TTK4155

Industrial and Embedded Computer Systems Design

NTNU – Trondheim
Norwegian University of Science and Technology

Lab lecture 2

- Address decoding and external RAM
- Problems from Ex1

General information

- Lab lecture => anything you want to add or remove? If you think more/specific data should be included please send me an email.
- Remember to get assignments approved.
- Lab access => You should have by now.
- Use QMS. Username=grN and password=grN (1<N<60)
- Arduino IDE => NOT allowed.

Reference group

- Need minimum 3 persons.
- Need diversity in programmes of study.
- As I mentioned one person has already signed. Please paritcipate....

Exercise 2: Address decoding and external RAM

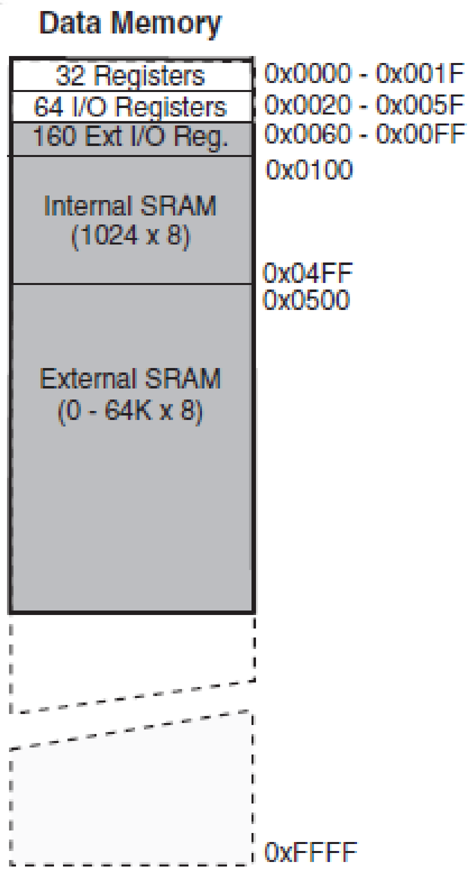
- In this exercise, you will
  - Begin using the external memory bus to connect peripherals
  - Partition the address space to accommodate for accessing several units as memory mapped I/O
  - Generate address decoding logic
  - Connect an SRAM IC
- In this lecture, we will discuss
  - Memory mapping
  - ATmega162 memory
  - Address decoding
  - Interface between circuits

Memory mapping

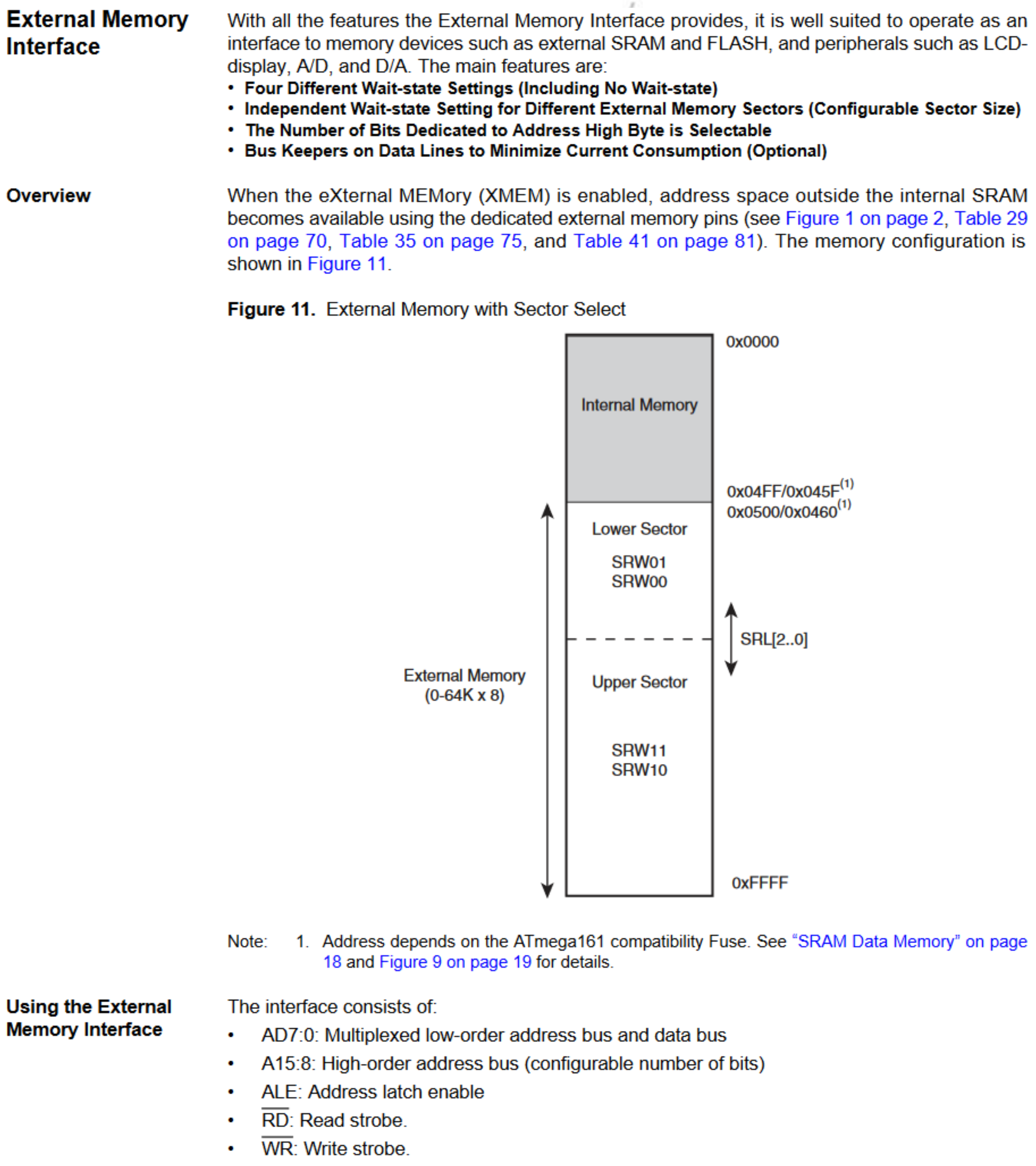
- The concept of accessing different units as a contiguous memory address space
- From the MCU's point of view, the address space is just a block of memory
  - Physically, this is not necessary true
  - Many different devices (memory and other I/O) can be connected
- We will partition the address space to accommodate for communicating with three external devices:
  - 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.



Datasheet



External Memory Interface

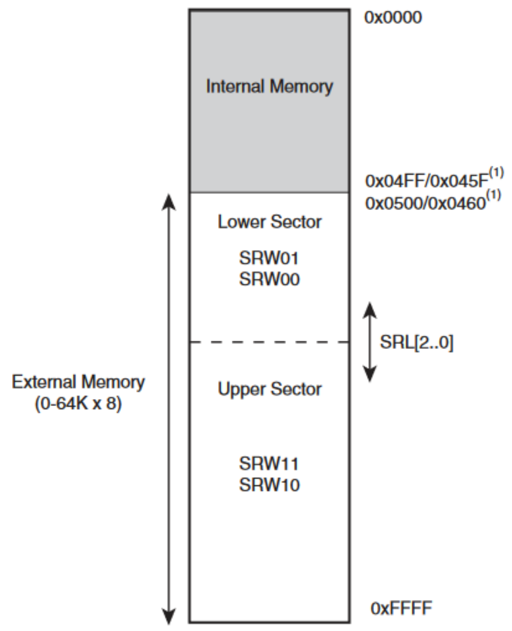
With all the features the External Memory Interface provides, it is well suited to operate as an interface to memory devices such as external SRAM and FLASH, and peripherals such as LCD-display, A/D, and D/A. The main features are:

- **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.

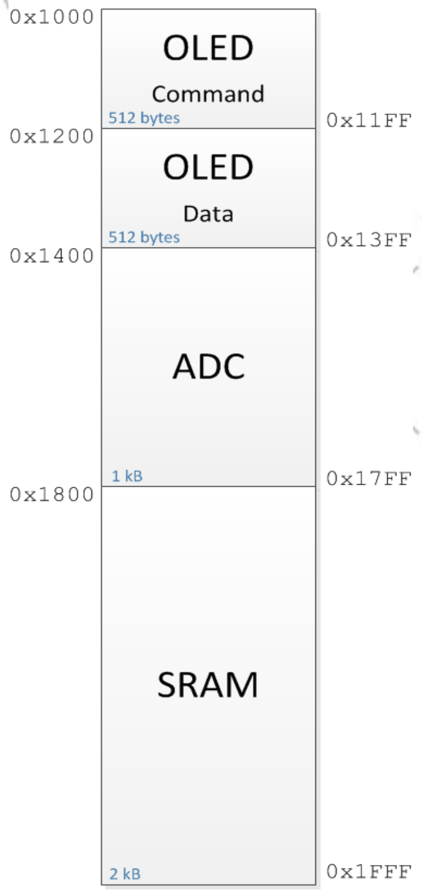
Using the External Memory Interface

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
- $\overline{\text{RD}}$: Read strobe.
- $\overline{\text{WR}}$: Write strobe.

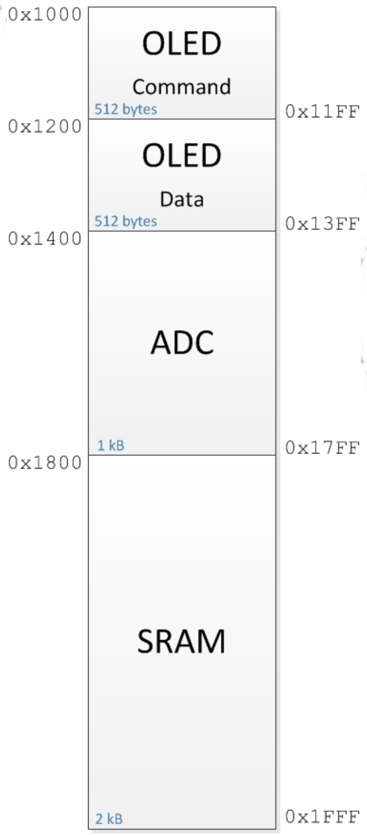
Memory mapping in this exercise

- We have more addresses than we need
- When using different addresses…
  - SRAM: different bytes are accessed
  - ADC: activated or deactivated
  - OLED: data or command write
- Follow the suggested partitioning
- Note – we “lose” 4 bits from the addresses because of the JTAG interface



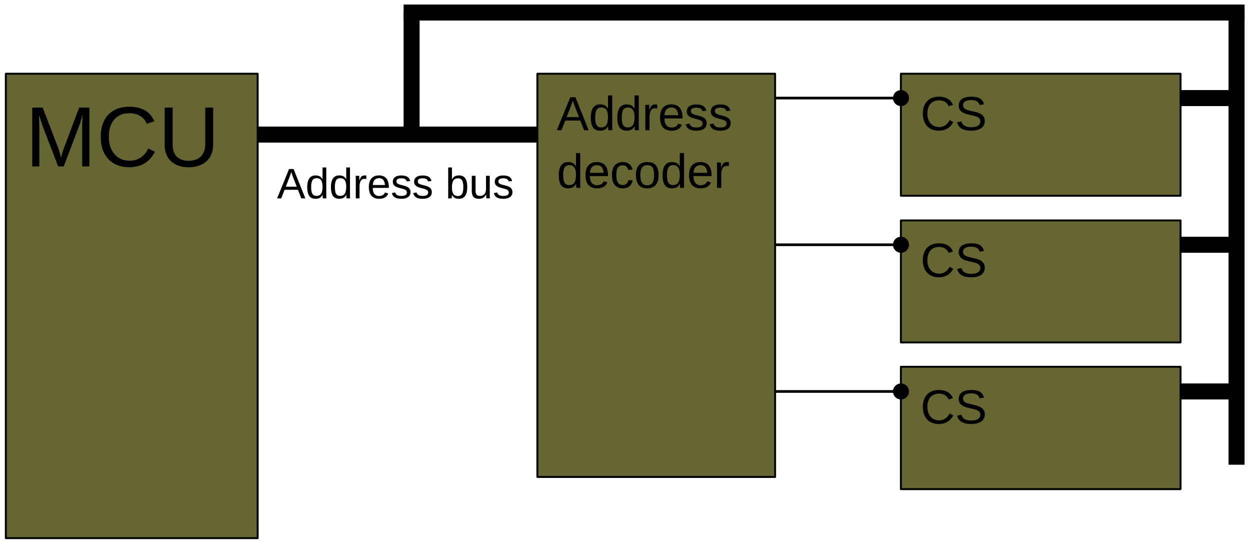
Suggested address partitioning

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

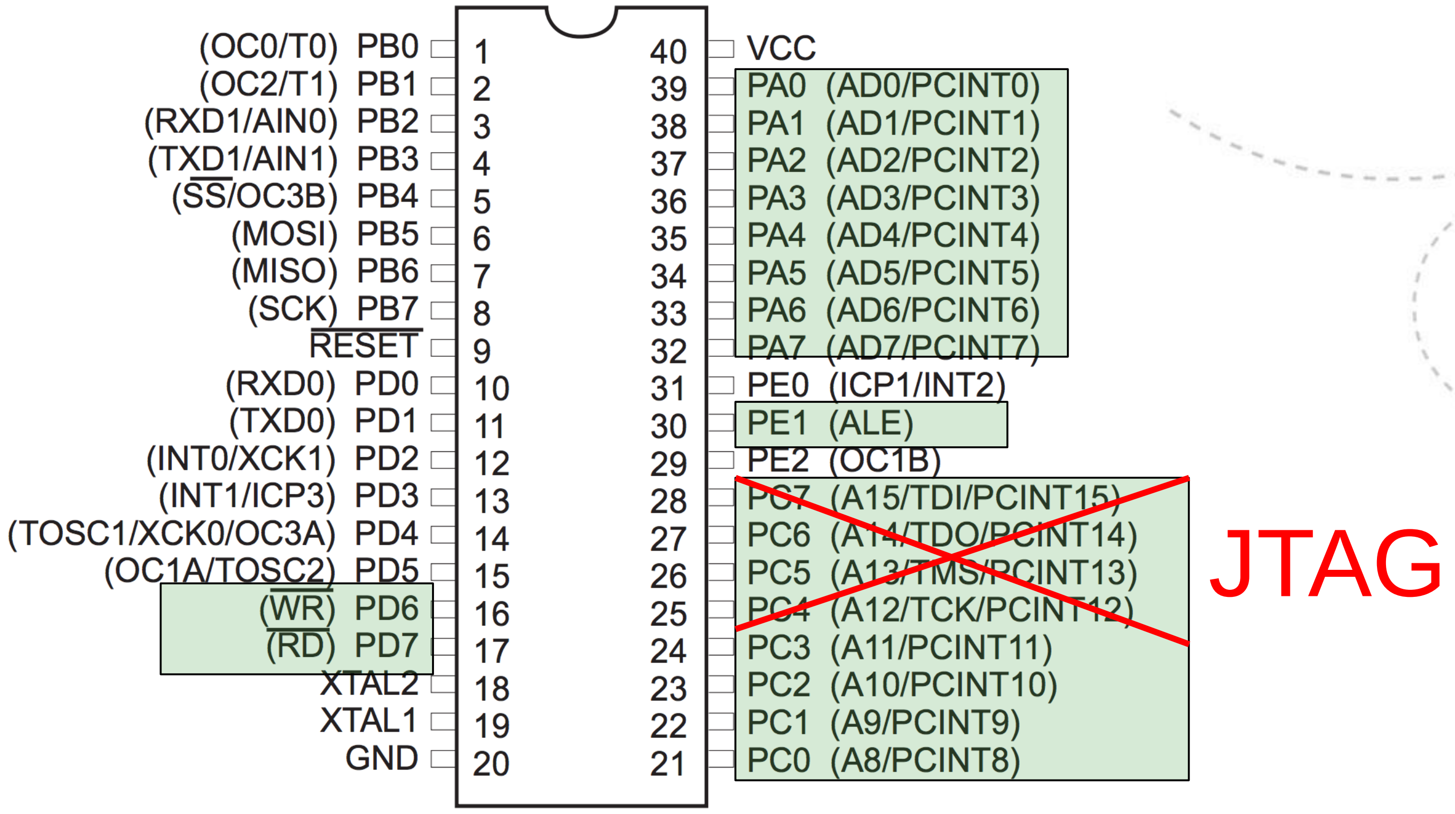


Address decoding

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



Address decoding



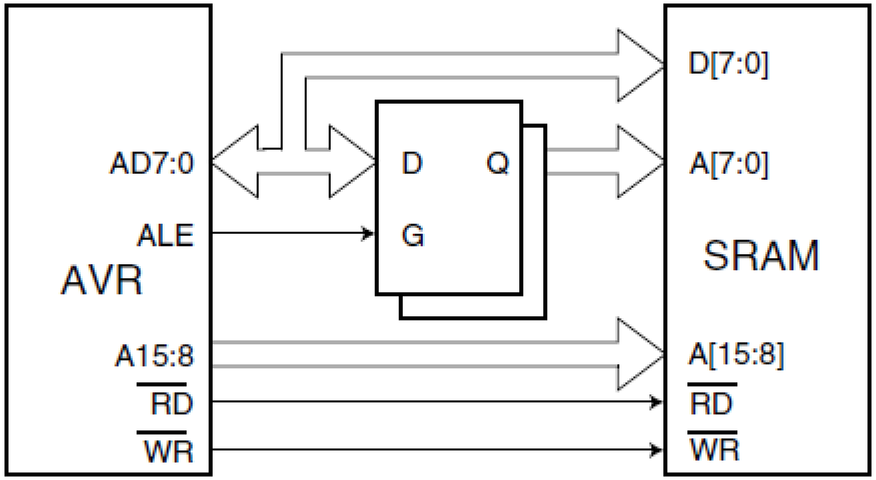
NTNU – Trondheim
Norwegian University of
Science and Technology

Implementing the address decoder

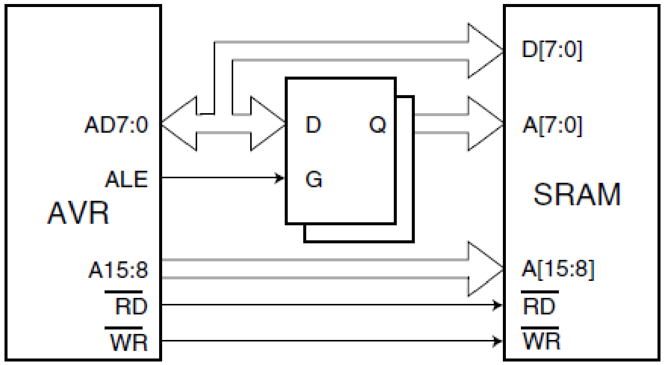
- By looking at the address bits, it should be possible to determine which unit that should be activated
- Simple Boolean logic
- General Array Logic (GAL)
  - Programmable logical ports
  - VHDL
  - Programming (”burning”)
- The assignment text contains a nearly complete example for VHDL and step-by-step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

Interface between circuits

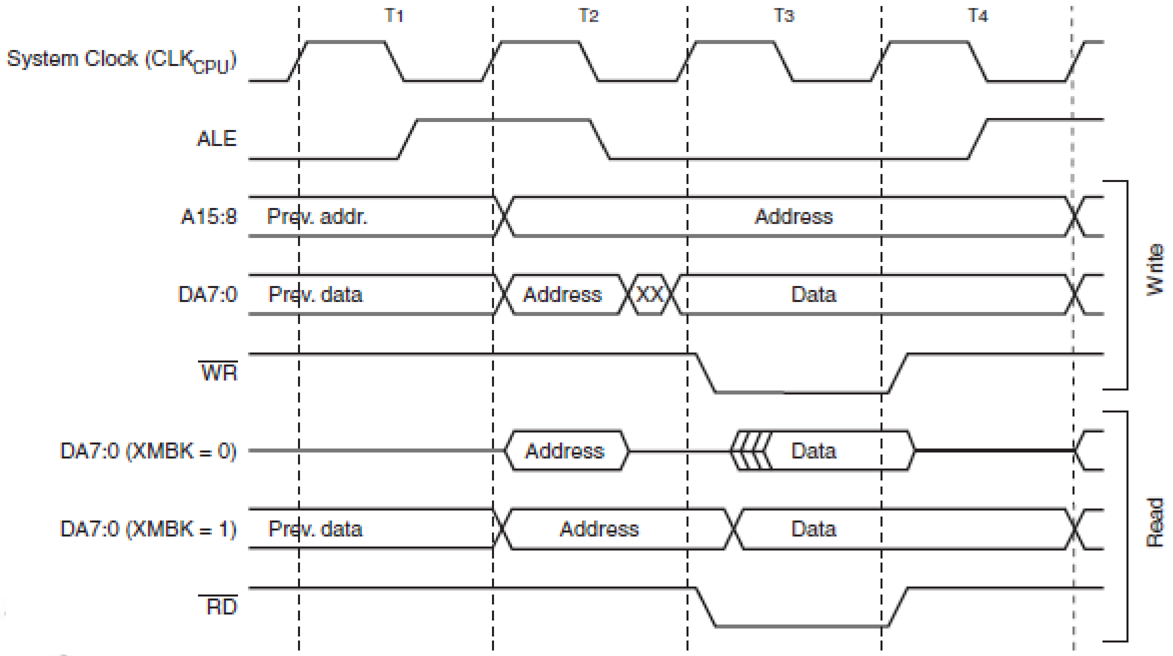
- AD7:0 – lower address bus AND data bus
- A15:8 – upper address bus
- ALE – Address Latch Enable
- RD – Read strobe
- WR – Read strobe



Latch



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

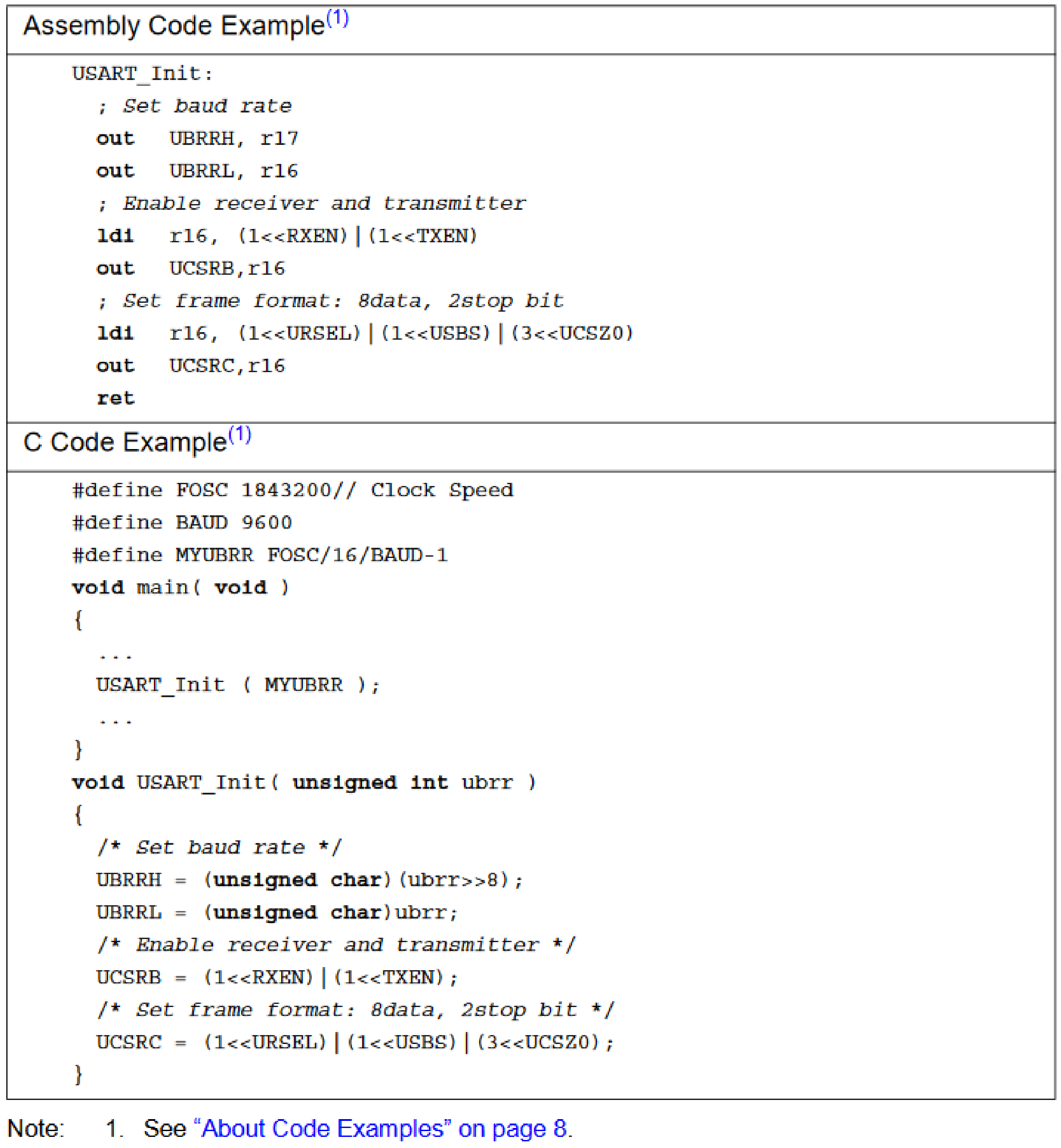


NTNU – Trondheim
Norwegian University of Science and Technology

Common mistakes EX1

- BAUD rate error
  - Symptoms: Jibberish output
  - Fuses: CKDIV8, EXT XTAL
  - Verify the CPU speed
- RX<->TX wiring
  - Symptom: No output
- Setting pullup instead of output
  - PORTx = 1, without DDRx = 1
- Holding reset down externally
  - Can program, but the program don’t run…
- Read data sheet code snippets.

Datasheet



Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out   UBRRH, r17
  out   UBRRL, r16
  ; Enable receiver and transmitter
  ldi   r16, (1<<RXEN)|(1<<TXEN)
  out   UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi   r16, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
  out   UCSRC,r16
  ret
```

C Code Example[1]

```
#define FOSC 1843200// Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
  ...
  USART_Init ( MYUBRR );
  ...
}
void USART_Init( unsigned int ubrr )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(ubrr>>8);
  UBRRL = (unsigned char)ubrr;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Note: 1. See "About Code Examples" on page 8.

Questions?

Auf wiedersehen