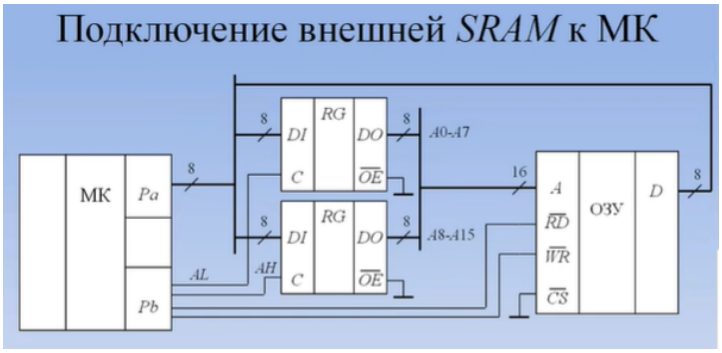
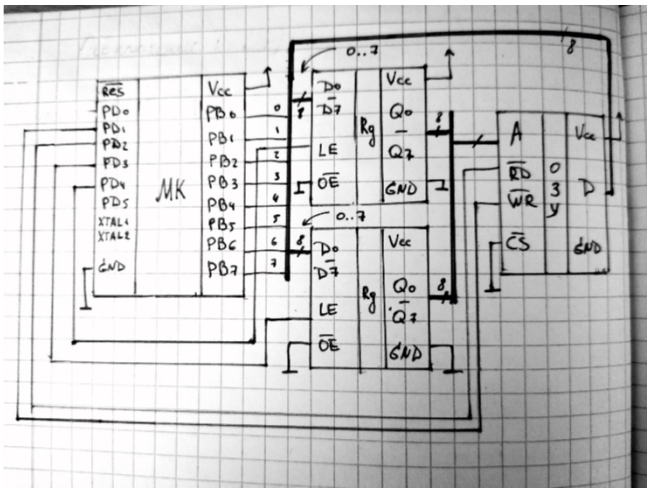
1. ОЗУ (параллельный 10-12 разрядный) к АВР, инициализация и считывание 2 байт, начиная с адреса Ad (обмен байтами).

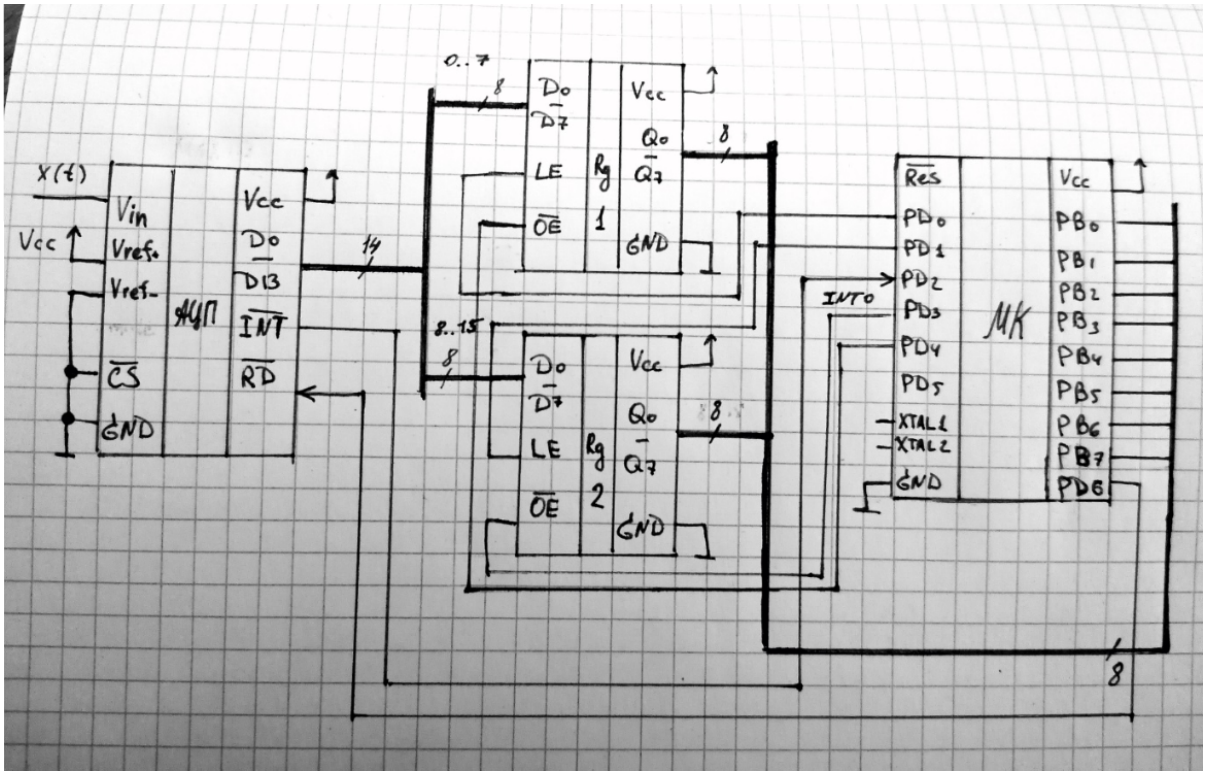




Код

```
.cseg
rjmp main
.org 0x0B

main:
sbi DDRB,0 //меняем направление порта B, PortB - выход
sbi DDRD,0 //меняем направление порта D, PortD - выход

//задаем старшую часть адреса на порт B
ldi R16,0b00000011
out PortB,R16

//формируем строб защелкивания старшей части адреса (Reg_1- сигнал LE)
sbi PortD,4

//задаем младшую часть адреса на порт B
ldi R16,0b11010011
out PortB,R16

//формируем строб защелкивания младшей части адреса (Reg_2- сигнал LE)
sbi PortD,3

//запись в ОЗУ
//выдача данных на порт B
ldi R16,0b11110011
out PortB,R16
//выдача строба WR на линию PD1
sbi PortD,1

//чтение из ОЗУ
cbi DDRB,0 //меняем направление порта B, PortB - вход
//выдача строба RD на линию PD2
sbi PortD,2
//чтение данных из порта B в регистр R17
in R17,PortB
```

2. АЦП (параллельный 12-14 разрядный) к АВР

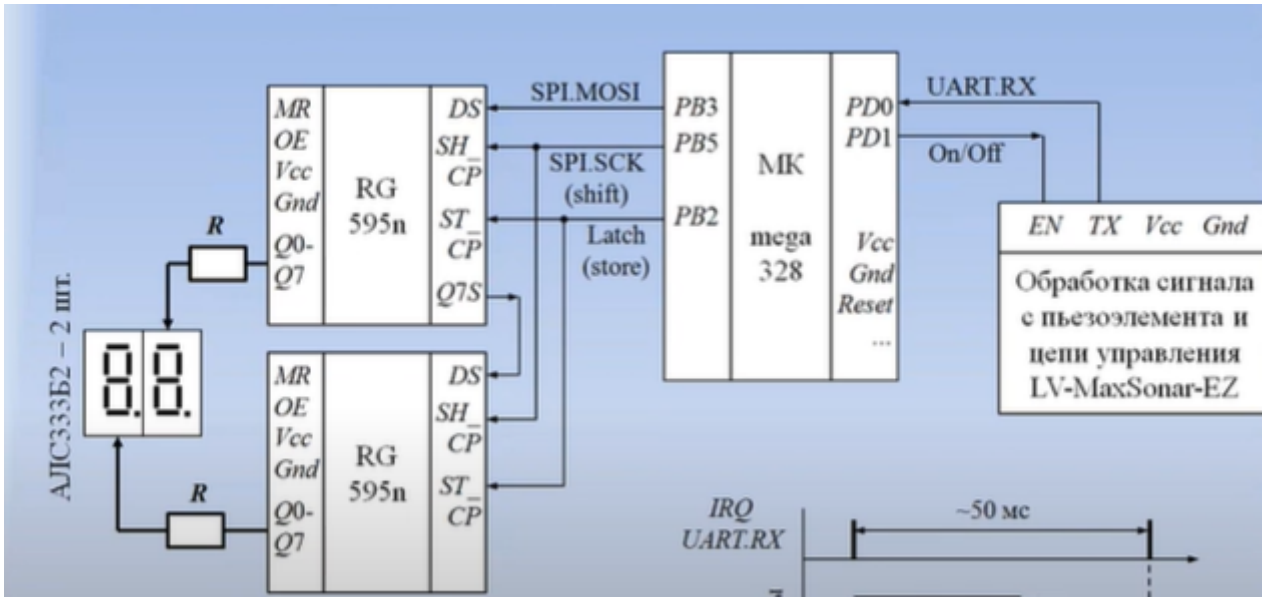


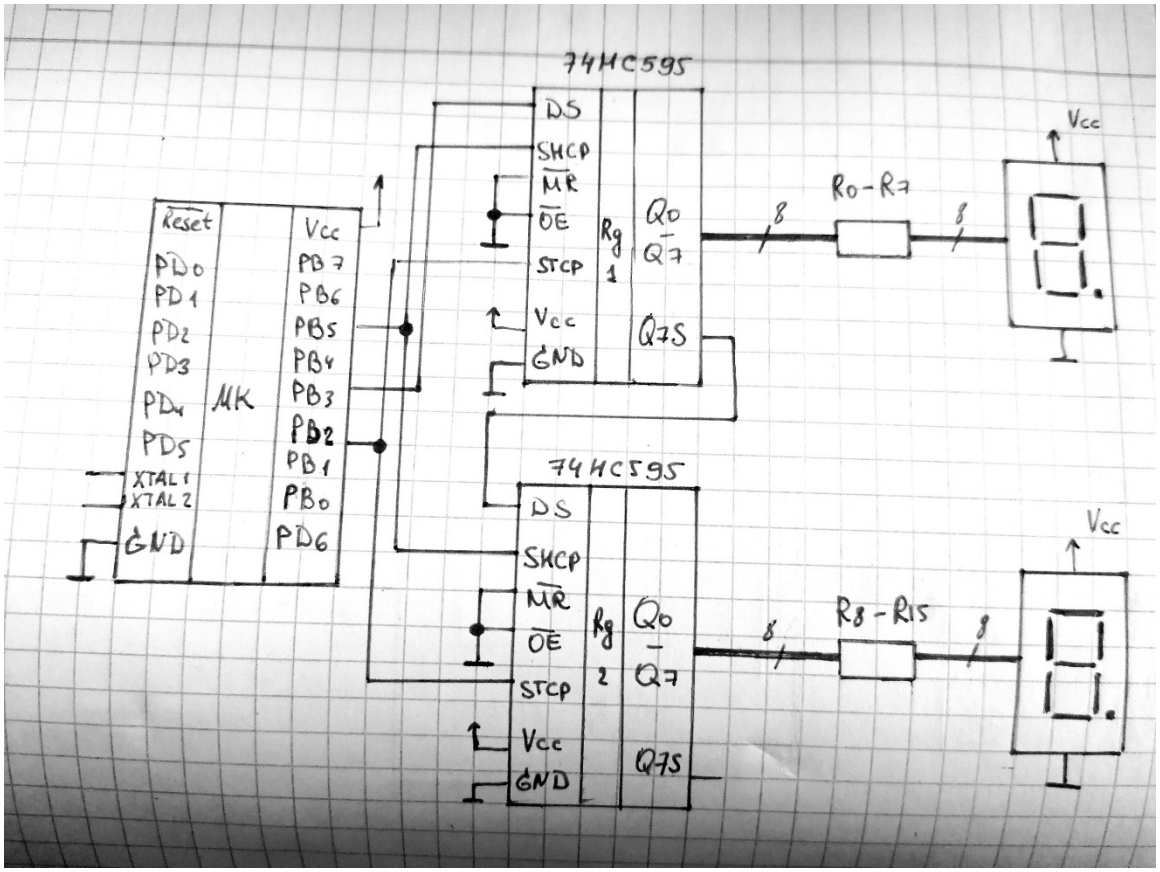
Код

```
.cseg
rjmp main
rjmp INT0_ready
.org 0x0B

INT0_ready:
//формируем строб защелкивания младшей части данных (Reg_1- сигнал LE)
sbi PortD,0
//формируем строб защелкивания старшей части данных (Reg_2- сигнал LE)
sbi PortD,1
//формируем сигнал PD4=0 - запрет работы регистра 2 - выход в третьем состоянии
sbi PortD,4

//читаем данные (младшую часть) АЦП
in R0,PinB
st Y+,R0 //помещаем результат в буфер, сдвигаем указатель

//формируем сигнал PD4=0 - запрет работы регистра 1 - выход в третьем состоянии
sbi PortD,3
//формируем сигнал PD3=0 - разрешение работы регистра 2
cbi PortD,4

//читаем данные (младшую часть) АЦП
in R0,PinB
st Y+,R0 //помещаем результат в буфер, сдвигаем указатель

sbi PortD,6 //посылает сигнал RD на АЦП, подготовка к следующему преобразованию от АЦП

//формируем сигнал PD3=0 - разрешение работы регистра 1
cbi PortD,3

cpi R28,$60+2*16 //проверяем сколько байт приняли
brlo go     // переход, если R28 < $60+2*16
ldi R28,$60 //реинициализация указателя Y на буфер - переполнение
go:
reti
```

```
main:
//настройка стека
ldi R16, $60+127
out SPL,R16

//Y - указатель на принятые с АЦП данные
ldi R28,$60
Ldi R29,0

cbi DDRB,0 //меняем направление порта B, PortB - вход

//меняем направление порта D, PortD.2 - вход, остальные - выход
ldi R16,4
out DDRD,R16

//формируем сигнал PD3=0 - разрешение работы регистра 1
cbi PortD,3
//формируем сигнал PD4=0 - разрешение работы регистра 2
cbi PortD,4

//формируем сигнал PD6=0 - запуск преобразования
cbi PortD,6

//ожидаем готовности данных INT0
loop: rjmp loop
```

5. Два семисегментных индикатора (последовательный) к АВР. Подключение, инициализация

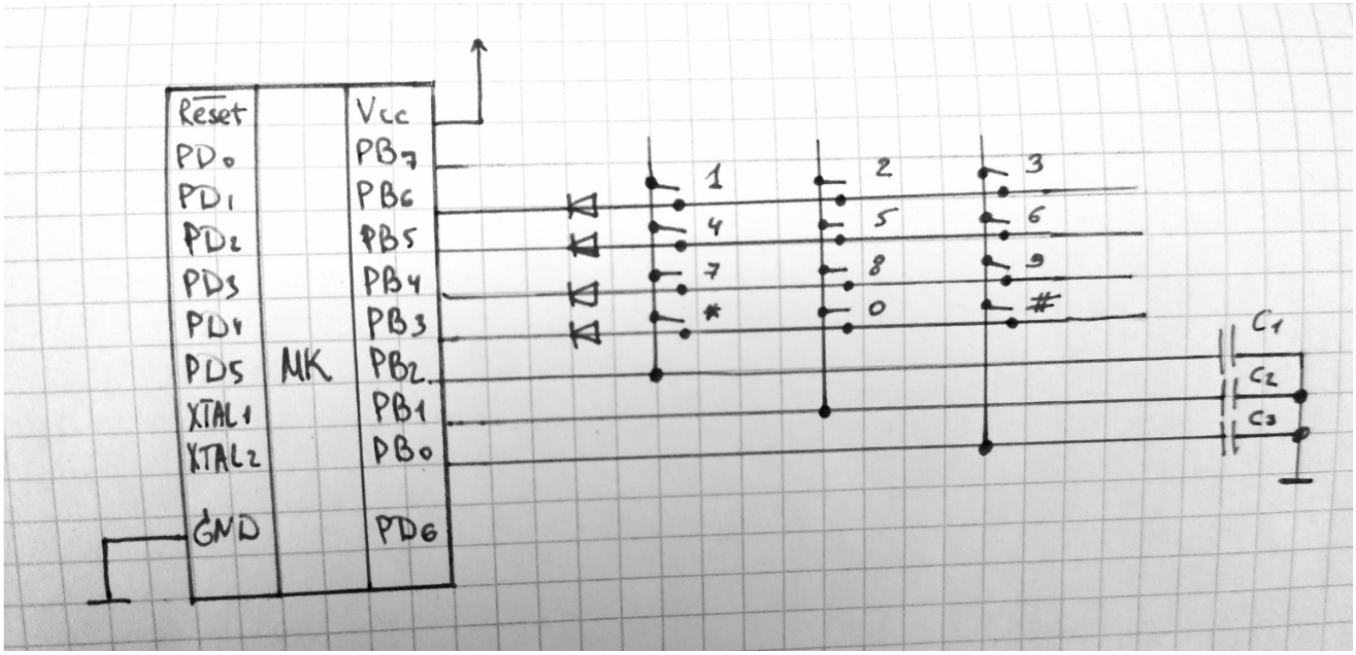




Код

```
.cseg
rjmp main
.org 0x0B

main:
sbi DDRB,0 //меняем направление порта B, PortB - выход

ldi R16, 0b11111100 //символ "0"
ldi R18,0 //для сравнения - все ли биты выдали
ldi R17,8 //количество битов для выдачи


indication:
cpse R17,R18 //сравнение - сдвинули все 8 бит в регистр 1? пропуск след.команды, если R17=0
rjmp no_reinit

//реинициализация ("0" записан в регистре 1, нужно сдвинуть этот символ в регистр 2, а в регистр 1
записать другой "0")
ldi R16, 0b11111100 //символ "0"
ldi R17,8           //количество бит
set //установка временного флага T - флаг выхода из цикла

no_reinit:
asr R16 //сдвиг вправо - флаг C содержит бит, выдвигаемый на линию PortB.3
dec R17 //уменьшаем счетчик бит для выдачи

brcs set_bit //если нужно выдать 1 - переход
cbi PortB,3  //выдаем 0
rjmp next

set_bit:
sbi PortB,3 //выдаем 1

next:
sbi PortB,5 //осуществляем сдвиг (сигнал SHCP)

brtc indication //переход, если T не установлен
cpse R17,R18    //все ли биты выданы в регистр 1 (регистр 2 уже заполнен), пропуск след.команды,
если R17=0
rjmp indication


sbi PortB,2 //осуществляем защелкивание - выдача на индикаторы (сигнал STCP)
```

Опрос клавиатуры



Код

```
.cseg
rjmp main
.org 0x0B

main:
ldi R16,0b10001111
out DDRB,R16 //направление = PB.0-PB.3 - выходы, PB.4-PB.6 - входы

ldi R16,255
out PortB,R16 //подаем 1 на все линии порта B

ldi R16,0b11110111 //начало схемы бегущего нуля, PB.3=0

//цикл беспрерывного опроса клавиатуры
cycle:
out PortB,R16 //запись в порт управляющего слова
nop //задержка, прежде чем читать содержимое порта
in R17,PinB //читаем порт B
nop //задержка, учет дребезга клавиш

neg R17 //активный сигнал - 1, инвертируем, чтобы получить код клавиши

//...анализ клавиши

ori R16,0b11110000
asr R16 //сдвиг вправо
cpi R16,255 //проверка - на все ли строки подали 0 поочередно
brne cycle//переход, если R16<>255
ldi R16,0b11110111//реинициализация для следующей схемы бегущего нуля
rjmp cycle
```