NTNU

Kandidat nr./*Candidate no.* 10104

Dato/*Date:* 30.11.2017 Side/*Page:* 1

Emnekode/*Subject* TTK4155

Antall ark/*Number of pages:* 14

Denne kolonnen er forbeholdt sensor

This column is for external examiner

Problem 1

a) TCN2 will trigger every 2^8 cycle. Let P be the pre-scaler and the desired frequency be 1 Hz. Then

$$\frac{F_{clk}}{P \cdot 2^8} = 1 \text{ Hz}$$

$$\Rightarrow P = \frac{F_{clk}}{2^8 \text{ Hz}}$$

$$= \frac{32768}{256}$$

$$= 128$$

Prescaler of 128 means

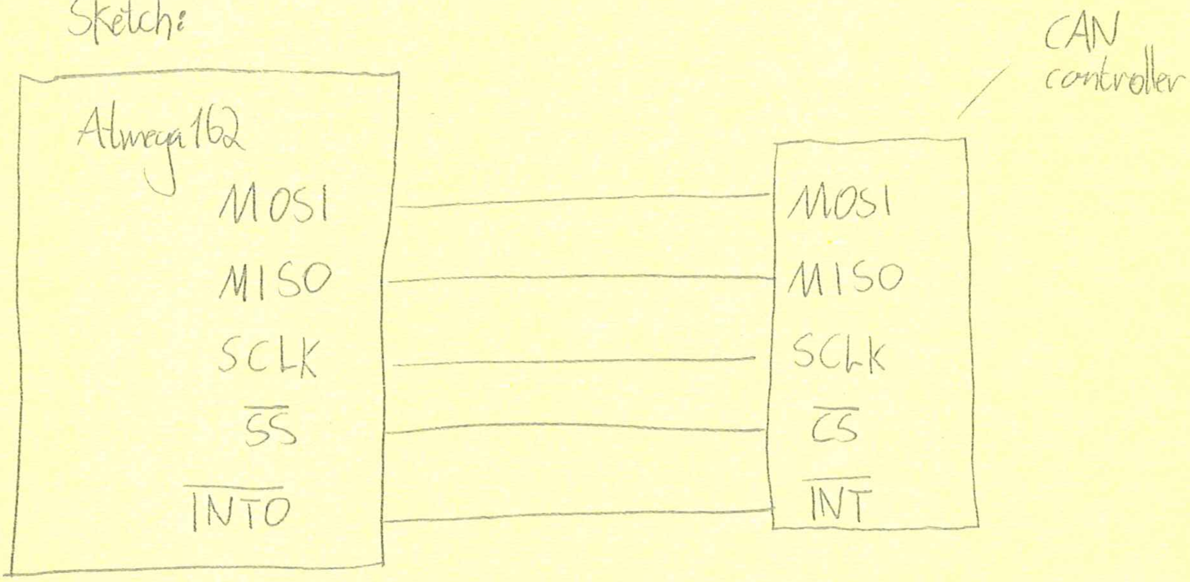
$CS0 = 1$

$CS1 = 0$

$CS2 = 1$

b) We connect the SPI related lines without any creativity, and to get interrupts we also connect the interrupt pin of the CAN controller to an external interrupt pin of the atmega162, for instance PD2 (INT0)

Sketch:



c) Assuming TOV2 is set up as in 1a, and that the system wont operate for more than 136 years ~ 2^{32} seconds.

```
static volatile uint32_t second_count;
static volatile uint32_t minute_count;
static volatile uint32_t hour-count;
static volatile uint16_t day-count;

ISR(TOV2_vect){
    ++second_count;
    if (second-count % 60 == 0){
      ++Minute_count;
      if (minute_count % 60==0){
          ++hour-count;
          if (hour-cont % 24 == 0){
            ++day-count;
          }
      }
    }
    //Build CAN_msg_t ,
    CAN_msg_t msg = {};
    msg.dlc = 4; //length is 4 bytes
    msg.dword_data[0] = second-count; //4 byte data access
                          should also have an id somewhere.
    CAN_transmit(msg);
}
```

NTNU

Kandidat nr./*Candidate no.* 10104

Dato/*Date:* 30.11.2017 Side/*Page:* 4

Emnekode/*Subject* TTK4155

Antall ark/*Number of pages:* 14

Denne kolonnen er forbeholdt sensor

This column is for external examiner

d) Since Node B has the same hardware, we set it up with the same interupt setup as in NodeA. We will use the external interupt from the CAN controller to read heartbeat messages, and we will use the overflow interupt with 1Hz (TOV2-vect) to monitor the status.

```
ISR(INT0_vect){
    CAN_msg_t msg = CAN_get_msg();
    if( msg.id == NODE_A_HEARTBEAT_ID){
        prev_hearbeat_A = heartbeat_A;
        heartbeat_A = msg.dword_data[0];
    }
    // clear interrupt aswell
}
ISR(TOV2_vect){
    if( heartbeat_A - prev_heartbeat_A > 2){
        ALARM_trigger();
    }
    prev_heartbeat_A--;
}
```

Denne kolonnen er forbeholdt sensor

This column is for external examiner

For (d) I assumed that the ~~variables~~ declaration

static volatile uint32_t prev_heartbeat_A, heartbeat_A;

~~was~~ was placed at the top.

For (c) and (d) I have assumed that the ~~…~~ CAN_msg_t struct looks something like this:

```
struct CAN_msg_t_ {
    uint32_t id;  // standard + extended identifier
    uint8_t dlc;  // data lenthg code
    union {
        uint8_t data[8];
        uint32_t dword_data[2];
    }
}
```

Kandidat nr./*Candidate no.* 10104

Dato/*Date:* 30.11.2017 Side/*Page:* 6

Emnekode/*Subject* TTK4155

Antall ark/*Number of pages:* 14

Denne kolonnen er forbeholdt sensor

This column is for external examiner

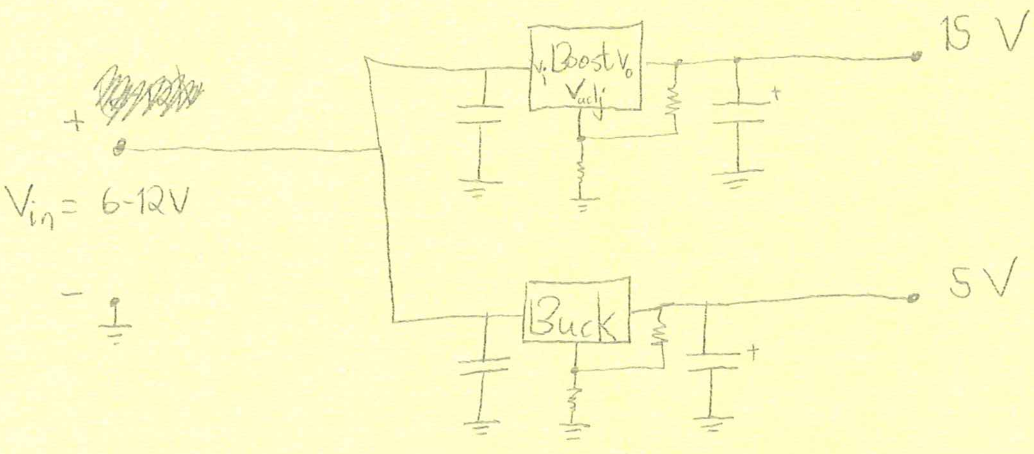
Problem 2

a) Since ~~...~~ node A will have to regulate the voltage both up and down, ~~...~~ (in two separate ways), our best bet is to employ ~~to~~ two switching converters, ~~...~~

A Buck - converter will be used to step down to 5V and a Boost - converter will be used to step up to 15V.

To ensure stable voltages, we should use capacitors for input filtering and output filtering. ~~...~~

This sort of setup could probably be realized with a single Buck-Boost IC, but for clarity I will draw two separate:



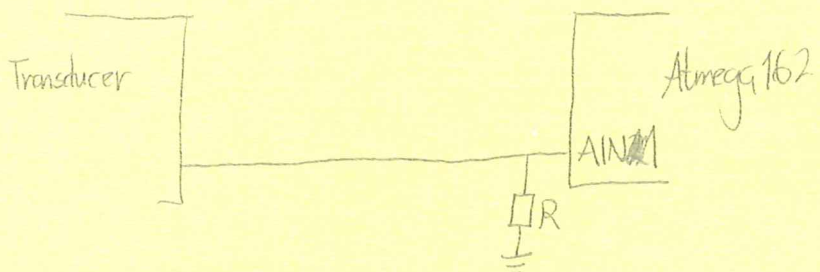
b) To generate a 10 ms = 0,010 s trigger signal, we should set

$$OCR1B = F_{CPU} \cdot 0.010\,s$$
$$= 4000000 \cdot 0.010$$
$$= \underline{\underline{40\,000}}$$

Note that we don't need a prescaler since this is a valid 16-bit value.

c) We can use the current pulse and feed it into the analog comparator AIN1 of the atmega162. In order to trigger an interrupt, we will set the ACBG and enable ACIC.
The comparator output will then be high unless the current pulse causes the voltage at AIN1 to rise above 1.1 Volts. Using ACIS = 10, we get interrupt on falling edge of the comparator.

To guarantee that the input voltage will be 1.1 V, we will connect it as follows



NTNU

Kandidat nr./*Candidate no.* 10104

Dato/*Date:* 30.11.2017 Side/*Page:* 8

Emnekode/*Subject* TTK4155 Antall ark/*Number of pages:* 14

Denne kolonnen er forbeholdt sensor

This column is for external examiner

Assuming the atmega162 has infinite input impedance on AIN1, the voltage will be determined by Ohms law:

$$V = R \cdot I$$

$$\Rightarrow R = \frac{V}{I}$$

To be on the safe side we make $V = 1.5V$ for the minimum current $2.5 mA$

$$\Rightarrow R = \frac{1.5\ V}{2.5\ mA} = \underline{600\ \Omega}$$

~~We now have a way to detect the return signal from the transducer. All that remains is to use another interrupt timer to time the time between sound pulse and returned signal.~~ Actually we only need to look at the value in the counter 1 register to find out how long it took. The value of TCNT will provide time information which can then be converted to distance. (level).

Denne kolonnen er forbeholdt sensor

This column is for external examiner

d) ~~static volatile~~

```
#define SPEED_SOUND 340.0f
#define TANK_HEIGHT 10.0f // random value
static volatile float level;
static volatile uint16 measurement;

ISR(TIM1_COMPB_vect){ // output compare match B
    // 10 ms have passed, read signal and transmit

    level = TANK_HEIGHT - SPEED_SOUND * measurement
                          (F_CPU * 2);
    // Disable this interrupt
}
ISR(ICF1_vect){
    // current pulse detected, store counter register
    measurement = TCNT1;
}
ISR(TOV2_vect){
    // Enable output compare match on B.

    // Transmit next measurement
    transmit_next_level = TRUE;
}
```

Denne kolonnen er forbeholdt sensor

This column is for external examiner

```
ISR(TIM1_COMPB_vect){
    level = TANK_HEIGHT - SPEED_SOUND * measurement
                          /(2 * F_CPU);

    // Disable this interrupt

    // Transmit message
      CAN_msg_t msg = {};
      msg.length = 4;
      msg.float_data = level;

      CAN_transmit(msg);
}
```

NTNU

Kandidat nr./*Candidate no.* 10104

Dato/*Date:* 30.11.2017 Side/*Page:* 11

Emnekode/*Subject* TTK4155

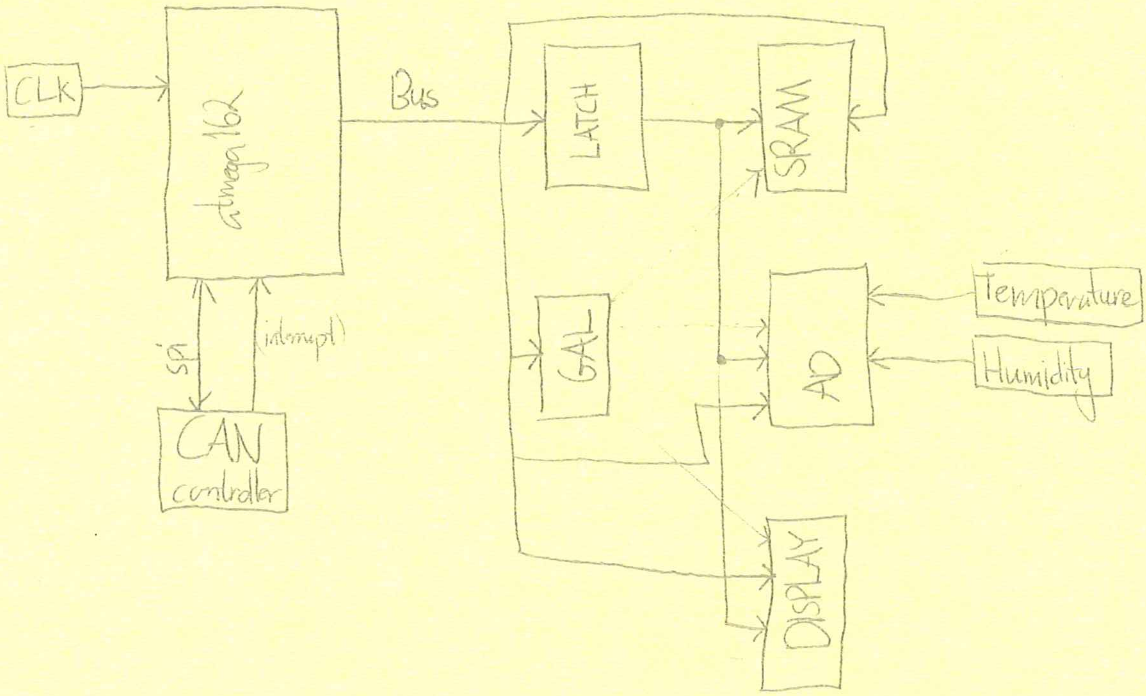
Antall ark/*Number of pages:* 14

Denne kolonnen er forbeholdt sensor

This column is for external examiner

Problem 3

a) Also need an address latch and a chip select device ~~[illegible]~~ (we will use a GAL).



NTNU

Kandidat nr./*Candidate no.* 10104

Dato/*Date:* 30.11.2017 Side/*Page:* 12

Emnekode/*Subject* TTK4155

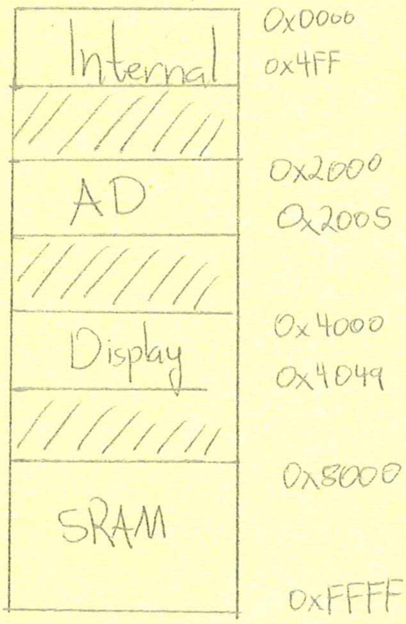
Antall ark/*Number of pages:* 14

Denne kolonnen er forbeholdt sensor

This column is for external examiner

b) Need to make room for AD, display, and SRAM. SRAM will occupy half the address space so we will place it in the upper half. Since display should be put at 0x4000, we can use 0x2000 as address for AD to get the simplest decoding logic.

~~ADD~~ MAP:



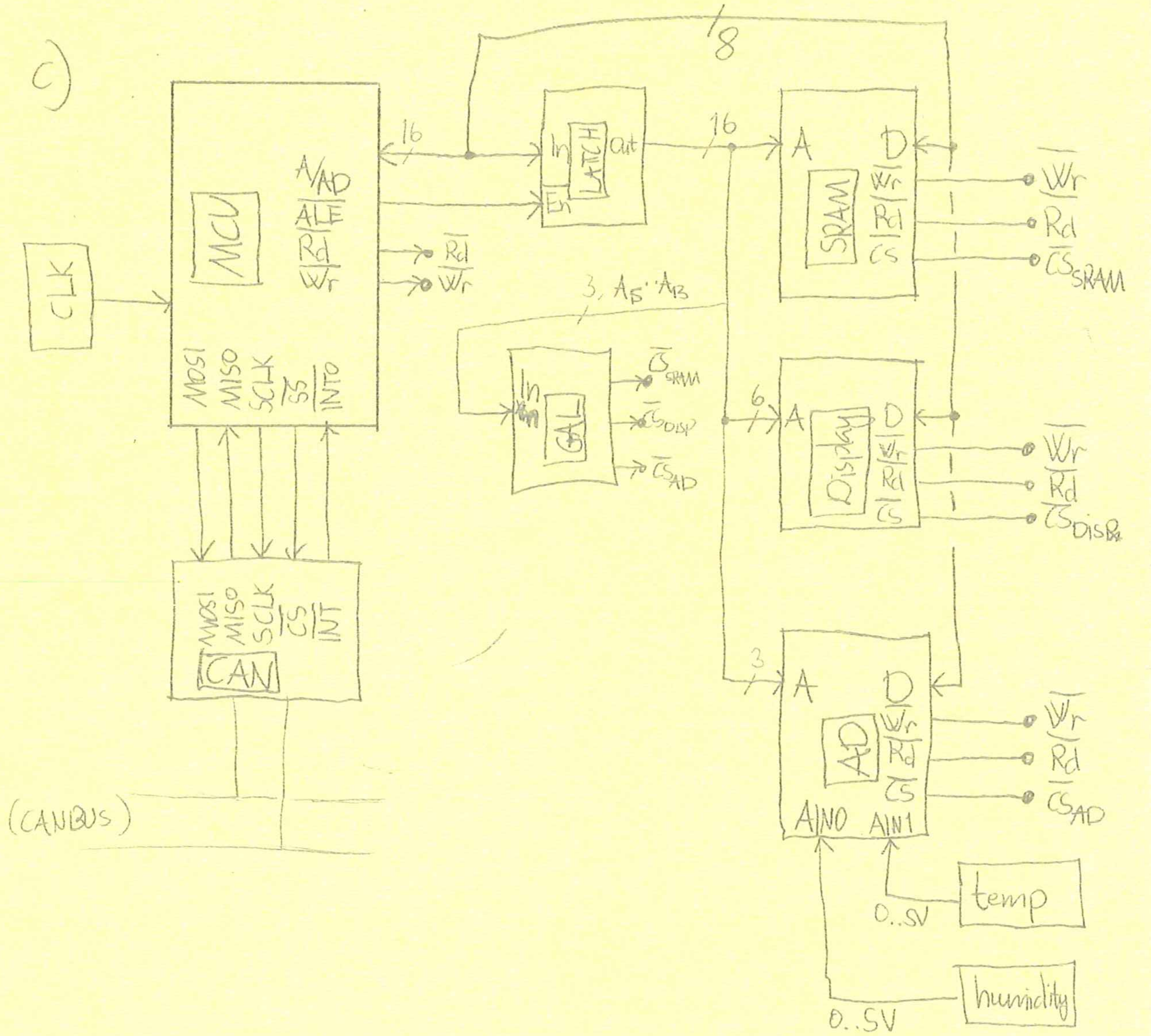
Let A_{15}, A_{14}, A_{13} denote the ~~the~~ three MSBs of the addresses. Then the partial address decoding approach to this will give

$$\overline{CS}_{AD} = A_{15} + A_{14} + \overline{A}_{13}$$

$$\overline{CS}_{DISPLAY} = A_{15} + \overline{A}_{14}$$

$$\overline{CS}_{SRAM} = \overline{A}_{15}$$

c)



Since the address and databus of the atmega162 are multiplexed, I've added an address latch that can "store" the address while the bus is used for data. Since all bus units have read/write capabillities, they all parallell need the read/write signals from the atmega162. The address bus into the AD and Display arent given explicitly, but the display needs atleast 6 and the AD needs atleast 3 to cover their address space.

NTNU

Kandidat nr./*Candidate no.* 10104

Dato/*Date:* 30.11.2017 Side/*Page:* ~~13~~ 14

Emnekode/*Subject* TTK4155 Antall ark/*Number of pages:* 14

Denne kolonnen er forbeholdt sensor
This column is for external examiner

The GAL will implement the decoding logic and thus only need the three lines A_{15}, A_{14} and A_{13}.

Den gule kopien beholder du/*Please keep the yellow page*

Fagtrykk Trondheim AS 12.2015