МПиМПС

Лекция 2.

Память микроконтроллера. Начальные адреса.

```
start:
     ldi r16, Low(RAMEND)
     out SPL, r16
     ldi r16, High(RAMEND)
     out SPH, r16
```

При трансляции языка ассемблера в машинный код первая строчка (код операции, соответствующий ей) размещается с нулевого адреса, далее следуют последовательно коды операций следующих команд.

Первые адреса - особенные, они нужны для работы прерываний (когда исполнение программы прерывается, PC переходит на “вектор” (адрес) соответствующего прерывания.

Решение: Освободить первые адреса, оставить там только команду JMP, для каждого прерывания на соответствующий обработчик. Указать значение адреса, с которого размещать коды операций, можно с помощью директивы ассемблера `.org`

Адреса векторов прерываний.

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

```
Addr Lbl  Code Comments
$000      jmp RESET ; Reset Handler
$002      jmp EXT_INT0 ; IRQ0 Handler
$004      jmp EXT_INT1 ; IRQ1 Handler
$006      jmp TIM2_COMP ; Timer2 Compare Handler
$008      jmp TIM2_OVF ; Timer2 Overflow Handler
$00A      jmp TIM1_CAPT ; Timer1 Capture Handler
$00C      jmp TIM1_COMPA ; Timer1 CompareA Handler
$00E      jmp TIM1_COMPB ; Timer1 CompareB Handler
$010      jmp TIM1_OVF ; Timer1 Overflow Handler
$012      jmp TIM0_OVF ; Timer0 Overflow Handler
$014      jmp SPI_STC ; SPI Transfer Complete Handler
$016      jmp USART_RXC ; USART RX Complete Handler
$018      jmp USART_UDRE ; UDR Empty Handler
$01A      jmp USART_TXC ; USART TX Complete Handler
$01C      jmp ADC ; ADC Conversion Complete Handler
$01E      jmp EE_RDY ; EEPROM Ready Handler
$020      jmp ANA_COMP ; Analog Comparator Handler
$022      jmp TWSI ; Two-wire Serial Interface Handler
$024      jmp EXT_INT2 ; IRQ2 Handler
$026      jmp TIM0_COMP ; Timer0 Compare Handler
$028      jmp SPM_RDY ; Store Program Memory Ready Handler
;
$02A RESET: ldi r16,high(RAMEND) ; Main program start
$02B      out SPH,r16 ; Set Stack Pointer to top of RAM
$02C      ldi r16,low(RAMEND)
$02D      out SPL,r16
$02E      sei ; Enable interrupts
$02F      <instr> xxx
```

```
.org 0
jmp start

.org 0x12
jmp TIM0_OVF

.org 0x1C
jmp ADC_handler

.org 100

start:
    ldi r16, Low(RAMEND)
    out SPL, r16
```

Задача подсчета времени

Мигание светодиодом с частотой 1 Гц.

Основной цикл:

```
sbi PORTD,4              ; Clocks: 2
call Delay               ; Clocks: 4
cbi PORTD,4
call Delay               ; rjmp основного цикла, Clocks: 2
```

Функция Delay:

```
Delay:
     ldi r19, 0xFF       ; Clock: 1
     ldi r17, 0xFF       ; Clock: 1
     ldi r18, 0x05       ; Clock: 1

Delay_loop:
     subi r19,1          ; Clock: 1
     sbci r17,0          ; Clock: 1
     sbci r18,0          ; Clock: 1
     brne Delay_loop     ; Clocks: 2, last: 1
     ret                 ; Clocks: 4
```

8 МГц = 8 000 000 Clocks, разделить на два, вычесть всё, что выполняется однократно, разделить на количество тактов в Delay_loop. Ответ: **C 34 FD**

Таймер/счетчик

Периферийное устройство микроконтроллера.

- Отсчет временных интервалов
- Работа в разном временном масштабе
- Подсчет импульсов, приходящих извне
- Работа от внешнего кварца (например, на 32.768 кГц)
- Генерация ШИМ-сигнала
- Установка прерываний, флагов

Работа с таймером/счетчиком

Основные регистры для работы (на примере таймера 0): TCCR0, TCNT0, OCR0, TIMSK, TIFR.

TCCR0: настройка режима работы таймера, режима работы пина Output Compare, настройка предделителя.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCNT0 - регистр таймера
OCR0 - регистр сравнения
TIMSK - настройка прерываний (для всех таймеров сразу)
TIFR - флаги прерываний (для всех таймеров сразу)

Режимы работы таймера

- Нормальный
- Clear on Compare (CTC)
- Fast PWM
- Phase Correct PWM

Figure 31. CTC Mode, Timing Diagram

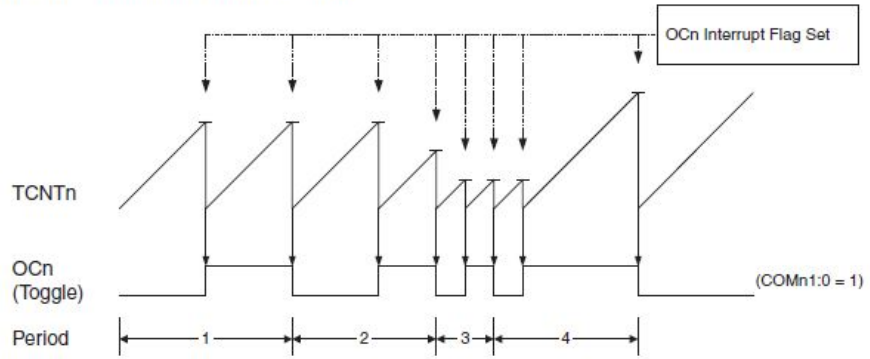


Figure 32. Fast PWM Mode, Timing Diagram

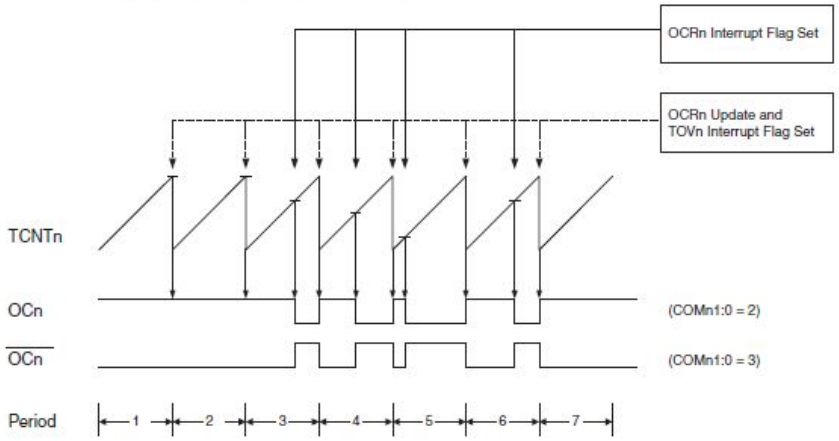
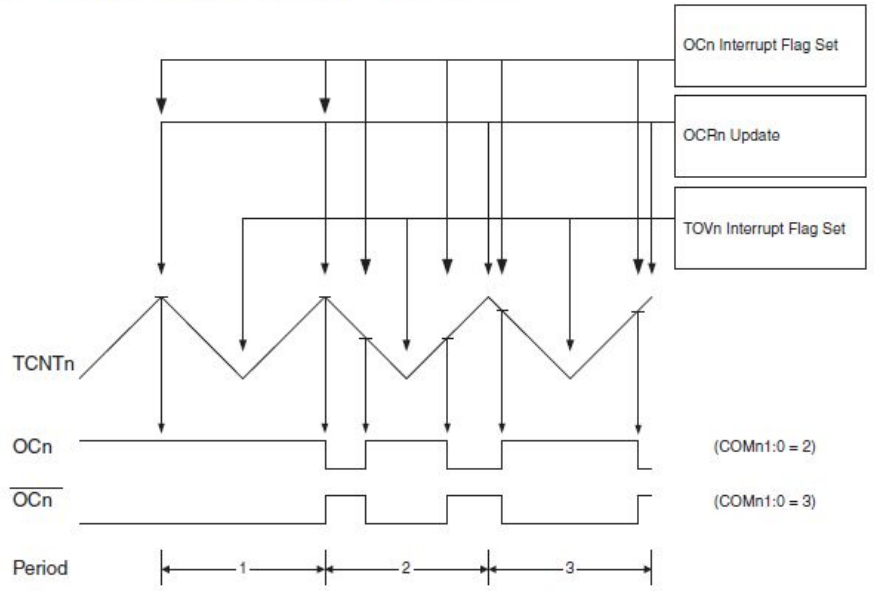


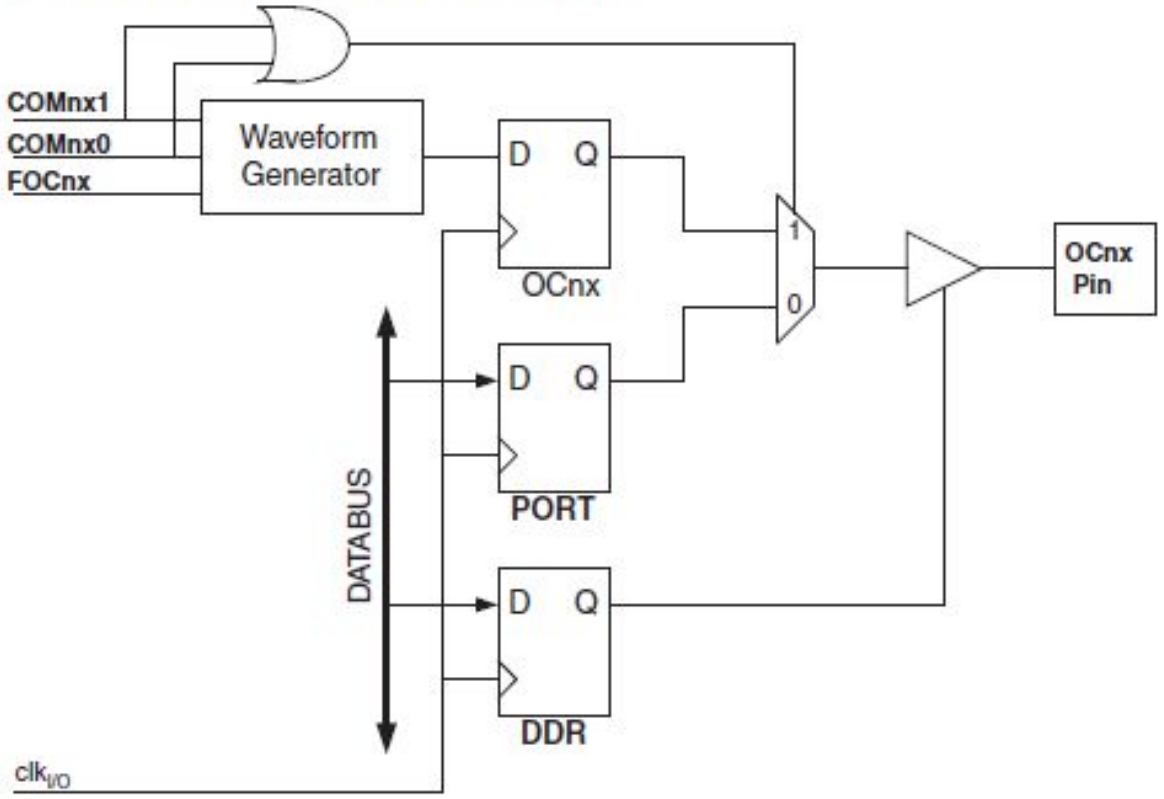
Figure 33. Phase Correct PWM Mode, Timing Diagram



Не забыть про Data Direction!

Режим работы “на вход” или “на выход” все ещё определяется соответствующим пином DDRx.

Figure 44. Compare Match Output Unit, Schematic



Режим счётчика

Можно считать внешние импульсы, поступающие на определенный пин (для таймеров 0 и 1). Настраивается также через TCCR0. Детектирование восходящего или нисходящего фронта.

Table 42. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

ШИМ (PWM)

Управлять яркостью светодиода можно, меняя коэффициент заполнения ШИМ-сигнала. При использовании интегрирующей RC-цепочки с помощью ШИМ-сигнала можно получить аналоговый сигнал.

(демонстрация)

АЦП

Аналого-цифровой преобразователь последовательного приближения.

Средние как по скорости, так и по достижимой точности измерения. Измерение осуществляется за ~25 тактов, точность измерения 10 бит.

Принцип работы: Перебирая разряды от старшего к младшему, формирует на внутреннем ЦАП напряжение, максимально приближенное к входному.

8 входов для АЦП (с режимом мультиплексирования), возможность использования дифференциальных входов.

Прерывание по окончанию измерения.

Как работать с АЦП?

ADMUX - Выбор опорного напряжения и источника входного (измеряемого) напряжения. Там же настраивается “выравнивание” 10-битного результата измерения по левому или правому краю двух 8-битных регистров результата измерения.

ADCSRA - включение, старт измерений, настройка авто-измерений, флаг прерывания, включение прерываний, настройка предделителя.

ADCL, ADCH - результат измерений.

SFIOR - регистр специальных функций (например, старт измерения по переполнению таймера).

Первая задача.

Плавное включение светодиода. Формирование ШИМ-сигнала с использованием таймера.

- Инкремент значения через Delay
- Инкремент через прерывания от второго таймера
- Плавное включение с плавным выключением

Чтение аналогового сигнала с потенциометра, регулирование яркости пропорционально повороту ручки потенциометра.

- автоматические измерения, обновление яркости в основном цикле
- пустой основной цикл, вся работа на прерываниях
- устранение ошибки нуля, измерения с адекватной скоростью