# МПиМПС

Лекция 7.

# Структура программы

Основной цикл

```
int main(void)
{
    while (1)
    {
    }
}
```

Можно добавить прерывания:

```
ISR(USART_RXC_vect)
{
}
```

# Работа с флагами

Флаги - переменные, в зависимости от их состояния выполняется то или иное действие.

```
while (1)
{
    if(flag1){ action1(); }
    if(flag2){ action2(); }
    if(flag3){ action2(); }
}
```

Можно для флагов завести отдельную структуру, использовать отдельные байты или даже биты.

```
struct {
    char flag1;
    char flag2;
    char flag3;
}flags;
```

```
if((flags.flag1 & (1 << 3)) == 1){ ... }
```

# Программный таймер

Настраиваете один из аппаратных таймеров, чтобы он давал прерывания. Внутри - вешаете обработчики для всех функций, которым нужен периодический запуск.

```
int timers[7];

ISR(TIMER0_COMP_vect)
{
    for(i = 0; i < 7; i++){
        if(timers[i] > 0){
            timers[i] -= 1;
        } else {
            flags |= 1<<i;
        }
    }
}

main:
if((flags & (1 << n)) != 0) { doAction(); }

doAction:
flags &= ~(1 << n); ... ; timers[m] = 1000;
```

# Программный таймер

Структуру таймера можно усложнить

```
volatile static struct {
    int number;
    int time;
}timers[7];

ISR(TIMER0_COMP_vect)
{
    for(i = 0; i < 7; i++){
        if(timers[i].number == 255) continue;
        if(timers[i].time > 0){
            timers[i].time -= 1;
        } else {
            flags |= 1<<i;
            timers[i].number = 255; //освободили таймер
        }
    }
}
```

# Ещё более сложная организация кода

Диспетчер задач

Диспетчер с учётом приоритета

Кооперативная многозадачность

Вытесняющая многозадачность