods
  - Timer and GPIO edge interrupt

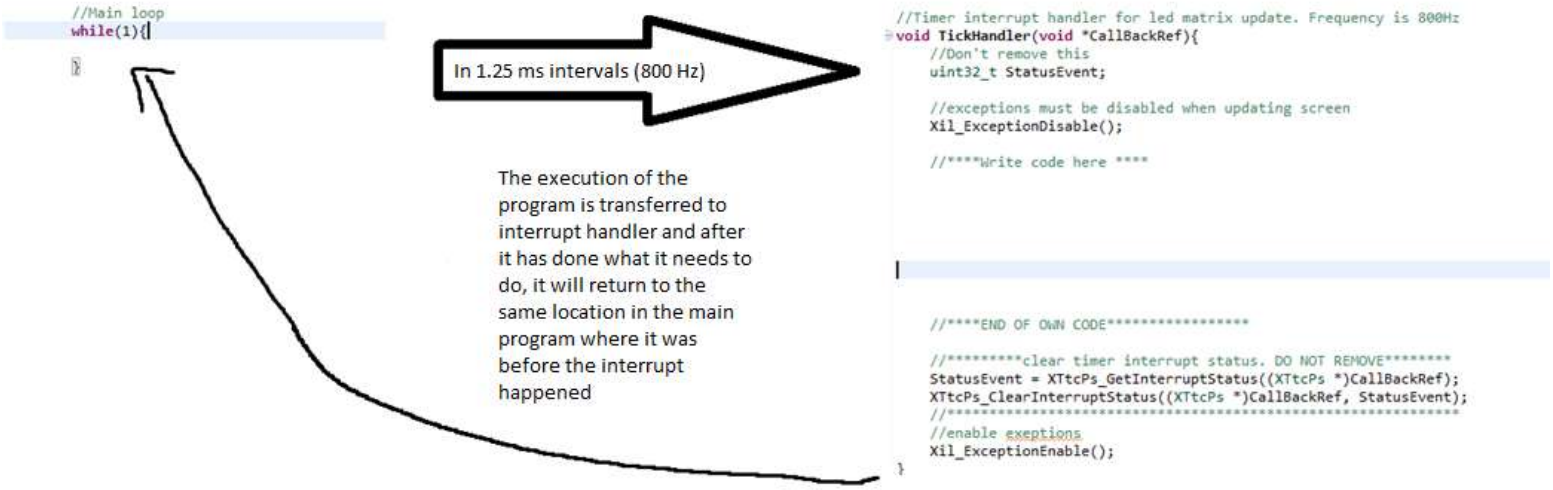
Tampere University

Timer interrupts

- **Very simply:** Timer interrupts are interrupts that are happening within certain intervals
  - In this exercise, timer interrupts are happening with 10 Hz and 800 Hz frequency.

Tampere University

Timer interrupts



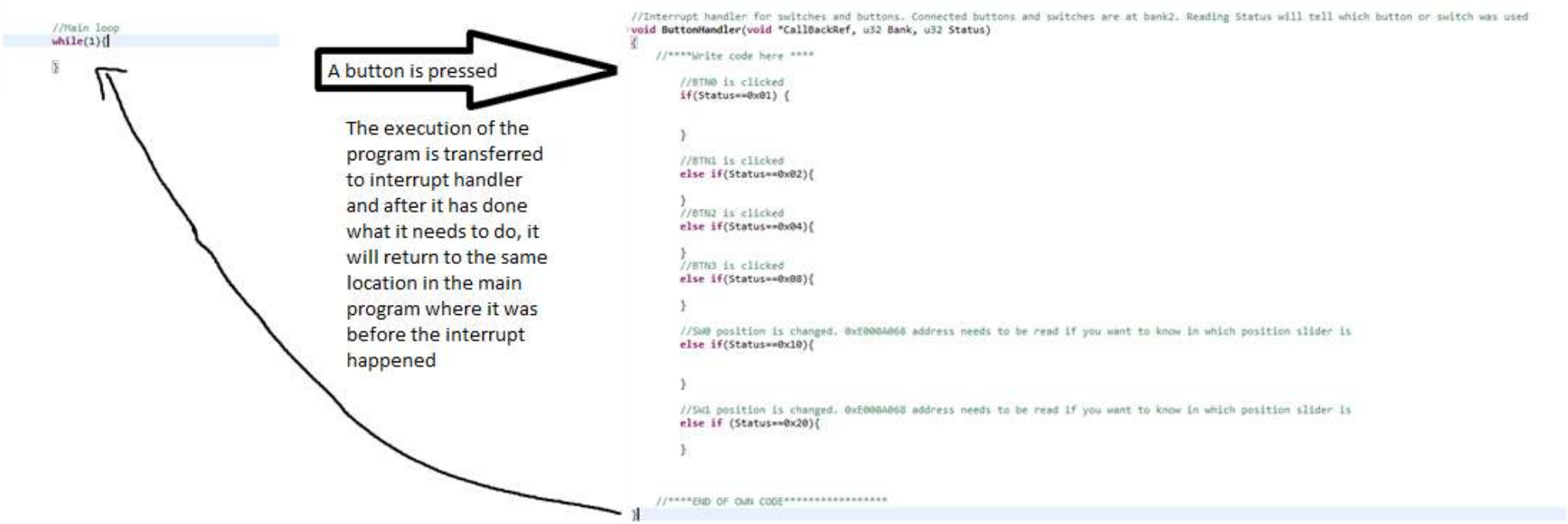
Tampere University

GPIO Edge interrupt

- Interrupt happens on voltage level change of I/O. Usually on low to high or high to low change -> on rising or falling edge.
  - In this exercise, buttons cause an interrupt on falling edge and switches on both edges

Tampere University

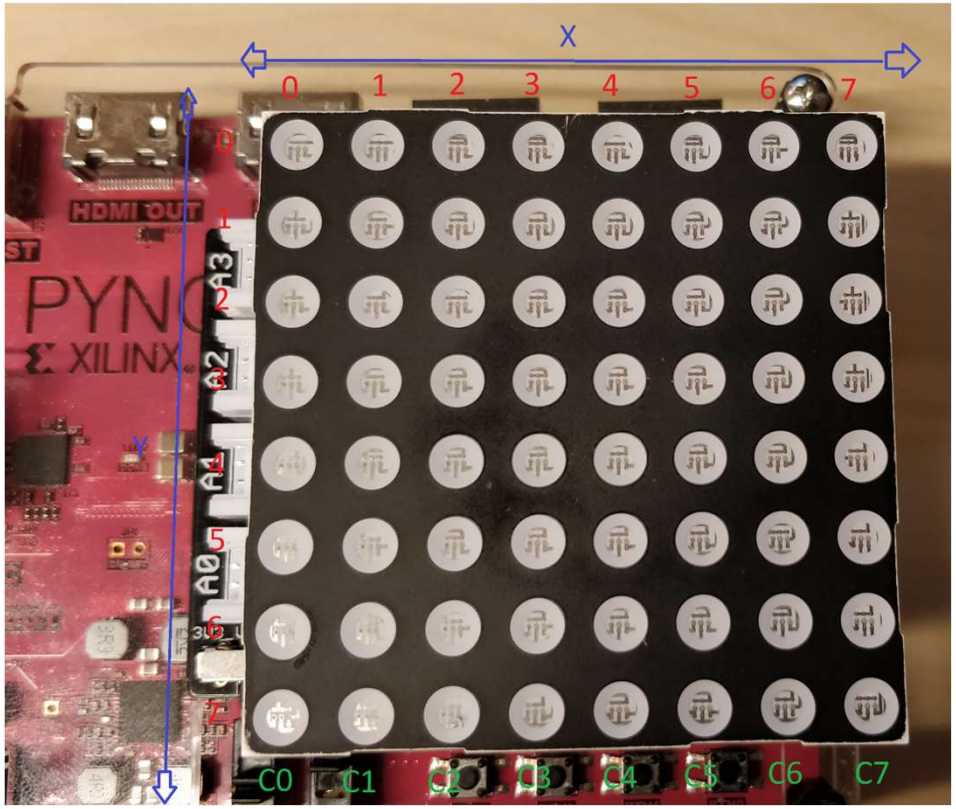
GPIO Edge interrupt

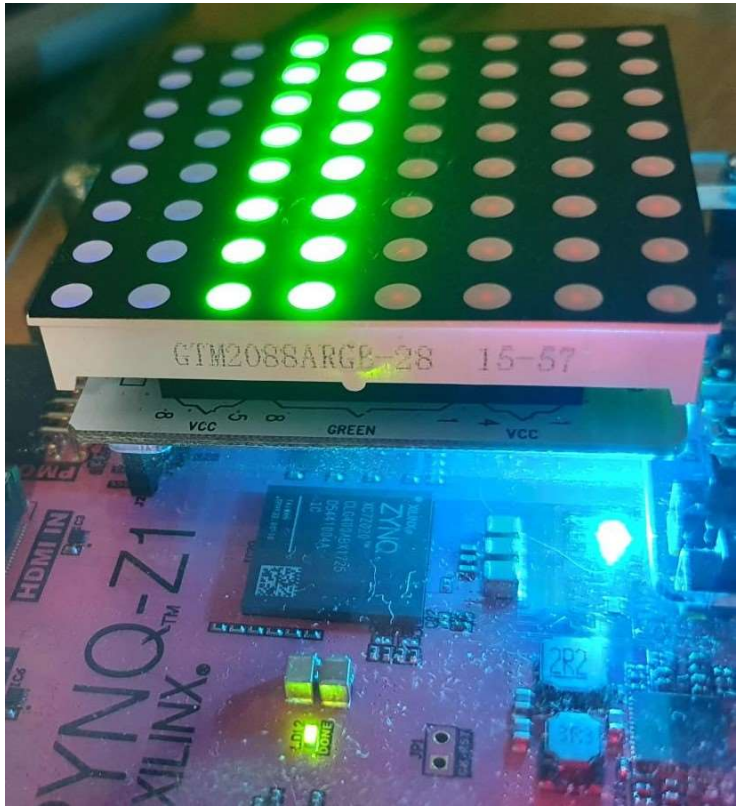


Tampere University

Led matrix

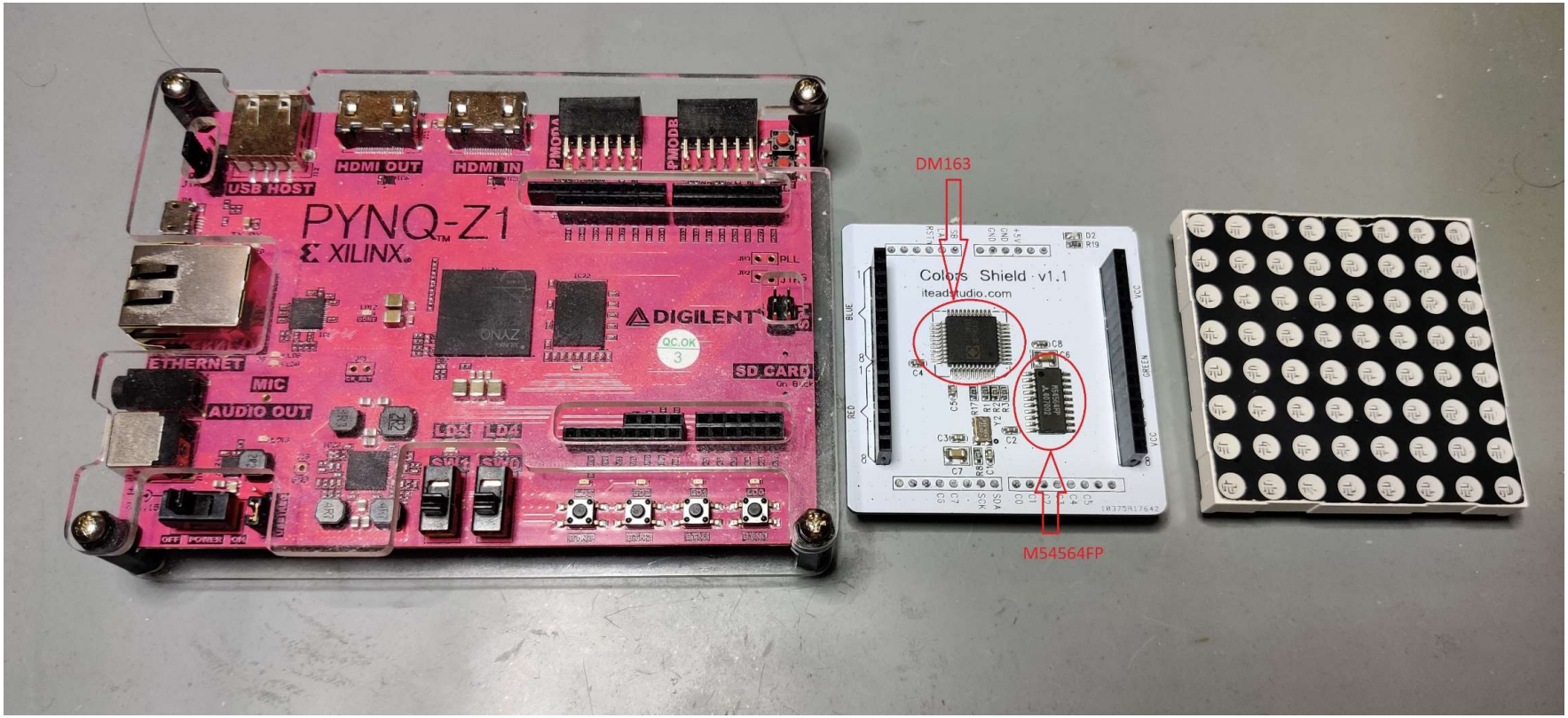
- If you get some random pixels when running the project, turn the matrix 180 degrees and try again!





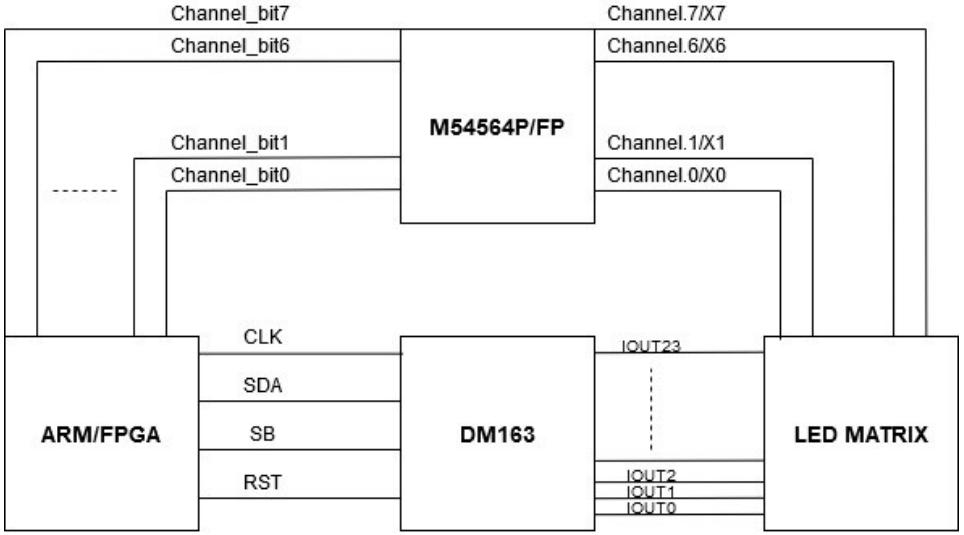
Tampere University

PYNQ LED matrix shield



Tampere University

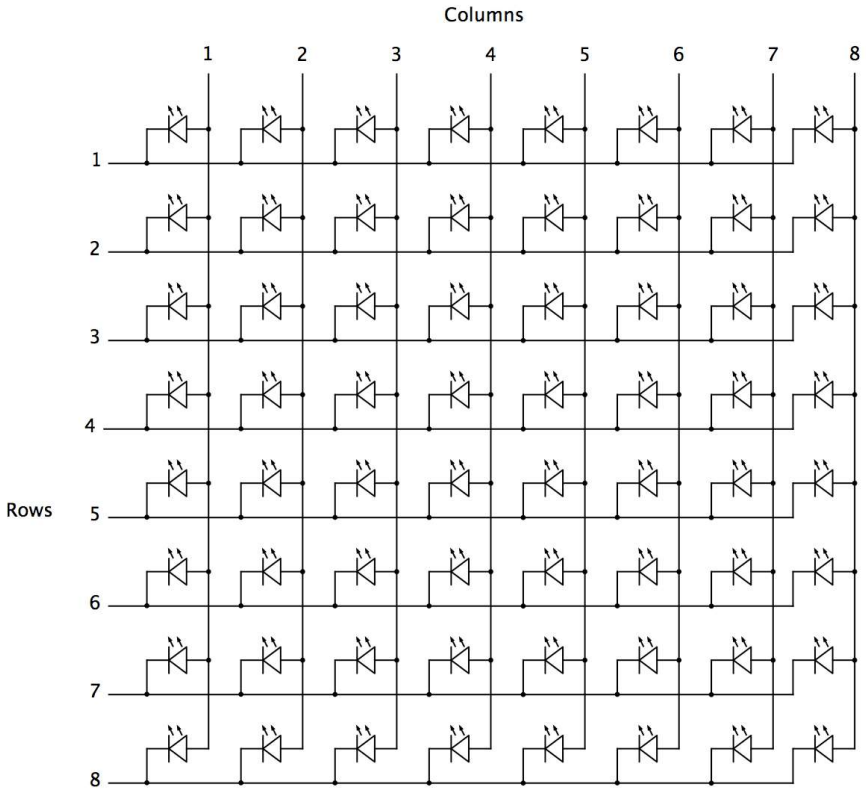
How LED matrix is connected



- ARM controls LED matrix indirectly
  - M54564 works as current source and ARM controls which channel is active
  - DM163 works as current sink

Tampere University

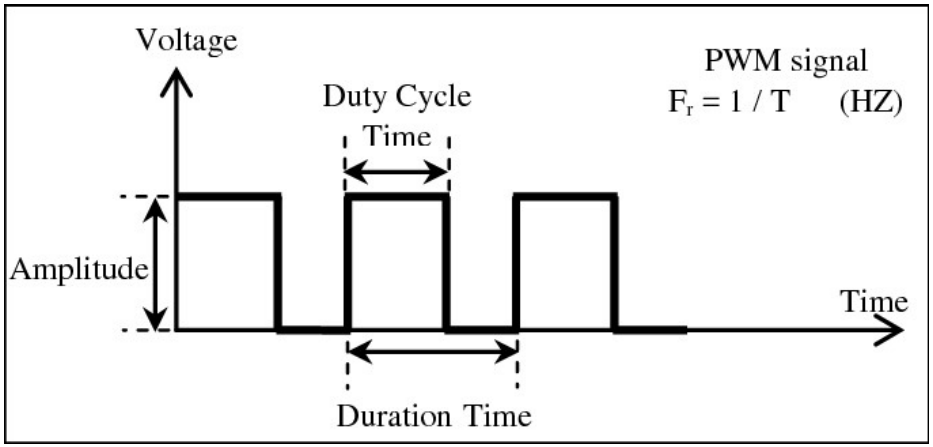
How LED matrix works in general



- This picture shows general idea how LED matrix works
  - Note this picture serves only as an example for one color of LED matrix
- You control vertical columns with channel bits and DM163 controls horizontal row inputs
  - Setting channel bit to ’1’ makes that column active since current can then flow through LEDs

Tampere University

PWM

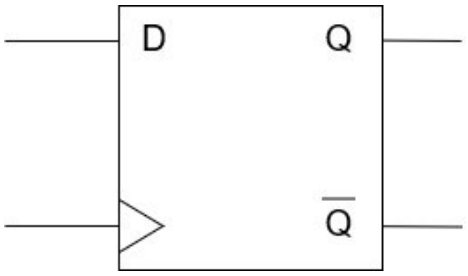


Source: https://www.researchgate.net/figure/PWM-signal-with-its-two-basic-time-periods_fig4_271437313

- PWM (Pulse Width Modulation) is continuing pulse with certain period of ON and OFF time.
- Most common usage is to control power like in this exercise.

Tampere University

D-flip-flop



D Flip Flop Truth Table

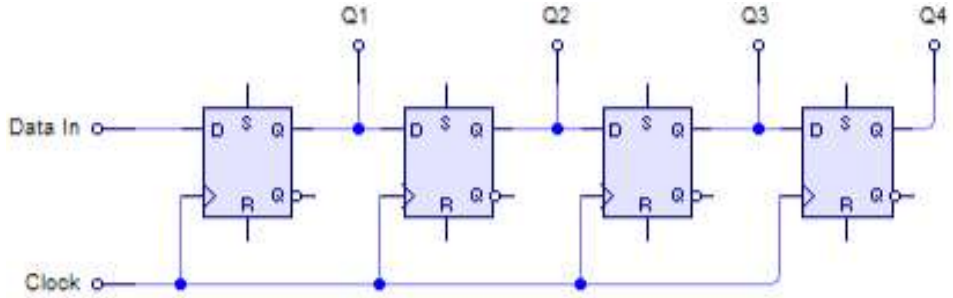
CL (Note 1)	D	Q	$\overline{Q}$
↗	0	0	1
↗	1	1	0
↘	x	Q	$\overline{Q}$
x	x	0	1
x	x	1	0
x	x	1	1

No Change
x = Don't Care Case

- Data out (Q) changes only on rising edge of the clock
  - Q takes value of input D on rising edge
- D-flip-flop is basic component in digital systems
  - Used as synchronizer and in many other applications
  - In this exercise, DM163 chip uses D-flip-flop to form shift registers

Tampere University

Shift register

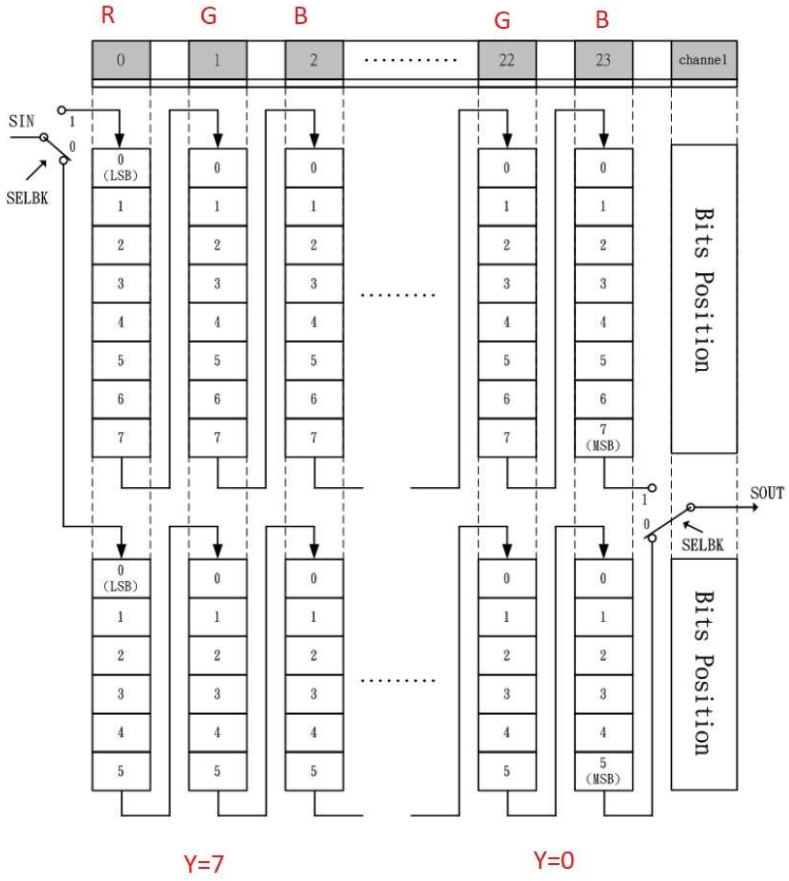


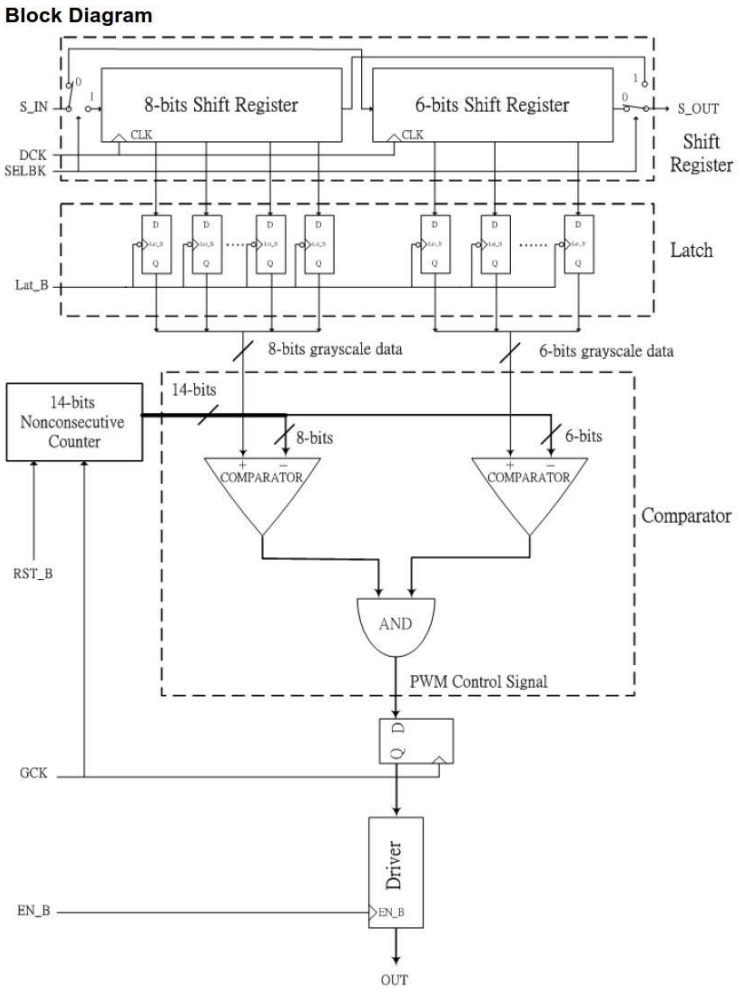
- Shift register is basically multiple D-flip-flops in series.
- Can be used to convert serial data to parallel form.

Tampere University

Shift register and block diagram

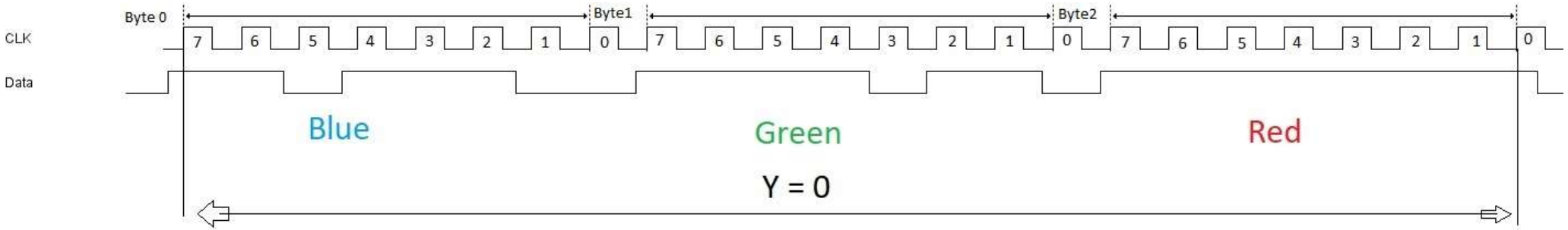
NOTE: Most Significant Byte is Blue





Tampere University

Serial data



NOTE: Bytes transmitted as FIFO (first in, first out)

Other important stuff

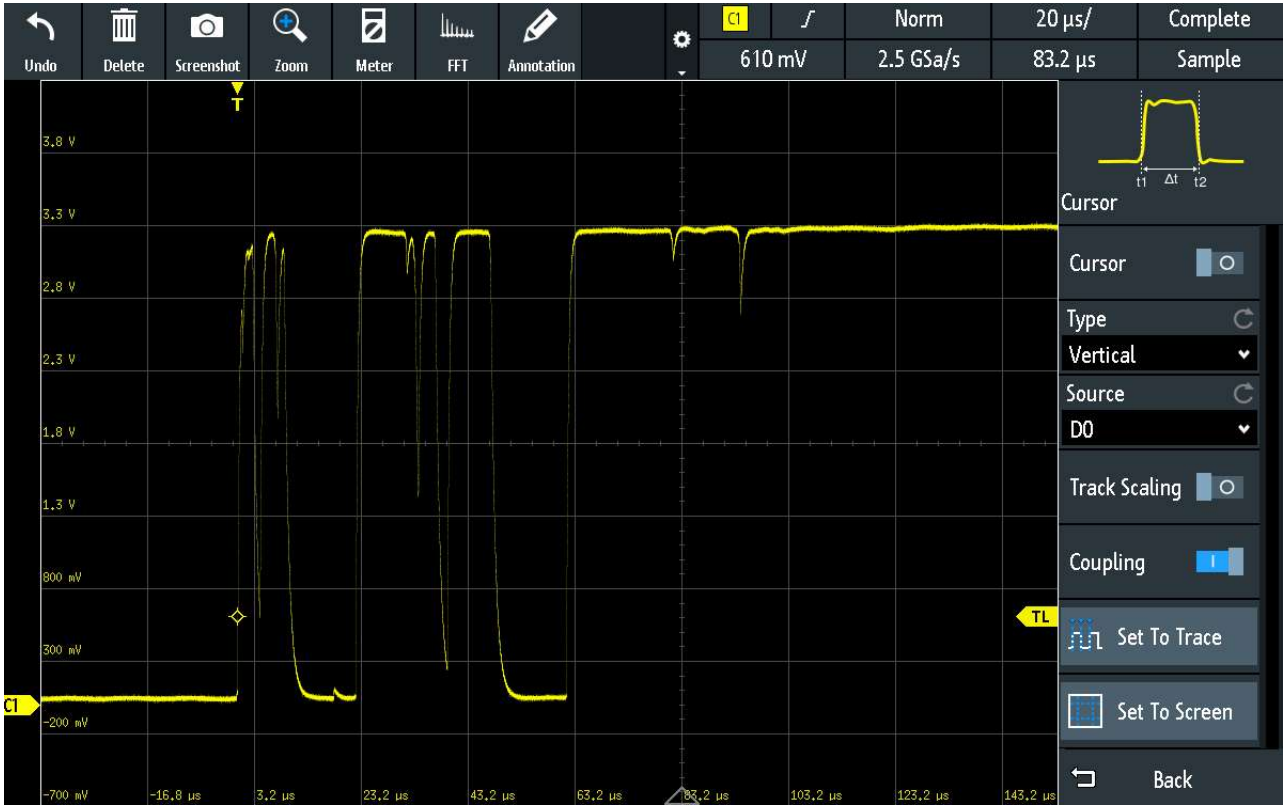
- SB-bit chooses if data is send to 6 or 8-bit register.
- Reset must be set to inactive state before sending data to LED matrix
  - Set it to ’1’, since reset is active low
- 6-bit register banks must have values higher than 0
  - Because pixel brightness (if simplified) = 6-bit register*8-bit register
- Channel bits (C0-C7) are used to set active column

Common problems

- Ghostpixels
  - Screen refresh cycle is a bit faulty (dim pixel right/leftside of ”real” pixel)
    - This is most likely bad programming (Channels active in wrong places)
- Sometimes program reads button to be pressed multiple times with one click
  - Switching vibration-> nothing can be done in this exercise, since it is more of an electrical feature.
  - Easy fix would have been adding a low-pass filter.
  - Another easy fix would be adding debounce circuit (software or hardware).
- In blinker.S, don’t use *;* to comment stuff, use *//* . IDE might show with *;* that stuff is commented, in reality it isn’t.
- Some micro-usb cables do not have datalines (programming PYNQ is impossible)

Tampere University

Button vibration measured with oscilloscope



Tampere University

In case of questions

- If any questions related to exercise
  - Ask in Mattermost
  - Send email: otto.eerola@tuni.fi

Tampere University

END