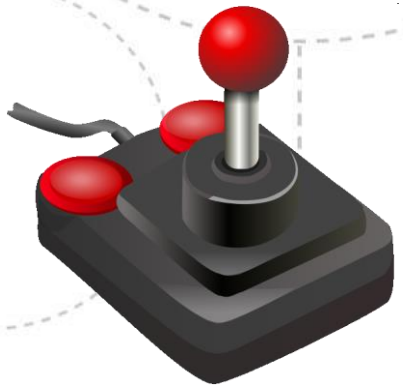
TTK4155

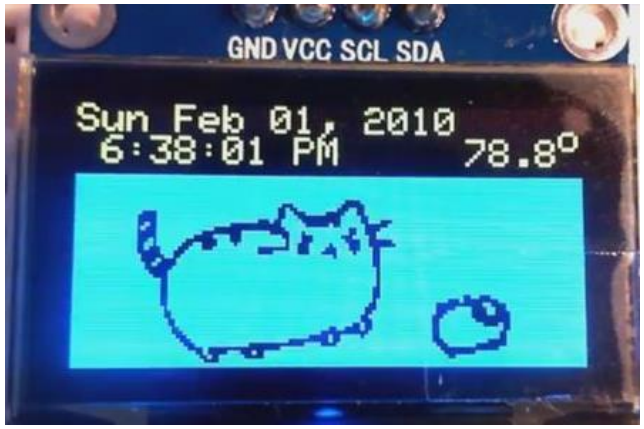
Industrial and Embedded Computer Systems Design

NTNU – Trondheim
Norwegian University of Science and Technology



Lab lecture 2

- Exercises 2, 3 and 4



Reference group

- 3 students.
- Need diversity in programmes of study.
- ....
- ....
- ....

Exercise 2, 3 & 4: Address decoding and memory mapped IO

- In these exercises, you will
  - Use the external memory bus to connect peripherals
  - Partition the address space for accessing 3 units as memory mapped I/O
  - Connect an SRAM, an ADC and an OLED as I/O

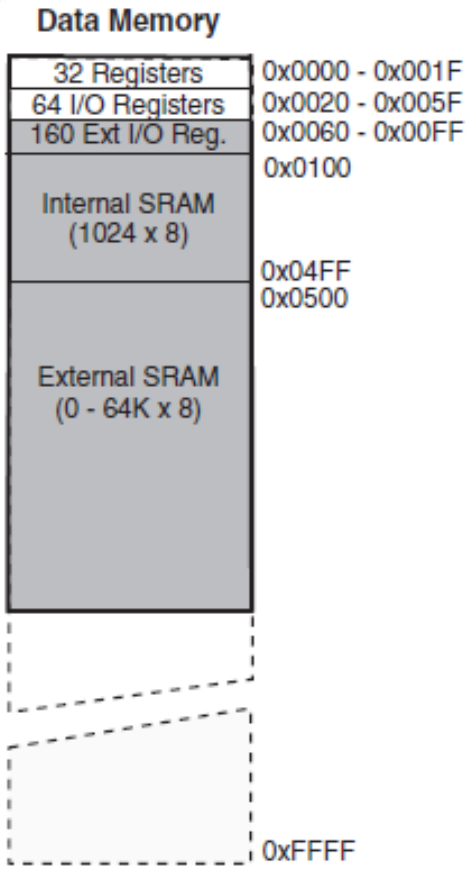
- In this lecture, we will discuss
  - XMEM and memory mapping
  - Address decoding
  - Interface between circuits

Memory mapping

- IO devices as a contiguous memory address space
  - Opposed to x86 special instructions
- From the MCU's point of view,
  - Address space is just a block of memory
  - Physically, this is not necessarily true, various IO devices can be connected
- In our case we will use
  - SRAM
  - ADC
  - OLED display
- Read "Designing Embedded Hardware" (15.6)

Memory mapping in ATmega162

- 16 bit addresses – 64 k address space.
- The first 1280 (0x0000 – 0x04FF) addresses are for registers, I/O, external I/O and internal SRAM.
- The remaining can be used for other units.



NTNU – Trondheim
Norwegian University of
Science and Technology

Datasheet

External Memory Interface

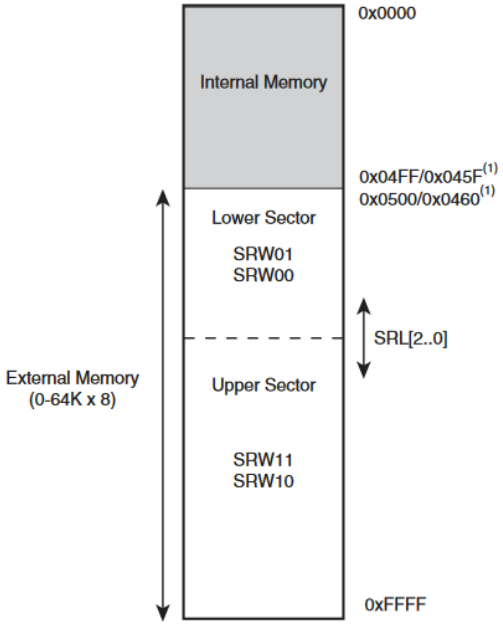
With all the features the External Memory Interface provides, it is well suited to operate as an interface to memory devices such as external SRAM and FLASH, and peripherals such as LCD-display, A/D, and D/A. The main features are:

- **Four Different Wait-state Settings (Including No Wait-state)**
- **Independent Wait-state Setting for Different External Memory Sectors (Configurable Sector Size)**
- **The Number of Bits Dedicated to Address High Byte is Selectable**
- **Bus Keepers on Data Lines to Minimize Current Consumption (Optional)**

Overview

When the eXternal MEMory (XMEM) is enabled, address space outside the internal SRAM becomes available using the dedicated external memory pins (see Figure 1 on page 2, Table 29 on page 70, Table 35 on page 75, and Table 41 on page 81). The memory configuration is shown in Figure 11.

Figure 11. External Memory with Sector Select



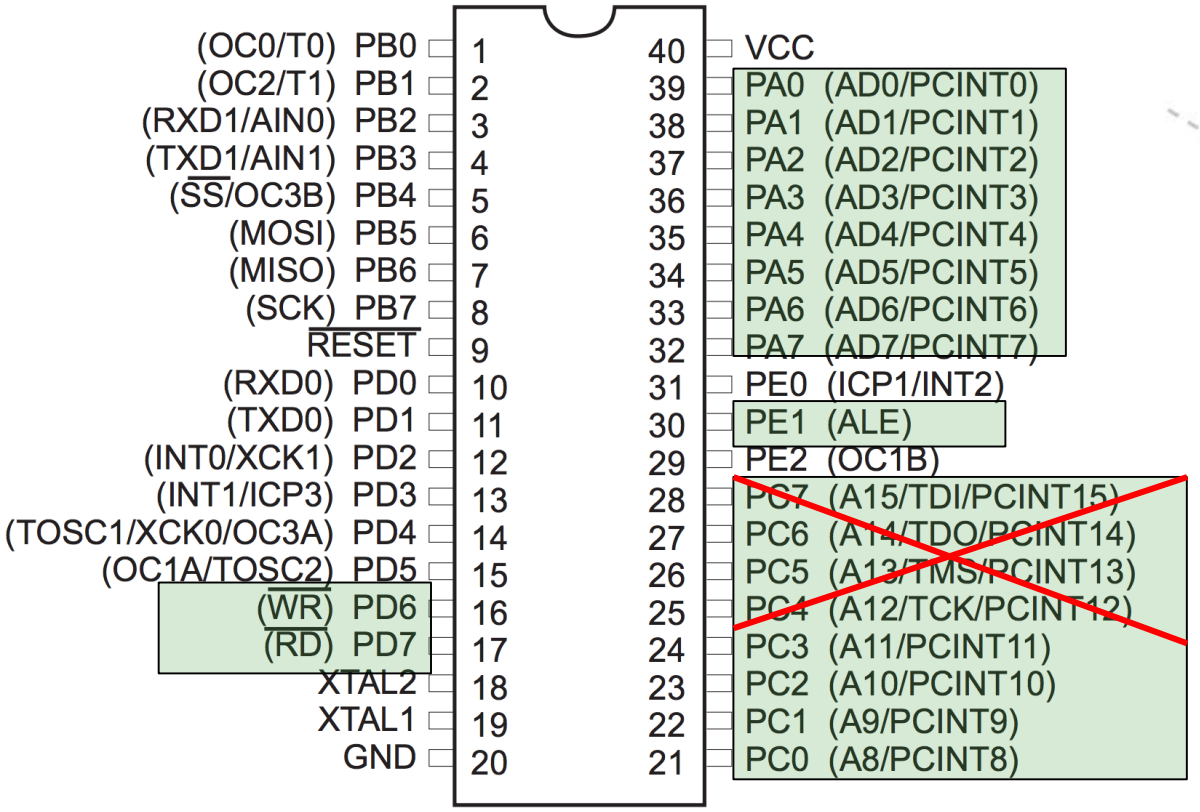
Note: 1. Address depends on the ATmega161 compatibility Fuse. See "SRAM Data Memory" on page 18 and Figure 9 on page 19 for details.

Using the External Memory Interface

The interface consists of:

- AD7:0: Multiplexed low-order address bus and data bus
- A15:8: High-order address bus (configurable number of bits)
- ALE: Address latch enable
- $\overline{RD}$: Read strobe.
- $\overline{WR}$: Write strobe.

Masking PC4-PC7



XMEM code example

- Init. function

```
void xmem_init(void){
        MCUCR |= (1 << SRE);        //enable XMEM
        SFIOR |= (1 << XMM0);       //mask unused bits
}
```

- Write function

```
void xmem_write(uint8_t data, uint16_t addr){
        volatile char *ext_mem = (char *) BASE_ADDRESS;
        ext_mem[addr]= data;
}
```

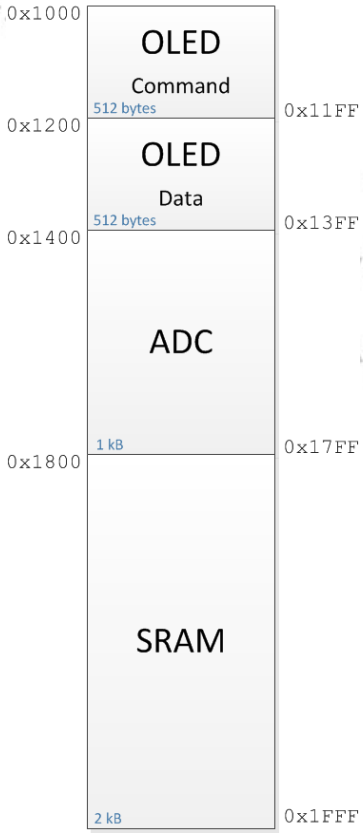
- Read function

```
uint8_t xmem_read(uint16_t addr){
        volatile char *ext_mem = (char *) BASE_ADDRESS;
        uint8_t ret_val=ext_mem[addr];
        return ret_val;
}
```

```
#define BASE_ADDRESS 0x1000
```

Suggested address partitioning

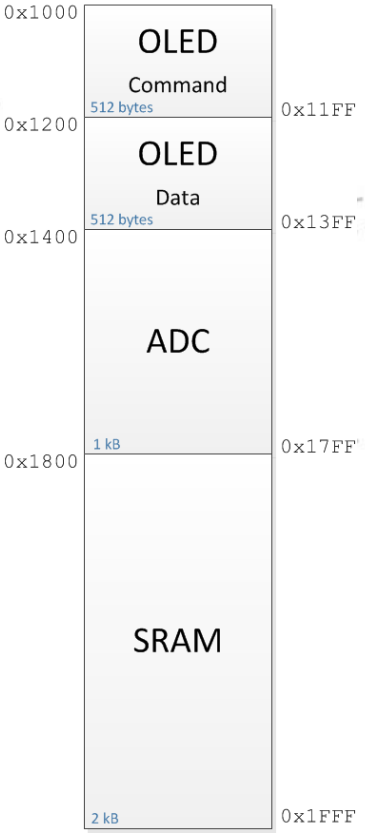
Unit	From – to (hex)	From – to (binary)			
OLED	0x1000	0001	0000	0000	0000
	0x13FF	0001	0011	1111	1111
	CS when:	0001	**00**XX	XXXX	XXXX
ADC	0x1400	0001	0100	0000	0000
	0x17FF	0001	0111	1111	1111
	CS when:	0001	**01**XX	XXXX	XXXX
SRAM	0x1800	0001	1000	0000	0000
	0x1FFF	0001	1111	1111	1111
	CS when:	0001	**1**XXX	XXXX	XXXX

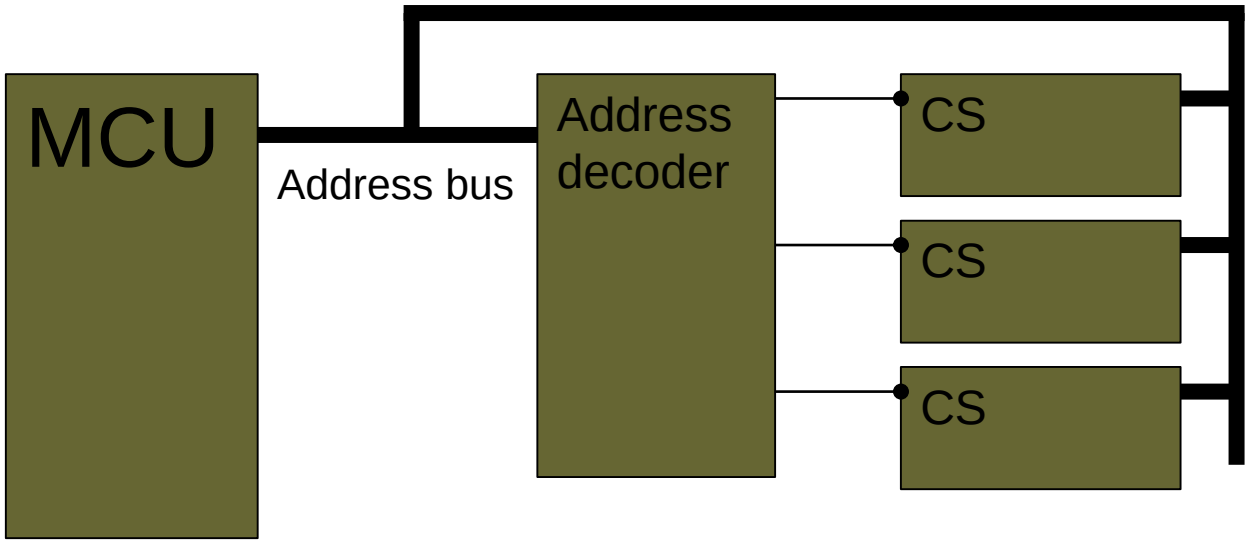


NTNU – Trondheim
Norwegian University of Science and Technology

Address decoding

- Shared data and memory bus → only one IC should be active at any time
- An address decoder handles this



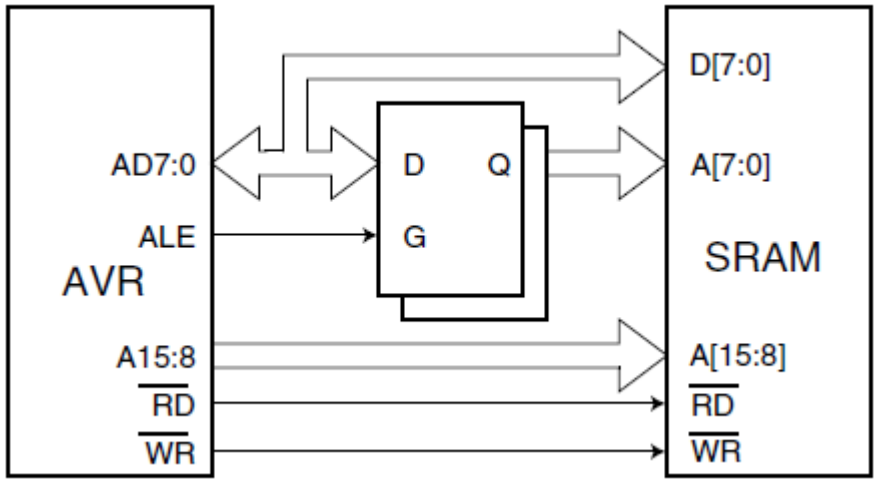


Implementing the address decoder

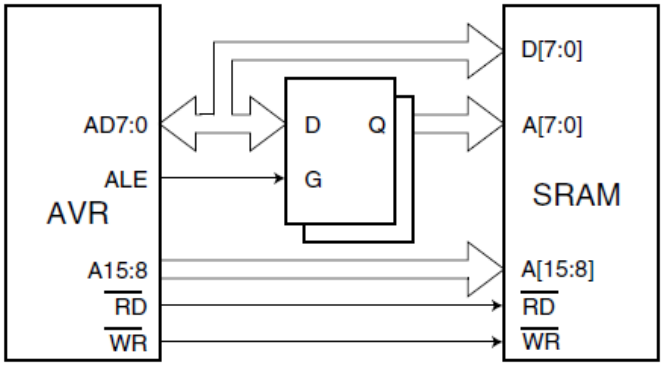
- By looking at the address bits, it should be possible to determine which unit that should be activated
- Simple Boolean logic
- Generic Array Logic (GAL)
  - Programmable logical ports
  - VHDL
  - Programming (”burning”)
- The assignment text contains a nearly complete example for VHDL and step-by-step instructions for programming the GAL

Interface with SRAM

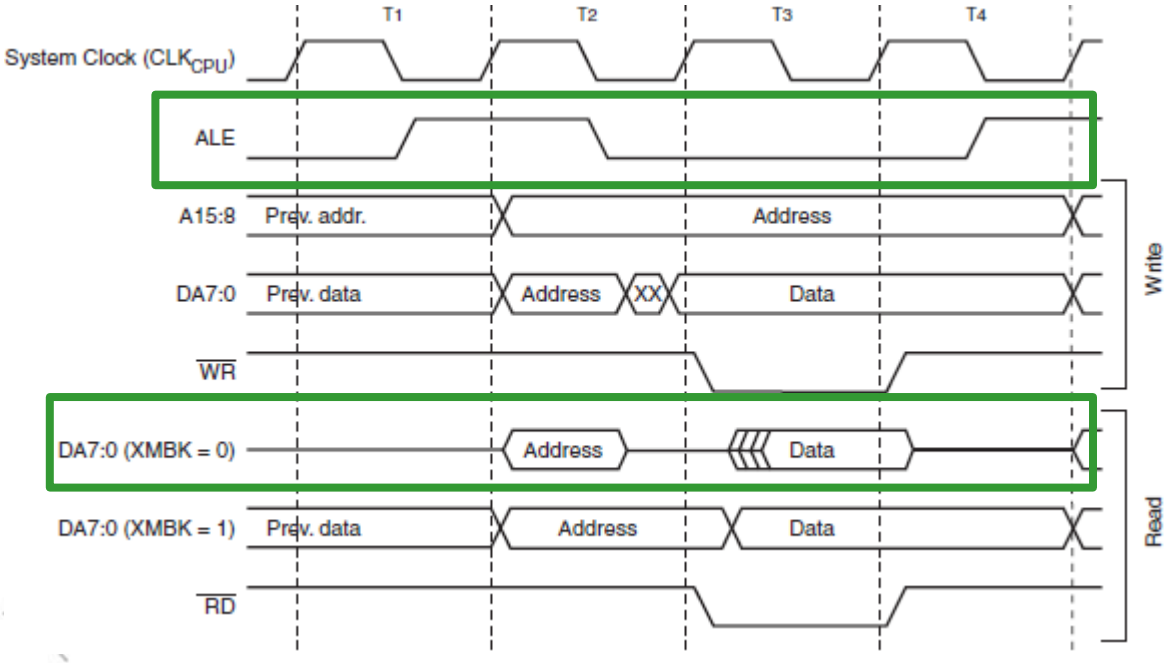
- AD7:0 – lower address bus AND data bus
- A15:8 – upper address bus
- ALE – Address Latch Enable
- RD – Read strobe
- WR – Read strobe



Latch

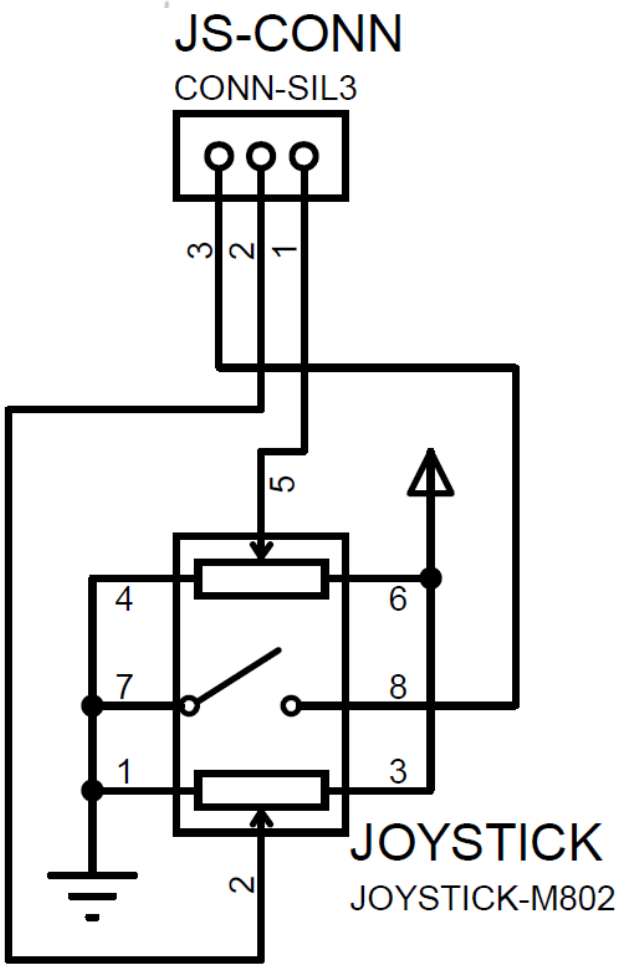


- To deal with address and data on the same pins
- First address, then data
- The address needs to be ”remembered” → Latch



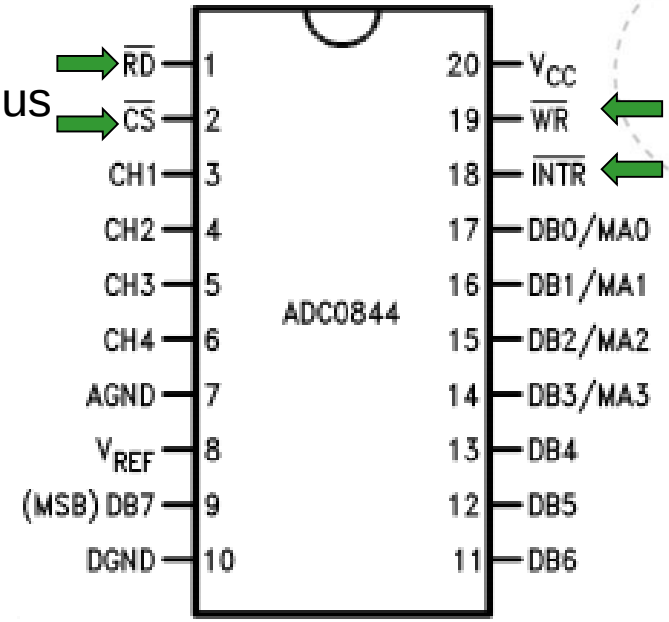
Ex_3:Joystick

- Two axis (X, Y)
  - Variable resistance
- One button
  - Short circuiting
- How to read position and buttons?
  - Button => PINx & 0x01

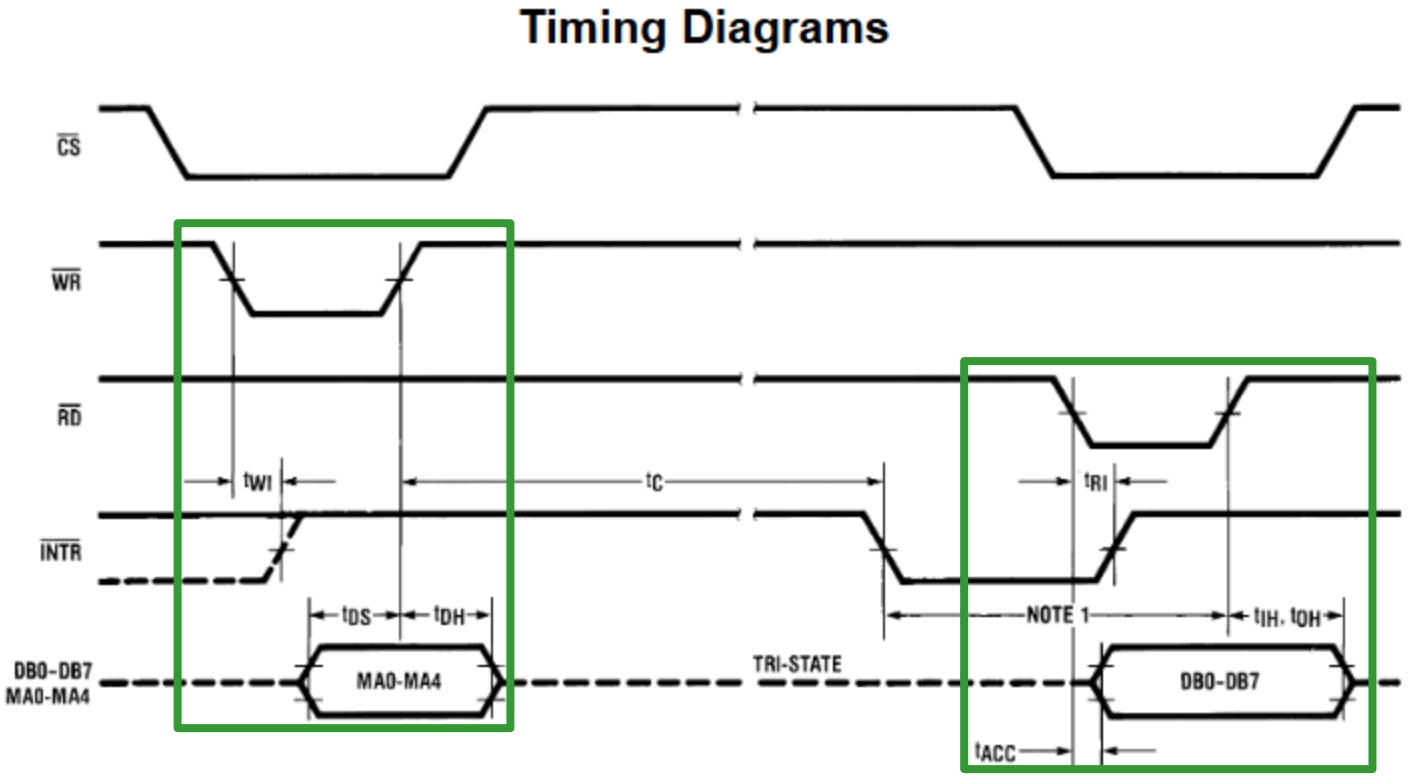


Ex_3:ADC in this project

- ADC0844CNN from National Semiconductor
  - 4 channels, SAR 8-bit ADC
  - Read analog value: 40 µs
  - 8 bit parallel data and mux address latch bus
  - CS
  - Single ended / differential
  - Interrupt output



Ex_3:ADC timing diagram



/WR and /RD => outputs from AVR
/INTR => input to AVR
/CS => output from GAL

Ex_3:Tips

- Use single ended ADC.
- Useful functions
  - adc_init (void);
  - uint8_t adc_read(uint8_t channel); \\volatile
  - adc_calibrate();
  - pos_t pos_read(void);
- Sliders => output is PWM use on-board LP filter and connect with ADC.

EX_4:OLED useful functions

- OLED_init(); \\ PDF:“OLED LY190-128064” section 9.4
- OLED_reset();
- OLED_home();
- OLED_goto_line(line);
- OLED_clear_line(line);
- OLED_pos(row, column);
- OLED_write_data(char); \\volatile
- OLED_print(char*);
- OLED_set_brightness(lvl);

EX_4:Example

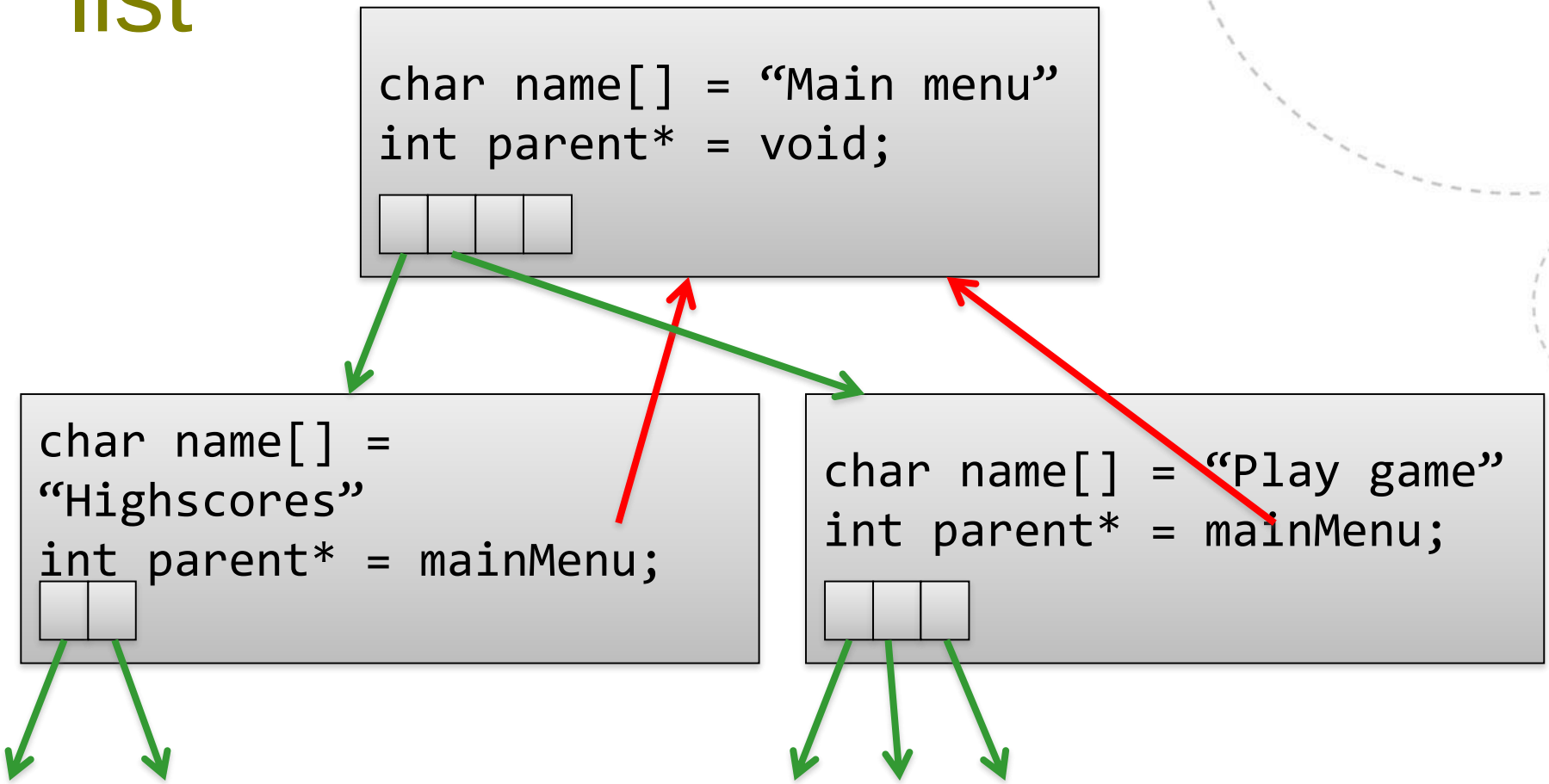
- Maybe you need an arrow?

```
void OLED_print_arrow(uint8_t row, uint8_t col)
{
  OLED_pos(row, col);
  OLED_write_data(0b00011000);
  OLED_write_data(0b00011000);
  OLED_write_data(0b01111110);
  OLED_write_data(0b00111100);
  OLED_write_data(0b00011000);
}
```

EX_4:Menu system

- Goal => selection and navigation is possible
- Ideas for menu subsystems:
  - Start new game.
  - See/reset high score.
  - Calibrate joystick.
  - Set difficulty
  - Debugging
- Can be a very simple or advanced (counted as extras).
- Take some time to design basic framework before implementation. Maybe extras are future additions.

EX_4:Menu suggestion – linked list



EX_4:Tips

- Remember to connect a jumper to EXTSEL on the multifunction board (see schematic).
- OLED initiation code is in the datasheet “OLED LY190-128064” in section 9.4.
- Read about storing static data in program memory (google AVR PROGMEM) .
- The character set is quite big; it would be a good idea to remove unused characters to reduce program size.
- 2 weeks! spend some time on brainstorming…

More tips…

- Use resources not mentioned in lab manual
  - Timers, use them as event scheduler e.g. read ADC, update OLED, send messages etc.
  - Interrupts, a bit ‘advance’ feature but can be very efficient
  - Energy saving modes, check sleep modes of atmega2560
  - For PID controller, use of a timer is a requirement for I & D parts
  - Use of debugger, online debugging is highly recommended
- Organize your code in device drivers/files

Questions?

Auf wiedersehen